ISSN: 2277-9655
Impact Factor: 1.852

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Cell Counting Attack and Browser Attack against TOR Network

Prof. Swati Patil[*1], **Mr. Surendra Patil**[2]
[*1] HOD of IT Department, GHRIEM, Jalgaon, India
[2] Computer Science and Engineering, GHRIEM, Jalgaon, India
Swati.patil@raisoni.net

Abstract

The onion router (TOR) allows to hide your identity various software under this categories are available that allows online anonymity network, supporting TCP applications over the Internet. It doesn't allow network surveillance or traffic analysis to get tracked but most of these software used equal size of cells (512B). In this paper we are comparing cell-counting attacks and browser attacks against TOR. Different from cell-counting attacks, these attacks can confirm anonymous communication relationships quickly and accurately by manipulating one single cell and pose a serious threat against Tor. A malicious entry onion router may duplicate, modify, insert, or delete cells of a TCP stream from a sender. The manipulated cells traverse middle onion routers and arrive at an exit onion router along a circuit. Because Tor uses the counter mode AES (AES-CTR) for encrypting cells, the manipulated cells disrupt the normal counter at exit onion routers and decryption at the exit onion router incurs cell recognition errors, which are unique to the investigated cell counting attacks. In Browser Attack, a user's web browser into sending a distinctive signal over the Tor network that can be detected using traffic analysis. It is delivered by a malicious exit node using a man-in-the-middle attack on HTTP. Both the attack and the traffic analysis can be performed by an adversary with limited resources. While the attack can only succeed if the attacker controls one of the victim's entry guards, the method reduces the time required for a traffic analysis attack on Tor from O(nk) to O(n + k), where n is the number of exit nodes and k is the number of entry guards.

Keywords: TOR, TCP, Anonymizer, Anonymity, cell counting, Browser, signal, AES.

Introduction

Any message sent over the internet contains routing information that can be used to identify the sender and receiver of the message. For many users of the internet, this poses a problem. Activists, Whistleblowers and human rights workers might want to be anonymous to avoid reprisals from oppressive governments or corporations. Military and law enforcement personal might want to be anonymous so that they can gather intelligence or conduct sting operations without identifying themselves online. People living in countries or working at companies with censored internet may use anonymity as a way to circumvent censorship measures. To this end, many anonymity systems have been developed with the goal of facilitating anonymous communication online. These anonymity systems are typically divided into two categories: low-latency systems and high-latency systems. High latency systems such as Babel, Mixmaster, and Maximum implement defense measures such as mixing, padding, batching, and reordering in an attempt to protect against a global passive adversary who can observe all networks trace [4]. However, such systems can only be used with high-latency communication methods like email, which limits their utility and also limits their user base. Low-latency systems generally do not attempt to protect against a global passive adversary, but are usable with a much wider variety of applications, including web browsers, chat clients, and video streaming. One popular low-latency anonymity network is Tor [4]. Tor works by routing a user's connection through three onion routers (ORs), which form a circuit and act as a chain of proxies for the connection. Messages being sent over the connection are layered with encryption so that each OR knows only its immediate source and destination. Onion routers are run by volunteers around the world. The routers are coordinated and cataloged by a small set of directory servers that provide information about the Tor network and available routers to Tor clients (which are often called onion proxies or OPs). While it does not protect against a global passive adversary, Tor does try to protect against a more limited adversary who can observe some of the traffic going over the network, or who controls some Tor routers. This is important because anyone can run a Tor router, and Tor users have no guarantee that router operators are not malicious. However, despite its design goals, Tor is commonly assumed to be vulnerable to several classes of attacks by non-global adversaries. In this paper, we will examine two attacks against TOR of those types of attacks: a passive end-to-end correlation attack whereby an attacker controlling the first and last routers in a circuit can compromise the anonymity of streams going through that circuit. While Tor is assumed to be vulnerable to these kinds of attacks, much prior work in this area has been done in simulation or only in theory. We seek to test the effectiveness of these attacks on the deployed Tor network, and to determine whether we can create a better attack by examining metrics of Tor traffic.

