A New Replay Attack Against Anonymous Communication Networks

Ryan Pries, Wei Yu, Xinwen Fu and Wei Zhao

***Abstract*— Tor is a real-world, circuit-based low-latency anonymous communication network, supporting TCP applications on the Internet. In this paper, we present a new class of attack, the *replay attack*, against Tor. Compared with other existing attacks, the replay attack can confirm communication relationships quickly and accurately and poses a serious threat against Tor. In this attack, a malicious entry onion router duplicates cells of a stream from a sender. The original cell and duplicate cell traverse middle onion routers and arrive at an exit onion router along a circuit. Since Tor uses the counter mode AES (AES-CTR) for encryption of cells, the duplicate cell disrupts the normal counter at middle and exit onion routers and the decryption at the exit onion router incurs cell recognition errors. If an accomplice of the attacker at the entry onion router controls the exit onion router and detects such decryption errors, the communication relationship between the sender and receiver will be discovered. The replay attack can also be used as a denial of service attack. We implement the replay attack on Tor and our experiments validate the feasibility and effectiveness of the attack. We also present guidelines to defending against the replay attack.**

***Index Terms*— Replay Attack, Tor, Anonymity, Mix Networks, AES**

I. Introduction

Concerns about privacy and security have received greater attention with the rapid growth and public acceptance of the Internet and the pervasive deployment of various wireless technologies. Anonymity has become a necessary and legitimate aim in many applications, including anonymous web browsing, location-based services (LBS) and E-voting. In these applications, encryption alone cannot maintain the anonymity required by participants [1], [2], [3].

Since Chaum pioneered the basic idea of anonymous communication systems, referred to as mixes [4], in 1981, researchers have developed various anonymity systems for different applications. Mix techniques can be used for either message-based (high-latency) or flow-based (low-latency) anonymity applications. Email is a typical message-based anonymity application, which has been thoroughly investigated [5], [6]. Research on flow-based anonymity applications has been active in recent years to preserve anonymity in low-latency applications, such as web browsing and peer-to-peer file sharing [7], [8].

Tor [8] is a popular low-latency anonymous communication network, supporting TCP applications on the Internet. At the time that this paper was written, there are 1044 Tor routers (i.e. onion routers) operating around the world, which form an overlay-based mix network[1]. To communicate with an application server, a Tor client selects an *entry* onion router, a middle onion router and an *exit* onion router in the case of default path length of 3. A circuit is built through this chain of 3 onion routers and the client negotiates a session key with each onion router. Then, application data is packed into cells, encrypted in an onion-like way and transmitted through the circuit to the server [8].

There has been extensive research work done on attacks degrading anonymous communication through Tor. Most existing approaches are based on traffic analysis [3], [9], [10], [11], [12], [13], [14]. Specifically, to determine whether Alice is communicating with Bob through Tor, the traffic-analysis attacks measure the similarity between the sender's outbound traffic and the receiver's inbound traffic in order to confirm their communication relationship. However, traffic-analysis attacks may have a high rate of false positives due to various factors such as Internet traffic dynamics and need a series of packets for statistical analysis.

In this paper, we present a new attack against Tor, the *replay attack*, which does not rely on traffic analysis and can confirm the communication relationship on Tor quickly and accurately, posing a serious threat against Tor. In the replay attack, an attacker may control multiple onion routers, similar to other exiting attacks [11], [3]. A malicious entry onion router duplicates cells of a TCP stream from a sender. The original cells and duplicate cells traverse middle onion routers and arrive at an exit onion router along a circuit. Tor uses the counter mode of Advanced Encryption Standard (AES-CTR) for encryption and decryption of cells at onion routers. The duplicate cell will disrupt the normal counter at the middle and exit onion routers and the decryption at the exit onion router will incur cell recognition errors. If an accomplice of the attacker at the entry onion router controls the exit onion router and detects such decryption errors, the communication relationship between the sender and receiver is discovered.

We implement the replay attack on Tor and our experiments validate the feasibility and effectiveness of the replay attack. This replay attack also poses a great threat against the availability of the anonymity service provided by Tor. We give guidelines on countermeasure against the replay attack. The replay attack presented in this paper is one of the first to exploit the Tor implementation.

The remainder of this paper is organized as follows: We introduce the basic operation of Tor in Section II. We present the details of the replay attack including the basic principle,

Xinwen Fu (corresponding author) and Ryan Pries are with the College of Business and Information Systems, Dakota State University, 820 N. Washington Ave. Madison, SD 57042 (Email:{Xinwen.Fu, priesr}@dsu.edu). Wei Yu is with the Department of Computer Science, Texas A&M University, 301 Harvey R. Bright Bldg, College Station, TX 77843 (Email: weiyu@cs.tamu.edu). Wei Zhao is with School of Science, Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY 12180-3590 (Email: zhaow3@rpi.edu).

