Large-scale Evaluation of Malicious Tor Hidden Service Directory Discovery

Chunmian Wang[†], Zhen Ling[†]*, Wenjia Wu[†], Qi Chen[†], Ming Yang[†], Xinwen Fu[‡]
[†]School of Computer Science and Engineering, Southeast University, China
Email: {chunmianwang, zhenling, wjwu, qichen, yangming2002}@seu.edu.cn
[‡]Department of Computer Science, University of Massachusetts Lowell, Lowell, MA, USA
Email: xinwen_fu@uml.edu

***Abstract*—Tor is the largest anonymous communication system, providing anonymous communication services to approximately 2.8 million users and 170,000 hidden services per day. The Tor hidden service mechanism can protect a server from exposing its real identity during the communication. However, due to a design flaw of the Tor hidden service mechanism, adversaries can deploy malicious Tor hidden service directories (HSDirs) to covertly collect all onion addresses of hidden services and further probe the hidden services. To mitigate this issue, we design customized honeypot hidden services based on one-to-one and many-to-one HSDir monitoring approaches to luring and identifying the malicious HSDirs conducting the rapid and delayed probing attacks, respectively. By analyzing the probing behaviors and payloads, we investigate a novel semantic-based probing pattern clustering approach to classify the adversaries so as to shed light on the purposes of the malicious HSDirs. Moreover, we perform theoretical analysis of the capability and accuracy of our approaches. Large-scale experiments are conducted in the real-world Tor network by deploying hundreds of thousands of honeypots during a monitoring period of more than three months. Finally, we identify 8 groups of 32 malicious HSDirs, discover 25 probing pattern clusters and reveal 3 major probing purposes.**

Keywords*—*Tor, onion address, honeypot hidden service, malicious Tor hidden service directory

I. Introduction

Tor is one of the most popular anonymous communication systems that provide anonymity on the Internet for both users and service providers. Diverse anonymous services, referred to as hidden services, including web services, email services, online chat services, etc., are deployed over the Tor network to protect the anonymity at the server side. The domain names of hidden services, i.e, onion addresses, are generated by hidden services and uploaded to hidden service directories (HSDirs). A HSDir in the Tor network works as a Domain Name System (DNS) server that is responsible for collecting hidden service descriptors that contain onion addresses uploaded from hidden services. Also, it responds to a request of an onion address from a Tor client so that the Tor client can learn the entry point information and create a path to the corresponding hidden service. Since Tor nodes are deployed by volunteers around the world, adversaries can deploy malicious HSDirs so as to discover the existence of the hidden services and collect onion addresses.

Once the adversaries collect the onion addresses, various attacks, e.g., probing attacks [1], DDoS attacks [2] and hidden service localization attacks [3], [4], can be performed to jeopardize the anonymity of the hidden services. For example, Biryukov *et al.* [1] deploy HSDirs in the Tor network to collect onion addresses and scan the ports of hidden services. Flow watermarks [3], [4] can be embedded into the Tor traffic at the client side and inspected at the hidden service side so as to locate the real IP address of the hidden service. Tan *et al.* [2] compute the six responsible HSDir locations of a target hidden service in advance and deliberately deploy malicious HSDirs near the responsible ones so as to force the victim hidden service to upload descriptors to the malicious ones. Then the malicious HSDirs can perform a DDoS attack by not responding to the Tor client requests.

To mitigate these issues, we propose a large-scale malicious Tor hidden service directory discovery approach and provide theoretical analysis of effectiveness of the approach so as to precisely identify malicious HSDirs that launch passive onion address collection attack and active hidden service probing attack in the Tor network. According to the time interval between the time of onion addresses collected by malicious HSDirs and the time of performing probing attacks, we classify the probing attacks into the rapid probing attack performed within one *descriptor uploading time period* (DUTP) and delayed probing attack performed in more than one DUTP. To identify these malicious HSDirs that launch the rapid probing attack, we customize the honeypot hidden services that only upload one onion address to one responsible HSDir instead of default six HSDirs within one DUTP, and then deploy them in the Tor network to achieve precise one-to-one HSDir monitoring. As a result, once our honeypot h (servi9996 (our)- (at)-271r)

* Corresponding author: Prof. Zhen Ling of Southeast University, China.

the malicious HSDirs that conduct the rapid probing attack within one DUTP. To this end, we build a honeypot onion address collection model to analyze the expected number of distinct HSDirs that can collect our honeypot onion addresses within one DUTP. Then we formally analyze the accuracy of the many-to-one monitoring approach for detecting malicious HSDirs conducting delayed probing attack.

Our major contributions are summarized as follows.