For applications that can tolerate high latencies, such as electronic mail, there are systems that achieve nearly perfect anonymity [1]. Such anonymity is difficult to achieve with low latency systems such as web browsing, however, because of the conflict between preventing traffic analysis on the flow of packets through the network and delivering packets in an efficient and timely fashion. Because of the obvious importance of the problem, there has been a great deal of recent research on low-latency anonymity systems. Tor, the second-generation onion router, is the largest anonymity network in existence today. In this paper we describe a new scheme for executing a practical timing attack on browsing the web anonymously with Tor. Using this attack, an adversary can identify a fraction of the Tor users who use a malicious exit node and then leave a browser window open for an hour. With current entry guard implementations, the attack requires the adversary to control only a single Tor server in order to identify as much as 0.4% of Tor users targeted by the malicious node (and this probability can be increased roughly linearly by adding more machines). The targeting can be done based on the potential victim's HTTP traffic

Basic Components and Operation of TOR

In this section, we first introduce the basic components of the Tor network. We then present its operation, including the circuit setup and its usage for transmitting TCP streams anonymously.

Components of the Tor Network

Tor is a popular overlay network for anonymous communication over the Internet. It is an open source project and provides anonymity service for TCP applications [11]. Figure 1 illustrates the basic components of Tor [6]. As shown in Figure 1, there are four basic components:

1) ***Alice* (i.e. *Client*).** The client runs local software called *onion proxy* (*OP*) to anonymizing the client data into Tor.
2) ***Bob* (i.e. *Server*).** It runs TCP applications such as a web service and anonymously communicates with *Alice* as the client over Tor.
3) ***Onion routers (OR).*** Onion routers are special proxies that relay the application data between Alice and Bob. In Tor, Transport Layer Security (TLS) connections are used for the overlay link encryption between two onion routers. The application data is packed into equal-sized cells (512 bytes as shown in Figure 2) carried through TLS connections.
4) ***Directory servers.*** They hold onion router information such as public keys for onion routers. There are directory authorities and directory caches. Directory authorities hold authoritative information on onion routers and directory caches download directory information of onion routers from authorities. The client downloads the onion router directory from directory caches.

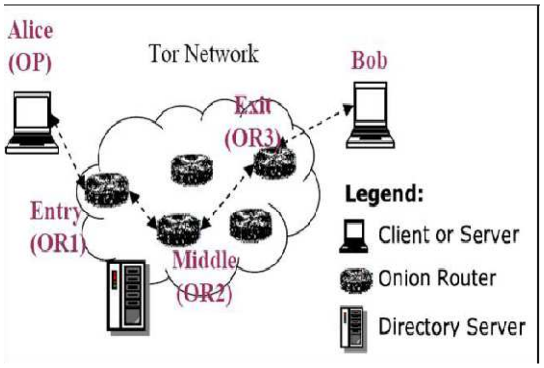

Fig.1: Tor Network

Functions of onion proxy, onion router, and directory servers are integrated into the Tor released software package. A user can edit a configuration file and configure a server to have different combinations of those functions. Figure 2 illustrates the cell format used by Tor. All cells have a three bytes header, which is not encrypted in the onion-like fashion so that the intermediate Tor routers can see this header. The other 509 bytes are encrypted in the onion-like fashion. There are two types of cells: the control cell shown in Figure 2 (a) and relay cell shown in Figure 2 (b). The command field (Command) of a control cell can be: CELL PADDING, used for keep alive and optionally usable for link padding, although not used currently; CELL CREATE or CELL CREATED, used for setting up a new circuit; and CELL DESTROY, used for releasing a circuit. The command field (Command) of a relay cell is CELL RELAY. Notice that relay cells are used to carry TCP stream data from Alice to Bob. The relay cell has an additional header, namely the relay header. There are numerous types of relay commands (Relay Command), including RELAY COMMAND BEGIN, RELAY COMMAND DATA, RELAY COMMAND END, RELAY COMMAND SENDME, RELAY COMMAND EXTEND, RELAY COMMAND DROP, and RELAY COMMAND RESOLVE2. We will explain these commands further in later sections of the paper when we discuss the Tor operations from protocol-level attacks perspective.

Cir_id	Comman	DATA

(a) Tor Cell Format

Cir_i	CMD	Relay	Recogniz	Stream id	Integrity

Length	DATA

(b) TOR Relay Cell Format
Fig. 2: Cell Format by Tor [8]

Selecting a Path and Creating a Circuit

In order to anonymously communicate with applications, i.e., browsing a web server, a client uses a way of source routing and chooses a series of onion routers from the locally cached directory, downloaded from the directory caches [7]. We denote the series of onion routers as the path through Tor [8]. The number of onion routers is referred to as the path length. We use the default path length of 3 as an example in Figure 1 to illustrate how the path is chosen. The client first chooses an appropriate exit onion router *OR*3, which should have an exit policy supporting the relay of the TCP stream from the sender. Then, the client chooses an appropriate entry onion router *OR*1 (referred to as entry guard and used to prevent certain profiling attacks [12]) and a middle onion router *OR*2. Once the path is chosen, the client initiates the procedure of creating a circuit over the path incrementally, one hop at time. The procedure of creating a circuit when the path has a default length of 3. Tor uses TLS/SSLv3 for link authentication and encryption.