[1]In this paper, we use *Tor router* and *onion router* interchangeably. For brevity, *router* is used for the same purpose and the meaning should be clear from the context.

algorithms and some discussion in Section III. In Section IV, we present our experiments on Tor to validate our findings. We give guidelines on countermeasures to the replay attack in Section V. We review related work in Section VI and conclude this paper in Section VII.

II. Basic Operation of Tor

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

A. Components of the Tor Network

Tor is a popular overlay network for anonymous communication on the Internet. It is an open source project and provides anonymity service for TCP applications [15]. Figure 1 illustrates the basic entities of Tor [16].

As shown in Figure 1, there are four basic entities:

1) *Alice* (i.e. *Client*). The client runs local software called an *onion proxy* (OP) to anonymize the client data into the Tor network.
2) *Bob* (i.e. *Server*). It runs 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. 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. There are *directory authorities* and *directory caches*. Directory authorities hold authoritative information on onion routers. Directory caches download directories of onion router information from authorities. Clients download the onion router directory from directory caches.

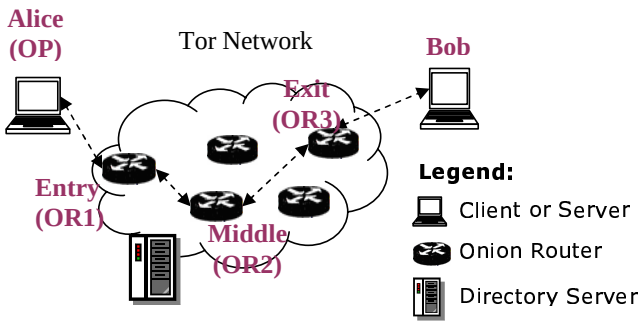
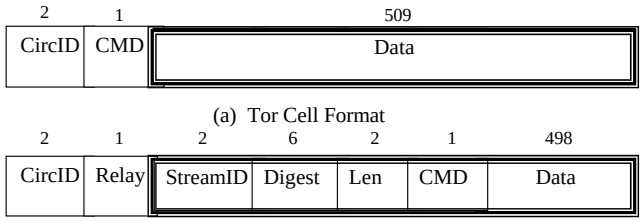


Fig. 1. Tor Network

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

Figure 2 illustrates the Tor cell format. 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 way. There are two types of cells: *control* cell in Figure 2 (a) or *relay* cell in Figure 2 (b). The control cell commands (*CMD*) are: *padding* (used for keepalive and optionally usable for link padding, although not used currently); *create* or *created* (used to set up a new circuit); and *destroy* (to tear down a circuit). The relay cell has an additional header (the relay header). There are numerous types of relay commands that routinely traverse the circuit, such as *relay begin*, *relay data*, *relay end*, *relay sendme*, *relay extend*, *relay drop*, and *relay resolve* (defined in or.h) [16]. We will explain them in later sections when we discuss Tor operations.



(a) Tor Cell Format

(b) Tor Relay Cell Format

Fig. 2. Tor Cell Format [8]

B. Selecting a Path and Creating a Circuit

To anonymously browse 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 [17]. We denote the series of onion routers as the *path* through Tor [18]. The number of onion routers is referred to as the *path length*. We take 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 $OR3$, 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 $OR1$ (referred to as the *entry guard*, which is used to prevent certain profiling attacks) and a middle onion router $OR2$.

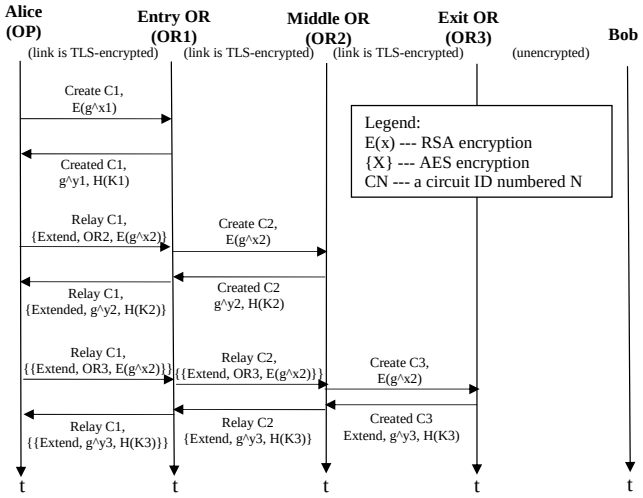


Fig. 3. Tor Circuit Creation [8]

Once the path is chosen, the client initiates the process of creating a circuit over the path incrementally, one hop at a

time. Figure 3 illustrates the procedure of creating a circuit when the path has a default length of 3. Tor uses TLS/SSLv3 for link authentication and encryption. In Figure 3, *OP* first sets up a TLS connection with *OR1* using the TLS protocol. Then tunneled through this connection, OP sends a *create* cell and uses the *Diffie-Hellman* (DH) handshake protocol to negotiate a forward symmetric key, kf_1, and a backward symmetric key, kb_1, with *OR1*, which responds with a *created* cell. Therefore, a one-hop circuit, *C1*, is created.