- To the best of our knowledge, we are the first to design and implement customized honeypot hidden services based on one-to-one and many-to-one HSDir monitoring approaches to identifying malicious HSDirs that collect onion addresses and launch probing attacks. Then we propose a novel semantic-based probing pattern clustering approach to classifying the malicious HSDirs in an attempt to shed light on the distinct purposes of the malicious HSDirs.
- We perform theoretical analysis of the capability and accuracy of the malicious HSDir identification approaches. In the one-to-one HSDir monitoring approach, we can derive the expected number of distinct monitored HSDirs with the given number of deployed honeypot hidden services. Moreover, we can obtain the theoretical detection accuracy of the many-to-one HSDir monitoring approach with the given number of deployed and probed honeypot hidden services.
- Extensive empirical experiments are performed to demonstrate the effectiveness of our approaches. We deploy 91,269 and 616,017 honeypot hidden services to conduct one-to-one HSDir monitoring experiments for one week and one month, respectively, and identify 17 malicious HSDirs performing the rapid probing attack. Then we use 23,835 and 7,680 honeypot hidden services to respectively perform one-month many-to-one HSDir monitoring in two phases and discover 19 malicious HSDirs performing the delayed probing attack. We derive 25 semantic probing pattern clusters by analyzing 45,278 probes and correlate the identified malicious HSDirs using the clusters.

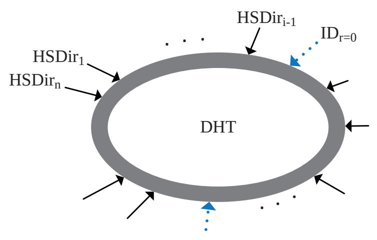
The rest of this paper is organized as follows. We introduce the hidden service mechanism in Section II. Then we introduce the threat model of the two attacks and propose our customized honeypot hidden services based on monitoring approaches to identifying all HSDirs performing the two attacks in Tor network, and then investigate the malicious HSDir clustering approach by analyzing the probing behaviors and payloads in Section III. In Section IV, we analyze the capability and accuracy of monitoring approaches. In Section V, we conduct large-scale evaluation in the real-world Tor networks to demonstrate the effectiveness of our approaches and discuss the limitations of the approaches. We review related work in Section VI. Finally, we conclude this paper in Section VII.

II. Background

In this section, we first introduce how the Tor hidden service mechanism works and then present the generation and usage of onion addresses that are employed to look up the hidden services.

A. Tor hidden service mechanism

Tor is an overlay network that not only provides users with communication privacy protection, but also allows content service providers to publish network services anonymously via a powerful hidden service mechanism. There are four components in the hidden service mechanism, including Tor clients, Tor relays, hidden servers, and HSDirs. Figure 1 illustrates the architecture of the Tor network that provides hidden service. **(1) Tor clients.** A Tor client installs a Tor software that works as a local SOCKS5 proxy and transmits the application layer data into the Tor network. **(2) Tor relays.** A Tor relay, referred to as an onion router, is used to create multi-hop circuits by Tor clients or hidden servers and responsible for relaying the data between the Tor clients and servers via the circuits. **(3) Hidden servers.** A hidden server supports a TCP network service (e.g., a web service) and generates an onion address that is used to uniquely identify the server in the Tor network. The service running on the hidden server is referred to as hidden service. **(4) HSDirs.** A HSDir is a type of Tor relays that not only relays data, but also collects the information from hidden servers, including their public keys. Moreover, HSDirs organized by a Distributed Hash Table (DHT) can assist a Tor client to anonymously look up a hidden server via an onion address.



En

Fig. 1: Tor hidden service mechanism

We illustrate how the hidden service mechanism works to protect the anonymity for a web server. To hide a network identity of a hidden server, the hidden server first selects a Tor relay as a *introduction point* (IPO) and creates a long-lived circuit with default three hops to the IPO. By default, the hidden server has three IPOs that act as reverse proxies to receive requests from Tor clients and forward the requests back to the hidden server via the three-hop circuits. Then the hidden server generates and uploads descriptors, including the information of IPOs, an onion address, a public key, the version of the hidden service, etc., to the HSDirs. A Tor client may obtain the onion address via out-of-band channel (e.g., public web forums). To access the hidden server, the Tor client sends the onion address to the HSDirs and downloads the descriptor of the hidden server. Then the Tor client selects a Tor relay as a *rendezvous point* (RPO) and create a circuit to the RPO. Next, the Tor client builds a three-hop circuit to the IPO and sends the information of the RPO to the hidden server so that the hidden server can establish a three-hop circuit to the RPO. Finally, the Tor client and the hidden server can communicate with each other via the six-hop circuit. Since any Tor relay in the circuit cannot infer the communication relationship between the Tor client and the hidden server, the hidden server can anonymously provide the service without exposing its identity.

