LETTER

Transparent Discovery of Hidden Service

Rui WANG[†], ***Student Member***, **Qiaoyan WEN**[†], ***Nonmember***, **Hua ZHANG**[†a)], ***Member***, **Sujuan QIN**[†], ***and*** **Wenmin LI**[†], ***Nonmembers***

SUMMARY Tor's hidden services provide both sender privacy and recipient privacy to users. A hot topic in security of Tor is how to deanonymize its hidden services. Existing works proved that the recipient privacy could be revealed, namely a hidden server's real IP address could be located. However, the hidden service's circuit is bi-directionally anonymous, and the sender privacy can also be revealed. In this letter, we propose a novel approach that can transparently discover the client of the hidden service. Based on extensive analysis on the hidden service protocol, we find a combination of cells which can be used to generate a special traffic feature with the cell-padding mechanism of Tor. A user can implement some onion routers in Tor networks and monitor traffic passing through them. Once the traffic feature is discovered, the user confirms one of the controlled routers is chosen as the entry router, and the adjacent node is the client. Compared with the existing works, our approach does not disturb the normal communication of the hidden service. Simulations have demonstrated the effectiveness of our method.

key words: *Tor, hidden service, anonymous, privacy, transparent*

1. Introduction

The Onion Router (Tor), with about three million users, is currently the most popular low-latency anonymous communication network [1]. It was first deployed in 2003, providing sender privacy to Internet users. Nonetheless, recipient privacy is equally required, which assists users not only to access information anonymously but also to publish information anonymously. Hidden service was released and deployed on Tor in 2004, allowing users to provide their services anonymously. At the time of writing, there are 6708 onion routers and 7608 descriptors within Tor network around the world.

Hidden service aims to protect users' privacy and put a constraint on malicious behavior performed by adversaries. However, some vulnerabilities can be utilized by adversaries. Extensive works have been conducted to degrade the anonymity of the hidden services which can be categorized into two types: passive analysis [3]–[8] and active analysis [2]. Overliver et al. proposed the first passive method based on traffic analysis to identify the hidden service [3], which targeted a previous version without entry guards [9]. Zhang et al. proposed an application-level method against hidden service [4]. They evaluated the time correlation between the web accessing and the generated traffic in the malicious onion router. Biryukov et al. exposed flaws both in the design and implementation of hidden service, allowing an attacker to measure the popularity of arbitrary hidden service, take down hidden services and deanonymize hidden services [5]. Ling et al. researched and deployed a hidden server discovery system [6]. They controlled the entry router and rendezvous point, using the timing analysis to identify the hidden server. Wang et al. presented a protocol-feature-based attack [7]. The attacker modified one bit of a cell at the entry router which caused decryption error at the hidden server and created an identifiable feature, and then the attacker could confirm whether its controlled router was chosen as the entry router for locating the hidden service. These passive approaches have a common feature in that the attackers need to control one side of the communication with a hidden service and intentionally send the *DESTROY* cell to make an identified signature, and with the development of the entry guards, passive attacks may suffer a high rate of false positives due to various factors. Murdoch presented the first active approach [2]. A Hidden server's temperature would vary while its workload changes, so he identified the hidden server by estimating its temperature through measuring the clock skew.

Existing works mainly focus on revealing the hidden server. In this letter, we propose a transparent discovery approach for locating the clients of hidden services according to the analysis on the hidden service protocol. First, we assume we can control the entry router connecting with the client, and this falls within the Tor threat model [1]. Second, we use the cell-padding mechanism of Tor to build a special combination of cells [10], denoted as a traffic feature. The controlled router monitors the traffic. If the traffic satisfies two key conditions, we decide that our controlled router is chosen as the entry router and mark the adjacent node in the circuit as the client. Finally, we implement the approach and make the evaluation. The obtained results demonstrate the feasibility and the validity of our approach. We aim to research and expose flaws of the hidden service which may pose a threat against security of Tor. Both owners of hidden services and third party attackers can use this approach to deanonymize hidden services and threaten the normal users' privacy, so we also give guidelines on countermeasures. Our work also has other significative functions. For example, the hidden service can be misused for providing malicious

Manuscript received May 9, 2016.
Manuscript revised July 1, 2016.
Manuscript publicized August 8, 2016.