To extend the circuit one hop further, the *OP* sends to *OR1* a *relay extend* cell, specifying the address of the next onion router, i.e., *OR2* in Figure 3. This *relay extend* cell is encrypted by AES in the counter mode (AES-CTR) with kf_1. Once *OR1* receives this cell, it decrypts the cell and negotiates secret keys with *OR2* using the DH handshake protocol. Therefore, a second segment *C2* of the 2-hop circuit is created. *OR1* sends *OP* a *relay extended* cell, which holds information for *OP* generating the shared secret keys, forward key kf_2 and backward key kb_2, with *OR2*. This *relay extended* cell is encrypted by AES-CTR with key kb_1. *OP* will decrypt the *relay extended* cell and use the information to create the corresponding keys. Encryption of late cells by these secret keys uses AES-CTR too.

Consequently, to extend the circuit to a 3-hop circuit, *OP* sends *OR2* a *relay extend* cell, specifying the address of the third onion router, i.e., *OR3* in Figure 3, through the 2-hop circuit. As we can see, the cell is encrypted in an onion-like way [16]. The payload is first encrypted by kf_1 and then by kf_2. The encrypted cell, like an *onion*, becomes thinner when it traverses an onion router, which removes one layer of onion skin by decrypting the encrypted cell. Therefore, when *OR2* decrypts the cell, it discovers that the cell is meant to create another segment of the circuit to *OR3*. *OR2* negotiates with *OR3* and sends a *relay extended* cell back to *OP*. This cell is first encrypted by kb_2 at *OR2* and then by kb_1 at *OR1*. *OP* decrypts the encrypted backward onion-like cell and derives the shared secret keys with *OR3*, forward key kf_3 and backward key kb_3.

In summary, *OP* negotiates secret keys with the three onion routers one by one and consequently creates a circuit along the path[2]. With the exception that the connection from the exit onion router to the server is not link encrypted, other connections along the path are protected by TLS within Tor. That is, cells encrypted in the onion-like way are also protected by link encryption.

C. Transmitting TCP Streams

Without loss of generality, we will use a short TCP stream, transferring 5 bytes of data “Hello” from Alice (OP) to Bob, as the example to illustrate how a TCP stream is tunneled through the circuit that has already been created. Figure 4 illustrates this simple example. Recall that at this stage, a client’s OP has established secret keys with other onion routers and can encrypt the application payload.

To transmit TCP data to Bob, Alice’s application (such as web browser) contacts the OP, which is implemented as a SOCKS proxy locally. The OP learns the destination IP address and port. OP sends a *relay begin* cell to the exit router *OR3*, and the cell is encrypted $\{\{\{Begin < IP, Port >\}_{kf_3}\}_{kf_2}\}_{kf_1}$, where the subscript refers to the key used for encryption of one onion skin. The three layers of onion skin are removed one by one each time the cell traverses an onion router through the circuit. When *OR3* removes the last onion skin by decryption, it recognizes the request of opening a TCP stream to the *port* at the destination *IP*, which belongs to Bob. Therefore, *OR3* acts as a proxy and builds a TCP connection with Bob and sends a *relay connected* cell back to Alice’s OP. The OP then accepts data from Alice’s application and transmits it to Bob through the circuit. The whole process is transparent to Alice, although she needs to configure her application to use the OP. When Alice’s application has no data to transmit, it will tear down the connection to her OP. As shown in Figure 4, after 5 bytes of data “Hello” in a *relay data* cell is transmitted, Alice’s application tears down the connection to *OP*. The OP then sends a *relay end* cell to *OR3* and *OR3* tears down the connection to Bob.

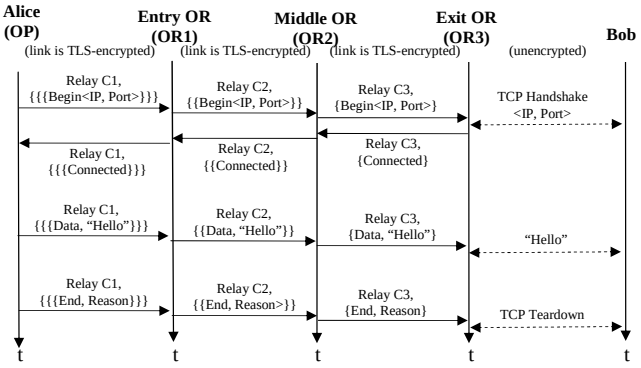


Fig. 4. TCP Connection Creation on Tor

III. Replay Attack

In this section, we first introduce the basic principle of the replay attack and then present the detailed algorithms, followed by discussion.