B. Onion address generation and usage

An onion address is generated using a pubic key of a hidden service. We take the version 2 of the hidden service mechanism as an example to demonstrate how the onion address is generated and used, since it is a popular version used to date. Specifically, each Tor relay and hidden server first generates a pair of RSA public and private keys. Denote the SHA1 digest of the public key by F, where its length is 160 bits. F is also referred to as the fingerprint of a hidden server or a Tor relay. All HSDirs that contain all the hidden service descriptors in the Tor network are sorted in terms of the fingerprint value in a DHT. Let $F[a : b]$ represent the byte sequence between a and b of F. To generate an onion address of a hidden service, the first 10 bytes of the fingerprint (i.e., $F[0 : 9]$) are encoded using Base32 to obtain 16 bytes string and then concatenates with a string ".onion". Therefore, an onion address with a total length of 22 bytes can be derived in terms of the hidden server public key. The public key included in the descriptor of the hidden service is uploaded to the responsible HSDirs. According to the onion address queried from Tor clients, the HSDirs can look up and send back the right descriptor to Tor clients. The Tor clients can learn sufficient information from the descriptor to build a circuit with the hidden service.

The hidden service computes two different descriptor IDs used to select six responsible HSDirs for uploading the descriptors as shown in Figure 1. The descriptor ID is computed by

$$ID_d = H(F[0 : 9]|ID_s), \tag{1}$$

where H is the SHA1 hash function that returns a result length of 160 bits, $|$ is a concatenation operation, and F is the SHA1 digest of the hidden server public key, i.e., the fingerprint of the hidden server. The ID_s is computed by

$$ID_s = H(t_p|d|r), \tag{2}$$

where t_p is the number of days since January 1, 1970, d is a descriptor cookie that is an optional field (d is null by default), and r is a binary value (i.e., 0 or 1). t_p is derived by

$$t_p = \frac{t + F[0 : 0] \times \frac{86400}{256}}{86400}, \tag{3}$$

where t is the current UNIX time when computing the descriptor ID. Therefore, the descriptor ID changes every 24 hours. Since r is a binary value, it is used to generate two distinct descriptor IDs each time.

The hidden service locates two positions in the DHT of HSDirs in terms of two descriptor IDs and selects two groups of HSDirs with three HSDirs in each group to upload the two descriptors to these two groups, respectively. Recall that the fingerprints of the HSDirs are used as the hash keys to build the DHT. The hash key space is 2^{160}. One descriptor ID of the hidden server is used as a key to look up three following consecutive fingerprints of the HSDirs for uploading the descriptors as shown in Figure 1. Since the descriptor ID changes every 24 hours, the descriptor can be uploaded to another six different HSDirs. Therefore, the DUTP is 24 hours.

Once a Tor client derives the onion address of the hidden server, it can download the corresponding descriptor and build a circuit with the hidden service. The Tor client can first derive the fingerprint (i.e., $F[0 : 9]$) by decoding the onion address using Base32 and then compute the two descriptor IDs in terms of the onion address based on Equation (1), (2), and (3), and look up the corresponding responsible HSDirs to download the descriptor of the hidden server. After the client extract the information of the IPOs from the descriptor, it establishes a connection with the hidden server through the IPO and tells the information of the RPO. Finally, the client accesses the hidden server through the circuit of the RPO.

III. Methodology

In this section, we introduce the threat model of malicious HSDirs and the malicious HSDir detection approach. Further, we present a novel semantic-based probing pattern clustering approach to classify the malicious HSDirs by analyzing the probing behaviors and payloads.

A. Threat model

We assume that an adversary can gather information of the Tor hidden services by first launching a passive onion address collection attack and then an active hidden service probing attack. We elaborate on the two attacks as follows.

Passive onion address collection attack. The adversary can deploy a number of HSDirs to passively collect the descriptors uploaded by the hidden services. Upon collecting the descriptors, the adversary can derive the onion addresses of the corresponding hidden services. Since the hidden service reselects six HSDirs every day, this passive onion address collection attack can theoretically collect all of the onion addresses as the collection time increases.

Active hidden service probing attack. In order to obtain more information about the hidden services, the adversary can deploy a number of Tor clients and leverage the collected onion addresses to actively establish connections to the hidden services to probe the open ports, or infer application layer protocol, or even launch various attacks. In particular, to probe an open port, the adversary chooses a collected onion address as a probing target and initiates multiple Tor circuits with distinct ports to the hidden services. If the circuit is destroyed by the hidden service, the port is not opened. Once locating the open port, the adversary tries to mimic various legitimate application requests (such as HTTP, SSH and SIP) to the target hidden service so as to determine the specific application running on the hidden service.

According to the time interval between the passive onion address collection attack and active hidden service probing attack, we classify the probing attack into two categories: **rapid probing attack** and **delayed probing attack**. The rapid probing attack is to initiate the probing tasks within one day when the HSDirs collect the onion addresses. The delayed probing attack is performed by the HSDirs that wait for a period (i.e., more than one day) to perform the probing task after collecting the onion addresses. Moreover, while performing the probing tasks, the adversaries may probe all the collected onion addresses or may probe some of them. Therefore, we further categorize the delayed probing attack into two classes: *delayed deterministic probing attack* and *delayed probabilistic probing attack*. We attempt to accurately and quickly locate the malicious HSDirs and determine their attack behaviors.

