Advances in Topological Vulnerability Analysis

Steven Noel[1], Matthew Elder[2], Sushil Jajodia[1], Pramod Kalapa[1], Scott O'Hare[3], Kenneth Prole[3]
[1] *Center for Secure Information Systems, George Mason University, Fairfax, VA, USA*
[2] *Symantec Research Labs, Symantec Corporation, Herndon, VA, USA*
[3] *Secure Decisions, Division of Applied Visions Inc., Northport, NY, USA*
snoel@gmu.edu, matthew_elder@symantec.com, jajodia@gmu.edu, pkalapa@gmu.edu,
scotto@securedecisions.com, kennyp@securedecisions.com

Abstract

Currently, network administrators must rely on labor-intensive processes for tracking network configurations and vulnerabilities, which requires a great deal of expertise and is error prone. The organization of networks and the interdependencies of vulnerabilities are so complex as to make traditional vulnerability analysis inadequate. We describe a Topological Vulnerability Analysis (TVA) approach that analyzes vulnerability dependencies and shows all possible attack paths into a network. From models of the network vulnerabilities and potential attacker exploits, we discover attack paths (organized as graphs) that convey the impact of individual and combined vulnerabilities on overall security. We provide sophisticated attack graph visualizations, with high-level overviews and detail drilldown. Decision support capabilities let analysts make optimal tradeoffs between safety and availability, and show how to best apply limited security resources. We employ efficient algorithms that scale well to larger networks.

1. Introduction

While we cannot predict the origin and timing of attacks, we can reduce their impact by knowing the possible attack paths through our networks. Reliance on manual processes and mental models is inadequate. Automated tools are needed for analyzing and visualizing vulnerability dependencies and attack paths, for understanding overall security posture.

Our approach to such full-context security is called *Topological Vulnerability Analysis* (TVA) [1][2]. TVA models network state and potential attacker exploits, combining these to generate an attack graph showing all possible ways an attacker can penetrate the network. TVA transforms raw security data into a roadmap that lets one proactively prepare for attacks. It supports both offensive (e.g., penetration testing) and defensive (e.g., network hardening) applications. The mapping of attack paths through a network via TVA provides a concrete understanding of how individual and combined vulnerabilities impact overall network security.

One focus of this work is to populate TVA models through automated agent-based network discovery, asset management, and vulnerability reporting technology. This information is constantly updated, and enables accurate modeling for TVA. This avoids active network-based vulnerability scanning, which can be intrusive for operational networks. We also form high-level modeling abstractions that reduce model complexity, providing better situational awareness and improving scalability.

2. Overview of Approach

Figure 1 is an overview of our approach for building and analyzing attack graphs via TVA. *Network Capture* builds a model of the network, in terms of relevant security attributes. *Vulnerability Database* represents a comprehensive repository of reported vulnerabilities, with each vulnerability record listing the affected software (and hardware). The *Exploit Conditions* encode how each vulnerability may be exploited (preconditions) and the result of its exploitation (postconditions). *Network Capture* represents data collection for a network to be defended, in terms of corresponding elements in *Vulnerability Database* and *Exploit Conditions*. Together, all these inputs are used to build an *Environment Model* for multi-step attack graph simulation.

The *Graph Engine* uses the *Environment Model* to simulate multi-step attacks through the network, for a given user-defined *Attack Scenario*. This engine analyzes vulnerability dependencies, matching exploit preconditions and postconditions, thus generating all possible paths through the network (for a given attack scenario). The system then provides sophisticated capabilities for interactive *Visual Analysis* of attack graphs [3]. It also computes *Optimal Counter Measures*, e.g., minimum number of network changes to thwart the attack scenario [4].

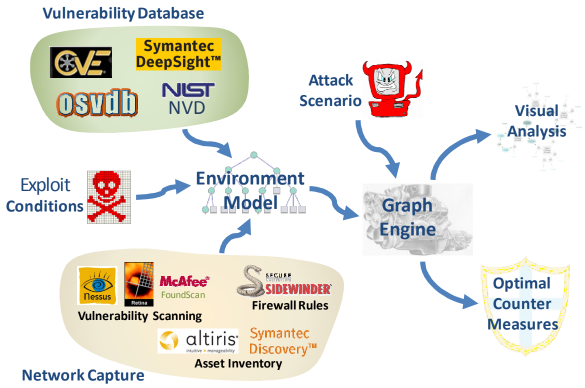

Figure 1. Topological Vulnerability Analysis (TVA)

TVA integrates with Nessus [5], Retina [6], and FoundScan [7] vulnerability scanners for populating its network model. It also processes data from the Sidewinder firewall [8] to capture network connectivity to vulnerable host services. Further, TVA integrates with Symantec Discovery (an Original Equipment Manufacturer version of Centennial Discovery [9]), providing more accurate and complete models. We have investigated Altiris Inventory Solution [10], which also incorporates asset inventory technology.