Cell-Counting-Based Attack

In this section, we first show that the size of IP packets in the Tor network is very dynamic. Based on this finding, we then introduce the basic idea of the cell-counting-based attack and list some challenging issues related to the attack and present solutions to resolve those issues.

Dynamic IP Packet Size of Traffic over Tor

In Tor, the application data will be packed into equal-sized cells. Nonetheless, via extensive experiments over the Tor network, we found the size of IP packets transmitted over Tor is dynamic. It can be observed that the size of packets from the sender to the receiver is random over time, and a large number of packets have varied sizes, other than the cell size or maximum transmission unit (MTU) size.

Basic Idea of Cell-Counting-Based Attack

The basic idea is as follows. An attacker at the exit onion router first selects the target traffic flow between Alice and Bob. The attacker selects a random signal, chooses an appropriate time, and changes the cell count of target traffic based on the selected random signals. In this way, the attacker is able to embed a signal into the target traffic from Bob. The signal will be carried along with the target traffic to the entry onion router connecting to Alice. An accomplice of the attacker at the entry onion router will record the variation of the received cells and recognize the embedded signal. If the same pattern of the signal is recognized, the attacker confirms the communication relationship between Alice and Bob.

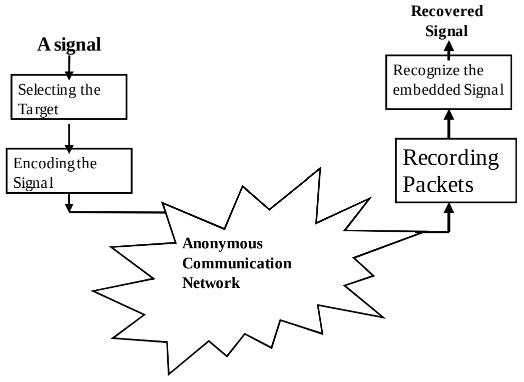

Fig 3: Cell-counting-based attack.

Browser Attacks

To browse the Internet anonymously using Tor, a user must use an HTTP proxy such as Privacy so that traffic will be diverted through Tor rather than sent directly over the Internet. This is especially important because browsers will not automatically send DNS queries through a SOCKS proxy. However, pieces of software that plug into the browser, such as Flash, Java, and ActiveX Controls, do not necessarily use the browser's proxy for their network traffic. Thus, when any of these programs are downloaded and subsequently executed by the web browser, any Internet connections that the programs make will not go through Tor first. Instead, they will establish direct TCP connections, compromising the user's anonymity, as shown in Figure 4. This attack allows a website to identify its visitors but does not allow a third party to identify Tor users visiting a given website. These active content systems are well-known problems in anonymous web-browsing, and most anonymizing systems warn users to disable active content systems in their browsers. In October 2006, Fort Consult Security [2] described how to extend this attack so that parties could identify Tor users visiting a website they do not control. The attacker uses a malicious exit node to modify HTTP traffic and thus conduct a man-in-the-middle attack, as shown in Figure 3. In particular, it inserts an invisible iframe with a reference to some malicious web server and a unique cookie. In rendering the page, the web browser will make a request to the web server and will retrieve a malicious Flash application. If Flash is enabled in the browser, then the Flash movie is played invisibly. The Flash application sends the cookie given to the user directly to the evil web server, circumventing Tor. The web server can then identify which web pages were sent to which users by matching the cookies with the Flash connections. In other words, all Tor users who use HTTP through that exit node while Flash is enabled will have their HTTP traffic associated with their respective IP addresses. However, if we assume that the number of malicious Tor servers is small compared to the total number of Tor servers, a normal user will get a malicious exit node only once in a while. As a result, this attack only works to associate traffic with the particular user for the length of time that the user keeps the same Tor circuit, or at most ten minutes by default.

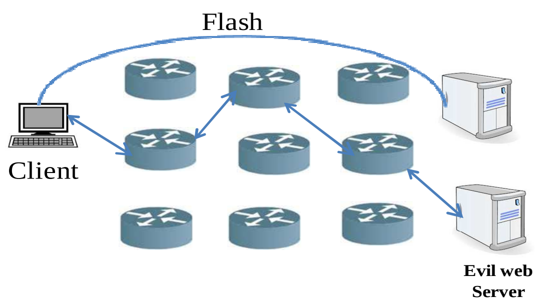

Fig 4: A browser attack using Flash included in a website

A Browser-Based Timing Attack

The attack of a Tor client without using invasive plugins like Java or Flash but with JavaScript instead.

