Future Generation Computer Systems 27 (2011) 67–77

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A potential HTTP-based application-level attack against Tor

Xiaogang Wang [a,b,*], Junzhou Luo [a], Ming Yang [a], Zhen Ling [a]

[a] School of Computer Science and Engineering, Southeast University, Nanjing, 210096, PR China

[b] Changzhou College of Information Technology, Changzhou, 213164, PR China

ARTICLE INFO

Article history:
Received 30 November 2009
Received in revised form
30 March 2010
Accepted 16 April 2010
Available online 24 April 2010

Keywords:
Web traffic pattern
Application-level attack
Anonymity
Tor

ABSTRACT

Tor has become one of the most popular overlay networks for anonymizing TCP traffic, however, the anonymity of Tor clients is threatened by various attacks exploiting traffic analysis or Tor's design features. Although considerable effort has been made to secure and improve Tor networks, little attention has been paid to various application-level attacks against Tor. In this paper, we present a potential HTTP-based application-level attack against Tor, which exploits both Tor's design features and HTTP's vulnerability to man-in-the-middle attacks. Such an application-level attack can efficiently and effectively compromise the anonymity of clients without using invasive plugins like Java or any other active content systems in a web browser, posing a serious threat to Tor. Our analytical and empirical results validate the feasibility and effectiveness of the attack. Based on our analysis of the potential attack mechanism, we propose corresponding countermeasures to thwart such potential application-level attacks against Tor, thereby effectively securing and improving Tor networks. Since the fundamental vulnerability exposed by this paper is not specific to web browsing via Tor but rather to the problem of other low-latency applications based on TCP streams, our study is critical for securing and improving low-latency anonymous communication systems.

1. Introduction

With the rapid growth and public acceptance of the Internet and the pervasive deployment of various wireless technologies, increasing attention has been focused on concerns about cyberspace security and privacy. Especially, the anonymity has become a necessary and legitimate aim in many applications such as anonymous web browsing [1], instant messaging and location-based services (LBS). In these scenarios, conventional encryption alone cannot achieve the anonymity required by participants [2]. Therefore, increasing effort has been made to develop effective and practical anonymous techniques. Since Chaum [3] pioneered the basic idea of anonymous communication techniques early in 1981, a number of anonymous communication systems have been designed to provide anonymity to the communicating parties.

Existing anonymous communication systems can be classified into two categories: message-based (high-latency) and flow-based (low-latency) techniques. The message-based technique [4] achieving nearly perfect anonymity only supports applications that can tolerate high latencies such as E-mail, while the flow-based technique [5,6] supports various low-latency applications such as web browsing and instant messaging. Tor [7], a circuit-based low-latency anonymous communication system, has become one of the most popular overlay networks in existence for anonymizing TCP traffic, such as anonymous web browsing on the Internet. The advantage of the scheme is owing to the implementation of strong, layered cryptography combined with a wide network of onion routers located across different administrative domains all over the world. We use the terms router and onion router synonymously throughout this paper.

Although low-latency anonymous communication systems can achieve efficient and timely delivery of packets, they are vulnerable to various attacks exploiting traffic analysis [8,9] or non-traffic analysis [10,11]. Traffic analysis attacks are the practices of inferring sensitive information from communication patterns, where features such as packet timings, sizes and counts can be used to correlate network flows and compromise the anonymity. Non-traffic analysis attacks exploit specific internal vulnerabilities of anonymous communication systems. Although a number of countermeasures have been proposed to defend against attacks primarily launched from lower protocol levels [12,13], little attention has been focused on various application-level attacks against anonymous communication systems. Specifically, web browsing that is based on the Hypertext Transport Protocol (HTTP) can be tricked into initiating specified web connections

* Corresponding author at: School of Computer Science and Engineering, Southeast University, Nanjing, 210096, PR China. Tel.: +86 25 8379 5595; fax: +86 25 8379 1010.
E-mail address: wxiaog@seu.edu.cn (X. Wang).

0167-739X/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2010.04.007

within a certain time interval to fetch malicious invisible objects, thereby generating a distinctive traffic pattern. The adversary who detects such a traffic pattern can launch a traffic analysis attack effectively and efficiently. Such application-level attacks can make conventional countermeasures ineffective, rendering a big challenge to existing anonymous communication systems. Therefore, more attention need to be devoted to securing and improving low-latency anonymous systems.

In this paper, we present a potential HTTP-based application-level attack against Tor, which exploits both Tor's design features and HTTP's vulnerability to man-in-the-middle attacks. We adopt the assumption that an adversary can control multiple routers, which is similar to that adopted by other attacks [11,14]. The potential attack first deploys a malicious Tor exit router to monitor a web request to the target web server. Upon detecting the web request of interest, the exit router will insert a forged webpage or modify the requested webpage received from the target web server. A client's web browser is then tricked into generating malicious web traffic in a pattern designed to be detected by an external observer using traffic analysis. An adversary can confirm the communication relationship between the client and the target web server by comparing the observed traffic pattern with the expected one, thereby compromising the anonymity of Tor clients. The attack is one of the first to exploit the vulnerability of anonymous web browsing via Tor. There are several unique features of this attack. First, the attack is highly efficient and able to identify the client based on only one web request. Second, the attack can achieve high secrecy by supporting normal web browsing. Third, the attack is difficult to detect and counter even if all active content systems such as JavaScript in the client's web browser are disabled. We evaluate the effectiveness of the attack by utilizing a statistical analysis approach as well as developing a prototype implementation of the attack. Our analytical and empirical results validate the feasibility and effectiveness of the potential attack.

Based on our analysis of the potential attack mechanism, we propose some countermeasures to thwart such potential application-level attacks against Tor. Minimizing the chance of choosing malicious routers can effectively mitigate the attack. Furthermore, an approach to abnormal traffic detection can be used to detect the forged webpage inserted by the malicious exit router, and the encryption of webpage based on HTTPS can prevent the malicious exit router from modifying the requested webpage, thus providing effective defenses against such attacks.

The remainder of this paper is organized as follows: We review previous work in Section 2. The procedure of web browsing via Tor is illustrated in Section 3. We present the details of the potential HTTP-based application-level attack including the basic principle and two attack schemes in Section 4. We analyze the effectiveness of the attack by utilizing a statistical analysis approach in Section 5. Our experimental result on Tor is presented in Section 6, validating our findings. Some countermeasures against the attack are discussed in Section 7. We conclude the paper along with some future research directions in Section 8.

2. Related work

Since low-latency anonymous communication systems support various widely deployed low-latency applications, such as web browsing and instant messaging, there has been a great deal of recent research on low-latency anonymity systems. A number of low-latency anonymous communication systems [5,6] have been proposed on the basis of mix techniques pioneered by Chaum [3] early in 1981. Notably, Tor [7] uses public key encryption on pre-determined sequence of onion routers to protect the transport of TCP flows, providing both sender anonymity and receiver anonymity. However, with the rapid growth and public acceptance of Tor, a number of techniques exploiting either traffic analysis or non-traffic analysis have been presented to attack Tor.

Existing traffic analysis techniques can be classified into two categories: passive traffic analysis and active watermarking techniques. Passive traffic analysis techniques [15,16] will record the traffic passively and identify the similarity between the sender's outbound traffic and the receiver's inbound traffic, thereby requiring thousands of packets to be observed to achieve reasonable accuracy of attacks. Active watermarking techniques [17,18] can actively embed watermarks into target network flows and correlate flows with similar watermarks, achieving high efficiency of attacks.

