Protecting a Corporate Network from Insider, Outsider and Collaborative Attacks

Dalal Hanna, Bachelor of IT (Honours)

A thesis submitted in total fulfilment for the degree of Doctor of Philosophy

School of Computing, Engineering and Mathematical Sciences

La Trobe University

Victoria, Australia

February 2022

Abstract

Broadcasting is one of the essential features of the Internet Protocol version 4 (IPv4). Two essential IPv4 services, namely, the ***Address Resolution Protocol*** (ARP) and the ***Dynamic Host Configuration Protocol*** (DHCP), use this feature to accomplish their tasks. Other less frequently used IPv4 applications that use IPv4 broadcasting services are the ***BOOTP*** (Bootstrap Protocol) and ***RIPv1*** (The Routing Information Protocol).

The ARP protocol establishes a *dynamic binding* between the hardware address of a *host*, the *media access control* address (MAC address) and the logical IPv4 address of the machine. Without this binding, the IPv4 protocol will not be able to transmit a packet from a source to a destination host. Thus, broadcasting is critical for the operation of the IPv4 protocol and cannot be eliminated.

Broadcasting introduces performance bottleneck in a *Local Area Network* (LAN). However, in normal traffic conditions, this performance degradation is tolerable. Whereas, the attacking community exploits this feature to launch several attacks against an IPv4 network. Almost 90% of the known IPv4 attacks exploit this broadcasting feature.

The attackers can be from anywhere on the Internet. We classify them as ***insiders*** and ***outsiders*** based on their location. Some attacks require collaboration between insiders and outsiders. We call it ***collaborative*** attacks. In Chapter 3, we provided a taxonomy of various high-intensive attacks that are due to IPv4 broadcasting. We also classify attacks that are due to insider, outsider and collaborative in nature. There are two motives for a malicious user to launch an attack against a host or a network: they are for personal and monetary benefits, and disrupting the victim's business model. Our taxonomy also involves classification of attacks based on this factor.

Numerous papers are available in the literature that provides a solution to specific subclasses attacks. However, most of them have inherent limitations. They either assume a priori relationship between a host and the network, or require modification to the standardized protocol stacks. Thus, these schemes are deemed to be unacceptable in a civilian environment, as several organizations promote the ***Bring Your Own Device*** (BYOD) concept.

The core theme of this thesis is to eliminate ***broadcasting*** in an IPv4 network without affecting core IPv4 functionalities. By doing so, we will eliminate several attacks that are due to IPv4 broadcasting.

A LAN consist of switches, routers, access points, wireless controllers, firewalls, servers and end hosts.

In Chapter 6, we created a framework called the ***PrECast*** (Protocol for Eliminating Broadcast) for eliminating IPv4 broadcasts in a LAN. The *PrECast* infrastructure accomplishes this task by converting broadcasting into network-wide multicasting involving only the corporate LAN switches. *PrECast* is transparent to the end-users and does not require any modification to the protocol stacks. *PrECast* does not assume any a priori relation between a host and the network. However, the *PrECast* infrastructure only requires a minimal modification to the corporate LAN switches.

The heart of the *PrECast* protocol is the existence of a multicast tree connecting all corporate LAN switches. In Chapter 5, we created a multicast tree involving all corporate LAN switches. We use the ***Core-based*** multicast tree creation paradigm for building the multicast tree.

An insider may connect a *new* switch, *compromised* switch, or a *decommissioned* switch to the multicast tree and inject fake information to compromise the network. Thus, *admission control*, *key revocation* and the *new key generation* (for new and compromised devices) are the crucial steps in the multicast tree design that cannot be overlooked. Currently, the traditional public-key cryptography such as RSA (Rivest–Shamir–Adleman) or its elliptical curve variant is used for addressing the above-mentioned issues. However, these algorithms suffer from key revocation issues. RSA also requires a *Public key infrastructure* (PKI) to distribute public keys in a network. PKI may be a single point failure and are frequently targeted by attackers. Without a secure communication channel between the PKI and every LAN switch, an attacker may pose as the PKI and distribute their own keys to compromise the system. Thus, in this thesis we introduced a new paradigm called ***pseudo-identity*** based encryption, which is a variant of RSA that can address all the mentioned shortfalls.

The IP protocol does not validate the authenticity of the source IP address in the packet header. This results in *IP address spoofing* attacks. Even though the proposed *PrECast* infrastructure solves several insider attacks, IP spoofing attacks are still possible. An insider may collaborate with an outsider through IP spoofing to launch *Denial of Service* (DoS) attacks. Chapter 7 proposed a *micro-NAT* architecture called the ***NAT++*** to patch these attacks that originate from a corporate network.

With the current Internet technology, a user has the freedom to communicate with any destination host. The destination host may be a *malware spreader*, *command and Control*

Centre, or an *attacker* and so on. IP reputation is a mechanism of awarding credit scores to IP addresses and their associated domain names based on their reputation. There are several IP reputation providers available in the Internet, such as Cisco's Talos. The credit scores are updated frequently by these providers. In Chapter 8, we created a *dynamic filter* based on the *reputation* of the destination IP address. This dynamic IP filter will stop corporate hosts from establishing a connection with destination hosts having lower reputation scores.

In any network, critical hosts such as the DNS servers, Gateway, email, and web servers will have public IP addresses. Since they offer essential services to both inside and outside hosts, their connection from the external network cannot be blocked. Any outsider with a malicious intention can fingerprint these devices.

Fingerprinting is a process of identifying a remote network device, services running on the remote devices, their Operating Systems (OS), security patches applied to hosts and network devices and so on. Attackers fingerprint a network and host to launch various attacks targeted towards network devices and important servers. Fingerprinting cannot be avoided. In Chapter 9, we created a proxy-like service called ***SDMW***. Rather than providing *factual information* about OS and services running on various hosts and network devices, *SDMW* will provide *fictitious information* to the attackers. Thus, defeating any potential attacks.

Chapter 11 deals with the conclusion and future directions.

Dedication

I have a short story to tell, I *loved a man more than anything in this world*, and one day I was terrified at the news of his death last year.

I dedicate this dissertation to the *soul of my beloved father*, ***Marcus Hanna***, who taught me that success only comes with hard work, patience and persistence.

Though you never get to see this, however, I am sure your spirit hovers over me because you left my world but did not leave my heart.

I miss you every day and wish you could be here when I take my hat high to see your beautiful smile that brings me joy and happiness.

I would also like to dedicate this thesis to my supervisor, mentor, critic and friend ***Prakash Veeraraghavan*** whose encouragement and knowledge continues to inspire me to be the best I can be.

Statement of Authorship

Except where reference is made in the text of this thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis accepted for the award of any other degree or diploma. No other person's work has been used without due acknowledgement in the main text of the thesis. This thesis has not been submitted for the award of any degree or diploma in any other tertiary institution.

Dalal Hanna
28 February 2022

Acknowledgements

My journey of a doctoral degree would have been arduous. I am able to achieve the goals with the help, guidance, and encouragement of my kind and supportive supervisors, *Prakash Veeraraghavan* and *Eric Pardede*.

I owe my deepest gratitude to them both for always keeping me on track.

Foremost, I would like to express my sincere gratitude and appreciation to my principal supervisor Prakash Veeraraghavan, for his support and encouragement to pursue my academic goals.

It would not have been possible to achieve this level of education without his constant guidance, personal generosity, and inspiration.

In particular, I wish to thank him from the bottom of my heart for sharing the excitement of learning and for always having placed trust and belief in me from our first meeting.

The inspiration for my research project came from the numerous discussions and high-spirited arguments that we had together over countless cups of coffee.

I am indebted for his guidance, constant faith in my ability to complete this thesis, and the constructive feedback he gave me during my journey have made me grow professionally and personally. It has been an honour working with you, and to have had your expertise through this journey, I can never adequately express my gratitude and appreciation to him.

I am highly indebted to Dr Eric Pardede for his insightful comments, contractive and timely feedback in completing this project, and for always being there for me in both the easy and hard times these past years.

I cannot thank him enough for everything he has done for me. I am very grateful for all his assistance in this adventure. I feel so lucky to have him as a supervisor and a great friend!

My sincere and heartfelt gratitude and appreciation to Professor Wenny Rahayu for all her kindness, guidance and mentoring that she has provided me as the chair of my dissertation committee. I appreciate her professionalism and expertise, from which I have learnt much throughout this adventure.

This work was supported by an Australian Government Research Training Program Scholarship. I thank the Australian Government for providing me the scholarship for conducting this research.

Last but not least, I would like to express my gratitude to my family, cousins and friends who have encouraged me along the way.

Publications from the Thesis

Hanna, D.; Veeraraghavan, P.; Soh, B. *SDMw: Secure Dynamic Middleware for Defeating Port and OS Scanning.* **Future Internet** 2017, 9, 67.
https://doi.org/10.3390/fi9040067

Hanna, D.; Veeraraghavan, P.; Pardede, E. *PrECast: An Efficient Crypto-Free Solution for Broadcast-Based Attacks in IPv4 Networks.* **Electronics** 2018, 7, 65.
https://doi.org/10.3390/electronics7050065

Veeraraghavan, P.; Hanna, D.; Pardede, E. *Building Scalable and Secure Multicast Delivery Infrastructure in a Local Area Network.* **Electronics** 2019, 8, 1162.
https://doi.org/10.3390/electronics8101162

Veeraraghavan, P.; Hanna, D.; Pardede, E. *NAT++: An Efficient Micro-NAT Architecture for Solving IP-Spoofing Attacks in a Corporate Network.* **Electronics** 2020, 9, 1510.
https://doi.org/10.3390/electronics9091510

Veeraraghavan, P.; Hanna, D.; Pardede, E. *Dynamic Filtering to thwart malicious activities in a network through IP Reputation and Policy Engine.* **Under submission**

Contents

Chapter 1

Introduction

In this chapter, we provide a brief introduction to the Internet, its growth, and potential issues. We also provide an overview of the OSI Model (Open Systems Interconnection Model), and the TCP/IP model. The terms and terminologies introduced in this chapter will help a reader in understanding the rest of the thesis.

The Internet is a global system of interconnected networks that use the common Internet protocol suite to communicate between network and devices. It is a *network of networks*. The network may consist of private, public, academic, business, and hobby scopes. These networks are linked by a broad array of diversified technologies. No single organization owns or regulates the Internet [43].

The Internet has become all too pervasive in our modern society. It has dominated our lives as never before. Every day, more bits of data flow across the Internet than grains of sands on all the beaches in the world. According to Cisco [13], the total global Internet traffic for 2011 was approximately 8.4×10^{18} bits per day. This excludes intranet traffic. Meanwhile, the University of Hawaii estimated that the number of grains of sands on all beaches in the world is approximated at 7.5×10^{18} grains [23]. Furthermore, Cisco predicted an annual increase of 32% of the Internet Protocol (IP) traffic. The Internet consists of millions of hosts of various types such as PCs, Laptops, phones, sensors, camera etc. They are connected through networks of various sizes. When it comes to network topology, new domains are added to the Internet almost every day. Thus, the Internet has passed the point where it is hard to visualize, comprehend or control the activities across the networks. There is another interesting statistic on the number of users on the Internet. In the year 2000, there were 413 million Internet users worldwide. However, there are 5,473,055,736 users as of 30JUN2022. The growth during 2020-22 is 1,416%. During the past five years, almost 640,000 new users connect to the Internet every day [44].

Since the Internet connects potentially diversified countries, the cyber laws, and the information regulation in one country will not users from other countries. There is not common code-of-conduct for the Internet. Consequently, protecting the network and privacy of the user data is the foremost important research priority today. A significant challenge for today's network engineers and security administrators is to develop techniques capable of detecting attempts to compromise the integrity and availability of the network.

In network terminology, a protocol is a set of well-established rules determining how data is transmitted between different devices in the same network. The Internet uses the ***Transmission Control Protocol*** and the ***Internet Protocol*** (TCP/IP) suite. Every device that wishes to be a part of the Internet must use these protocols without any modification. These protocols are designed with no security in mind. Both the research community and developers of these protocols did not anticipate the exponential growth of Internet. Thus, these protocols offer minimal protection against eavesdropping.

The original intent for the creation of the Internet is to share information and resources. It has now revolutionized the way we communicate. Today, the Internet is no longer seen as a medium for sharing information but has become part of our day-to-day life. In almost everything we do, we use the Internet: ordering pizza, paying utility bills, making online transactions, chatting with family and friends, social media etc. The Internet had also freed us from the geographic fetters. The Internet has impacted at all levels during COVID-19 lockdown by providing unbounded possibilities for collaboration, working from home, online education and research, and socializing. A wealth of corporate and personal information is stored online and is accessible through the Internet. Breach of privacy and confidentiality of data may cause substantial financial loss and possibly human loss.

Due to rapid proliferation of the Internet, several people, such as elders, children are forced to adapt to technology without proper training and orientation. During the early era of computing, every computer user must be a good programmer and a debugger. They must develop their own program in solving problems. Now, this is not the case. There is no need for a computer user to know about its operating principles or the underlying hardware. This is the same with Internet use. An Internet user need not worry about the operations of the Internet protocol or its potential strength and pitfalls. Even though it is easy to use the Internet to perform day-to-day tasks, most users are not aware of its darker side. Taking advantage of this situation, malicious internet users (often called *attackers* or *actors*) target innocent users. The *financial benefit* is the *primary motivation* for these attacks. Attackers may also modify assessed information, causing data integrity violation. Unauthorized access to private data, a computer, or a network system, is a crime in many jurisdictions and is often accompanied by severe consequences. Since the

Internet connects politically diversified regions, one region's law may not impact attackers from other areas.

The past several years has seen many stories about some countries attacking other countries, and interfering with internal politics. The theft of intellectual property can give a country a significant advantage in international trade. Defending against the fallout from state-sponsored cyber-warfare and cyber espionage will continue to be the priority for cybersecurity professionals.

Protecting Internet users is one of the ever-going challenging problems. Even with the latest technology, both the research and commercial communities were unable to stop attackers targeting innocent individuals (and organizations). It is estimated that in 2020, the financial loss due to cyber-attacks mounts to a whooping 1 Trillion USD [19].

No economist or financial modellers can predict the potential loss in the next five years. Even though this fact is disappointing, it challenges the research community and the industry to provide seamless solutions to initiate and prevent cyber-attacks. Cyber attackers target not only innocent end-users but also large corporate networks. There are several reasons behind this. However, the prime factors were the financial advantage and the corporate information theft. Again, corporate information theft may also be used for attacker's financial advantage or used by their competitors for the market advantage. Attackers sell the stolen information through dark websites.

Network security is essentially a battle between a *Penetrator* who tries to find *holes* and the *Administrator* who tries to *close* them before it is being found. *Security is only as strong as the weakest link.* The attacker often uses the weakness found in the protocol, software and hardware, or all of them to penetrate the network.

Current cyber-attacks and weaknesses in the network systems provided us with a solid motivation to propose various solutions to protect a corporate network. In this thesis, we provided a *holistic*, *simple* and a *scalable* solution to solve several security related problems in a corporate network. The focus of this thesis is not to protect individual internet users. We provided a solution to protect a corporate network (medium and large-scale) and its end-users.

As an introductory chapter, we provide here some of the basic terms and terminologies used throughout the thesis.

The Internet consists of an extensive network spanning across the globe and a suite of protocols and standards that governs the operation and links billions of devices worldwide. The network consists of several million devices like switches, routers and firewalls, and end devices like PCs, printers, mobile phones, sensors etc. A corporate network consists of a few to several hundreds of network hardware such as switches and routers. The network may be protected by a corporate firewall that enforces corporate IT policies. End devices such as PCs, printers, laptops, phones, sensors etc., are connected to this network. They must adhere to the corporate IT policy. The organization usually sets corporate IT policies in consultation with security experts. Some organizations (such as the military) will allow only their devices to be connected to their network, whereas other organizations allow the concept of *bring your own device* (BYOD). Educational institutions are one of the examples that implement BYOD. Even though BYOD offers flexibility, it offers several security challenges. A user may inadvertently connect a malware-infected device with a corporate network. This may spread the malware to other inside hosts. Users with malicious intent may purposefully connect an infected device to the network to spread the infection. Protecting a corporate network from all these users are necessary to secure their intellectual property.

1.1 The OSI Layer

Before we discuss the Internet protocols, we present a brief introduction to the OSI model. The Open Systems Interconnection model (OSI model) is a theoretical model that characterises and summarises a communication process between two end devices. The OSI model partition the complex communication process between two end-devices into seven disjoint abstraction layers. Each intermediate layer offers specific services to the immediate layer above and seeks services from the immediate layer below.

We present a quick introduction to the OSI layered architecture. For more details, readers may refer to Tanenbaum and Wetherall [89]. The OSI representation model is presented in Figure 1.1.

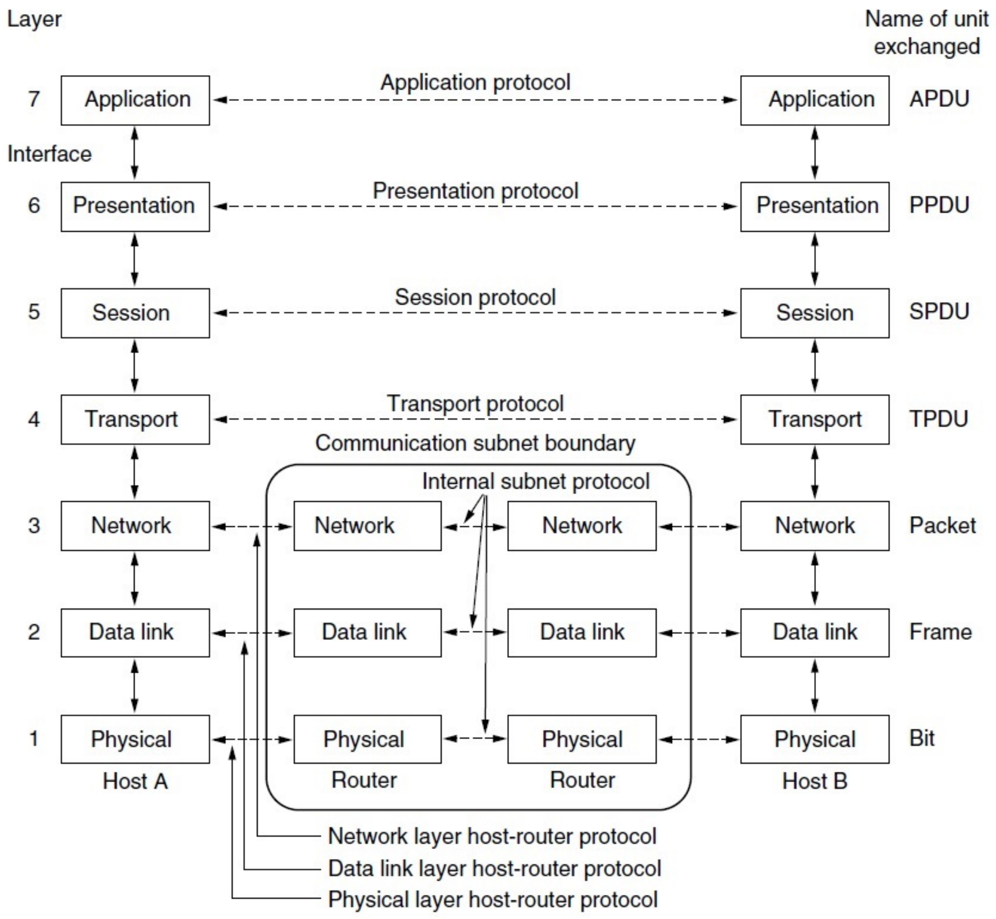


Figure 1.1: The OSI reference model [89]

- **Layer 1 (or) the Physical Layer:** Physical Layer is the lowest in the OSI stack. It is responsible for moving information between two devices connected by a single physical link. This layer deals with transmitting the raw bits (0 and 1) over a communication medium. Thus, this layer is concerned about bit representation, how long a bit last, how to initiate a connection and how to terminate a connection. The physical layer provides the abstraction of bit transport, which is independent of the link technology. This layer is also concerned about standardizing interconnecting hardware.

- **Layer 2 (or) The Datalink Layer:** This is the second layer in the OSI stack. This layer breaks up the data that arrives from the higher layer into small chunks called Data Frames. These frames are transmitted sequentially. If the service is reliable, the receiver confirms the correct receipt of each frame by sending back an acknowledgement frame.

  This layer is concerned with how to keep a fast transmitter from drowning a slow receiver in data. Some traffic regulation mechanism may be needed to let the transmitter know when the receiver can accept more data.

The medium access control (MAC) sublayer is concerned about who will access the medium at a given time.

- **Layer 3 (or) The Network Layer:** The primary function of the network layer is to logically concatenate a set of links to form an abstraction of an end-to-end link. To communicate between two devices, some form of addressing and routing is essential. This layer is concerned about providing a unique network-wide logical address to every device. Routes can be based on static tables that are stored in the network devices or computed dynamically. This layer also includes a simple congestion management scheme. However, this layer does not offer any reliability and provides only a *best-effort* service.

- **Layer 4 (or) the Transport Layer:** This is one of the essential and complex layers of the OSI model. This layer is an actual end-to-end protocol. In other words, a program on the source machine carries on a conversation with a similar program on the destination machine, using the message headers and control messages. Thus, the source and the destination does not need to bother about the hardware and the links used by intermediate devices in the routing process. Unlike the network layer, this layer provides *reliable connection-oriented service.*

  The Transport layer implements a complex (application level) flow control mechanism and congestion management scheme. This layer is responsible for offering reliable communication between a source and a destination device.

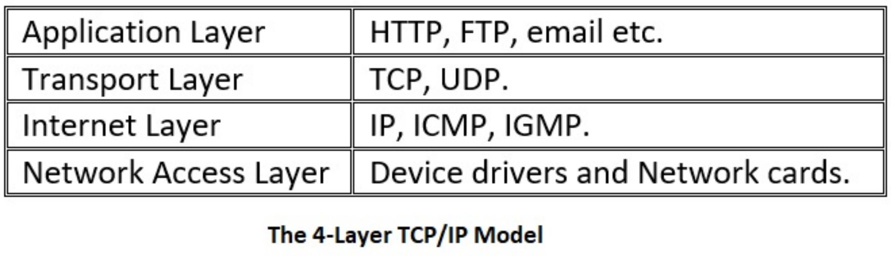
- **Layer 5 (or) The Session Layer:** The session layer allows users on different machines to establish sessions between them. Services such as dialogue control, token management, and synchronization are part of this layer.

- **Layer 6 (or) The Presentation Layer:** This layer is concerned with the syntax and semantics of the information transmitted. Thus, two devices with different internal data representations may be able to communicate with each other.

- **Layer 7 (or) The Application Layer:** The application layer contains various protocols that users commonly need.

There are several advantages due to the OSI model:

1. The OSI model act as a reference model.



Application Layer	HTTP, FTP, email etc.
Transport Layer	TCP, UDP.
Internet Layer	IP, ICMP, IGMP.
Network Access Layer	Device drivers and Network cards.

The 4-Layer TCP/IP Model

Figure 1.2: The 4 Layer TCP/IP model [89]

2. The OSI model act as guidance in developing any networking protocol.
3. Being a layered architecture, changes to one layer will not affect the rest of the model, as long as this layer's services to its higher layer are unchanged.

We now discuss the Internet Protocol Suite.

1.2 The Internet Protocol Suite

The Internet protocol suite does not follow the OSI model. However, there is a similarity between these two models.

The Internet protocol suite consists of two workhorse protocols Viz: ***The Transmission Control Protocol (TCP)*** and the ***Internet protocol (IP)***. These protocols provide end-to-end connectivity specifying how data should be packed, addressed, transmitted, routed, and received at the destination.

TCP/IP is a four-layer system as shown by Figure 1.2

The ***link layer*** is sometimes called the ***network access layer*** that includes the *network interface card* (NIC), their associated drivers for the corresponding operating system, and like devices are interconnected to form a *Local Area Network* (LAN).

The ***Network layer***, generally referred to as the ***Internet layer***, handles the movement of packets around the network. This layer is responsible for concatenating LANs to form a *Wide Area Network* (WAN) and then the Internet. Routing of packets takes place at this layer. Since the data packet needs to be moved as fast as possible, this layer has no concern about the reliability of the packet transmission. The Internet Protocol (IP) operates at this layer. IP is the workhorse protocol of the TCP/IP protocol suite. All TCP, User Datagram Protocol (UDP), Internet Control Message Protocol (ICMP), and Internet Group Management Protocol (IGMP) data gets transmitted as IP datagrams. IP provides an unreliable, connectionless datagram delivery service. By unreliable, we mean there are no guarantees that an IP datagram successfully gets to its destination.

IP provides best-effort service. When something goes wrong, such as a router temporarily running out of buffers, IP has a simple error handling algorithm: throw away the datagram and try to send an ICMP message back to the source. The upper layers must provide any required reliability (e.g., TCP).

The ***Transport layer*** provides a flow of data between two hosts. Two protocols are operating at this layer. They are The Connection-Oriented Transmission Control Protocol (TCP) and The Connectionless User Datagram Protocol (UDP).

The TCP provides a reliable flow of data between two hosts. It is concerned about dividing the application data into reasonable chunks (called segments) that can be handled by the lower layers, acknowledging the received packets and the transmission of missing segments.

The UDP provides a much simpler service to the application layer. It transmits a datagram from one host to another, but there is no guarantee that the destination host will receive the data segments due to unreliable IP protocol. This protocol associates no retransmission mechanism.

The ***Application layer*** handles the details of the particular application. There are several popular TCP/IP applications for example, Telnet, HTTP, FTP, SMTP, SNMP, and DNS.

A mapping between the OSI and the TCP/IP model is given in Figure 1.3.

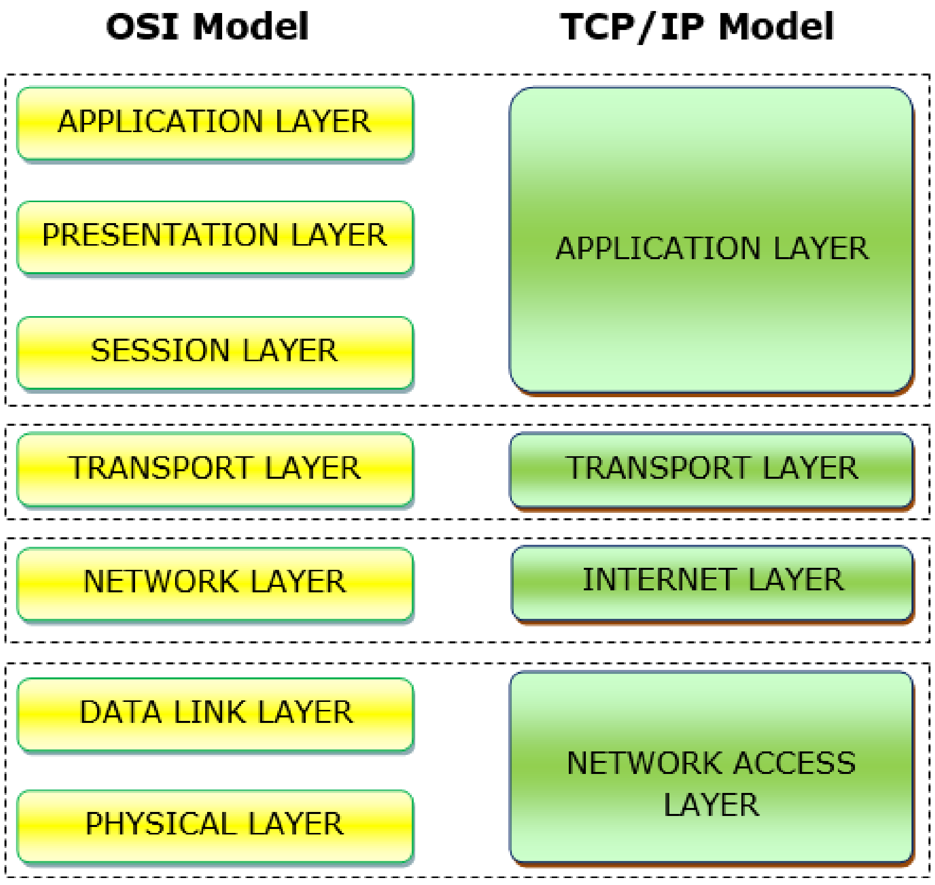


Figure 1.3: Comparison between the TCP/IP and the OSI Reference Models

Even though we mentioned these protocols briefly, a detailed description can be found in Stevens [76]. As necessary, we discuss the protocol frame format and individual fields at the appropriate chapters necessary to understand this thesis better. These are summarized from [76].

1.3 Thesis organization

The thesis is organized as follows: Chapter 1 is introductory, where we present some basic network terminologies.

In Chapter 2, we present essential reasons for a corporate network to be protected from malicious users.

Broadcasting is one of the essential features in the Internet Protocol Version 4 (IPv4). Without this feature, no-host can communicate in an IPv4 network. However, several network attacks are due to this feature. In Chapter 3, we create a taxonomy of attacks that are due to IPv4 broadcasting. This is an important chapter that bring-forth the strength of our contributions.

Chapter 4 deals with the scope and our research contribution. In this chapter, we provided an overview of the following topics: A simple introduction to the problem under study, the scope and the research questions addressed in this thesis.

To provide an efficient solution to several attacks that are due to IPv4 broadcasting, it is essential to create a multicast delivery tree in a corporate network. In Chapter 5, we deal with the creation of the multicast delivery tree. We also address several challenges that arise in a corporate network.

As we mentioned before, broadcasting is one of the essential features of an IPv4 network. If we eliminate the IPv4 broadcasting, several attacks can be stopped. Chapter 6 provided an infrastructure that seamlessly converts broadcast to a multicast communication in an IPv4 network. We call this as the ***PrECast*** framework.

With the introduction of the *PrECast* architecture, several attacks due to IPv4 can be effectively eliminated in an IPv4 network. However, an inside malicious host may spoof the IP address of another legitimate inside host to create few other attacks. These attacks are eliminated through a micro-NATting architecture. This proposal is presented in Chapter 7.

Detecting the heartbeat of any malicious software is an interesting problem. If we can effectively stop any inside host trying to establish a connection with a malicious outside host, several attacks can be thwarted. In Chapter 8, we presented a dynamic IP filtering mechanism based on the destination IP reputation.

Port scanning is a precursor to several network attacks. Attackers scan either the target device or the network for any open ports. They may also fingerprint the host's Operating System and security patches applied before attacking the system. In Chapter 9, we proposed a solution to defeat fingerprinting a network or hosts.

Chapter 10 deals with the consolidation of our research contributions. This chapter contains the following topics: A consolidation of the outcomes of Chapters 5–9 accompanied by any necessary experimentation and discussion of results. We also discuss the strength of our contributions. We provided emphasis on why our approach provides the best solution/ or set of best solutions.

Chapter 11 deals with concluding remarks.

In this thesis, we present the literature survey and the related work in each chapter related to the problem discussed in the chapter. This is to make the thesis readable and engaging.

Chapter 2

Why we need to Secure a Corporate Network?

In this chapter, we provide various reasons for protecting a corporate network. The proliferation of the Internet forces everyone to adapt to technology without proper training. Malicious Internet users take advantage of this situation and target both individuals and companies (irrespective of their size) for their financial advantages. In this chapter, we argue various reasons for protecting a corporate network from malicious users. In this thesis, we have not excluded individual Internet users. They are the end-users of their Internet Service Provider (ISP).

2.1 Malicious Internet Users

2.1.1 A brief history of Hacking

The history of hacking dates back more than 150 years. In 1878, exactly two years after the telephone system was invented, a group of teenagers hired to run the switchboard system were laid-off as they were trying to learn how the system works to make free long-distance calls. There are reports that either individuals or groups target organizations to steal their information for their advantage. Till 1950, most of the computer hacking activities were sponsored by different countries against their enemies. From this period until the late 80s, hackers focused their attention on hacking automated telephone systems to make free phone calls (called *phreaking*).

In the late 1950 and until the late 1970s, computers were much different from present-day desktops or laptops. These were powerful mainframe computers owned by large universities, military and government agencies, and multinational giants. Being time-sharing in nature, computer programmers had to fight for access time. Because of the time and money involved, computer programmers began looking into ways to get the most out of these machines. The program ***geeks*** created what they called ***hack*** that would modify and improve the performance of a computer OS to allow tasks to be completed in a shorter time.

1988 was one of the significant years in hacking history. The ***Morris Worm*** or The ***worm of 2nd November 1988*** was one of the first computer worms distributed through the Internet [61]. It also resulted in the first conviction in the United States under the 1986 Computer Fraud and Misuse law. The worm code was written by a Cornell University student *Robert Tappan Morris* and launched on 2nd November 1988 using MIT's computer. During the cleaning process, the computer must be disconnected from the Internet. US Accounting department estimated a loss of USD 10,000,000.00. The Morris worm has sometimes been referred to as the ***Great Worm***, because of the devastating effect on the Internet at that time, both in overall system downtime and in psychological impact on the perception of security and reliability of the Internet. The Morris Worm prompted the DARPA to fund to create CERT (Computer Emergency and Response Team) for looking into any security issue of the Internet.

1995 was another critical year in Internet history. This year has seen two major attacks that changed the world.

Russian hacker Vladimir Levin is arrested in Britain after allegedly using his laptop to break into Citibank's computer network and transfer funds to various accounts worldwide [94]. He was then extradited to the U.S. He was sentenced to three years in prison and ordered to pay Citibank $240,000. The exact amount of money stolen by Levin remains unknown but estimated to be around $10 million.

This year has seen another famous attack due to Kevin Mitnick hacking into Tsutomu Shimomura's computer [52]. Even after 25 years, Mitnick's hacking approach is used as a ***standard*** way to penetrate any network or machine.

From the timeline of hacking activities until 1950, computer hackings were sponsored. Then there was a shift in the hacker's focus. Hackers attempted to get into the telephone system to make free long-distance calls. Furthermore, during this time, hackers were not cracking the system to take financial advantage. However, after personal computers, including laptops and the Internet gained popularity, hackers started targeting individuals and banks to take their advantage. Even though several patches were offered to the protocol stack, hackers use all possible means to get into a network, including bugs in the host OS, intermediate networking devices, security policy implementations and not but not least, the social engineering approach.

2.1.2 Threat Actors

Threat actors are individuals or groups of individuals responsible for a security incident that can impact individuals and organizations [83]. In this thesis, we call them *malicious users*, *hackers*, or *threat actors*. There are several types of threat actors [83]. They are:

- **Script kiddies:** As the name suggests, these types of attackers lack the necessary skills to impact an individual or an organization. They use the existing tools and scripts written by someone to hack into a network or a computer. Most of these tools are often found through ***dark websites***. Script kiddies have no significant motive. They often hack into a computer or network for fun, revenge against their friends or someone in their social circle or a small financial gain.

  Traditional *firewalls*, *Intrusion Detection System* (IDS) and a robust corporate security policy can stop these types of attacks.

- **Organized crime groups:** The primary purpose of these groups is to steal information, scam people, and make money. The size and the number of this group are increasing every day. They actively recruit intelligent young kids through various motivation. These groups target not only large organizations but also individuals of different social status. In the last three years, the Internet world had witnessed several attacks due to Organized Crime groups in the form of Ransomware [68].

- **State sponsors and governments:** These are individual agents or groups acting on behalf of a country or a state. Their tasks greatly depend on what the state sponsor is interested in. Mainly, they are involved with stealing data, intellectual property of enemy countries, research-and-development data from major manufacturers, government agencies, and defence contractors. State sponsors have several different agendas. Once they achieve their goals, they also sell some of their stolen information through dark websites for their personal benefit.

  Every single day, this world is witnessing state-sponsored attacks originating from different countries. Being sponsored by countries, these countries tend to recruit people with highly technical skills to achieve different tasks. The members of this community employ diversified techniques to penetrate a network or systems. The type of attacks that originate from State sponsors and governments are hard to defend. However, having a more robust cyber incident and response policy can minimize attacks from these groups.

- **Hacktivists:** These people carry out cybersecurity attacks aimed at promoting a social or political cause. Even though they do not harm an individual or an organization, the weaknesses exposed by hacktivists can be used by cybercriminals for their advantage.

- **Terrorist groups:** These groups are motivated by political or religious beliefs. These groups target national infrastructure such as Power Grids and National Irrigation systems to cause national disaster. These groups are not interested in any financial gain. Instead, they target more human loss.

Initially, the term *hacker* was used for a *computer enthusiast*. A hacker is a person who explores the working of a computer system and the network. As they develop a deep understanding of the system, they also know about the system's weaknesses. However, they may not have any malicious intent. Over time, the term hacker is used for individuals who exploit a system with malicious intention.

This thesis provides various proposals to protect a corporate network, mainly from organized crime groups. Our proposal also minimizes the impact of hacking from other hacking groups.

2.2 Vulnerabilities, Exploits and Kits

Vulnerabilities are a weakness in the system design, implementation, software, or code. A corporate network (or any network) consists of specialized network hardware such as switches, routers, wireless controllers, and Wireless Access Points. Users connect to the network using their devices such as computer system, mobile hosts, and phones. Each of these devices is managed by an Operating System. An operating system (OS) is system a software that manages computer hardware, software resources and provides common services for computer programs. User applications run on top of the OS. User applications are called by several names such as Application Programs (or) apps, utility software and application software. They are computer programs designed to perform certain specific activities. The operating system acts as an intermediary between Application Programs and the computer hardware.

A typical OS consists of several million lines of codes written by many different coders. A complex application program may also consist of several thousand lines of codes. Thus, they may have design faults.

2.2.1 Operating system and Application Vulnerabilities

Both OS and applications are vulnerable to various exploits by cybercriminals [65]. When we mention OS, we mean the OS is running on every network device, including the end-hosts. The vulnerabilities may result from lousy coding and bug in the software system due to flawed algorithms. Computing consists of just a set of programs running on processors, including instructions embedded in chips, firmware, device drivers, etc. Thus, in one form or the other, the protection of a program is at the heart of security in computing. Program flaws include everything from a misunderstanding of program requirements, coding error, compatibility between different programs and hardware. The system developers may also intentionally leave a backdoor for debugging purposes (called ***trapdoors***). The attackers may exploit this to gain entry into the system.

2.2.1.1 Trapdoors

The system developers may also intentionally leave a backdoor for debugging purposes called trapdoors. They are not due to a faulty design. A trapdoor is a secret entry point to a program that allows anyone to access the system through usual security access procedures. Whenever a system is locked either intentionally or unintentionally, trapdoors may be used to unlock the system. However, trapdoors must be documented; access to them must be controlled through a robust access control policy, and they should be used with a complete understanding of the potential consequences.

2.2.1.2 Privilege escalation

The OS provides different access privilege to different end-users. For example, the ***administrator*** got the highest privilege in a civilian network, and the ***guests*** got the lowest privilege. Privilege escalation can be defined as an attack involving gaining illicit access to elevated rights or privileges beyond what is intended or entitled for a user. An attacker exploits a bug, design flaw or wrong system configuration to launch this attack.

The ***buffer overflow*** is a classical method used by the hacker community for privilege escalation.

Buffer overflow, or buffer overrun, is an anomaly where a program while writing data to a buffer, overruns the buffer's boundary and overwrites adjacent memory locations. Writing outside the bounds of a block of allocated memory can corrupt data, crash the program, or cause the execution of malicious code. Buffer overflows are not easy to discover. Nevertheless, attackers have managed to identify buffer overflows in a staggering array of products and components. In classic buffer overflow exploits, the attacker sends data to a program, which it stores in an undersized stack buffer. Then the stack is overwritten, including the function's return pointer. Whenever the function returns, it transfers control to a malicious code contained in the attacker's data. The malicious code provides an escalated privilege to the attacker, usually from a lower privilege to

administrator privilege. Whenever an attacker gets an administrator privilege, they will install a backdoor through which they can enter the system bypassing the system security procedures.

2.2.2 Virus, Worms, Trojans and keyloggers

Computer programs by themselves are seldom security threats. However, users with malicious intent can make a program to access data and other programs that they are not authorised to. Malicious code or a rouge program is the general name of any program that has an intention to damage user data or disturb the normal operation of the computer. There are several types of malicious codes available today. A well-known malicious code is a ***Virus*** [65].

A virus is a malicious program segment that cannot run by itself. Instead, it needs a host program to run. Whenever a virus attaches to a host, it is said to have infected the host program. The infected program acts like a virus, and in turn, the newly infected program will start infecting non-malicious codes.

A virus can be either transient or resident. A transient virus runs when its attached program executes and terminates when the attached program terminates. A transient virus infects other programs and causes destruction while the infected program is executed. A resident virus usually loads along with the OS and resides in the computer memory in an active state. Since these malicious codes are resident in the memory, they may be able to sniff user's sensitive information like passwords, pins, or other sensitive information and transmit to remote hosts. They can also perform Denial of Service (DoS) attacks by corrupting the system files.

A ***Trojan Horse*** is a piece of malicious code that has a second, non-obvious malicious effect in addition to its primary effect. These malicious codes are gaining popularity due to several software offered for free, including program cracks and keygens. They advertently gather user's keystrokes and other sensitive information and transmit it to a remote host conveniently.

A ***worm*** is a program that spreads copies of itself through a network. The destructive purpose of a virus and worms are different. A virus will destroy user-files, filesystem, boot sectors and if possible, firmware to paralyse the system. In contrast, a worm multiplies itself to occupy as many systems and network resources as possible to launch a Denial-of-Service attack. In addition, unlike a virus that needed a host program to infect a machine, worms are standalone. Instead, they needed network services to replicate themselves.

Keylogging or ***keystroke logging*** is an act of recording a user's keystroke covertly so that the person using the keyboard is unaware that their actions are being monitored. They may also be used for legitimate purpose like human-computer interaction. Keyloggers are also used in a company for troubleshooting purpose; families use keyloggers to monitor their kid's activities over the Internet legally. However, a malicious user may use keyloggers to steal passwords and login/password covertly. Keyloggers are of two different types: they are either hardware-based or software-based. Even though hardware-based keyloggers are used for a legitimate purpose, malicious attackers may use this for wrong intention. They deploy this type of hardware in public computing facilities such as in library, airport etc. Software-based keyloggers are computer programs used to record key strokes for troubleshooting purposes. These programs are used by attackers to record login, password and other user sensitive information. Software-based keyloggers must be installed covertly in a target machine before an attacker use them.

2.2.3 Protocol Vulnerabilities

As with software codes, protocols are implemented through software that contains several thousand lines of codes. Thus, they may have some implementation faults. As we mentioned in Chapter 1, TCP/IP was designed with no security in mind. The Internet protocol suite assumes that no internet user had any malicious intent. Due to this false assumption, there are several vulnerabilities in the protocol specification. We presented a detailed survey of various vulnerabilities in our earlier paper [37].

Since protocol vulnerabilities are the topic of this thesis, we discuss them in-depth in various chapters. Thus, we skip their discussion here.

2.2.4 Social Engineering

Social engineering is a broader term used for malicious activities that involve human interaction. It uses psychological manipulation to trick users into making security mistakes or giving away sensitive information. The social engineering approach is different from the attacks we discussed before. The attacks we discussed before were launched through exploiting system flaw, incorrect specification, coding error or hidden doors. Social engineering attacks are launched through people (or users) who lacks a good understanding of the corporate IT policies and potential implications. What makes social engineering especially dangerous is that it relies on human error rather than vulnerabilities in software and operating systems. Mistakes made by legitimate users are much less predictable, making them harder to identify and thwart than a malware-based intrusion.

The social engineering approach requires a longer time to penetrate a network or a system than other types of attacks. The attackers need excellent interpersonal skills apart from technical skills to launch this attack. Social engineering attacks come in many different forms and can be performed anywhere where human interaction is involved.

More details on Social Engineering Attacks are presented in [85].

2.2.5 Weak corporate policies

Corporate IP policies play a significant role in securing their network. Most of the time, this exercise is overlooked. Unless each employee uses the IT infrastructure properly, it is not possible to secure corporate infrastructure. It takes only one employee to open malware to launch a ***ransomware attack*** or a ***data breach***. Thus, everything an organization does to stay secure, from implementing technological defences to physical barriers, relies on people using them correctly. This fact cannot be overlooked due to the cost involved. Most organizations make some trivial axioms and create their IT policies. End users are not provided with enough training. Everyone in the organization must understand their role and their contribution towards securing the infrastructure.

Social engineers often exploit weak corporate policies to launch attacks. One can always notice a positive correlation between weak IT policies and social engineering attacks. More details on the impact of a good corporate information security policies are presented in [45].

2.2.6 The Origin of Malicious users

The origin of a malicious user is critical in defending various types of attacks. Usually, every network infrastructure is protected by some form of firewalls. To add an extra layer of security, ***Network Address Translation*** (NAT) may also be used to hide the inside network from the outside world. Thus, certain types of attacks are not possible from an external network.

In practice, no security mechanism (other than access control) is used within a network. Thus, if malicious attackers are within a corporate network, they can cause substantial damages.

An ideal situation will be a collaborative approach, wherein a malicious insider collaborates with a malicious external attacker to launch various attacks. Based on their location, we can classify them as:

- Insider,
- Outsider and
- Collaborators.

In this thesis, we formulate our research theme based on the location of the attackers.

Chapter 3

Attacks that are due to IP Broadcasting

In this chapter, we discuss various network attacks that are due to the broadcasting nature of the IPv4 and the underlying Ethernet protocols. By grouping these attacks, we create a taxonomical order. Providing holistic solution to all these attacks is the main motivation of this thesis. As mentioned before, several authors provided solution in solving one or few of these attacks. However, we provided a scalable solution that can solve all these attacks.

The prime motive behind Chapter 3 is to construct a taxonomy of attacks that are due to IPv4 broadcasting. Several *insider*, *outsider* and *collaborative* network-based attacks are due to IPv4 broadcasting. Based on the extensive literature survey presented in this thesis, none of the solutions available in the literature attempt to provide a solution in totality. In this chapter, we established the fact that the root cause of several attacks is due the broadcast nature of the IPv4 protocol.

The work presented in this chapter is an unpublished work of the candidate and will be published soon.

3.1 The Broadcast Vector

Broadcasting is one of the essential features of any network protocol. It is applicable whenever a source host (in a LAN) has no path to the destination host or the host running a particular service is unknown to the source. In this case, the source host will broadcast a data packet to every active host in the network. All other active hosts, except the intended recipient, will drop this packet. Intended recipients will send a ***unicast reply*** to the source host. Under a healthy network environment, this is not seen as an ***issue***. However, the efficiency of any broadcasting protocol depends on how quickly a host can conclude whether the recipient is the intended recipient of the message. By and large, broadcasting is confined to only Layer 2-LAN technologies such as Ethernet and Token ring. Broadcasting is not supported in a WAN environment due to its performance issues. Ethernet use all-ones in their destination address to represent a broadcast packet. Token ring uses a special value in the control frame to indicate a broadcast frame.

Broadcasting introduces several systems and network performance issues and consumes the network bandwidth. Every active host's Network interface card must interrupt the CPU to process a received broadcast packet. This will affect the CPU performance, especially when a host needs to process several broadcast packets within a short burst of time. The same applies to switches as well. Most often, a host does not benefit from processing the broadcast packet. This is because the host is not the destination being sought, it doesn't care about the service that is being advertised, or it already knows about the service.

Ethernet is the de facto Layer-2 LAN protocol for the 21st century. This protocol is inherently designed as a *broadcast protocol* implemented on a shared medium (called the *bus*). The bus networks are extended to form a *tree topology* using Ethernet hubs. Ethernet uses the *best-effort* delivery mechanism and requires *zero-configuration* at the client-side. The original Ethernet communication follows a *half-duplex* transmission. However, the modern-day Ethernet got less resemblance to the original Ethernet. This is due to *switches* replacing hubs. We discuss more about this in this chapter under the section on "MAC Flooding". Switched Ethernet provides *full-duplex* communication, and *collision of frames* are the history of the past in a switched environment. The new generation Ethernet is no longer seen as a *broadcast protocol.* However, broadcasting is a feature still available whenever a Layer-3 protocol requires. *Ethernet: The Definitive Guide* [80] provides comprehensive detail on the Ethernet protocol. Readers may be referred to this book for more detail on the Ethernet protocol.

3.2 The Attack Taxonomy

There are two motives for a malicious user to launch an attack against a host or a network:

1. Personal and monetary benefits.
2. Disrupting the victim's business model.

In some cases, the attackers use a combination of both. The attacker might aim to directly profit from their perceived ability to disrupt the victim's services by demanding payment to avoid the disruption. Recent ***Ransomware attacks*** fall into this category.

Cyber-attacks can be broadly classified based on the motive of the attacks. Attacks can also be classified based on the location of the attacker. As we discussed in Chapter 2, attacks can be classified as:

- **Insider attack:** Where the attackers mostly come from a corporate LAN. The main motive behind this type of attack is for monetary gain and personal advantage.
- **Outside attack:** Here, the attackers are from an external network. Disrupting the victim's business model is the primary motive for this of attackers.
- **Collaborative attacks:** In this classification, attackers collaborate with insiders to present a more powerful and destructive attack.