B. Basic idea

Our objective is to identify the malicious HSDirs that collect onion addresses and then cluster their probing patterns so as to shed light on the real purposes of adversaries. To this end, we first collect the information of all the HSDirs in the Tor network since the HSDirs are publicly available.

Then we deploy a number of honeypot hidden services to upload descriptors to all the HSDirs so as to lure malicious HSDirs to perform the probing attacks. To accurately locate the malicious HSDirs performing the rapid probing attack, we generate descriptors of the honeypot hidden services and each honeypot hidden service only chooses one responsible HSDir to upload its descriptor instead of default six HSDirs. Then we wait for 24 hours to see if a honeypot hidden service receives probes. The one-to-one HSDir monitoring approach can effecitively identify the malicious HSDirs performing the rapid probing attack. To locate the HSDirs performing the delayed probing attack, we customize the honeypot hidden service to upload the descriptor to two HSDirs in each DUTP, i.e., one fixed monitored HSDir and one responsible HSDir. The responsible HSDir is used to enable the adversary to correctly find the HSDir and derive the IPO information to establish circuits to the honeypot hidden service. Since the honeypot hidden service continuously uploads the descriptor to a new responsible HSDir in each DUTP, we leverage multiple honeypot hidden services to inspect one HSDir that may conduct delayed probing attack in order to reduce the false positive detection caused by one of the malicious responsible HSDirs. Finally, we analyze the probing behaviors and payloads and investigate a semantic-based probing pattern clustering approach to classifying and inferring the purposes of the adversaries.

C. Identifying the rapid probing attack

To accurately identify the malicious HSDirs performing the rapid probing attack, we customize and deploy our honeypot hidden service to only upload its descriptor to one responsible HSDir. Recall that a vanilla hidden service uploads its descriptor to the six responsible HSDirs until the next DUTP (i.e., 24 hours) comes. We leverage this time period to monitor the potentially malicious HSDirs that launch the rapid probing attack. To achieve precise one-to-one HSDir monitoring, we customize a honeypot hidden service to select the first HSDir of the six responsible HSDirs as the monitored HSDir. After the DUTP, we redeploy a honeypot hidden service with a new descriptor and choose a new monitored HSDir. The one-to-one HSDir monitoring approach ensures that an onion address of a honeypot hidden service can only be learned by one monitored HSDir. Once the honeypot hidden service is probed, we can determine that the monitored HSDir is malicious. Since the responsible HSDirs are randomly selected in terms of the hidden service mechanism, we need to continuously deploy new honeypot hidden services to generate right onion addresses to upload to all the HSDirs. To address this issue, we formally analyze the expected number of deployed honeypot hidden services to monitor all the HSDirs within one day in Section IV. Moreover, the honeypot hidden services record the probing timestamps, ports and payloads for further analysis.

D. Identifying the delayed probing attack

We customize the honeypot hidden service to upload two HSDirs in each DUTP, i.e. one fixed monitored HSDir and one randomly selected responsible HSDir, to determine whether the monitored HSDir conducts the delayed probing attack or not. The adversary performs the delayed probing attack after several DUTPs. Since a hidden service always randomly chooses HSDirs in each period to upload the descriptors, it can hardly upload the descriptor to the same monitored HSDir in these time periods. To address this issue, we deliberately force the honeypot hidden service to upload its descriptor to the same monitored HSDir in every DUTP. Consequently, the monitored HSDir is able to collect the latest descriptor of the honeypot hidden service during our monitoring period. In addition, a responsible HSDir is also selected by the honeypot hidden service to allow the monitored HSDir to access it if the monitored HSDir intends to perform the delayed probing attack. However, a new responsible HSDirs is randomly selected as the DUTP changes. When the honeypot hidden service is probed by adversaries, it is nontrivial to differentiate the probing of the monitored HSDir from that of one of the responsible HSDirs that may be malicious. Moreover, the longer the monitoring period lasts, the more responsible HSDirs can collect our honeypot descriptors. Therefore, it can increase the false positive detection of malicious HSDirs using one-to-one HSDir monitoring approach.

To reduce the false positive detection, we adopt many-to-one HSDir monitoring approach, i.e, using multiple honeypot hidden services to monitor the same HSDir. Since there are around 4,000 HSDirs, a large number of hidden services deployed to monitor each HSDir may cause performance degradation in the Tor network. To mitigate this issue, we first use three honeypot hidden services to monitor each HSDir in the first monitoring period. If a honeypot hidden service receive the probes, the corresponding monitored HSDir is considered as a potential malicious one. In the second monitoring period, ten honeypot hidden services are deployed to monitor each potential malicious HSDir to further improve the true positive detection. If the number of probed honeypot hidden services is greater than a threshold, we can confirm that the monitored HSDir is malicious. We perform theoretical accuracy analysis of the many-to-one HSDir monitoring approach for identifying a malicious HSDir in Section IV.