A number of traffic analysis attacks have been presented to determine whether the client is communicating with the target web server via Tor. Murdoch and Danezis [8] presented a low-cost traffic analysis technique that allowed an outside observer to infer which routers were being used to relay a circuit's traffic, but could not trace the connection to the initiating client. Zhu et al. [12] proposed the scheme using mutual information for the traffic similarity measurement, while Levine et al. [15] utilized a cross-correlation technique for the traffic similarity measurement. Overlier and Syverson [19] studied an attack that used one compromised router to generate false resource claims to attract traffic. The approach is also based on the traffic timing similarity to associate circuits, thereby locating the hidden servers in the Tor network. Bauer et al. [20] studied an attack that was based on some malicious routers and used traffic analysis techniques. Fu et al. [21] studied a flow marking scheme to degrade the anonymity in a flow-based wireless mix network by utilizing frequency domain analytical techniques. Based on the distribution of traffic timing, Peng et al. [22] analyzed the secrecy of timing-based watermarking traceback techniques. Although the timing-based active watermarking techniques can identify the clients efficiently and achieve high robustness against timing perturbation, little attention has been paid to various application-level attacks exploiting traffic analysis.

Non-traffic analysis attacks exploit Tor's design features to compromise the anonymity of Tor clients. Murdoch [10] investigated an attack to reveal hidden servers in Tor by exploiting the fact that the clock deviations of a target server should be consistent with the server's load. The replay attack against Tor [11] exploits the fundamental design of Tor by duplicating cells to disrupt the normal counter at middle and exit onion routers. Although such a non-traffic analysis attack can quickly confirm the communication relationship, it will break the current anonymous communication circuit and make Tor construct a new circuit, thereby degrading the performance of Tor.

Covert channels [23] have also been proposed to preserve privacy in various scenarios including watermarking and data privacy [24]. Shah et al. [25] proposed *JitterBugs*, a class of inline interception mechanisms that covertly transmitted data by perturbing the timing of input events in order to affect externally observable network traffic. Borders and Prakash [26] designed filters to help detect anomalies in outbound HTTP traffic by using metrics such as request regularity, bandwidth usage, inter-request delay time, and transaction size. Takahashi and Lee [27] assessed VoIP covert channel threats that utilized an IP phone conversation to illicitly transfer information across the network. Our study is also related to the covert channel.

Recently, some attacks have been presented to compromise the anonymity of web browsing via Tor. Such attacks exploit invasive plugins like Java or Flash to establish direct TCP connections for circumventing Tor, allowing a third party to identify Tor clients. Especially, Abbott et al. [14] presented the browser-based attack against Tor by exploiting JavaScript instead of using invasive plugins, such as Flash, Java, and ActiveX Controls. In order to guarantee the anonymity of web browsing, some countermeasures have been proposed to defend against such attacks. Crispo et al. [28]

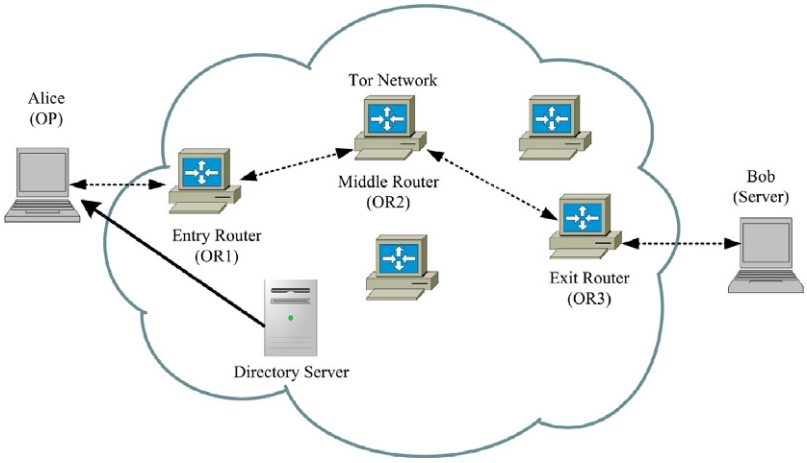


Fig. 1. Tor network.

(a) Tor control cell format. (b) Tor relay cell format.

Fig. 2. Tor cell format.

proposed some alternative means to provide WWW browser security. Most anonymizing systems can warn clients to disable active content systems in their browsers. However, the disadvantage of this defense is that disabling the active content systems would preclude the use of many popular web services in the process. The Semantic Link Network (SLN) [29] is a loosely coupled semantic data model for managing Web resources, which can be used to discover the phishing target of a suspicious webpage [30]. In addition, some anonymity protection tools such as *Polipo* can be used to act as a web proxy with advanced filtering capabilities for protecting privacy.

The potential HTTP-based application-level attack exploits only the features of Hypertext Markup Language (HTML) combined with compromised routers in Tor, making the attack more difficult to detect and counter. Therefore, more attention need to be focused on defending against such application-level attacks.

3. Web browsing via Tor

In this section, we first introduce the Tor network, then we present the procedure for constructing a circuit in Tor, and finally we discuss the procedure of web browsing via Tor in detail.

3.1. Tor network

Tor is a popular overlay network for anonymous communication on the Internet, which is an open source project providing anonymity service for TCP applications. It exploits onion routing that is based on layered encryption to hide the source of TCP traffic. Fig. 1 shows the fundamental architecture of the Tor network consisting of four basic entities.

(1) Alice (i.e., Client). The client runs local software called an onion proxy (OP) to construct circuits and anonymize the client data into the Tor network.
(2) Bob (i.e., Server). The server runs various TCP applications such as a web service and anonymously communicates with Alice.
(3) Onion routers (OR). Onion routers are special proxies that relay the application data between Alice and Bob. Each OR maintains a Transport Layer Security (TLS) connection to every other OR or OP. A circuit is a path of three onion routers by default through the Tor network, namely an entry onion router OR1, a middle onion router OR2 and an exit onion router OR3, respectively.
(4) Directory servers. They are responsible for maintaining and providing onion router information for Tor clients.

Functions of onion proxy, onion router and directory server are all integrated into the same Tor software package. A client can edit a Tor's configuration file and configure a computer to have any combination of those functions.

To send TCP data through a circuit, a client first chooses three onion routers and then constructs a Tor circuit. The onion router information can be downloaded from directory servers in the Tor network. The entry router knows the client's identity, i.e., the client's IP address. The exit router knows both the destination's identity and the actual TCP data sent, and the middle router simply exchanges encrypted data between the entry router and the exit router along a particular circuit. The application data is packed into equal-sized cells and encrypted in the onion-like fashion. All cells are successively encrypted with three session keys that have been negotiated by OP with each OR in the path. Therefore, anyone monitoring the encrypted traffic cannot use cell size to help identify a client.

The structure of the Tor cell is illustrated in Fig. 2. Each cell is 512 bytes, and consists of a header and a payload. The three-byte header includes a circuit identifier (*CircID*) and a command (*CMD* or *Relay*). This header is not encrypted in the onion-like fashion so that the intermediate Tor routers can see it. The 509-byte payload is encrypted in the onion-like way. There are two types of cells: control cells in Fig. 2(a) and relay cells in Fig. 2(b).