We now discuss several classes of attacks that are due to IPv4 broadcasting.

3.2.1 Vulnerabilities in the IPv4 network

3.2.1.1 A Brief Overview

The Internet uses the Transmission Control Protocol and the Internet Protocol (TCP/IP) suite of protocols designed with no security in mind. Both the research community and developers of this protocol did not anticipate the exponential growth of the Internet. The earlier design of this protocol assumes that all Internet users use the network only for a legitimate purpose. This assumption is no longer valid.

The TCP/IP protocol was designed in the 1970s and has undergone several changes. However, it still has several design ambiguities. These protocols are described in their respective ***Request For Comments*** (RFCs) (791 and 793 for IP and TCP, respectively [69], [70]) for developers to read and understand when implementing. However, there are areas where the RFCs leave certain decisions to the developer. The protocol specification did not specify how specific modules need to be implemented, especially when dealing with initial parameters and specific flow control algorithms.

Malicious attackers take advantage of specification shortfalls and ambiguities to launch several attacks for financial gain. Apart from specification shortfalls and ambiguities, certain operational features are implemented in a specific way. This may pose some design faults. One such design fault in the IPv4 protocol is broadcasting. In this chapter, we discuss several possible network attacks due to the broadcast nature of the IPv4 protocol.

As we mentioned before, broadcasting introduces several systems and network performance issues and consumes the network bandwidth. Thus, attackers use broadcasting as the best vehicle for launching several attacks, including DoS and DDoS attacks. Broadcast transmission is also used in scenarios where a source host is looking for a specific service, and the host is not aware who is providing this service in the network. In the absence of a destination-based authentication, any malicious node can pretend to be the destination node and launch a *Man-in-the-middle* (MITM) attack.

IPv4 provides three types of communication between hosts. They are ***unicast***, ***multicast*** and ***broadcast*** communications. Unicast communication happens between one transmitting host and one receiving host. Multicast communication happens between either one source and multiple receivers or communication between many sources and many receivers. Multicasting is regarded as ***group communication***. The ***group*** must be established before the communication. A broadcast communication happens between one sender and every active host in the network. There is no pre-existing relationship between the transmitting host and other hosts in the network in broadcast communication.

IPv4 provides three separate address spaces for these communications. The IPv4 address is a 32-bit number that uniquely identifies a machine's network interface (host). This 32-bit address is divided into two parts, namely the ***network part*** and the ***host part***. The network part specifies the unique number assigned to a particular network—the host part of the IPv4 address that is assigned to each host. The host part uniquely identifies a machine on a network. For hosts that belong to the same subnet, the network part of the address is the same, but the host part must be different.

IPv4 addresses are also divided into various classes. In ***Class A*** IP space, the first 8 bits are allocated for the network part, and the last 24 bits are allocated to the host part. The first 16 bits are allocated to the network part in ***Class B*** address space, and ***Class C*** address space got 24 bits for the network part. In a ***class D*** network, all the 32 bits are allocated for the network part.

The ***Class D*** IP address space is reserved for multicast group communication. The host portion consists of all ***zero*** bits that are not used for a host. The host of ***all zero bits*** is assigned to denote a subnet. IPv4 provides a unique address for subnet-wide broadcast. The host of ***all one bit*** denotes a broadcast address. Whenever a host of all-one bit is used as a destination address for a particular subnet, the IP packet is delivered to all active hosts in the network. When the IPv4 broadcast packet is encapsulated in the Ethernet frame, the destination MAC address is the broadcast MAC address ***FF-FF-FF-FF-FF-FF*** in hexadecimal (or 48 1s in binary).

The readers are referred to Stevens [76] for more information on IP addresses and their classes.

3.2.1.2 Attacks on a LAN

Although attacks originating inside a LAN are not new, they have mainly been ignored by the network administrators. There are three common reasons for this.

1. Most of the time, network administrators prioritise external attacks and do not understand the risks involved in attacks originating from the inside network.
2. Since the attacks originate from inside the network, administrators are either confident with their end-users or a proper threat control mechanism is expensive (money, configuration, and ongoing maintenance).
3. Risks from the insiders are presumed to be negligible.

The potential impact arising from insider attacks can be devastating if unmitigated. Thus, providing a seamless solution for insider attacks is one of our motivations.

Broadcasting is one of the essential features in the Internet Protocol Ver 4 (IPv4). IPv4 broadcasting assumes that the underlying Datalink protocol supports efficient broadcasting. The IPv4 protocol will not function without this broadcasting feature. Network services such as the ***Address Resolution Protocol*** (ARP), The ***Dynamic Host Configuration Protocol*** (DHCP) and the ***Bootstrap Protocol*** (BOOTP) use the underlying broadcast feature to accomplish their tasks.

If the context is clear, we call the IPv4 address simply as an IP address.

Network services like ARP, DHCP and BOOTP are effectively exploited by insiders to launch MITM attacks. Monetary and personal benefits are the main motive behind these attacks.

We now discuss ARP based attacks.

3.2.1.3 ARP based attacks

We briefly describe the operation of the ARP protocol to understand its weaknesses.

The ARP operation

The LAN protocols operate on the lower two layers of the OSI stack, namely the *physical layer* and the *data link* layer. Unlike the TCP and the IP stack implemented through software stacks, the LAN protocols are implemented through hardware. They form a part of ***Network Interface Controllers*** (NIC). Each LAN host has a unique 48-bit address called ***Media Access Control*** (MAC) address embedded into the firmware of its NIC. Within the LAN, nodes communicate through the use of this address.

The IP protocol catenates multiple LANs to establish the Internet. Let a host A wish to communicate with another host B on the Internet. As a first step, the IP stack installed on A must decide whether the destination host resides on the same LAN as that of A or not. This is by comparing the destination host's *network address* with A's *own network address*. If both the hosts are in the same LAN, the IP layer must seek the help of the LAN protocol to forward the IP packet to the destination host. If the destination host is not in the same LAN, the IP protocol must forward the packet to the default gateway in the same LAN with the help of the LAN protocol. In either case, we need the help of the LAN protocol to establish end-to-end communication.

To communicate between two IP hosts in a LAN environment, the source host must be aware of the destination host's MAC address. Since there is no correlation between a host's IP address and its MAC address, a dynamic mechanism must establish this mapping in a LAN environment. *The ARP protocol provides this dynamic mapping.*

Since the ARP protocol poses a severe security pitfall in a LAN, we briefly describe their operation here for completeness in this thesis. Stevens [76] provides a good reference for ARP protocols.

The ARP is a *request-response* protocol whose messages are encapsulated by a link-layer protocol. ARP messages are communicated within the boundaries of a single local area network and are never routed across internetworking nodes.

The ARP protocol provides two essential functions:

1. Establish a dynamic mapping between an IP address and its associated MAC address. We use the term *dynamic* here because the ARP protocol automatically adapts to changes in the IP binding over time without any reconfiguration.

2. Each end-host maintain a table containing this dynamic mapping, called the ***ARP table***. The main purpose of creating this ARP table is to ensure that it is readily available whenever a mapping is needed. This table is also called the ***ARP cache table***. To create dynamism, ***aging*** is used. The aging is also referred to as ***ARP cache timer***. This timer value may vary between different OS.

The ARP consists of two types of operations, namely the ***request*** and the ***reply***. The *ARP requests* are broadcasted to every host in a LAN, and an *ARP reply* is unicasted.

Whenever host A requires the MAC address of the destination host B, whose IP address is $A.B.C.D$, A will issue an ARP request. The ARP request contains the following information.

Host A's IP address, MAC address and Host B's IP address. The destination address used for this request is $FF:FF:FF:FF:FF:FF$, which is a LAN wide broadcast. Every active LAN host will receive this broadcast, irrespective of whether they are using an IP stack or not. The destination host B got enough information to unicast its reply to A. If there is no LAN host with the given IP address, then there will be no reply.

Every LAN host receiving this broadcast message will add A's IP address and the associated MAC address to its ARP cache table. This entry may be used whenever is necessary. Host A will add B's MAC and IP binding in its *ARP-cache table*.

There is another particular type of ARP, called the ***gratuitous ARP*** (G-ARP). Whenever Host A boots or obtains its IP address from a DHCP server, A needs to ensure that its IP address is unique in the network. Thus, it will broadcast the *ARP request* for its own MAC address. No host will reply if A's IP address is unique in the network. If any host replies to this *ARP-request*, A's IP stack will issue a ***duplicate IP*** message and abort binding with the given IP address.

We discuss various vulnerabilities due to ARP's simple operation.

ARP Vulnerabilities

Authentication provides a means of verifying a user, a program or what someone claims. Traditional ARP protocol work through *mutual trust*. It assumes that all its users are *trustable*. Inside attackers exploit this weakness to their advantage. There are several research works available in the literature that proposes some form an authentication. We review them in Chapter 6. However, authentication defeats the transparent operation of the ARP protocol and adds unnecessary overheads. It can also be seen that attackers may target the authentication mechanism and launch a DoS attack.

ARP Reply Vulnerability:

Let a host A issue an ARP request for another destination host B in the network. When A receives the first ARP-reply, the input module of A will set the state flag to RESOLVED. The hardware address received during this process is then cached. For every subsequent frame from A to B will use this resolved hardware address. Any subsequent *ARP-reply* will be “discarded”. There is no mechanism in the ARP protocol to verify the authenticity of the received message. Thus, if an attacker rushes to transmit an ARP reply, he may attract all B's traffic towards himself. He may either drop or inspect the packets for any valuable information. He may also forward the packet to the legitimate host B and thus become a ***man-in-the-middle*** (MITM). An attacker does not need any sophisticated tool or possess the unique skill to launch the MITM attack. Since these tools can be downloaded from the Internet, any novice user can launch this attack.

The ARP request vulnerability:

Whenever a host A issues an *ARP-request*, every active host in the LAN knows A's IP address and the MAC address. Thus, every host will enter this information or update the previously entered information for the MAC and IP binding for A. Since A's *ARP-requests* is not authenticated, any malicious host can issue a falsified *ARP-request*, thus poisoning the ARP cache table of every LAN host. As before, this attack is easy to launch and needs no specific skills.

Unsolicited ARP reply:

A malicious host may broadcast an *unsolicited ARP-reply* with a spoofed IP address and its own MAC address. As with ARP-request vulnerability, this mechanism will poison the ARP-cache table of every LAN host.

Modern-day Operating Systems (OS) implements a TCP/IP stack that drops any unsolicited ARP reply from any LAN host (even if they are for any legitimate purpose). This will defeat an attacker using this strategy to poison ARP cache tables. However, the other two vulnerabilities still exist. These attacks are widely known as ***ARP poisoning*** or ***ARP spoofing***. By performing ARP-spoofing attacks, the attacker will become a MITM. He can then attract the victim's Layer-3 traffic towards him. ARP-spoofing may allow an attacker to intercept data frames on a network, modify the traffic or stop all the traffic. Thus, an attacker can breach the confidentiality and integrity of messages. This attack can only be used on networks that use ARP and is thus limited to local area network segments.

The ARP-poison attack is a precursor to a plethora of LAN attacks in a network. In ARP-poisoning attacks, the victims are usually the default gateway for the network, DHCP servers, proxy servers or any other vital hosts in the network. Through ARP-poisoning, an attacker can launch the following attacks:

- Gateway and proxy hijacking
- SSL-intercept attack
- DoS attack
- DNS hijacking
- DHCP hijacking

If the ARP-poison victim is the default-gateway or the network's proxy server, the attacker can attract all IP traffic of the entire network towards him. He can then look for all valuable information in the network. According to the Calyptix report [15], the SSL-intercept attack constituted 11% of the overall Internet attacks in 2016. If an attacker can divert all the IP traffic towards him, he can then create an SSL-intercept to obtain all sensitive data from the network users.

Through ARP-poisoning, the attacker can launch a DoS attack through the following ways:

- By dropping all packets.
- The attacker can issue a random TCP-RST (TCP-Reset [76]) command to an already existing TCP connection.
- Forwarding packets to the wrong destination.

DNS hijacking is a growing threat [46]. Domain Name System (DNS) hijacking intercepts legitimate DNS requests from the user and matches them to compromised IP addresses hosted on the attacker's servers. To perform DNS hijacking, an attacker launches a MITM attack, which subverts the users' DNS requests and directs them to their own compromised DNS server. The attacker's compromised DNS server uses a DNS switching Trojan to attach the wrong IP address to the user's DNS request, directing him to a spoofed website. This attack is popularly known as pharming and could be employed by scammers in a phishing campaign aimed at stealing personal information. Interestingly, legitimate ISP providers also engage in ill-feting activity to suit their selfish interests, including placing ads or collecting statistics for big data analysis [46].

By hijacking a DHCP server, an attacker can create a rogue DHCP server. A rogue DHCP server is a DHCP server on a network that is not under the administrative control of the network staff. Through rogue DHCP address leasing, the attacker can offer a compromised gateway address. He/she can also direct all DNS queries to an external compromised DNS server or one set by the attacker. The primary purpose is to launch a MITM attack and direct corporate traffic to an external network.

The proxy and gateway hijackings have a similar effect compared with DNS and DHCP hijackings. The attacker can also target individual hosts.

The G-ARP vulnerabilities:

Every host will issue a G-ARP message before binding its IP address, whether its address is statically assigned or obtained dynamically through a DHCP server. It is expected that no host will issue an *ARP-reply* for a G-ARP message. If some host issue a reply, the issuing host will not bind the current IP address. This is because the IP address to be bind is already in use. This mechanism can be seen as a potential weakness. Due to a lack of ARP authentication, any malicious host can issue a reply to a G-ARP message. This process will inhibit any new host from joining the network. This attack falls under a class of DoS attacks.

MAC Flooding:

This attack is not an attack against the ARP protocol. Nevertheless, use ARP as a tool in launching this attack. The Ethernet is the de facto LAN protocol. Ethernet is broadcast in nature. Every active host in a LAN will constantly listen for an incoming frame. Every frame transmitted using the Ethernet protocol is broadcasted to every active host in the network. Only the intended receiving host will start accepting the frame. All other hosts will stop listening to the incoming frame.

However, a malicious host can listen to every frame in the network. This is called a ***promiscuous listen***. This may be regarded as a ***design fault***. The original Ethernet was designed for a ***bus topology***. Network hardware such as ***hubs*** seamlessly implement this topology without using cumbersome ***taps***. Hubs are regarded as ***signal repeaters***. They repeat the signal received from the incoming port to every hub port in a ***synchronous*** fashion. Thus, a collision of two or more frames is possible. Collisions are handled through an ***exponential backoff algorithm***.

Switches replace hubs in the current LAN environments. Unlike hubs, switches do not broadcast a frame to everyone. They are regarded as ***store-and-forward*** devices. A switch will not synchronously broadcast to every port when a frame is received from a port, unlike a hub. Instead, the frame is stored. The switch will check the frame for any possible errors. If there are any errors, the received frame is dropped. There is no retransmission mechanism built into the Ethernet or the IP protocol. The higher-layer protocols must handle this. If the received frame is error-free, the switch will then check the destination address. If the destination address is a broadcast address, then the frame is broadcasted through every port. If the destination address is a unicast Ethernet address, the switch will forward the frame towards a port the destination is connected to.

The success of the above operation depends on the fact that the switch needs to be aware of the MAC address of every host connected through every of its port. This mechanism is called ***transparent bridging***. A switch using transparent bridging technology learns the physical addresses of the hosts connected to its port by examining the source address of frames when it receives through a port. The port number and the received MAC address is entered in a high-speed memory called the ***Content-Addressable Memory*** (CAM). For example, when a switch receives a frame on one of its ports from host A, it records the physical port and the MAC address of host A so that subsequent frames to that host are sent directly to the recorded port. If the destination address is not found in the CAM table, the bridge floods the frame to all ports except to the port that the frame arrived on. By going through the source address learning process, transparent bridges build the address table that is used in all subsequent communication.

A switch will broadcast a frame through all its ports only under the following two conditions:

- The received frame contains broadcast address as its destination address.
- The destination MAC address is a unicast address and is not available in the CAM table.

In a transparent bridging environment, promiscuous listen will not yield any benefit once the CAM table is fully built. The promiscuous listener will only receive frames addressed to them or everyone (broadcast in nature).

MAC flooding is a mechanism of stopping a switch in constructing a CAM table, thus benefiting a promiscuous listener. The CAM table kept in a switch got limited memory. Once this table is full, no new entries can be added. Thus, the frames addressed to these MAC addresses are switched to every port. This will temporarily be causing the switch to function as a hub. The attackers flood the switch with bogus ARP packets. This will fill up the CAM table quickly.

Port Stealing:

This attack exploits a switch's CAM creation algorithm using forged ARP replies. In port stealing, the attacker crafts ARP replies using the victim's MAC address as the source and its own MAC address as the destination. As soon as the switch port receives this ARP reply, the CAM algorithm will associate the attacker's port to the victim's MAC address. Any traffic towards the victim will now be switched to the attacker's port. To reset the CAM table to its original, the attacker will issue an ARP request looking for the victim's MAC address. Once the victim replies, the CAM is reset.

3.2.1.4 The DHCP Protocol

The ***Dynamic Host Configuration Protocol*** (DHCP) is a network protocol used to dynamically assign every end-host with an IP address and other communication parameters such as the *Default Gateway*, *Domain Name System* (DNS) information and so on. This protocol eliminates the need for configuring each host individually.

The DHCP uses a *client-server* architecture, and this protocol design follows the BOOTP protocol. The difference is that DHCP can dynamically allocate an IP address from a pre-configured pool of addresses and can reclaim an address whenever it is no longer needed (such as a workstation is switched off). Apart from allocating and reclaiming addresses, DHCP can also be configured to deliver platform-specific communication parameters to end hosts.

Like the ARP protocol, DHCP messages are not routed across a LAN. However, DHCP Relay Agent can be configured in a LAN to forward DHCP messages across different LANs. The main advantage of this scheme is to have one DHCP server configured in a network serving multiple LANs.

Whenever a client needed an IP address in a DHCP environment, it will broadcast a *request*. Like the ARP protocol, DHCP has several vulnerabilities due to the broadcast nature of the protocol.

The DHCP vulnerabilities

We now discuss some of the DHCP vulnerabilities. The DHCP protocol is used by the network administrators to dynamically assign an IP address and communication parameters to an end host. Before we discuss their vulnerabilities, we will present a brief overview of the operation of the DHCP protocol. More details are presented in Stevens [76].

The DHCP follows a client-server model. It employs a connectionless *User Datagram Protocol* (UDP) to implement its service. The DHCP protocol employs a *four-stage* process in communicating the IP address and the required communication parameters. The acronym that describes this process is *DORA*. During the ***Discovery process***, a client issues a LAN-wide broadcast message looking for a DHCP server and request an IP address. Whenever a DHCP server receives this message, it reserves an IP address available from a pool of pre-configured IP addresses. The reserved IP address is then communicated to the client through the ***DHCP-Offer*** process. A client may receive more than one DHCP-offer message. However, a client can accept only one DHCP-offer.

A client can confirm a DHCP-Offer message through a ***DHCP-Request*** message. Before confirming an IP address, the client will produce a gratuitous ARP to find any other host present in the network with the same IP address. If there is no reply by other hosts, then there is no host with the same IP configuration in the network and the message is broadcast to the server showing the acceptance of the IP address. This is again a broadcast message. When other DHCP servers receive this message, they withdraw any offers that they have made to the client and return the offered IP address to the pool of available addresses.

During the last phase, the DHCP-server transmits a ***DHCP-Acknowledgement*** message to the client.

We now discuss its vulnerabilities:

As with the ARP process, there is no authentication built into the protocol. In the absence of an authentication mechanism, the following are the critical vulnerabilities:

1. **Unauthorized DHCP Servers:** Any client may pretend to be a DHCP server and offer different communication parameters to hijack connections towards them. Since the DHCP-Discover message is a broadcast message, there is no control on who is supposed to receive this message or not. A client cannot verify the authenticity of the DHCP-offer messages.

2. **Bogus DHCP-clients:** A malicious host can issue several DHCP-requests and consume all the IP available addresses in the DHCP pool. Once the address space is exhausted, no new host can join the network. This is a form of DoS attack.

3. **G-ARP vulnerability:** DHCP-request phase is crucial for the client. Before this phase, the client will ensure that the offered address is not in use. For this purpose, the client will issue a G-ARP message. As before, any malicious host can issue a reply. This will inhibit the DHCP-client from accepting the offer. This is another form of a DoS attack.

To launch the above three attacks, there is no particular tools or skills are necessary. Most of the required tools required for launching DHCP attacks can be downloaded from the Internet.

3.2.1.5 The BOOTP protocol

Diskless nodes use ***the BOOTP protocol*** defined in RFC 951 [72] to obtain their IPv4 address and boot images from a remote server. BOOTP is defined as a replacement for the ***Reverse Address Resolution Protocol*** (RARP) defined in RFC 903 [71]. Since the diskless stations are getting obsolete, the use of this protocol is infrequent. BOOTP is replaced with ***Preboot eXecution Environment*** (PXE) protocol. PXE environment describes a simple client-server environment that boots a software assembly, retrieved from a network, on a PXE-enabled client Network Interface Card (NIC). PXE uses a small set of Extensible Firmware Interface industry-standard network protocols such as DHCP and TFTP to boot a system and install Operating System (OS) from a remote server located in the Local Area Network.

All the weaknesses we discussed for the ARP and DHCP protocols apply to BOOTP as well. An attacker becomes a MITM and installs a compromised OS to the requested machine.

3.2.1.6 Attacks from a WAN

In this subsection, we discuss various attacks that originate from an external network. Monetary benefits are not the primary motivation for launching these types of attacks. Victim's service disruption is the primary motive for launching these types of attacks.

Apart from the BOOTP, the ARP and the DHCP protocols, native IPv4 services utilize broadcasting services. Apart from unicast and multicast addresses, IPv4 provides a unique address that any host can reach all the nodes in its subnet. A 32-bit IPv4 address got two parts in its binary form, namely the network and host parts. The host part of the IPv4 address ranges from all zeros to all ones. The host portion of an IPv4 address that consists of a string of all ones specifies its broadcast address. This feature is not a design fault. However, attackers consider this as one of the weaknesses in the IPv4 protocol. The IP protocol (v4 as well v6) does not authenticate the use of a source address.

Host from any IPv4 network can spoof an IP address that belongs to any other IP subnet. Combining these two weaknesses, several IPv4 attacks are possible in a network. The Internet world continues to witness several attacks exploiting these two weaknesses. We review some of the most used attacks that cost millions of dollars loss worldwide.

IP Denial of Service Attack (DoS)

To launch a DoS attack, we need a malicious host who is anywhere in the network and a target victim. A Distributed DoS (DDoS) is a variation of the DoS attack performed by several malicious (or compromised) hosts towards a single target victim.

Figure 3.1 depicts the DoS and the DDoS attacks from an attacker to a victim.

Ping of Death:

Ping is an ICMP application that checks the end-to-end connectivity between two devices. By default, the ping requesting host sends a small IP packet (of size 64 bytes) towards a target host. The target host then transmits a ping-reply. The ping application has the option of sending a larger IP payload. The ping of death takes advantage of this and sends data packets above the maximum limit (65,536 bytes) that IP allows. This requires the payload to be fragmented at the source and reassembled at the destination host (the victim). Sending thousands of packets within a short interval of time to the victim will chock its resources, resulting in the victim crashing or not offering its intended service (known as DoS).

To execute the *ping* command, an attacker must know the address of the victim. Depending on how the attacker knows the address of the victim, we can classify it into

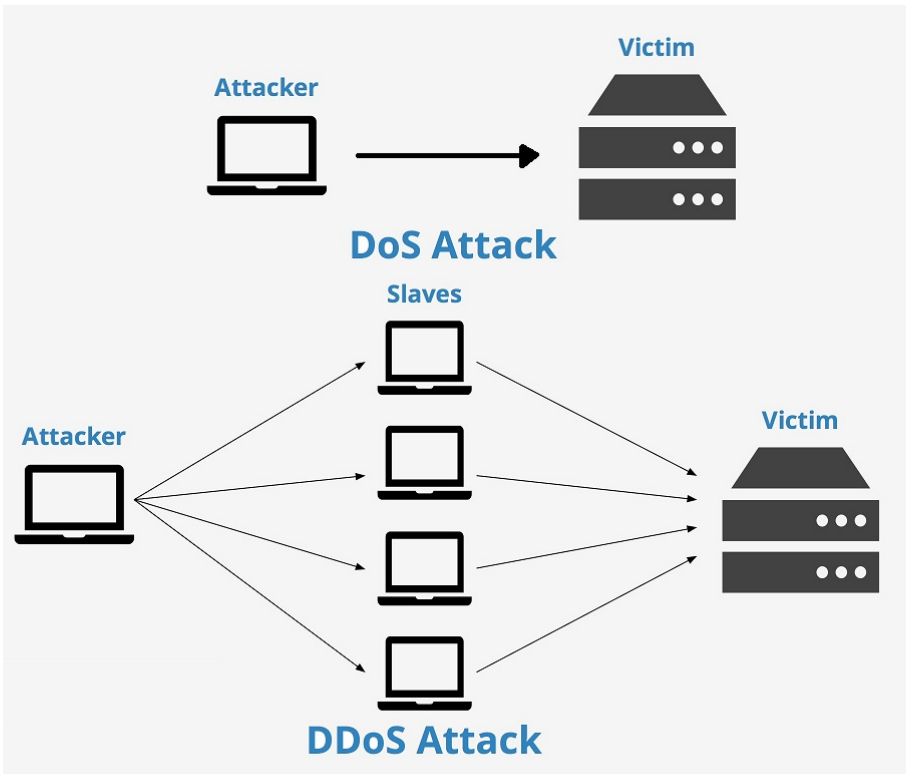


Figure 3.1: The DoS and DDoS attacks

three categories:

- **Insider or collaborative:** If the target victim is an end-host, the attacker must physically access the victim's PC to find its address. In this case, the attacker may be a part of the victim's network or collaborate with someone inside the victim's network.
- **Router or a server:** The victim is a gateway router or a server whose IP address can be resolved through DNS servers.
- **Blind Ping flood:** This method involves using other programs (or methods) to find the victim's address before an attack is launched.

Sending thousands of packets within a short interval of time is crucial in launching a DoS attack. To perform this, the attackers exploit IP spoofing and the broadcast address. To send thousands of packets within a short timeframe, the attackers use the following mechanism. The attacker will spoof the victim's IP address and send an ICMP ping request a large payload, using a broadcast address as its destination. This ICMP request will be sent to every active host in the broadcast address's subnet. Every host will reply with the same payload size towards the victim. Thus, one ICMP request with a large payload using the victim's spoofed IP address sent to a broadcast address will result in hundreds of ICMP replies arriving at the victim in a shorter time. To amplify this attack, the attacker may use multiple compromised machines to replicate this mechanism. To launch this type of attack, an attacker does not require any special skills or tools. Since any novice attacker can launch this attack, this is one of the most dominating attacks continued to be witnessed by the internet community.

We now present several other attacks that are based on similar principles.

The Smurf Attack:

The Smurf IP DoS attack [50] is a classic example of a DoS attack that exploits the IP broadcast mechanism using ICMP-echo messages. Smurf attack uses an ICMP-echo request with a large payload towards the victim. There are three parties involved in this attack: the attacker, the intermediary, and the victim. The intermediary is not a single host, rather all active hosts in a specific IP subnet. One of the criteria for being an intermediary network is having higher upstream bandwidth than the victim's downstream traffic. It also must have many active hosts in the network.

To launch this attack, the attacker first needs to create a forged ICMP-echo-request packet with the victim's IP address as the source address and the IP broadcast address of the intermediary network as the destination address. Every active host will receive this echo-request in the intermediary network. They, in turn, send an ICMP-reply to the victim's machine. In this case, a single forged ICMP-echo-request from the attacker will result in hundreds of ICMP-echo-replies targeted towards the victim. Thus, the victim is subjected to network congestion that could potentially make the network unusable. If the intermediary network has less upstream bandwidth, this attack will also introduce network congestion in the intermediary network.

To amplify this attack, the attacker may use multiple compromised machines to replicate this mechanism. To launch this type of attack, an attacker does not require any special skills or tools. Since any novice attacker can launch this attack, this is one of the most dominating attacks continued to be witnessed by the internet community.

Since this attack is widely used, border routers are configured not to allow any ICMP messages inside a network. Certain firewalls monitor the frequency of the ICMP messages received per minute for a destination network. If this frequency is in an acceptable range, the received ICMP messages are forwarded towards the network. If this frequency is beyond a normal, acceptable limit, the received ICMP packets are dropped.

Due to the above mechanism of thwarting Smurf attacks, the attacker community find different ways of attacking a victim using similar techniques. This gives rise to the ***Fraggle attack***.

Fraggle attack:

A Fraggle attack [50] is a variation of a Smurf attack where an attacker sends a large amount of spoofed random UDP traffic to *ports 7* (*Echo*) and 19 (*Chargen*) to an IP broadcast address, with the intended victim's spoofed source IP address. It works very similarly to the Smurf attack in that many computers on the network will respond to this traffic by sending traffic back to the spoofed source IP of the victim, flooding it with traffic.

Since TCP and UDP packets are important user data, they cannot be dropped or delayed. Thus, most of the border routers allow TCP and UDP data across the network. To stop *Smurf* and *Fraggle* attacks, most modern-day routers by default do not forward IP broadcast packets. However, attackers are still using compromised domains to launch these types of attacks.

The DNS amplification Attack:

An attack like Smurf and Fraggle attacks but hard to defend is the DNS amplification attack [25],[30], [90] . The attacker turns a small DNS query into a much larger payload directed at the target network through this technique. The attacker sends a DNS lookup request using the spoofed IP address of the victim to vulnerable DNS servers that support the open recursive relay. The original request is often relayed through botnets for a more extensive base of attack and further concealment.

These amplifications can increase the size of the requests from around 40 bytes to 4000 bytes, which is well above the maximum Ethernet packet size. The packets larger than the path MTU (*Maximum Transmission Unit*) need to be broken down for transmission and then reassembled at the destination host. This will consume more of the victim's network resources. Like the Smurf attack, botnets [67] may furthermore be used to increase the amplitude of this attack. The attack is hard to protect against using firewalls, as it comes from valid-looking servers with valid-looking traffic.

According to Arbor's report [1], the average size of DNS reflection amplification attacks is growing significantly. The maximum observed data rate of the DNS reflection amplification attack size in the first half of 2016 was 480 Gbps. According to them, DDoS remains a commonly used attack type due to the ready availability of free tools and inexpensive online services that allow anyone with a grievance and an Internet connection to launch an attack. This has led to increased frequency, size, and complexity of attacks in recent years.

The attacker who performs the DDoS attack may not benefit directly. They might use the DoS attack to criticise the company or government organisation for exhibiting undesirable political, geopolitical, economic, or monetary behaviours. A DDoS attack might also aim to punish the victim for refusing an extortion demand or for disrupting the attacker's business model. However, many DDoS victims never learn what motivated the attack [46].

Taxonomy of attacks:

At the highest level, we divide various classes of attacks based on the attackers' origin (such as inside a LAN or from outside a corporate network). We can also classify based on the personal benefit of the attackers. We colour-code each attack cell based on the motive of the attackers. We use the flags I, O, C to denote if the attack vector is related to inside, outside or collaborative attackers.

The network attack involves all three types: *insider*, *outsider*, and *collaborative*. *Insider attacks* come only from the LAN network. *Outsider* and *collaborative attacks* are from the LAN and the WAN segments. Attacks within the LAN segment can be divided into *MITM* and *Non-MITM* based attacks. Most of the MITM attacks results from *ARP-spoofing*. *Address exhaustion*, *MAC flooding*, and *G-ARP* based attacks result in DoS. Except for these attacks, all other LAN based attacks are launched for personal benefit.

Most of the WAN based attacks results in either *DoS* or *DDoS*. Through these attacks, an attacker will not gain personally. *Port scanning* and *DNS hijacking* are *non-DDoS* based attacks that can be launched from anywhere in the network. They are launched for personal gain. Port scanning may also be used for launching a zero-day attack, resulting in a DoS.

Another class of attacks that we have not discussed here is due to bugs in software or OS. To launch this attack, the attackers can be anywhere in the network. They are launched for both personal benefits as well as for DoS. However, these classes of attack are not within the scope of the thesis.

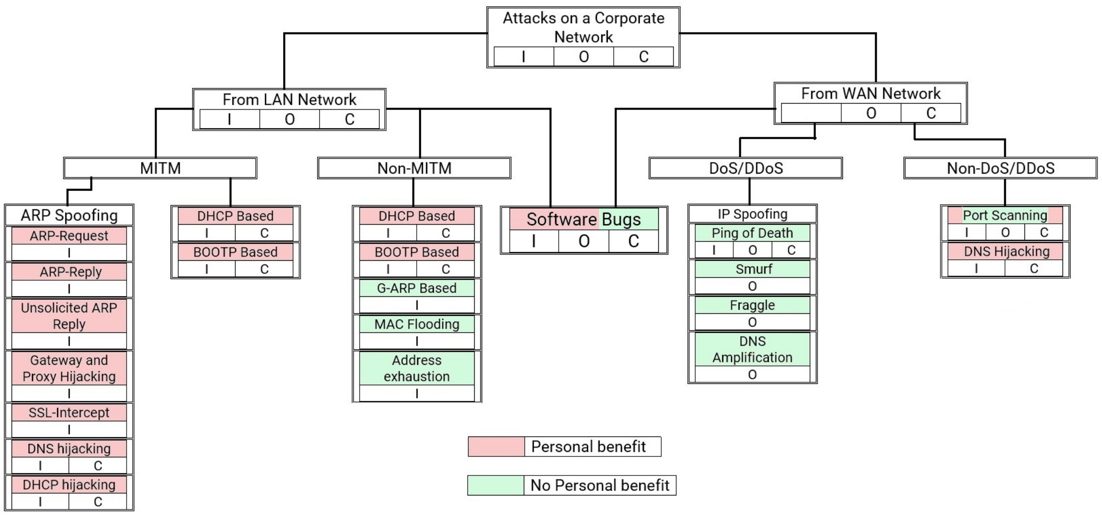


Figure 3.2: The Taxonomy of network-based attacks

Chapter 4

Thesis Scope and Research Contributions

This chapter provides a detailed overview of the following:

1. A simple introduction to the research problem under study in this thesis.

2. The scope and the Research questions addressed in this thesis.

Chapter 1 introduces the *TCP/IP* and the *OSI* model. This chapter acts as standardization for this thesis. Thus, whenever we mention ***Layer 2, 3*** etc., Chapter 1 provides an unambiguous reference to the appropriate technology and its associated tasks we are referring to.

Chapter 2 provides an overview on why a corporate network needs to be protected. Even though we mention it as a ***Corporate network***, this chapter's content applies to every large network. This chapter provides a finer difference between various types of attackers. The cyber-defense mechanism is different for different types of attackers.

Our contribution starts from Chapter 3. In Chapter 3, we provided an extensive discussion and a survey on various attacks that are due to IPv4 broadcasting. At the end of Chapter 3, we provided a taxonomy of various attacks that are due to IPv4 broadcasting.

4.1 A simple introduction to the problem

Chapter 3 contains our important contributions that sets a strong platform for our research contributions. Chapter 3 commences with an overview of the IPv4 protocol and its weaknesses. The major IPv4 weaknesses arise from the following:

- Broadcasting nature of the IPv4 protocol.

- The IPv4 protocol does not validate the source address present in its header.

There are numerous attacks that have been carried out because of the above two vulnerabilities and their vectors. To provide an effective and customizable solution, it is important to know the location of the attackers. An attacker can be anywhere on the Internet to penetrate a corporate network to obtain sensitive information. The location of the attacker is vital to launch different classes of attacks. We articulate this fact in the following section. As we discussed in Section 3.2, there are two motives for a malicious user to launch an attack against a host or a network; they are for *Personal and monetary benefits*, and *Disrupting the victim's business model*.

Based on the location of the attackers, they are classified as:

Insider attack: Where the attackers mostly come from a corporate LAN. The main motive behind this type of attack is for monetary gain and personal advantage.

Outside attack: Here, the attackers are from an external network. Disrupting the victim's business model is the primary motive for these attackers.

Collaborative attacks: In this classification, attackers collaborate with insiders to present a more powerful and destructive attack.

There are numerous solutions available in the literature focusing on solving one or a few security-related attacks in the network. However, in this thesis, we wish to provide an extensive framework for solving several cyber-attacks using a single solution. This is by examining the root cause of various cyber-attacks. Our proposed framework solves several existing cyber threats that are due to *insider*, *outsider* and *collaborative* actors. The solution proposed in this thesis is highly modular. Depending on the network requirements, modules can be added or removed without affecting the operation of the rest of the modules. This is one of the salient features of our proposal.

4.2 The Scope and the Research questions

This section focuses on the scope and research questions addressed in this thesis. As of September 2022, there is only 41.6% adaptation to the IPv6 protocol [35]. In certain countries, the penetration rate is quite low. Worldwide, still, 59% of the population is using the IPv4 protocol. Thus, the scope of the thesis is to protect any corporate network from an insider, outsider and collaborative attacks that are due to the IPv4 protocol. Providing similar solutions to attacks that are due to the IPv6 protocol is out of the scope of the thesis and will be dealt with in our future work.

4.2.1 Insiders attack

The ***man-in-the-middle*** attack is one of the most potent attacks ever witnessed by the Internet community. Personal benefit is the only motive behind launching this type of attack. The attacker intercepts the traffic between one or a few sources and a destination to launch this attack. By doing so, he will obtain all information that originates from source hosts.

To launch this attack, the attacker can be anywhere in the network. Once an IP packet (irrespective of either IPv4 or IPv6 protocol is in use) leaves any network, until it reaches the destination network, the IP packet will traverse through the ISP network and international trunks. Thus, hijacking a packet from ISP or from an international trunk is impossible (unless the attackers are state-sponsored actors). Thus, it is realistic to assume that, in this case, the actor node may either be in the source network or in the destination network. In this case, MITM attacks fall into "insiders attack".

There are numerous ways to launch a MITM attack as an *insider*.

1. The attacker may listen *passively* to the traffic exchanged between a source and the destination hosts. For this attack to be effective, no encryption must be used for the communication between the source-destination pair. This attack is popularly known as *eavesdropping or tapping*. In the present-day network, it is impossible to launch this type of attack, as most network implementations use some form of encryption for communication.

2. The *wiretap* attack is similar to the above attack. In this case, the attacker creates a *tap* to direct a copy of messages for him for analysis.

   Since *wiretaps* introduces a physical change to the network topology, it may immediately be known to the network managers.

3. If the attacker has access to the communication infrastructure, he may create SPAN (*Switched Port Analyzer*) ports, similar to wiretaps. Even though SPAN ports cannot be detected easily, the attacker needs to have access to the network resources to launch this attack, which is impossible.

One of the easiest ways to become a MITM is to *exploit* the underlying ARP protocol. Through *ARP poisoning*, an attacker may dynamically become the MITM. It is tough to detect ARP poisoning attacks. By poisoning default gateways, DHCP and critical servers, an attacker may get all sensitive information about an organization. This is one of the most powerful attacks witnessed by the Internet community, but no practical solution is available.

Based on Chapter 3 and the *attack taxonomy*, all insider attacks are due to ARP poisoning. Numerous solutions are available to address one or a few of the subclasses of ARP poisoning attacks in a network. However, there is no solution in totality. Besides, most of the solutions proposed in the literature require either modification to the protocol stacks or an *a priori* relation between a host and the network. We presented a detailed literature review on the strength and weaknesses of various ARP poisoning solutions in Chapter 6.

This is the primary motivation that leads to *Research question 1.*

Research Question 1: How to solve the insider attack without changing the protocol stack and without establishing any *a priori* relation between host machine and the network.

4.2.2 Outsider Attacks

Denial of Service (DoS) constitutes 99% of outsider attacks. Personal motivation is not the purpose of this attack. Disrupting the victim's business is the primary motivation for launching these attacks. DoS attacks are launched through the vulnerabilities in the IPv4 protocol we listed above.

Port scanning is another attack that can be launched by insiders, outsiders and through a collaborative approach. Once a vulnerable port is identified in a destination host or a network, the attackers exploit the vulnerability to become an insider. They can then launch insider attacks. If no vulnerabilities are found during port scanning, the attackers wait until a zero-day exploit is available. Currently, there is no practical solution available in the literature to stop port scanning attacks.

The above two attacks motivate us toward *Research question 2.*

Research Question 2: How to effectively stop outsider-based attacks in an IPv4 network?

4.2.3 Collaborative attacks

We now have the third attack class that requires active collaboration between an insider and an outsiders. Hosts having public IP addresses are exposed to the outside world. Any other hosts on the Internet can reach any other host with a public IP address. Thus, they are susceptible to both insider and outsider attacks. To hide all the network hosts from external hosts, NAT is used. NAT was originally invented to address the shortage of IPv4 address space. In addition to this feature, NAT provides some form of security. Since a NAT box hides internal hosts from external hosts, launching a DoS attack against a NATed host is impossible.

Moreover, a NAT box will not allow unsolicited connections from an external host. Thus, the port-scanning attack is not possible in a NATed environment. The safety features provided by a NAT can be defeated through collaborative attacks. In a collaborative attack, an external attacker may actively collaborate with an insider to expose all NATed hosts. This attack is used through IP spoofing. It is hard to detect and eliminate this type of attack in a network.

Whenever we mention insider in a collaborative attack environment, it may either refer to an active attack vector present inside a network or an external attacker may install malicious codes in a corporate network without the knowledge of its user.

The above attack motivates us toward *Research question 3.* Research question 3 consists of two sub questions.

Research Question 3: How to stop the following attacks:

(a) Collaborative attacks where an insider spoof the IPv4 address of another corporate host to open NAT ports.

(b) How to inhibit malicious outsiders from installing malicious codes?

4.3 Research contributions in this thesis

This section highlights the research contributions presented in this thesis. This thesis addresses all the three research questions mentioned in the previous section.

4.3.1 Contribution on Addressing Insider Attacks

All insider attacks are due to *ARP poisoning.* The ideal candidate for the solution must not involve any *a priori* relationship between hosts and the network, no changes to the existing protocol stack or standards, and minimal configuration.

The fundamental root cause of the ARP poisoning problem is the broadcast nature of the IPv4 protocol. Thus, the broadcast feature of the IPv4 protocol needs to be eliminated without affecting the core functionalities of the protocol. The heart of the IPv4 protocol is the ARP protocol. Without the ARP protocol, IPv4 will not function. However, ARP is inherently a broadcast-based protocol. Similarly, DHCP and the BOOTP protocols are designed based on the broadcast feature of the IPv4 protocol. Thus, whenever the broadcast feature of the IPv4 protocol is removed, sufficient care must be exercised to ensure that ARP, DHCP and BOOTP operations are not affected.

In Chapter 6, we presented the core contribution of this thesis, namely the *PrECast infrastructure.* The *PrECast* infrastructure eliminates broadcasting in the IPv4 network by converting them into an authenticated multicasting – no need to change the IPv4 protocol stack in a host or a network device. The *PrECast* infrastructure can be implemented in network devices using software service. There is no need to change the underlying ARP, DHCP and BOOTP protocols. The proposed PrECast system can run as a *software service* inside corporate LAN switches.

The core component of a *PrECast* infrastructure is the corporate multicast tree connecting all LAN switches and critical servers such as DHCP and DNS servers. In Chapter 5, we proposed an algorithm for constructing a multicast using the *Core-based tree* paradigm. Traditional public key cryptography, such as RSA, has a disadvantage, namely the key-revocation. In a large corporate network, LAN switches are decommissioned quite frequently, and new switches are added. Thus, the new devices must be provided with the new keys, and the keys for the old decommissioned switches must be revoked. To address this requirement, we proposed a modified *identity-based encryption* technique. The attackers always target PKI servers to launch DoS attacks. Our proposed encryption system can work without the presence of a PKI.

Through the contributions presented in Chapters 5, 6, we can eliminate broadcasting in IPv4 without affecting its core functionalities. Thus, we can effectively thwart any ARP poisoning problem due to insiders.

4.3.2 Contribution on Addressing Outsider Attacks

As we mentioned in Section 3.2.1.1, *port-scanning* and *DoS* attacks are launched by outsiders. The DoS attack is launched due to two vulnerabilities that are present in the IPv4 network, as we listed in Section 3.2.1.1. IP-spoofing is possible in any IP network, irrespective of the version in use. Solving the problem globally is impossible as this requires us to change the IP protocol stack. However, we defeat IP spoofing in a corporate network. In Chapter 7, we proposed a micro-NATting architecture called NAT++, which prevents IP spoofing in a corporate network. Since NAT++ architecture also hides all the corporate hosts from the external network, it is impossible to launch a DoS attack against any corporate host. However, outsiders can still launch DoS attacks against corporate servers that are facing externally through the use of public IP addresses. However, this type of attack can be protected by using *Cloudflare* or similar technologies.

The *port-scanning* attacks are classified under *Fingerprinting.* Fingerprinting is a process of identifying the remote network devices, services running on the devices, their Operating system (OS), and the information on various security patches applied to the host's OS and applications. Fingerprinting network equipment has several potential applications and benefits in network management and security. It is also useful to understand the network structures and their behaviour. The attacking community may use this feature to penetrate any corporate network. To do this, an attacker may fingerprint hosts and network devices and map their vulnerabilities. A suitable attack is launched based on the detected vulnerabilities.

We provided a simple and effective solution in Chapter 9 to defeat port-scanning. Our solution work like a *proxy* server. Whenever an attacker fingerprints a host inside a corporate network, our proxy server provides them with fictitious information about the ports and the vulnerabilities, depending on the time of the day.

4.3.3 Contributions on Addressing Collaborative Attacks

There are two classes of attacks that come under this category. An inside host can spoof the IP address of another host to open ports for outsiders to penetrate. IP spoofing also results in DoS attacks. Also, to protect inside hosts from outsiders, NAT may be used. In Chapter 7, we provided a solution called NAT++, which addresses IP spoofing and hiding the inside hosts (utilizing a private IP address) from the outside attacker. NAT++ is a modified form of a NAT that combines the best features of NAT and solve IP spoofing problem.

An innocent corporate user may click messages with hidden links that lead to malware sites and be redirected to sites that host malware through ill-formed adware. By doing so, an outsider can install malicious codes on a corporate host, and thereby the attacker can remotely connect with an inside host. In Chapter 8, we provided a solution for monitoring every incoming and outgoing connection through the reputation of the outside host. This proposal uses commercial systems such as Cisco's Talos to evaluate the reputation of any IP address.

Chapter 5

Building a Scalable and Secure Multicast Delivery Infrastructure in a Local Area Network

Internet Protocol (IP) *multicasting* is a method for *one-to-many* and *many-to-many* communication between hosts in an IP network. This communication happens in a real-time synchronous fashion. It is a valuable mechanism for distributing management data in a Local Area Network (LAN). Management data includes frequent updating of host *Operating System* (OS), *security patches*, *OS update for network hardware*, *new configuration updates*, etc. In the absence of any *admission control* or *source identification*, any host with malicious intent can disseminate malicious codes or rootkits exploiting the underlying multicast framework. Routing protocols like *RIPv2* and *OSPF* use a specific form of authentication to exchange routing information with their peer routers. However, their authentication and the distribution of routing information in its present form has numerous security and performance-related issues. Motivated through these problems, in this chapter, we propose an efficient and scalable multicast architecture for distributing management and routing information in a LAN. We use a *Core-based Tree* (CBT) for constructing the multicast delivery tree and the *pseudo-identity-based* encryption of the underlying cryptosystem. We also demonstrate that our proposed multicast architecture is immune to several prevalent attacks.

The content of this chapter is derived from the candidate's published article Veeraraghavan, Hanna and Pardede [92]

5.1 The Prelude

In a corporate network, delivering Operating System (OS) updates and security patches to all legitimate hosts is a demanding task. Similarly, updating OS, patches, and deploying new configurations to all networking devices is challenging. The challenge gets increased by several folds if the organization is spread across multiple cities. Attackers may exploit the complexity of this process to deploy malware and rootkits, pretending to be an OS update.

Large companies may have hundreds of network devices and several thousand hosts (e.g., university networks). They may spread across multiple campuses. Managing hosts and network devices from a central location is a tedious task. ***Multicasting*** is a valuable paradigm that is exploited effectively to handle these tasks. In networking terminology, multicasting is a group communication, where data transmission is addressed to a group of hosts rather than individuals. Multicast delivery can be one-to-many or many-to-many. There are several security pitfalls in the implementation and the use of multicast schemes in a corporate network. This will be addressed in detail in Section 5.2.

In a large-scale network, any rogue or malicious router may immediately become a peer to other routers and send either a forged default route or route to every external network with the best cost. This attack will make a routing table in every legitimate router to point to the malicious router for the best path. The rogue router does so to launch a Denial-of-Service (DoS) attack or to sniff the network for any valuable information. To provide solution to this attack , existing routing protocols use some form of authentication and multicast mechanism to disseminate routing updates. However, they require heavy maintenance. We will discuss this issue in Section 5.2.