A. Basic Principle

Figure 4 illustrates the basic principle of replay attack.

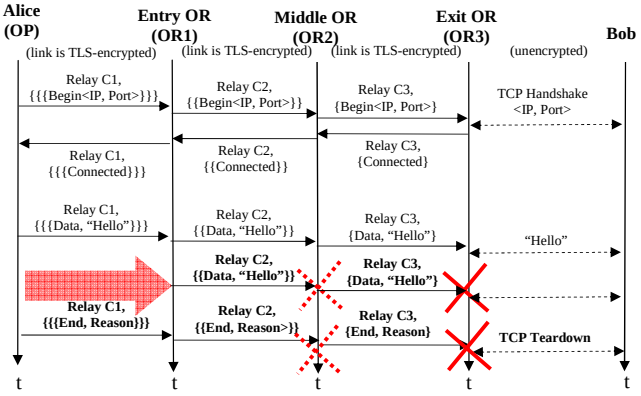


Fig. 5. Replay Attack on Tor

[2]Each onion router checks a flag within the cell to determine whether the cell reaches its end. In this way, the encrypted cell has a fixed size and its length does not swell as in the public key encryption case [4]

Recall that the purpose of this attack is to confirm that Alice is communicating with Bob. Assume that an attacker can gain control of the entry and exit routers used by a given circuit. The attack starts from the malicious entry router. The entry router first attempts to identify a target cell from the TCP stream data on a circuit and duplicate that cell. When the cell is duplicated, the cell's source IP and the time of duplication will be logged. This duplicate cell traverses the circuit and consequently arrives at the exit router. The attacker at the malicious exit router should detect an error caused by this duplicate cell and record the time, the original cell's destination IP address and port. In this way, it is confirmed that the target cell is using the entry router and exit router. Since the entry router knows the sender of the cell is Alice and the exit router knows its receiver is Bob, the communication relationship between the sender and receiver is confirmed.

Now we explain in detail what causes the decryption error. When a data cell is duplicated at *OR1*, its decryption at *OR2* and *OR3* fails because the cell's onion layers are encrypted using AES in the counter mode and the counter is disturbed by the duplicate cell. In the counter mode, encryption and decryption need to keep a synchronized value, *a counter*. The encryption of a cell at the OP increases the AES counter by one. The three routers along the path increase the counter and decrypt the original cell successfully. When *OR1* duplicates a cell, the duplicate cell causes *OR2* and *OR3* to increase the counter and this takes the client and routers out of synchronization. In the current Tor implementation, default actions to this alarm are: *OR3* tears down the circuit and the client builds another circuit for continuous communication.

B. Algorithms

We implement the replay attack based on the Tor release version of 0.1.1.26[3]. To implement the replay attack, we need to modify the source code of the client, entry router and exist router. To make the replay attack successful, selecting the cell to be duplicated at the entry router and confirming the duplicate cell at the exit router are two important issues in the replay attack. We will present the detailed algorithms in the following.

1) Selecting Cells To Be Duplicated: At an entry router, an attacker needs to carefully identify the cell to be duplicated. If a cell is selected during the circuit setup process, the duplicate cell passing through the entry router will cause numerous protocol errors and immediately cause circuits to fail upon their creation. Therefore, the replay attack needs to duplicate cells of TCP stream data. Because cells are encrypted, the only information we know about the cells are the circuit ID and cell's type (control or relay) in the cell header at *OR1*. Obviously, the replay attack shall only work with relay cells. There are various feasible ways of determining the relay cells. From the process of creating a circuit in Figure 3, the attacker can determine when the circuit is built and TCP stream relay starts. Since the default path length is 3, the circuit is built if one *Create* and two *Relay* cells are transmitted on the forward path, and one *Created* cell and two *Relay* cells on the backward path.

[3]Newer versions of Tor have not changed the algorithms involved in this paper.

To detect cells containing TCP data, an attacker can observe if a connection is built from the exit Tor router to a server. Recall that an exit Tor router acts as a normal proxy and creates a TCP connection to the server. A connection setup between the exit to a server indicates that some client has started a TCP session communicating with the server. The connection teardown between the exit to the server indicates that the client stops the TCP session. The malicious exit router can detect such behavior and buffer cells containing TCP data for later replay.

The attacker can even use a loop-control style to detect the status of the TCP stream. The attacker at the exit router has the full information of the target TCP connection and notifies the attacker at the entry router *OR1*. The attacker at *OR1* can then choose which cells to duplicate. Since the attacker at the exit router knows everything about the TCP connection, it can also duplicate the response cells back to the client and the attacker at *OR1* can detect the duplicate cells.