Control cells (*CMD*) are responsible for managing circuits, where the *CELL_CREATE* and *CELL_CREATED* cells can be used to set up a new circuit, and the *CELL_DESTROY* cell can be used to tear down a circuit. Relay cells (*Relay*) are used to send relay commands and application data along a circuit. A relay cell has an additional relay header at the front of the payload, which contains a stream identifier (*StreamID*), an end-to-end checksum for integrity checking (*Digest*), the length of the payload (*Len*), and a relay command (*R_CMD*). The entire contents of the relay header and the relay cell payload are encrypted or decrypted together as the relay cell moves along the circuit. There are numerous types of relay commands that routinely traverse the circuit, such as

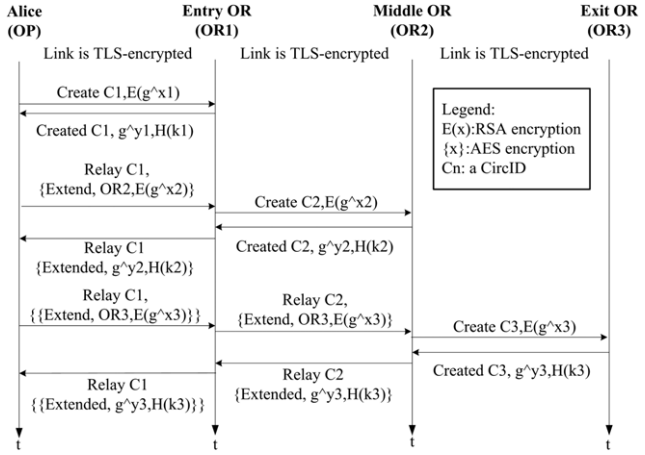


Fig. 3. Procedure of circuit construction.

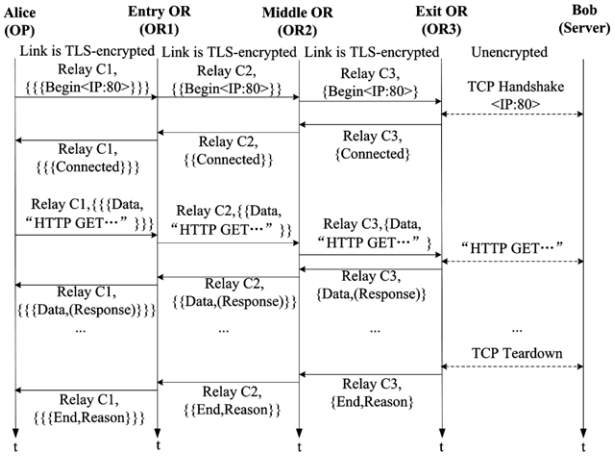


Fig. 4. Web browsing over a Tor circuit.

RELAY_EXTEND used to extend a hop, *RELAY_BEGIN* used to open an application stream, *RELAY_DATA* used to forward the data, and *RELAY_END* used to close a stream cleanly. We will explain them in later sections.

3.2. Circuit construction

To construct a circuit, OP first chooses three routers preferably with high bandwidth and high uptime from the locally cached directory. Since the exit router, acting as a proxy for the client, relays the data between the initiating client and the requested web server, it should have an exit policy supporting the relay of the TCP stream from the client. The OP then constructs the circuit incrementally and negotiates secret keys with the three onion routers one by one. The procedure of circuit construction is illustrated in Fig. 3.

OP first sets up a TLS connection with the entry router OR1 by using the TLS protocol. Then tunneled through this connection, OP sends OR1 a *CELL_CREATE* cell and negotiates a key $K_1 = g^{x1y1}$ with OR1 by using the Diffie–Hellman (DH) key agreement protocol, where the parameter g is usually called a generator. OR1 responds with a *CELL_CREATED* cell, meaning a successful creation of a 1-hop circuit $C1$ between OP and OR1.

To extend the circuit one hop further, the OP sends OR1 a *RELAY_EXTEND* cell, specifying the address of the middle router OR2. Upon receiving this cell, OR1 first decrypts the cell and negotiates secret keys with OR2 to create a second segment $C2$ of the 2-hop circuit, and then sends OP a *RELAY_EXTENDED* cell encrypted by AES in the counter mode (AES-CTR). OP will decrypt the *RELAY_EXTENDED* cell and use the information to create the corresponding keys with OR2. Similarly, OP can create a 3-hop circuit by sending OR2 a *RELAY_EXTEND* cell through the established 2-hop circuit. When OR2 decrypts the cell, it discovers that the cell is meant to create another segment of the circuit to OR3. OR2 consequently negotiates with OR3 and sends a *RELAY_EXTENDED* cell back to *OP*. OP will decrypt the *RELAY_EXTENDED* cell and use the information to create the corresponding keys with OR3. Therefore, a 3-hop circuit is successfully constructed.

3.3. Web browsing over a Tor circuit

To browse the Internet anonymously via Tor, a client must use an HTTP proxy such as *Polipo* so that his traffic will be diverted through Tor rather than sent directly over the Internet. This is especially important because the browser will not automatically send DNS queries through Tor. The procedure of web browsing over a Tor circuit is illustrated in Fig. 4, where the circuit has already been constructed.

To access a web server (i.e., Bob), Alice's web browser will ask her OP, which is implemented as a SOCKS proxy locally, to establish a connection to Bob.

When OP learns the destination IP address and port, it sends a *RELAY_BEGIN* cell to the exit router OR3. The cell is encrypted as $\{\{\{Begin\langle IP, Port\rangle\}_{k3}\}_{k2}\}_{k1}$, where each subscript refers to the key used for encryption of one layer of onion skin. When OR3 removes the last layer of onion skin by decryption, it recognizes the request of opening a TCP stream to the *port* at the destination *IP*, which belongs to Bob. As a result, OR3 acts as a proxy of the client and establishes a TCP connection to Bob by using the TCP handshake protocol. When the TCP connection to Bob is established, OR3 sends a *RELAY_CONNECTED* cell back to Alice's OP. The OP then accepts data from Alice's web browser and sends a *RELAY_DATA* cell to the exit router OR3, which is encrypted as $\{\{\{Data, \text{“}HTTP, GET \ldots\text{”}\}_{k3}\}_{k2}\}_{k1}$.

When OR3 recognizes the *GET* request initiated by OP, it asks Bob to return the requested webpage, and sends a *RELAY_DATA* cell back to Alice's OP in response to OP's web request. The whole process is transparent to Alice, although she needs to configure her web browser to use the OP.

After delivering the requested data, the web server will close the TCP connection to OR3. OR3 then sends a *RELAY_END* cell along the circuit to OP and sends nothing more along the circuit for that stream. In the current Tor implementation, if the exit router receives an unrecognized cell, an error will occur, and the circuit is torn down.

4. HTTP-based application-level attack

We discover a potential HTTP-based application-level attack against Tor. Such an application-level attack can identify a Tor client without using invasive plugins like Java or any other active content systems in a web browser, thereby posing a serious threat to Tor. In this section, we first introduce the basic principle of the attack and then present two potential attack schemes.

4.1. Basic principle

We assume that an adversary can control multiple routers, which is similar to that adopted by other attacks [11,14]. The assumption is valid, since Tor is operated in a voluntary manner. The basic principle of the attack resides in the fact that Tor's design features combined with HTTP's vulnerability to man-in-the-middle attacks can be exploited to launch a potential HTTP-based application-level attack effectively and efficiently. An adversary can use a malicious exit router in Tor to conduct a man-in-the-middle attack by modifying the target webpage. When a client initiates a web request to a target web server, a malicious exit

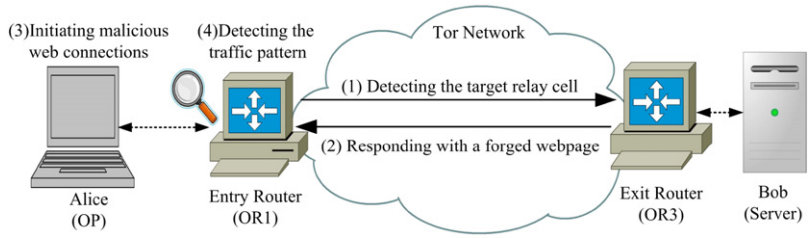


Fig. 5. Forged webpage injection attack.

router on the circuit will either inject a new forged webpage or just modify the target webpage fetched from the web server. Such a malicious webpage contains specified web links corresponding to invisible objects. Upon receiving the malicious webpage, the web browser will be tricked into initiating malicious web connections to fetch those invisible objects immediately. As a result, an accomplice of the adversary at the entry router on the circuit will detect such a distinctive traffic pattern. By comparing the observed traffic pattern with the expected one caused by the malicious webpage, the adversary can effectively confirm the communication relationship between the initiating client and the target web server.