In the following chapter (Chapter 6), we propose a new paradigm called *PrECast* to address several security issues such as *Address Resolution Protocol* (ARP)-Poisoning, *man-in-the-middle* attack, etc., in a corporate network. The *PrECast* protocol cannot be implemented without the existence of a secure multicast tree in a LAN. In addition to using in *PrECast* infrastructure, the proposed algorithm can also be efficiently used to distribute OS and patches to both hosts and network devices, and can be effectively used by routing protocols to distribute their routing information across the network.

5.2 The Motivation for Building Scalable and Secure Multicast Delivery Infrastructure in a Local Area Network

Currently, any multicast architecture deployed in the network has the following problems:

1. (a) They have either weak or no admission control mechanism.
2. (b) Multicast mechanisms that use a shared key for admission control have key revocation issues.
3. (c) Multicast mechanisms that depend on a Public-Key Infrastructure (PKI) have single-point failure.

Even though there are several articles available in the literature discussing various performance and security related issues in multicasting protocol, to our knowledge, none of these articles address all the above issues in total.

Router authentication is an important problem in a corporate network. Without proper router authentication in place, any malicious router (or a host that pretend to be a router) may propagate fictitious routing information. The current router authentication scheme has several performance issues.

We now address these issues in detail in this section.

5.2.1 Multicast Delivery Mechanism and Its Weaknesses

IP multicasting is a method for group communication. The number of receiving hosts may vary between one to several thousands. Thus, this technique scales well in a larger receiver community where the number of hosts is dynamically changing. Irrespective of the number of receiving hosts, the transmitting host is expected to send a packet only once. Network hardware like switches and routers replicate the packets to be sent to multiple receivers [55].

IP multicasting consists of an IP multicast group address, a multicast distribution tree, and the tree creation algorithm.

IP multicast address is a logical address used by source and receiving hosts for group communication. In an IPv4 network, any address from the *Class D* address range is used for this purpose. In an IPv6 network, the prefix *FF00::/8* is used for multicast communication.

After a multicast address is identified to be used for group communication, a multicast distribution tree must be constructed for this group communication. In this thesis, we use a Core-based Tree (CBT) to construct the distribution tree that is presented in Section 5.4.

In a traditional multicast operation, an active source host may not know about the receivers of the group. IP multicasting is a useful mechanism in distributing management data in a LAN network. Management data include frequent updating of host OS, security patches, switches and routers' OS updates and patches, new configuration updates, etc. Thus, in a corporate environment, host and network devices can be updated from one or more central locations without visiting each place.

Even though multicasting is extensively used in a LAN, it poses several challenges. We list them here:

- Any rogue host may poses as a legitimate server and deploy malicious scripts to end-hosts and network hardware. This script may be used by an attacker (either an insider or an outsider) to launch different attacks at a later time. Thus, the network must have a mechanism to detect the authenticity of a source.

- In a traditional multicast scheme, there is no admission control. Any host with malicious intent may join the multicast tree and listen for OS and security patch updates. Then, they can launch an insider attack based on the history of the patches that were applied.

- In a corporate network, PCs and network hardware are replaced constantly. Thus, membership revocation is an important issue. An attacker may use the old profile of a decommissioned host or network hardware to join the network and launch an attack.

Thus, any robust multicast implementation in a corporate network must address the issues mentioned above. Traditionally, several crypto-techniques are used to address these issues. Shared-key cryptography like ***Data Encryption Standard*** (DES) or ***Advanced Encryption Standard*** (AES) has been used for a long time. Every host and network devices use a single key. ***Message Authentication Code*** (MAC) or its ***crypto-variant HMAC*** (keyed-hash message authentication code) and the shared key are used for data integrity and authentication. Even though using a shared-key scheme solves the admission control problem, it does not identify a source node. A legitimate host with malicious intent may join the network using the shared key and pretend to be a server pushing malicious scripts to every end-host and network device. Key-revocation and key-installation are other significant issues.

The current practice is to install the keys manually. It may be impossible to automate this process unless the automation process uses a secure side-channel. Again, the establishment of the secure side-channel has similar challenges. In this case, using a side-channel consumes additional bandwidth and opens the door for further attacks. Manual key installation is possible in a smaller network; however, the process cannot be scaled to a larger network. Whenever a host or a device left the network, a new key needs to be generated and installed on to every device.

Public key cryptography is used to solve several issues mentioned above. It provides identity (through digital certification), authentication and admission control. A ***Public-key Infrastructure*** (PKI) consists of a set of rules, policies, and procedures on creating and distributing public-keys in a network. Key-management is an essential component of a PKI. In a PKI, a ***Certificate Authority*** (CA) plays a significant role. A PKI may not function without a CA. There must be a secure communication channel established between every host in the network and the CA to obtain public keys. A man-in-the-middle attack is possible without this secure channel. In a PKI, CA is a single point of failure and is frequently targeted by attackers from inside and outside networks.

5.2.2 Router Authentication

Apart from the management data, routing protocols must also deliver routing table updates securely to their peer routers. In a large-scale network, any rogue or a malicious router may immediately become a peer to other routers and send either a forged default route or bogus route to every network with the best cost. This will make the routing table in every legitimate router point towards the malicious router for the best path. The rogue router does so to launch a DoS attack or to sniff the network for any valuable information.

Router authentication is used to avoid this problem. Neighbouring routers use the authentication information to verify the contents of the routing information received from their peers. In other words, router authentication prevents routers in the network from accepting and processing routing updates from unauthorized routers that a hacker can use to create a DoS attack.

Not all routing protocols support authentication. Protocols such as *Intermediate System-to-Intermediate System* (IS-IS), *Open Shortest Path First* (OSPF), *IP Enhanced Interior Gateway Routing Protocol* (EIGRP) and *Routing Information Protocol* version 2 (RIPv2), and *Resource Reservation Protocol* (RSVP) allows network administrators to configure an authentication method and password [24].

The following authentication methods are supported:

1. **Plaintext based authentication:**

In plain-text authentication, *symmetric keys* are used by peer routers for authentication. Some protocols allow multiple symmetric keys to be configured for authentication purposes. Whenever multiple keys are used, keys are uniquely numbered for authentication purposes. In a plain-text authentication, an updating router sends its router update along with its key in plain-text and the key-number (in case multiple keys are configured). The remote router that receives this routing update first compares the received key with its stored key along with the key number. If the two keys match, the remote router then will process the routing update; otherwise, it will ignore the routing update. Since keys are transmitted in plain-text, this scheme is highly insecure.

2. **Message Digest Version 5 (MD5) and Keyed-Hash Message Authentication Code (HMAC):**

   In this scheme, keys are not sent like a plain-text authentication. Instead, keys are used to create a *message digest*. Based on the key number, a receiving router will recompute the hash value. If there is a match, the routing updates are processed; otherwise, they are rejected.

As we mentioned earlier, one or more keys need to be manually configured onto every router. This task is labor-intensive. Whenever a router is decommissioned, keys need to be removed to avoid attackers using these devices to launch an attack. The network administrators must also ensure that the startup configuration on each router is protected using passwords. Otherwise, an attacker can gain the knowledge of these keys.

5.3 Pseudo-Identity Based Encryption Framework

Identity-based cryptography is a system where a receiving host's publicly known identity is used as their public-key. This feature eliminates the need for having a PKI. However, Identity-based encryption has several implementation-related issues and key revocation problems. A k-threshold cryptography is a cryptosystem where k or more shareholders can collaboratively perform the role of a PKI (or a CA). Thus, this system is immune to a single-point failure. This section presents an innovative digital certificate mechanism using a modified threshold cryptography scheme called *Pseudo-identity-based encryption*. This framework combines the best features from both the identity and the k-threshold cryptographic systems.

Our proposed scheme identifies the origin of every message sent in a multicast tree. Since the keys are not forgeable, any such malicious activities are immediately known to the receiving host and the servers, thus facilitating key revocation action. The main advantage of our proposed scheme is that it can work without constant access to a public-key Infrastructure or a Certificate Authority. Our scheme satisfies the identical security requirements as that of the underlying public-key cryptography and incurs the same memory and run-time complexity.

As we mentioned in Section 5.2.1, using one key for all end-hosts and networking devices is not an efficient strategy. It does not identify a legitimate server against a malicious end-host. In addition, key revocation is an issue. Whenever a host leaves a system, a new key must be installed on every existing device. This process is labour-intensive. Public key cryptosystem and digital certificates can be used to identify servers and end-hosts. They also provide efficient admission control. However, in order for a public-key cryptosystem to function well in a network, every host must have a secure channel to the CA. This causes maintenance overhead and is a single point failure. Since public-key cryptography answers all the three problems we posted in Section 5.2, it is desirable to use a public-key cryptosystem with the flexibility of not having a constant connection with the CA.

Identity-based cryptography is a class of public-key cryptography proposed by Shamir [75]. This scheme uses a publicly known string such as an individual's name, organization, email address, domain name, phone number, etc., as a public-key. Shamir's scheme uses a trusted third party to deliver private keys to users upon verification of their identity. This process is similar to the issuance of a certificate in a PKI. Identity-based encryption is attractive as there is no need to fetch the receiver's public key before encryption. Even though Shamir proposed this framework, he was unable to provide a concrete implementation. Thus, identity-based encryption has remained an open problem for several years. His conjecture was independently solved by Boneh and Franklin [3] and Cocks [12].

Boneh and Franklin's solution is based on Weil Pairing and used the ***Bilinear Diffie–Hellman*** (BDH) framework over an elliptical curve finite field as their underlying hard problem. Their algorithm is called ***BasicIdent***. It has been proved that the protocol is semantically secure under the BDH assumption in a random-oracle model. Even though their algorithm is computationally secure, it is not chosen cipher-text secure. However, *Fujisaki–Okamoto* transformation allows for conversion to a scheme having this property called *FullIdent* [14]. Due to its high CPU and memory requirements, it is hard to implement this algorithm in every environment.

Compared to Boneh and Franklin's model, Cock's solution is easy to implement. His solution used the traditional quadratic residues modulo problem as their underlying hard-problem. However, as his solution uses bit-by-bit encryption, and it is not economical. In addition, Cook's solution does not preserve the identity of a recipient. This is done by passively observing the ciphertext.

Key revocation is one of the significant problems in identity-based encryption. If a private key is compromised, a user may not be offered a new key-pair. This is because the user's public-key represent his/her publicly known identity like email address, phone number etc., which are hard to change.

Since identity-based cryptography still relies on a trusted third party for obtaining the private key and the master public-key, it still suffers from a single point failure.

In [91], Veeraraghavan introduced the concept of ***Pseudo-identity based encryption***. Pseudo-identity based encryption uses the publicly available identity (such as email or mobile number) as the user's public-key. This concept is borrowed from Identity-based cryptography. However, his proposed system work without the CA. In addition, his proposal is less computationally intensive than identity-based encryption.

In this section, we modify the pseudo-identity based encryption to be used in a LAN environment.

Since the pseudo-identity based encryption algorithm is based on threshold cryptographic scheme, we briefly present an overview of Shamir's [74] threshold cryptographic scheme for completeness. Threshold cryptography was introduced by Shamir [74]. In a (***k, n***)-threshold cryptographic scheme, the secret key d is divided among n-shareholders such that

- Any adversary can compute the global secret key d by knowing k or more shares.
- It is impossible to compute the global secret key d by knowing (k-1) or fewer shares.

In a k-threshold cryptographic-system, the number k is chosen so that an adversary can break in at most (k-1) shareholders. Until recently, only organizations such as militaries, governmental agencies, and certificate authorities used this technology. However, after an October 2012 attack [84] targeting large public websites using ciphertext-compromise, RSA made this technology publicly available.

One of the salient features of a k-threshold cryptosystem is that any k or more shareholders can collaborate to create a new shareholder and create a digital certificate using the global secret key.

The threshold cryptography is based on polynomial-interpolation. For a detailed work on creating a share and digital signatures, please refer to Shamir [74].

In a threshold cryptographic scheme, the private share of each shareholder may not have any direct relation with the underlying public-key cryptography. Thus, a private share does not need to have a public-key component. In an earlier paper Veeraraghavan [91], combined the best features of both identity-based encryption and threshold cryptography, removed their potential weaknesses so that the new system can be used effectively.

The new system was called Pseudo-identity based encryption. We now outline the modification made to the threshold cryptography. For detailed work, the reader may refer to the original paper [91].

- The threshold cryptographic scheme is modified in such a way that every private share has a public-key component.
- A one-to-one relationship is established between a public-key component and the identity of a shareholder.

However, in [91], Veeraraghavan did not address the key-revocation issues. Since this is a very critical problem, we incorporate this feature in our work. We also modify the protocol to suit in a LAN environment.

We outline the pseudo-identity based system here. More details are presented in Veeraraghavan [91]. Whenever we made any modification in our work, we describe them in detail in this thesis.

As a first step, the designer must identify the underlying public-key cryptosystem. In our case, we consider RSA as our underlying public-key cryptosystem. Once this is identified, the CA must construct the global private and the public-key pair for the underlying cryptosystem. In our application domain, the CA may be a part of the IT department that maintains the inventory of PCs (end-hosts), servers, and network devices (routers and switches). They play a significant role in maintaining the list of active devices in the network.

Let N be the modulus for the underlying RSA. The CA then computes a non-trivial public and private key pair (d,e) for the system. The CA must ensure that d and e form a strong key-pair. It is easy to see that both d and e are odd numbers and $d * e \equiv 1 (mod\, \phi(N))$, where $\phi(N)$ is the ***Euler's totient function***. The next task is to distribute d to k or more shareholders through a threshold system polynomial of degree (k-1). Let $f(x) = d + R(x)$, where $R(x) = a_1 x + a_2 x^2 + \ldots + a_{k-1} x^{k-1}$ be the system polynomial of degree (k-1). The coefficients $a_i' s$ belong to some ring of integers $\mathcal{Q}$. Except k and the modulo N, all other parameters such as the key-pairs and the system polynomial are kept secret.

For a node with identity A, its secret share is $SK_A = f(A) = (d + R(A))\ (mod\, N)$. Note that SK_A may not have a public-key component. The following results are from [91], which ensures that SK_A may have a public-key component.

Theorem 1 *The necessary condition for SK_A to have a public-key component is that SK_A is an odd number and SK_A is not a multiple of N, P and Q, where $N = P * Q$ is the modulo of the underlying cryptosystem and P, Q are the prime decomposition of N.*

Since $SK_A = d + R(A)$, SK_A is an odd number if and only if $R(A)$ is an even number (as d is an odd number). The following theorem from [91] ensures that $R(A)$ is an even number for any A.

Theorem 2 *Let $R(x) = a_1x + a_2x^2 + \ldots + a_{k-1}x^{k-1}$, where $a_i's$ belong to some ring of integers $\mathcal{Q}$. $R(i)$ is an even number for every integer i if and only if the number of odd $a_i's$ are even.*

We now outline how the threshold key system is modified so that every secret share has a corresponding public-key component.

Step 1: The network administrator and the system administrator take inventory of all the current end-hosts and the network devices connected. Each device is identified through a unique name. This is in line with the best practices in network design [63]. Names are assigned to many different types of resources (such as routers, switches, servers, end-hosts, printers, etc.). A good naming model should let a user transparently access service by name rather than address.

At this stage, the network and the system administrator must also consider provision for scalability. It is a common practice to allow 10% growth per annum. Once the names are created, including for the reserved one, we move to step 2.

Step 2: Let X_i be the non-forgeable identity of the i-th shareholder. Similar to the identity-based cryptography, the non-forgeable identity may represent its unique device name, MAC address, IP address etc. It's private and public-key pair is computed as follows: Choose the smallest integer r_i^1 such that $SK_i^1 = f(X_i + r_i^1) = d + R(X_i + r_i^1)(mod N)$ has a public-key component PK_i^1 and it is not distributed to any shareholder before. Since the modulo n is usually a very large integer, such r_i^1 will always exist.

Now, SK_i^1 is the secret share for the node $\boldsymbol{X}$ with the non-forgeable ID X_i and PK_i^1 is its public-key component for this corresponding SK_i. $< SK_i^1, PK_i^1, N >$ is loaded on to X_i's configuration during the initial setup process.

Step 3: In this step, we establish a one-to-one relationship between a node's public identity and its public-key. Let $X_1, X_2, \cdots, X_n$ be the non-forgeable IDs of all the nodes in the network and their public-keys are $PK_1^1, PK_2^1, \cdots, PK_n^1$. The CA then computes the ***public-key polynomial*** $P(x) = b_0 + b_1x + b_2x^2 + \ldots + b_{n-1}x^{n-1}$ of degree $n-1$ whose b_i's are given as follows:

$$\begin{bmatrix} 1 & X_1 & X_1^2 & \ldots & X_1^{n-1} \\ 1 & X_2 & X_2^2 & \ldots & X_2^{n-1} \\ 1 & X_3 & X_3^2 & \ldots & X_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_n & X_n^2 & \ldots & X_n^{n-1} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{n-1} \end{bmatrix} = \begin{bmatrix} PK_1^1 \\ PK_2^1 \\ PK_3^1 \\ \vdots \\ PK_n^1 \end{bmatrix}.$$

Since the identities of all the nodes are different, the above matrix equation will have a unique solution. It is easy to see that $P(X_i) = PK_i^1$. We call this polynomial as a hash-polynomial for our threshold cryptosystem. The CA will load this hash-polynomial onto every end-host and device in the network during initial configuration. This polynomial is used to generate the public-key of any end-host or a device in a network. If all of the coefficients of this hash-polynomial are known to an adversary, he will not be able to compromise the system. The main advantage in distributing this polynomial is that, if a node knows the identity of any other node, it can easily compute its public-key without the presence of a CA.

After this step, there is no need for the existence of a CA.

5.3.1 Distributed CA

We now discuss a strong motivation on why a k-threshold cryptosystem is considered in this work. As we mentioned before, it is desirable to use a public-key cryptosystem with the flexibility of not having a constant connection with the CA. k-threshold cryptographic scheme is a candidate that satisfies this requirement. In a k-threshold cryptosystem, at least k parties can collaborate with each other to perform the role of a CA. Thus, a single point attack is not possible.

While setting up the system, the network administrator chooses the system-wide parameter k. The integer k is chosen so that an adversary can break in at most (k-1) nodes, but not k nodes. The administrator then configures k or more specific nodes in the network (such as servers and switches) to serve as a decentralized-CA. We call these nodes *prime-nodes*. These nodes are responsible for generating new keys and hash-polynomials. One node is chosen as the *leader* node among these prime nodes and another as a *secondary leader*. The leader node coordinates with other prime nodes in generating the keys and hash polynomial.

These prime nodes may use the sub-tree of the proposed CBT tree to communicate securely between them. They may use a different session key as well as tag their frames as key-maintenance frames, so that non-prime nodes may be unable to decrypt them.

5.3.2 Key Replacement and Key Revocation

It is pretty natural that hosts and network devices, such as routers and switches, are replaced regularly. The new, replaced device may have the same old name. One of the biggest challenges in the industry is how to ensure that decommissioned devices are not reconnected to the network to launch a replay attack using old profiles. Identity-based encryption cannot handle this case, as there is no change in the public identity of the replaced host. We now explain how the key-replacement problem can be solved using pseudo-identity based encryption without changing the node's identity.

Our key-replacement procedure also handles cases where a particular key is compromised.

Without loss of generality, let X_1 be the host whose key $< PK_1^1, SK_1^1 >$ needs to be replaced. However, we wish to keep its identity X_1. Previously, we have used r_1^1 as the smallest integer such that $SK_1^1 = f(X_1 + r_1^1) = d + R(X_1 + r_1^1)(mod\, N)$ has a public-key component PK_1^1. Now, choose the next higher integer r_1^2 such that $SK_1^2 = f(X_1 + r_1^2) = d + R(X_1 + r_1^2)(mod\, N)$ has a public-key component PK_1^2.

Once new keys are found for X_1, the hash-polynomial is recomputed. The polynomial is securely communicated to all existing and active nodes through the multicast tree architecture discussed in the following section. Whenever a decommissioned node uses its old keys to launch a replay attack, it may not be able to communicate with the existing nodes, as existing nodes use its new keys.

Whenever a key is compromised, we follow a procedure similar to the "key revocation" algorithm to generate a new pair of keys.

Since our protocol involves strict admission control using the multicast tree, any node with identity X_i being suspended in connecting to the multicast service network can be enforced easily.

5.4 Core Based Multicast Tree

In networking terminology, multicasting is group communication where data transmission is addressed to a group of hosts rather than to individuals. There are many multicast routing protocols available for IP networks, such as ***Core-based tree*** (CBT), ***Distance Vector Multicast Routing Protocol*** (DVMRP) and ***Protocol Independent Multicast*** (PIM). Each of these protocols has its strengths and weaknesses.

The heart of a multicast protocol is creating a delivery tree (called multicast delivery tree) through which multicast data packets are delivered to end-hosts. DVMRP and PIM may create as many delivery trees as the number of senders. This architecture may be helpful in one-to-many communications such as video broadcasting but may not be efficient in many-to-many communications.

There are four variants of PIM [55] based on whether they are working on a dense or sparse mode. The two modes are used based on the size of the network in order to minimize the communication cost. They are: PIM Sparse Mode (PIM-SM), PIM Dense Mode (PIM-DM), Bidirectional PIM (Bidir-PIM) and PIM Source-Specific Multicast (PIM-SSM).

Core-based tree was introduced by Ballardie et al. [2]. His main motivation for the work is to make IP multicast scalable. This is by constructing a tree of routers from one or more core nodes. The location of these core nodes is statically configured by the network administrator. Other routers in the network are added dynamically as new members join the multicast group.

In this thesis, we considered CBT for constructing our multicast tree due to its advantages over DVMRP and PIM. For each multicast group communication, DVMRP and PIM construct shortest-path trees or a combination of a shortest-path tree and a unidirectional shared tree. This requires the multicast tree routers to keep as much state information as that of the number of senders. However, there is a single delivery tree in CBT, irrespective of the number of senders. Thus, there is a reduction of state information that needs to be maintained at each router.

In a dense network, a shared tree may have some potential disadvantages. If every sender uses the same shared tree, then the traffic along the shared links could be heavy. This is not the case if the sources use separate trees. However, in our case, every network device needs to transmit to every other network device in the multicast tree. Thus, a shared tree is the preferable one.

CBT is designed with the following objectives: scalability, efficient bandwidth utilization, reduction of state information that needs to be maintained at each router. Since these parameters are critical to us, we use CBT as the underlying delivery tree creation algorithm. We modify the CBT algorithm to suit our environment.

While constructing the CBT-delivery tree, we assume that switches in the network already formed a spanning tree to avoid loops. The Spanning Tree Protocol (STP) prevents loops while still allowing for redundancy in a switched network. Details for STP can be found in the IEEE 802.1d, 802.1q, and 802.1s specifications [22]. STP-enabled switches communicate with each other to discover redundant paths in the network. This discovery results in a spanning tree that defines a single path to a given destination and alternate paths in a network outage or configuration change.

We now outline our proposed multicast tree creation algorithm as follows:

Step 1: Core Selection

The network administrator configures one or more switches in the network as the core nodes for the CBT delivery tree. The following parameters are the key deciding factors in choosing these switches as the core nodes: The strategic location of these switches in the network (especially for reducing the depth of the CBT delivery tree), CPU, memory and the backplane speed of these switches.

In Section 5.3.1, we defined prime nodes. The primary, secondary, and some of the prime nodes may be cores of the CBT. By choosing these nodes, they can collaborate with each other's prime nodes in the network to create new pairs of keys and generate hash-polynomials.

Reducing the depth of the CBT delivery tree may result in reduced end-to-end latency. The administrator wishes to select only powerful switches to be the core of the network. Switches that are not powerful enough may not handle CBT tasks on top of the standard switching operations.

Step 2: Switches That Are One-Hop Away from Core Nodes

Let S be a switch directly connected with a core switch C through port P. S wishes to join the CBT-delivery tree. S then broadcasts ***CBT-Join*** message to every switch connected with S. There is a vendor-neutral standard called Link-Layer Discovery Protocol (LLDP) ratified as IEEE 802.1AB. The primary purpose of this protocol is for devices to advertise their identity, capabilities and discover neighbours in a Local Area Network (LAN). Information gathered through LLDP are stored in the device's Management Information Database (MIB).

The join message from S contains the switches public identify. Switches that receive this message but are not a part of the CBT delivery tree yet will drop this message. In our case, the message is received by a core switch C. The following steps are taken by C to authenticate S and add it to the CBT delivery tree:

1. Based on S's public identity, C will fetch its public-key Pub_S. This is done through the hash polynomial by inputting the node's public identity. This polynomial will accept a node's public identity and outputs its public-key.

2. C creates a random challenge K. This challenge is used as a session key during the authentication phase

3. Encrypt K with Pub_S (i.e., $m = Pub_S(K)$) and send it to S through port P.

Switch *S*'s Response for the Challenge *m*

1. S decrypts m using its private key, known only to S; thus retrieving the plain-text challenge K.

   If any other rouge switch impersonates S's identity, it may not able to retrieve K, as S's private key is known only to S. In a similar fashion, man-in-the-middle attack is also not possible. These are the unique features of our protocol. In a traditional public-key crypto-system, C may not know the public-key of S. This may result in impersonation and man-in-the-middle attacks. To overcome these

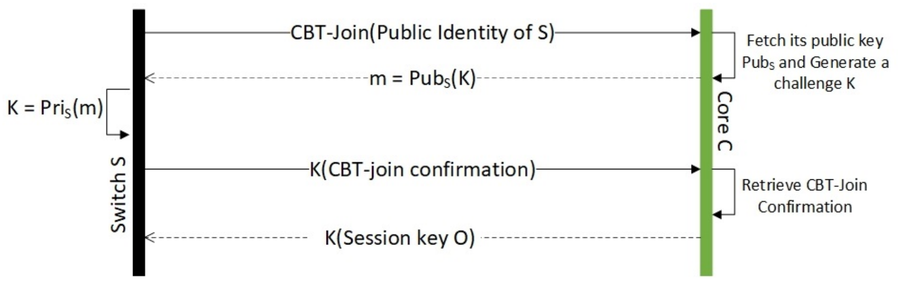

CBT-Join(Public Identity of S)
m = Pub$_S$(K)
K = Pri$_S$(m)
K(CBT-join confirmation)
K(Session key O)
Switch S
Core C
Fetch its public key Pub$_S$ and Generate a challenge K
Retrieve CBT-Join Confirmation

Figure 5.1: CBT-Join for switches that are one hop away from a core node.

attacks, Public Key Infrastructure (PKI) may be installed in the network that contains the repository of all public keys. However, a PKI system may be a single point failure.

2. After retrieving the challenge, S creates a ***CBT-join confirmation*** message, encrypt using the challenge K (***K(CBT-join confirmation)***) and send it to C through port P.

   The switch S may be connected with multiple core nodes. In this case, it will receive multiple challenges. S will retrieve all the challenges; however, it will send ***CBT-join confirmation*** to one core switch. To the rest of them, S will send ***CBT-join reset*** message encrypted using their challenges. Even though this process consumes CPU and time, S informs other core switches implicitly that it is going to connect with other core nodes.

3. After receiving ***K(CBT-join confirmation)***, C retrieves the ***CBT-join confirmation*** message from S. This process implicitly confirms that S is a legitimate switch with the claimed public identity.

4. Let O be the ongoing session key in the CBT-multicast process. C sends $K(O)$ to S and adds port P (as a forwarding port) and switches S to the CBT delivery tree. The switch S sets a reverse pointer to port P for the reverse path to its core node C. Now, S has the knowledge of the ongoing session key O and the part of the CBT-delivery tree. The message flow diagram outlining this step is given in Figure 5.1.

Step 3: Switches That Are More Than One-Hop Away from a Core Node

As before, let S be a switch that wishes to join the CBT-delivery tree.

1. S broadcasts a ***CBT-Join*** message to every switch connected with S, if it is received by a switch that is not a part of the CBT-delivery tree yet will ignore the message.

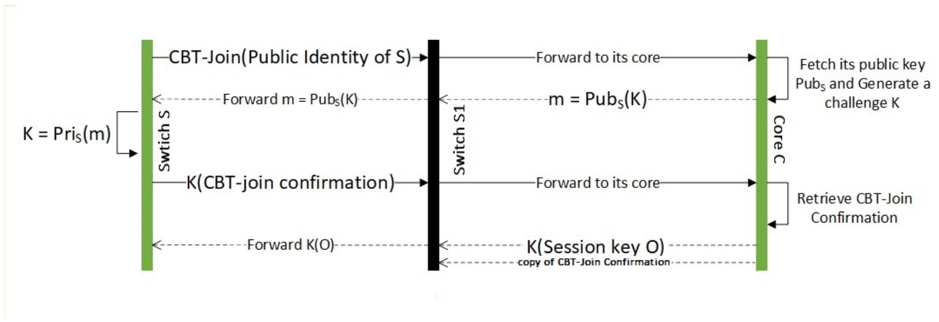


Figure 5.2: CBT-Join for switches that are more than one hop away from a core node.

2. If the message is received by a switch $S1$ that is already a part of the delivery tree, it will forward to its attached core through the reverse path.

3. Let C be the core node that receives the ***CBT-Join*** message from S forwarded by $S1$. If C receives the forward from multiple switches, it will process only the message with the best cost (shortest hop, least latency, etc.). Messages forwarded through other switches will be ignored. C then follows the same challenge-and-response scheme outlined in Step 2 to authenticate S and add to the delivery tree. The core also sends a copy of the ***CBT-Join Confirmation*** message to $S1$. Thus, $S1$ knows that S is authenticated to join the CBT tree. The switch S then creates a reverse pointer to $S1$, and $S1$ creates a forwarding pointer to S.

4. As before, if S received multiple challenges from different core nodes, it will process the one with the best cost. For all other core nodes, S will issue ***CBT-join reset*** messages.

The message flow diagram outlining this step is given in Figure 5.2.

5.4.1 Tree Maintenance

On-going maintenance of the multicast-tree is one of the important aspects in a multicast communication. Nodes may join and leave the network anytime at their will. In our case, there are two types of nodes in the network:

- ***Network devices such as switches, routers, and servers:*** They are expected to be available in the network on a 24/7 basis. They leave the network only when they are down, during maintenance or during the decommissioning process.

- ***End-host*** who may join and leave the network any time.

We do not use explicit ***keep-alive*** messages to maintain the status of a switch. This is implicitly inferred through physical layer (port connectivity) or link-layer switching activity of a switch. Explicit keep-alive messages consume more bandwidth.

We now discuss the case when a switch S wishes to leave the network.

We again spilt the case into two sub cases depending on whether S has children with respect to the CBT delivery tree or not.

Subcase 1.1: S is a leaf node with respect to the CBT delivery tree.

In this case, whenever S leaves the network, no other switch will be affected. To leave the network, S initiates a ***CBT-leave*** message, digitally signs it, and forwards it to the core C to which it is associated with. Signing the message digitally will ensure that no rouge switch or a host launch a DoS attack by initiating a ***CBT-leave*** message to disrupt the normal operation of the CBT. The core C will in turn multicast ***CBT-leave ACK*** containing S's identity and digitally signed by the core's private key. The digital signature of the core will ensure that the message originates from C, not from any rouge switch. Every switch in the CBT delivery tree receiving this information will remove S from its state or cache information. In the case of $S1$, it will close the forwarding port to S. Thus, in the future, no multicast data packet will be forwarded to S by $S1$ through port P.

There is no need to refresh the ongoing session key as no switch will forward its multicast data packet to S, including new session key.

The above mechanism works well when S is a leaf node. However, if S has one or more children, they will lose the CBT connectivity whenever S leaves the network. Thus, we follow a different mechanism whenever S has one or more children.

Whenever a host node wishes to leave the network, it will follow the same procedure outlined in Subcase 1.1. Since host nodes are leaves of the CBT, leaving the network will not affect the rest of the network.

Whenever a switch leaves the network, all hosts connected to the switch are forcefully disconnected from the CBT. Since switches leave the network only in case of a power failure, hardware or a link failure or due to maintenance. In this case, the nodes connected to these switches also lose their network connectivity.

In what follows, children means switches and routers (network hardware), but not end-hosts.

Subcase 1.2: S has one or more children (switches) with respect to the CBT delivery tree

Before we deal with this case, we present some fundamental theory that will facilitate to handle this case. Let G be a connected graph. A vertex u in G is a ***cut-vertex***, if the removal of u from G makes G a disconnected graph.

Let G be the graph of the underlying topology of the network. If G is a 2-connected graph (the size of the minimum cut-vertex is 2), then there will be at least two vertex disjoint paths between any two vertices of G.

We now return to Subcase 1.2. Let S be the CBT-node that wishes to leave the network, and S has one or more children with respect to the delivery tree. Let C be the core node to which S is connected. Let $T1$ be the sub-tree of the delivery tree that consists of the reverse path from S to C and all the descendant nodes of S with respect to the delivery tree.

Let $S1$ be any arbitrary child of S and P be the reverse path from $S1$ to C through S. Since the underlying topology is a 2-connected graph, there exists another path $P1$ between $S1$ and C not containing S. The proof is obvious. If every path between $S1$ and C passes through S, then the removal of S from the topology will disconnect C and $S1$. Thus, S is a cut-vertex of the underlying topology, contradicting the fact that the underlying topology is a 2-connected graph.

Based on the above property, we analyse P and $P1$ starting from $S1$. Let $U1$ be the first node in $P1$ (from $S1$ towards C) that is not in $T1$ (the sub-tree defined before) and U be the node that precedes $U1$ in $P1$. Such a $U1$ exists, as $P1$ is a path between $S1$ and C not containing S.

We now make an important observation. Since $U1 \notin T1$, S leaving the CBT delivery tree will not impact the connectivity of $U1$ with the CBT delivery tree. By our choice, U is one hop away from $U1$. If U establishes connectivity with $U1$, then the entire descendants of $S1$ with respect to the delivery tree T reestablishes connectivity with their core node C through $U1$.

Based on the above important observation, we handle Subcase 1.2. The steps are as follows:

1. S multicast ***CBT-intention to leave*** message to every descendant of S. This message is digitally signed by S to establish the origin of the sender.

2. Every descendant of S receiving this message will broadcast a ***CBT-rejoin*** message. This message is encrypted with the ongoing session key. Any switch receiving this message will not be rebroadcast. Let K1 be a switch that receives this message. If K1 is also an affected node, K1 will not ignore this message. It will keep in the process queue, until it gets reconnected to the CBT delivery tree.

3. Let K be a switch that receives this message from a switch L, and K is not impacted by S leaving (this is known from the fact that K has not received the ***CBT-intention to leave*** message from S). Since this message is encrypted using the ongoing session key, there is no need to re-authenticate L. Thus, K creates a

forwarding pointer to L; create a ***CBT-rejoin accept*** message; and encrypt with the ongoing session key and transmit to L.

4. On receiving this message, if L is accepting this message, it will perform the following tasks:

   - Create a reverse pointer to K.
   - Transmit ***CBT-rejoin ACK*** to K to finalize the connection. On receiving the ACK message, K will create a forward pointer to L.
   - Multicast to every node in $T1$ ***CBT-reconnected*** message. On hearing this message, S will initiate ***CBT-leave*** like in Subcase 1.1.
   - If L received ***CBT-rejoin accept*** from multiple switches, it will choose the one with the best cost (as defined before). For all other switches, L issues a ***CBT-reset*** message.
   - L might have received ***CBT-rejoin*** from several switches before it reestablished its connectivity with its core node. L will now process their request by issuing them with a ***CBT-rejoin ACK*** message; these switches will either connect with L or issue a ***reset*** message if they already have established connection.

5. As soon as S receives the first ***CBT-reconnected*** message from its descendant, it will initiate ***CBT-leave***. This message will be handled similar to sub case 1.1.

We are now left with two cases; the link between S and $S1$ is broken and S is down due to abrupt power failure.

Case 2: The Link between S and S1 is Broken.

Let $S1$ be the ancestor for S with respect to the delivery tree. As before, the underlying topology is a 2-connected graph. Thus, we can find an alternative path between S and its core C without passing through $S1$ (or using the port connected between S and $S1$). We handle this case by slightly varying Subcase 1.2.

As like Subcase 1.2, S will multicast ***CBT-disconnected*** message to every descendant of S followed by a ***CBT-rejoin*** message. On hearing ***CBT-disconnected*** message, every descendant of S, including S, will follow like Subcase 1.2 to reestablish connectivity with the CBT delivery tree.

From the link-layer activities, $S1$ will also note that either S or the link connecting S and $S1$ is down. $S1$ issues ***CBT-disconnected*** message on behalf of S to its core node. The core node and $S1$ will remove its forward pointer to S and wait for it to get reconnected.

Case 3: Switch S is Down Due to Abrupt Failure

Let $H1$, $H2$, $\cdots$, Hr be the children of S. As soon as S is down, these children will notice it from the link-layer activities with S. As like Case 2, these switches multicast ***CBT-disconnected*** message to every descendant of them followed by ***CBT-rejoin*** message. Since the underlying topology is 2-connected, they will get reconnected to the CBT delivery tree. Similar to Case 2, $S1$ will inform its core switch and remove its forward pointer to S.

We listed all possible CBT-messages at the of this chapter.

5.4.2 Security Analysis

In this subsection, we analyse how robust our algorithm is in terms of security requirements.

5.4.2.1 Impersonation Attack

There are two types of impersonation attacks possible in a network. In the first case, a malicious device (switch, router, or a host) without an active connection to the network may try to join the network using someone's identity. In the second case, a legitimate device may already join the network and pretend to be someone else.

The first case is not possible. In order to decrypt the challenge, a node must need its private key. A node impersonating someone else will not have the knowledge of the private key of the impersonating node.

In the latter case, if M wishes to impersonate L, it needs to have the knowledge of L's private key in order to digitally sign the messages. Thus, the second case is also impossible in our proposal.

5.4.2.2 Compromised Key

Whenever a key of a particular host or a device is compromised (due to a number of various reasons, which is beyond the scope of this paper), the core will generate a new pair of public and private keys for the compromised node. In our proposed model, we are not revoking the identity of the compromised node. Using the new hash polynomial, every end-host will identify the compromised host through its newly generated keys; thus, the old keys are naturally removed from the system.

5.4.2.3 Replay Attack

Whenever a device is decommissioned, its profile is removed from the entire corporate database. However, in a large organization, host profiles are stored in multiple locations and may be difficult to remove all at the same time. This deficiency may be exploited by the attackers. They may reconnect the device to the corporate network and launch a replay attack using its old profile.

In our proposed system, as soon as a new system is installed, we create a new set of public and private key pairs, without changing its publicly known identity. Our system will also distribute the new hash-polynomial securely to all nodes. Whenever a decommissioned system uses its publicly known identity, only its new keys are used by other hosts. Whenever it initiates a ***CBT-join***, the response to this ***CBT-join*** by means of a challenge will be encrypted using its new public-key. Since the decommissioned node has no knowledge of its new private key, it may not be able to join the network using its old profile. Any unsolicited message sent by the decommissioned node will be ignored, as its digital signature may not be valid.

5.4.2.4 Single Point Failure

In the case of a PKI based system, the certificate authority is a single-point failure. In our case, the certificate authority is not needed once the system is created. Since we use k-threshold cryptography as the underlying cryptosystem, a new set of keys can be generated through collaborating with k-hosts in the network. Thus, the job of the CA is distributed among k-nodes in the network. Hence, it is impossible to bring the network down by attacking a single host or a server in the network.

5.5 Chapter Summary

In this Chapter, we proposed the use of pseudo-identity based encryption along with core-based tree architecture to create a secure multicast delivery mechanism in a corporate LAN. As we discussed in the previous section, our proposed architecture is immune to a number of security attacks that are possible through the use of the existing methods. Our system can be used efficiently for the following applications:

- The secure multicast established through our proposed system can be used to deploy OS updates, patches, and device updates, including the update of new configuration. Since the messages are digitally signed, only the legitimate server can perform this operation.

- Routing protocols like RIPv2 and OSPF can effectively use this tree to send their route updates to other routers in the network.

- In Chapter 6, we propose *PrECast* architecture to provide an efficient crypto-free solution to a number of LAN based attacks that are due to IPv4 broadcasting. *PrECast* assumes the existence of a secure multicast tree in the LAN. The muticast tree constructed in this Chapter will be used as a building-block for the *PrECast* architecture.

Chapter 6

PrECast: An efficient crypto-free solution for broadcast-based attacks in IPv4 networks

Broadcasting is one of the essential features in the *Internet Protocol Ver 4* (IPv4). Attackers often exploit this feature of the IP protocol to launch several attacks against a network or an individual host. Attackers may either be a part of a Local Area Network (LAN) or outside a LAN to launch these attacks. There are numerous solutions available in the literature to solve problems resulting from IP broadcasting. However, all these solutions target a specific problem that results from IP broadcasting.

Furthermore, these solutions use either a computationally-intensive cryptographic scheme, the *a priori* relation between the host and the network or a modified protocol stack at every host. In this chapter, we provide a seamless and transparent solution to eliminate IP broadcasting and thus eliminate all problems related to IP broadcasting. Our proposed solution is crypto-free and does not need any modification to the protocol stack.

Chapter 6 consists of our prime contribution to this thesis. In this chapter, we proposed the *PrECast framework* to eliminate IPv4 broadcasting. The *PrECast* framework is built utilizing the *CBT* framework proposed in Chapter 5. Most of this chapter contents are derived from the candidate's publication [38].

6.1 The Prelude

This chapter focuses our attention on the broadcasting feature of the IPv4 protocol in a LAN environment. Several essential IP-based services like the Bootstrap Protocol (BOOTP) and the Dynamic Host Configuration Protocol (DHCP) are based on IP broadcasting. The Routing Information Protocol Ver.1 (RIPv1) uses IP broadcasting to disseminate the routing information [76]. Ethernet broadcasts are used by the Address Resolution Protocol (ARP) to translate IPv4 addresses to MAC (Media Access Control) addresses.

Since a source host has no information about the destination host in broadcast transmission, any malicious host may pretend to be the intended destination host and launch several attacks in a network. An attack may originate from a host inside a LAN or a host outside the network. However, all the attacks are targeted towards a LAN. In Chapter 3, we created a taxonomy of attacks that result from IPv4 broadcasting. One such important attack is the Address Resolution Protocol (ARP) poisoning problem. We discuss them in detail in Section 6.2.

There are numerous papers available in the literature to solve security-related problems that result from IP broadcasting. However, all papers concentrate on solving only a specific subclass of a bigger problem. No holistic approach is made to address this problem in its entirety. Furthermore, every solution either uses cryptographic algorithms to establish a relationship between an end-host and the network or uses a non-standard protocol to connect to the network. These assumptions defeat the generality offered by protocol features and the new *Bring Your Own Device* (BYOD) strategy. This forms the primary motivation for the research work presented in this chapter.

In this chapter, we follow the seven-tier architecture of the *Open Systems Interconnection model* (OSI) for our reference. The reader may refer to [89] for the activities of each layer in the OSI stack. If the context is clear, and we denote IPv4 just as IP.

Unlike other works available in the literature, in this thesis, we provide an efficient and seamless solution for all security-related problems that are due to IPv4 broadcasting. Our proposed scheme, called ***PrECast*** (*Protocol for Eliminating Broadcast*), eliminates broadcasting at the network layer without losing the functionality of broadcasting. The *PrECast* protocol has the following strengths:

- By design, *PrECast* solves every security-related issue that is due to IPv4 broadcasting.
- Unlike other papers available in the literature, the proposed solution is crypto-free and does not require any modification to a protocol or configuration at the client side. Thus, our scheme does not violate any protocol standards and support BYOD.
- Implementing our scheme requires minimal configuration at the core network, and its operation is autonomous.

- Our solution is future proof and scalable. It can easily be implemented in a Software-Defined Networking (SDN) environment.

6.2 Mitigating the ARP-Poisoning Attack

Since the ARP-poisoning attack is the precursor to several broadcast-based LAN attacks, we review the current development in this topic in this subsection. Recall that ARP-poisoning is an attack in which a malicious actor sends falsified ARP messages over a local area network. This results in the linking of an attacker's MAC address with the IP address of a legitimate computer or server on the network. Once the attacker's MAC address is connected to an authentic IP address, the attacker will begin receiving any data that are intended for that IP address.

Solutions to ARP -poisoning can broadly fall into two categories:

1. **Algorithm-based solution:** Here, solutions are mostly implemented within a client-server architecture. An agent running on either a client or the server monitors for ARP-poison attacks and provides solutions.
2. **Hardware-based solution:** Here, network hardware like switches and routers run specialized software. These software can detect and mitigate ARP poisoning.

In this section, we analyse the strengths and weaknesses of each class of solutions.

6.2.1 Algorithm-based Solutions

Bruschi, Ornaghi and Rosti [4] proposed a cryptographic scheme to address the ARP-poisoning problem, called a *Secure Address Resolution Protocol* (S-ARP). They introduced a set of functionalities that enable an integrity and authenticity check on the content of ARP replies using asymmetric key cryptography. In their scheme, every host in the network has a *public* and a *private key* pair, certified by a *local trusted party* on the LAN. The certificate from the trusted party provides a binding between the host identity, its public key and its IP address. Each host sends its signed certificate containing the public key and the IP address to the *Authoritative Key Distributor* (AKD), which inserts the public key and the IP address in a local database after validation. Any S-ARP-enabled host is identified by its IP address and the public/private key pairs. Using this key pair, S-ARP provides message authentication. However, S-ARP does not offer any *traffic confidentiality.* This feature is left to the higher layers in the OSI stack.

In S-ARP, all *ARP-reply* messages are digitally signed by the sender with the corresponding private key. On the receiving side, the signature is verified using the host's public key. If the public key of the sender is not available, the receiving host will send a request to the *AKD*. The *AKD* in turn sends it to the requesting host in a digitally-signed message. The first step in the S-ARP scheme is to identify the *AKD* and distribute through a secure channel its public key and *MAC* address to all the other hosts. Such an operation may be performed manually when a host is installed on the LAN for the first time.

S-ARP has several disadvantages. We list some of the main disadvantages here:

- S-ARP requires public and private keys for every host in the network. The key pair has to be generated either by the host itself or by the network administrator. Leaving it to the network administrator adds an overhead. In the first case, not every host may generate a strong key pair. An attacker may impersonate a node that has a weak key pair. In addition to this, this technique may not scale in a large-scale network.

- The public key and the MAC address of the *AKD* need to be sent to every host securely. Thus, this scheme can only be implemented in a restricted private network environment.

- Key revocation is one of the critical issues that are hard to implement using this scheme.

- An attacker may launch a *DDoS* attack against the *AKD* to bring the entire network down.

Lootah, Enck and McDaniel [54] proposed the *Ticket-based Address Resolution Protocol* (TARP) to solve the ARP-poisoning problem. Their main objective was to provide a cost-effective solution to the ARP-poisoning problem. The major flaw in the design of the ARP protocol is the lack of authentication. Due to this flaw, an attacker can forge an ARP reply to inject a new address association into the victim's cache (called reply spoofing) and forge an ARP reply to replace an address association in the victim's cache (called *entry-poisoning*). TARP addresses these two vulnerabilities through *attestation* (called *tickets*) generated by a centralized server. Their ticketing scheme is based on [57, 78]. Each ticket has a validation period.

The TARP process is similar to the ARP mechanism: a host that requires a $<$MAC, IP $>$ binding will broadcast an ARP request. The host with the requested IP address sends a reply, along with its ticket. The validity of the ticket can be verified using the public key of the *Local Ticketing Agent* (LTA). Since an adversary does not know the private key of the LTA, he/she may not be able to forge a ticket. To make their proposal less complex, they propose using the DHCP server for the role of LTA.

Since TARP adds the missing authentication to the ARP protocol, it solves the ARP-poisoning problem. Compared with the S-ARP protocol, TARP is less complex. There is no need for every host to have a public and private key pair. The same functionality of S-ARP is achieved through the use of the *digital certificate* from the LTA.

However, we note that their proposed scheme has the following vulnerabilities:

- Like any centralized service, LTA is a single point of failure. Launching a *DDoS* attack against LTA can bring the entire network down.

- There is no mechanism for distributing the public key of LTA to all clients. Without a secure delivery mechanism, any intruder can pretend to be the LTA, distribute his public key, and sign the <MAC, IP > bindings. In S-ARP, this requirement is explicitly mentioned.

- TARP in its current state cannot detect rogue DHCP servers. In a multilayered network, there is a latency between the legitimate DHCP server and some clients. This may be exploited by a rogue DHCP server that distributes IP addresses and tickets.

- To make the protocol more secure, TARP must provide an efficient ticket revocation mechanism that securely notifies all active clients about which tickets are no longer valid. Any such mechanism will incur severe network overhead in a dynamic network like a public hotspot or university network.

- Even though TARP solves the ARP-poisoning problem, the broadcast mechanism is still used for the ARP request. Thus, TARP suffers performance degradation due to the broadcast mechanism.