The attacker can choose an appropriate time for replaying cells. Recall that once the replay happens, the corresponding circuit cannot be used any more because the AES counter is disturbed along the path. If the attacker replays the cells when a TCP connection is still running, the circuit will be torn down and the client will use other circuits to continue the TCP connection. The attacker may also replay the cells when the circuit is not occupied by any TCP connection and before the circuit is torn down normally. Such an attack will not degrade the TCP performance very much and can be stealthy.

2) Confirming Duplicate Cells: Recall that when the cell duplication is applied at a malicious entry router, decryption errors will happen at the exit onion router if the TCP stream is using that circuit. The exit router records the destination IP address, port number and time stamp, and the entry router records the time stamp of duplicate cells. We use Network Time Protocol (NTP) to synchronize the time of the entry router and exit router. By correlating the time of sending the duplicate cell with time of the decryption error ocurring, we can confirm that the decryption is actually caused by cell duplication. In addition, since the entry router knows the sender of the TCP stream and the exit router knows the receiver of the TCP stream, the communication relationship between the sender and receiver is confirmed.

C. Discussion

As discussed above, the replay attack enables fast and accurate confirmation of the communication anonymized by Tor, posing great threats to the anonymity service. We now discuss it from various perspectives, such as its malicious impact, the issue of controlling Tor routers, and noise reduction.

1) Broad Impacts: The replay attack is a very powerful one. As we can see, to identify the communication relationship of a TCP stream, the attack is only based on one single duplicate cell. The determination of duplicate cells is quite simple and can be carried out quickly and accurately. This makes the replay attack quite different from other existing traffic-analysis attacks, which requires tuning parameters and balancing the trade-off between the false positive and detect rate [19], [10], [3], [11], [13], [20]. The replay attack is also robust to the

network size, traffic dynamics, and other anti-traffic analysis strategies such as batching schemes [2], [21].

The replay attack can be malicious in various ways against the privacy service provided by Tor. First, it can be used to randomly profile both clients and servers hidden by Tor. For any client who uses a circuit across the entry router and exit router, the attacker can discover their Internet behavior and compromise their privacy. Second, the replay attack can also be used to launch additional attacks, such as denial-of-service. If the malicious entry router generates duplicate cells, it will cause corresponding exit routers to tear down connections. This may slow down Tor if there exist multiple malicious entry routers which keep sending such replay cells. In addition, Tor directory authorities monitor the activities of routers and may blacklist those routers which unexpectedly drop connections. Although those malicious entry routers are the root cause of the connection drop, those innocent exit routers become scapegoats. Due to the anonymity maintained by Tor, it is not trivial to trace back those malicious entry routers.

2) Controlling Onion Routers: We have assumed that the attacker controls some entry and exit routers. This is a reasonable assumption due to the principle of Tor design, e.g., voluntary-oriented operation. Anyone can set up entry routers and exit onion routers. As long as a router has a self-designed exit policy enabling access to external services, this router becomes an exit router. To become an entry router, a Tor router must meet some criteria. If a router has a mean time between failure (MTBF) not less than the median for active routers or at least 10 days, it becomes a *stable* router. A stable router can be promoted to an entry guard if its bandwidth is either at least the median among known active routers or at least 250KB/s [17]. This set of criteria are not difficult to meet in the real-world. Experiments in Section IV-B will confirm this claim.

The replay attack can be more flexible. The requirement of a malicious exit router is not necessary in the replay attack if an attacker can monitor outbound stream from an exit router. This kind of traffic monitoring capacity has been widely used by other exiting attacks [19], [10], [3], [11], [13], [20]. To this end, using various network traffic monitoring tools, the attacker can record the destination address and port of outbound TCP streams from an exit router. If the duplicate cell is identified at an entry onion router and a TCP stream from this exit router aborts abruptly, this gives a positive sign that the TCP stream from the sender travels along that exit router, addressed to the corresponding receiver.

3) Noise Reduction: We will now discuss the noise reduction related to the replay attack. The false positive of this attack comes from unexpected decryption errors caused by non-duplication events. Based on our month long experiments on exit routers in Tor, we have not recorded such unexpected errors. This confirms that the false positive rate of the replay attack against Tor is very low.

In order to further decrease the false positive rate, the attacker may replay multiple buffered cells from a single TCP stream at the same time. For each duplicate cell, we assume that the detection rate and false positive rate of the replay attack is p_d and p_f, respectively. We now derive the detection rate P_D and false positive rate P_F for replaying n duplicate cells. When n decryption errors is detection at the exit Tor router, the probability that all errors are not caused by the duplicate cells is $(1-p_d)^n$, and the detection rate $P_D = 1-(1-p_d)^n$. The corresponding false positive rate is $P_F = p_f^n$. Therefore, by choosing an appropriate n, an attacker can achieve a high detection rate and small false positive rate.