[†]The authors are with State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, 100876 China.
a) E-mail: zhanghua_288@bupt.edu.cn

DOI: 10.1587/transinf.2016EDL8100

contents such as drug trade and child pornography. Our approach can help internet service providers or network administrators to investigate clients who are accessing these malicious services.

2. Transparent Discovery

In this section, we describe our approach. Without loss of generality, we assume that we can control the entry router adjacent to the client. This is reasonable because onion routers of Tor network are set up in a voluntary manner [1], the same assumption was also made in attacks [3]–[8] against Tor network. Note that we manipulate a hidden server and intend to find out who accesses our hidden service.

2.1 Transparent

All Tor traffic is packed into fixed-size (512 bytes) cells, which classified into two types: control cell and relay cell [1]. *PADDING* cell is one kind of control cell currently used to implement connection keeping alive [10]. If there is no other traffic, we find that Tor nodes may send each other a *PADDING* cell every few minutes to maintain the circuits. Based on this mechanism, the *PADDING* cells sent by the *introduction point* (InP) will not be noticed by the routers along the circuit and discarded at the recipient side, and the hidden service still work normally.

2.2 Introduction Circuit Creation Procedure

Figure 1 shows the creation procedure of the introduction circuit [1]. The entry has to forward ($1 \times$ *CREATE* + $2 \times$ *RELAY_COMMAND_EXTEND* + $1 \times$ *INTRODUCE1*) cells up to the InP and ($1 \times$ *CREATED_FAST* + $3 \times$ *RELAY_COMMAND_EXTENDED* + $1 \times$ *ACK*) cells down to the client. The special combination of cells composes a cell-count feature.

After the introduction circuit has been created, (1) the client assembles an introduce message encrypted by the hidden service's public key including the address of the *rendezvous point* (RP), and sends a *RELAY_COMMAND_INTRODUCE1* cell to the InP containing the introduce message. (2) Upon receiving this cell, the InP repacks the cell into a new *RELAY_COMMAND_INTRODUCE2* cell and sends it along the corresponding circuit to the hidden server (If *PK_ID* is unrecognized, the *RELAY_COMMAND_INTRODUCE1* cell will be discarded, then the steps 1 and 2 repeated). (3) The InP replies a *RELAY_COMMAND_INTRODUCE_ACK* cell to the client. (4) Upon receiving this ACK cell, the client tears down the circuit to the InP. Then, the client waits to start communicating with the hidden server. The steps are shown in Fig. 2.

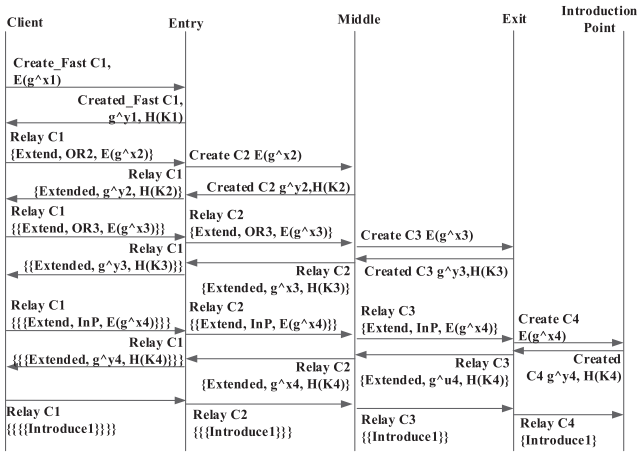

Fig. 1 Client building a three-hop circuit to the introduction point.