In [58], Nam et al. proposed an enhanced version of ARP called *MR-ARP* to prevent the ARP-poisoning attack in Ethernet-based LAN environments through a voting mechanism. *MR-ARP* determines the owner of the IP address by giving a higher priority to the previous owner in the case of conflict of the MAC address of the owner. If an *MR-ARP* node observes a conflict for the owner of an IP address that is not registered in its own ARP cache, the *MR-ARP* triggers voting on the owner of that IP address among the neighbour nodes to make a decision based on the voting results. Since the decision is made through a voting mechanism, *MR-ARP* must ensure that fairness in the voting mechanism is maintained. In a wired LAN, all nodes have the same transmission rate and thus are likely to send a similar number of voting packets during a particular interval of time. However, this is not possible in a wireless network. This is because the transmission rate in a wireless network may be different for different nodes due to the traffic rate adaption based on the signal-to-noise ratio. To resolve this unfairness in a wireless segment, *MR-ARP* introduces a computational puzzle in the voting procedure. Thus, *MR-ARP* has a different fairness mechanism for different nodes.

In a later paper [62], one of the authors of [58] proposed some generalization to *MR-ARP* and called it the *Generalized MR-ARP* (*GMR-ARP*). The fairness in voting was improved in *GMR-ARP* by dropping reply packets that arrive too early. We noted that both proposals [58, 62] have a number of shortfalls:

- In order to implement *MR-ARP* or *GMR-ARP*, the protocol stack at the client-side needs to be redesigned. Any end-host must have the new protocol stack installed before connecting to the network. This is an undesirable requirement in a public environment (like a university, airports, or restaurants).
- Their voting process consumes a lot of network overhead. This process will also consume resources on every participating host as the broadcasting mechanism is used by the voting process.
- Their proposal requires every host to respond to a voting request as early as possible, and each host is expected to send a certain number of voting replies. However, how soon a node sends its voting reply depends on the number of processes running on the host and the queue length of the buffer at every host. An attacker could use this mechanism to launch a DoS against every host in the network.
- Both *MR-ARP* and *GMR-ARP* require every host to maintain a long-term <MAC, IP > binding table. This consumes memory. Furthermore, to keep the ARP cache entries active, hosts have to send an *ARP-request* to every entry in the long-term table. This incurs unnecessary network overhead.
- Since the neighbourhood density is unknown to every host in the network, and it is hard to set a threshold value for the number of good votes.
- Even though the authors proposed a scheme to eliminate the overhead due to cryptographic tools, their proposed scheme incurs more traffic due to the voting mechanism in addition to the existing ARP broadcasting overhead.

In general, any algorithm-based solution to eliminate ARP-poison attacks has the following problems:

- Most of the research proposals introduced some form of message authentication through the use of cryptography. They also used a centralized certificate authority to issue digital certificates. The use of any cryptographic tool at the client-side will introduce CPU latency due to computational requirements. Severe network latency is introduced if the cryptographic computations are done at the server-side. Thus, any scheme that uses cryptography is only suitable for a smaller network.

- DDoS is a potential problem for any scheme that depends on a centralized service.
- Redesigning or implementing a modified TCP/IP stack at the client-side is possible only in a smaller private network. For this type of network, a simpler alternative is to use a static ARP table that eliminates the ARP-poisoning problem and improves the performance in the absence of any ARP broadcasting. Implementing a modified TCP/IP stack is impossible in a network that uses the concept of BYOD.
- There are few non-cryptographic techniques available in the literature to eliminate the ARP-poisoning attack; however, they use collaborative neighbour-based message authentication. In order for these protocols to work, the network has stringent conditions. One condition is to ensure that the number of well-behaving nodes exceeds the number of attackers. These conditions can easily be met in a private network but are very difficult to ensure in a public network. Furthermore, in the absence of any incentive for message authentication (or voting) for other nodes, selfish nodes might restrain from voting to save their CPU and other network resources. Thus, a minimum threshold for message authentication may not be maintained.

6.2.2 Hardware-Based Solution

From the critical papers reviewed in Subsection 6.2.1, it is apparent that every algorithm-based solution requires either a modification to the protocol stack and/or the use of heavy cryptographic tools, along with previously-established stateful information between a host and the network. Thus, they may not support BYOD concepts.

Cisco introduced a hardware-based solution to address this security concern. Their algorithm runs on their switches and routers and thus does not need any modification at the client device.

Cisco [10] introduced several security features in their switches to protect a LAN. *DHCP snooping* protects the network from rogue DHCP servers. The network administrator configures the port to which a legitimate DHCP server is connected as a trusted port. The remaining ports are configured as untrusted ports. The switch will drop all DHCP-server messages that originate from untrusted ports. The switch also builds and maintains a DHCP binding database consisting of the MAC address, IP address, lease time, binding type, VLAN and interface-ID. In a *DHCP-exhaustion attack*, an attacker can exhaust the pool of available IP addresses of the DHCP server within seconds. This is done by flooding *DHCP-discover* packets requesting new IP addresses. The DHCP-snooping technique does not only mitigate rogue DHCP servers but also defends against *DHCP-exhaustion* attacks.

Even if switch-port security is in place, tools such as *Yersinia* or *dhcpstarv* are used to randomize the field *Client Hardware Address* (CHADDR) inside the DHCP payload to send multiple *DHCP-requests* [100]. This process exhausts the DHCP address space within a short time. However, DHCP-snooping is clever enough to read the payload of the DHCP protocol and verify that the source MAC address and CHADDR are the same to defeat such starvation attacks. This extra feature needs to be enabled through an optional command.

Cisco's patented *Dynamic ARP Inspection* (DAI) [10] is a security feature currently built into their switches that validates ARP packets in a network. *DAI* intercepts, logs and discards ARP packets with invalid IP-to-MAC address bindings. This capability protects the network from Layer-2 MITM attacks. The *DAI* feature learns the <MAC, IP > binding from the DHCP-snooping table. Even though the *DAI* feature solves the ARP-poisoning problem in a LAN, it still has the following problems:

- In a stacked switch environment, DHCP-snooping is managed on the stack master. When a new switch joins the stack, the switch receives the DHCP-snooping configuration from the stack master. Whenever a member leaves the stack, all DHCP-snooping address bindings associated with the switch age out. However, in a non-stacked multi-layered environment, DHCP-snooping is hard to implement.

- As mentioned in [86], many network components require changing MAC addresses. Therefore, care should be taken when defining protections against ARP-poisoning. This includes the *Hot Standby Router Protocol* (*HSRP*), hosts configured with load balancing solutions such as *Microsoft NLB* (*Network Load Balancing*), clustering solutions, and so on. However, disabling the *DAI* feature on these ports could weaken the security. An intelligent attacker may exploit this weakness and launch an attack.

- As we mentioned before, *DAI* learns the legitimate <MAC, IP > bindings from the DHCP-snooping database. Thus, all static <MAC, IP > bindings must be manually entered into the DHCP-snooping database. For *DAI*, an *Access Control List* (ACL) must be created to include all static <MAC, IP > bindings. This process creates more overheads for the system administrator. Any change in the binding must be manually entered into the ACL. It is more common in an organization to assign static IP addresses to servers and printers. As soon as they are decommissioned, their bindings need to be removed from the ACL, as well as from the DHCP-snooping database.

- The DHCP-snooping configuration requires a highly skilled network administrator when *DHCP Option 82* is enabled. For more detail on *DHCP-Option 82*, please refer to [76]

- *DAI* will not protect wireless hosts against the ARP-poisoning attack.

To our knowledge, none of the research work in the past addressed the shortfalls mentioned above. This is a main motivating factor for our solution architecture in the following section.

6.3 The PrECast Solution Architecture

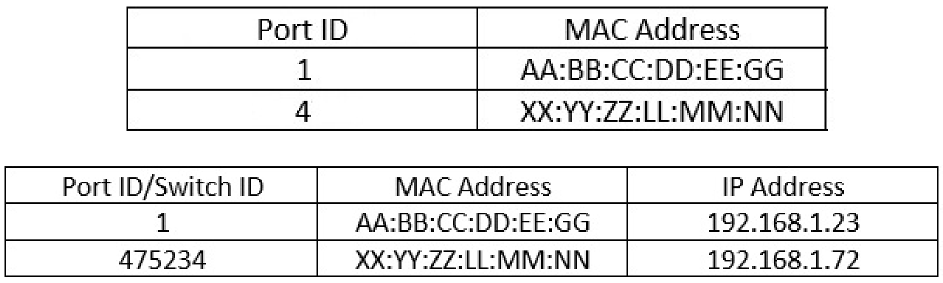
In this section, we provide a detailed specification of our proposed *PrECast* architecture to eliminate IPv4 broadcasting based attacks resulting from IPv4 broadcasting. Similar to the DAI architecture, our solution is hardware-based and thus requires no configuration at the client side. We focus on critical network and system performance metrics while designing our solution. They are efficiency, scalability, flexibility and zero-configuration at the client-side. Broadcasting consumes considerable network bandwidth, CPU and buffer resources at every single network device, such as switches and hosts on the network. However, several network protocols like ARP, DHCP and the *Routing Information Protocol* (RIPv1) depend heavily on broadcasting services. We estimated that almost 70% of the broadcast traffic is due to the ARP protocol in our university network.

Before we describe the *PrECast* architecture, we explain how LAN switches are modified to run our proposed solution.

6.3.1 Modification to Switching Devices

IPv4 broadcasting is restricted to an IP subnet and a single Ethernet broadcast domain. The Ethernet broadcast domain consists of Ethernet switches, repeaters and hubs. Since repeaters and hubs are obsolete, today's network uses only switches to create a local-area segment. In this subsection, we present what essential software modification needs to be done at switches that will support the *PrEcast* protocol. This modification must be done by the switch manufacturer.

Since Ethernet is the most popular and the de-facto Layer-2 protocol, we demonstrate our solution through Ethernet segments.



Port ID	MAC Address
1	AA:BB:CC:DD:EE:GG
4	XX:YY:ZZ:LL:MM:NN

Port ID/Switch ID	MAC Address	IP Address
1	AA:BB:CC:DD:EE:GG	192.168.1.23
475234	XX:YY:ZZ:LL:MM:NN	192.168.1.72

Figure 6.1: Traditional *Content Addressable Memory* (CAM) table and a modified CAM table.

By design, Ethernet switches do not care about the content of any higher-layer protocols encapsulated inside an Ethernet frame. As a first step, a manufacturer modifies this feature so that switches open an Ethernet frame and record the source and the destination IP address present in an IP packet. Today's corporate network uses Layer-3 switches, mainly for *QoS support*, *manageability*, *access control*, and so on. Thus, these switches need no modification and support Layer-3 features natively.

Switches use a special type of memory called *Content Addressable Memory* (*CAM*) to store *port* numbers and *MAC* address relations in order to switch a frame effectively [96]. *PrECast* adds one more field to include the IP addresses of the hosts associated with the MAC addresses. We call this table a *Modified CAM* (*MCAM*) table. The structure of the traditional CAM table and the *MCAM* table is presented in Figure 6.1. The first column represents the *port ID* or the *switch ID*. If a host is directly connected to the switch, this field represents the *port* number to which the host is connected. If the *MAC* address corresponds to a remote host, then this field represents the *ID* of the switch to which the host is connected. The *MCAM* table can either be kept at the *DRAM* of a switch or the *CAM* architecture can be modified to include a 32-bit IP address along with other standard information.

If a switch cannot be modified, it can still be a part of the *PrECast* infrastructure. However, the IPv4 broadcast-related issues still exist and are confined to hosts that are connected with a switch that does not support the *PrECast* protocol.

6.4 Setting up the PrECast Infrastructure

Since switches are the building blocks of a local area network, the network hardware must be physically secured as the first line of action in securing a network. Furthermore, the running configuration on these switches needs to be protected through the use of strong passwords. Any attempt to change the access privilege level or the password must trigger an alarm.

The network administrator must install the cryptographic key for each switch. Only the legitimate switches that the system administrator configures will take part in the *secure multicast* process outlined in our proposal. Any other switches that try to join the multicast group will be rejected. This may include switches that are plugged into the network by end-users to expand their connectivity, which may not support the *PrECast* infrastructure. The administrator must also set appropriate time-out values mentioned in Section 6.4.1

These are the only activities that require the administrator's involvement. All other processes mentioned in the *PrECast* protocol are automatic and do not need any involvement from the administrator. Since the inter-switch communication needs to be protected in a corporate environment from eavesdropping, the administrator needs to install certificates and private keys irrespective of the *PrECast* architecture. Thus, the processes that involve the administrator in the *PrECast* protocol do not require an extra overhead.

6.4.1 The PrECast Algorithm

The first step in the *PrECast* algorithm is to build the *MCAM* table dynamically. A host can either obtains its IP address dynamically from the DHCP server or its IP address is assigned manually by the system administrator. In these two cases, we describe how *MCAM* tables are built.

In any corporate network, there will be few hosts that use static IP address; for example, *DHCP* servers, primary and secondary *DNS* servers, default gateway, and so on. Since this thesis deals with issues that arise due to IPv4 broadcasting, we assume that hosts that are connected to our network have Layer-3 connectivity through IPv4. To gain Layer-3 connectivity to the network, every host on the network must have a unique IP address assigned to it. The IP addresses can be either assigned statically by the system administrator (in this case, the IP address will never change) or a host may obtain its IP address dynamically from a DHCP server.

6.4.1.1 IP Address is Configured to a Host Manually

We assume that the IP address is assigned to Host A manually.

The conventional CAM table build-up process is as follows: During the boot-up process (or during the IP address binding process), Host A will issue a *G-ARP* (Gratuitous ARP [76]) broadcast packet to everyone to ensure that the assigned IP address is unique in the network. We assume that Host A is connected to Switch S. The switch port to which Host A is connected is the first network device to receive this encapsulated Ethernet frame. In the conventional process, the switch will record the port from which this Ethernet frame is received and the source MAC address present in the frame to its CAM table. Once this is done, the switch will forward this frame to every other active port in the switch, except the port from which this frame was received. Other switches receiving this broadcast frame will follow the same process as that of Switch S. In the event of any IP address conflict, a host may send a unicast reply to this G-ARP request. The G-ARP reply will be switched back to Host A using the entries available in the CAM table of every switch in the reverse path.

We modify this conventional process as follows. Note that we have not made any changes to the protocol standards and operation, thus even if a switch does not support the *PrECast* process, it can still coexist with the *PrECast* infrastructure.

Since our *PrECast* protocol works transparently, Host A broadcasts a *G-ARP* packet through switch port S_A of the switch S. This packet is then be verified by the switch for any possible IP address conflict as follows:

- Switch S will first check with its *MCAM* table for a possible IP address conflict by referring to its *MCAM* table. In case of any IP conflict, Switch S will craft a *G-ARP* reply and inform Host A.

- If there is no entry available for the same IP address in its *MCAM* table, S will *multicast* the *G-ARP* request to all other switches that are securely known to S. Switches that receive this *G-ARP* frame will check with their *MCAM* table for a possible address conflict. The switch that detects an address conflict will send a *G-ARP* reply to S; S will in turn craft a reply back to Host A.

- If the IP address is not mapped onto any other MAC address, S will not receive any reply from other switches until the time-out set out by the ARP-standard. After this time-out, switch S will add the switch-port S_A, the MAC address of Host A and its IP address to the *MCAM* table.

- In the absence of any *G-ARP* reply, Host A will bind the IP address like the conventional process.

6.4.1.2 A Host Obtained its IP Address through a DHCP Server

In this case, after booting the system, Host A will broadcast a *DHCP-discover* packet. This packet in general is sent to every host in the network. The DHCP server available in the host's IP network will unicast its reply to the host with the IP address and other relevant network information.

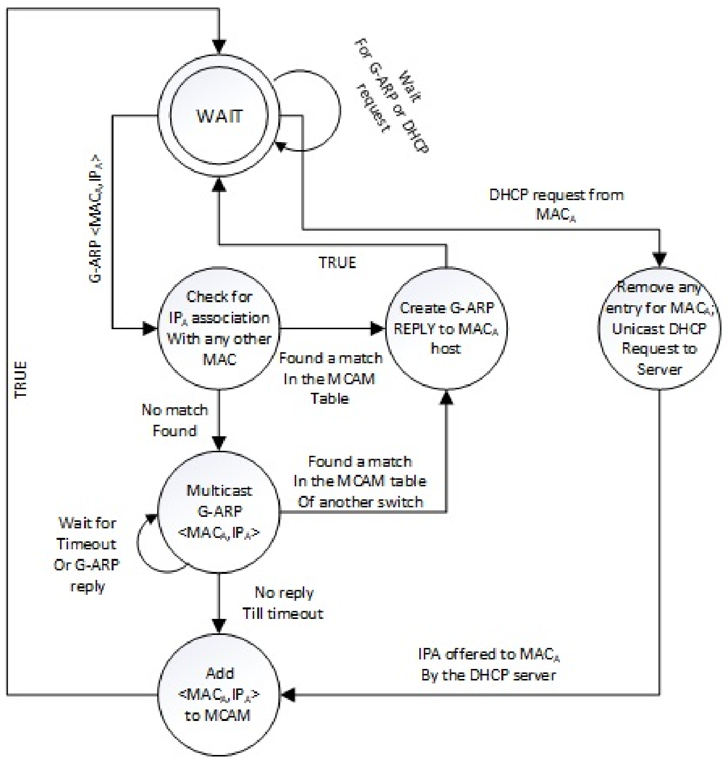

WAIT
Wait For G-ARP or DHCP request
G-ARP <MAC_A, IP_A>
DHCP request from MAC_A
TRUE
Check for IP_A association With any other MAC
Create G-ARP REPLY to MAC_A host
Found a match In the MCAM Table
Remove any entry for MAC_A; Unicast DHCP Request to Server
TRUE
No match Found
Multicast G-ARP <MAC_A, IP_A>
Found a match In the MCAM table Of another switch
Wait for Timeout Or G-ARP reply
No reply Till timeout
IPA offered to MAC_A By the DHCP server
Add <MAC_A, IP_A> to MCAM

Figure 6.2: MCAM table build-up process

We modify this traditional scheme to eliminate broadcasting due to the *DHCP-discover* process in the network as follows. Like the previous subsection, the switch port S_A receiving this *DHCP-discover* packet will note down Host A's MAC address and unicast to the DHCP server available in the network. The DHCP server information is known to the switch through the manual configuration process outlined in "the boot strap process". The switch will wait for the *DHCP-offer* message from the server. Whenever the DHCP server offers the lease of an IP address to the requested host, the switch will complete the *MCAM* table that will now contain the *port ID* (which is S_A), the *MAC* address and the *IP* address of Host A.

Since there are only two different ways of obtaining the IP address for a host, in both schemes, the switch will transparently build its MCAM table.

The state transition diagram of the MCAM table build-up process of a switch is presented in Figure 6.2.

6.4.1.3 The PrECast ARP request Process

Our approach is similar to the conventional ARP request process. Host A needing to know Host B's MAC address will broadcast an ARP request packet. This packet will be received by Switch S through the switch port S_A. Instead of forwarding to every host in the network, *PrECast* will modify the ARP process as follows:

- Switch S will check its *MCAM* table for a possible <MAC,IP >binding for Host B. If the required ARP-mapping information is available, S will send a unicast *ARP-reply* to Host A.
- If there is no binding information available at Switch S, S will securely multicast the *ARP-request* to all other switches in the network. The switch to which Host B is connected (say Switch T) will send a multicast reply containing the <MAC, IP >binding of Host B to every switch in the network. Switches receiving this multicast reply will add Switch T's ID, <MAC, IP > in their *MCAM* table for possible use in the sequel. Eventually, Switch S will send an *ARP-reply* back to Host A based on the information obtained from Switch T.
- If no switch knew about Host B, like the conventional ARP process, the ARP request will go for time-out. This will be known to all switches in the network, through the absence of any multicast reply.

As can be seen from the above steps, the *PrECast* ARP algorithm is transparent to end-hosts. Hosts neither need a modified protocol stack, nor to be aware of the existence of the *PrECast* system running in the network.

The state-transition diagram of an *ARP-lookup* process for the proposed *PrECast* algorithm is presented in Figure 6.3.

6.4.1.4 *MCAM* Table Maintenance

We now present the *MCAM* table maintenance process due to the ageing process. The traditional *ARP-cache* table stored at every host has a *time-out* value. The *time-out* value depends on the particular OS running on the host. Having a *time-out* value has two advantages. Since we are removing any stale ARP cache entries from the memory, having a *time-out* value can conserve the memory. *Time-out* also has another important purpose. Within the duration of the time-out, the binding relation between the MAC and IP address of a host might have changed. For example, in a DHCP-based environment, the IP address lease may have expired, and the same IP address is allocated to another host. In a static IP environment, the administrator might have replaced a faulty network adapter that resulted in a change in the binding (this event may be quite rare in a network).

Since the *MCAM* table kept at a switch is similar to the *ARP-cache* table, it is essential to perform the above maintenance. This is to ensure that the entries kept in the *MCAM* table are current and valid.

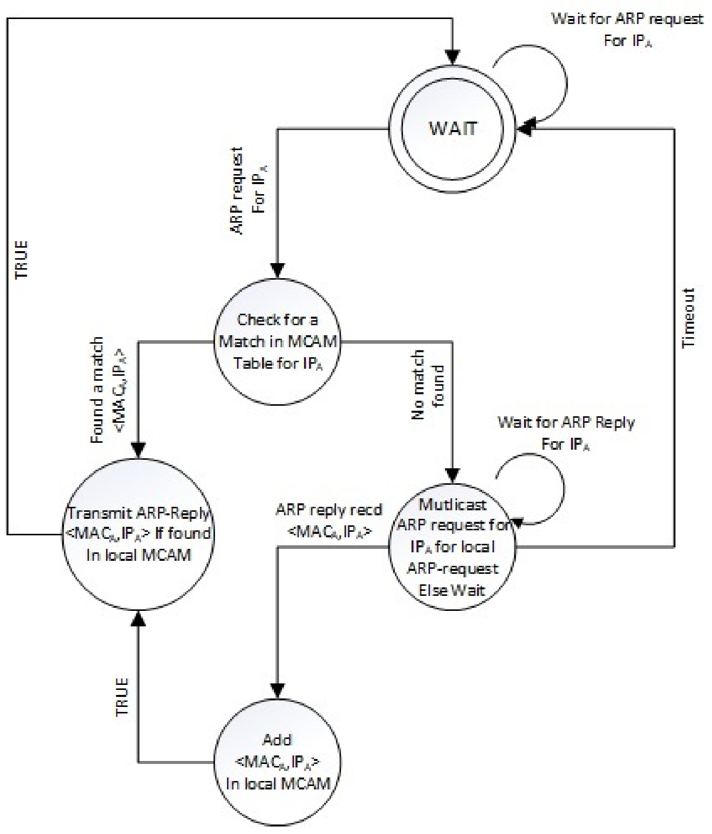

Wait for ARP request For IP_A
WAIT
ARP request For IP_A
TRUE
Timeout
Check for a Match in MCAM Table for IP_A
Found a match <MAC_A,IP_A>
No match found
Wait for ARP Reply For IP_A
Transmit ARP-Reply <MAC_A,IP_A> If found In local MCAM
ARP reply recd <MAC_A,IP_A>
Mutlicast ARP request for IP_A for local ARP-request Else Wait
TRUE
Add <MAC_A,IP_A> In local MCAM

Figure 6.3: Protocol for Eliminating Broadcast (PrECast) ARP-lookup process.

In our proposed scheme, we consider two types of time-out values. The first one is the *forced-time-out* value. The *forced-time-out* option is used whenever there is a change in the MAC and IP binding relation. The second one is the *traditional time-out* value that deals with stale entries. We explain them below in detail.

The Forced-Time-Out:

Let t be the *time-out* value. Either between $[0, t]$, or while an <MAC,IP >entry is flagged as dormant, a host may leave the network. Whenever a host leaves the network, we remove its <MAC,IP>binding from every switch in the network. In a dynamic environment, the same IP address is issued to another host with a different MAC address. We give below an example of cases where a *forced-time-out* is used to remove entries from every switch in the network.

Let us assume that Host A is connected to Switch S. There are a number of cases that can trigger a *forced-time-out*. They are:

- Host A issue a *DHCP-release* message after closing its connection.
- The DHCP server has issued a *lease expiry* message to Host A, and Host A has not responded to it.
- No $L1$ or $L2$ activity from Host A for a period of time (this period of time is a parameter set by the system administrator).
- Host A has changed its network card.

If one or more of the above cases happen, Switch S will trigger an *unsolicited ARP* multicast reply with $< ID_S, MAC_A, \emptyset >$ with a zero time-out value. Here, ID_S is the *switch ID* for S; MAC_A is the MAC address of Host A; and the IP field is empty. All switches that receive this multicast reply will remove the entry for Host A from their *MCAM* table.

The Normal-Time-Out:

Every entry present in the *MCAM* table has a time-out value. If the switch detects a new frame with a particular *MAC* and IP combination, then the timer for the corresponding entry in its *MCAM* table is reset to the current system time. This means that, whenever a <MAC, IP >binding combination is in use, the corresponding entry remains fresh in the MCAM table. If a <MAC, IP>binding is not in use for the entire duration set in the *time-out* value, then the corresponding entry in the *MCAM* table is flagged as *dormant*. This process is different from the *ARP-cache* time-out process, wherein any stale entry will be removed from the memory. The main reason for us to propose this is to ensure that, whenever the same <MAC, IP>binding is needed, it is readily available without incurring any load on the network.

If a host connected to a switch issues an ARP request to another host for which its binding is flagged as *dormant*, the switch will change this entry from *dormant* to *active*, and a new timer value is set. Rather than removing an unused entry from the *MCAM* table, flagging them as dormant saves a pair of multicast request and a reply, if the entry is needed at a later point of time.

Based on the above two time-out values, *PrECast* can assure that the active entries kept in the *MCAM* table of any switch are always fresh.

6.4.1.5 Robustness of the PrECast Protocol

In this subsection, we analyse the robustness of the *PrECast* protocol. As we mentioned before, a switch that does not implement the *PrECast* protocol can still coexist with the *PrECast* infrastructure. However, the broadcast-based IPv4 attacks still exist and are confined to hosts that are connected only to the unsupported switches. The *PrECast* switch-port to which a non-supportive switch is connected will record the <MAC,IP >mapping of all the hosts connected with the non-supportive switch in its *MCAM* table.

We now discuss the case when a *PrECast* supportive switch, say S_1, goes down. Assume that the switch S_1 is connected with port P of a switch S_2. Whenever, S_1 is down, S_2 will not detect any Layer-1 activity on port P. If there are no activities for the time set out by the administrator, S_2 will initiate the forced-time-out procedure set in Section 6.4.1.4. S_2 will multicast an *unsolicited ARP-reply* for all the hosts connected with switch S_1. All active switches that receive this *unsolicited ARP-reply* will remove the host entries from their *MCAM* table. A similar procedure is followed if a host connected to a switch port is down.

If the binding between a host and the switch is broken and still there is a Layer-1 activity between the switch and the host, a forced-time-out procedure will not be invoked; rather, the respective *MCAM* entry will be marked as *dormant*. Whenever the broken link is restored, the *MCAM* entry will be changed from *dormant* to *active*. Thus, this process will not incur an extra overhead.

6.5 Performance Analysis of *PrECast*

In this section, we compare the performance of *PrECast* against the conventional *ARP* protocol. First, we present the asymptotic analysis dealing with the message complexity and the convergence time. These two parameters are inter-related and important for the overall network performance. In the second part of this section, we analyse the performance of *PrECast* using simulation.

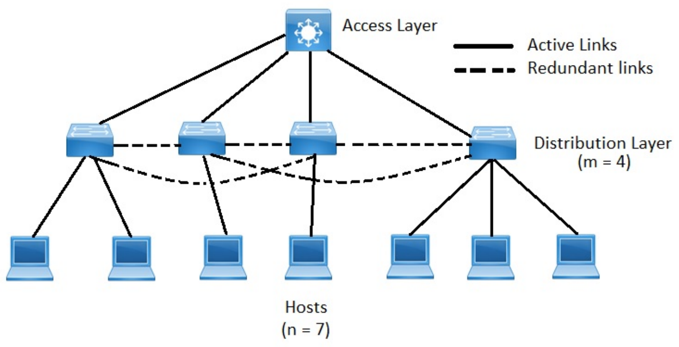


Figure 6.4: An example of a network topology

6.5.1 Communication Message Complexity

Communication message complexity in our context determines the amount of network traffic due to the traditional ARP scheme and the proposed *PrECast*.

As we mentioned before, ARP broadcast is applicable only to a single IP broadcast domain. We assume the underlying network topology in which *PrECast* is deployed follows Cisco's recommended three-tier architecture. We assume that there are n active hosts in the network. These include PCs, mobile hosts, printers, fax machines, etc. Our LAN has m *access layer* switches and a *distribution layer* switch. The hosts are connected to the *access layer* switches, and the *access layer* switches are connected to the *distribution layer* switch.

Even though access layer switches and the distribution layer switch can be connected in any fashion to offer redundancy and fault tolerance, they eventually form a *tree* architecture to avoid any loop in the network. Thus, the entire topology can be viewed as a *rooted-tree* with $(n + m + 1)$ vertices and $(n + m)$ edges. The *root* of the tree is the distribution layer switch. The m access layer switches are the children of the distribution layer switch, and the hosts form *leaf* nodes. A pictorial representation depicting this fact is given in Figure 6.4.

Whenever a host broadcast a new *ARP request*, the request needs to be forwarded through every active port of the access and the distribution layer switches. Thus, a new *ARP request* traverses through every edge of the tree. Since there are n active hosts in the network, the communication message complexity for the traditional ARP is $O(n\times(n+m))$ $= O(n^2 + nm)$.

In *PrECast*, let a host issue a new *ARP-request* for a destination. If the destination's <MAC, IP >binding is not available in the *MCAM* table of the host's switch, the switch will multicast a request to the remaining m switches. Thus, in this case, the multicast request traverses only through m-edges connecting $(m + 1)$ switches. Thus, the worst case message complexity is $O(nm)$.

Even though, the worst case complexity of *PrECast* is $O(nm)$, in reality, *PrECast* issues a smaller number of multicast requests. This is because, at the startup, a switch has more <MAC, IP >bindings stored in its *MCAM* table than an end-host, which knows nothing in a traditional ARP scheme.

As we can observe, *PrECast* has a savings of $O(n^2)$ messages, even in the worst case scenario. The savings is significant in terms of bandwidth, host CPU and other performance resources in a medium to a large-scale network.

6.5.2 Convergence Analysis

One of the optimization schemes available to reduce the number of ARP broadcast requests is to cache the recent <MAC, IP >bindings. If the MAC address of the recently-received *ARP-reply* is needed at a later point in time, it can be retrieved from the cache. According to the standard, all active hosts on the network hearing an *ARP broadcast* from Host A should update their own ARP cache entry for Host A. Even though this feature potentially reduces the number of ARP broadcasts in the network, it poses a serious security problem. Thus, several operating systems do not support this feature. Furthermore, any unsolicited ARP reply will be silently dropped.

In the traditional ARP process, the ARP cache table is empty for every host during the startup. Every time a host issues an ARP resolution, it adds an entry to the ARP cache table. There are no more ARP broadcasts in the network under the following two conditions:

1. Every host knows about the MAC address of all other hosts in the network.

2. All hosts know about the MAC address of other frequently-communicating hosts in the network.

Condition (1) is a stronger condition and includes Condition (2). There is no need for every host to know about all other hosts in the network. Thus, the ARP convergence mostly depends on Condition (2). For the completeness of the work, we analyse both of these conditions.

Let us assume that there are r frequently-used hosts in the network (like default gateway, DNS server, and so on). Let an arbitrary host A need the <MAC, IP >binding for Host B. Host A will issue an ARP broadcast, only when the MAC address is not available in A's ARP cache table. Thus, the probability of Host A issuing an ARP broadcast for Host B is $(n-1-t)/(n-1)$, where t is the number of ARP entries present in A's *ARP-cache* table at the time of a request. Initially, when A's *ARP-cache* table is empty, this probability is one. When there is only one host in the network not known to A, this probability is $1/(n-1)$. When the remaining $n-1$ host's <MAC,IP >bindings are available with a host, then this probability is zero. After this stage, Host A will not issue any *ARP broadcast.*

The convergence time is the time it takes for this probability to change from one to either $1/(n-1)$ or $(n-r)/(n-1)$, depending on whether we are interested in Condition (1) or (2). The quickest convergence happens when every ARP resolution results in a broadcast message and a new entry is added to the ARP cache table of a host.

In order to keep the *ARP-cache* fresh, it has a *time-out* value. The *time-out* process is discussed in the later part of this chapter. In the absence of any *time-out:*

1. The quickest time in which Host A knows about the remaining $(n-1)$ hosts in the network is $(n-1)/\lambda$, where λ is the arrival rate of the *ARP-requests* with the probability $\prod_{i=1}^{n-1}(\frac{n-i}{n-1})$.

2. The quickest time in which Host A knows about all the frequently-used r hosts in the network is r/λ with the probability $\prod_{i=1}^{r}(\frac{n-i}{n-1})$.

We now analyse the convergence of *PrECast.* We assume that there are n_1, n_2, $\cdots$, n_m number of hosts connected with each access-layer switch, respectively. We analyse the quickest convergence of *PrECast* through *edge covering* for graphs. Before we establish a bijective relation between *PrECast* and edge covering, we define some of the standard graph-theoretic terminologies used in this paper.

A *graph* G consists of a finite non-empty set $V = V(G)$ of *vertices* together with a set $E = E(G)$ of unordered pairs of distinct vertices of V. The pair $e = \{u, v\}$ of vertices in E is called an *edge* of G. We also write an edge $e = \{u, v\}$ as $e = uv$.

If $e = uv \in E$, then u and v are called *adjacent vertices* and e is *incident* with each of its two vertices u and v.

If every pair of vertices of a graph is adjacent, it is then called a ***complete*** graph. A complete graph on n vertices is denoted by K_n.

A graph $G = (V, E)$ is said to be an r-partite graph (where r is a positive integer greater than one) if its vertex set can be partitioned as $V = V_1 \cup V_2 \cup \cdots \cup V_r$, such that if uv is an edge of G, then u is in some V_i and $v \in V_j$ for some $i \neq j$. If $uv \in E$ for every $u \in V_i$ and $v \in V_j$ for $i \neq j$, then G is called the complete r-partite graph. It is denoted by $K_{n_1,n_2,\cdots,n_r}$, where $\mid V_i \mid = n_i$.

A subset M of the edge set E of a graph $G = (V, E)$ is an ***independent edge set*** or ***matching*** in G, if no two edges of M have a common vertex. A matching M is ***maximal*** if there is no other matching M' such that $\mid M' \mid \geq \mid M \mid$. A maximal matching M is called a ***maximum matching*** if M has the maximum number of edges than any other matching.

Let $G = (V,E)$ be a graph. A set $S \subseteq E$ is called an edge cover of G if every vertex of G is incident with at least one edge of S. S is called a ***minimum edge cover*** of G if $\mid S \mid$ is of minimum size among all edge covering of G.

Standard graph-theoretic terms not defined here can be found in Bondy and Murty [7].

As we mentioned before, if two hosts A and B are connected to the same switch that implements the *PrECast* protocol, then the ARP resolution issued by A for B will be handled by the switch itself. However, if B is connected to another switch and the <MAC, IP >binding for B is not available with the switch to which A is connected, the *ARP request* will be multicasted. Let T be the topology that consists of a distribution layer switch connected with m access layer switches S_1, S_2, $\cdots$, S_m. Let n_1, n_2, $\cdots, n_m$ be the number of hosts connected with these switches, respectively. Without loss of generality, we assume that $n_1 \leq n_2 \cdots \leq n_m$.

For the given topology T running *PrECast*, we construct a graph $G = (V, E)$ as follows: The set of vertices of G comprises the hosts in the network. If $uv \in G$ are connected in G if and only if the *ARP resolution* from u to v may result in a multicast, the necessary and sufficient condition for this to happen is that u and v must be connected to a different switch. Since this relation is symmetric, the resultant graph is an undirected graph. It is easy to see that the resultant graph is the complete m-partite graph $K_{n_1,n_2,\cdots,n_m}$. For the rest of this section, T and G denote the above topologies.

The following theorem establishes a relation between an edge covering of G and how all switches in T learn about the MAC address of all hosts in the network.

Theorem 3 *Let $S \subseteq E(G)$. S is an edge cover of G if and only if all the switches in the network learn the <MAC, IP >bindings of every hosts in the network through ARP resolution either from host u to v or from host v to u for all $uv \in S$.*

Proof Let a host u be connected to a switch S. For other switches in the network to know the MAC address of u, either there must be an *ARP request* from u or to u that resulted in a multicast among switches. If there are r-pairs of hosts u_i, v_i through which all switches learn the MAC address of every host in the network, then $S = \{u_iv_i \mid 1 \leq i \leq r\}$ is an edge cover of G.

Conversely, let $S \subseteq E(G)$ be an edge cover of G. Let u be any arbitrary vertex of G. We can then find a $v \in V(G)$ such that $uv \in S$. By the definition of G, the hosts representing u and v are not connected with the same switch. Thus, an ARP resolution either from u or from v towards the other host will result in a multicast. Based on this multicast, all switches in the network learn about the MAC address of both u and v.

The following corollary is immediate from the above theorem.

Corollary 1 *Let $S \subseteq E(G)$ be a minimum edge cover of G and $k = \mid S \mid$. Then, exactly k ARP resolutions are needed in T for all switches in the network to learn the <MAC, IP >binding of every host in the network.*

We now focus our attention towards finding the minimum edge cover number for a complete m-partite graph.

The following theorem is true for any graph without any isolated vertex.

Theorem 4 *Let a graph G have no isolated vertices. Then, the cardinality of the minimum edge cover C of G is equal to the cardinality of the maximum matching M of G increased by $\mid V(G) \mid -2\times \mid M \mid$.*

Proof Let M be a maximum matching of G. Then, M saturates $2\times \mid M \mid$ vertices. Thus, there are $\mid V(G) \mid -2\times \mid M \mid$ unsaturated vertices of G. Since M is a maximum matching, the set of M-unsaturated vertices of G forms an independent set. The maximum matching M can be extended to a minimum edge cover C of G by choosing an edge from $E(G) - M$ for every M-unsaturated vertex. Thus, $\mid C \mid = \mid M \mid + \mid V(G) \mid -2\times \mid M \mid$.

The following two theorems are due to Sitton [77], who calculates the size of the maximum matching for a complete m-partite graph.

Theorem 5 *Let $K_{m_1,m_2,\cdots,m_n}$ be a complete multipartite graph with m_i vertices in the i-th part labelled so that $m_1 \leq m_2 \leq \cdots \leq m_n$. If $m_n \geq m_1 + m_2 + \cdots + m_{n-1}$, then:*

1. *The number of edges in any maximum matching M is $m_1 + m_2 + \cdots + m_{n-1}$.*
2. *The maximum matching is obtained by connecting all vertices in the parts of $m_1, m_2, \cdots, m_{n-1}$ to m_n.*

Theorem 6 *Given any complete multipartite graph $K_{m_1,m_2,\cdots,m_n}$ with $m_1 \leq m_2 \leq \cdots \leq m_n$ and $m_n < m_1 + m_2 + \cdots + m_{n-1}$, then any maximum matching will have $\lfloor n/2 \rfloor$ edges, where $n = m_1 + m_2 + \cdots + m_{n-1} + m_n$.*

Based on all the above theorems, we have the following results:

Theorem 7 *Let $G = K_{n_1,n_2,\cdots,n_m}$ be a complete multipartite graph with n_i vertices in the i-th part labelled so that $n_1 \leq n_2 \leq \cdots \leq n_m$. If $n_m \geq n_1 + n_2 + \cdots + n_{m-1}$, then the number of edges in the minimum edge cover S is n_m.*

Proof: Let M be a maximum matching. Then, from Theorem 6, $\mid M \mid = (n_1 + n_2 + \cdots + n_{m-1})$. Let S be a minimum edge covering for G. From Theorem 5, $\mid S \mid = \mid M \mid +$ $\mid V \mid$ - $2\times \mid M \mid = \mid V \mid - \mid M \mid$. That is, $\mid S \mid = (n_1+n_2+\cdots+n_m)$ - $(n_1+n_2+\cdots+n_{m-1}$ $) = n_m$.

Theorem 8 *Let G be the complete multipartite graph defined above. If* $n_m < n_1 + n_2 + \cdots + n_{m-1}$, *then the number of edges in the minimum edge cover of G is* $\lceil n/2 \rceil$, *where n* $= n_1 + n_2 + \cdots + n_m$.

Proof Let M be a maximum matching and S be a minimum edge covering for G. From Theorem 7, $\mid M \mid = \lfloor n/2 \rfloor$. We prove this theorem based on whether the G has n odd or even number of vertices.

Case 1: . Let n be an even number, and $n = 2k$.

Then, $\mid M \mid = k$. Thus, $\mid S \mid = k$ + $2k$ - $2k = k$. In this case, the maximum matching itself is a minimum edge covering for G.

Case 2: Let n be an odd number, and $n = 2k + 1$.

Then, $\mid M \mid = k$. Thus, $\mid S \mid = k + (2k + 1)$ - $2k = k + 1 = \lceil n/2 \rceil$. In this case, there is only one vertex of G that is not saturated by the maximum matching M. Thus, S contain edges of M along with one edge that is incident with the M-unsaturated vertex of G.

Based on Theorems 7 and 8, we can precisely estimate how soon *PrECast* converges depending on how many hosts are connected to each access layer switch. Let λ be the arrival rate of *ARP requests* in the network. Then *PrECast* converges in n_m/λ or $\lceil n/2 \rceil/\lambda$ time depending on whether $n_m \geq n_1 + n_2 + \cdots + n_{m-1}$ or not.

6.5.3 Simulation

In this subsection, we present the message complexity and convergence analysis through simulation. We wrote a discrete event simulator using *C* language. We implemented the traditional ARP protocol and *PrECast* for comparison. We set the *ARP arrival rate* to be two packets per minute for all hosts in the network. We considered a topology of having five access layer switches connected with a distribution layer switch. There are 20 hosts connected with every access layer switch. Our simulation topology is given in Figure 6.5. We ran the experiment for 60 min and collected the statistics on message complexity. We repeated the experiment 100 times and calculated the average to remove any simulation artefacts.

In our simulation, we considered the ARP protocol with two different types of ARP-cache clearance processes. In the first type, all entries in the ARP cache table are cleared

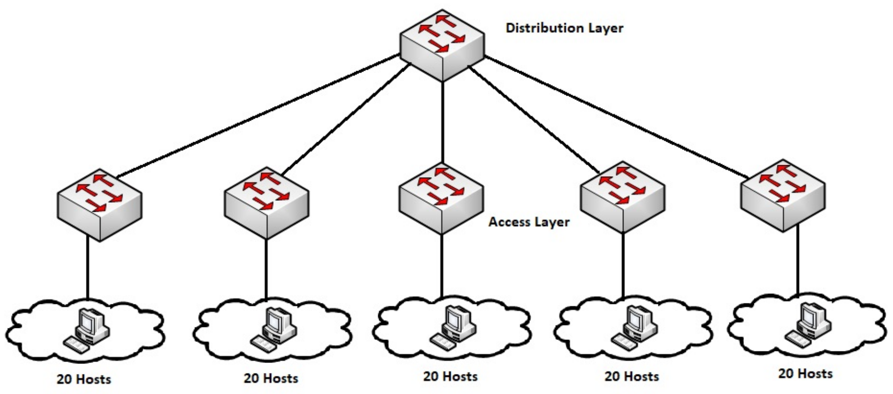


Figure 6.5: Simulation topology.

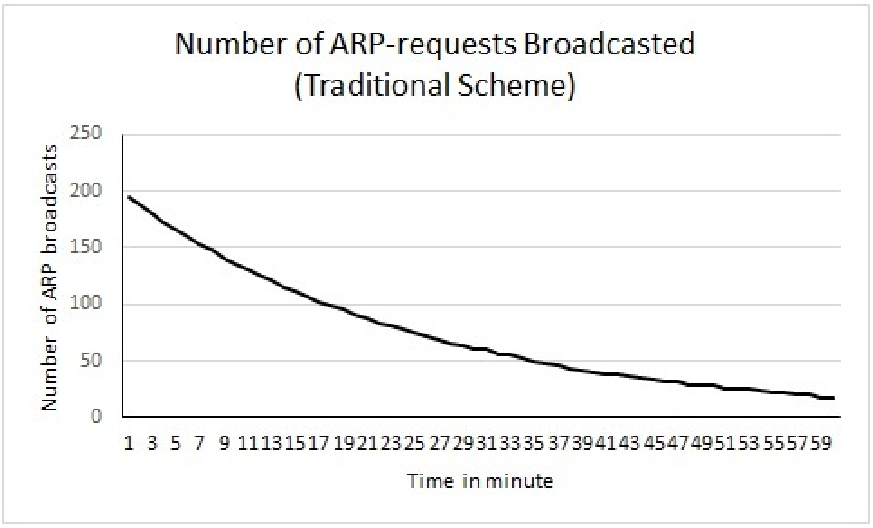


Figure 6.6: Traditional ARP process with no time-out.

after the time-out. In this case, if the host needs the MAC address of a recently-used host, it needs to initiate the ARP resolution again. The second ARP-cache clearance scheme only clears the ARP-table entries that were not used until the time-out period. All recently-used entries are kept intact. For more detail on ARP-cache algorithms, please refer to [76]. The performance graphs are presented in Figures 6.6–6.9. It can clearly be seen that the *PrECast* protocol outperforms the conventional ARP protocol.

6.6 Analysis of PrECast

In this section, we analyse the strength of our proposed architecture by comparing with the existing solution available in the literature.

- From Chapter 3, it is apparent that broadcasting in IPv4 is responsible for several security problems. However, in the literature, every proposal only addresses a

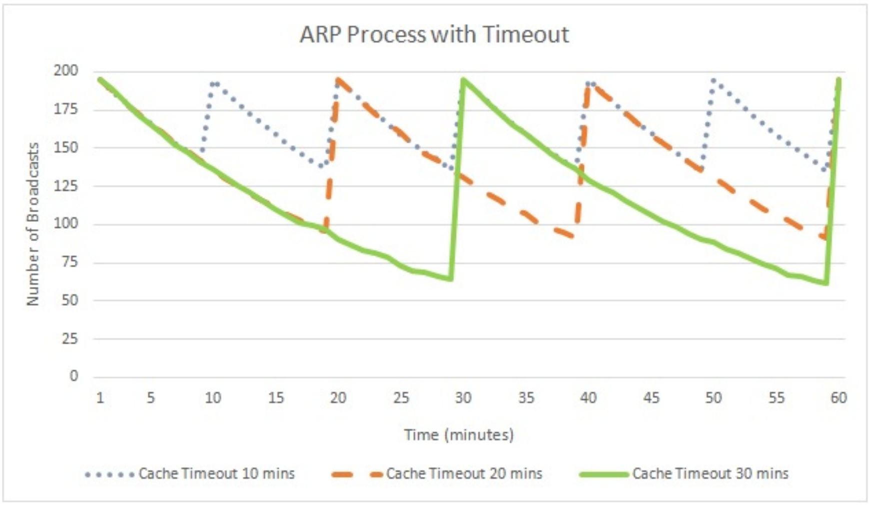


Figure 6.7: ARP process with time-out = 10, 20 and 30 min (Type 1).

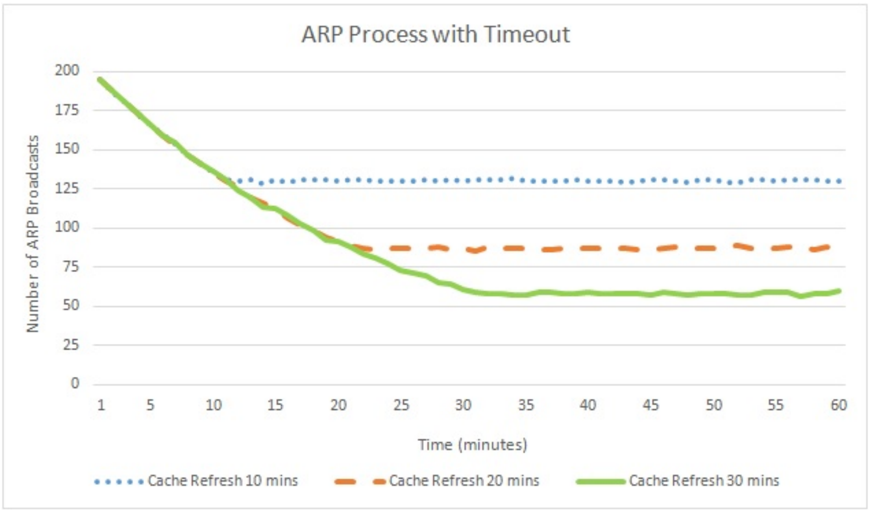


Figure 6.8: ARP process with time-out = 10, 20 and 30 min (Type 2).