E. Clustering malicious HSDirs

We analyze the probing behaviors of adversaries to shed light on the probing purposes of the attacks in the Tor network. To quickly collect hidden service onion addresses in the Tor network, adversaries usually deploy multiple malicious HSDirs to improve efficiency and probe these collected onion addresses simultaneously. Different adversaries may adopt distinct probing behaviors to retrieve the information of the hidden services, including the open ports, the application types of the hidden services, etc. According to the different probing behaviors, we try to categorize the adversaries into several groups. Note that we record the probed timestamps, port numbers, onion addresses and payloads. To this end, we first segment a sequences of continuous probes into one probing task in terms of the probing time interval. Since 95% of hidden service connection establishment time is less than $65s$ [5], we choose the time interval $65s$ to segment the probes.

Once deriving the segmented probing tasks, we analyze the probing payloads so as to enrich the semantic features of the probes. We extract the probing semantic features of a probe and concatenate the probed port number with the semantic feature to represent a individual probe. Specifically, We use a string "port_feature" to indicate a probe, where the "_" symbol is used for concatenation. According to our empirical analysis, the semantic feature of a probe includes three types, i.e, probing TCP ports, applications, and vulnerabilities. Therefore, we use the string "TCP", a string of an application name (e.g., "SSH", "SMB" and "HTTP"), and a string keyword in the payload of a vulnerability (e.g., "dmdt" [6] and "tnmp" [7]) to represent the feature of three types, respectively. If a payload is a series of hex stings, we use "HEX" to represent the semantic feature of a potential vulnerability probe. Moreover, if a HTTP request is used to probe the HTTP application, we extract more features to represent the HTTP probe, including the information of the HTTP version, request method (e.g., GET, POST, etc.), path, and user agent. For example, the string "80_http_HTTP/1.1_GET_/_Mozilla/5.0+(X11;+Linux+x86_ 64;+rv:68.0)+Gecko/20100101+Firefox/68.0" indicates that the TCP port 80 is probed and the HTTP feature includes the "HTTP/1.1" version, "GET" method, "/" path, and a user agent (i.e., the rest string). In this way, we enrich the probes in each task using the semantic features.

We leverage an edit distance, i.e., the *Levenshtein* distance, to measure the similarity between two probing tasks. Since a probing task includes a sequence of probed port numbers and semantic features arranged in chronological order, the *Levenshtein* distance is used to compute the minimum amount of operations, e.g., insertions, updations or deletions of the port numbers required to transform one probing task to another. Let $\mathcal{S}(T_a, T_b)$ be the similarity between the probing task T_a and T_b, where the numbers of the probes of tasks T_a and T_b are $|T_a|$ and $|T_b|$, respectively. We then normalize the similarity by

$$\mathcal{S}^0(T_a, T_b) = \frac{|T_a| + |T_b| - \mathcal{S}(T_a, T_b)}{|T_a| + |T_b|} \tag{4}$$

According to the normalized similarity, we employ a density-based clustering algorithm, i.e., *DBSCAN* [8], to cluster the probing tasks so as to discover diverse probing purposes. In particular, if the normalized similarity of two probing tasks is more than ε, i.e., $\mathcal{S}^0(T_a, T_b) > \varepsilon$, the two tasks are classified into a group, i.e., a neighborhood, where the radius of a neighborhood ε defines how similar the neighbors (i.e. the probing tasks) are in neighborhoods. Once we obtain the clusters of the probing tasks, we can further analyze the purposes of adversaries that are illustrated in Section V.

IV. Analysis

In this section, we theoretically analyze the capability of the one-to-one HSDir monitoring approach and the accuracy of the many-to-one HSDir monitoring approach, respectively.

A. Capability analysis

We analyze the total number of customized honeypot hidden services deployed to upload honeypot descirptors to all of the HSDirs so as to identify malicious HSDirs performing the rapid probing attack in one DUTP. Recall that, to identify the attack, one honeypot hidden service randomly generates one descriptor at one time in terms of the Equation (1), (2), as well as (3) and then uploads it to the first responsible HSDir. Therefore, if a randomly generated honeypot descriptor ID is located between the fingerprint of the i^{th} HSDir and the $(i-1)^{th}$ HSDir, i.e., between F_i and $F_{i\ 1}$, the honeypot hidden service can upload the descriptor to the i^{th} HSDir. We have the following theorem to compute the **collection probability** that a HSDir can collect a honeypot descriptor in one DUTP.