TVA then matches host configuration information gathered through asset inventory with a database of reported vulnerabilities. The result is an enumeration of vulnerabilities associated with each host. There are a number of vulnerability databases available, maintained by the government, commercial companies, and the security community. Examples include NIST's National Vulnerability Database (NVD) [11], the Bugtraq security database [12], the SecurityFocus forum [13], the Open Source Vulnerability Database (OSVDB) [14], and the Common Vulnerabilities and Exposure (CVE) referencing standard [15].

A key vulnerability data source for our TVA tool is Symantec DeepSight [16]. DeepSight is a commercial database that includes data from Bugtraq and SecurityFocus. TVA integrates with DeepSight's XML data feed, which is platform-independent and standards-based (web services). TVA then maps asset data from Symantec Discovery to the corresponding DeepSight/Bugtraq vulnerability descriptions. We also leverage DeepSight for modeling exploit conditions, e.g., local versus remote attacks, privileged versus non-privileged user access, etc. In our experiments, analysis of 25,000 DeepSight vulnerabilities (covering 340,000 different software packages) indicates that about 60% of these vulnerabilities represent some form of incremental network penetration, which we have modeled as TVA exploits.

Once the attack model (network and potential exploits) is defined, our TVA system generates an attack graph for a given user-defined attack scenario. The scenario may define particular starting and/or ending points for the attack, so that the graph is constrained to lie between them, or may be completely unconstrained (all possible starting and ending points). Attack graphs can also guide the placement of intrusion detection sensors [17], correlate intrusion alarms [18], handle missed alarms, and filter false alarms.

It has been suggested [19][20] that worst-case complexity for this kind of attack graph analysis is $O(n^4)$ or even $O(n^6)$, for n hosts in the network model. However, we have made improvements that reduce worst-case complexity to $O(n^2)$ [21]. Using a host-centric representation, we do not search blindly for dependency edges from among a flat set of exploits. We also avoid man-in-the-middle attacks (e.g., port forwarding or address spoofing), which involve triples rather than pairs of hosts, and have limited value for risk analysis. Further, we group hosts into protection domains (e.g., subnets) [22], with the implication of full reachability within a domain, reducing complexity to $O(n)$ within each domain. In terms of the database of potential attacker exploits, complexity is $O(e)$, for e exploits.

3. Illustrative Example

Consider a network separated from the Internet by a firewall. The network is divided into 3 subnets, with one host in each subnet: a DMZ web server, an internal client, and an internal server. The DMZ web server is running Microsoft Windows Server, with Internet Information Services (IIS), Apache/MySQL/PHP, and Tomcat servlets. The client is running Microsoft Windows XP, client security software, an office productivity suite, and other utilities. The internal server is running Apache/MySQL/PHP, the Symantec Discovery asset inventory server, and Altiris Inventory Solution with associated software (e.g., IIS, Microsoft SQL Server). The Altiris Agent is deployed on each internal machine to collect asset inventory data.

The firewall blocks direct access to the internal server and client subnets from the Internet. Thus, from the outside, Nessus will be unable to detect any vulnerabilities on the internal server and client. In fact, if the firewall uses network address translation (NAT), Nessus cannot even discover the existence of machines on these 2 subnets (i.e., they lack Internet-routable addresses). From the outside of the firewall, the only machine exposed is the DMZ server. In particular, the firewall blocks all traffic except HTTP traffic to the DMZ server's TCP port 80. From behind the firewall, Nessus shows a variety of vulnerabilities on the DMZ server. But from the Internet, only the web server vulnerability is exposed. A Nessus scan from the DMZ to the internal subnets identifies any internal vulnerabilities permitted through the firewall. In our network, MySQL traffic is permitted between the DMZ web server and the internal server, and two exposed MySQL vulnerabilities allow an attacker to access the internal server (from the DMZ web server).

The question is whether an attacker can compromise the internal server from the Internet. Figure 2 shows the resulting attack graph, using TVA with Nessus scans alone. This shows that an attacker starting on the Internet can first penetrate through the firewall and compromise the DMZ server, exploiting a vulnerability on its web server installation. Then, from the DMZ server, the attacker can access the internal server via exploitation of the two MySQL vulnerabilities.