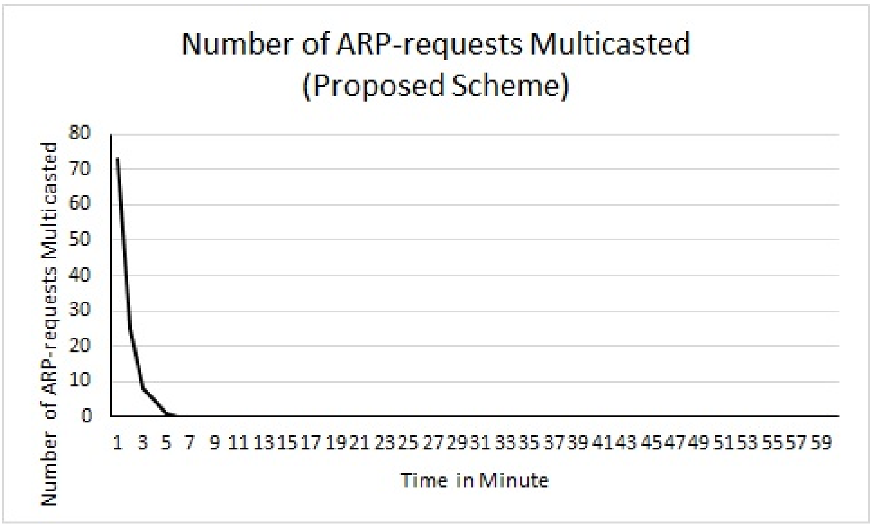


Figure 6.9: Number of ARP multicasts in PrECast.

particular type of security pitfall like *ARP-poison*, *DHCP-snooping*, etc. One type of solution cannot be modified to provide a solution to another type of attack. However, our holistic approach provides a solution to all security pitfalls that are due to IPv4 broadcasting.

- Several proposals that are available in the literature use cryptographic schemes, the key distribution centre and a pre-defined relation between a host and the network, etc. Firstly, the use of cryptographic schemes may not be efficient for a mobile host due to its CPU and power requirements.

- For a solution that uses a *key distribution centre* (KDC), the *KDC* will be a single-point attack. Thus, any *DoS* attack towards *KDC* can bring the entire network operation down.

- A pre-defined relation between a host and a network (like only the registered MAC addresses' area is allowed to connect to the network) can be imposed on a small private network. However, this relation cannot be extended in a public network like an airport or university network, which encourages the concept of *BYOD*.

  Our proposed solution does not use cryptographic schemes, thus being power- and CPU-friendly on mobile devices. Since there is no *KDC* involved in our solution, it is not a single-point failure. Further, since there is no need to establish a predefined relation between a host and a network, our solution not only supports *BYOD*, but also can scale from a smaller to a larger network.

- **Message overhead:** Several proposed solutions in the literature require the use of control messages, thus incurring more overhead. However, our proposed protocol did not incur any extra overhead.

- **Transparent implementation:** If any implemented solution is known to an end-user/attacker, potentially, the solution may be targeted for attacks. However, our proposed solution may be implemented transparently without end-user's knowledge.
- Our proposed solution requires fewer configurations at the core network, compared with other crypto-based schemes that require and maintain the installation of a *KDC*.
- Our proposed solution requires zero-configuration at the host side. A host may not be aware of the existence of such a solution. Thus, any hacking attempt exploiting IP broadcasting is recorded by our system, and the network administrator can take immediate action against a host with malicious intention.
- In our proposal, we used message complexity and convergence time as our performance parameters. Since, we have not changed the state-diagram of the protocol, the CPU load at the switches, end-devices and the *RTT* (round-trip time) is the same as without our protocol.
- Our solution is scalable; since our proposed solution runs as a service inside a switch, it can be implemented on any switch that runs through a network operating system. It can also be implemented using the SDN (software-defined network) architecture.

6.7 Chapter Summary

The proposed *PrECast* framework eliminates several LAN-based attacks that are due to poisoning and DHCP based attacks. Poisoning attacks result in MITM attacks. To launch these attacks, the attacker must be a member of a corporate network. Personal benefit is the primary motivation for these attacks. However, the *PrECast* infrastructure does not eliminate a malicious insider. The following attacks are still possible through malicious inside hosts.

A malicious host inside a corporate network may spoof the IP address of another host either within the corporate network or outside the network. This may result in various spoofing related attacks mentioned in Chapter 3.

The corporate end-hosts may be exposed to the outside network (if these hosts are given public IP addresses). This may results in attacks such as *DDoS* attacks such as *Smurf*, *Fraggle* and *DNSamplification*.

In Chapter 7, we provide a solution to address the above two problems. This solution is built on top of the *PrECast* infrastructure.

Chapter 7

NAT++: An efficient micro-nat architecture for solving ip-spoofing attacks in a corporate network

The Internet Protocol (IP) version 4 (IPv4) has several known vulnerabilities. One of the main vulnerabilities is that the protocol does not validate the correctness of the source address carried in an IP packet. Users with malicious intentions may take advantage of this vulnerability and launch various attacks against a target host or a network. These attacks are popularly known as ***IP Address Spoofing attacks***. One of the classical IP-spoofing attacks that cost several million dollars worldwide is the *DNS-amplification attack*. Currently, the availability of solutions is limited, proprietary, expensive, and requires expertise. The Internet is subjected to several other forms of amplification attacks happening every day. Even though IP-Spoofing is one of the well-researched areas since 2005, there is no holistic solution available to solve this problem from the root. Also, every solution assumes that the attackers are always from outside networks. In this chapter, we provide an efficient and scalable solution to solve the IP-Spoofing problem that arises from malicious or compromised inside hosts. We use a modified form of ***Network Address Translation*** (NAT) to build our solution framework. We call our framework as ***NAT++***. The proposed infrastructure is robust, crypto-free, and easy to implement. Our simulation results have shown that the proposed *NAT++* infrastructure does not consume more than the resources required by a simple NAT.

In Chapter 7, we provided an efficient micro-natting architecture for solving IP-spoofing attacks in a corporate network. Most of this chapter contents are derived from the candidate's publication [93].

7.1 The Prelude

IPv4 has several known vulnerabilities. One of the important vulnerabilities is that the protocol does not validate the correctness of the source address carried in an IP packet. Users with malicious intentions may take advantage of this vulnerability and launch various attacks against a target host or a network. This attack is popularly known as *IP-Address Spoofing* or simply *IP Spoofing.* IP address spoofing is the creation of IP packets with a *false source IP addresses* to impersonate another host or a network device. The false IP address may belong to the local subnet or a remote network. The IP address spoofing is primarily employed to launch a ***Distributed Denial of Service*** (DDoS) attack. Packets with spoofed IP addresses are more difficult to filter in the target network since each spoofed IP packet appears to come from a legitimate IP address. It is estimated that individual DDoS attacks can cost an organization up to US$ 50,000 [39]. However, due to these attacks, large enterprises not only lose substantial money but also their credibility and lose their market share [16, 49]. Thus, protecting a network from these attacks is an utmost priority.

IP Spoofing is one of the well-studied areas since year 2005. There are several research articles and commercial software systems available to protect a network from IP address spoofing attacks. However, until today, no network is immune to IP Spoofing attacks. IP Spoofing is not only exclusive to the IPv4 network but also applicable to the IPv6 network in the absence of *IPSec.* Even though there are several research articles and commercial products available in the literature to stop this attack, most of the article reviewed by us assumes that the attack originated from an external network. None of the research work known to us in the literature address IP spoofing attacks that originate internally to the network either through a malicious insider host, internal nodes that colluded with external hosts, or compromised internal nodes. This form a strong motivation for this work.

This chapter addresses the IP-Spoofing problem that arises from the internal network. This consists of two scenarios: an inside host that spoofs an IP address that belongs to the same corporate network; an inside host that spoofs an IP address that belongs to an external network. In both scenarios, we provide an efficient and scalable solution to stop the IP spoofing problem. Our solution uses a modified form a Network Address Translation (NAT) architecture. The NAT was originally invented to address the shortage of IPv4 address space. However, in this chapter, we use NAT only to hide the network and hosts from the external network. We call our proposed system as ***NAT++***. *NAT++* does not need any modification to the protocol stack at the client-side. The *NAT++* requires that every corporate switch support Layer-3 and above and can run and maintain the other services of the *NAT++* solution.

In Chapter 6, we introduce the *PrECast* infrastructure for a corporate network that solves various poisoning problem that leads to the Man-in-the-middle (MITM) attack in a Local Area Network (LAN). The *NAT++* solution is implemented on top of the *PrECast* infrastructure to provide robust security.

The following are the advantages of the proposed *NAT++* system:

1. *NAT++* effectively eliminates several important network attacks that are due to IP-address Spoofing.

2. Due to the distributed nature of our design, *NAT++* consumes very less system and network resources. This is evident from our simulation.

3. Our solution is modular and scalable. Thus, we may able to offer a solution to unknown attacks in the future without much of a redesign.

4. *NAT++* does not use any crypto-algorithms and it does not need any modification to the protocol suite. Thus, our scheme does not violate any protocol standards.

5. As an indirect benefit, *NAT++* controls the network diameter whereby avoiding chaining at the access layer.

7.2 The State-of-Art in Address Spoofing

In this section, we review some of the key works available in this area. Network communication involves two types of addresses. One is the Data link-layer address called the *Media Access Control Address* or simply the MAC address. This address is also sometimes referred to as a *Physical address*. MAC address is a unique identifier assigned to a *Network Interface Controller* (NIC). MAC addresses are primarily assigned by device manufacturers and are referred to as Hardware address. MAC address is stored in hardware, such as the network card's read-only memory, or by a firmware mechanism. Until a few years ago, MAC addresses are hardcoded in a ROM and cannot be changed. For all new generations of NIC cards, the default MAC address can be changed temporarily through software. Malicious users may take advantage of this feature and launch MAC-address spoofing attacks. Since the focus of this thesis is at the IP-layer, we will not focus on this type of attack.

The second address is the *IP-address*, called the *logical address*. IP addresses are assigned to hosts either manually or dynamically through a Dynamic Host Configuration Protocol (DHCP). Communication between any two hosts on the Internet requires the use of source IP address and the destination IP address. The IP protocol (both v4 and v6) does not validate the correctness of the source address carried in an IP-packet. Only the destination IP address is needed for routing the IP packet from the source to the destination. The source IP address is used only by the destination host for sending the reply. In the absence of any validation, IP address spoofing involves the creation of IP packets with a false source IP addresses to impersonate other hosts or network devices in a network. The malicious intent may be the main motive behind IP address spoofing. IP address spoofing is used primarily to launch DDoS attacks. IP address spoofing is also effectively employed whenever there is a trust-relation that exists between hosts in a corporate network. In a corporate network, it is common that one or more servers may establish inherent trust relations among them. If a user successfully logged into one machine, then they can automatically connect with other machines without the requirement for a *username* and *password*. By spoofing a connection from a trusted machine, an attacker on the same network may be able to access the target machine without authentication. There are numerous solutions available to thwart this attack. Most of them use some form of digital certificates and the use of encryption. Kerberos is an example of such a system [51].

The majority of IP spoofing will result in a DDoS attack [87, 47]. Whenever an IP packet carries an impersonated IP address, it is termed as a “spoofed IP packet”. A spoofed IP packet does not cause any harm to the path routers; rather the intention is to only harm the victim or his network. The research community started working on combating IP spoofing as early as 2000 [64, 5]. The early work involves IP filtering at the ingress and the egress routers. Research on IP spoofing can be classified into broadly three categories namely:

- Identifying spoofing packets
- Mitigating spoofing attacks
- Pinpointing an attacker’s real location

A classical survey on the above topics is due to Ehrenkranz and Li [28]. After this work, unfortunately, there are no up to date surveys available on this topic. This may form a motivation for our future work. In a recent paper [56], Lichtblau et al. proposed a detection, classification, and analysis of inter-domain traffic with a spoofed IP address. Their work involves filtering at the border routers based on known IP ranges used inside the network. Since the backbone speed is increasing day-by-day, filtering at the border router may introduce unwanted delays. Also, Cisco’s network design [63] recommends no filtering done at the Core-layer of the network. This solution may stop an inside host in spoofing an IP address that is external to the network. However, this solution will fail to detect an inside host spoofing an address that belongs to the same corporate network.

In 2007, Wang et al. [95] proposed a hop-by-hop filtering mechanism for mitigating IP Spoofing. In the absence of any incentive for path routers to perform packet filtering, this proposal works well only in the ingress network. Hop-by-hop filtering is better in terms of detecting spoofing than the algorithm presented in [56]. It can detect a host spoofing an IP address that belongs to another subnet, but not on the same subnet.

In [59], Mirkovic et al. proposed a self-learning algorithm for filtering spoofed packet in the upstream traffic. Their algorithm learns to detect spoofed traffic upstream from the victim by combining information about the traffic's expected route and the sender's response to a few packet drops. Even though their algorithm may work well in a lighter load, on a heavy DDoS attack, detection may not be of any use; rather immediate mitigation strategies are needed. This algorithm exhibits the same weakness as that of [56] in detecting internal IP address spoofing.

The lack of authentication in the Internet's data plane facilitates IP spoofing attacks. Fonseca et al. [31] proposed a method to establish a hop-by-hop authentication mechanism to track down the source of spoofed IP packets. Tracking down the source may thwart further attacks from the same attacker. However, this scheme does not effectively eliminate IP spoofing. In addition to this, establishing authentication in the data plane is difficult due to the amount of workload in processing individual IP packets (or IP flows). This algorithm behaves like [95] in detecting IP spoofing attacks.

The papers such as [56, 95, 59, 31] and the articles reviewed in the survey [28], the authors assume that no host spoof an IP address that belong to the same IP subnet. This deficiency forms a strong motivation in providing our *NAT++* framework.

Broadcasting is one of the essential features of the IPv4 protocol. This is replaced with multicasting in the IPv6 protocol. Two essential services, namely the Address Resolution Protocol (ARP) and the DHCP are built based on IPv4 broadcast services. To launch a DDoS attack, IP spoofing must be combined with the IPv4 broadcasting feature. IP spoofing alone may not result in a DDoS attack [87, 47].

The following are some of the important classes of attacks that are due to the combination of the broadcast feature in the IPv4 protocol and IP-address spoofing. These attacks are classified under "Amplification Attacks". The details are presented in Chapter 3

- The Smurf Attack
- The Fraggle attack
- The DNS-Amplification attack

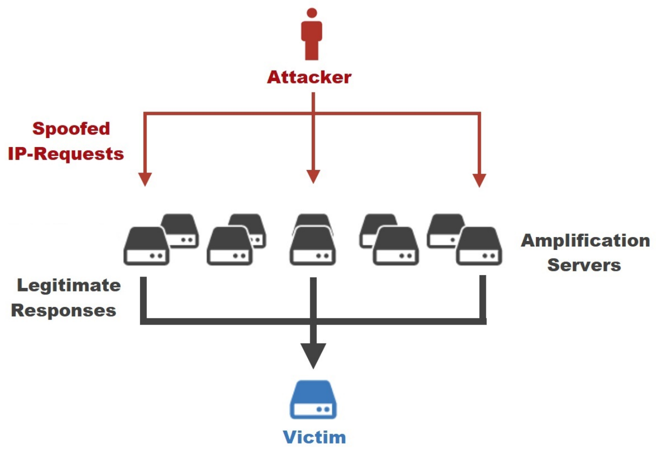


Figure 7.1: Typical DoS attack.

There are other amplification attacks such as *Network Time Protocol* (NTP), The *Simple Service Discovery Protocol* (SSDP) and The *Simple Mail Transfer Protocol* (SMTP) amplification are possible by exploiting the vulnerabilities of the IPv4 protocol. A detailed description of these attacks is presented in [38].

For all the above-mentioned attacks, botnets [67] may be used as an attacker to increase the amplitude of these attacks. DNS-amplification attack is hard to detect as they appear to originate from valid servers and targeted towards a legitimate host. For Smurf, Fraggle, and DNS-amplification attacks, the attacker can be anywhere on the Internet.

Figure 3.2 provides a taxonomy of attacks that exploits the IP-broadcasting and IP-Spoofing vulnerabilities in the IPv4 protocol. Figure 7.1 depicts a mechanism through which an attacker launch DoS attacks mentioned in the taxonomy. Here an attacker spends only a small amount of resources (such as bandwidth or CPU) to cause a significantly greater number or higher level of target resources to malfunction or fail through amplification.

7.2.1 Port Scanning Attack

This attack is different from the two types of attacks (spoofing and poisoning) described above. This attack is not launched by exploiting any vulnerabilities in the Internet protocol. Every cyberattack starts with the port scan attack, whereby the attackers scan the target network for any potential vulnerabilities or running unpatched services.

Network Address Translation (NAT) provides a seamless solution to all the attacks mentioned above. This is by hiding the campus hosts from an external network. We describe the strengths and weaknesses of NAT in terms of security in the next section.

7.3 Network Address Translation (NAT)

NAT [73] is a method of remapping one IP address space into another by modifying the address information in the IP packet header while they are in transit. NAT was originally invented to address IPv4 address depletion and scaling in routing [73]. The address reuse solution is to place NAT at the border of stub domains. Each NAT box has a table consisting of pairs of local IP addresses and globally unique addresses. The IP addresses inside the stub domain are not globally unique and thus can be reused. The main advantage of NAT is that it can be installed without any changes to routers or hosts.

Even though NAT was originally designed to address the IPv4 address depletion problem, it provides some level of security to end-hosts. A NAT box hides the internal hosts from external networks. However, the traditional security mechanisms such as firewalls, Intrusion Detection System and Intrusion Prevention System cannot be ignored. By hiding the host's IP address, NAT provides the following advantages:

1. **DoS protection:** This is because the host's private addresses are not known to public hosts. Using the same reasoning, NAT protects infrastructure.
2. NAT can be used to effectively limit the use of protocols and ports.
3. NAT can eliminate network and port scanning done by rogue devices from the Internet.
4. NAT blocks unsolicited connections that are initiated from an external network (external to the NAT realm)

Even though NAT has the above advantages, it has the following disadvantages:

- NAT box itself is a bottleneck. Attackers may launch DoS attacks against NAT boxes that will stop the entire network operation.
- Since NAT limit the use of protocols and ports, it is hard to detect any potential malware that infected the inside network. This is because; no external network can detect malware's heartbeat.

The following property made us consider NAT as a potential candidate for protecting a corporate against an IP-Spoofing attack. Whenever a connection is made from a host in a private realm to a public realm, for every communication, an entry is created in a NAT look-up table based on $<$ Protocol, Source IP, Source Port, destination IP, destination port $>$, where Destination IP and port belong to a public realm. Replies arriving from

$<$ Destination IP, port, protocol $>$ are forwarded to inside hosts. If there are no matching entries, packets are silently dropped. Thus, any unsolicited replies originating from the public realm are blocked. This important security feature is used along with the *PrECast* infrastructure to provide robust security in a corporate network.

7.3.1 Attack against NAT Boxes

Traditionally NAT boxes are deployed at the gateway of an IP subnet (that uses a private realm). This IP subnet may represent a building, department, or unit in a corporation. It may consist of several hosts ranging from 20 to 100. When using NAT boxes, all traffic between NATed subnets and the Internet must go through the NAT gateway. For this reason, the network administrator must ensure that NAT boxes have greater throughput and low latency. The NAT boxes should also have fast processors that can examine and translate packets quickly. We now discuss some of the vulnerabilities that are not addressed by NAT. For our discussion, let us assume that M be a malicious host and V be a victim. Both M and V are inside the same NATed network.

1. Since NAT boxes hide inside the network, it protects the inside network from DoS. However, consider the following scenario:

   (a) M spoofs V's private IP address and create a Smurf or a Fraggle attack through an external amplifier. The NAT box will faithfully create a forwarding table and forward it to the amplifier network. The destination field in the forwarding table is the broadcast address of the amplifier network. One spoofed IP packet from M will result in 1000 s of packets from the amplifier network targeted towards V. The NAT box will keep dropping them after checking the source IP address (no match with individual host addresses from the amplifier network). Thus, this is effectively a DoS attack targeted towards the NAT box. Since the NAT box is processing all the incoming packets, it may not able to perform its translational task.

   (b) Whenever M spoofs V's IP address and launch a DNS-amplification attack through compromised DNS servers, all the incoming packets are faithfully translated and forwarded towards the victim V, as the there is a match between the source and the destination IP in the forwarding table. Thus, the DNS-amplification DoS attack is still possible in a NATed network if the actor node is inside the NATed network.

2. NAT effectively stops port scanning attacks originating from the Internet. However, it cannot protect the network from a port scanning attack originating from the

inside network. As before, we also demonstrate that if there is a malicious actor node V inside the NATed network, he can open up the port for an external host to come through. Let M_1 be a malicious inside host, M_2 be an outside host and V be the victim host. Let us assume that M_2 wishes to connect to V on port P. Under normal circumstances, this is not possible in a NATed network. Any request to V on port P will be silently dropped by the NAT box as there was no forwarding entry. A forwarding entry may be created with the help of M_1. As a first step, M_1 sends a spoofed IP packet with V's IP address and P as its source port to M_2. NAT box will faithfully create a forwarding entry for V towards M_2. Once this spoofed IP packet is received by M_2, the can then connect with port P of his victim V.

Using this method, any malicious inside host may facilitate an external host in performing port scanning or connect to a service.

Thus, NAT by itself does not offer security. However, the interesting feature of NAT is that it hides the inside network from the outside network. In this work, we modify NAT and augment with the *PrECast* infrastructure to offer more robust security for corporate networks.

There are three different types of NAT translations. The majority of NAT boxes map several hosts with private IP realm to one public IP address. This public IP address is assigned to an outgoing interface of the NAT box. The NAT box on that network has a private address in that address space. This type of NAT boxes is known as Many-to-one NAT or a Port Address Translator (PAT). Home gateways are an example of this type of NAT implementation. Even large networks can be connected to the Internet using a single public IP address. This type of NAT may have performance limitations, such as high latency, does not support enough applications, and so on. To address this limitation, we have many-to-many NAT. The Many-to-Many NAT policy allows us to translate a group of addresses into a group of different addresses. Usually, the private realm has more addresses than the public realm. This may sometime result in starvation. To avoid this problem, we have one-to-one NAT. In this type of NAT boxes, one private IP address is mapped on to one public IP address. RFC 2663 [73] refers to this type of NAT as Basic NAT. In this type of NAT, only the IP addresses, IP header checksum, and any higher-level checksums that include the IP address are changed.

In a NAT scheme, if the private IP realm has more than one host, then any arbitrary host can launch IP-spoofing attacks that are mentioned above. Thus, to address a solution, we wish to limit the number of hosts in a private IP realm to one. The only NAT scheme that provides this feature without wasting IP address space is the one-to-one NAT. Also, among the three different types of NAT, one-to-one NAT offers lesser latency and higher throughput. Thus, in our *NAT++* architecture, we prefer to use one-to-one NAT due to various advantages, both in terms of performance and security.

A one-to-one NAT alone does not solve the issues mentioned above. By definition, a one-to-one NAT associates one private IP address to one public IP address. This process does not guarantee that there is exactly one physical host in the private IP realm. Thus, one-to-one NAT alone does not solve the IP-Spoofing problem. *NAT++* is a one-to-one point-to-point micronatting architecture that is compatible with the *PrECast* protocol. We present its detail in Section 7.4.

7.4 The *NAT++* Architecture

In this section, we describe the detailed operation of *NAT++* architecture.

The *NAT++* is deployed on top of the *PrECast* infrastructure defined in Chapter 6.

The *PrECast* protocol operates through the following three phases:

1. Bootstrap Phase
2. Secure multicast tree (called the *PrECast*-tree) construction phase (ref Chapter 5)
3. The Network operation phase

We now explain these phases in brief. We reemphasis these phases here as *NAT++* is integrated through these phases. The detailed operations of the *PrECast* protocol and the construction of a *Secure Multicast Tree* is presented in the respective chapters.

During the bootstrap phase, each switch in the LAN starts creating the *PrECast* table. This table is similar to the MAC-Address table kept in every switch. The *PrECast* table is specific to each switch. This table consists of the following information:

- Port/Switch ID: If a host is connected to a local switch port, this field represents the port-number to which the host is connected. If a host is connected with a remote switch, this field represents the "Switch ID". Each switch in a LAN has a unique Switch ID.
- MAC Address: This field stores the MAC address of the host associated with this Port/Switch ID.
- IP Address: This field stores the IP address of the host associated with this Port/Switch ID.

The heart of the *PrECast* protocol is the secure multicast tree that consists of all campus LAN switches to which hosts are connected. In our earlier chapter (Chapter 5), we presented an efficient algorithm for creating and maintaining a secure multicast tree using a Core-based Tree. This tree is used for distributing broadcast messages between LAN switches securely. We call this tree as a *PrECast-tree*.

The heart of the network communication is the two protocols namely the ARP and the DHCP. During the bootup stage, every host must obtain an IP address to communicate with the rest of the world.

Whenever a host broadcasts a *DHCP-Discover* message, the switch to which the host is connected will note the host's MAC address and unicast the request to the registered DHCP server available in this network. The switch ports to which the requesting host and the DHCP-Server are connected will tunnel the broadcast communication. Once the DHCP handshake is complete, the allocated IP address is mapped against the host's MAC address in the *PrECast* table. By doing this way will eliminate the DHCP-poison problem.

The *PrECast* protocol handles the ARP process much like the conventional process, except that the ARP broadcast requests are multicasted among the switches. The immediate switch to which the requested host is connected will answer the ARP-request based on the information available in its *PrECast* table. If the requested information is not available in its local table, the switch will then multicast the ARP-request to every LAN switches. Eventually, the switch to which the host is connected will "reply". This information is entered into the *PrECast* table by all LAN switches.

We now provide a detailed overview of the *NAT++* architecture and how *NAT++* can be integrated with the *PrECast* infrastructure to eliminate several attacks that are due to the two important classes vulnerabilities mentioned in this chapter. Figure 7.2 represents how *NAT++* is integrated with the *PrECast* infrastructure. The proposed *NAT++* framework addresses the IP-Spoofing problem that arises from internal hosts. However, *NAT++* by itself cannot eliminate the "poisoning problem" in a LAN. Thus, we propose to implement *NAT++* on top of the *PrECast* infrastructure.

The pictorial representation of our proposed architecture is presented in Figure 7.2. Every host is connected to a switch-port. The connection between a host and its switch-port is regarded as a point-to-point connection. A NAT instance is running on every switch-port (as a software service) that translates the host's private IP address to a public IP address. All modern switches support this feature.

7.4.1 The Micro Natting Architecture

In Section 7.3, we presented important advantages of NAT in terms of offering security by hiding inside-network from an external network. However, as we discussed in Sec-

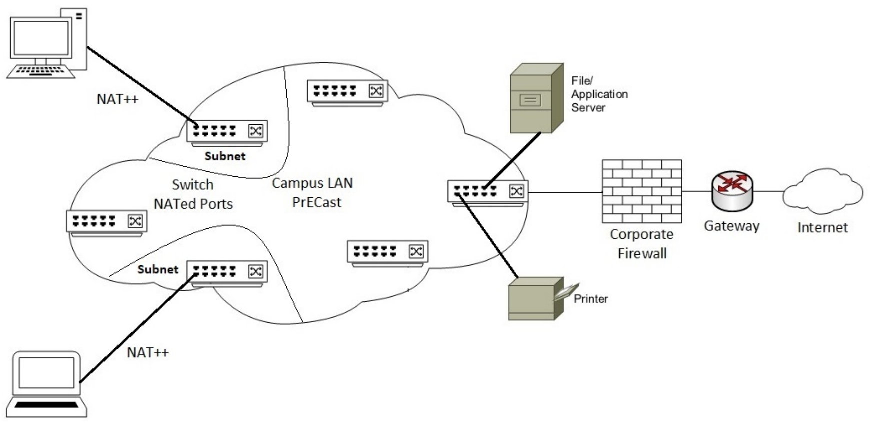


Figure 7.2: Integration of *NAT++* with the *PrECast* Infrastructure

tion 7.3.1, a malicious inside host can open a port, so that an outsider can launch an attack. In our proposed *NAT++* architecture, we consider one-to-one NAT, where each switch port run a NAT engine.

System Bootstrap stage:

During the system setup phase, the network administrator selects one or more LAN switches in the network as the Core switches for the multicast delivery tree. Based on our previous work (ref Chapter 5, all LAN switches form a secure multicast tree. This tree is used for data communication as outlined in our *PrECast* (ref. Chapter 6).

As outlined in Subsection 7.3, every LAN switch stores a table that consists of the following information:

- **Port ID:** This is the port number of the current switch the host is connected. For hosts that are connected with remote switches, this field represents the Switch ID.
- **MAC Address:** This field holds the MAC address of the host connected to this port/Switch ID.
- **IP Address:** IP address of the host that has this MAC address.

In a corporate network, there will be a few hosts that use static IP addresses. For example, DHCP servers, primary and secondary DNS servers, default gateways, file, and print servers. These hosts are connected with certain switch ports. Information about these hosts is manually populated by the network administrators on the respective switches' *PrECast* table. It is important to note that important devices in the network that have static IP addresses will not participate in the *NAT++* process. Thus, the switch ports to which these devices are connected will not run the *NAT++* engine. Since these devices are trusted devices, they will not launch an IP-Spoofing attack. Except for these switch ports, by default, all other switch ports will run the *NAT++* engine.

Every switch port runs a NAT engine. For our discussion, let us assume that P_1 be a switch port of a LAN switch S_1. Let H_1 be a host connected to the port P_1. Then the connection between P_1 and H_1 can be regarded as a point-to-point (P2P) connection. The switch port P_1 also acts as a proxy between H_1 and the rest of the network. All LAN hosts except servers obtain an IP address through DHCP servers. They all participate in the *NAT++* process. If any malicious host tries to connect to the network using a static IP address, the *NAT++* engine will silently drop its IP packets.

We now explain the DHCP-DORA process through our *NAT++* mechanism:

1. **DHCP-Discover Process:** In this step, the host H_1 sends a *DHCP-discover* message. This message is a Layer 2 and 3 broadcast message. Once this message is received by switch port P_1, the switch S_1 issues a modified *DHCP-Discover* message. This is a unicast message sent at Layer-2. The IP source address (IP SRC) is still 0.0.0.0 (as H_1 has not yet obtained any IP address) and the IP destination address (IP DES) is 255.255.255.255. However, the source MAC (MAC SRC) is the MAC address of port P_1 and the destination MAC address (MAC DES) is the MAC address of the DHCP server connected to this network. Rather than issuing a network-wide broadcast, this packet is sent to the DHCP-server as a unicast Layer-2 frame. Doing this way protects the network from the DHCP-poisoning attack in the network.

2. **DHCP-offer process:** When a DHCP-server receives a *DHCP-Discover* message from a client, the DHCP server reserve an IP address (say $A.B.C.D$) and makes a lease offer by sending a *DHCP-OFFER* message to the client. This message contains the client's ID (traditionally a MAC address; in our case, it is the MAC address of port P_1), the IP address that the server is offering (i.e., $A.B.C.D$), the subnet mask, the lease duration, and the IP address of the DHCP server making the offer.

   At Layer-3, the source address is the IP address of the DHCP-server. The destination address is 255.255.255.255. This is a Layer-3 broadcast packet. At the Layer-2 frame, MAC SRC is the MAC address of the DHCP server and the MAC DES is the MAC address of P_1. At Layer-2, it is a unicast frame.

   The received message is translated by the switch S_1 and send to H_1 as follows: In the translated message, the Layer-3 IP-SRC is the IP address of P_1. According to our *NAT++* design, for all switch ports, their IP address is 192.168.0.1 with the subnet mask of 255.255.255.252 and the default gateway of 192.168.0.1. The primary and the secondary DNS address provided by the DHCP server is copied to the translated packet. However, the offered public IP address of $A.B.C.D$ is **not passed** on to H_1; rather a private IP address of 192.168.0.2 is passed on to the host. With our proposed design, **every host** in the network will have the same private IP address 192.168.0.2. For the Layer-2 frame, the MAC-SRC is the MAC address of P_1, and the destination MAC address is the MAC address of H_1.

3. **DHCP-Request Process:** Once the DHCP-offer message is received by H_1, it replies with the DHCP-request process confirming the allocated IP address. For this reply message, the IP-SRC is still 0.0.0.0 and the destination IP address is 255.255.255.255; MAC-SRC is the MAC address of H_1 and the destination MAC address is the MAC address of P_1. As before, this received message is translated by switch S_1.

Once the DHCP server receives this message from Switch S_1, it will then send a *DHCP-ACK* message. Switch S_1 will translate and send it to H_1.

At the end of this DHCP phase, the client H_1 is issued with an IP address 192.168.0.2 with a subnet mask of 255.255.255.252 and the default gateway of 192.68.0.1. This process has segmented the entire LAN into a micro-segment that consists of only P_1 and H_1. Thus, the new broadcast domain consists of only two hosts namely H_1 and P_1. At Layer-3, the connection between P_1 and H_1 are point-to-point.

The port P_1 acquired a public IP address of $A.B.C.D$ from the DHCP server. This address is used for NATting the connection from H_1. Thus, by design, our NAT is a one-to-one NAT and it hides every host from every other host in the network.

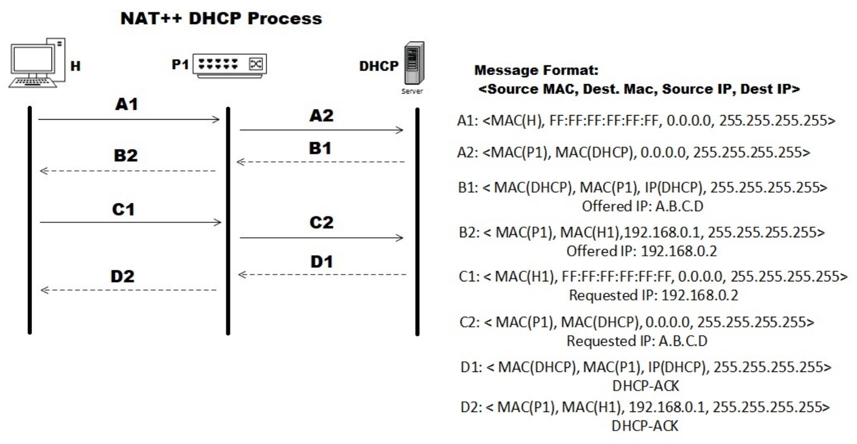
We present the sequence diagram depicting the DHCP address process in Figure 7.3. In this work, we used the private IP network 192.168.0.0/30 for the P2P connection between a host and its switch port. However, this may be replaced with any private IP space with a subnet mask of 255.255.255.252 (or /30).

7.4.2 The Network Operation

In this subsection, we outline some of the essential network operations performed under our proposed design.

ARP-Request and *Reply* are precursors to communication between any two hosts in a network. Thus, we address how *ARP-Request* and *replies* are handled by our *NAT++* protocol.

ARP Request and Reply: From the host point of view, the network is a P2P network connection between the host and the switch port. Thus, ARP-request and replies are not relevant in this network.



```
Message Format:
<Source MAC, Dest. Mac, Source IP, Dest IP>
A1: <MAC(H), FF:FF:FF:FF:FF:FF, 0.0.0.0, 255.255.255.255>
A2: <MAC(P1), MAC(DHCP), 0.0.0.0, 255.255.255.255>
B1: < MAC(DHCP), MAC(P1), IP(DHCP), 255.255.255.255>
        Offered IP: A.B.C.D
B2: < MAC(P1), MAC(H1),192.168.0.1, 255.255.255.255>
        Offered IP: 192.168.0.2
C1: < MAC(H1), FF:FF:FF:FF:FF:FF, 0.0.0.0, 255.255.255.255>
        Requested IP: 192.168.0.2
C2: < MAC(P1), MAC(DHCP), 0.0.0.0, 255.255.255.255>
        Requested IP: A.B.C.D
D1: < MAC(DHCP), MAC(P1), IP(DHCP), 255.255.255.255>
        DHCP-ACK
D2: < MAC(P1), MAC(H1), 192.168.0.1, 255.255.255.255>
        DHCP-ACK
```

Figure 7.3: Sequence Diagram for the DHCP Process.

- **Case 1:** H_1 wishes to communicate with $A_1.B_1.C_1.D_1$ where the destination host is within the campus LAN.

  As a first step, P_1 will evaluate if $A_1.B_1.C_1.D_1$ lie within the campus LAN. This is by comparing its Network ID based on its public address $A.B.C.D$ and the network ID of $A_1.B_1.C_1.D_1$.

  Since P_1 is a NATting point, it will first create an entry in the NAT-translational table. The source address is 192.168.0.2 and the destination address is $A_1.B_1.C_1.D_1$. The ports are copied as it is. P_1 then NAT it and forward it to the network using a modified ARP process specified in the *PrECast* infrastructure. As the first step, S_1 will check its *PrECast* table if there is a binding entry for <MAC, $A_1.B_1.C_1.D_1$ >binding. If it already exists, S_1 will transmit the IP packet encapsulated with a Layer 2 frame.

  If there is no binding information available in its *PrECast* table, S_1 will multicast an ARP-request for the IP address $A_1.B_1.C_1.D_1$. The switch T to which the host $A_1.B_1.C_1.D_1$ is connected will multicast the ARP-reply to all LAN switches, including to S_1. S_1 will then will transmit the IP packet encapsulated with a Layer 2 frame. All other switches, including S_1, will add T's Switch ID, MAC of $A_1.B_1.C_1.D_1$ and the IP address $A_1.B_1.C_1.D_1$ to their *PrECast*-table.

  If no switches in the network knew about $A_1.B_1.C_1.D_1$, the ARP-request will go for a timeout. After ARP-timeout, P_1 will issue an ICMP-host not found error to H_1 and remove the relevant entry from the NAT translational table.

- **Case 2:** H_1 wishes to communicate with $A_1.B_1.C_1.D_1$ where the destination host is outside the campus LAN.

The switch S_1 checks the network ID of the IP packet destined to $A_1.B_1.C_1.D_1$. In this case, $A_1.B_1.C_1.D_1$ is outside of the subnet $A.B.C.D$. S_1 will NAT and forward the packet to its default gateway as in Case 1.

In both cases, the *NAT++* engine checks if the source address is 192.168.0.2. If not, it will drop it silently. Using this mechanism, we effectively eliminated IP spoofing without incurring any extra overheads. There is no complex source address checking or crypto-algorithm used in our scheme. The source address authentication is inherent from the P2P relation.

7.4.3 Network Maintenance

It is a practice in a corporate LAN that network administrators frequently push OS and software patches to every LAN hosts. With our new design, this service may not be affected. The network administrator may set local policy to obtain updates from a corporate server (that resides within the corporate LAN) frequently. Thus, this maintenance service is not affected by our proposed design.

Another task of a network administrator is to perform port scanning on all LAN hosts (mostly the corporate hosts and not BYO machines). The purpose of this exercise is to identify any vulnerable ports that are open to the external network.

Port Scanning has its legitimate uses inside the network, and therefore, needs to be supported by the proposed solution. The information gathered by a port scan has many legitimate uses including network inventory and the verification of the security of a network. The network scanner involves sending some especially crafted TCP/IP packets to the target host and observing their reply. The network scanning is always initiated from a client host towards a target host.

In our proposed *NAT++* implementation, all end hosts apart from the important servers are hidden inside a NATed network. Any network host, including an administrator trying to reach an inside host, their connection will be considered to be "unsolicited" and will be dropped by the NAT engine. This is one of the strengths of our proposal. However, to facilitate port scanning for a legitimate purpose, we need to propose an alternative method.

In a NATed environment, port scanning is still possible from a host within the NAT realm. Since in our case, we have a P2P network, port scanning must be initiated only from 192.168.0.1. 192.168.0.1 is assigned to each switch ports to which end-hosts are connected. Thus, port scanning must be initiated only from the switch, not from any other hosts. Since the network administrators have legitimate access to corporate switches, they can connect with individual switches and perform port scanning across all connected hosts. By design, no other hosts can successfully perform a port scan.

7.4.4 The Attack Vector

In this subsection, we discuss various possible attacks against the *NAT++* system.

IP Spoofing from Internal hosts: By design, this type of attack is not possible. Whenever a *NAT++* port detects an outgoing IP packet with the return address not belonging to its only host, the packet will be dropped immediately without forwarding to the network. Thus, preventing spoofed IP address being generated inside the corporate network.

Address Spoofing by an External Host: We consider an external host H spoofing the corporate IP address $A.B.C.D$. If the host network does not implement the *NAT++* protocol or other IP-address spoofing solutions, the IP packet will be forwarded to the destination host B faithfully. Based on the spoofed source address, Host B will send its reply to A. Being an unsolicited reply, the *NAT++* engine will drop this packet silently.

Address spoofing that leads to DDoS attack: As in the previous case, in the absence of any forwarding entry, the NAT port will drop all the unsolicited replies arriving for the destination Host A. If this attack leads to “resource starvation”, then only the particular switch to which Host A is connected will be affected. The rest of the network can function without any problem. However, if the state-information is maintained at the corporate firewall, all the unsolicited replies can be dropped even at the gateway; thus, not affecting any network or host services.

Port Scanning attack: Every public-facing host, or a server may be subjected to this attack. By our design, all these hosts have private IP addresses. Since they are behind a NAT port no external host can probe their TCP or UDP ports.

Network Chaining: Controlling the network diameter in a corporate network provides low and predictable latency. The strict control of the network topology at the access layer should be maintained [63]. Users at the access layer tend to add switches to the existing network topology inappropriately to get connectivity to their personal devices. This problem is commonly known as “chaining”. Our proposed *NAT++* infrastructure seamlessly solves this problem without any extra requirements.

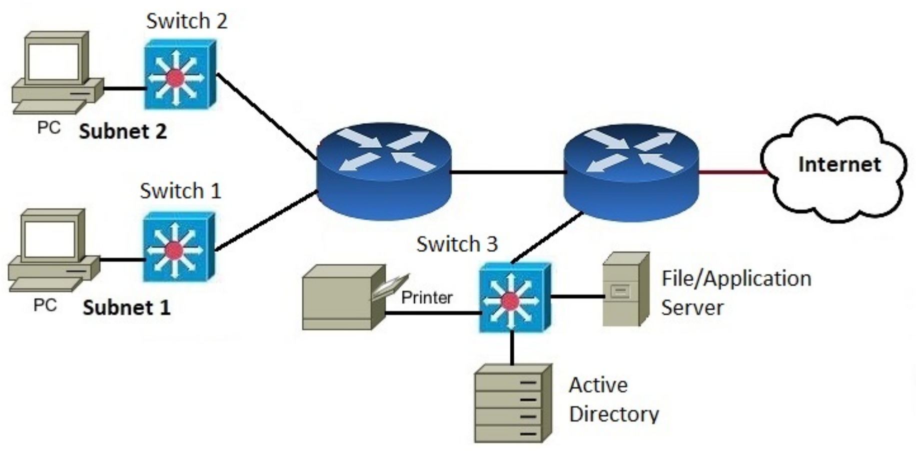


Figure 7.4: Potential challenges and Configuration issues: Experiment 1.

7.4.5 Sharing Resources in a Corporate Network: Potential Challenges

In a corporate network, it is necessary to share resources across its users. This includes printers and shared storage. With the invention of Cloud-based services, many corporates are moving towards cloud-based storage services that provide more convenience, security, and redundancy. One such example is Microsoft's OneDrive. However, an organization may be interested in sharing printers and share local files (ex. Group drives). The "Server Message Block" (SMB) Protocol [60] is a Network resource sharing protocol native to Microsoft Windows and widely supported under various platforms through Open Source Samba project [88]. The SMB protocol operates at the Application Layer of the OSI stack. It relies on lower-tier protocols for its transport. SMB uses NetBIOS over TCP/IP. The SMB protocol implements a client-server architecture. The current version (3.1.1) supports AES-128-GCM for encryption. This protocol is effectively exploited in every corporate network to share network resources. SMB uses TCP port 445 and UDP ports 139.

To evaluate potential challenges and configuration issues, we conducted two experiments under different environments.

1. We implemented a one-to-one NAT in a small network as outlined in Figure 7.4.
2. We implemented a one-to-one NAT in our University segment (as in Figure 7.5) and evaluated the ease of accessing shared services that are already available in our network.

Experiment 1: One-to-one NAT as per Figure 7.4.

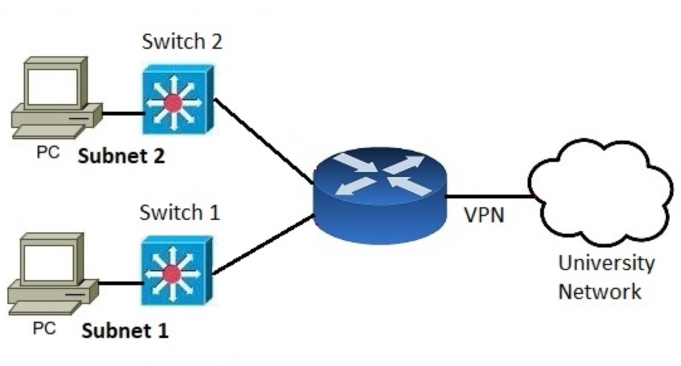


Figure 7.5: Potential challenges and Configuration issues: Experiment 2.

In this experiment, we use the SMB protocol for creating a shared folder and a print server. We use the Windows Active Directory to create three access groups namely *Student*, *Staff*, and *Admin*. We use access control policies on the shared folder and the print server. *Students* have no access to the shared folder and print services. The *Staff* has read access to shared folders and no print services. *Admin* has full read and write access to the shared folder and full print services.

We split this experiment into two scenarios: In the first scenario, we hide the file server and the print server inside a one-to-one NAT. In the second scenario, file and print servers were connected to a switch port with no one-to-one NAT involved, as per our proposal. In this case, both the machines were given a public IP address within the corporate IP range.

Challenges faced in Scenario 1: With the default one-to-one NAT configuration on Switch 3, the hosts were unable to access both print and file server, irrespective of the user access right at the host. This is because, one-to-one NAT on Switch 3 was denying access to servers, as these connections were considered to be "unsolicited".

Lesson Learned: We created a TCP hole punching at the NAT engine at Switch 3 for port 445 and UDP port 139 for the SMB protocol to listen for any incoming connections. However, hole punching is a manual process that needs to be done carefully (not to open other TCP or UDP ports by accident) by the network administrator. The hole punching permits the NAT-engine to accept unsolicited connections to these TCP and UDP ports.

Once we created a hole punch, we were able to access these services according to the access control policies we established for users. However, we incurred an average delay of 100 ms before we initiate these services.

Challenges faced in Scenario 2: Since the print and file servers have a public IP address, accessing them was easy compared with Scenario 1. Compared to Scenario 1 with hole punching, the latency in accessing these services could not be detected.

Challenges faced in Scenarios 1 and 2: In the absence of any “service catalog”, it is difficult to know what services are available in the network.

Lesson Learned: We created a simple intranet web page that consists of a catalog of network services available in the network along with its “Server name”. The name server is configured to resolve the Server name to its IP address to access these services.

Based on two scenarios, we noted that while file and print servers were configured with a public corporate IP address and no NAT, they provide less latency. Since all internal end-hosts (other than important servers) are protected by *NAT++*, there is no threat to these services (including poisoning or spoofing). With proper firewall rules, these services may be protected from external threats. This validates our proposal of not hiding essential servers under *NAT++*.

Experiment 2: One-to-One NAT integrated with the Campus network (Figure 7.5)

In this experiment, we also launched two attacks, one originating from the host and another towards a host.

IP-Spoofing originating from a host: We created a spoofed IP packet with a source IP address that belongs to the public IP range. Since this address was not the same as 192.168.0.2, the NAT engine silently dropped; thus, preventing IP-Spoofing originating from inside hosts.

IP-Spoofing originating from the external host: We created a spoofed packet with a public source IP address that is being used by PC connected with Switch 1. This packet was then launched from an external network where there is no *NAT++* exists. The destination host faithfully sends its reply to the corporate host. Since this IP packet is deemed to be an unsolicited reply, the one-to-one NAT engine silently dropped this packet; thus, preventing IP-Spoofing originating from an external network

7.5 Simulation and Performance Analysis

In this section, we perform two different performance benchmarking to evaluate the strength of the proposed *NAT++* framework.

- Analyse the performance of the *NAT++* protocol against various NAT standards [73]. The purpose of this simulation exercise is to evaluate the additional workload introduced by the proposed *NAT++* architecture in comparison with various standard NAT implementations.

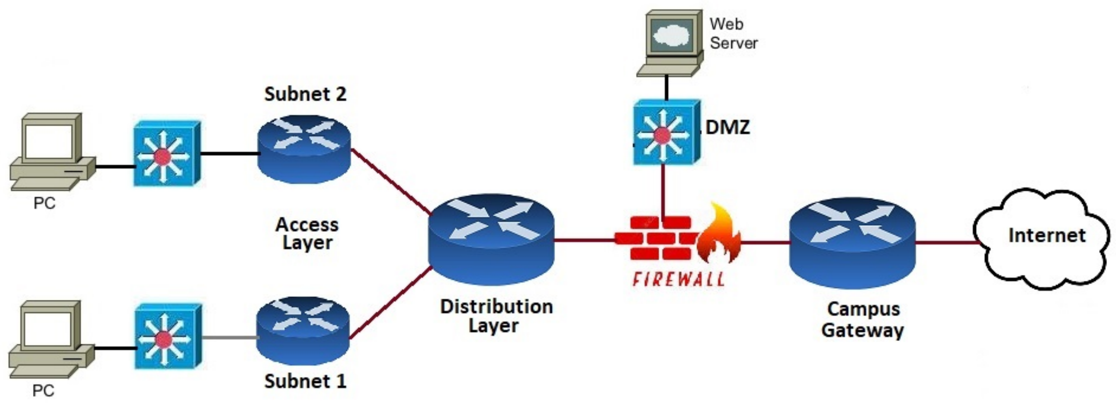


Figure 7.6: Experimental Topology.