Based on the basic principle described above, two potential attack schemes can be constructed to compromise the anonymity of Tor clients. Scheme 1 called the *forged webpage injection* attack is more efficient by injecting a forged webpage, while Scheme 2 called the *target webpage modification* attack is stealthier by modifying the target webpage.

4.2. Scheme 1: forged webpage injection attack

The *forged webpage injection* attack can identify a Tor client more efficiently by injecting a new forged webpage at a malicious exit router. The attack procedure illustrated in Fig. 5 is as follows.

- **Step 1: Detecting the target relay cell**. An adversary inserts two malicious routers into the Tor network. One acts as an entry router, and the other acts as an exit router. The exit router is responsible for detecting the target relay cell sent from Tor clients. When the exit router OR3 finds that OP is attempting to access the target web server by checking the received *RELAY_BEGIN* cell, it first records the circuit identifier (*CircID*) and the stream identifier (*StreamID*) locally and then responds with a *RELAY_CONNECTED* cell to OP. Upon receiving the *RELAY_CONNECTED* cell, OP sends OR3 the web request that is fetched from client's web browser. The web request is enveloped in a *RELAY_DATA* cell encrypted as $\{\{\{Data, \text{“}HTTP, GET \ldots\text{”}\}_{k3}\}_{k2}\}_{k1}$.
- **Step 2: Responding with a forged webpage**. Once OR3 detects such a *RELAY_DATA* cell from the marked circuit and stream identifiers, it logs the current time and responds with a forged webpage in Fig. 6. Each inserted malicious web link denoted as an *img* tag corresponds to an existing empty *gif* file with the size of 35 bytes.
- **Step 3: Initiating malicious web connections.** Upon receiving such a *RELAY_DATA* cell containing the forged webpage, the web browser is tricked into initiating malicious web connections to fetch those required images within a specified time interval $\boldsymbol{w}$. OP will actually send multiple *RELAY_BEGIN* cells to OR3 along the same circuit. When these cells traverse the circuit and arrive at OR3, OR3 knows that such *RELAY_BEGIN* cells are only used to generate a distinctive traffic pattern. Therefore, OR3 responds with a *RELAY_CONNECTED* cell for each *RELAY_BEGIN* cell. Similarly, upon receiving each *RELAY_DATA* cell requiring a corresponding image, OR3 will responds with an empty image file for each request followed with a *RELAY_END* cell to close the connection. After the specified time interval $\boldsymbol{w}$, the web browser will be required to initiate a new redirecting connection to its original target web server, i.e., Bob.

```
<html>
<head>
<meta http-equiv="refresh" content="w;url= URL_Bob">
</head>
<body>
<img src=" URL_Bob/img1.gif" style="display:none"/>
<img src=" URL_Bob/img2.gif" style="display:none"/>
<img src=" URL_Bob/img3.gif" style="display:none"/>
... ...
<img src=" URL_Bob/imgm.gif" style="display:none"/>
</body>
</html>
```

Fig. 6. Forged webpage.

- **Step 4: Detecting the distinctive traffic pattern**. The entry router needs to carefully detect the traffic flow consisting of forward and backward cells within the specified time interval. Although the entry router cannot recognize the encrypted content of each cell traversed through it, it is able to check the circuit identifier (*CircID*) and cell command (*CMD* or *Relay*) by decrypting each cell, thereby detecting such a distinctive traffic pattern. Each web connection triggered by an *img* tag is required to transfer two relay cells in one direction, and another three cells in the reverse direction. Since the specified number of *img* tags is $\boldsymbol{m}$, the expected distinctive traffic pattern consists of $5\boldsymbol{m}$ relay cells traversed through the entry router. Such a traffic pattern can be detected with a matched-filter approach to capture a segment of traffic cells for the best traffic match. The observed traffic pattern is presented as a feature vector, which is a set of parameters that describe the number of forward and backward cells traversed through the entry router. Once the distinctive traffic pattern is detected, the entry router will log the source IP address and the time of the occurrence of such a traffic pattern. This detecting mechanism can exclude the possibility that the pattern is generated by the circuit construction process. Such a mechanism can also exclude the possibility that the pattern is generated by normal anonymous web browsing process, as the probability of having $2\boldsymbol{m}$ relay cells forwarded and $3\boldsymbol{m}$ relay cells transmitted backwards within the specified time interval for OP is very small.
- **Step 5: Confirming the communication relationship**. The distinctive traffic pattern is considered to be detected if the observed traffic pattern is matched with the expected one based on similarity metric, such as the Euclid distance. The adversary can use Network Time Protocol (NTP) to synchronize the time of the entry router and the exit router. By correlating the forged webpage injection time with the traffic pattern detection time, the adversary can confirm that the distinctive traffic pattern is actually caused by injecting a forged webpage at the exit router.

Furthermore, since the entry router knows the identity of the client who initiates the TCP stream and the exit router knows the web server associated with the TCP stream, the communication relationship between the initiating client and the web server is confirmed.

If the client is accessing other web servers via Tor during the specified time interval, the expected traffic pattern will be disturbed by an additional traffic, thereby decreasing the accuracy of the *forged webpage injection* attack.

4.3. Scheme 2: target webpage modification attack

As the *forged webpage injection* attack requires a specified waiting period for the client to get the target webpage, a potential attack scheme called the *target webpage modification* attack can be constructed to achieve more secrecy. The *target webpage modification* attack deploys a malicious Tor exit router to modify HTTP traffic passing through it. Some invisible web links corresponding to empty images are inserted into the target webpage rather than into a new forged webpage. Those inserted links accompanying with original links contained in the target webpage will also generate a distinctive traffic pattern. The attack procedure is as follows:

- **Step 1: Detecting the target relay cell.** An adversary first inserts two malicious routers into the Tor network. One acts as an entry router, and the other acts as an exit router. The exit router is also responsible for detecting the target relay cell sent from Tor clients.
- **Step 2: Modifying the target webpage.** Whenever the malicious exit router detects a web request to the target web server, it modifies the target webpage destined for the Tor client by inserting some malicious web links corresponding to empty small images.
- **Step 3: Initiating mixed web connections.** Upon receiving the modified webpage, the Tor client's web browser is tricked into initiating mixed web connections corresponding to both the inserted links and those originally contained in the target webpage.
- **Step 4: Detecting the distinctive traffic pattern.** The malicious entry router only needs to log traffic cells passing through it on each circuit, attempting to detect the distinctive traffic pattern.
- **Step 5: Confirming the communication relationship.** The distinctive traffic pattern can be detected using traffic analysis. By correlating the webpage modification time with the traffic pattern detection time, the adversary can effectively confirm the communication relationship between the initiating client and the target web server, thereby revealing the client's identity.

Since the *target webpage modification attack* only works by modifying the target webpage instead of inserting a new forged webpage, the accuracy of the attack may be decreased due to the additional traffic caused by the target webpage itself. However, the attack can be repeatedly launched targeting a certain client to improve the accuracy of the attack. If the target webpage contains less web links and the size of the target webpage is smaller, the circuit will contain less "noise". Therefore, an adversary will have a non-trivial chance of identifying a client in a reasonable amount of time.

5. Analysis

In order to study the potency of the discovered attack, we first analyze the effectiveness of the attack under full control and partial control of malicious routers on a given circuit, respectively. Then, we discuss the threat effect of the attack.