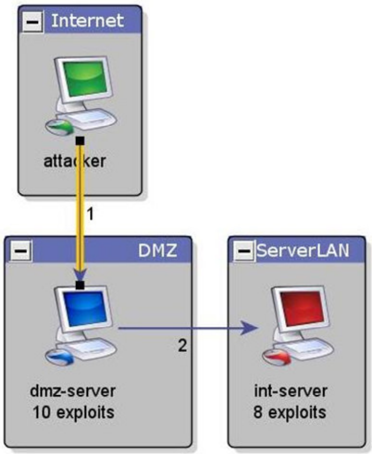

Figure 2. Attack graph for example network

A Nessus scan from the Internet would reveal no vulnerabilities on the internal server or client, so that there is no direct attack from the Internet to the internal machines. Because the firewall blocks traffic originating from the server subnet to the client subnet, and blocks all traffic from the DMZ to the client subnet, there is no attack path to the client at all. That is, a Nessus scan from the DMZ to the client reveals no vulnerabilities. In this case, a Nessus scan of the client within its own subnet (no intervening firewalls) detects no vulnerabilities (or even open ports), because of the client's personal firewall (part of the security suite).

However, the model based solely on Nessus data paints an incomplete picture of the attack paths through this network. In this case, while Nessus shows no vulnerabilities on the internal client, our approach of correlating asset inventory against a vulnerability database reveals otherwise. A detailed software inventory of the internal client machine from Altiris identifies 99 software executables.

We compare products and versions from Altiris against a vulnerability database to determine the vulnerabilities associated with each application. For many applications, a particular product and version maps to many vulnerabilities. For Microsoft products, this correlation process involves an extra step of first determining how many vulnerabilities are associated with the Microsoft product and version installed, then comparing the Microsoft patches and hotfixes installed (data that is also collected by the Altiris Inventory Solution product), thus determining which Microsoft vulnerabilities are unpatched. So, for example, while there are almost 200 vulnerabilities associated with Microsoft Windows XP SP2, many of the older vulnerabilities are patched on the client machine. On the other hand, no Microsoft hotfixes are applied on the client for Microsoft Office components after Office SP2. Thus all of the dozens of vulnerabilities associated with that version of Microsoft Office are relevant to the client.

These client-side vulnerabilities are associated with software that has no network service running. Still, these vulnerabilities represent vectors by which an attacker might obtain access to the client machine. Significant numbers of vulnerabilities are associated with web browsers and plug-ins. The typical scenario for these increasingly widespread vulnerabilities is that a user running a vulnerable web browser or plug-in visits a web site with malicious content that exploits the client-side vulnerability. Another important class of vulnerabilities is associated with document processing applications, e.g., infected documents via e-mail.

Figure 3 shows the attack graph resulting from our higher-fidelity model. This shows that it is actually possible to attack the client directly from the Internet, via 12 different client-side vulnerabilities. Strictly speaking, the client needs to make an outbound connection (e.g., web site visit) to a compromised server. But the firewall allows this, so it is correct to model the server as the attacker and the client as the victim. The attacker on the client can then compromise the internal server, through a firewall hole allowing access from client to server.

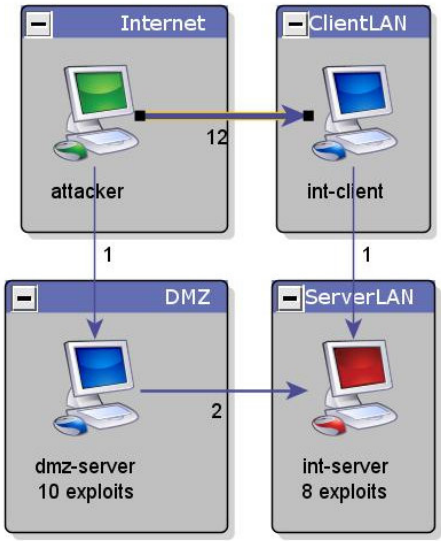

Figure 3. Attack graph for higher-fidelity model

This example illustrates the importance of accounting for such client-side vulnerabilities. Services on an internal server would typically be exposed to internal clients, and may be vulnerable. In this case, our higher-fidelity model has uncovered attack paths that would have otherwise been undetected through a model populated by a vulnerability scanner.