IV. Evaluation

In this section, we use a real-world implementation to investigate the effectiveness and feasibility of the replay attack.

A. Experiment Setup

We carry out the experiment on Tor [22]. Figure 6 shows the experimental setup. We use two malicious onion routers as the Tor entry onion router and exit onion router. The entry router, client (Alice) and server (Bob) is located in an office on campus[4]. The exit onion router is located in an off-campus location. Machines on campus and off-campus are on different public IP segments connecting to different ISPs.

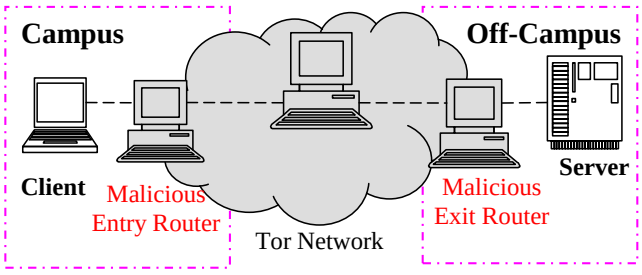


Fig. 6. Experiment Setup

To reduce the impact of the replay attack on Tor's normal behavior, we conduct experiments in a partially controlled environment. We change segments of Tor client code for debugging purposes. The Tor client would only build circuits through the designated malicious exit onion router and the designated entry onion router in Figure 6. The middle onion router is selected using the normal Tor routing selection algorithm. Recall that the goal of the replay attack is to confirm that the client is communicating to the server. We create a small client and server application to send and receive data through TCP. The test server binds to port 41 and receives packets and displays relevant connection information to the screen for debugging and measurement. The Tor client utilizes *tsocks* [23] to automatically transport its outbound TCP stream through the OP using SOCKS. By using the Tor configuration file and manipulating parameters, such as *EntryNodes*, *ExitNodes*, *StrictEntryNodes*, and *StrictExitNodes* [18], we setup the client to select the malicious onion routers along the circuit. The exit onion router uses the default exit policy from Tor and the entry onion router's exit policy only allows it to be used as an entry or middle onion router.

B. Experimental Results

The publicly available bandwidth information of onion routers from the Tor directory servers confirm that the set of

[4]The office is located at Dakota State University campus

criteria for becoming an entry onion router are not difficult to meet. Figure 7 shows the onion router's bandwidth distribution on Tor based on directory information collected 10:30am on August 18, 2007. The mean value of the bandwidth is only around 57KB/s. After running for about 5 days, our entry router with a bandwidth of 200KB/s became an entry guard.

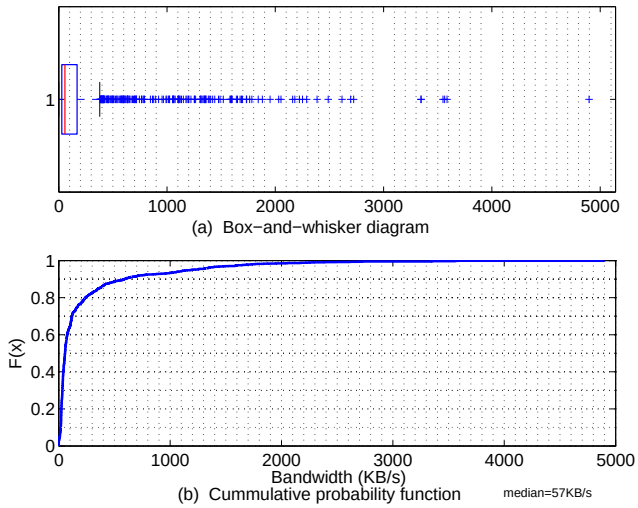


Fig. 7. Onion Routers' Bandwidth Distribution on Tor: Bandwidth Median=57KB/s; (a) Box and Whisker Plot of Bandwidth; (b) Cumulative Distribution Function of Bandwidth

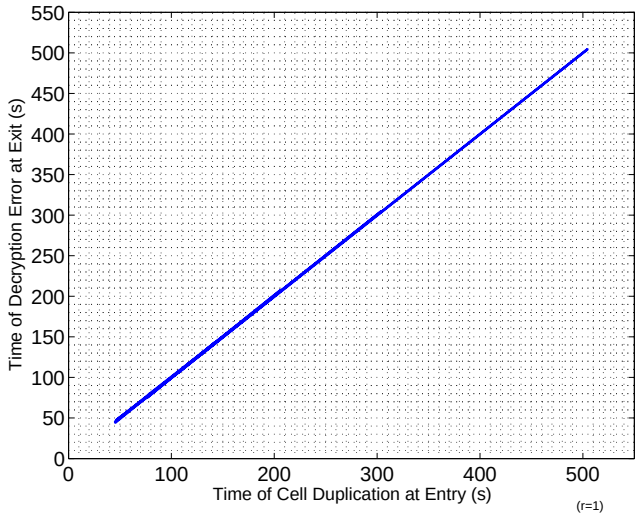


Fig. 8. Correlation Between Time of Duplicating Cells and Time of Detecting Decryption Error