5.1. Controlling entry and exit routers

The success of the potential attack relies on the assumption that the adversary controls the entry and exit routers on a given circuit. This is valid due to the principle of voluntary-oriented operation and Tor's router selecting scheme which prefers high-bandwidth, high-uptime routers in Tor. Suppose that there are N routers, of which K are malicious. Since the client uses one circuit at a time in the attack, the adversary can compromise an unfair percentage of entry and exit routers by configuring its routers and exaggerating its resource claims. As long as a router has a self-designed exit policy enabling access to external services, this router can become an exit router. If a router has a mean time between failures (MTBF) not less than the median for active routers or at least 10 days, it becomes a stable router. This set of criteria is not difficult to meet in the real world.

Let b_i be the bandwidth advertised by the ith router. Then the probability p_i that the ith router is chosen by OP is $b_i/(\sum_{j=1}^{N} b_j)$. Therefore, the probability that a malicious router can be chosen as an entry router on a given circuit is p_i $(i = 1, \ldots, K)$. As a router is not allowed to be chosen twice for the same circuit according to the Tor protocol specification, another malicious router has the chance p_e to be chosen as an exit router on the same circuit. We have,

$$p_e = \sum_{\substack{j=1 \\ j \neq i}}^{K} \frac{p_j}{1 - p_i}. \tag{1}$$

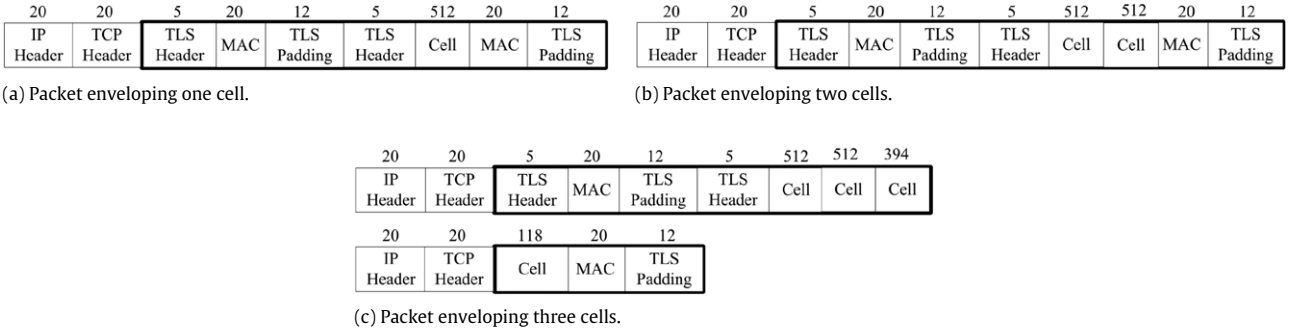
Therefore, the chance S_{attack} that an adversary is able to control the entry and exit routers successfully on a given circuit can be computed as follows:

$$S_{attack} = \sum_{i=1}^{K} p_i.p_e = \sum_{i=1}^{K} p_i. \left(\sum_{\substack{j=1 \\ j \neq i}}^{K} \frac{p_j}{1 - p_i} \right). \tag{2}$$

This derivation implies that the chance S_{attack} increases as the probability p_i increases. As long as p_i $(i = 1, \ldots, K)$ is large enough by exaggerating resource claims, an adversary can succeed in launching the potential HTTP-based application-level attack with a high probability. The analysis result is consistent with the observations made by Bauer et al. [20], which indicates that an adversary can compromise approximately 46% circuits with 9% malicious routers within Tor.

Furthermore, we suppose that Tor uses N routers in the network, v of which exit port 80, and the adversary inserts K evil routers in the network, of which m are exit routers that embed malicious web links. The m exit routers can be noticed by Tor clients, but the other K–m routers only log data, and give no indication of malice. We assume that all Tor routers are equal if the adversary has average bandwidth. Then m/v of all Tor circuits will receive a malicious webpage embedded with malicious web links, and the Tor client has a K/N chance of having his web traffic go through an evil entry router. Therefore, there is a negligible chance that the client will not be identified by correlating the client with the logged web server.

The detection rate of the potential attack may be decreased due to Internet traffic dynamics and other streams. For example, while detecting the distinctive traffic pattern, the entry router may receive other unrelated streams within the specified time interval. In addition, the web browser may generate web traffic with a pattern similar to the distinctive one within the time interval, when rendering some webpages. Based on accessing 100 different web servers via Tor, we have not recorded such occasions. This confirms that the false positive rate of the attack is low. Furthermore, the effectiveness of the attack can be improved by



(a) Packet enveloping one cell.

(b) Packet enveloping two cells.

(c) Packet enveloping three cells.

Fig. 7. Packet format enveloping Tor cell(s).

using a threshold-based traffic detection method. Therefore, an adversary can achieve high detection rates and low false positive rates by choosing an appropriate number of images and the time interval.

5.2. Controlling exit routers only

In the potential attack, the entry router is required to detect the distinctive traffic pattern on the basis of Tor cell information observed. The attack seems to be partially mitigated by choosing a specific set of trusted entry routers to make the adversary control exit routers only. However, the requirement of a malicious entry router is not necessary in the attack if an adversary can sniff the packets transmitted via the link between OP and the entry router. As each packet transferred on a TLS link corresponds to some Tor cells enveloped in the packet, an adversary can correlate the traffic pattern of packets with that of Tor cells. According to the lengths of packets sniffed, an adversary can also detect the distinctive traffic pattern generated by the flow of encrypted packets, which corresponds to the specific malicious webpage. Without loss of generality, we assume that the Maximum Transmission Unit (MTU) of the link between OP and the entry router is 1500 bytes.

The structure of the IP packet that envelops Tor cell(s) is presented in Fig. 7. Such a packet traverses through the TLS link between OP and the entry router.

Since cells may be buffered in a circuit queue at the entry router according to the Tor protocol specification, more Tor cells will be packed into one IP packet for improving efficiency. Fig. 7(a) illustrates the structure of the IP packet that envelops only one cell. The packet consists of an IP header, a TCP header, an empty TLS application record and a TLS application record containing only one cell. The TLS application record includes a TLS header (5 bytes), a TLS message (not to exceed 2^{14} bytes), a MAC (Message Authentication Code, 20 bytes) and a TLS padding field (12 bytes). A TLS header has a content type field, a version field and a length field. The content type field identifies the record layer protocol type contained in this record, with the length of one byte. In our case, we concern the TLS application record with the content type of 23. The version field identifies the major or minor version of TLS for the contained message, with the length of 2 bytes. The length field identifies the length of protocol message(s), not to exceed 2^{14} bytes. Fig. 7(b) illustrates the structure of the IP packet that envelops two Tor cells, and the size of the packet is 1138 bytes. Since the size of the IP packet that envelops three cells exceeds the MTU (1500 bytes), this IP packet will be segmented into two packets: one with the size of 1500 bytes and the other with the size of 190 bytes. The structure of the IP packet that envelops three cells and the way how to segment a packet with the size over the MTU is illustrated in Fig. 7(c).

An adversary can correlate the traffic pattern generated by the flow of Tor cells with that generated by the flow of the encrypted IP packets on the TLS link. Either a *RELAY_BEGIN* cell or a *RELAY_CONNECTED* cell can be mapped into an encrypted IP packet, with the size of 626 bytes in all. As the embedded malicious links correspond to small empty images, each *RELAY_DATA* cell can also be mapped into an IP packet, thereby generating a distinctive web traffic pattern corresponding to the malicious webpage that is modified at the malicious exit router. By appropriately choosing web links for those empty images, the adversary can recognize the distinctive web traffic pattern generated by the flow of sniffed IP packets, thereby identifying the Tor client.

5.3. Threat effect

The potential attack relies on the assumption that the adversary controls the entry and exit routers on a given circuit. Obviously, the larger number of onion routers that the adversary can control, the higher probability of attack success that will be achieved. In the current Tor implementation, an adversary can compromise an unfair percentage of entry and exit routers by configuring its routers and exaggerating its resource claims.