A major benefit of correlating asset inventory and vulnerability databases is uncovering vulnerabilities not seen from remote vulnerability scans. These "dark" vulnerabilities are an important input to attack graph analysis, for a more complete picture of the network security posture. An important class of vulnerabilities detected by this approach is client applications. Client-side vulnerabilities have a major impact on enterprise network security posture. For example, client-side web applications represent about 60% of vulnerabilities documented in 2007 [23].

4. Analysis and Visualization

To make TVA attack graphs feasible for realistic networks, we need scalable mathematical representations and algorithms. Modeling the attacker's control over the network as monotonic (increasing over time), we need only represent the dependencies among exploits (preconditions and postconditions), rather than explicitly enumerating every sequence of exploits. The resulting exploit-dependency attack graphs grow only quadratically (as opposed to exponentially) with the number of exploits, so that it becomes feasible to apply them for realistic networks. The assumption of monotonicity is quite reasonable, corresponding to the conservative assumption that once an attacker gains control of a network resource, he need not relinquish it to further advance the attack. That is, attack behavior is monotonic at a reasonable level of detail.

Based on a given attack scenario, the attack graph can be constrained by specific starting and ending points. The scenario could also be less constrained, such as finding all possible attack starts leading to one or more goals, or finding all possible paths from particular starting points. For example, one may wish to know how a particular critical system can be compromised from all possible starting points. Or, one may want to know all systems that could be compromised from a particular starting point, or even from all possible starting points. Our TVA implementation supports each combination of specified/unspecified attack start/goal.

In their raw non-aggregated form, attack graphs can quickly become too complex for easy understanding. To help manage attack graph complexity, we aggregate the graph to higher levels of abstraction, providing better situational awareness. An important high-level abstraction in TVA is the *protection domain*, which represents a set of machines that have full access to one another's vulnerabilities. In a raw (non-aggregated) form, the graph would be fully connected within a protection domain. Instead, we list the machines in a protection domain, along with exploits against each of their vulnerabilities. Then we implicitly rely on the fact that once an attacker takes control of a machine within a protection domain, he can exploit all vulnerabilities on machines within it. We thus need not explicitly list every n^2 (fully-connected) exploit dependency within the protection domain.

In TVA, a high-level view (Figure 4) displays attack relationships among protection domains, which can be opened individually or in groups for deeper views of attack properties and relationships. In this process, no graph information is lost; one has merely to expand a folder to acquire information at a lower level. A complete listing of exploits and associated details for any selected component is available at all times. This supports in-depth analysis of exploit details, while overall topology and network relationships are kept simple and understandable within the main graph view.