- Compare the Central Processing Unit (CPU)-load on the first-hop network devices for an existing IP-spoofing solution and the *NAT++*.

There are two important parameters related to the performance of an IP-flow between two devices. The performance may either apply to end-host devices or the intermediate network devices such as a router:

1. CPU-load on the intermediate devices.
2. IP-Processing delay on the intermediate devices.

Whenever a NATting mechanism is implemented on an intermediate device, the delay due to the NAT process is added along with the IP-processing delay. As we discussed before, One-to-One NAT, Many-to-Few NAT and Many-to-One NAT (PAT) introduce different delays due to the complexity of their process.

In this section, we compare our proposed *NAT++* scheme against the existing NAT standards through the above parameters.

Simulation Setup:

To evaluate the performance of the proposed *NAT++* protocol, we created a small campus network given in Figure 7.6. We have connected 12 PCs each to Subnet 1 and Subnet 2 through Cisco 3560 switches. Cisco 2800 series routers were used at the access layer. ISR4000 series routers were used for the distribution layer and the gateway.

Since we have no access to the Cisco IOS API while performing this work to implement our proposed protocol, we implemented the entire architecture using the OPNET modeler [101] to evaluate the strength. OPNET is a commercial discrete event simulation and modeling software that provides a comprehensive suite of vendor-models, in particular, Cisco switches and Routers. The use of these switches and routers in modeling provides identical performance benchmarking as if we are using the original equipment. Thus, OPNET is used mostly in the commercial environment for designing large-scale corporate networks. Due to its flexibility, power, and the availability of vendor-specific models, we used OPNET for conducting this research.

Traffic Modelling

Each PC fetch web pages from the server. Once a page-fetch is complete, the host waits for random seconds (follows an exponential distribution with a mean of 120 s) before making another page request. For each connection request, the latency is recorded.

Experiment 3:

In this experiment, we evaluate the IP processing delay due to the routing process at the router and the CPU load on the router. This experiment is conducted without any NAT. These baseline results are used for benchmarking purposes.

Experiment 4: One-to-One NAT

In this experiment, we created a one-to-one NAT at the subnet router (Subnet 1 and 2). These routers handle NAT while an IP-packet exit though its exit-interface connecting Subnets 1, 2, and the Distribution Layer router. The router also needs to maintain the translational state-information in a table for translating a reply from the server. Whenever a TCP handshake happens between the client and a server, the router performs the NATting task using the translational table. This is an additional load compared with the baseline experiment.

Experiment 5: Many-to-Few NAT

In this scheme, the total available IP address is less than the number of hosts Whenever all public IP addresses are in use, any new outgoing connection is put on a queue until an on-going connection is closed after page-fetch. This may affect the Queuing delay.

Experiment 6: Many to One NAT or PAT

In this experiment, the IP address of the exit interfaces of the Subnet 1 and 2 routers are used for PAT. There is more processing involved in this type of NAT compared with one-to-one and many-to-few NAT.

Experiment 7: *NAT++*

This experiment implements our *NAT++* protocol. This scheme is similar to one-to-one NAT; however, every switch port handles only one NAT connection compared with the router (or traditionally NAT boxes) that handle all NAT connections. The NATed connections are directly switched on to the router's ingress interface (similar to VLAN trunking). In a traditional one-to-one NAT, all the NATting load is handled by the gateway router along with routing overload. In the case of *NAT++*, the NATting process is offloaded to LAN switches, and thus gateway routers incur no additional load due to the NATting process. Therefore, due to this experiment, the load on to the gateway router is similar to the baseline test.

In all other experiments except *NAT++*, the LAN switches work as a Layer-2 device. However, in *NAT++*, it has to perform the NATting process. Thus, edge LAN switches incur additional CPU load and queuing delay due to the NATting process.

Experiment 6: Performance Benchmarking with a Similar solution

We implemented a solution similar to the one proposed in [95, 31]. To detect IP spoofing more effectively and efficiently, the IP Source address checking process is implemented at the Ingress of the subnet routers. Every host that generates an IP packet will "digitally sign" the packet header to establish its identity. As soon as this packet is received by the router's ingress link, the digital signature is verified. Once the identity of the source host is verified, the router then checks the source IP address. If there is a mismatch, the packet will be dropped silently.

This task involves processing at the host side and the verification at the router side. Signing and verification require CPU cycles. In this experiment, we evaluate the CPU load incurred at the host and the router side.

7.5.1 The Results

In this subsection, we present the results obtained through our experiments.

CPU Workload on the router:

In Figure 7.7, we presented the CPU load on the gateway router. As we expected, the baseline experiment without any NATting incurred the least CPU load. The CPU load due to our proposed *NAT++* infrastructure is also similar to the baseline test. As we explained before, this is because the NATting process is done at the LAN switch. The next higher CPU load was due to one-to-one NAT followed by Many-to-few and many-to-one NAT.

In Figure 7.8, we presented the CPU load on the 3560 LAN switch that was due to the *NAT++* process. We then wished to compare the total CPU load incurred by the *NAT++* process. Thus, we added the CPU load incurred at the Switch and the gateway router and compared it with one-to-one NAT. The comparison is presented in Figure 7.9. As we can see from this result, the total IP processing delay is similar to one-to-one NAT.

IP Processing delay:

From the results presented in Figure 7.10, it can be seen that IP processing delay at the router for No-NAT and *NAT++* are similar. This is because *NAT++* needs no translation similar to the no-NAT process. This is followed by one-to-one NAT and many-to-few NAT. Many-to-few NAT incurs the largest IP processing delay.

Traditionally, routers implement fast switching feature. Fast switching still uses the CPU, but after a packet has been forwarded, information about how to reach the destination is stored in a fast-switching cache. Whenever another packet to the same destination

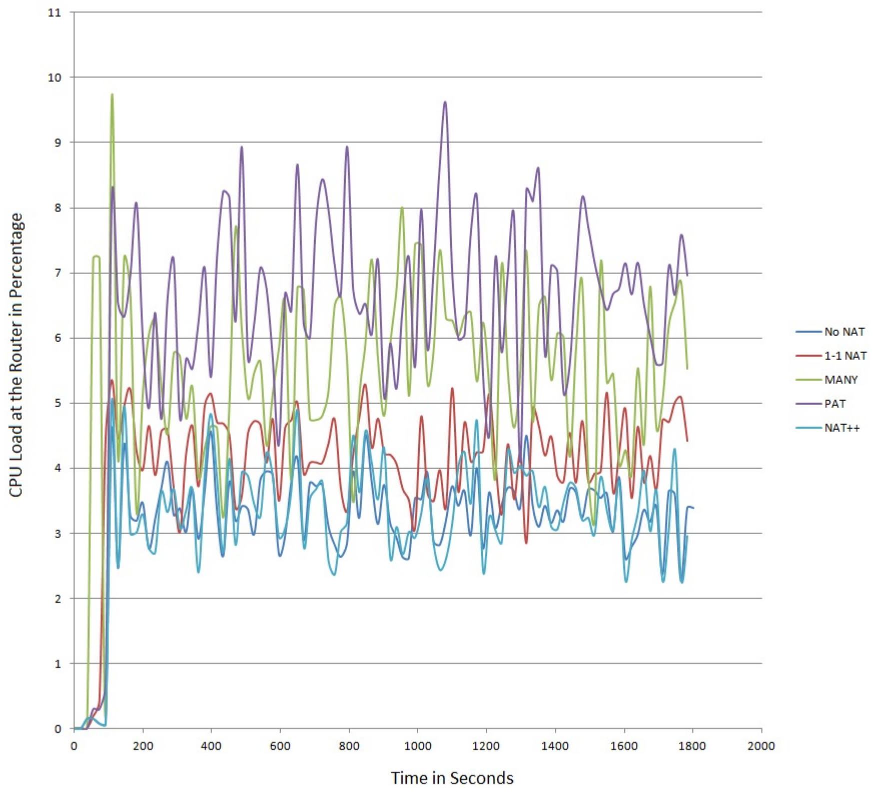


Figure 7.7: CPU Workload on the router.

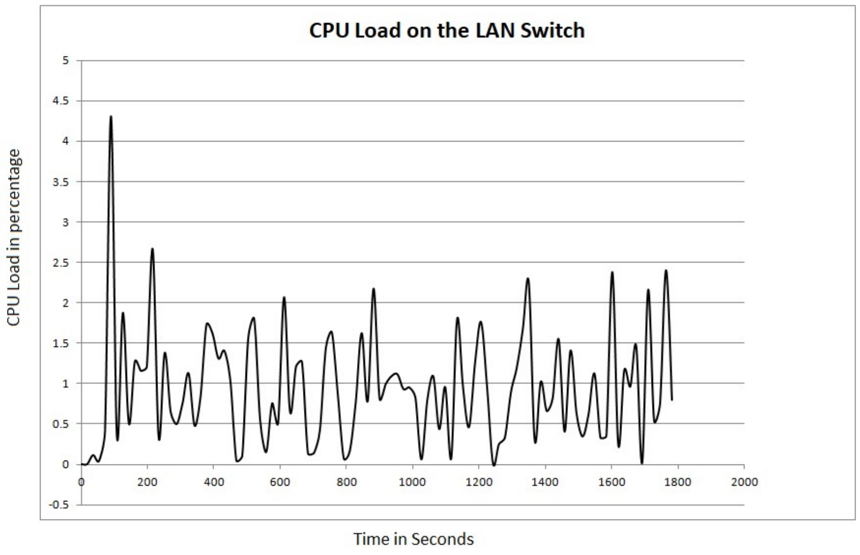


Figure 7.8: CPU Workload on the switch.

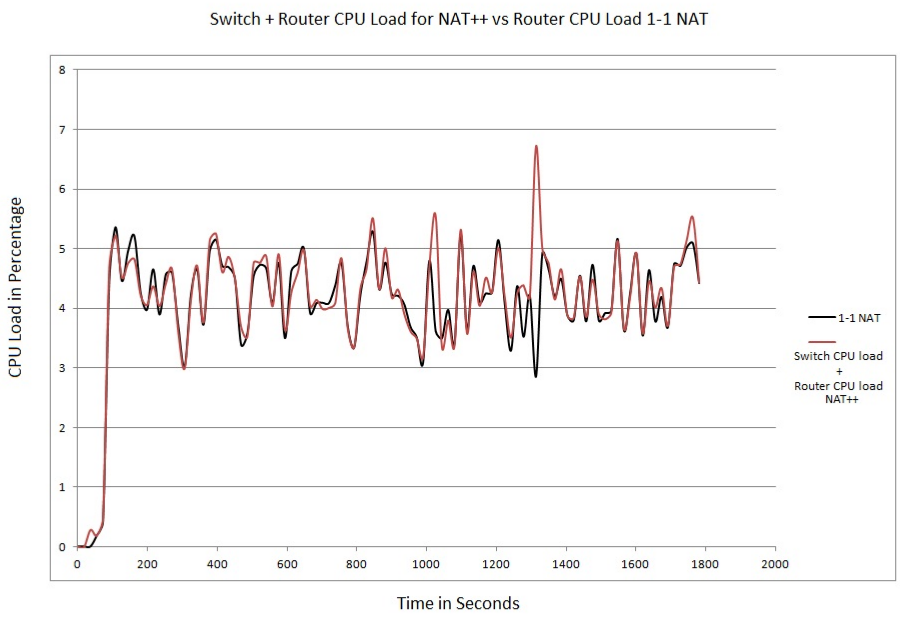


Figure 7.9: Total CPU Workload on the router and the switch.

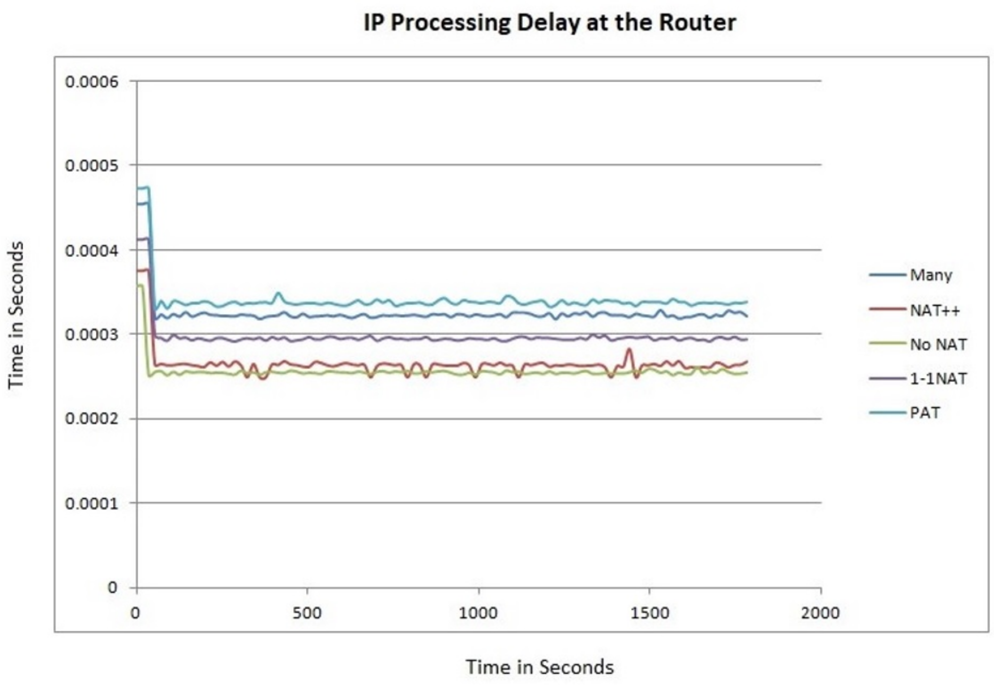


Figure 7.10: IP Processing delay.

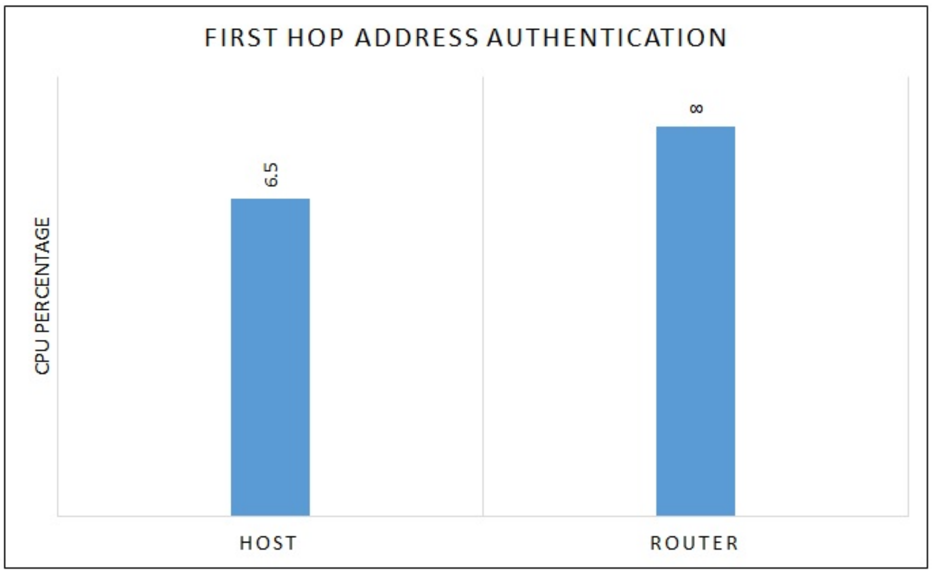


Figure 7.11: CPU load based on first-hop authentication.

arrives at the router, the router will use the cache entry to forward through the correct interface.

Benchmarking with a similar solution:

We implemented a solution similar to the one proposed by Wang et al. [95] and Fonseca et al. [31]. To detect IP spoofing more effectively and efficiently, IP Source address checking is implemented at the Ingress of the subnet routers. Every host that generates an IP packet will "digitally sign" the packet header to establish its identity. As soon as this packet is received by the router's ingress link, the digital signature is verified. We implemented 1024-bit RSA as our underlying cryptosystem.

This task involves processing at the host side and the verification at the router side. Whenever an IP packet is received by the router's ingress interface, as a first step, the integrity of the received packet is checked using the host's public key.

Signing and verification require CPU cycles. In this experiment, we evaluate the CPU load incurred at the host and the router side. This is presented in Figure 7.11. The first hop header authentication incurred a total CPU load of 14.5%, whereas the *NAT++* incurred only a CPU load of 4.5% on an average. This is because *NAT++* does not use any cryptographic mechanism to check the validity of the source IP address.

7.6 Chapter Summary

In this chapter, we proposed a simple and scalable solution to thwart the IP-Spoofing problem that arises from an internal network. Our solution uses a modified form of NAT maintained at every router port. The proposed *NAT++* architecture effectively eliminates DoS attacks originating from inside malicious nodes. Since every internal host are hidden from every host, including external hosts, IP-spoofing attacks from the external network is not possible. The proposed architecture also solves the port-scanning attack from the external and internal networks. If the corporate firewall is configured with an effective flood protection mechanism, we can effectively eliminate DoS attacks that are originating from the external network. We implement *NAT++* along with the *PrECast* infrastructure that eliminates MITM attacks in a LAN. Together with *NAT++* and *PrECast* infrastructure, a network can defend against several known attacks.

NAT++, along with the *PrECast* infrastructure, solves several LAN based attacks. However, an internal host may either voluntarily or involuntarily establish a connection with a malicious destination host, click spam links, download malware, visit websites hosting malicious codes and so on. The *NAT++* with *PrECast* infrastructure will not provide a solution to these attacks. Chapter 8 provides a dynamic IP filtering mechanism to solve these attacks. The dynamic IP filtering mechanism uses the destination IP reputation score to determine whether to allow or deny a connection to a destination host.

Chapter 8

Dynamic Filtering to thwart malicious activities in a network through IP Reputation and Policy Engine

Currently, we have no mechanism to evaluate which end-host got malicious intention or not. Thus, detecting and thwarting connections to malicious IP hosts is one of the critical problems in the Internet world. Existing firewalls provide some form of rudimentary solution based on certain statically before-known conditions. However, they are not sufficient. A legitimate host may connect with a malicious host due to malware infection, clicking an embedded link in a phishing email or an infected DNS connection.

IP reputation is a mechanism of awarding a credit score to an IP address or its associated domain. The credit score may be awarded based on spam origin, the origin of malware spread, and the address of a "Command-and-Control" system. There are public and private IP reputation databases available on the Internet. However, they are primarily standalone. Some of the IP reputation databases are application-specific. This chapter integrates the public IP reputation database to provide an efficient and seamless IP filtering at the first hop on any Local Area Network (LAN). We name our proposed infrastructure as IP Reputation and Policy Engine (IPRPE). The latency introduced by IPRPE is an essential deciding factor that determines the efficacy of the proposed system. We performed a real-time simulation. Our simulation results show that the latency introduced by IPRPE is well within the acceptable range.

The dynamic firewall-based filtering mechanism proposed in this chapter can be applied to both the IPv4 and the new IPv6 protocols. The contents presented in this chapter are unpublished work of the candidate.

8.1 The Prelude

As we mentioned in previous chapters, the Internet has witnessed an exponential growth. We now discuss the aftermath of this exponential growth. Due to rapid proliferation, several people, such as elders, children are forced to adapt to the technology without proper training and orientation. Even though it is easy to use the Internet to perform day-to-day tasks, most users are unaware of its darker side.

Every Internet user establishes a connection with hundreds of other Internet hosts to perform their tasks. While performing the task, the intention of the other end-host is not known all the time. Some hosts may have malicious intent. Taking advantage of this situation, malicious Internet users (often called attackers or actors) target innocent users. Financial gain and obtaining user private data are their primary motivation. Using the user's private data, the attacker may perform "identity theft". In a survey [8], it was estimated that one in three Americans will be impacted by identity theft at some point in their life. Identity theft is a menace in several developing countries. Exploitation using the Internet and obtaining private data is punishable by law in several countries. However, attackers conceal their identity or maybe from different countries where such a civil law is not enforceable.

Attackers may find several different ways to compromise an end host, to steal user information. Mostly, it is done by installing malicious software in the target computer. The malicious payload is delivered to an end-host without their knowledge. Attackers use several different techniques to deploy the payload depending on the end user's knowledge. Popular techniques include using social engineering, phishing emails, infected DNS, compromised websites, and infected USB flash drives. Among the various form of malicious software, Trojan and Spyware are the most popular ones. Attackers program this malware to perform specific tasks whenever some triggering conditions are met. They run under "stealth-mode" without being detected by the system or the network administrators. Infected malware is programmed to collect sensitive user information such as keystrokes, specific documents, system blueprints etc. Active malware needs to establish "Command-and-Control" (C&C) channel with a remote system to communicate the collected sensitive information or a "command" to perform any other specific tasks. The malware also transmits their "heartbeat" to their C&C. The heartbeats are small beacon messages sent to their C&C to keep their sessions alive [18]. The detection of malware activities in a network is an interesting problem, both from the research and commercial community.

Attackers frequently use social engineering approaches to infect any end-systems [97]. The end-system here may represent user's personal device or a corporate device. Phishing attacks are a subset of social engineering strategies employed by attackers to extract login and other private data. For launching phishing attacks, the attackers create an email concocting a logical scenario, and fabricate the source address as a “trusted source”. The phishing emails may also contain URL (Uniform Resource Locator) that hosts malware or infected embedded documents that may install malware by exploiting end-users’ system vulnerabilities. According to Verizon data breach investigation report [26], 93% of the successful data breach are through phishing and pretexting attacks (attack in which the attacker creates a scenario to try and convince the victim to give up valuable user and network specific information).

Like malware detection, detecting and stopping phishing attacks is an exciting problem for the academic community and the industry. Phishing emails accomplish their expected outcome with full cooperation from an end-user. With the current security technologies, an end-user can always have the freedom of clicking a phishing email and reveal private information. Thus, it is hard to stop phishing attacks. Modern firewalls provide some form of rudimentary protection against malware and phishing attacks. The attackers always find cleverer ways to circumvent any protections.

This chapter provides a simple and scalable solution based on IP reputation to detect and potentially stop malware communication and phishing attacks. Our solution is called “Dynamic IP filtering (DIF)”. The heart of the DIF framework is the “IP Reputation and Policy Engine (IPRPE)”. The DIF can either be integrated with any existing firewalls or work as a distributed, stand-alone system to offer a dynamic IP filtering based on source and destination IP and its domain reputation. In a distributed mode, the DIF architecture is tightly integrated with the *PrECast* architecture. Even though we don’t recommend the standalone mode due to various problems listed in previous chapters, for the completeness of this thesis, we provided the standalone mode.

8.2 Detecting Malicious and infectious connections

Preventing a user from accessing hosts containing malicious codes is a complex problem in computer security. An end-user may initiate a connection to a site that hosts malicious contents (we call them malicious sites) either knowingly or unknowingly. In the first instance, the user may have evil intent to download malicious content to infect the local system and the network. Alternatively, users may access sites not authorized by their IT (Information Technology) policy (such as sites that host obscene pictures). These sites may direct them to sites that contain malicious codes. In the latter scenario, we have the following cases:

- A previously infected system is trying to communicate with its C&C.

- User may click a link provided through a phishing email that takes them to malicious sites.
- Users may visit legitimate websites that contain infected “Adware” that may redirect users to malicious websites.
- An infected (or hijacked) DNS may direct legitimate users to malicious sites.

We now analyse the current cyber-solutions to solve these problems. A firewall is traditionally used to protect the corporate network periphery from external networks. They enforce access control policies. Access control policies define the scope of who can access which services. Firewalls can also prevent incoming or outgoing connections based on the application layer payload (using keywords or smart filtering). They also detect malware based on their known signatures. However, they cannot detect unknown malware.

Similarly, signature-based malware detection will not detect new malware. There are several signature-based malware detection techniques available in the literature. Each technique got some advantage over the other. A good review of various malware detection techniques, their advantages and disadvantages are presented in Xue, and Sun [99]. They also proposed their own malware detection technique based on the network behaviour evidence chain. Any machine learning technique based algorithm will have its inherent disadvantages. Machine learning techniques need data to train the system in detecting attacks. Therefore, they may not detect new malware or phishing attacks that differ from the existing attacks. Malware writers may also use the existing machine learning algorithms to circumvent its detection.

While Xue and Sun [99] focused on the physical behaviour of malware in preventing their attacks, Prowell [66] devised a solution to focus on the malware heartbeat instead of its physical behaviour. His solution’s underlying assumption is that malware infection will produce a measurable change in the power consumption state of a device that an outside detector can pick up. He assumed that any increase in the power consumption was due to a malicious malware’s heartbeat. This assumption may apply to an embedded device. The power consumption of a general-purpose device such as a PC or a smartphone varies depending on the CPU usage by various applications and the GPU (Graphics Processing Unit) usage. Thus, we are not convinced that Prowell’s assumption will be valid on a general-purpose machine.

”Heartbeat network” was a new discrete structure coined by Wei et al. [27]. It is natural to assume that hosts with the same heartbeat have the same applications (or infections) and share homogeneous vulnerabilities. Wei et al. created the “heartbeat association graph” as follows: Nodes producing the same heartbeats are grouped as one node. An “undirected association” is established between different nodes based on the relation between various different heartbeats. Weights may be associated with different nodes. Weights are also associated with links between nodes to establish the strength of the association between these nodes. Thus, the resultant graph is an edge and vertex-weighted undirected graph. This topology is very useful in analysing the relationship between different botnets and malware. Unless we collect a long-term characteristic on various malware and botnets, it is not easy to group similar nodes and construct a heartbeat association graph.

BotMelt is a trigger-based system proposed by Kang et al [9] to analyse malware infections. Their analysis involves fingerprinting every malware based on its triggering conditions. They have not considered the infection characteristics of malware. Thus, if two malware execute based on the same triggering condition, BotMelt will group them as one. However, they performed extensive work on characterising various triggering conditions for four prevalent botnets and listed their potential outcome. Even though their work is extensive in characterising various triggering conditions, we found two potential issues. As we mentioned before, malware are fingerprinted based purely on their triggering condition, rather than considering their destructive nature. Thus, it may lead to grouping virulent and less harmful malware together. We have no access to malware codes. Thus, we may not know how to find all the triggering conditions to fingerprint malware. Some malware action may be triggered on a particular condition (such as the birthday virus) that may not be known until the condition is executed.

Malware mimics real network traffic such as SSL, HTTP, SMTP, and DNS to communicate with their C&C. Malware also encrypt their payload so that it is difficult for a firewall to detect its content. Thus, it is often difficult to differentiate between malware traffic and the real traffic in a network. This condition leads to false positives, as, the malware frequently changes its domain of contact and its pattern to mimic normal traffic. Gracia and Pechoucek [81] devised a ”directed acyclic graph” based approach to better detect the malware infection and separate them from the normal traffic. Their mechanism involves the grouping of connections generated by similar malwares. Their approach is like Wei et al. [27]. Thus, they also suffer similar limitations.

Every human can be associated with a “reputation and characteristic” score based on the history of their behaviour. Can this property be replicated to Internet communication? Internet communication requires end-hosts to have unique IP addresses. End-host’s IP address is not permanent and may change. However, the end-host obtain an IP address from a pool that some ISP owns. Thus, the ”reputation and characteristic” can be associated with an IP address like humans.

IP reputation is a mechanism of awarding credit scores to IP addresses and their associated domain name. The credit score can be awarded either by a single organization or several organizations through collaborative means. Awarding a credit score based on collaborative means have several advantages and will be unbiased. Credit scores are awarded through several different parameters. These parameters may be different for a different organization that awards this credit score. For example, one organization may be interested in only constructing spam filtering. They may award credit scores based on spam origin, whereas another organization may award credit scores based on the origin of malware spread. A comprehensive credit score scheme may involve several different parameters. Attackers often use a spoofed source IP address or a compromised machine to send malicious files to the intended destination. They do so to hide their identity. There are not many works available in the literature in exploiting the IP reputation scheme to secure a network.

Traditionally, SMTP servers use IP reputations servers to stop unsolicited bulk and spam emails. In [29], Esquivel, Akella and Mori proposed an SMTP spam filtering scheme based on IP reputation. They award credit scores based on how many spam emails originate from a particular IP address. They classify SMTP senders into three main categories: legitimate servers, end-hosts, and spam-gangs. They study the effectiveness of IP filtering mechanisms across these three categories. It is surprising to note from their finding that both legitimate servers and spam-gangs use DNS Sender Policy Framework to pass simple authentication checks. One of this paper's critical outcomes is that spam-gangs must be continuously updated to maintain a higher accuracy level. It is also important to note that updating the spam-gangs continuously is not an easy task.

From the literature we reviewed, several works are available to provide ranking to domains and IP addresses based on the history of their malicious activities. However, to our knowledge, none of the work available in the literature provides an efficient mechanism to use the ranking to thwart malicious attacks. In this proposal, we are not creating a ranking mechanism for providing values for trust based on malicious activities' history. Instead, we assume a Global database contains IP domains and address ranges and their associated reputations. Our assumptions are realistic, as there are several public databases available on the Internet. Some of the examples of publicly available databases are Talos [17], IP Quality Score [41], Symantec IP Reputation [42], Webroot Bright Cloud IP Reputation Service [98], and Cyren IP Reputation Check [20].

Based on the IP reputation score provided by public databases, we provide an efficient algorithm to stop malware activities at the end-host. Our algorithm also acts as a dynamic filtering mechanism at the first hop. The administrator sets enforceable rules at the first hop itself to allow or deny connections to various networks. We describe its detailed design in the following section.

8.3 The DIF architecture

Currently, IP reputation scores are used only for research purposes. Public databases such as Talos [17], IP Quality Score [41], Symantec IP Reputation [42], Webroot Bright Cloud IP Reputation Service [98], and Cyren IP Reputation Check [20] are used through their proprietary system. All these database providers offer API (Application Programming Interface) for their clients to develop and customize their own applications. However, there are no such applications available. This chapter provides a framework for integrating the public IP reputation database to stop an end-host in establishing a connection with an IP address with a low reputation score.

The DIF architecture contains three essential stages. We provide a snapshot below. More details on each of the stages are presented in Subsection 8.3.1.

Stage 1: Subscription to a Global IP reputation service

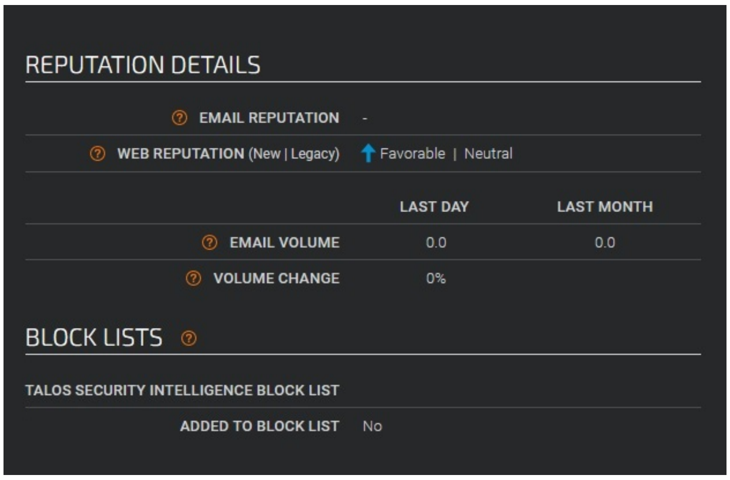
The brain behind the DIF architecture is the Global IP reputation servers. Every organization that wishes to implement the proposed DIF infrastructure must subscribe to one or more Global IP reputation services.

This stage involves querying the Global IP reputation servers to obtain a credit score for an IP address or a URL string. Based on the reputation threshold value set by the network administrator, the "allow" or "deny" decision is taken for a particular IP address or the URL. In this thesis, we consider Talos [17] as our Global IP reputation server. To provide more clarity to readers, we provide a brief overview of the "Talos Threat Intelligent System" (we refer to this system as "Talos").

Talos provides an interface for querying its database to obtain a reputation score based on an URL or an IP address. We demonstrate this feature through three examples.

For the first example, we searched their database using a "known reputable URL" (we did not explicitly provide the URL or IP address here for privacy reasons). Even though the query returns several valuable data related to the URL, we present only information related to this work for demonstration purposes. The URL is classified as "favourable" by Talos (readers may refer to Talos website to obtain more information on their classification) and is not "blacklisted by Talos". The outcome obtained for this example is presented in Figure 8.1.

In the second example, we search the Talos database through an IP address. The reputation obtained by Talos is presented in Figure 8.2. This IP address is known for spreading "spam messages". Thus, its email reputation is "poor". Since no HTTP server is running in this IP address, its "Web" reputation is "neutral". However, this IP address is blocked by two other public reputation databases. Thus, any organization may "deny" an IP connection either to this IP address and "deny" an incoming connection from this


REPUTATION DETAILS

EMAIL REPUTATION	-	
WEB REPUTATION (New | Legacy)	Favorable | Neutral	
	LAST DAY	LAST MONTH
EMAIL VOLUME	0.0	0.0
VOLUME CHANGE	0%	

BLOCK LISTS

TALOS SECURITY INTELLIGENCE BLOCK LIST

ADDED TO BLOCK LIST No

Figure 8.1: Talos Example 1: Known Reputable URL.

IP address.

In the third example, we once again used an URL for the search. The results obtained from Talos is presented in Figure 8.3. This domain is known for spreading malware and launching phishing attacks through emails. Thus, its reputation is worse and blacklisted. This URL is also classified as an "attacker". Thus, we set "deny" for this URL.

Stage 2: Active IP and URL Filtering

Active filtering is performed for a destination IP address or an URL during this stage. The filtering is done based on the reputation score obtained in Stage 1. If the IP reputation score is to "allow" the connection, the packet will be forwarded towards the destination IP address or the URL. If the IP reputation score is below the threshold and "deny" the connection, the IP packet will be dropped at the corporate network. Thus, a corporate host will not connect with another host with a lower reputation score.

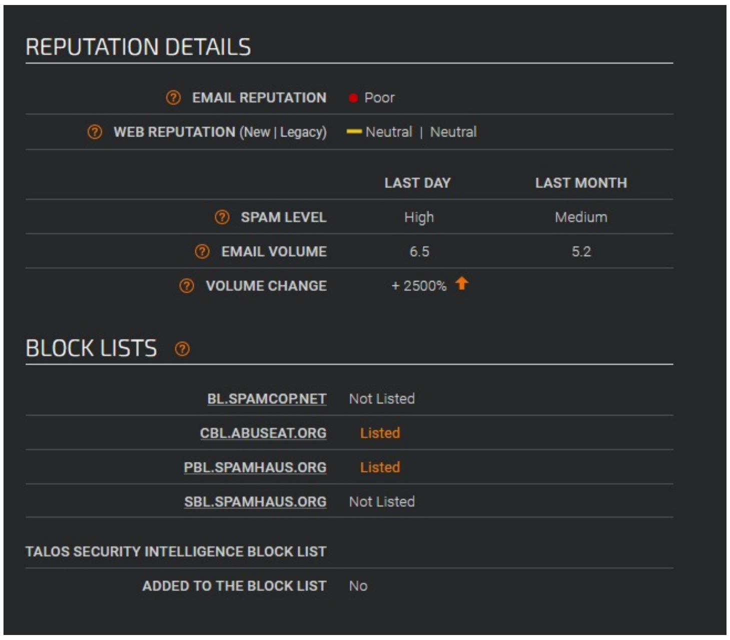
Stage 3: The recency of the IP Reputation Database

The reputation score for an IP address or URL is dynamic and keeps changing with time. Stage 3 is concerned with constantly updating the reputation score for all the frequently used IP connections.

We now provide some salient features of the proposed DIF architecture.

- Since IP domain reputations are checked for every IP flow, any host trying to establish a connection with a malicious outside domain can immediately be detected. A host may either be malicious or compromised. Our future work involves creating an intelligent honeypot to analyse the behaviour of hosts that are trying to connect with malicious outside hosts.

- An inside host infected with malware may try to send its heartbeat to an external command-and-control server (known through its "IP reputation"). By picking the



REPUTATION DETAILS

EMAIL REPUTATION	● Poor	
WEB REPUTATION (New | Legacy)	Neutral | Neutral	
	LAST DAY	LAST MONTH
SPAM LEVEL	High	Medium
EMAIL VOLUME	6.5	5.2
VOLUME CHANGE	+ 2500%	

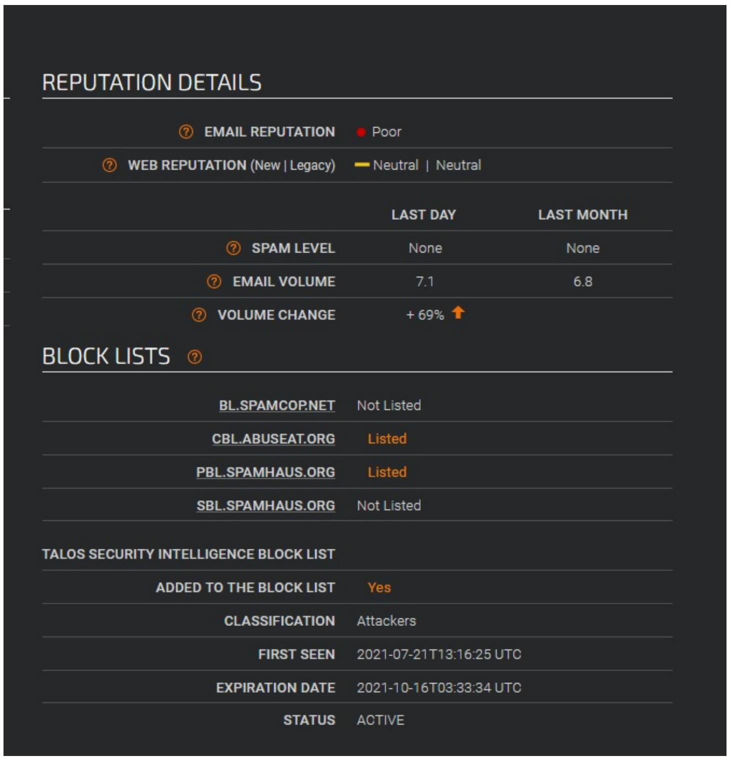
BLOCK LISTS

BL.SPAMCOP.NET	Not Listed
CBL.ABUSEAT.ORG	Listed
PBL.SPAMHAUS.ORG	Listed
SBL.SPAMHAUS.ORG	Not Listed

TALOS SECURITY INTELLIGENCE BLOCK LIST

ADDED TO THE BLOCK LIST	No

Figure 8.2: Talos Example 2: Search through an IP address.



REPUTATION DETAILS

EMAIL REPUTATION	● Poor	
WEB REPUTATION (New | Legacy)	Neutral | Neutral	
	LAST DAY	LAST MONTH
SPAM LEVEL	None	None
EMAIL VOLUME	7.1	6.8
VOLUME CHANGE	+ 69%	

BLOCK LISTS

BL.SPAMCOP.NET	Not Listed
CBL.ABUSEAT.ORG	Listed
PBL.SPAMHAUS.ORG	Listed
SBL.SPAMHAUS.ORG	Not Listed

TALOS SECURITY INTELLIGENCE BLOCK LIST

ADDED TO THE BLOCK LIST	Yes
CLASSIFICATION	Attackers
FIRST SEEN	2021-07-21T13:16:25 UTC
EXPIRATION DATE	2021-10-16T03:33:34 UTC
STATUS	ACTIVE

Figure 8.3: Talos Example 3: Search through an URL string.

heartbeat, the network administrator may detect the presence of malware.

- IP reputation service may also act as a simple IP policy-based firewall.

In a corporate network, the DIF architecture works under two different modes. They are centralised and the distributed modes. We now present a detailed operation of the DIF architecture under centralised and distributed modes.

8.3.1 The Distributed Mode

We now describe the operation of the distributed DIF operational mode, and its strength and weaknesses. The distributed DIF operational mode contains a "pre-processing" stage before Stages 1, 2 and 3 mentioned above.

All the LAN hardware such as switches, routers, firewalls, and proxy servers actively participate in the DIF infrastructure in the distributed mode. In this stage, we construct a multicast tree (refer Chapter 5 for more details) consisting of all the LAN hardware parts of the DIF infrastructure. For the LAN hardware to communicate securely, a secure communication channel must be established. Secure communication is achieved through the use of public and private keys. This is called the "pre-processing" stage.

The distributed mode operates as follows: Whenever a new IP flow is detected at the first hop of a LAN switch, the reputation of the destination IP address is obtained from the corporate DIF database server. Based on the "allow" or "deny", a forwarding decision is taken at the LAN switch. IP reputations are cached locally at the LAN switches for efficiency.

Pre-Processing stage:

During this stage, one or more corporate DIF servers are identified. These servers obtain IP reputation scores from a Global database. This stage involves constructing a multicast tree that consists of all the LAN switches, routers and corporate DIF servers. Once a multicast tree is constructed, secure communication channels are established based on the private and public key framework.

The following are the challenges in this stage.

- We need to choose the most appropriate multicast tree construction algorithm that includes corporate DIF servers and all the LAN switches. There must be a provision for dynamic tree management. The algorithm must include dynamic reconfiguration of the tree when some intermediate nodes fail, removal of LAN switches (if they were decommissioned or compromised) and the addition of new LAN switches.

- We need to choose the appropriate admission control mechanism for the legitimate LAN switches to subscribe to the multicast tree. The admission control algorithm must also deny illegitimate switches trying to connect to the multicast tree. Illegitimate switches may either try to connect using their own identity or through some spoofed identity. In either case, the admission control scheme must deny their admission.

- This stage involves creating and maintaining an effective key-management system to establish secure communications between LAN switches and the DIF server. Even though public key infrastructure (PKI) can be used, PKI systems suffer a single point of failure in a Denial of Service attack (DoS).

In Chapter 5, we provided an efficient multicast tree construction algorithm based on the core-based tree multicast tree construction paradigm. To provide dynamic admission control, we introduced an enhanced cryptosystem called the pseudo-identity based encryption. The eco-system designed in Chapter 5 addresses all the three challenges mentioned above.

Thus during the pre-processing stage, we construct a multicast tree as per the algorithm outlined in Chapter 5.

Stages 1 and 2:

This step is vital for the DIF process. For every IP flow, the reputation score of the destination IP address must be available at the first hop LAN switch before a forwarding decision is made.

All the LAN switches and the corporate DIF servers build their IP reputation table gradually during this stage. IP reputation scores are fetched from a Global DIF-server on-demand and added dynamically to the cache table.

The corporate DIF servers must establish a secure communication channel between themselves and the Global reputation server. Secure communication channels are established through the use of secure tunnels.

At the start, the DIF cache at every LAN switch and the DIF server is empty. A cache-timeout value is chosen by the network administrator and is configured onto every LAN switch. We discuss more this feature under Stage 3.

Let H be a host connected to a switch S. H transmits a packet to a destination host D with an IP address IP_1. The switch S to which the host H is connected will receive this packet. The switch S will check its local DIF-cache if a reputation entry for the IP address IP_1 is available. If there is an entry and the reputation is to "allow forwarding", S will forward H's packet towards the next hop (or switch towards the default gateway). If the reputation is to "Deny forwarding", the packet from H towards D will be silently discarded without issuing any ICMP error to the host H.

We now discuss the case when there is no IP reputation entry available for IP_1 at S.

If there are no IP reputation entries for IP_1 available in its local cache or the available entry is expired, the switch S will send a "unicast" IP reputation request to the DIF server. This message must be digitally signed by the switch S for authentication. On receiving this request, the DIF-server will check its digital signature to ensure that S is a legitimate corporate LAN switch, not decommissioned or compromised switch. If the digital signature is not valid, then the request will be dropped. Whenever the digital signature is valid, the DIF-server will check its local database. If an entry is available and the entry is not stale, the DIF-server will multicast the IP reputation for IP_1 through the CBT to all LAN switches. The multicast message will be encrypted through the session key. Any intermediate node that receives this message and is not a part of this communication will not be able to decrypt this message. Every switch that receives this message will decrypt the message using the session key and add it to their DIF cache if the entry is not already available. Switch S will also add the entry for IP_1 to its DIF-cache. Depending on "allow" or "deny", a decision will be taken. Like a routing process, the same decision is taken for every IP packet between any host connected with a switch and D until the "Time To Live" expired or S received an "unsolicited update for IP_1" from the DIF-server updating its prior permission.

Note that the decision is not taken based on the <*source, destination* >pair. However, we can include this feature with ease. By doing so, we can restrict who can communicate between each other or not.

We now discuss the case when the IP reputation entry for IP_1 is unavailable at the corporate DIF server. If there is no reputation entry available for IP_1, the DIF server will request the Global system such as the Talos to provide a reputation for IP_1. As soon as the reply is received, the DIF-server will multicast the reply as before. The "reputation message" is also cached. Any future request from any switch will be handled without referring to the Global system until its expiry.

In the distributed DIF architecture, LAN switches make the forwarding decision depending on the IP reputation score. Thus, DIF architecture is not a replacement for the traditional firewall. The corporate firewall may provide an extra layer of security, such as deep-packet inspection, Intrusion detection, and flood control.

The flow chart that outlines these stages is presented in Figure 8.4 and Figure 8.5

Stage 3:

This stage is more concerned about the storage and search efficiency. Initially, there are no IP reputation entries stored in every LAN switch and the corporate DIF server. As soon as a new IP flow is detected, the corresponding IP reputation of the destination address is added to every LAN switch and at the corporate DIF server. As time progresses, more entries are added to the IP reputation table, increasing its size proportionate to

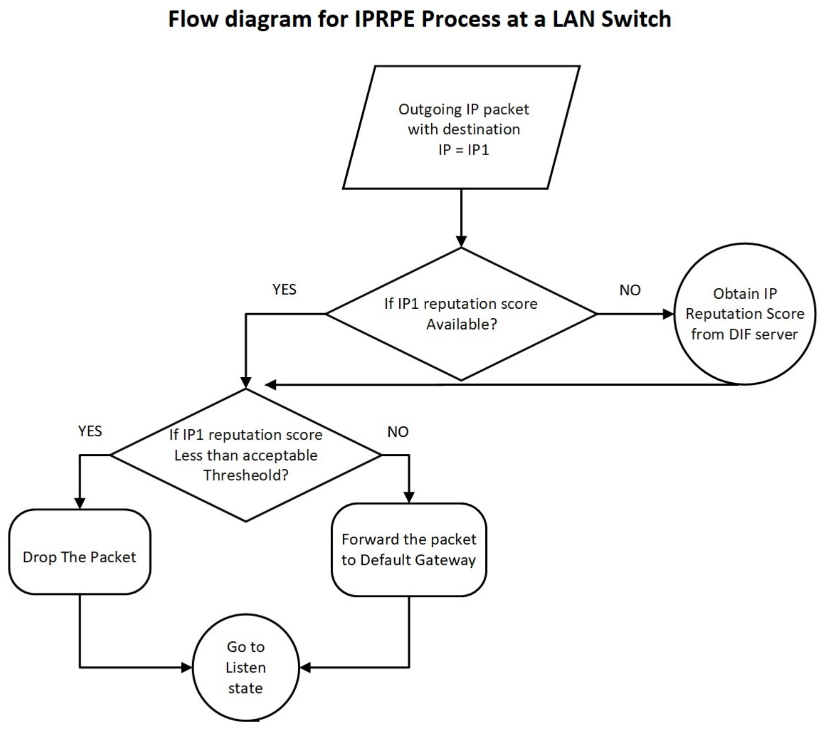

Flow diagram for IPRPE Process at a LAN Switch
Outgoing IP packet with destination IP = IP1
If IP1 reputation score Available?
YES
NO
Obtain IP Reputation Score from DIF server
If IP1 reputation score Less than acceptable Thresheold?
YES
NO
Drop The Packet
Forward the packet to Default Gateway
Go to Listen state

Figure 8.4: The Process flow diagram at a LAN Switch for the distributed DIF architecture.

the number of IP flows. Due to advances in solid-state technologies, high-speed storage is not an issue. Whenever an IP packet arrives at a switch, this table kept at the switch needs to be searched to find a match. If an entry is not available, the respective LAN switch must request the corporate DIF server to fetch the reputation from the Global server. The worst-case time complexity for the database search is $O(n)$, where n is the number of entries kept in the table. We now need to find various ways to optimize the storage and the search complexity. There may be some IP address that is not in use for a longer time. Thus, removing their entries from the table might be efficient to conserve the storage space and reduce the search time. To achieve this, we introduced various flags. We now discuss them in-depth.