Since the attack relies on only one secret malicious webpage, the attack is secret and efficient for identifying the communication relationship of a TCP stream. The detection of the target web request is quite simple and can be carried out quickly and accurately by the exit router. There is no need for the attack to select cells of TCP stream data, while the *Replay Attack* [11] needs to identify and select appropriate cells.

The *forged webpage injection* attack seems more conspicuous than the *target webpage modification* attack, but Tor's low speed may make such a behavior less conspicuous. In the *forged webpage injection* attack, Alice can see nothing strange on her web browser except waiting for some additional time. Since the client's existing circuit is still available, the performance of Tor will not be degraded more by the attack, thus preventing Tor directory authorities from monitoring the activities of routers and blacklisting those routers with misbehaved activities. The HTTP-based application-level attack can be used to randomly profile both clients and servers hidden by Tor. Notably, the attack can work even if all active content systems in the browser are disabled, thereby posing a serious threat to Tor.

The fundamental vulnerability exposed by this paper is not specific to web browsing via Tor but rather to the problem of applications based on TCP streams. Therefore, our study is critical for securing and improving Tor.

6. Evaluation

We developed a prototype implementation of the potential HTTP-based application-level attack based on Tor 0.2.0.31 to investigate its effectiveness and feasibility.

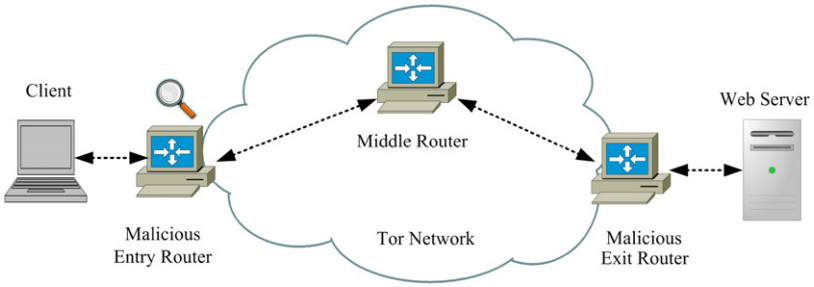


Fig. 8. Experiment setup.

6.1. Experiment setup

We conducted experiments in a partially controlled environmental setup in Fig. 8. Two malicious onion routers were deployed as the Tor entry onion router and exit onion router, respectively. The entry onion router and the client were located at one university campus, while the exit onion router and the target web server were located at another university campus. All computers were on different IP address segments connecting to the China Education and Research Network (CERNET). We experimented on webpages generated by ourselves in order to avoid legal issues.

The Tor client used *Internet Explorer* 7.0 running on *Windows XP Professional SP2* to access the target web site http://jssec.seu.edu.cn through the patched OP using *Polipo* 1.0.4. The OP code was modified for debugging purposes to construct circuits through the designated entry and exit routers. By setting the Tor's configuration file and corresponding parameters, such as *EntryNodes*, *ExitNodes*, *StrictEntryNodes*, and *StrictExitNodes*, we made the client select our malicious routers on the circuit. The entry router was configured to log the information of every cell traversed through it, which would be exploited later in identifying the suspected client in Scheme 1 and Scheme 2, respectively. The entry router's exit policy only allowed it to be used as an entry or middle router. The middle router was selected by using the default Tor routing selection algorithm. The exit router was designed to inject a forged webpage in Scheme 1 and modify a target webpage in Scheme 2, respectively. The default exit policy from Tor was configured for our malicious exit router.

6.2. Experimental results and analysis

When implementing the potential application-level attack, we made the client access the same web server every 20 s to validate the accuracy of the attack. In the *forged webpage injection* attack, the revised code for the exit router was implemented to inject a new forged webpage whenever it detected a web request to the target web server http://jssec.seu.edu.cn. The exit router was also required to record the time of injecting the forged webpage. The forged webpage contained 8 malicious *img* tags and the refresh time interval was 10 s. Each *img* tag with an invisible attribute corresponded to an existing empty *gif* file with the size of 35 bytes, which could be packed into one cell. In the *target webpage modification* attack, the exit router was designed to insert designated number of malicious web links into the target webpage whenever it detected a web request to the same target web server http://jssec.seu.edu.cn. The exit router was also required to record the time of inserting the malicious web links. Each inserted malicious web link denoted as a malicious *img* tag corresponded to an existing empty *gif* file with the size of 35 bytes. The observed traffic pattern was presented as a feature vector, which was a set of parameters that described the number of forward and backward cells traversed through the entry router. The distinctive traffic pattern was detected if the observed traffic pattern was matched with the expected one based on Euclid distance measure.

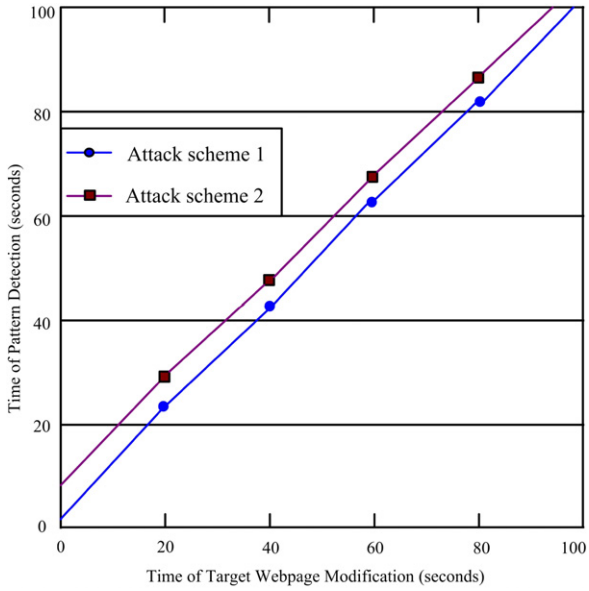


Fig. 9. Correlation between the time of webpage modification and the time of pattern detection.

Since the entry router knew the identity of the client who initiated the TCP stream and the exit router knew the web server associated with the TCP stream, we could confirm the communication relationship between the client and the target web server by correlating the webpage modification time with the traffic pattern detection time.

We used the correlation coefficient ρ to measure the strength of correlation between the webpage modification time and the traffic pattern detection time. Correlation coefficient was defined as follows:

$$\rho_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}.\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \tag{3}$$

where x was the time when the target webpage was modified at the exit router and y was the time of the occurrence of the distinctive traffic pattern at the entry router. $\bar{x}$ and $\bar{y}$ were the mean values of x and y, respectively. The result illustrated in Fig. 9 shows that the delay between the webpage modification time and the pattern detection time is less than 4 s in Scheme 1 and the actual correlation coefficient is one. The delay depends on both the performance of the circuit being used and the number of web links contained in the malicious webpage. The delay in Scheme 2 may

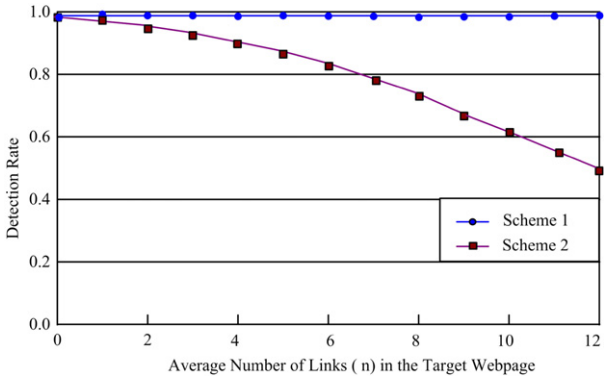


Fig. 10. Detection rate comparison between Scheme 1 and Scheme 2.

exceed 5 s and the actual correlation coefficient is close to one due to the dynamic size of the target webpage. This strongly confirms the effectiveness of the application-level attack on Tor.