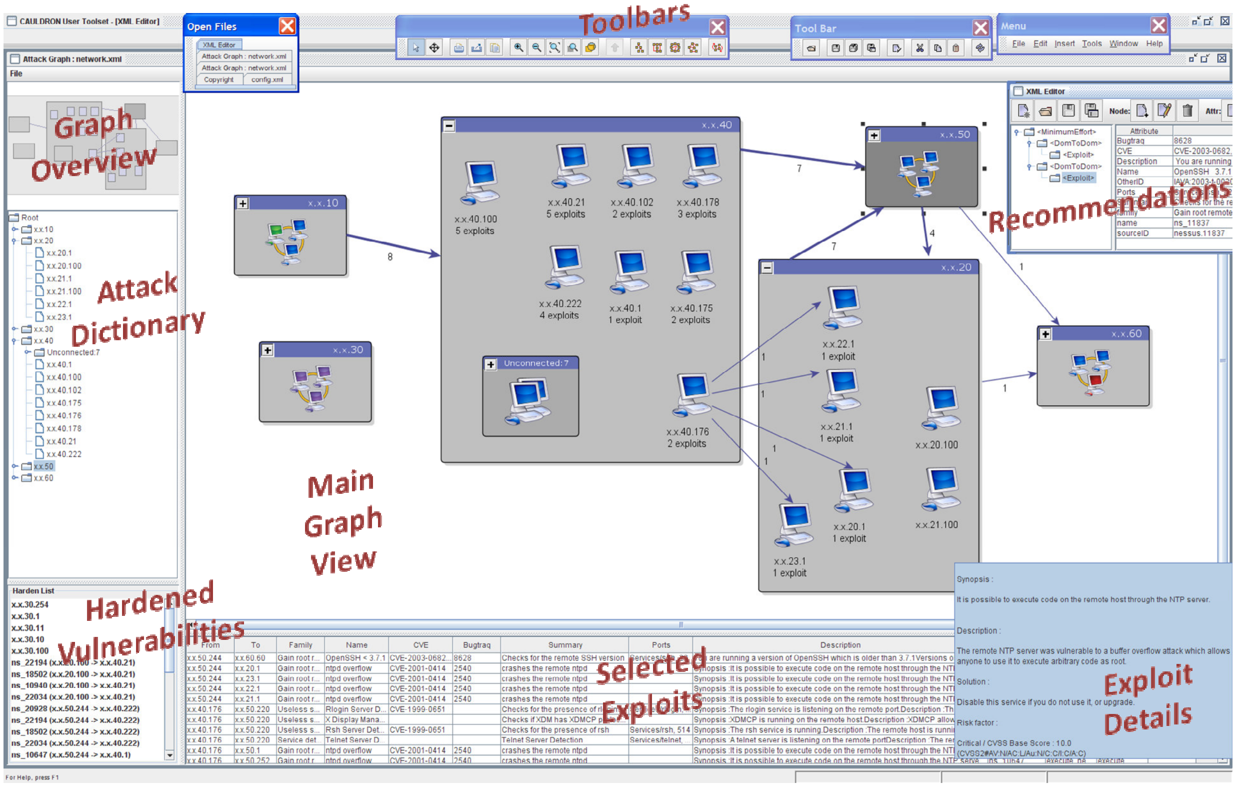

Figure 4. Attack graph visualization interface

Our TVA tool also emulates the hardening of machines and exploitable vulnerabilities to study the effects of remediation and what-if scenarios. Exploring the attack graph, the analyst is often faced with multiple options for remediation. This involves choosing a machine or set of machines to protect (harden), or identifying specific exploits to protect against. We display the attack graph effects that occur when a specific machine or protection domain is hardened or when a specific exploit is neutralized. Hardened elements are maintained in a log, e.g., for reporting. The TVA tool also generates recommendations automatically, i.e., first layer (from start), last layer (from goal), and minimum set that that separates start from goal.

To aid user navigation, the TVA tool maintains a global overview of the entire attack graph at all times, which can be used to pan the main graph view. The tool also has a graphical (tree) attack dictionary of all graph elements. The various graph views are linked, so that selecting an element in one view cause it to be selected in all views. A variety of toolbars are available for commonly used tools. This includes a suite of interactive layout tools, with manual repositioning as well as full-scale layout algorithms, continuously available to restructure the display.

5. Related Work

Early work in attack graph generation was based on explicit enumeration of attack states, which had serious scalability problems [24][25][26]. Under a practical assumption of monotonic logic, attack graph complexity was shown to be polynomial rather than exponential [19]. Graph complexity has been further reduced, to worst-case quadratic in the number of hosts [21][22]. Commercial capabilities for attack graph analysis remain limited, especially in the area of visualization for large-scale graphs [27][28]. A more detailed review of attack graph research (as of 2005) is given in [29].

Attack graph research has largely focused on scalability, with relatively little work on aspects of model population. Notable exceptions include [30][31][32], although these are more theoretical frameworks than practical model population. In most studies that apply attack graphs to real networks, models are populated via the Nessus vulnerability scanner. But network-based (remote) scanners such as Nessus have fundamental limitations, especially regarding access to host data and potential disruption to operational systems.

6. Summary and Conclusions

Attack graphs provide a powerful way of understanding the context and relative importance of vulnerabilities across systems and networks. Attack graph analysis depends on a complete and accurate model of the network. Typically such models have been built using data from network (remote) vulnerability scanners such as Nessus. However, remote scanning has fundamental limitations regarding the information available about target hosts. We propose a new way of building attack graph models, using data from asset inventory correlated with a vulnerability database. We demonstrate this approach using a small testbed network, and describe some validations we have conducted in operational environments. Our testbed experiments use Symantec Discovery asset inventory, correlated against the Symantec DeepSight vulnerability database. We compare the resulting attack graphs against those from a baseline network model using Nessus scans. Our approach reveals host vulnerabilities not detected by Nessus, including the important class of client-side vulnerabilities. The result is a more complete and accurate assessment of enterprise network security.

7. Acknowledgements

This material is partially based upon work supported by Homeland Security Advanced Research Projects Agency under the contract FA8750-05-C-0212 administered by the Air Force Research Laboratory (Rome); by Air Force Research Laboratory (Rome) under the contract FA8750-06-C-0246; and by Federal Aviation Administration under the contract DTFAWA-04-P-00278/0001.