The Time-to-Live value (TTL): The TTL is defined for an IP reputation entry kept at the corporate DIF server. This value is not applicable to the LAN switches. The TTL value specifies how long (in seconds) the reputation for a particular IP address/domain is valid before the corporate DIF-server contact the Global reputation system to obtain any update on the reputation. This timer is built for dynamism. This value is set by the Global reputation system, which knows how frequently the reputation may change. Let us discuss the case when there is a change in the reputation for a particular

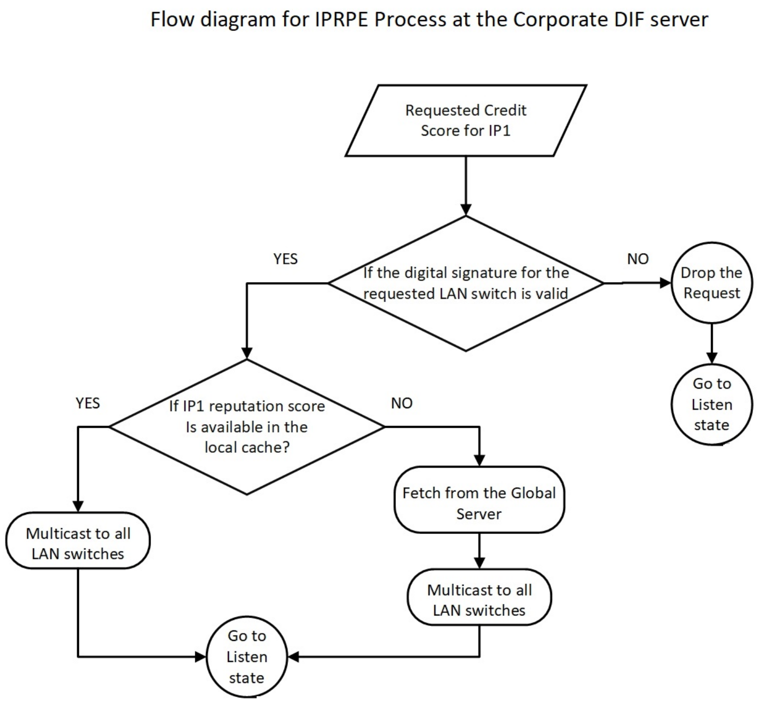

Flow diagram for IPRPE Process at the Corporate DIF server
Requested Credit Score for IP1
If the digital signature for the requested LAN switch is valid
YES
NO
Drop the Request
Go to Listen state
If IP1 reputation score Is available in the local cache?
YES
NO
Multicast to all LAN switches
Fetch from the Global Server
Multicast to all LAN switches
Go to Listen state

Figure 8.5: The Process flow diagram at the Corporate DIF Server for the distributed DIF architecture.

IP address/domain within the expiry of this TTL period. In that case, the Global DIF system will issue an "unsolicited" update to the corporate DIF server. The corporate DIF-server will, in turn, inform all the LAN switches of this change.

The TTL value does not accelerate the search time or reduce the storage. However, this value provides the recency of the IP reputation.

Within the duration of the TTL window, the reputation for an IP address may change from "allow to deny" or "deny to allow". This will mostly happen to borderline nodes. Once this change happens, the update must be communicated to every corporate DIF server. This is done through "Unsolicited updates".

Unsolicited updates can only be provided by the Global DIF system. The Global DIF system can monitor the IP reputation of every address constantly.

The DIF Stale timer: This value applies to every individual IP reputation entry stored in every LAN switch. This parameter is used for efficient memory management and searching. A sample is provided in Table 8.1.

Whenever an IP address/domain reputation is received from the Global DIF-server, it is stored in a local database at the corporate DIF-server and subsequently at all the LAN switches. As more entries are added, the table size grows in a "linear" order and the table-searching. Whenever a packet arrives at a switch S, the switch needs to search this

IP/Domain	DST
IP_1	s_1
IP_2	s_2
$\cdots$	$\cdots$
IP_n	s_n

Table 8.1: DST Table

IP/Domain	S_1	S_2	$\cdots$	S_k	TTL
IP_1	1	0	$\cdots$	1	t_1
IP_2	1	1	$\cdots$	1	t_2
$\cdots$			$\cdots$		$\cdots$
IP_n	0	0	$\cdots$	1	t_n

Table 8.2: TTL Table

table to make a forwarding decision. We do not want to store entries that are not used for a longer time. Thus, the DIF-Stale-timer (DST) represents the number of seconds such that if no end-host connected to a switch S is sending any IP packet to a destination IP address/domain IP_1, the entry will be removed from the DIF-cache table from S.

A sample of a DST table is presented in Table 8.1. For each stored IP address/domain, there is a countdown timer. We also introduce a small buffer time Δ. If the countdown timer + Δ become zero for a particular IP address, its entry is removed from the DST table. If the switch requires the reputation for the same IP address again, it needs to request the corporate DIF-server again. As soon as the entry is removed from S, S will also send a unicast message to the corporate DIF-server that the reputation entry for IP_1 is removed. There may be an active IP flow between another switch and IP_1. The corporate DIF-server maintains a table as in Table 8.2 to record active IP-flow information between various switches and IP domains.

The TTL-table contain IP/domain information, a list of all registered switches in the network, and a TTL countdown timer. For each IP address, the value 1 for a switch S_i signifies that the particular IP reputation is in use at the switch S_i. If this value is 0 for IP_1 for every switch, then no switch has any active connection with IP_1, and their DST is expired.

If there is no active IP-flow between any end-host and IP_1 for the expiry of DIF-stale-timer, the corporate DIF-server will remove IP_1's entry from its DIF-cache table once its TTL value expired. Once the entry is removed, the corporate DIF-server will not contact the Global reputation server to renew IP_1's reputation. Doing this way will reduce the number of messages exchanged between the Global and the corporate DIF servers. It will also save the DIF-cache table space.

The DIF-stale-timer is a count-down timer that is vital for network performance. The

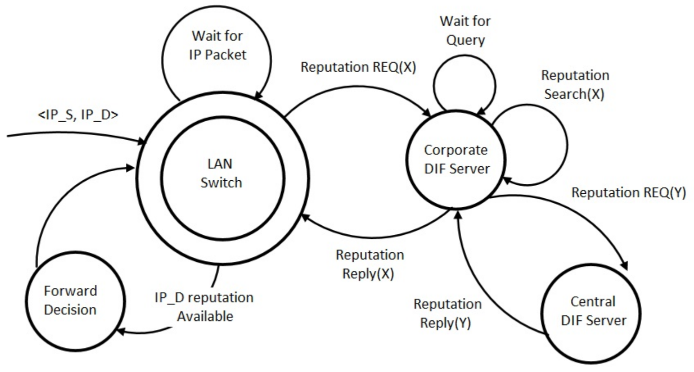


Figure 8.6: DIF "request-response" process.

network administrator sets the maximum value. This value is universal and applies to every IP flow and the LAN switch. Whenever an end host sends an IP packet towards a particular IP destination/domain, its DIF-stale-timer is "reset" to the default "maximum value". As soon as the DIF-stale-timer becomes zero for a particular IP address, the LAN-switch will remove the entry from the DIF-cache table, as we discussed before.

Unsolicited Update:

The Global DIF-server can only provide this service. The corporate DIF-server will constantly check for updates from the Global IP reputation server. Suppose the corporate DIF-server received an update from the Global server that changed a reputation from either "allow" to "deny" or the other way around. In that case, this information needs to be communicated to all the network switches immediately. Under this scenario, the DIF-server will issue an "unsolicited multicast update" to update all the switches. The DIF-server will digitally sign this message. The digital signature ensures that no rogue switch can launch a DoS attack by denying access to every possible IP address. Switches receiving this update will verify its authenticity first; once it is verified, they will immediately update their DIF-cache.

The state-transition diagram for the IP reputation request-response and DST process is given in Figures 8.6 and 8.7

Advantages of the Distributed mode: We now present some of the important advantages of the Distributed DIF operational mode.

- In this mode, IP reputations are checked at every first-hop switch. If the destination IP address has any malicious history, the packets are dropped. Thus, there is no need for any malicious packet to travel across multiple hops until the corporate firewall detects it.

- In a distributed DIF infrastructure, malware heartbeat can be detected at the first

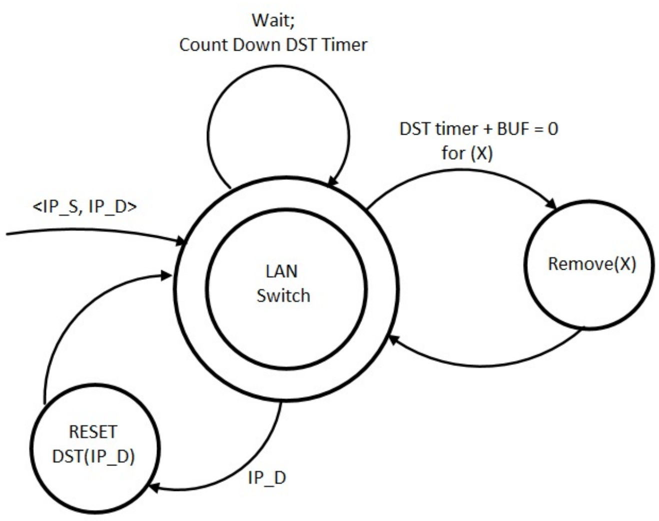


Figure 8.7: The DST process.

hop itself. The early detection process prevents the spread of malware to every host in the network. Similarly, the spread of worms can be prevented in the network through the distributed DIF infrastructure.

Whenever an end-host exhibits malware or worm infected behaviour, its MAC address is immediately noted. Connection from this host to another host in the network is blocked through "dynamic MAC filtering" at switches. Even if the host moves to another part of the network, its connection will be blocked. This process prevents the infection from spreading to other hosts within the corporate network.

- Whenever a switch is down, the rest of the network can function normally.

Let us discuss some of the disadvantages of the Distributed mode. Some of the disadvantages are common to the *PrECast* and the *NAT++* infrastructures.

Disdvantages of the Distributed mode:

- Every LAN switch and the corporate DIF servers must form a multicast tree in the distributed DIF mode. Thus, the LAN switches must be multicast-capable.
- Every LAN switch must be able to store and check the IP reputation score for every IP flow. Thus, the switches must be operating in Layer-3 of the OSI stack.
- Every LAN switch must have sufficient CPU and memory capacity to perform the IP filtering operations without incurring colossal latency

The distributed DIF mode has the following disadvantage based on human resource requirements:

- More expertise and human resources are needed to configure and maintain the DIF operations effectively.

8.3.2 The Centralized mode

The distributed DIF operation has few disadvantages, as we outlined before. The LAN switches must have the handling capabilities to perform the DIF operations at high speed. Another consideration is the human factor. The distributed DIF operation requires an experienced network manager who can create an efficient multicast tree, public, private key pairs, maintain the crypto key-systems and LAN hardware.

In several organizations, the above requirements may not be satisfied. Even though the required hardware is available, they may not have experienced network managers or cannot recruit their time to manage the distributed system.

To address the disadvantages posed by the distributed DIF system, we now propose the Centralized DIF mode. The centralized mode retains several critical features of the Distributed DIF mode. However, it does not require the overhead due to tree creation, cryptosystem, and human resource requirements.

We now describe the operation of the centralized DIF mode. Unlike the Distributed mode, the centralized more does not require the pre-processing stage. Thus, there is no tree creation stage, installing keys and maintaining IP reputation scores at every LAN switch.

The centralized DIF mode has the following stages:

- Filtering through IP reputation score check (Stages 1 and 2).
- Storage and search efficiency (Stage 3).

The centralized model is similar to the distributed model; however, the corporate DIF-server makes a decision rather than individual LAN switches.

Let a host H connected to a switch S send a packet with a destination address IP_1. As with a traditional campus LAN network, switch S will forward the packet to its default gateway. At some point in time, this packet will reach the corporate DIF server.

The corporate DIF server will check its local database if an IP reputation score is available for IP_1. If the reputation score for IP_1 is not available, then the corporate DIF server will request the score from the global DIF server. The IP packet will be placed on a buffer queue until the reputation score for IP_1 is obtained from the global DIF server. Once the reputation score for IP_1 is received, a decision is made. If the reputation score for IP_1 is above the required threshold, the packet is forwarded to the next hop. If the reputation score is less than the set threshold value, the IP packet is dropped.

Storage and Search Efficiency: This stage is like Stage 3 of the Distributed mode. However, the DIF Stale timer does not apply to this infrastructure. Only the TTL value applies to this infrastructure. The procedure for updating the TTL value is similar to Stage 3.

We now discuss some of the advantages and disadvantages of the centralized DIF infrastructure.

Advantages:

- Centralized DIF infrastructure does not require the complex pre-processing stage such as multicast tree creation, creating and maintaining public and private key pairs for all LAN devices. Thus the overhead involved in the pre-processing stage is reduced.
- There is no need to maintain an IP reputation score at every LAN switch. Thus, this mode does not require any special switches. Similarly, there is no need to maintain the DIF-stale timer at every switch.
- Centralized mode consumes less human resources and does not require experienced network administrators.
- There is no need to make any changes to the existing infrastructure. A centralized DIF server can run as a "software service" in a server. This configuration is similar to corporate proxy servers.

We now discuss some of the disadvantages.

Disadvantages:

- A malicious payload or a connection towards a malicious host needs to travel to the corporate DIF server before making a decision. Thus, an infected machine may spread its infection to other hosts in the network.
- An infected machine may freely roam to every subnet and spread its infection to other hosts in the network. It is hard to stop the spread of this infection.
- The corporate DIF server will be a single point of failure. Thus, the server needs to be protected with redundant architecture.
- Several vulnerabilities highlighted in the previous chapters are also true for this mode.

8.3.3 Protecting the corporate DIF server

Both in the centralised and the distributed DIF infrastructure, the corporate DIF server plays an important role. Without the corporate DIF server, the entire DIF operation will fail. Thus, the DIF server needs to be protected from both internal and external networks from attacks. *PrECast* along with the *NAT++* infrastructure protects a corporate network from various internal attacks. The proposals we present in this subsection will add an extra layer of protection to the DIF server.

Protection from the Internal Network:

Any malicious insider will stop the DIF operation (in either mode) by compromising the DIF server. In the case of a distributed DIF mode, a malicious insider may compromise a LAN switch and launch an attack against the DIF server from a compromised switch. The following steps are taken to ensure the protection of the DIF server from the Internal network. This proposal is crucial where *PrECast* and *NAT++* is not installed in a network. However, if *PrECast* along with the *NAT++* infrastructure is implemented in a corporate network, this proposal adds an extra layer of security to the DIF server.

- The server may fail due to non-attack related factors. Thus, it is recommended to have a redundant server in the network. This will ensure the availability of a corporate DIF server at all the time.
- Any direct connection coming from an end-host towards the DIF server is denied. Let IP_{DIF} is the IP address of the corporate DIF server. By providing less than the threshold credit score for IP_{DIF}, by natural means, any connection for a host towards the DIF server is denied. This process works both in the centralized and the distributed modes.
- The validity of the session key and the digital signature for every unicast packet originating from a LAN switch is checked. If the keys are invalid, the packets are dropped. Apart from this, any switch requesting services other than IP credit score from the DIF server is denied.

Protection from External Network:

Protecting the DIF server from the external network is critical. The following steps ensure this protection.

- Attackers may launch a DDoS attack against the corporate DIF server to paralyse the network. Thus, flood control is enabled on the corporate firewall. Primarily, explicit flood control firewall rules are written for the DIF server.

- Only connection coming from the Global IP reputation server is allowed towards the corporate DIF server. The firewall will check the session key and the digital signature of the Global IP reputation server before allowing the packet towards the corporate DIF server. If the session key or the digital signature fails, the Global IP reputation server packet is dropped.
- Connections from IP addresses other than the Global IP reputation server towards the corporate DIF server are denied at the firewall.

8.4 Performance Analysis

In this section, we analyse the performance of the DIF infrastructure in terms of latency introduced due to active IP filtering and IP lookup.

In any network communication, the packet latency is introduced during the following stages.

- **During a connection establishment stage:** In a UDP (or a pure IP) based connection, the latency may occur before a packet is sent to a destination for the first time. In a TCP based connection, the latency could happen before the three-way handshake.

  The latency that occurs before a connection is established is tolerable by most users. This type of latency is similar to launching an application using a slower (and older) CPU.

- **Latency in a live, ongoing IP connection:** This type of latency occurs due to the routing process or the firewall checking every packet based on specific keywords, source, destination IP address, TCP ports and protocol. The amount of latency added to every IP packet is "non-deterministic" and depends on the load on the firewall (or on the intermediate devices).

  This type of latency, called "jitter", is not suitable for a live network connection. Jitter is unhealthy for a real-time voice and video connection. Users may also experience unresponsive connections. Thus, jittering will cause a bad user experience.

The DIF infrastructure checks the IP reputation score only during the connection establishment stage. Once the packet is "allowed", the same decision is made until the end of the connection. This process is similar to a "fast switching" process in routing. Thus, the DIF infrastructure does not add any "jitter" to a live connection. This is another advantage of our proposal.

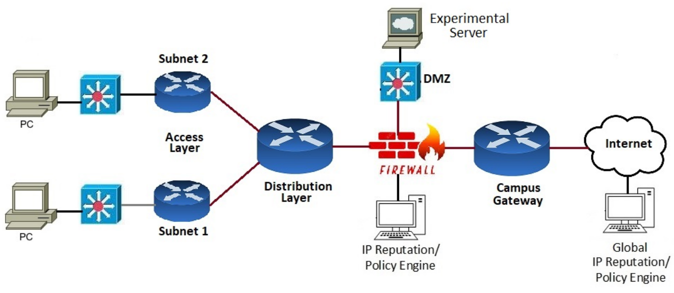


Figure 8.8: Experimental Topology.

We now estimate the latency during the connection establishment (IP reputation score lookup) stage.

Experimental Setup:

In order to evaluate the performance of the proposed DIF infrastructure, we created a small campus network given in Figure 8.8. We have connected 12 PCs each to Subnet 1 and Subnet 2 through Cisco 3560 switches. Cisco 2800 series routers were used at the access layer. ISR4000 series routers were used for the distribution layer and the gateway. We obtained 5000 random destination IP addresses from our live network. For these 5000 random IP addresses, we determine the IP reputation score using Talos. For HTTP and email servers, Talos provides the following hierarchical classification based on threats. They are “trusted”, “favourable”, “neutral”, “questionable”, “untrusted” and “unknown”. If either HTTP or email reputation is below “neutral”, we classify the IP reputation as “deny”. If both the reputations are “neutral or above”, we classify the IP reputation as “allow”.

The IP addresses and their associated reputation recoreds are stored in a data file. The data file is located in a remote server located approximately 7000 miles apart from our campus network. This server emulates the Talos system. Since the real-time Talos system is available only based on “subscription”, we simulate here the offline operation.

We created a static routing table for all these 5000 random IP addresses to direct them towards the experimental server kept within the campus LAN. The experimental server will serve every request made from the 24 PCs connected with Subnet 1 and 2.

Traffic Modelling:

Each PC initiates a connection (randomly chosen between TCP, UDP or IP) to one of the 5000 IP addresses from the pool. In reality, this could be any Internet address, and the reputation for this address is readily available with the Global reputation server. Each connection last for 120 seconds. Once a connection is torn down, the host waits for a random second (follows an exponential distribution with a mean of 300 seconds) before making another connection with another random IP address from the pool. For each connection request, the additional latency due to the DIF infrastructure is recorded.

The proposed DIF infrastructure introduces three different types of latencies.

- When the requested IP reputation is available in the local cache of a LAN switch, an “allow” or “deny” decision must be taken by the LAN switch. This process requires searching the local database and make a decision. The “search” process introduces latency.
- Whenever the request IP reputation information is not available with a LAN switch, the switch needs to fetch the reputation information from the campus DIF server. This latency must be more than if the reputation information is available with the local LAN switch.
- When the requested IP reputation information is not available with the corporate server, the Corporate DIF server needs to fetch the IP reputation score from the Global IP reputation server. This increases the latency.

8.4.1 Latency Due to the DIF Infrastructure

The introduction of a new packet filtering in the existing network adds extra latency. We wish to evaluate the minimum, maximum, and average latencies introduced by the DIF infrastructure through this experiment. We run the experiment with and without the DIF infrastructure to evaluate the additional latency due to the DIF infrastructure.

The extra Latency due to the Distributed DIF Infrastructure:

Depending on the destination IP address, the distributed DIF infrastructure introduces three different classes of latencies. These latencies depend on whether the IP reputation information is available with a local LAN switch, or it needs to be fetched from a corporate DIF server and is available with the corporate DIF server. Lastly, the requested IP reputation score is unavailable with the corporate DIF server and needs to be fetched from the Global IP reputation server.

For the experimental topology given in Figure 8.8, we compute the three different types of latencies introduced by the distributed DIF infrastructure.

A forwarding decision is taken immediately when the IP reputation score is available with the LAN switch for an outgoing IP packet. The additional latency incurred in this case is only due to searching its local IP reputation cache. The minimum, the maximum and the average additional latencies are 1ms, 4ms and 2.469ms, respectively.

We now discuss the case when the IP reputation score is not available with a LAN switch. However, we assume that the corresponding IP reputation score is available with the corporate DIF server. In this scenario, a LAN switch must buffer an outgoing IP packet until its reputation is available locally. This process involves the following steps:

- The LAN switch needs to unicast the IP reputation request to the corporate DIF server.
- The corporate DIF server needs to search its local reputation database, fetch the information and multicast to all LAN switches in the corporate LAN.

Thus, all the above tasks introduce considerable latency than there is no DIF infrastructure.

From our simulation, the minimum, the maximum and the average additional latencies are 41ms, 47ms, and 43.65ms.

The third scenario is that the requested IP reputation is not available with the corporate DIF server, and the corporate DIF server needs to fetch the information from the Global IP reputation server. The corporate DIF server will multicast to all LAN switches as soon as the IP reputation is received from the Global IP reputation server.

In this scenario, the minimum, the maximum and the average additional latencies are 151ms, 165ms, and 157.73ms, respectively.

We summarized all these latencies in Table 8.3.

Task	Min (ms)	Max (ms)	Average (ms)
IP reputation score is available with the LAN Switch	1	4	2.467
IP reputation score needs to be fetched from the corporate DIF Server	41	47	43.65
The corporate DIF server needs to fetch from the Global DIF server	151	165	157.73

Table 8.3: The Minimum, Maximum and the Average Latency due to the Distributed DIF Infrastructure

We now discuss the additional latencies that arise due to the centralized DIF infrastructure. The corporate DIF server itself takes a forwarding decision in the centralized DIF infrastructure.

If the IP reputation score is available for the arriving IP packet, the decision is taken immediately. In this scenario, the central DIF server needs to search its local IP reputation database for a match. This process is similar to the scenario when the IP reputation information is available with a LAN switch. Thus, the minimum, the maximum and the average additional latencies are 1ms, 5ms, and 3.12ms, respectively.

Whenever the IP reputation is not available with the corporate DIF server, it needs to fetch the IP reputation information from the Global IP reputation server. A forwarding decision is taken as soon as the corporate IP reputation server receives the IP reputation. Compared with the distributed DIF infrastructure, there is no need to multicast to any LAN switches. The average additional latency, in this case, is 115.50ms. The minimum and the maximum additional latencies are 108ms and 125ms.

We summarize these latencies in Table 8.4.

Task	Min (ms)	Max (ms)	Average (ms)
IP reputation score is available with the corporate DIF Server	1	5	3.12
The corporate DIF server needs to fetch from the Global DIF server	108	125	115.50

Table 8.4: The Minimum, Maximum and the Average Latency due to the Centralized DIF Infrastructure

8.5 Chapter Summary

In this chapter, we proposed a framework for efficient IP filtering at the first of a network based on the reputation of a destination IP address. We called this framework *Dynamic IP Filtering.* The DIF can operate in two modes, namely the distributed mode and the centralised mode. Excessive latency and jitter are not suitable for a healthy network. The DIF architecture introduces only a minimal latency during the connection establishment stage. The proposed architecture will not introduce jitter in the network. Thus, the DIF infrastructure will not adversely affect user experience in the network.

Our real-time simulation results show that the latency introduced by the DIF infrastructure is within the acceptable limit. We run the experiment in a public network where there is no QoS enforced. However, if there is a tunnel between the corporate DIF server and the Global IP reputation server with adequate QoS, the latency can be significantly reduced.

The centralized and the distributed modes have their advantages and disadvantages. The distributed mode is superior compared with the centralized mode. The distributed mode can stop the spreading of malware (based on heartbeats) and worms (based on rapid socket connections). Any end-host can be dynamically isolated from the network using MAC filtering. A blocked MAC address can be multicasted to every LAN switch. Thus, a compromised node cannot freely roam around the network. However, this feature is not possible in a centralized DIF infrastructure. However, both the modes stop connection to an IP host that has less reputation. Thus, even an ignorant user clicks a link in a phishing email, the connection to a malicious site is thwarted.

Whenever a user URL is redirected to another site (in case of a URL redirect attack), the reputation of the new IP destination address is checked. If its reputation is low, the connection is aborted. Thus, the DIF infrastructure stops URL redirection attacks. Similarly, the DIF infrastructure stops malicious adware and malicious popups.

Since the corporate DIF server is the brain behind the DIF operations, it becomes the central point of attack. The attack can come from both internal and external networks. We use the proposed DIF filtering technique to thwart attacks coming from internal end-hosts. Any attacks coming from a compromised LAN switch in case of a distributed mode is stopped effectively. Explicit firewall rules are enforced for the DIF server to restrict connections from an external network. Only connections coming from the Global IP reputation server is permitted after checking the digital signature.

Chapter 9

SDMw: A Secure Dynamic Middleware for defeating port and OS scanning

In Chapters 6 and 7, we provided the *PrECast* framework and *NAT++* to solve various security-related problems in a LAN environment. Chapter 8 provided an enforceable dynamic IP filtering mechanism. The dynamic IP filtering mechanism work based on the reputation of the destination IP address. Critical hosts such as the DNS servers, Gateway, email, and web servers will have public IP addresses in a corporate network. Since they offer essential services to both inside and outside hosts, their connection from the external network cannot be blocked. Any outsider with a malicious intention can fingerprint these devices. They can identify the OS running on these devices, information on various security patches applied to these devices, and the critical services running on these devices. Based on this information, they may be able to launch attacks now or a zero-day attack whenever a security hole is found at a later point in time. *PrECast*, *NAT++* and the *DIF* infrastructure cannot solve this problem. In this chapter, we provide a dynamic middleware called SDMw to solve these attacks. The SDMw framework solves, insider, outsider and collaborator attacks that are due to fingerprinting.

Fingerprinting is a process of identifying the remote network devices and services running on the devices, including operating systems (OS) of the devices, and hosts running different OSs. Several research proposals and commercial products are available in the market to defeat fingerprinting. However, they have performance limitations and expose themselves to attackers. In this chaper, we utilize some real-time fault-tolerance concepts (viz. real-time/dynamic, detection/locating, confinement/localizing and masking/decoy) to propose a *plug-and-play* adaptive middleware architecture called ***Secure Dynamic Middleware*** (*SDMw*) with a view to defeat attackers fingerprinting the network, without exposing itself to the attackers. We verify that the proposed scheme works seamlessly and requires zero-configuration at the client side.

The contents in this chapter are derived from the candidate's publication Hanna, Veeraraghavan, Soh [37].

9.1 The Prelude

As we mentioned in the Introdutory chapter, the *Internet* uses the Transmission Control Protocol and the Internet Protocol (*TCP/IP*) suite of protocols that were designed with no security in mind. Both the research community and developers of this protocol did not anticipate this exponential growth. Thus, these protocols offer minimal protection against eavesdropping.

Fingerprinting a network equipment has several potential applications and benefits in network management and security. It is also useful to understand the network structures and their behaviour. However, to penetrate a corporate network, it is essential to fingerprint their network topology, the software system and the services used in the network. Fingerprinting is done either from a single remote computer connected to the Internet or from a group of computers. The attackers often collaborate from different parts of the world to launch an attack against a network. The attackers use either a spoofed *IP* address or some compromised proxy servers to hide their identity. To launch a distributed scanning, the attacker may install zombies and control them from a central remote computer.

Several research proposals and commercial products are available in the market to defeat fingerprinting. Some of them have performance limitations, whereas others expose themselves to the attacker. In this chapter, we propose a *plug-and-play* adaptive middleware architecture to defeat attackers fingerprinting a network, without exposing itself to the attackers. We also demonstrate that our proposed scheme works seamlessly and requires zero-configuration at the client side.

In our earlier chapters we provided the integrated ***PrECast*** and ***NAT++*** framework to protect a corporate network from various attacks. However, fingerprinting the devices are still possible from the external network. Based on device fingerprinting, attackers may launch DDoS or zero-day against important network hardware to stop the operations. Thus, augmenting fingerprint spoofing mechanism proposed in this chapter with the integrated ***PrECast*** and ***NAT++*** framework will protect the network from external attacks.

9.2 Network and Host Fingerprinting: An Attacker Perspective

A precursor to a *port scanning attack* is network and host fingerprinting. Our proposed architecture will focus on these two elements: fingerprinting and port scanning. We will review port scanning in Section 9.3, and, in this section, we review the state-of-the-art in network and host fingerprinting from the attacker's viewpoint. Network security is a critical element in any organization; however, security has become one of the primary concerns of computer professionals and organizations today. The aim of network security is to protect private data and also to ensure that the network resource and Internet connections are not being compromised and misused.

To intrude into any network, as the first task, an attacker needs to fingerprint the network. As human fingerprints uniquely identify a person, we can identify an *Operating System* on a network through specially crafted *TCP* and *IP* packets (called packet fingerprinting) with specific parameters. The *TCP/IP* specification has several ambiguities dealing with initial parameters. Different *OS* vendors interpret these ambiguities in a different way. By analysing initial values of *protocol flags*, *options* and *packet fields*, we can determine the *OS* installed on a host. Based on the *OS*, we can also discover the hardware (like switches and routers). Even though fingerprinting a network is not seen as an attack, they often serve as a prelude to further attacks by exploiting known bugs in the remote host's *OS*.

Fingerprinting is done either actively or passively. Active fingerprinting involves sending some specially crafted *TCP/IP* packets to the target network and observing their reply. The replies are then matched against known *Operating Systems* and *Services.* A malicious use of the fingerprinting technique is to construct a database of a noted *IP* address and the corresponding *Operating Systems* for an entire network. When someone discovers a new exploit, the attacker can target only those machines running the vulnerable *OS*. Almost every system connected to the *Internet* is vulnerable to fingerprinting, including *webcams*, *fax machines*, *printers* and so on.

The *TCP/IP* protocol was designed in the 1970s and has undergone several changes. However, it still has several design ambiguities. Both of these protocols are described in their respective *Request For Comments* (*RFCs*), (791 and 793 for *IP* and *TCP*, respectively) for developers to read and understand when implementing. However, there are areas where the *RFCs* leave certain decisions to the developer. The protocol specification did not specify how certain modules need to be implemented, especially when dealing with initial parameters and the use of specific flow control algorithms.

1 *Time-to-Live (TTL) Value*: Several *OSs* set different initial *TTL* values. For example, *Windows 10* (Microsoft Corporation, Redmond, WA, U.S.A) uses a default value of 128; *Linux* use 64 and *Solaris* (Oracle Solaris, Redwood Shores, CA, U.S.A) use 254.

2 *TCP-Initial Sequence Number (ISN)*: Even though almost every *OS* uses non-guessable *ISN* (after Kevin Mitnik's famous attack), the microscopic drift in the Central processing unit (CPU) clock cycle is used in the literature to enumerate the number of hosts inside a Network Address Translated (*NATed*) network.

Even though every host behind a corporate network uses *Network Time protocol* (*NTP*) to synchronize their clocks, there is always a microscopic drift in their clock. This is due to inaccuracy in the clock crystals. The *TCP-ISN* generation algorithm uses the system clock information for the randomization process. For a host outside a *NATed* network, every connection appears to originate from a *single IP* (or a small pool of *IP*) source. However, the microscopic drift in the individual computer's *Realtime clock* inside a *NATed* network is exploited to count the number hosts behind a *NATed* network.

3 There are few *TCP* options that will not reveal the *OS* information by itself. However, it can be used along with other parameters to narrow down the *OS* detection. They are *TCP-timestamp* option, *Window Scaling*, *Maximum Segment Size (MSS)*, and *Explicit Congestion Notification* (*ECN*).

4 *IP-Identification field (IP-ID)*: The *IP-ID* field is a 16-bit field in the IP header. IP packets can also be tagged with a *Don't Fragment* (*DF*) bit, in which case the *ID* field can be ignored. Since the source host has no knowledge about the *path-MTU* (maximum transmission unit), every *IP* packet contains a unique *IP-ID* value. *Windows* 95, 98 and *NT* family increment the *IP-ID* field by 128, rather than 1. From *Windows* 2000, it was set to 1 like any other *OS*. Currently, *IP-ID* did not reveal much of information about *OS*; however, different *IP-ID* mappings are used to enumerate the number of hosts inside a *NATed* environment. When replying to an *ICMP-Echo* message, *Linux* kernel use an *IP-ID* value of 0 and the *Don't Fragment* (*DF*) Flag set to 1. This makes it easy for the attacker to identify *Linux* kernel.

5 If a TCP-synchronize (*TCP-SYN*) or a TCP-Finish (*TCP-FIN*) segment is sent to a closed port, different *OS* running on a host will behave differently. This will facilitate *OS* detection.

Traditionally, NAT was designed to address solution to the problem of *IPv4* address depletion. Today, it is used for several different purposes, including the protection of the corporate network. Since *private IP* addresses are *non-routable*, they cannot be used as a *destination* address in the *Internet*. Thus, an attacker may not be able to fingerprint hosts that are behind a NATed network. Because of this unique feature, the use of *NAT* is widely preferred for hiding a corporate network through the use of *private IP* addresses. Obtaining information about hosts behind a *NATed* network is an interesting research area. Several research articles are available in the literature on counting the number of hosts behind a *NATed* network.

There are several commercial and free-to-use tools available for fingerprinting *OS*. However, the most complete and the widely used *OS* fingerprinting tool is the ***Network Mapper*** (*NMAP*) [36]. It uses a database of over 500 fingerprints to match various *OS* and hardware platforms. Fingerprints for new hardware and *OS* are added everyday through the online community [36]. The *NMAP* system identifies the target *OS* and hardware by sending up to fifteen *TCP*, *UDP* and Internet Control Message Protocol (ICMP) probes. These probes are specially crafted packets designed to exploit various ambiguities in the standard *RFC* implementation of the protocol. *NMAP* compares responses with the known pattern for finding out the remote *OS*.

In [48], Khono et.al. used the microscopic drift in the clock-skew to fingerprint remote devices, rather than *OS*. Using their proposed method, an attacker can remotely probe a block of address to determine if it corresponds to virtual hosts. We note that the block of addresses must be in a public *IP* range. In addition, to improve the accuracy, Khono's algorithm needs long-term data. Thus, it is efficient to use tools like *NMAP* at first to narrow down the search before launching Khono's algorithm.

In a practical environment, it is not possible to use several packets for fingerprinting a host (or *OS*). In a recent paper, Shamsi et al. [82] proposed a single packet *OS* fingerprinting. As mentioned in their work, the current fingerprinting tools like *NMAP* are not well suited in the *Internet* environment due to large amount of traffic generated, and the intrusive nature of the probes. They presented a proposal to fingerprint an *OS* using a single *TCP-SYN* frame. They also build a database consisting of 116 *Operating Systems*. However, the downside is that new firewalls issue a TCP-Reset (TCP-RST) segment for any incoming connection to hosts that do not run any registered *TCP* services. Thus, in an unknown environment, Shamsi's proposed method may trigger several false positives due to firewalls issuing *TCP-RST* segments.

The above-mentioned methods use active fingerprinting. Using the above methods, an attacker can fingerprint any host that has a public *IP* address. Passive fingerprinting systems simply sniff the network traffic and opportunistically classify the hosts as they observe their traffic. This is more time-consuming and difficult to comprehend compared to active fingerprinting. Most of the time, the traffic sniffer may not be on the active-path between the source and the destination host to sniff their traffic. However, it is hard to detect the presence of a passive system in the network. Fingerprinting works well in distinguishing different types of known Operating Systems or Network Services, but it may not detect different versions of the same operating system or network service.

9.3 Port Scanning

Once the fingerprinting is complete, an attacker will construct a database of *IP* addresses and the corresponding *Operating Systems* for an entire network. When someone discovers a new exploit, the attacker may target only those machines running the vulnerable *OS*. The next step is to perform a port scanning. It is the process of probing a host for open ports and other services running on the host. Several *TCP* and *UDP* services listen at various ports for possible incoming connections. For example, The Hypertext Transfer Protocol (HTTP) servers listen for connection at port 80. Some ports may easily be exploitable compared to other ports. Services listening on these ports may allow under privileged connections to manipulate system files or install additional services. Port scanning is a method for discovering exploitable communication channels.

Port scanning is divided into two main parts: *Horizontal* and *Vertical* scanning. In a horizontal scanning, the same port (number) is scanned over multiple hosts in the same network. This type of scanning is useful for attackers who want to gain control of victim hosts by exploiting a known vulnerability of a certain service running on that port. In a vertical scanning, multiple ports are scanned on the same host machine. This technique is common for an attacker who is gathering information to fingerprint a host machine and to attack at a later point of time, for example, *Web servers*, The Domain Name System (DNS), Dynamic Host Configuration Protocol (DHCP) servers and so on.

There are several techniques available for surveying the open and closed *TCP/UDP* ports on a target machine. Each technique has its own strength and weaknesses. The most common ones are as follows: [32]

(1) **TCP-connect()**: This is a traditional way of making a *TCP* system call to establish a connection to an interesting port in a target host. If the destination host is listening on a particular port, the `connect()` request will succeed; otherwise, the port is unreachable. This method has the following two best advantages compared with other methods: the first advantage is that it does not require any administrative privilege to launch this request. The second one is the speed. The attacker can

hasten the scan using many sockets in parallel. However, the downside is that this type of scan can be easily detected and can be filtered.

(2) **TCP SYN scanning**: the attacker sends a *TCP-SYN* packet as if he is going to open a "real" connection and waits for the SYN-Acknowledgement (SYN-ACK). A *SYN-ACK* indicates that the port is open and listening. A TCP-Reset (TCP-RST) indicates that the port is either not available or not willing to accept a connection at this point of time. Since we are trying to establish a "Simplex" connection between the attacker and the target host, this technique is often referred to as "half-open" scanning. One of the disadvantages is that newer firewalls immediately issue a *TCP-RST* segment without even referring the packet to the destination host.

(3) **TCP-FIN Scanning**: This method is used to overcome the disadvantage of the *TCP-SYN* scanning. Newer firewalls and packet-filters watch for any *TCP-SYN* segment sent to a prohibited port (i.e. ports that are not open). Once a series of *TCP-SYN* is detected, they block the attacker's *IP* address either permanently, or for a timeout period. However, if an attacker transmit a TCP-Finish (TCP-FIN) segment to a closed port, the host tends to reply to the *TCP-FIN* segment with a proper *TCP-RST*. *TCP-FIN* segments may pass through the network without getting detected.

There are several other methods available in the literature like *Fragmentation attack*, *Reverse Indent scanning*, *ICMP-ping* etc. However, they are either less frequently used or patched by the latest firewalls. There are a number of free and commercial tools available that perform detailed port scanning. In contrast, *NMAP* is the most powerful scanning tool available for free-to-use [36]. *NMAP* can also be used for a number of other purposes. Even though there are a number of tools ready for the use of performing port scanning, there are very few tools available for detecting port scanning attempts.

Most of the solutions, including *Antivirus*, *Firewalls*, and an Intrusion Detection System (IDS), can indicate an occurrence of a port-scanning attack or an unauthorized port activity. As port scans are usually performed before an actual attack, indications of port scan activity give up a clue that an attack might follow in the future. Thus, having an intelligent system that predicts the occurrence of attacks soon is preferable.

In the recent past, port-scanning detection has received a lot of attention by researchers. Several intelligent algorithms using *Artificial Intelligence*-techniques were invented to detect possible port scanning attacks. Intrusion-Detection Systems use these algorithms to block port scanning attempts. For a detailed review on various port scanning techniques, one may refer to [33]. This book provides extensive treatment to network scanning. In [21], the authors have used data mining techniques for network intrusion detection. They built a class of prediction-model for identifying known intrusions and their variations, anomaly/outlier detection schemes for detecting attacks whose nature is unknown. They validated their model using experimental results on the *DARPA-1998* data set, as well as on live network traffic at the University of Minnesota. However, the validity of their model for detecting current intrusions need to be evaluated.

In [79], Soniya and Wisey used a Neural network approach to detect *TCP-SYN* scanning using packet counts. They trained a Neural Network to capture the behaviour for normal as well as port scan data. Their results show the normal and abnormal behaviour of *TCP-SYN*, *SYN-ACK* and *TCP-FIN* packet sequence. A deviation from the normal behaviour is used to effectively detect *TCP-SYN* scanning without maintaining a state information.

In [6], Baig and Karman used a time independent feature set to detect port scans. Their model can detect slow and random attacks in real time with reduced memory needs. Their proposed model was tested using *DARPA-99* data set. In [53], Leckie and Kotagiri used a probabilistic approach to detect port scans. Their model considered both the number of destinations and the ports accessed by an attacker, in order to determine how unusual the access is.

However, in all the above research work, or in commercial *IDS* systems, the authors and the developers assumed that the attackers scan the network in a short duration. That is, the interarrival rates between the successive scan packets are small. Even though this assumption was valid until few years back, it is not valid any more. An attacker may use either a proxy chain to launch a scanning or deploy zombies to launch distributed scanning. In both cases, every scan packet arrives from a different *IP* address. Thus, blocking a single *IP* address is not effective anymore. Nowadays, attackers use a combination of zombies along with proxy-chains to launch a more powerful attack, which is hard to detect.

9.4 SDMw: The Secure Dynamic Middleware Architecture

Due to increase in OS vulnerabilities, enterprise network administrators regularly perform *OS* audits. They must also audit for any open unauthorised ports running underprivileged services. The audits also help them in maintaining enterprise network policies. Network administrators use tools such as *NMAP* to perform this audit. Thus, *OS* fingerprinting and port-scanning are considered a healthy way to maintain enterprise security policies.

However, the hacking community use this mechanism for their advantages. If an enterprise deploys a commercial tool to defeat fingerprinting, then the tool itself will exhibit its behaviour to the attackers. In that case, an attacker may launch a distributed denial-of-service (DDoS) attack against the tool. Rather than defeating a fingerprint or port scan process, in this chapter, we propose the *SDMw* architecture for sending fictitious information about the network to the attacker.

SDMw is a *plug-and-play* adaptive middleware that will exhibit different OS characteristics at different times, with a view of defeating attackers fingering the network in a transparent manner.

9.4.1 The SDMw Architecture

As a first step in our research proposal, we collect several popular *OS* fingerprints. The *SDMw* system uses each specific *OS* fingerprint at different points of time, that does not reflect the host OS.

9.4.1.1 The Unified Header Generation Algorithm (UHGA)

We consider several popular *OSs* like *Windows*, *Linux*, *Solaris*, etc. As a first step, we adopt a unified algorithm to determine the *TCP/IP* parameters (including *IP-ID*, TCP-Initial Sequence Number (TCP-ISN), *timestamp*, the choice of a *congestion control* algorithm, *window size*, etc.) that are typical to the *OS* we considered. Let every host in the network use the same unified algorithm transparently. For an attacker, every host (including legacy hardware devices like printers, fax etc.) appears to run the same *OS*, (either known *OS*, in case we adapt known *OS* fingerprint parameters; or unknown *OS* if we design our own algorithm to decide on the *TCP/IP* parameters).

9.4.1.2 SDMw Placement

Ideally, *SDMw* is placed between a corporate network and the external world. Due to its light weight, it can run as a proxy service in any subnet or it can run as a proxy middleware in every host. However, placing the *SDMw* system as a proxy service in every subnet is the most appropriate placement. If the *SDMw* service runs on every host, then the network administrator will not be able to fingerprint a host *OS* for legitimate audit purposes. In addition, updating the *SDMw* system in every host consumes time and is unrealistic.

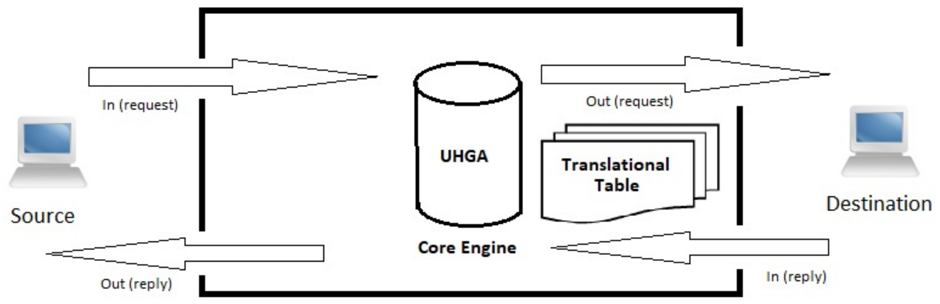


Figure 9.1: The Secure Dynamic Middleware black box.

9.4.1.3 The SDMw Black Box

The black box of the proposed system is shown in Figure 9.1. The core engine consists of a *pluggable* module that determines different versions of *OS TCP/IP*-kernel parameters at different periods of time. This implements the *UHGA*. The *in–out* table is a *translational table*, similar to the one kept in a *Socks-proxy* or a *TCP-intercept* system. For every pair of a source and a destination host, depending on the time, the *SDMw* system intercepts the connection and generates *TCP/IP* header parameters that appear to represent a fingerprint of a particular *OS* at that time. For different times, *SDMw* will use different *OS TCP/IP* parameters. By doing this way, either the *OS* detection by remote attacker is masked or provides ambiguous results.

We now discuss how *SDMw* handles connectionless and connection-oriented services originating from inside hosts.

9.4.1.4 Connectionless Datagram Service from Inside Hosts

UDP and *IP* datagrams offer connectionless services. The term connectionless means that *UDP* and *IP* do not maintain any state information about successive datagrams. Each datagram is handled independently from all other datagrams. For connectionless *UDP* services, *source IP address*, *source port*, *destination IP address* and the *destination ports* (called a *socket*) information are important. Each *UDP* connection is uniquely identified by this 4-tuple. An attacker may not be able to fingerprint a system just based on this parameter. Thus, the *SDMw* system preserve this 4-tuple without modification. If the *source IP address* and the *source port* information is changed (as in the case of a *Network Address Translation* (NAT) or a Port Address Translation (PAT) system), a *reference translational table* must be maintained in order for the reply to be forwarded to the right host. If a *NAT/PAT* is not used along with *SDMw*, the proposed *SDMw* system will not modify the original 4-tuple. Thus, it will improve the system performance by reducing a *table-lookup latency.*

At Layer-3, *IP-ID* field, *TTL*, *Explicit Congestion Notification* (ECN) and *Fragment flags* are traditionally used for fingerprinting a host. These parameters are chosen by the host's *OS* on per packet basis, and they need to be changed in order for a host to be protected from fingerprinting. If we assume that *IP fragmentation* is not needed along the path between a host and the *SDMw*, then changing them by the *SDMw* is not going to affect the communication between the end-hosts. Since these parameters are applied on per packet basis, there is no need for the *SDMw* to maintain a translational table. Our proposed *SDMw* architecture will generate these parameters according to the chosen *UHGA* algorithm and replace the one in the IP packet.

9.4.1.5 Connection-Oriented TCP Service from Inside Hosts

Even though *TCP* and *UDP* uses the same network layer (*IP*), *TCP* provides a totally different service to the application layer than *UDP* does. *TCP* provides a connection-oriented, reliable, byte stream service. Like *UDP* services, *TCP* connections are also identified by the 4-tuple, *source IP address*, *source port*, *destination IP address* and the *destination ports.* The term connection-oriented means the two applications using *TCP* (normally considered a client and a server) must establish a *TCP* connection with each other before they can exchange data. The logical connection is established and maintained by means of 32-bit *sequence* and *acknowledgement* numbers. This process starts with an *Initial Sequence Number* (*ISN*). *TCP* is very sensitive on establishing, maintaining and tearing a connection. An on-going connection will be terminated if there is a mismatch in the sequence number.

As before, from a socket, no host *OS* information is exposed to the attacker. However, compared to *IP* and *UDP*, *TCP* has several design ambiguities that are traditionally exploited to fingerprint a host. Our proposed *SDMw* architecture preserves the original socket information. However, we change other *TCP* parameters, including the sequence numbers that can mask the *OS* detection. Since we are changing the *logical connection* information, we need to maintain a *translation table* in order for the *reply* from the *destination* to be forwarded to the end-host properly without tearing an on-going connection. While deciding on the *congestion window* parameters, *SDMw* chooses the minimum of the one it generates based on the *UHGA* and the host's *advertised congestion window* parameters. Based on the particular *OS* chosen by the *UHGA* algorithm, the *TCP* parameters are determined on a per-socket basis. Entries in the translation table are maintained for $2 \times MSL$ (maximum segment lifetime) wait period before they are removed.