Theorem 1. *The* ***collection probability*** *of the i^{th} HSDir can be derived by*

$$p_i = \frac{L_i}{\sum_{i=1}^{m} L_i} \tag{5}$$

where m is the total number of HSDirs in the Tor network, and L_i is the difference of the fingerprint values between the i^{th} HSDir and the $(i-1)^{th}$ HSDir in the DHT, i.e., $L_i = F_i - F_{i\ 1}$.

we denote a random variable by X_i and define

$$X_i = \begin{cases} 1, \text{if the } i^{th} \text{ HSDir collects a honeypot descriptor} \\ 0, \text{otherwise} \end{cases} \tag{6}$$

After deploying ℓ honeypot hidden services, ℓ descriptors are uploaded in one DUTP. Then the probability that none of ℓ honeypot descriptors is collected by the i^{th} HSDir is

$$P(X_i = 0) = (1 - p_i)^{`} \tag{7}$$

Then the number of different HSDirs that collect honeypot descriptors is

$$X^{`} = \sum_{i=1}^{m} X_i \tag{8}$$

Since

$$\begin{aligned} E(X_i) &= P(X_i = 1) \times 1 + P(X_i = 0) \times 0 \\ &= P(X_i = 1) = 1 - P(X_i = 0) = 1 - (1 - p_i)^{`}, \end{aligned} \tag{9}$$

we can have the following theorem.

Theorem 2. *The expected number of distinct HSDirs that collect honeypot descriptors after uploading ℓ honeypot descriptors in one DUTP becomes*

$$\begin{aligned} E(X^{`}) &= E(\sum_{i=1}^{m} X_i) \\ &= E(X_1) + E(X_2) + \cdots + E(X_m) \\ &= 1 - (1-p_1)^{`} + \cdots + 1 - (1-p_m)^{`} \\ &= m - \sum_{i=1}^{m} (1-p_i)^{`} \end{aligned} \tag{10}$$

Figure 2 illustrates the relationship between the expected number of the distinct HSDirs collecting honeypot descriptors and the number of deployed honeypots. According to the real-world Tor data, there are 4,000 HSDirs on June 30, 2021. As we can see from the figure, 95% of HSDirs can collect honeypot descriptors in one DUTP by deploying $90,000$ honeypots. Since deploying a large number of honeypots in one DUTP may take up a lot of Tor network resources, we can deploy these honeypots in different periods in practice. Since we adopt one-to-one monitoring approach to detecting the potential malicious HSDirs that perform the rapid probing attack, our honeypot hidden service can identify such malicious HSDirs with 100% accuracy.

B. Accuracy analysis

We analyze the accuracy of the many-to-one HSDir monitoring approach to identifying the delayed probing attack. Recall that a honeypot descriptor is uploaded to a fixed monitored HSDir and a new randomly selected responsible HSDir in each DUTP. As a result, after several periods, one of the responsible HSDirs may be malicious and performs the delayed probing attack, so that it can cause a false positive detection of the approach. We denote the number of malicious HSDirs performing the delayed probing attack by u. Then the probability that a customized honeypot hidden service randomly selects a malicious responsible HSDir is u/m. After d DUTPs, a honeypot hidden service uploads its descriptors to d responsible HSDirs, in addition to uploading descriptors to

5

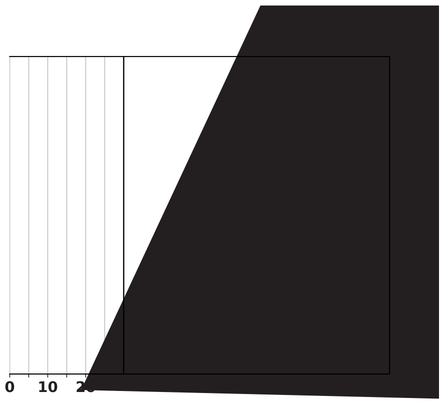


Fig. 2: Deployment number of honeypot hidden services

a fixed monitored HSDir. Once the honeypot hidden service is probed, we can infer that either the monitored HSDir or one of the responsible HSDirs is malicious. Let $p_c = (1 - \frac{u}{m})^d$ be the probability that all of the d responsible HSDirs are benign. Then the probability that at least one of the d responsible HSDirs is malicious becomes $1 - p_c$. If we deploy H honeypots and more than or equal to t out of H honeypots receive probes, the probability that all the probes come from either the malicious monitor HSDir or the malicious responsible ones is

$$p_b = \sum_{i=t}^{H} \binom{H}{i} [1 - (1 - p_s)p_c]^i [(1 - p_s)p_c]^{H \quad i} \tag{11}$$

where p_s $(0 < p_s \leq 1)$ is the probability that a monitored malicious HSDir performs *delayed probabilistic probing attack*, i.e, only probing a fraction p_s of all collected onion addresses. If $p_s = 1$, the adversary adopts the *delayed deterministic probing attack*, i.e., probing all of the collected onion addresses. In addition, the probability that all the probes only come from the malicious responsible HSDirs can be calculated by

$$p_r = (1 - p_s)^H \sum_{i=t}^{H} \binom{H}{i} (1 - p_c)^i {p_c}^{H \quad i} \tag{12}$$