8. References

[1] S. Jajodia, S. Noel, B. O'Berry, "Topological Analysis of Network Attack Vulnerability," in *Managing Cyber Threats: Issues, Approaches and Challenges*, V. Kumar, J. Srivastava, A. Lazarevic (eds.), Springer, 2005.

[2] S. Jajodia, S. Noel, "Topological Vulnerability Analysis: A Powerful New Approach for Network Attack Prevention, Detection, and Response," *Indian Statistical Institute Monograph Series*, World Scientific Press, 2008.

[3] S. Noel, M. Jacobs, P. Kalapa, S. Jajodia, "Multiple Coordinated Views for Network Attack Graphs," *ACM Workshop on Visualization for Computer Security*, 2005.

[4] L. Wang, S. Noel, S. Jajodia, "Minimum-Cost Network Hardening Using Attack Graphs," *Computer Communications*, 29, 2006.

[5] Nessus Vulnerability Scanner, http://www.nessus.org.

[6] Retina Security Scanner, http://www.eeye.com/.

[7] FoundScan, http://www.foundstone.com/.

[8] Sidewinder, http://www.securecomputing.com/.

[9] Discovery, http://www.centennial-software.com/.

[10] Altiris, http://www.altiris.com/.

[11] National Vulnerability Database, http://nvd.nist.gov/.

[12] Bugtraq, http://www.securityfocus.com/vulnerabilities.

[13] Security Focus, http://www.securityfocus.com/.

[14] Open Source Vulnerability Database, http://osvdb.org/ .

[15] Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org/.

[16] Symantec DeepSight Threat Management System, https://tms.symantec.com/Default.aspx.

[17] S. Noel, S. Jajodia, "Optimal IDS Sensor Placement and Alert Prioritization Using Attack Graphs," *Journal of Network and Systems Management*, September 2008.

[18] S. Noel, E. Robertson, S. Jajodia, "Correlating Intrusion Events and Building Attack Scenarios through Attack Graph Distances," *20th Annual Computer Security Applications Conference*, 2004.

[19] P. Ammann, D. Wijesekera, S. Kaushik, "Scalable, Graph-Based Network Vulnerability Analysis," *ACM Conference on Computer and Communications Security*, 2002.

[20] R. Lippmann, K. Ingols, C. Scott, K. Piwowarski, K. Kratkiewicz, M. Artz, R. Cunningham, "Validating and Restoring Defense in Depth Using Attack Graphs," *MILCOM Military Communications Conference*, 2006.

[21] S. Noel, J. Jajodia, "Understanding Complex Network Attack Graphs through Clustered Adjacency Matrices," *21st Annual Computer Security Applications Conference*, 2005.

[22] S. Noel, S. Jajodia, "Managing Attack Graph Complexity through Visual Hierarchical Aggregation," in *Visualization and Data Mining for Computer Security*, 2004.

[23] D. Turner, M. Fossi, E. Johnson, T. Mack, J. Blackbird, S. Entwisle, M. K. Low, D. McKinney, C. Wueest, *Symantec Global Internet Security Threat Report Trends*, 2008.

[24] C. Phillips, L. Swiler, "A Graph-Based System for Network-Vulnerability Analysis," *New Security Paradigms Workshop*, 1998.

[25] R. Ritchey, P. Ammann, "Using Model Checking to Analyze Network Vulnerabilities," *IEEE Symposium on Security and Privacy*, 2000.

[26] O. Sheyner, J. Haines, S. Jha, R. Lippmann, J. Wing, "Automated Generation and Analysis of Attack Graphs," *IEEE Symposium on Security and Privacy*, 2002.

[27] Skybox Security, http://www.skyboxsecurity.com/.

[28] RedSeal Systems, http://www.redseal.net/.

[29] R. Lippmann, K. Ingols, "An Annotated Review of Past Papers on Attack Graphs," Lincoln Laboratory Technical Report ESC-TR-2005-054, 2005.

[30] F. Cuppens, R. Ortalo, "LAMBDA: A Language to Model a Database for Detection of Attacks," *Workshop on Recent Advances in Intrusion Detection*, 2000.

[31] S. Templeton, K. Levitt, "A Requires/Provides Model for Computer Attacks," *New Security Paradigms Workshop*, 2000.

[32] R. Ritchey, B. O'Berry, S. Noel, "Representing TCP/IP Connectivity for Topological Analysis of Network Security," *18th Annual Computer Security Applications Conference*, 2002.