9.4.1.6 Connection from External Hosts to an Internal Host

Any external host may initiate either a connectionless, or a connection-oriented service to an inside host. Since *SDMw* is positioned between the external and internal hosts, like a firewall, any such connection must pass through *SDMw*. If an external host is initiating a connection to an unregistered port of a host, rather than the end-host handling this message, *SDMw* handles on behalf of the end-host. Traditionally, attackers fingerprint a remote host by sending probe packets to closed, as well as open ports. They also do port scanning in order to launch an attack. To protect the end host from fingerprinting and attacking at a later point of time, *SDMw* replies on behalf of end-hosts. Depending on the particular *OS* fingerprint chosen by the *UHGA* algorithm, the proposed *SDMw* architecture will handle the request as if the particular *OS* is running on the end-host. If an end-host is running a registered service, then *SDMw* will forward the connection request to the end-host. Thus, it is necessary to keep all the registered service information with the proposed *SDMw* system.

If an external host initiates a connectionless *IP* service such as *ICMP*, the *IP* packet will be dropped by *SDMw* without referring to the end-host. *SDMw* follows in a similar fashion if either an *UDP* or a *TCP* connection is initiated to an unregistered port. *SDMw* will allow only connection to a *publicly-registered port* (like *HTTP* ports).

Whenever an internal host requests a connection (or sends an *IP* packet) destined to an external *Host B*, the packet is received by the *ingress* link of the black box. Based on the source, destination and the current time, the *UHGA* algorithm will generate and replace the original *TCP/IP* header fields. Other application layer fields are kept intact. The source information, including the original *TCP/IP* fields are copied to the same row as in the incoming packet at the *Egress-column* and the packet is transmitted through the *outside* (*egress*) link. Once the *reply* packet is received, the *reverse* of ingress–egress entry is used to relay the reply to the inside host.

This process is similar to a *Socks-proxy* server. However, the main difference is that the *SDMw* system rewrites the entire protocol header fields to pretend as if the machine is using a certain *OS*.

For connectionless *UDP* and *IP* packets, it is not necessary for the *SDMw* system to use the *in–out* table. However, for *TCP* connection, the system must maintain the *in–out* table for forwarding purposes. Figure 9.2 depicts the operation of the *SDMw* system when using a *TCP* connection. This process is similar to *TCP-intercept* or a *Socks proxy* server.

In communication, deadlock occurs when *Process A* is trying to send a message to *Process B*, which is trying to send a message to *Process C*, which is trying to send a message to *Process A*. This type of deadlock must be handled by the Operating System.

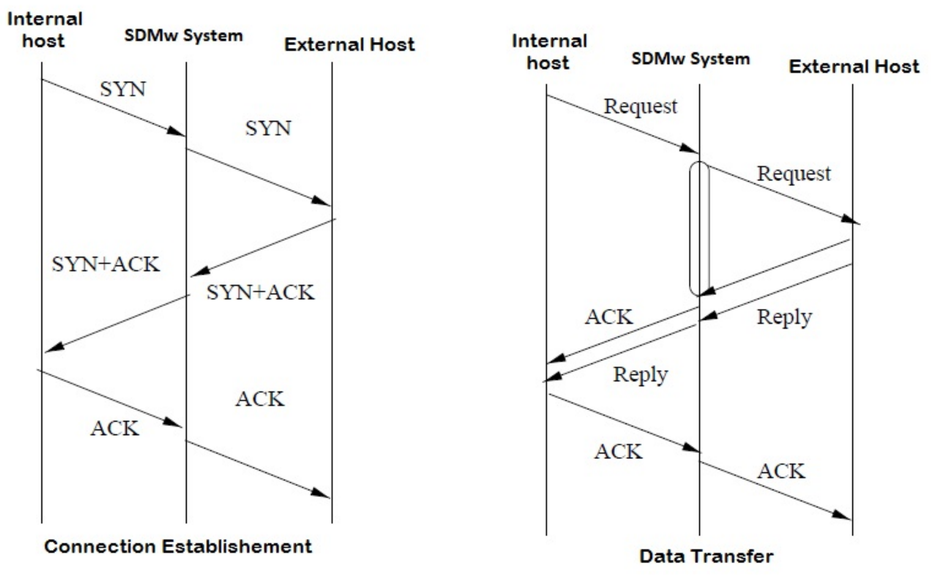


Figure 9.2: *Transmission Control Protocol connection establishment.*

Discussing system and hardware deadlock is beyond the scope of this thesis. In the area of distributed algorithms, a deadlock may arise between two algorithms A and B, when Algorithm A is expecting an input from Algorithm B and Algorithm B is expecting an input from Algorithm A in order for them to proceed to the next state.

However, from the *state-transition* diagram of the *TCP/IP* protocol [76], the deadlock due to the *TCP/IP* algorithm is not possible. The protocol specification has a clearly defined set of timing and procedures to either drop a packet/frame or to abort a connection in the event of any mismatch in the expected parameters or a long wait.

In a proxy-middleware system, deadlock occurs, whenever a *reply* from a destination host cannot be forwarded to the correct host; i.e., the destination address in a received unicast packet matches those of several inside hosts (due to multiple matches in a translation table). This may be due to a poorly implemented software system or not following the *IETF* specifications.

It is easy to establish the following result.

Theorem 9 *The proposed SDMw architecture works without any deadlock if and only if the underlying UHGA produces unique outputs without deadlock.*

Our proposed *SDMw* architecture only changes certain fields in the *TCP/IP* header (according to some established *OS* header generation scheme). We keep the source and the destination address intact. Since the underlying *UHGA* uses known *OS* header generation algorithms, it will not result in potential deadlock.

For connectionless *ICMP* and *IP* packets, the proposed *SDMw* system keeps the *source* and the *destination IP* address intact. *SDMw* only changes the *source IP-ID* and *flags* based on the current *OS* fingerprint in use. The potential reply from a destination host solely depends on the *Source IP* address, and thus cannot cause any possible deadlock. Even if two pairs of *¡source IP address, destination IP address¿* use the same set of parameters, it is not going to cause any potential deadlock.

For a *connection-oriented TCP* service, *SDMw* keeps the socket information unchanged. The proposed *SDMw* changes the *TCP* frame information like *ISN*, *MSS*, *timestamp* and other *OS* specific fields. Changing these fields are not going to cause any potential deadlock. However, to maintain an orderly *TCP* communication, *SDMw* maintains a *translational* table that keeps track of the changes made to sensitive *TCP* parameters like *ISN*, on-going *sequence* and *ACK* numbers. Since the socket information is kept intact, the *TCP* algorithm guarantees a deadlock-free operation.

9.4.2 The Performance of the *SDMw* System

In this section, we evaluate the performance of the proposed *SDMw* system in terms of the latency. We implemented a simple test-bench topology that consists of two hosts running *Windows OS*. They are connected to a switch, which is then connected to the *SDMw* system. The striped-down prototype of the proposed *SDMw* is implemented in a *Linux* box with two network cards. One network card (say *CARD1*) is connected to the switch and the other network card (*CARD2*) is connected with a server machine running the *HTTP* service. The *SDMw* is implemented as a *proxy-intercept* service between these two network cards. The *SDMw* uses the native *Linux TCP/IP kernel* behavior. The prototype architecture is presented in Figure 9.3. We tested a connectionless *ICMP* and a connection-oriented *TCP* service from *Host 1* and *Host 2* to the server. Whenever an *IP* packet is received at *CARD1*, we record the current system time.

1. If the *protocol number* in the *IP* packet is 1 (*ICMP*), we keep the source and the destination *IP* address unchanged. We change *IP-ID* field and *flags* to reflect the underlying *TCP/IP* kernel behavior (as *SDMw* is using the native *Linux* kernel behavior). We then record the time in which the packet was forwarded to *CARD2*. The difference between the two timestamps gives us the latency due to *SDMw* processing, which includes *OS* scheduling latency. To minimize the latency due to running system services, we ran the system with only essential drivers and services. When a reply comes from the server, we simply forward it to the destination host without any modification.

2. If the *protocol number* in the *IP* packet is 6 (*TCP*), as before, we record the time

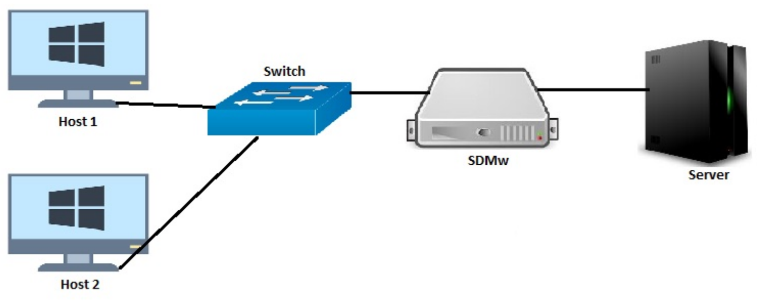


Figure 9.3: Secure Dynamic Middleware (*SDMw*) test bench topology.

at which the packet was received. We then inspect the *TCP* flag to determine its type.

- If the received segment is a *SYN* (say with *initial sequence number* ISN_o) frame, then the source host is trying to establish a new *TCP* connection with the server. In this case, the *SDMw* system creates a *new ISN* (ISN_n), based on the underlying *Linux* kernel and records the mapping between ¡ISN_o, ISN_n¿ in the translational table. Except for the socket information, *SDMw* modifies other *TCP* window parameters to reflect the underlying *Linux* behavior. The modified *TCP* segment is wrapped on to an *IP* packet, time-stamped and sent to the Server through *CARD2*.
- Whenever a reply (*SYN-ACK* for ISN_n) arrives from the server, based on the translational table, *SDMw* must translate the *SYN-ACK* for ISN_o.
- If a *DATA* segment with an *implicit ACK* is received from an inside host, then it must be a part of an on-going connection. The *Sequence (SEQ) number* for the *DATA* component must be changed by the *SDMw* using the translational table. This is to reflect the change in the *ISN*. *SDMw* adds (ISN_n-ISN_o) to the *original SEQ* number given by the host's *TCP* segment to reflect the new *SEQ* number.

  Since the *SEQ* number from the down-stream traffic (server to host) is unchanged by *SDMw*, the implicit *ACK* for the segment from the server is unchanged.
- If a *DATA* segment from the server is received addressed to an inside host, the *SEQ* numbers pertaining to the *DATA* are unchanged. However, any implicit *ACK* (for the up-stream traffic) from the server is changed to reflect

the implicit *ACK* for the original *TCP-DATA* segment from the host using the translational table.

- If the received *TCP*-segment is a *FIN* segment from a host, similar to the *DATA* segment, the *SEQ* number for the *FIN* segment is changed to reflect the change in the *ISN*.

  Any *FIN* segment from the down-stream traffic is unchanged.

SDMw introduces the following latency for connection-oriented and connectionless *IP* services:

1. The *protocol* field of the incoming *IP* packet needs to be examined in order to perform the correct action.

2. For a connectionless *IP* or *UDP* services, only the *IP-ID* and *flag* fields needs to be changed. There is no need to maintain any reference table information.

3. For a connection-oriented *TCP* service, the latency depends on whether the incoming segment is a *TCP-SYN* or a *DATA/ACK/FIN* segment. For a new connection, *SDMw* must create a new *ISN* depending on the *OS* fingerprint in use, and create a mapping between the incoming *ISN* and the new *ISN* in the translational table. *DATA/ACK/FIN* segments are a part of an ongoing *TCP* connection. In this case, based on the socket information, *SDMw* searches the table finds the ¡ISN_o, ISN_n¿ mapping. Based on this mapping, the *Sequence number* field in the *TCP* segment must be updated. For *DATA/ACK/FIN* segments, *SDMw* introduces *table-lookup* latency along with *Sequence number* updating latency.

We collected statistics for 200 *ICMP* packets (100 from each host) and 200 *TCP-DATA/ACK/FIN* segments. We also collected the latency associated with *TCP-SYN* segments associated with the data connection. The performance graph is presented in Figure 9.4. As it can be seen from the performance graph, the *SDMw* system did not introduce considerable latency due to the header rewriting process. For *TCP-DATA/ACK/FIN* segments, the average latency is about 2.2 ms. For the associated *TCP-SYN* segments, the latency is 2.3 ms. For connectionless *ICMP* packets, the latency is 1.1 ms. The latency can be improved if we run the *SDMw* algorithm in an application-specific integrated circuit (ASIC) processor with highspeed *content addressable memory.*

9.5 Chapter Summary

In this chapter, we proposed a novel idea to misguide an attacker during the fingerprinting process. We call our proposed system *SDMw–Secure Dynamic Middleware.* Our proposed

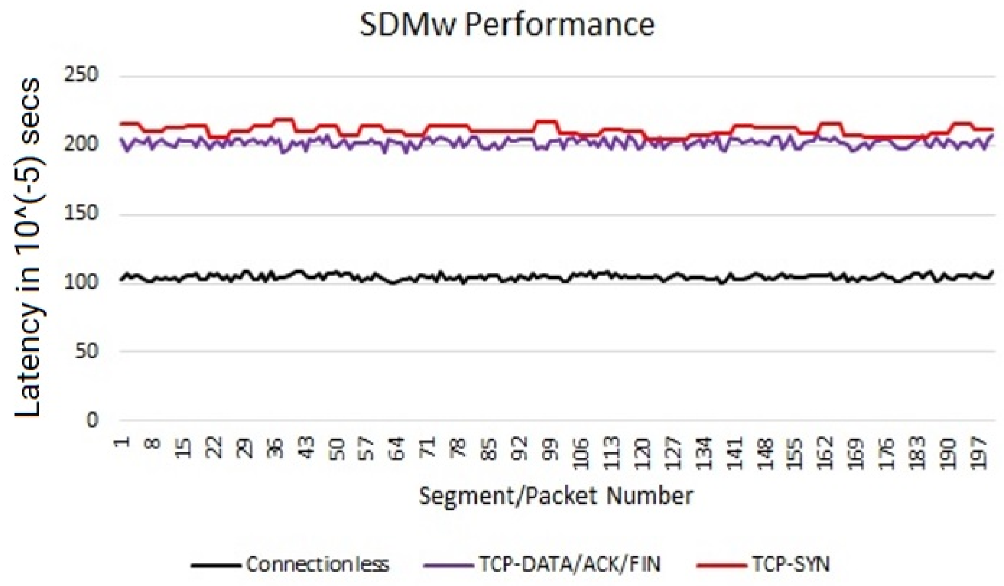


Figure 9.4: *SDMw performance.*

system has several advantages compared with similar models or systems that attempt to prevent fingerprinting by hackers:

1. Systems that try to defeat fingerprinting or hide the network particularly from *NMAP* expose themselves to fingerprinting. However, our proposed system misguides the attacker by showing different *OS* characteristics at different times.

2. The *SDMw* system does not introduce severe latency. Since the *SDMw* system is rewriting the header, *NAT* can be incorporated with *SDMw*. In addition, *SOCKS5* can be integrated with *SDMw* as well.

3. The *SDMw* system may be used effectively as a *Firewall*. Since the *SDMw* system acts like a *proxy*, it can inspect incoming packets and decide whether to allow the packet into the external network or not. Unlike traditional firewalls, our proposed *SDMw* system can implement firewall rules based on either a single machine or a port or groups of hosts.

Chapter 10

Consolidation of the research outcome

In this chapter, we provide a consolidation of our research outcomes. This chapter also provides the strength of our solutions and their limitations.

In this thesis, we addressed three research questions mentioned in Chapter 4. We addressed three fundamental problems in this thesis: *stopping insider, outsider and collaborative attacks* launched against a corporate network.

The ARP poisoning is one of the critical attacks launched by an insider. This attack is launched by exploiting the broadcast nature of the IPv4 protocol. In Chapter 6, we eliminated broadcasting in an IPv4 network by converting broadcasting into secure multicasting. The proposed *PrECast* infrastructure can run as a *software service* in corporate LAN switches. The only requirement is that all the corporate switches must support Layer-3. As of now, almost all manageable switches support Layer 3. Thus, our assumption is not unrealistic. Besides this, *PrECast* infrastructure does not require changing the protocol stack at every host. Unlike other solutions available in the literature, our proposal does not assume any *a priori* relationship between corporate hosts and the network, thus, supporting BYOD. In addition to this feature, *PrECast* infrastructure does not use any cryptographic schemes; thus, it is simple to configure and maintain.

Since *PrECast* infrastructure is unique and no similar proposal is available in the literature, we could not compare and benchmark against other proposals. However, we evaluated the performance of the *PrECast* infrastructure against the conventional ARP protocol to measure the additional latency introduced by our proposal. We also analyzed the message complexity of the *PrECast* protocol using mathematical modelling—our simulation results and mathematical modelling favour our proposal.

The *PrECast* infrastructure assumes the existence of a multicast tree connecting all corporate LAN switches and critical servers (such as DHCP and DNS). In Chapter 5, we created a multicast tree based on the *Core-based tree* (CBT) paradigm. Authentication of LAN devices is one of the biggest challenges. Robust admission control and key revocation mechanism must be in place in a corporate network. Traditional public key cryptography, such as RSA, cannot address the Key revocation process effectively. Also, RSA requires the existence of a PKI in the network. Attackers always target PKI servers to paralyze cryptographic operations in a network. We proposed a new cryptographic scheme called *pseudo-identity-based encryption* to address these issues. We provided a mathematical algorithm on how key revocation and key management are handled without the presence of a PKI.

The multicast tree used in Chapter 5 can also be used in distributing OS patches, installing updates and deploying new configurations to all devices securely.

Together with the CBT, the *PrECast* infrastructure stops insider attacks that are due to ARP poisoning.

We now discuss the collaborative attack. Attackers commonly exploit IP spoofing in launching DoS/DDoS attacks against any network. A corporate network may use NAT to hide its hosts from an external network. In this case, malicious inside hosts might spoof another legitimate host to open the NAT ports for an external attacker to enter the network. This fall under collaborative attack. To thwart both issues, we designed a modified form of NAT called the NAT++ infrastructure. NAT++ is essentially a micro-NATting architecture where no corporate host knows the address of another host. Since NAT++ is essentially a NAT, it hides the internal hosts through private IP addresses. Thus, NAT++ defeats a port-scanning attack, both within and outside the network.

Since NAT++ is a modification of a NAT, we compared the performance of NAT++ against the NAT scheme. The performance results show that NAT++ incurs only additional acceptable latency. Implementing NAT++ does not add any additional processing load on the LAN switches.

Finally, we discuss outsider attacks. DoS/DDoS is the common attack vector employed by outsiders in targeting a corporate network. Port-scanning is another attack frequently employed by the attacker community in penetrating any network. An outside attacker may install malicious codes so that he can control the host to launch collaborative attacks.

Since NAT++ hides the internal network from the external network through the use of private IP addresses, it solves IP spoofing and port-scanning attacks.

In Chapter 9, we provided a middleware-based solution called SDMW to address the problem of port scanning. Rather than defeating the port-scanning attack, the proposed SDMW middleware will mislead the attacker by providing them with wrong information about the network and hosts. Thus, potential attacks may be avoided.

Origin	CBT	PrECast	NAT++	IPRPE	SDMW
Insider	X	X	X	X	NA
Outsider	X	NA	X	X	X
Collaborative	X	NA	X	X	NA

Table 10.1: Contribution from the thesis

In a corporate network, an inside host may communicate with any outside host at its will. However, an outsider may be malicious and thus could spread malicious codes. Therefore, the reputation of the outside host is vital in deciding whether to allow or deny a connection. In Chapter 8, we provided a solution based on the dynamic firewall to evaluate the reputation of any outsider before allowing a connection. Since this action is performed on all incoming and outgoing connections, the dynamic firewall-based proposal will thwart any malicious outsider from installing codes on an inside host. The connection is aborted even if an inside host clicks a malicious link accidentally or on purpose. Our proposal uses commercial IP reputation-based systems such as Talos. We evaluated the performance of this proposal in terms of the additional latency introduced. The system introduces only acceptable latency that is healthy for a TCP connection. Since IP reputations are cached and reused, no additional latency is introduced for subsequent connections. A suitable timeout value is selected to provide changes to the reputation.

In this thesis, we have provided the following frameworks:

1. CBT-based multicast tree creation and maintenance algorithm.
2. The *PrECast* infrastructure
3. The NAT++ infrastructure
4. IP Reputation-based dynamic filtering
5. Dynamic middleware for defeating OS and network fingerprinting

It is interesting to note that the above frameworks can work independently of each other, or work in conjunction with others to protect the entire network from all three classes of attacks mentioned in this thesis. This is one of the salient features of this thesis.

In table 10.1, we summarize different frameworks provided in this thesis and which classes of attacks they provide the solution.

Chapter 11

Conclusion and Future Directions

In this chapter, we provide concluding remarks and future direction for the problems discussed in this thesis.

11.1 Concluding Remarks

In this thesis, we developed the following frameworks:

1. Creation of a secure and scalable multicast delivery tree based on the CBT paradigm. We also modified the identity-based encryption mechanism to suit our environment.
2. The *PrECast* infrastructure for the elimination of broadcasting in IPv4 network.
3. The *NAT++* infrastructure for creating micro segmenting in a network to defeat IP spoofing attacks.
4. An enforceable dynamic IP filtering mechanism using IP reputation of the destination host.
5. The *SDMw* middleware to defeat port scanning.

We now discuss concluding remarks in detail.

Broadcasting is one of the essential features in the IPv4 network that cannot be eliminated. The attacking community exploits this feature and launch various attacks against the IPv4 network. Some of the episodes are launched for personal benefits, whereas the others are launched to disrupt the normal operations; thus, resulting in financial loss, loss of reputation and drop in share prices.

The IPv4 (and the IPv6) protocol do not validate the IP packet header's source address field. The source IP address is only used for the destination host to send a *reply* message—this weakness results in IP-Spoofing attacks.

The attackers can be from anywhere on the Internet. We classify them as *insiders* and *outsiders* based on their location relative to a network. Some attacks require collaboration between insiders and outsiders. We call it *collaborative* attacks.

Combining the broadcasting and non-validating features, an attacker can launch several attacks against a network and end-hosts. In Chapter 3, we presented a taxonomy of various attacks that are based on IP broadcasting and IP spoofing. We colour-coded them to denote whether the attackers can derive personal benefit by launching these attacks. We also classify attacks that are due to insider, outsider and collaborative in nature.

The Man-in-the-middle (MITM) attack is one of the most destructive among other attacks. Here, the attacker ensures that all the communication between a source and the destination host passes through the attacker. This attack was launched for personal benefit and is due to the broadcast nature of the IPv4 protocol. There are numerous attacks that fall into this category. Several solutions are available in the literature that targets a specific subclasses of the MITM attack. Even then, they have severe limitations and cannot be implemented in a corporate network that accepts the BYOD policy.

By eliminating IPv4 broadcasting, we can potentially stop several classes of MITM attacks that are launched by exploiting this feature. Elimination of IPv4 broadcasting is one of the main themes of this thesis. In Chapter 6, we proposed the *PrECast* framework for achieving this theme. The *PrECast* infrastructure provides a seamless mechanism for eliminating the MITM attack in a corporate LAN network.

The current *PrECast* proposal requires the existence of a multicast delivery tree in a corporate LAN that involves all switches. Chapter 5 proposed an algorithm to construct a multicast tree based on the *Core-based tree* paradigm. We also proposed a variant of RSA called the *pseudo-identity* based encryption for efficient key management. The proposed pseudo-identity-based encryption addresses several shortfalls in the existing public-key cryptography. The designed multicast tree, along with the proposed encryption algorithm, may also be used to distribute network-wide OS updates and security patches to all the network devices and the end-hosts.

The proposed *PrECast* infrastructure is modular and more features can be added without affecting the core functionalities. *PrECast* in its present form can only solve the MITM attacks in a corporate network. However, several LAN-based attacks are still possible. They are:

- IP-Spoofing: An insider host may spoof another corporate host to launch a collaborative attack. An insider may also spoof an external IP address to launch various attacks. This will reduce the reputation of the insider's network.

- If a LAN host is assigned a public IP address, it may be exposed to an outsider. In this case, the inside hosts are susceptible to port-scanning and unsolicited connection establishment attacks.

To protect the network from the above two attacks, we provided a micro-NATting architecture called *NAT++* in Chapter 7. *NAT++* creates a micro segmentation of the entire network. Every LAN host forms a peer-to-peer network with its connected switch port. By design, we provided every LAN host to have the same private IP address 192.168.0.2 with a subnet mask of 255.255.255.252. The address of the default gateway is 192.168.0.1, which is the switch port where this host is connected. The switch port automatically translates the private IP address to a public IP address based on the standard one-to-one NAT protocol. This process defeats IP address spoofing. Since all LAN hosts are assigned a private IP address, they are not exposed to the external network. Thus, it is not possible to perform a port scan from an external network, or perform unsolicited connection establishment from an external network.

In the current IP infrastructure, a source host may have a will to connect to any destination host on the Internet. The destination host may be an attacker or a malware spreader. *PrECast*, along with *NAT++*, cannot stop a host from establishing a connection with a malware spreader willfully. To address this problem, in Chapter 8, we proposed a dynamic filtering mechanism based on the reputation of the destination IP address to either *allow* or *deny* a connection. This filtering mechanism is a network-wide dynamically enforceable rule. We call this mechanism as *DIF*. The *DIF* can work in either centralized or distributed mode. We presented the advantages and the disadvantages of these modes. Being modular, the *DIF* infrastructure can work independently. If an organization cannot implement the *PrECast* and the *NAT++* infrastructure, they can still implement the *DIF* infrastructure to protect their network from various attacks and malware infections.

We provided *PrECast*, *NAT++* and the *DIF* infrastructure to protect a network from various attacks. There will be few public-facing devices and hosts in any network—for example, gateway routers, web and email servers and so on. Attackers from the external network may fingerprint these devices to launch several attacks, including the zero-day attack. It is not possible to stop someone fingerprinting a network or a host. Chapter 9 provided the *SDMW* framework to provide fictitious information during the fingerprinting process. By doing so, we may be able to thwart several future attacks due to fingerprinting. As with the *DIF* infrastructure, the *SDMW* framework can be implemented as a standalone module or with the *PrECast* and *NAT++* infrastructure.

11.2 Future Directions

In this section, we provide future direction related to the problems discussed in this research.

In Chapter 6, we provided the *PrECast* framework. We were unable to implement the *PrECast* framework in real network hardware due to non-availability of Cisco APIs. In Chapter 6, we analysed the theoretical performance of the *PrECast* architecture in terms of the message complexity and time to converge. We also simulated the *PrECast* protocol and compared its performance against the traditional ARP protocol. However, certain real-time performance parameters like CPU load on the switches and ARP-table lookup latency due to the *PrECast* protocol cannot be evaluated using simulation or theoretical results. Our future work involves implementing the *PrECast* protocol in real hardware and studying the performance parameters such as message complexity, time to converge, CPU load and ARP table lookup latency.

We now summarize the implementation tasks and the main guidelines of the experimental test.

As the first task, we modify the kernel of a switch operating system, so that the switching process opens a Layer-2 frame and records the source IP address in order to create the *MCAM* table. The *MCAM* table is kept in the RAM of the switches. We then start an inter-switch communication process to create a multicast group. All *PrECast*-enabled switches join this multicast group. The pre-configured keys are installed by the network administrator in these switches for secure inter-switch communication. As per the state-transition diagram presented in Figures 6.2 and 6.3, *PrECast* switches handle ARP and DHCP queries. Then, we collect short-term and long-term statistics and compared with the results presented in chapter 6. We strongly believe that the results obtained from the experimental network, when available, will complete and corroborate the results presented in this thesis.

The *PrECast* proposal may not offer a solution to the *ARP-poisoning* attack in a domain where the switches do not support the *PrECast* infrastructure. *PrECast* can isolate the *ARP-poisoning* problem to an unsupported switch domain in a mixed environment. Our future work involves running *PrECast* as a *proxy-like* service to solve the *ARP-poisoning* attack in a mixed environment. It is still possible to launch an attack if any malicious code modifies the *MCAM* table kept at a switch. Protecting the *MCAM* table from any malicious code is also our future work.

In Chapter 7, we provided a micro natting architecture. Both *PrECast* and the *NAT++* offered a solution to protect a corporate network from attacks that are due to vulnerabilities in the IPv4 protocol. However, if the network implements a dual-stack architecture containing both IPv4 and IPv6 protocols, an attacker may still penetrate the network by exploiting some of the vulnerabilities found in the IPv6 protocol. Our future work involves solution to IPv6 vulnerability issues. We strongly believe that the *Neighbor Discovery Protocol* of IPv6 has similar vulnerabilities to the ARP. Our future work focus on creating a dual-stack approach in implementing *PrECast* and the *NAT++* through IPv4 and IPv6 services.

In Chapter 8, we provided a dynamic filtering based on IP reputation. Our proposal will work seamlessly for both the IPv4 and the IPv6 protocols. In the present work, the connection is aborted whenever there is a connection from an inside host towards a destination host with a low reputation score. Our future work involves allowing this type of connection towards a "honeypot" to observe the nature of attacks. By doing so, we may gather more information on various trojans, worms, malware and malicious connections, and their behaviour. By doing so, we may be able to identify the attacker's motivation, thus stopping any future attacks. We may also be able to create an extensive *Cyber incident response* policy based on the reason for these attacks.

Fingerprinting is possible irrespective of the version of the IP protocol in use. The *SDMw* middleware work for both IPv4 and the IPv6 protocols without any modification.

Bibliography

[1] *Arbornetworks. Global DDoS Attack Data for the First Half-2016.* (2016). Available online: `https://www.arbornetworks.com/arbor-networks-releases-global-ddosattack-data-for-1h-2016` (accessed on 15 Jan. 2022).

[2] Ballardie, A.J.; Francis, P.F.; Crowcroft, J. *Core Based Trees.* ACM SIGCOMM Comput. Commun. Rev. 1993, 23, 85–95.

[3] Boneh, D.; Franklin, M. *Identity-based encryption from the weil pairing.* In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; Springer: Berlin/Heidelberg, Germany, 2001; pp. 213–229.

[4] Bruschi, D.; Ornaghi, A.; Rostin, E. *S-ARP: A secure address resolution protocol.* In Proceedings of the IEEE 19th Annual IEEE conference on Computer Security Applications, Las Vegas, Nevada USA, 8–12 December 2003.

[5] Bremler-Barr, A.; Levy, H., *Spoofing prevention method.* In Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom), Miami, FL, USA, 13–17 March 2005; Volume 1, pp. 536–547.

[6] Baig, H.U.; Kamran, F. *Detection of Port and network scan using time independent feature set.* In Proceedings of the IEEE Conference on Intelligence and Security Informatics, New Brunswick, NJ, USA, 23–24 May 2007; pp. 180–184.

[7] Bondy, J.A.; Murty, U.S.R. *Graph Theory With Applications*; Springer: New York, NY, USA, 2011

[8] Bazzell, M., *Hiding from the Internet: Eliminating personal online Information*, CreateSpace Independent Publishing Platform, North Charleston, SC, USA (2015)

[9] Byeongho Kang, JISU YANG, Jaehyun So, and Czang Yeob Kim, *Detecting Trigger-based Behaviors in Botnet Malware*, Proceedings of the 2015 Conference on

research in adaptive and convergent systems, October 2015 Pages 274–279, Prague, Czech Republic.

[10] Bhaiji, Y. *Network Security Technologies and Solutions*; (CCIE Professional Development Series); Cisco Press: Indianapolis, IN, USA, 2016.

[11] *BGP Table Data.* Available online: `https://bgp.potaroo.net/index-bgp.html` (accessed on 15 Jan. 2022).

[12] Cocks, C. *An identity-based encryption scheme based on quadratic residues.* In Proceedings of the IMA International Conference on Cryptography and Coding, Cirencester, UK, 17–19 December 2001; Springer: Berlin/Heidelberg, Germany, 2001; Volume 2260, pp. 360–363.

[13] Cisco, *Global IP Traffic Forecast and Methodology*, 2006-2011, Available online: `http://www.hbtf.org/files/cisco_IPforecast.pdf` (accessed on 15 Jan. 2022).

[14] Chatterjee, S.; Sarkar, P. *Identity Based Encryption*; Springer: New York, NY, USA, 2011.

[15] Calyptix. *Top 7 Network Attack Types in 2016.* 2016. Available online: `https://www.calyptix.com/top-threats/top-7-network-attack-types-2016/` (accessed on 15 Jan. 2022).

[16] *Cyber Security Market Size, Share & Trends Analysis Report,* By Component, By Security Type, By Solution, By Service, By Deployment, By Organization, By Application, By Region, And Segment Forecasts, 2020–2027, Grandview Research (June 2020)

[17] *Cisco Talos*, Available online: `https://talosintelligence.com/` (accessed on 15 Jan. 2022).

[18] *Command and Control Server*, Available online: `https://www.trendmicro.com/vinfo/us/security/definition/command-and-control-server` (accessed on 15 Jan. 2022).

[19] *Cybersecurity Global loss*, Available online: `https://www.washingtonpost.com/politics/2020/12/07/cybersecurity-202-global-losses-cybercrime-skyrocketed-nearly-1-trillion-2020/` (accessed on 15 Jan. 2022).

[20] *Cyren IP Reputation Check*, Available online: `https://www.cyren.com/security-center/cyren-ip-reputation-check` (accessed on 15 Jan. 2022).

[21] Dokas, P.; Ertoz, L.; Kumar, V.; Lazarevic, A.; Srivastava, J.; Tan, P., *Data mining for network intrusion detection.* In Proceedings of the NSF Workshop on Next Generation Data Mining, Baltimore, MD, USA, 1–3 November 2002; pp. 21–30.

[22] Donahue, G.A. *Network Warrior*, 2nd ed.; O'Reilly Press: New York, NY, USA, 2011.

[23] Davidoff, S.; Ham, J. *Network Forensics: Tracking Hackers through Cyberspace*; Prentice Hall Publication: Upper Saddle River, NJ, USA, 2012.

[24] Doyle, J.; Carroll, J.D. *Routing TCP/IP*; Cisco Press: Indianapolis, IN, USA, 2012; Volume 1.

[25] Deshpande, T.; Katsaros, P.; Smolka, S.A.; Stoller, S.D., *Stochastic Game-Based Analysis of the DNS Bandwidth Amplification Attack Using Probabilistic Model Checking.* In Proceedings of the Tenth IEEE European Dependable Computing Conference, Newcastle, UK, 13–16 May 2014.

[26] *2020 Data Breach Investigations Report*, Available online: `https://enterprise.verizon.com/resources/reports/dbir/` (accessed on 15 Jan. 2022).

[27] Ding Wei, Hua Zidong, Li Panhui, Gong Qiushi and Cheng Yuxi, *Botnet Host Detection Based on Heartbeat Association*, Proceedings of the 2020 4th International Conference on Cryptography, Security and Privacy, January 2020 Pages 42–46, Nanjing, China.

[28] Ehrenkranz, T.; Li, J. *On the state of IP spoofing defense.* ACM Trans. Internet Technol. 2009, 9, 6.

[29] H. Esquivel, A. Akella and T. Mori, *On the effectiveness of IP reputation for spam filtering*, 2010 Second International Conference on COMmunication Systems and NETworks (COMSNETS 2010), Bangalore, 2010, pp. 1-10, doi: 10.1109/COMSNETS.2010.5431981.

[30] Fachkha, C.; Bou-Harb, E.; Debbabi, M. *Fingerprinting Internet DNS Amplification DDoS Activities.* In Proceedings of the 6th IEEE International Conference on New Technologies, Mobility and Security (NTMS), Dubai, UAE, 30 March–2 April 2014.

[31] Fonseca, O.; Cunha, Í.; Fazzion, E.; Meira, W.; Junior, B.; Ferreira, R.A.; Katz-Bassett, E. *Tracking Down Sources of Spoofed IP Packets*, CoNEXT'19. In Proceedings of the 15th International Conference on Emerging Networking EXperiments and Technologies, Orlando, FL, USA, 9–12 December 2019; pp. 51–53.

[32] Gordon Foydor Lyon. *The Art of Port Scanning*; Phrack Magazine, 1997; Volume 7. Available online: `http://phrack.org/issues/51/11.html#article` (accessed on 15 Jan. 2022).

[33] Gordon Foydor Lyon. *Nmap, Network Scanning*; Pub. Insecure. Com: Sunnyvale, CA, USA, 2008.

[34] Graziani, R. *IPv6 Fundamentals: A Straightforward Approach to Understanding IPv6*; Cisco Press: Indianapolis, IN, USA, 2017.

[35] *Google IPv6 Statistics*. Available online: `https://www.google.com/intl/en/ipv6/statistics.html` (accessed on 22 Sep. 2022).

[36] Gordon Foydor Lyon. Nmap: *The Network Mapper-Free Security Scanner*. Available online: `https://nmap.org` (accessed on 15 Jan. 2022).

[37] Hanna, D.; Veeraraghavan, P.; Soh, B. *SDMw: Secure Dynamic Middleware for Defeating Port and OS Scanning*. Future Internet 2017, 9, 67.

[38] Hanna, D.; Veeraraghavan, P.; Pardede, E. *PrECast: An Efficient Crypto-Free Solution for Broadcast-Based Attacks in IPv4 Networks*. Electronics 2018, 7, 65.

[39] *Here's how much money a business should expect to lose if they're hit with a DDoS attack*. Available online: `https://www.techrepublic.com/article/heres-how-much-money-a-business-should-expect-to-lose-if-theyre-hit-with-a-ddos` (accessed on 15 Jan. 2022).

[40] *How the Internet Has Changed Everyday Life*, Available online: `https://www.bbvaopenmind.com/en/articles/internet-changed-everyday-life/` (accessed on 15 Jan. 2022).

[41] *IP Quality Score*, Available online: `https://www.ipqualityscore.com/` (accessed on 15 Jan. 2022).

[42] *IP Reputation Investigation*, Available online: `https://ipremoval.sms.symantec.com/` (accessed on 15 Jan. 2022).

[43] *The Internet*, Available online: https://en.wikipedia.org/wiki/Internet (accessed on 15 Jan. 2022).

[44] *The Internet*, Available online: https://ourworldindata.org/internet (accessed on 15 Jan. 2022).

[45] *Information Security Policy*, Available online: https://www.itgovernance.co.uk/blog/5-information-security-policies-your-organisation-must-have (accessed on 15 Jan. 2022).

[46] Info Sec Institute. *How to Stop DNS Hijacking.* 2014. Available online: http://resources.infosecinstitute.com/stop-dns-hijacking/ (accessed on 15 Jan. 2022).

[47] *IP Spoofing: Simple Manipulation of Data Packets by Attackers.* Available online: https://www.ionos.com/digitalguide/server/security/ip-spoofing-fundamentals-and-counter-measures/ (accessed on 15 Jan. 2022).

[48] Kohno, T.; Broido, A.; Claffy, K., *Remote physical device fingerprinting.* IEEE Trans. Dependable Secur. Comput. 2005, 2, 93–108.

[49] Kammoun, N.; Bounfour, A.; Özaygen, A.; Dieye, R., *Financial market reaction to cyberattacks.* Cogent Econ. Financ., Jan 2019, 7, Issue 1.

[50] Kasperskey Lab. *What is a Smurf Attack?* Available online: https://usa.kaspersky.com/resource-center/definitions/smurf-attack (accessed on 15 Jan. 2022).

[51] Kerberos: *The Network Authentication Protocol.* Available online: https://web.mit.edu/kerberos/ (accessed on 15 Jan. 2022).

[52] Kevin Mitnick, Available online: https://en.wikipedia.org/wiki/Kevin_Mitnick (accessed on 15 Jan. 2022).

[53] Leckie, C.; Kotagiri, R., *A probabilistic approach to detecting network scans.* In Proceedings of the Eighth IEEE Network Operations and Management symposium (NOMS), Florence, Italy, 15–19 April 2002; pp. 359–372.

[54] Lootah, W.; Enck, W.; McDaniel, P. *TARP: Ticket-based address resolution protocol.* Comput. Netw. 2007, 51, 4322–4337.

[55] Loveless, J.; Blair, R.; Durai, A. *IPMulticast Vol 1: Cisco IP Multicast Networking*; Cisco Press: Indianapolis, IN, USA, 2016.

[56] Lichtblau, F.; Streibelt, F.; Kruger, T.; Richter, P.; Feldmann, A. *Detection, classification, and analysis of inter-domain traffic with spoofed source IP addresses.* In Proceedings of the 2017 Internet Measurement Conference, London, UK, 1–3 November 2017; pp. 86–99.

[57] McDaniel, P.; Aiello, W.; Butler, K.; Loannidis, J. *Origin authentication in inter-domain routing.* Comput. Netw. 2006, 50, 2953–2980.

[58] Nam, S.Y.; Kim, D.; Kim, J. *Enhanced ARP: Preventing ARP poisoning based man-in-the-middle attacks.* IEEE Commun. Lett. 2010, 2, 187–189.

[59] Mirkovic, J.; Kline, E.; Reiher, P. *RESECT: Self-Learning Traffic Filters for IP Spoofing Defense.* In Proceedings of ACSAC 2017; ACM: New York, NY, USA, 2017.

[60] *Microsoft SMB Protocol and CIFS Protocol Overview.* Available online: `https://docs.microsoft.com/en-us/windows-server/storage/file-server/file-server-smb-overview` (accessed on 15 Jan. 2022).

[61] *Morris worm*, Available online: `https://en.wikipedia.org/wiki/Morris_worm` (accessed on 15 Jan. 2022).

[62] Nam, S.Y.; Djuraev, S.; Park, M. *Collaborative approach to mitigating ARP poisoning-based man-in-the-middle attacks.* Comput. Netw. 2013, 57, 3866–3884.

[63] Oppenheimer, P. *Top-Down Network Design*; Cisco Press: Indianapolis, IN, USA, 2010.

[64] Park, K.; Lee, H. *On the effectiveness of route-based packet filtering for distributed DoS attack prevention in power-law internets.* In Proceedings of ACM SIGCOMM Data Communications Festival, 15–26 2001.

[65] Pfleeger, C.P., Pfleeger, S.L., and Margilies, J., *Security in Computing*, Prentice-Hall publications, Upper Saddle River, NJ, USA, 2015.

[66] Prowell S., *Heartbeat – Cyber Anomaly Detection through Side-Channel Analysis of Periodic System Function Invocation*, Available online: `https://www.ornl.gov/sites/default/files/TIP-Heartbeat-2018.pdf` (accessed on 15 Jan. 2022).

[67] Rodriguez-Gomez, R.A.; Macia-Fernandez, G.; Garcia-Teodoro, P., *Survey and taxonomy of botnet research through life-cycle.* ACM Comput. Surv. 2013, (45), doi:10.1145/2501654.2501659.

[68] *Ransomware*, Available online: `https://www.cisa.gov/stopransomware` (accessed on 15 Jan. 2022).

[69] *RFC 791: The Internet Protocol*, Available online: `https://datatracker.ietf.org/doc/html/rfc791` (accessed on 15 Jan. 2022).

[70] *RFC 793: The Transmission Control Protocol*, Available online: `https://datatracker.ietf.org/doc/html/rfc793` (accessed on 15 Jan. 2022).

[71] *A Reverse Address Resolution Protocol (RARP)*, Available online: `https://datatracker.ietf.org/doc/html/rfc903` (accessed on 15 Jan. 2022).

[72] *RFC951: Bootstrap Protocol (BOOTP)*, Available online: `https://datatracker.ietf.org/doc/html/rfc951` (accessed on 15 Jan. 2022).

[73] *RFC 2663: IP Network Address Translator (NAT) Terminology and Considerations*, 1999. `https://tools.ietf.org/html/rfc2663` (accessed on 15 Jan. 2022).

[74] Shamir, A. *How to share a secret.* Commun. ACM 1979, 22, 612–613.

[75] Shamir, A. *Identity-based cryptosystems and signature schemes.* In Workshop on the Theory and Application of Cryptographic Techniques; Springer: Berlin/Heidelberg, Germany, 1984; pp. 47–53.

[76] Stevens, W.R, *TCP/IP Illustrated Volume 1*; Addison-Wesley Longman Publications, Boston, MA, USA, 1995.

[77] Sitton, D. *Maximum matchings in a complete multipartite graph.* Electron. J. Undergrad. Math. 1996, 2, 6–16.

[78] Seo, K.; Lynn, C.; Kent, S. *Public-key infrastructure for the secure border gateway protocol (S-BGP).* In Proceedings of the IEEE Conference on DARPA Information Survivability, Washington, DC, USA, 12–14 June 2001.

[79] Soniya, B.; Wisey, M., *Detection of TCP SYN scanning using Packet counts and neural network.* In Proceedings of the IEEE International Conference on Signal Image Technology and Internet Based Systems, SITIS, Bali, Indonesia, 30 November–3 December 2008; pp. 646–649.

[80] Sprgeon, C.E., and Zimmerman, J., *Ethernet: The Definitive Guide*, 2nd Edition, O'Reilly Media, Inc., Sebastopol, CA, USA, 2014.

[81] Sebastian Garcia and Michal Pechoucek, *Detecting the Behavioral Relationships of Malware Connections*, Proceedings of the 1st International Workshop on AI for Privacy and Security, August 2016 Article No.: 8, Pages 1–5, The Hague, Netherlands.

[82] Shamsi, Z.; Nandwani, A.; Leonard, D.; Loguinov, D. Hershel: *Single-Packet OS Fingerprinting.* IEEE/ACM Trans. Netw. 2016, 24, 2196–2209.

[83] Santos, Omar, *Cisco CyberOps Associate CBROPS 200-201 Official Cert Guide*, Cisco Press, Hoboken, NJ, 2020.

[84] Simonite, T., *To Keep Passwords Safe from Hackers, Just Break Them into Bits.* Available online: `https://www.technologyreview.com/s/429498/to-keep-passwords-safe-from-hackers-just-break-them-into-bits/` (accessed on 15 Jan. 2022).

[85] *Social Engineering Attacks*, Available online: `https://www.imperva.com/learn/application-security/social-engineering-attack/` (accessed on 15 Jan. 2022).

[86] Sans Infosec. *Layer 2 Network Protections against Man in the Middle Attacks.* Available online: `https://isc.sans.edu/forums/diary/layer+2+network+protections+against+man+in+the+middle+attacks/7567/` (accessed on 15 Jan. 2022).

[87] Steven Bass, *Spoofed IP Address Distributed Denial of Service Attacks: Defence-in-Depth* (2001) SANS Reading Room. Available online: `https://www.sans.org/reading-room/whitepapers/threats/paper/469` (accessed on 15 Jan. 2022).

[88] *The Samba Protocol.* Available online: `https://www.samba.org/` (accessed on 15 Jan. 2022).

[89] Tanenbaum, A.S.; Wetherall.D.J., *Computer Networks, (5th Ed.)*, Pearson International, Boston, MA, USA, 2010.

[90] US-CERT. *DNS Amplification Attacks.* Available online: `https://www.us-cert.gov/ncas/alerts/TA13-088A` (accessed on 15 Jan. 2022).

[91] Veeraraghavan, P. *Pseudo-identity based encryption and its application in mobile ad hoc networks.* In Proceedings of the 2011 IEEE 10th Malaysia International

Conference on Communications, Kota Kinabalu, Sabah, Malaysia, 2–5 October 2011.

[92] Veeraraghavan, P.; Hanna, D.; Pardede, E. *Building Scalable and Secure Multicast Delivery Infrastructure in a Local Area Network.* **Electronics** 2019, 8, 1162. https://doi.org/10.3390/electronics8101162

[93] Veeraraghavan, P.; Hanna, D.; Pardede, E. *NAT++: An Efficient Micro-NAT Architecture for Solving IP-Spoofing Attacks in a Corporate Network.* **Electronics** 2020, 9, 1510.
https://doi.org/10.3390/electronics9091510

[94] Vladimir Levin (Russian Hacker), Available online: `https://en.wikipedia.org/wiki/Vladimir_Levin` (accessed on 15 Jan. 2022).

[95] Wang, H.; Jin, C.; Shin, K.G., *Defense Against Spoofed IP Traffic Using Hop-Count Filtering.* IEEE/ACM Trans. Netw. 2007, 15, 40–53.

[96] Wikipedia. *Content-Addressable Memory.* Available online: `https://en.wikipedia.org/wiki/Content-addressable_memory` (accessed on 15 Jan. 2022).

[97] *What is Social Engineering*, Available online: `https://www.webroot.com/au/en/resources/tips-articles/what-is-social-engineering` (accessed on 15 Jan. 2022).

[98] *Webroot BrightCloud IP Reputation Service*, Available online: `https://www.webroot.com/au/en/business/threat-intelligence/internet/ip-reputation` (accessed on 15 Jan. 2022).

[99] Xue.L and Sun G., *Design and implementation of a malware detection system based on network behavior*, SECURITY AND COMMUNICATION NETWORKS, Vol 8, pp. 459-470, 2015.

[100] *Yersinia.* Available online: `http://www.yersinia.net/index.htm` (accessed on 15 Jan. 2022).

[101] Yang, H.; Lu, Z. *Unlocking the Power of OPNET Modeler*; Cambridge University Press: Cambridge, UK, 2012.