In our experiments, the client sends a message packed in one cell to the server every 10 seconds. The revised code for the entry onion router records the time of duplicating the data cell. and the revised code for the exit onion router records the time of decryption error and carry out the detection. We use the *correlation coefficient* r to measure the strength of correlation between the time of generating duplicate cells and the time of observing decryption errors. Correlation coefficient is defined in

$$r = \frac{\sum_{x,y}(x-\bar{x})(y-\bar{y})}{\sqrt{\sum_{x}(x-\bar{x})^2}\sqrt{\sum_{y}(y-\bar{y})^2}}, \quad (1)$$

where x is the time of cells duplicated at the entry onion router and y is the time of decryption errors incurred at the malicious exit. $\bar{x}$ and $\bar{y}$ are the mean values of x and y, respectively.

Figure 8 shows relationship between the time of duplicating data cells and the time of decryption errors. As we can see from this figure, there is a perfect positive match, since the actual correlation coefficient between them is *one*. This strongly confirms that the replay attack can indeed confirm the communication relationship if the sender and receiver are using Tor. In addition to its accuracy, the replay attack can be very fast, since it only needs to launch and recognize a single duplicate cell.

V. Guideline of Countermeasures

We have demonstrated the threat of the replay attack against Tor. We now discuss possible countermeasures to this attack.

1) Minimizing Number of Compromised Entry Routers: Recall that the replay attack requires an attacker to fully control at least one entry onion router, which may advertise the false bandwidth resource and promote compromised servers to be entry onion routers of Tor. There are two possible ways to minimize the chance that compromised servers become entry onion routers. First, the path selection algorithm may be evolved and select only fully trusted and dedicated ones through strict authentication and authorization. Second, countermeasures may be developed to detect false bandwidth advertisements from an attacker's Tor router trying to become Tor entry guards (Sybil attacks) [24]. For example, Tor's path selection protocol can be augmented to allow onion routers to proactively monitor each other and validate other onion routers' bandwidth [25]. A reputation system could also help facilitate this countermeasure. In this way, the attacker will have less of a chance to control the entry onion router and the effectiveness of replay attack will be reduced. However, this approach cannot completely address the replay attack, since the attacker can still contribute high bandwidth if it has the resources.

2) Monitoring Duplicate Cells. Recall that the replay attack needs to send the duplicate cells. If duplicate cells can be detected and drop at the middle router before they reach the exit onion router, the effectiveness of the attack will be significantly reduced. To this end, one naive way is to allow the middle onion router along the circuit to detect duplicate cells by buffering historical cells. However, this will give more overhead to onion routers, since Tor requires using a pair of memory buffers for reading and writing data from each TCP stream [26].

Another way to detect the replay attack is to have the Tor clients and exit routers monitor the connection for abnormality. Since the replay attack will cause connections to be torn down and a client to switch to another circuit, a frequent such connection teardown and circuit switch may indicate the possibility of the replay attack. But the client cannot rely on the reported reason of circuit teardown for this monitoring

purpose since the malicious exit node may change the reason code. When a replay attack is used to confirm the communication relationship which does not exist, exit routers other than the malicious one will receive the duplicate cells and detect decryption errors. Such decryption errors may indicate a high possibility of the replay attack.

VI. Related Work

Most attacks against low-latency flow-based anonymous communication networks proposed in the literature belong to the traffic-analysis attack. The basis of this class of attack is to determine whether Alice is communicating with Bob through an anonymous communication network based on measuring the similarity between Alice's outbound traffic and Bob's inbound traffic. For example, Zhu *et al*. [10] proposed the scheme of using mutual information for the traffic similarity measurement. Levine *et al*. [9] utilized a cross-correlation technique for the traffic similarity measurement. Murdoch *et al*. [11] investigated the traffic timing similarity on Tor by using some compromised Tor servers. Overlier *et al*. [3] studied an attack that uses one compromised mix node to identify the "hidden server" anonymized by Tor. Their approach is also based on the traffic timing similarity to associate circuits, and thereby locate the hidden servers. Bauer *et al*. [12] studied an attack that is based on some malicious onion routers and they use traffic analysis techniques. There are other attacks that intend to achieve both accuracy and invisibility [14].