Therefore, in theory, we have the false positive detection, i.e, $\frac{p_r}{p_b}$, that the monitored HSDir is misidentified as a malicious one if we deploy H honeypots and more than or equal to t honeypots receive probes. Then the detection accuracy becomes

$$\mathcal{A} = 1 - \frac{p_r}{p_b} \tag{13}$$

Figure 3, 4, 5, and 6 show the detection accuracy versus t, p_s and H. Given that Sanatinia *et al.* [9] conclude that the estimated the number of malicious HSDirs is around

Fig. 3: Accuracy versus t, p_s with 50 malicious HSDirs (H=3)

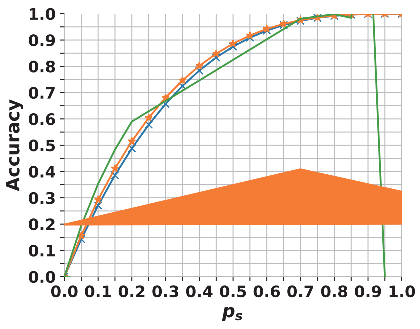


Fig. 4: Accuracy versus t, p_s with 100 malicious HSDirs (H=3)

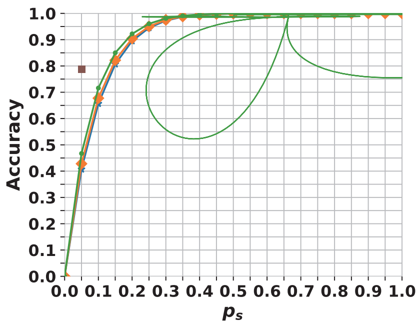


Fig. 5: Accuracy versus t, p_s with 50 malicious HSDirs (H=10)

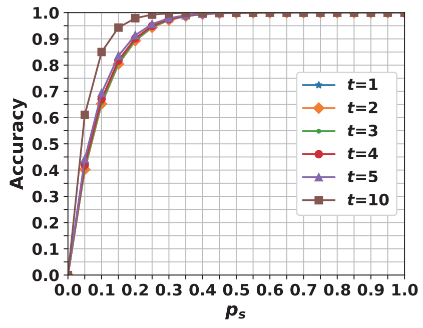


Fig. 6: Accuracy versus t, p_s with 100 malicious HSDirs (H=10)

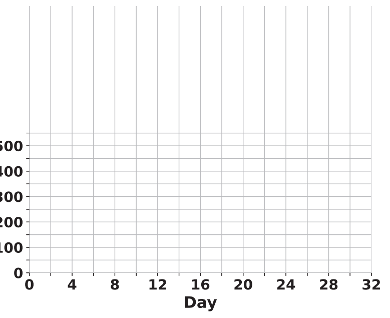


Fig. 7: Number of probed distinct honeypot hidden services

Fig. 8: The top 8 most probed ports

C. Delayed probing attack detection

We conduct a one-month many-to-one HSDir monitoring experiment to identify malicious HSDirs performing delayed probing attack. To this end, we monitor a total of 7,945 HSDirs that appear in the Tor network from Oct. 1 to Oct. 15, 2020. Recall that we perform two-phase experiments to reduce the influence in the Tor network. In the first phrase, we deploy 23,835 honeypots to monitor the 7,945 HSDirs from Oct. 20 to Nov. 20, 2020. We use 3 customized honeypots to monitor one HSDir. During the monitoring period, the 823 honeypots receive a total of 45,278 probes. Table I illustrates the number of probed honeypots that monitor the same HSDir and the number of the monitored HSDirs performing the probing attack in the two phases. In the first phase, among the 7,945 HSDirs, 725 and 32 HSDirs probe one and two out of three of their monitoring honeypots, respectively. 11 HSDirs probe all of the three monitoring honeypots. We treat HSDirs that conduct at least one probe as potential malicious HSDirs.

To monitor the 768 potential malicious HSDirs, we conduct the second phase experiment from December 2, 2020 to January 2, 2021. Since 10 honeypots are used to monitor each HSDir, we deploy a total of 7,680 honeypots. During this monitoring period, 417 out of 7,680 honeypots receive a total of 6,937 probes. According to the theoretical analysis in Section IV, if we deploy 10 honeypots for each monitored HSDir and the number of probed honeypots is at least 3, the detection accuracy can achieve 94.5%. Therefore, as shown in Table I, we conclude that 19 (i.e., 9+1+9) malicious HSDirs performing delayed probing attack are identified in the second phase. 9 out of 19 malicious HSDirs perform the *delayed deterministic probing attacks*, while the rest conduct the *delayed probabilistic probing attacks*.