A detection rate was measured as the number of distinctive patterns that were correctly identified over the total of 20 rounds of repetitions during the experiment. Similarly, a false positive rate was measured, for a given client, as the number of distinctive patterns that were falsely identified over the 20 rounds of repetitions. Fig. 10 shows the detection rate comparison between Scheme 1 and Scheme 2, where the detection threshold is 2.

The result shows that the detection rate of Scheme 1 (i.e., the *forged webpage injection* attack) is independent of the average number of links denoted as n in the target webpage, provided that the client stops accessing other web servers via Tor when detecting the distinctive traffic pattern. This confirms the effectiveness of the application-level attack against Tor. The detection rate of Scheme 2 (i.e., the *target webpage modification* attack) decreases as the number of web links contained in the modified webpage increases, which is consistent with the analysis in Section 5. Obviously, there is a performance tradeoff between accuracy and secrecy of the attack.

The empirical result also shows that the detection rate of the application-level attack may be decreased due to the interference caused by a process constructing new circuits while detecting the malicious traffic pattern at the same entry router. Although a higher decoding threshold can achieve a higher detection rate for a given number of links in the target webpage, it may also increase the false positive rate. A separate experiment demonstrates that false positive rates less than 1% can be achieved when the detection threshold is less than 2.

7. Guideline of countermeasures

As Tor has been widely used for anonymizing TCP traffic, various countermeasures against application-level attacks are critical for securing and improving Tor. Based on our analysis of the attack mechanism, we propose some useful defenses against such an application-level attack besides disabling all active content systems in a web browser. Such countermeasures can be used to make the adversaries' job much harder. Specifically, the abnormal traffic detection method can be used to thwart the *forged webpage injection* attack, and the encryption of webpage based on HTTPS can effectively defend against the *target webpage modification* attack.

7.1. Minimizing the chance of choosing malicious routers

The success of the potential application-level attack relies on the assumption that the adversary controls the entry and exit routers on a given circuit. Therefore, minimizing the chance of choosing malicious routers is an effective defense against the attack. There are several ways to minimize the chance that malicious routers are chosen on the client's circuit.

First, if the number of malicious Tor routers is smaller, compared to the total number of Tor routers, a Tor client will have less of a chance to choose a malicious router on his Tor circuit by deploying more normal Tor routers. Second, the circuit construction algorithm may be evolved and only select fully trusted and dedicated ones through strict authentication and authorization, where a reputation system could also help facilitate this countermeasure. In this way, the adversary will have less chance of controlling the entry and exit routers. Therefore, the effectiveness of the application-level attack will be reduced.

The distinctive traffic pattern caused by the malicious webpage can be detected, provided the Tor client keeps using the same Tor circuit within the specified time interval. The Tor algorithm for routing traffic over circuits prefers older circuits, and the length of time that the client keeps using the same Tor circuit is at most ten minutes by default according to the Tor protocol specification. Therefore, preventing the client's browser from browsing over the same circuit may reduce the effectiveness of the attack.

7.2. Detecting abnormal traffic

The *forged webpage injection* attack can work even if all active content systems in a web browser are disabled. Furthermore, the web browser using HTTPS is still vulnerable to the *forged webpage injection* attack. Therefore, the potential attack seems more difficult to detect and counter. However, in the *forged webpage injection* attack, the target webpage will not be delivered to the client until all invisible images specified in the forged webpage have been fetched. Such an abnormal behavior accompanying with the malicious traffic can also be detected by the client. Therefore, detecting the abnormal traffic initiated from the client's browser can be an effective defense against the *forged webpage injection* attack. For example, a secure web browser plugin can be deployed to detect such an abnormal traffic. Whenever an abnormal traffic is detected, the client may be warned to take further actions for defending against the attack. Such a secure web browser plugin can be designed and deployed in a way similar to the method of deploying anonymity protection tools such as *Polipo*, which acts as a web proxy with advanced filtering capabilities for maintaining anonymity.

7.3. Exploiting HTTPS

The *target webpage modification* attack works by modifying the target webpage instead of injecting a new forged webpage. If the client has a chain of trust to the web server, he may access the target web server by using the Hypertext Transfer Protocol over Secure Socket Layer (HTTPS). Since HTTPS is secure against man-in-the-middle attacks by encrypting webpage information, tunneling HTTP over SSL can prevent a malicious exit router from either reading or modifying the data it is transporting. Although such a defense cannot completely address the *forged webpage injection* attack, it can be used to effectively defend against the *target webpage modification* attack.

Since clients will often accept self-signed certificates as valid despite the browser warning, a malicious exit router could thus trick careless clients by replacing webpages with malicious versions that are also signed, but with forged certificates. Therefore, using HTTPS can provide reasonable security against such application-level attacks so long as the client can trust the servers he visits and correctly verifies certificates. Moreover, some sites do not allow a client to communicate with them in a secure fashion. For example, https://www.google.com currently redirects to http://www.google.com. Therefore, exploiting HTTPS is not something a Tor client can implement unilaterally. Every web site that he visits must allow the client to do so.

8. Conclusion and future work

Tor has been widely used for anonymizing TCP traffic, however, the anonymity of Tor clients is threatened by various attacks exploiting traffic analysis or design features of Tor. Although increasing attention has been devoted to securing and improving Tor networks, little attention has been focused on various application-level attacks against Tor. We discover a potential HTTP-based application-level attack against Tor, which can compromise the anonymity of Tor clients by exploiting both Tor's design features and HTTP's vulnerability to man-in-the-middle attacks. A malicious exit Tor router can be deployed as a man-in-the-middle of clients' communications, making clients vulnerable to man-in-the-middle attacks from adversaries that would otherwise be unable to perform such attacks. Also fundamental to the attack is the fact that web browsers can be tricked into initiating malicious web connections within a certain period, generating distinctive network traffic in a pattern designed to be detected by an external observer using traffic analysis. Our analytical and empirical results validate the effectiveness and feasibility of the attack.

Based on our analysis of the potential attack mechanism, we propose corresponding countermeasures to make the adversaries' job much harder. Minimizing the chance of choosing malicious routers can effectively mitigate both the *forged webpage injection* attack and the *target webpage modification* attack. Furthermore, the abnormal traffic detection method can be used to thwart the *forged webpage injection* attack by deploying secure web browser plugins, and the encryption of webpage based on HTTPS can effectively defend against the *target webpage modification* attack. Since the fundamental vulnerability exposed by this paper is not specific to web browsing via Tor but rather to the problem of other low-latency applications based on TCP streams, our study is critical for securing and improving low-latency anonymous communication systems.

Due to Tor's fundamental design and current design of the web, defending against application-level attacks still remains a challenging task and we will investigate it in our future research. Also, we will investigate new attacks against Tor by exploiting other application features and develop corresponding countermeasures for securing and improving Tor networks.

Acknowledgements

This work is supported by National Natural Science Foundation of China under Grant Nos. 60903161, 60903162 and 90912002, China Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 200802860031, Jiangsu Provincial Natural Science Foundation of China under Grant No. BK2008030, China National S&T Major Project under Grant No. 2009ZX03004-004-04, State Key Laboratory of Information Security, Jiangsu Provincial Key Laboratory of Network and Information Security under Grant No. BM2003201, Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant No. 93K-9, and the Opening Project of Shanghai Key Laboratory of Integrate Administration Technologies for Information Security under Grant No. AGK2008002.