2.3 Detailed Discovery Process

In our approach, upon receiving the *RELAY_COMMAND_INTRODUCE1* cell, the InP needs to generate traffic with a special feature which can help to confirm that our controlled router is chosen as the entry router. The InP is the exit router of the hidden server, so a hidden service can modify its config file to specify some InPs. A third party or an attacher can publish or push phishing onion addresses (e.g., silkroadxxxxxxxx.onion) to users and redirect the users to their controlled hidden service [5]–[7]. An attacker can also control the InP by another way: because onion routers of Tor are set up in a voluntary manner [10]. By implementing high bandwidth routers in Tor network, the attacker has chance to control the entry router or exit router of a circuit. For example, by running 23 onion routers with high bandwidth into Tor network, the chance that any of these routers to be chosen as the entry is about 13.8% [7]; Ling et al. made a stronger assumption that an attacker controlled both the entry router and the exit router in a Tor circuit, and the real world catch probability can confirm over 60% by injecting around 4% of onion routers with long uptime and high bandwidth [11].

Based on Tor's *PADDING* cell mechanism [1], we can manipulate a combination of cells which will not noticed by the routers along the circuit and discarded at the recipient side. Steps of our approach are shown in Fig. 3 and described as follows:

- A user builds circuit to one of the indicating InP.
- Upon receiving the *RELAY_COMMAND_INTRODUC-*

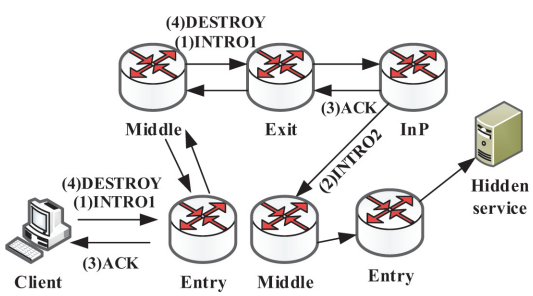

Fig. 2 Client initialing the hidden service.

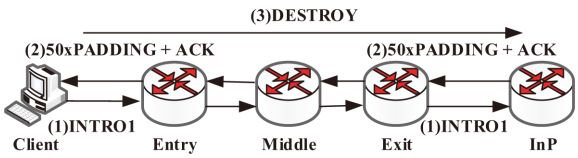

Fig. 3 Revealing the entry router.

E1 cell, the InP sends 50 *PADDING* cells back to the client along the circuit which will not noticed by the routers along the circuit and then silently dropped by the client. Then, the InP replies the *RELAY_COMMAND_INTRODUCE_ACK* cell to the client.

- The client sends a *CELL_DESTROY* cell to the InP to close the circuit.

In this process, our controlled router monitors the traffic of the circuits traveling through it. When the controlled router captures a *DESTROY* cell over a circuit, it will check the following two key conditions:

- Whether the *DESTROY* cell was received just after the *RELAY_COMMAND_INTRODUCE_ACK* cell.
- Whether the number of the transmitted cells match the cell-count feature: 4 cells up and 55 ($1 \times CONTROL + 3 \times RELAY + 1 \times ACK + 50 \times PADDING$) cells down the circuit. Based on the traffic feature, we can distinguish the case when the controlled router was chosen as the entry from the case when it was chosen as the middle.

If the two key conditions are satisfied, we decide that our controlled router was chosen as the client's first hop entry router. Due to the client directly connects with the entry node, and then we can locate the client's real IP address.

If a client closed the circuit after receiving n PADDING cells (for instance, $n = 10$), we can still locate this client. Because the client need to send *DESTROY* cell to close the circuit, and time consumed by transmitting empty load PADDING cells is very short, and then considering about the network environment factors, we may obtain another cell-count-feature which can be used to locate client:

- First, before the client disconnected the circuit after receiving 10 PADDING cells, our controlled entry router has completely transmitted 50 PADDING cells, so our approach is still effective.
- Second, our controlled entry router did not finished sending 50 PADDING cells to the client before the client closed the circuit. However, when the client closed the circuit, he also had to send the *Destroy* cell. Based on our statistics, a PADDING cell is sent every 4 minutes 50 seconds, 10 PADDING (or less than 10) cells can be seen as a cell-count-feature between client and entry router and are enough to locate the client.
- Third, the entry router can be set to transmit a PADDING cell every 4 minutes 50 seconds, and redundant PADDING cells in the same cycle will be discarded.