There is little research efforts that have been conducted on the non-traffic analysis based attacks. Murdoch [27] investigated an attack to reveal hidden servers of Tor by exploiting the fact that the clock deviations of a target server should be consistent with the server's load. Differently, the replay attack studied in this paper exploits the fundamental design of Tor. This attack is powerful, capable of launching and detecting one single duplicate cell.

VII. Conclusion

In this paper, we introduced a new class of replay attack against Tor, and the attack can achieve fast and accurate confirmation of the anonymized communication relationship. In particular, an attacker at a malicious entry onion router duplicates data cells from the inbound TCP stream. The duplicate data cell will be carried along the circuit of Tor and causes decryption errors at a malicious exit onion router. Based on the high correlation between the time of duplicating cells at the entry onion router and receiving a decryption error at the exit onion router, the attacker can confirm the communication relationship between the sender and receiver accurately and quickly, significantly degrading the anonymity service provided by Tor. Via real-world implementation and experiments, effectiveness and feasibility of the replay attack are validated. The replay attack can also be used to blacklist Tor routers as a denial of service attack. Our study is critical for securing and improving Tor.

Due to the Tor's fundamental design, defending against the replay attack remain challenging tasks and we will investigate them in our future research. Also, we believe that other vulnerabilities exist in Tor and we plan to conduct a thorough investigation of them and develop corresponding countermeasures.

References

[1] Q. X. Sun, D. R. Simon, Y. Wang, W. Russell, V. N. Padmanabhan, and L. L. Qiu, "Statistical identification of encrypted web browsing traffic," in *Proceedings of IEEE Symposium on Security and Privacy (S&P)*, May 2002.

[2] X. Fu, Y. Zhu, B. Graham, R. Bettati, and W. Zhao, "On flow marking attacks in wireless anonymous communication networks," in *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*, April 2005.

[3] L. Overlier and P. Syverson, "Locating hidden servers," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[4] D. Chaum, "Untraceable electronic mail, return addresses, and digital pseudonyms," *Communications of the ACM*, vol. 4, no. 2, February 1981.

[5] G. Danezis, R. Dingledine, and N. Mathewson, "Mixminion: design of a type iii anonymous remailer protocol," in *Proceedings of the 2003 IEEE Symposium on Security and Privacy (S&P)*, May 2003.

[6] C. Gülcü and G. Tsudik, "Mixing email with babel," in *Proceedings of the Network and Distributed Security Symposium (NDSS)*, February 1996.

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

[8] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," in *Proceedings of the 13th USENIX Security Symposium*, August 2004.

[9] B. N. Levine, M. K. Reiter, C. Wang, and M. Wright, "Timing attacks in low-latency mix-based systems," in *Proceedings of Financial Cryptography (FC)*, February 2004.

[10] Y. Zhu, X. Fu, B. Graham, R. Bettati, and W. Zhao, "On flow correlation attacks and countermeasures in mix networks," in *Proceedings of Workshop on Privacy Enhancing Technologies (PET)*, May 2004.

[11] S. J. Murdoch and G. Danezis, "Low-cost traffic analysis of tor," in *Proceedings of the IEEE Security and Privacy Symposium (S&P)*, May 2006.

[12] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against anonymous systems," in *Proceedings of ACM Workshop on Privacy in the Electronic Society (WPES)*, October 2007.

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

[14] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "Dsss-based flow marking technique for invisible traceback," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, May 2007.

[15] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: anonymity online," http://tor.eff.org/index.html.en, 2007.

[16] R. Dingledine and N. Mathewson, "Tor protocol specification," http://tor.eff.org/svn/trunk/doc/spec/tor-spec.txt, 2007.

[17] N. Mathewson, "Tor directory protocol, version 3," http://tor.eff.org/svn/trunk/doc/spec/dir-spec.txt, 2007.

[18] R. Dingledine and N. Mathewson, "Tor path specification," http://tor.eff.org/svn/trunk/doc/spec/path-spec.txt, 2007.

[19] 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.

[20] W. Yu, X. Fu, S. Graham, D. Xuan, and W. Zhao, "Dsss-based flow marking technique for invisible traceback," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy (S&P)*, 2007 May.

[21] A. Serjantov, R. Dingledine, and P. Syverson, "From a trickle to a flood: active attacks on several mix types," in *Proceedings of Information Hiding Workshop (IH)*, February 2002.

[22] "Tor: anonymity online," http://tor.eff.org/, 2007.

[23] "A transparent socks proxying library," http://tsocks.sourceforge.net, 2007.

[24] 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.

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

[26] "Theonionrouter/torfaq," http://wiki.noreply.org/noreply/TheOnionRouter/TorFAQ, 2007.

[27] S. J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of the 13th ACM Conference on Computer and Communications Security (CCS)*, November 2006.