References

[1] R. Oppliger, Privacy protection and anonymity services for the World Wide Web (WWW), Future Generation Computer Systems 16 (4) (2000) 379–391.
[2] M. Liberatore, B.N. Levine, Inferring the source of encrypted HTTP connections, in: Proc. 13th ACM Conference on Computer and Communications Security, CCS, Alexandria, VA, USA, 2006, pp. 255–263.
[3] D.L. Chaum, Untraceable electronic mail, return addresses, and digital pseudonyms, Communications of the ACM 24 (2) (1981) 84–90. doi:10.1145/358549.358563.
[4] G. Danezis, R. Dingledine, N. Mathewson, Mixminion: design of a type III anonymous remailer protocol, in: Proc. IEEE Symposium on Security and Privacy, S&P, Berkeley, CA, USA, 2003, pp. 2–15.
[5] M.K. Reiter, A.D. Rubin, Anonymous web transactions with Crowds, Communications of the ACM 42 (2) (1999) 32–48. doi:10.1145/293411.293778.
[6] M.J. Freedman, R. Morris, Tarzan: a peer-to-peer anonymizing network layer, in: Proc. 9th ACM Conference on Computer and Communications Security, CCS, Washington, DC, USA, 2002, pp. 193–206.
[7] R. Dingledine, N. Mathewson, P. Syverson, Tor: the second-generation onion router, in: Proc. 13th USENIX Security Symposium, San Diego, CA, USA, 2004, pp. 303–320.
[8] S.J. Murdoch, G. Danezis, Low-cost traffic analysis of Tor, in: Proc. IEEE Symposium on Security and Privacy, S&P, Oakland, CA, USA, 2005, pp. 183–195.
[9] X. Wang, S. Chen, S. Jajodia, Network flow watermarking attack on low-latency anonymous communication systems, in: Proc. IEEE Symposium on Security and Privacy, S&P, Oakland, CA, USA, 2007, pp. 116–130.
[10] S.J. Murdoch, Hot or not: revealing hidden services by their clock skew, in: Proc. 13th ACM Conference on Computer and Communications Security, CCS, Alexandria, VA, USA, 2006, pp. 27–36.
[11] R. Pries, W. Yu, X. Fu, W. Zhao, A new replay attack against anonymous communication networks, in: Proc. IEEE International Conference on Communications, ICC, Beijing, China, 2008, pp. 1578–1582.
[12] Y. Zhu, X. Fu, B. Graham, R. Bettati, W. Zhao, On flow correlation attacks and countermeasures in mix networks, in: Proc. Workshop on Privacy Enhancing Technologies, PET, Toronto, Ontario, Canada, 2004, pp. 207–225.
[13] W. Yan, E. Hou, N. Ansari, Defending against traffic analysis attacks with link padding for bursty traffics, in: Proc. 5th Annual IEEE System, Man and Cybernetics Information Assurance Workshop, SMC, West Point, NY, USA, 2004, pp. 46–51.
[14] T. Abbott, K. Lai, M. Lieberman, E. Price, Browser-based attacks on Tor, in: Proc. 7th International Symposium on Privacy Enhancing Technologies, PET, Ottawa, Ontario, Canada, 2007, pp. 184–199.
[15] B.N. Levine, M.K. Reiter, C. Wang, M. Wright, Timing attacks in low-latency mix systems, in: Proc. Financial Cryptography, FC, Key West, FL, USA, 2004, pp. 251–265.
[16] M.K. Wright, M. Adler, B.N. Levine, C. Shields, Passive-logging attacks against anonymous communications systems, ACM Transactions on Information and System Security 11 (2) (2008) 1–34. doi:10.1145/1330332.1330335.
[17] Y.J. Pyun, Y.H. Park, X. Wang, D.S. Reeves, P. Ning, Tracing traffic through intermediate hosts that repacketize flows, in: Proc. 26th IEEE International Conference on Computer Communications, INFOCOM, Anchorage, AK, USA, 2007, pp. 634–642.
[18] W. Yu, X. Fu, S. Graham, D. Xuan, W. Zhao, DSSS-based flow marking technique for invisible traceback, in: Proc. IEEE Symposium on Security and Privacy, S&P, Oakland, CA, USA, 2007, pp. 18–32.
[19] L. Overlier, P. Syverson, Locating hidden servers, in: Proc. IEEE Symposium on Security and Privacy, S&P, Berkeley, CA, USA, 2006, pp. 100–114.
[20] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, D. Sicker, Low-resource routing attacks against Tor, in: Proc. ACM Workshop on Privacy in the Electronic Society, WPES, Alexandria, VA, USA, 2007, pp. 11–20.
[21] X. Fu, Y. Zhu, B. Graham, R. Bettati, W. Zhao, On flow marking attacks in wireless anonymous communication networks, in: Proc. 25th IEEE International Conference on Distributed Computing Systems, ICDCS, Columbus, OH, USA, 2005, pp. 493–503.
[22] P. Peng, P. Ning, D.S. Reeves, On the secrecy of timing-based active watermarking trace-back techniques, in: Proc. IEEE Symposium on Security and Privacy, S&P, Berkeley, CA, USA, 2006, pp. 334–348.
[23] D.V. Bailey, D. Boneh, E.J. Goh, A. Juels, Covert channels in privacy-preserving identification systems, in: Proc. 14th ACM Conference on Computer and Communications Security, CCS, Alexandria, VA, USA, 2007, pp. 297-306.
[24] Y. Li, V. Swarup, S. Jajodia, Fingerprinting relational databases: schemes and specialties, IEEE Transactions on Dependable and Secure Computing 2 (1) (2005) 34–45. doi:10.1109/TDSC.2005.12.
[25] G. Shah, A. Molina, M. Blaze, Keyboards and covert channels, in: Proc. 15th Conference on USENIX Security Symposium, Vancouver, BC, Canada, 2006, pp. 59–75.
[26] K. Borders, A. Prakash, Web tap: detecting covert web traffic, in: Proc. 11th ACM Conference on Computer and Communications Security, CCS, Washington, DC, USA, 2004, pp. 110–120.
[27] T. Takahashi, W. Lee, An assessment of VoIP covert channel threats, in: Proc. 3rd IEEE International Conference on Security and Privacy in Communication Networks, Nice, France, 2007, pp. 371–380.
[28] B. Crispo, P. Landrock, V. Matyas Jr., WWW security and trusted third party services, Future Generation Computer Systems 16 (4) (2000) 331–341.
[29] H. Zhuge, Y. Sun, The schema theory for semantic link network, Future Generation Computer Systems 26 (3) (2010) 408–420.
[30] L. Wenyin, N. Fang, X. Quan, B. Qiu, G. Liu, Discovering phishing target based on semantic link network, Future Generation Computer Systems 26 (3) (2010) 381–388.

Xiaogang Wang received the M.Sc. degree in Computer Science and Engineering in 2006 from Southeast University, Nanjing, PR China, where he is currently working toward the Ph.D. degree in Computer Science and Engineering. He is an Associate Professor in the Department of Computer Science and Engineering, Changzhou College of Information Technology, Jiangsu Province, PR China. His current research interests include digital watermark, anonymous communication, and digital identity management.

Junzhou Luo received the Ph.D. degree in Computer Network from Southeast University, PR China, in 2000. He is currently the Dean and Professor in the School of Computer Science and Engineering, Southeast University, PR China. His current research interests include network security and management, grid and service computing, and next generation network architecture. He is a member of the IEEE Computer Society and co-chair of IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design.

Ming Yang received the M.Sc. and Ph.D. degrees in Computer Science and Engineering in 2002 and 2007, respectively, from Southeast University, Nanjing, PR China. He is currently a Lecturer in the School of Computer Science and Engineering, Southeast University, Nanjing, PR China. His current research interests include digital watermark, anonymous communication, and wireless network.

Zhen Ling received the M.Sc. degree in Computer Science and Engineering in 2006, from Southeast University, Nanjing, PR China, where he is currently working toward the Ph.D. degree in Computer Science and Engineering. His current research interests include digital watermark, anonymous communication, and digital identity management.