TABLE I: The distribution of received probes in the first phase and second phrase

First phrase	Honeypots (#)	0	1	2	3	-	-
	HSDirs (#)	7177	725	32	11	-	-
Second phrase	Honeypots (#)	0	1	2	3	4	10
	HSDirs (#)	493	216	40	9	1	9

D. Probing purpose analysis

By analyzing the probing information recorded by the honeypot hidden services, we try to identify the probing behaviors and purposes of adversaries. We use the probing data collected in the first phase of identifying delayed probing attacks as we monitor all of the HSDirs in the one-month experiment. During this experiment, we receive a total of 45,278 probes that probe more than 800 distinct honeypots. Figure 7 illustrates the number of daily probed and accumulated probed distinct hidden services, respectively. We can see that a probing spike appears about every 4 days. It implies that the adversaries may periodically perform delayed probing attacks using the collected onion addresses. Figure 8 illustrates the top 8 probed ports. In fact, a total of 143 distinct ports are probed in the one-month experiment. The most probed port is port 80 with 6164 probes, followed by port 443 with 361 probes. These two ports are respectively used to run the HTTP and HTTPS services by default. It indicates that the service of interest to the adversaries is the Web service. Other ports that may be open for the Web services are also probed, such as 81, 8080, 8008, 8000, etc. The port 113 often runs an identification/authorization service and can be used by the adversaries to probe the server for the connection information, such as user identity. The port 22 is used for the SSH service by default and is often probed by the adversaries for discovering weak login passwords. In addition, diverse special ports are probed as well. For example,

7

TABLE II: Semantic features of probes

Probes	Probe type	Semantic feature	Number
With payloads (17,588)	TCP port probes	TCP	7,708
	Application probes (5,953)	**HTTP**	4,607
		SSH	259
		SIP	240
		SMTP	231
		Ident	206
		SMB	203
		RTSP	190
		JDWP	15

TABLE V: Malicious HSDirs correlation

Group	HSDir (#)	Probing attack	Probing purposes	Cluster (#)
1	1	Rapid probing	Vulnerability probes	1
2	2		Only port 80 probes	1
3	1			1
4	10			1
5	7	Delayed probing		2
6	5		Hybrid probes	1
7	2			1
8	4	Both		1

conduct the vulnerability probes. 20 HSDirs in the 2^{nd}, 3^{rd}, 4^{th}, and 5^{th} group only focus on port 80. 10 and 3 out of 20 HSDirs performing the rapid probing attack conduct a single TCP port 80 probe and multiple repeating TCP port 80 probes, respectively. The rest 7 out of 20 HSDirs performing the delayed probing attack launch multiple repeating HTTP application probes with two different HTTP user agents. 11 HSDirs in the rest groups perform HTTP(S) and SSH application probes, i.e., hybrid probes. 4 HSDirs in the last group perform both the rapid and delayed probing attacks.

F. Discussion

We notice that 9 probing patterns are used by the 32 identified malicious HSDirs. However, we derive 25 probing clusters by analyzing 1,649 probe tasks. It indicates that we do not identify all of the malicious HSDirs. There are two major reasons that we cannot find all the malicious ones. Firstly, since the malicious HSDirs do not always keep online, our honeypots cannot monitor them in our entire experimental period and can miss the probes. Then it can significantly reduce our detection

REFERENCES

[1] A. Biryukov, I. Pustogarov, F. Thill, and R.-P. Weinmann, "Content and popularity analysis of tor hidden services," in *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pp. 188–193, 2014.

[2] Q. Tan, Y. Gao, J. Shi, X. Wang, and B. Fang, "A closer look at eclipse attacks against tor hidden services," in *Proceedings of the IEEE International Conference on Communications (ICC)*, pp. 1–6, 2017.

[3] A. Biryukov, I. Pustogarov, and R.-P. Weinmann, "Trawling for tor hidden services: Detection, measurement, deanonymization," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, pp. 80–94, 2013.

[4] Z. Ling, J. Luo, K. Wu, and X. Fu, "Protocol-level hidden server discovery," in *Proceedings of the 32nd IEEE International Conference on Computer Communications (INFOCOM)*, pp. 1043–1051, 2013.

[5] K. Loesing, W. Sandmann, C. Wilms, and G. Wirtz, "Performance measurements and statistics of tor hidden services," in *Proceedings of the International Symposium on Applications and the Internet*, pp. 1–7, 2008.

[6] Isakbaetle3r9, "Am I in trouble? (flask/strange requests):learnpython." https://www.reddit.com/r/learnpython/comments/1cj1kt/am_i_in_trouble_flaskstrange_requests/, 2021.

[7] vosec, "Imagenow perceptive content server 7.1.4 dos | opposition security." https://www.oppositionsecurity.com/imagenow-7-1-4-dos/, 2021.

[8] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD)*, vol. 9600.00091 (v)192-62.99801 (1-erce1351,)-349.99593 1.991.

98] Sanatnita
(detifymaing)4335.0029 *Forcisedings*
CNSDQ,onfeferenceations
1-e1273–35/, 2191.