JavaScript alone is not powerful enough to discover the client's IP address, but combined with a timing attack similar to the one presented by Øverlier and Syverson [1], an adversary has a non-trivial chance of discovering a client in a reasonable amount of time. To implement this attack using only the HTML meta refresh tag, but the JavaScript version is simpler so we discuss it first. This attack is partially mitigated by entry guards, which has become a standard feature of Tor.

1. The attacker first sets up the necessary resources.
(a) The attacker inserts two malicious nodes into the Tor network: one to act as an entry node, and the other to act as an exit node.
(b) The attacker sets up a web server that receives and logs JavaScript connections.
2. The malicious exit node modifies all HTTP traffic destined for Tor clients to include an invisible JavaScript signal generator that generates a unique signal for each Tor client.
3. The Tor client's web browser executes the JavaScript code, sending a distinctive signal to the web server. This traffic passes through the Tor circuit, and the client is still anonymous.
4. Approximately every ten minutes, the Tor client chooses a new circuit. Eventually, an unlucky Tor client picks and uses the malicious entry node.
5. The attacker performs traffic analysis to compare the signals on each circuit passing through his entry node with the various signals received by the web server. A match reveals the Tor client's identity and its corresponding traffic history during the time it used the malicious exit node.

The entry node only needs to log the traffic pattern that passes through it on each circuit, and the exit node only needs to perform the code injections in the HTTP traffic. Although for clarity we described the attack with multiple machines, the malicious Tor nodes and the web server can all be implemented on the same machine. If the user is browsing the web while using the malicious entry node, the traffic analysis can be difficult because the additional traffic introduces "noise" to the signal. However, if the user stops browsing the web, the channel contains little "noise" and the traffic analysis is easy. A method of a browser based timing attack for simplifying the even if the user does continue browsing the web using TOR. For most traffic analysis attacks, the attacker must control both the exit node and entry node at the same time. For our attack, if a client leaves a browser window open running the JavaScript signal generator, and at any later point chooses a malicious entry node, then the timing attack can reveal his identity. Since this only requires the right choice of an entry node, the probability that the client is compromised each time he chooses a new circuit is roughly 1/ ne, where ne is the number of available entry nodes. If the attacker had to get control of both the entry and exit nodes at the same time, the probability would then be

ISSN: 2277-9655
Impact Factor: 1.852

1/ ne*nx, where nx is the number of available exit nodes. The signal generator allows us to decouple the need to control an exit node and an entry node at the same time, decreasing the expected time to compromise the client. As with any such traffic analysis attack, the adversary can further decrease the time the attack takes by increasing the number of malicious Tor entry nodes [10].

Conclusion

A cell-counting-based attack and browser attack against TOR. This two attacks is difficult to detect and able to quickly and accurately confirm the anonymous communication relationship among users on Tor. In cell counting based attacker at the malicious exit onion router slightly manipulates the transmission of cells from a target TCP stream and embeds a secret signal (a series of binary bits) into the cell counter variation of the TCP stream. An accomplice of the attacker at the entry onion router recognizes the embedded signal using our developed recovery algorithms and links the communication relationship among users.

In browser attack exploits the web browser code execution environment to perform end-to-end traffic analysis attacks without requiring the attacker to control either party to the target communication. There are two security problems that our attack exploits: HTTP's vulnerability to man-in-the-middle attacks and web browsers' code execution feature. Tor places the exit node as a man-in-the-middle of clients' communications. Thus, using Tor may actually decrease the anonymity of users by making them vulnerable to man-in-the-middle attacks from adversaries that would otherwise be unable to perform such attacks.