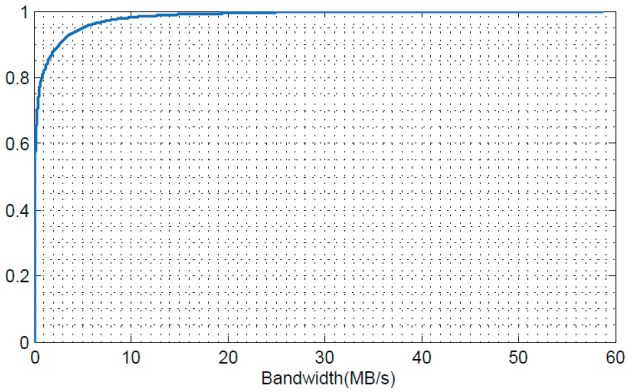

Fig. 4 Empirical CDF of bandwidth of all routers in the Tor network.

2.4 Countermeasures

First, protecting the entry guard router from abused by setting the guard rotation interval longer and taking into account how long the onion routers has been online when assigning entry flags to them. Note that if this countermeasure does not carefully implemented, it will cause a number of downsides like reduced end-user quality of experience and malicious routers accumulating.

Second, users of hidden service had better go to reliable web-sites to acquire safety onion addresses of hidden services.

3. Evaluation

According to the bandwidth information collected from the directory server on May 20, 2015, there are 6709 active onion routers on Tor, including 1572 guard routers, 1073 exit routers and 346 double routers. Figure 4 shows the bandwidth distribution of onion routers. The mean value of the bandwidth is only around 1MB/s. In addition, Tor adopts a bandwidth weighted node selection algorithm in order to enhance its performance [1], and the key factors for choosing the entry routers are high bandwidth and long online time. Based on this algorithm, if we manipulate some nodes with bandwidth of 20 Mbytes/s, the relationship between the catch probability and the number of routers is shown in Fig. 5. Note that the entry guard weight is $max(0, 1 - \frac{B_{total}+k \cdot b}{3(B_{exit}+B_{double})})$ where k denotes the total number of manipulated nodes, b denotes a manipulated router's bandwidth and B_{double} denotes the bandwidth of routers which get both entry-flag and exit-flag.

We manipulated an onion router into Tor network, which runs the Tor stable version 0.2.5.12 on an Ubuntu server version 12.04.4 LTS with a bandwidth of 300Mbit/s located in Florida USA. In order to estimate the reliability of the traffic feature, we used *tcpdump* to collect statics on the number of forwarded cells per circuit into a record file and analyzed the file with *wireshark*. We examined 1501 circuits through the entry node, none of the circuit exhibited

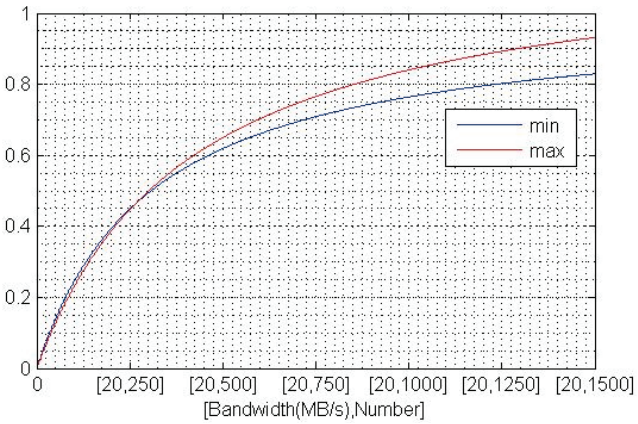

Fig. 5 Probability that at least a circuit traverses through the controlled entry.