References

[1] *L. Øverlier and P. Syverson. Locating Hidden Servers. In Proceedings of the 2006 IEEE Symposium on Security and Privacy, May 2006.*

[2] *P. Syverson, G. Tsudik, M. Reed, and C. Landwehr. Towards an Analysis of Onion Routing Security. Workshop on Design Issues in Anonymity and Unobservability, July 2000.*

[3] *N. B. Amir Houmansadr and N. Kiyavash,"RAINBOW: A robust and invisible non-blind watermark for network flows," inProc. 16th NDSS, Feb. 2009, pp. 1–13.*

[4] *X. Fu, Z. Ling, J. Luo, W. Yu,W. Jia, and W. Zhao, "One cell is enough to break Tor's anonymity," in Proc. Black Hat DC, Feb. 2009*

[5] *[Online]. Available: http://www.blackhat.com/presentations/bh-dc-09/Fu/BlackHat-DC-09-Fu-reak-Tors-Anonymity.pdf*

[6] *G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: Design of a type III anonymous remailer protocol," in Proc. IEEE S&P, May 2003, pp. 2–15.*

[7] *G. Smillie, Analogue, Digital Communication Techniques. London, U.K.: Butterworth-Heinemann, 1999.*

[8] *R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second generation onion router," in Proc. 13th USENIX Security Symp., Aug. 2004, p. 21.*

[9] *Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in Mix networks," in Proc. PET, May 2004, pp. 735–742.*

[10] *S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of Tor," in Proc. IEEE S&P, May 2006, pp. 183–195*

[11] *X.Wang, S. Chen, and S. Jajodia, Network flow watermarking attack on low latency anonymous communication systems," in Proc. IEEE S&P, May 2007, pp. 116–130.*

[12] *K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Lowresource routing attacks against anonymous systems," in Proc. ACM WPES, Oct. 2007, pp. 11–20.*

[13] *D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. Communications of the ACM 24(2), February 1981.*

[14] *D. Clark. Design Philosophy of the DARPA Internet Protocols. In Proceedings of the ACM Special Interest Group on Data Communications, pages 106–114, August 1988.*

[15] *R. Dingledine, N. Mathewson, and P. Syverson. Tor: The Second-Generation Onion Router. In Proceedings of the 13th USENIX Security Symposium, August 2004.*

[16] *B. N. Levine, M. Reiter, C. Wang, and M. Wright. Timing Attacks in Low-Latency Mix Systems (Extended Abstract), In Proc. Financial Cryptography, pages 251– 265, February 2004.*

[17] *V.Fusenig, E.Staab, U.Sorger, and T.Engel, "Slotted packet counting attacks on anonymity protocols," in Proc. AISC, 2009, pp.53–60.*

[18]W.Yu,X. Fu, S.Graham, D.Xuan, and W.Zhao, “DSSSbased flow marking technique for invisible traceback,”inProc.IEEES&P,May2007, pp.18–32.

[19]X. Wang, D. S. Reeves, S. F. Wu, and J. Yuill, “Sleepy watermark tracing: An active network-based intrusion response framework,” in Proceedings of 16th International Conference on Information Security (IFIP/Sec), June 2001.

[20]S. C. X. Wang and S. Jajodia, “Network flow watermarking attack on low-latency anonymous communication systems,” in Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P), May 2007.

[21]C. G¨ulc¨u and G. Tsudik, “Mixing email with babel,” in Proceedings of the Network and Distributed Security Symposium (NDSS), February 1996.

[22]M. Reiter and A. Rubin, “Crowds: Anonymity for web transactions,” ACM Transactions on Information and System Security, vol. 1, no. 1, 1998

[23]K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Lowresource routing attacks against anonymous systems,” in Proceedings of ACM Workshop on Privacy in the Electronic Society (WPES), October 2007.

[24]N. Mathewson, “Tor directory protocol, version 3,” http://tor.eff.org/svn/trunk/doc/spec/dir-spec.txt, 2008.

[25]G. Danezis and R. Clayton, “Route fingerprinting in anonymous communications,” in Proceedings of the Sixth IEEE International Conference on Peer-to-Peer Computing, September 2006.

[26]M. Wright, M. Adler, B. N. Levine, and C. Shields, “Defending anonymous communication against passive logging attacks,” in Proceedings of the IEEE Symposium on Security and Privacy, May 2003.

[27]K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, “Low resource routing attacks against anonymous systems,” University of Colorado at Boulder, Tech. Rep., August 2007.

[28]K. Harfoush, A. Bestavros, and J. W. Byers, “Measuring bottleneck bandwidth of targeted path segments,” in Proceedings of the IEEE INFOCOM, April 2003.

[29]X. Wang and D. S. Reeves, “Robust correlation of encrypted attack traffic through stepping stones by manipulation of inter-packet delays,” in Proceedings of the 2003 ACM Conference on Computer and Communications Security (CCS), November 2003.

[30]P. Peng, P. Ning, and D. S. Reeves, “On the secrecy of timing-based active watermarking trace-back techniques,” in Proceedings of the IEEE Security and Privacy Symposium (S&P), May 2006.