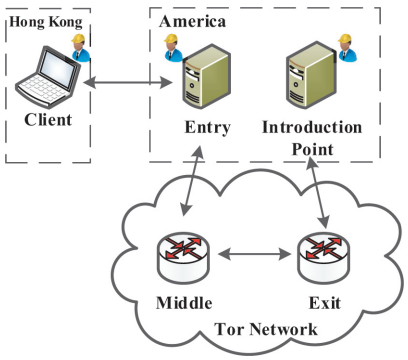

Fig. 6 Experiment setup.

the traffic pattern of 4 cells up the circuit and 55 cells down the circuit. We also implemented an experiment to count the number of *PADDING* cells of an *onion router* (OR). We ran an *onion proxy* (OP) and monitored the traffic 24 hours, and we did not transfer any data through this OP. From the experiment result, we found that the OP exchanged a *PADDING* cell by about every 4 minutes 50 seconds, and totally got 312 *PADDING* cells during the experiment period. This result demonstrate the cell-count feature of 50 *PADDING* cells in our method is unique and effective.

Figure 6 illustrates our approach for locating the client. We modified the general options *EntryNodes* and *ExitNodes* in the *torrc* file (Tor's config file) of the client and the hidden server respectively, so the client chose one of our controlled routers as the entry node and the hidden server chose a specified router as InP. We run the experiment 500 times for deriving the true positive rate. For each *RELAY_COMMAND_INTRODUCE_ACK* cell we successfully collected the corresponding cells matching the traffic feature and got no false positives.

4. Conclusion

In this letter, we have presented a discovery approach for locating the clients of hidden services. Of particular importance, it can transparently find the real IP addresses and does not disturb the normal communication of the hidden service. Simulations have demonstrated the effectiveness of the proposed method.

Acknowledgements

The authors would like to thank the reviewers for their valuable and constructive comments, which helped to improve the presentation of the Letter. This work is supported by NSFC (Grant Nos. 61300181, 61502044), the Fundamental Research Funds for the Central Universities (Grant No. 2015RC23).

References

[1] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The second-generation onion router," Proc. USENIX Security Sysposium, pp.303–320, San Diego, CA, USA, 2004.
[2] S.J. Murdoch, "Hot or not: Revealing hidden services by their clock skew," Proc. ACM Conf. on Comput. Commun. Security (CCS), pp.27–36, Alexandria, VA, USA, Nov. 2006.
[3] L. Overlier and P. Syverson, "Locating hidden servers," Proc. IEEE Security and Privacy Symposium (S&P), pp.98–114, Berkeley, CA, USA, May 2006.
[4] L. Zhang, J.Z. Luo, M. Yang, and G. He, "Application-level attack against Tor's hidden service," Proc. IEEE Int. Conf. on Pervasive Computing and Applications (ICPCA), pp.509–516, Port Elizabeth, South Africa, Oct. 2011.
[5] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for Tor hidden services: Detection, measurement, deanonymization," Proc. IEEE Security and Privacy Symposium (S&P), pp.80–94, San Francisco, CA, USA, May 2013.
[6] Z. Ling, J. Luo, K. Wu, and X.Fu, "Protocol-level hidden server discovery," Proc. IEEE Int. Conf. on Comput. Commun. (INFOCOM), pp.1043–1051, Turin, Italy, April 2013.
[7] R. Wang, Q. Wen, H. Zhang, and X. Li, "A novel protocol-feature attack against Tor's hidden service," IEICE Trans. Inf. & Syst., vol.E99-D, no.4, pp.839–849, April 2016.
[8] Z. Ling, J. Luo, K. Wu, W. Yu, and X. Fu, "TorWard: Discovery of malicious traffic over Tor," Proc. IEEE Int. Conf. on Comput. Commun. (INFOCOM), pp.1402–1410, Toronto, Canada, April 2014.
[9] T. Elahi, K. Bauer, M. AlSabah, R. Dingledine, and I. Goldberg, "Changing of the guards: A framework for understanding and improving entry guard selection in Tor," Proc. ACM Workshop on Privacy in Electronic Society, pp.43–54, Raleigh, NC, USA, Oct. 2012.
[10] The Tor Project, Inc., "Tor Protocol Specification," https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt, accessed 2016.
[11] X. Fu, Z. Ling,W. Yu, et al., "Network forensics through cloud computing," Proc. Int. Workshop on Security and Privacy in Cloud Computing, pp.26–31, June 2010.