Identification of MEEK-Based TOR Hidden Service Access Using the Key Packet Sequence

Xuebin Wang[1 2], Zhipeng Chen[4], Zeyu Li[3(✉)], Wentao Huang[3], Meiqi Wang[1 2], Shengli Pan[3], and Jinqiao Shi[3(✉)]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
shijinqiao@bupt.edu.cn
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] School of Cyberspace Security, Beijing University of Posts and Telecommunications
[4] National Internet Emergency Center(CNCERT/CC), Beijing, China

Abstract. Tor enables end user the desirable cyber anonymity with obfuscation technologies like MEEK. However, it has also manifested itself a wide shield for various illegal hidden services involved cyber criminals, motivating the urgent need of deanonymization technologies. In this paper, we propose a novel communication fingerprint abstracted from key packet sequences, and attempt to efficiently identify end users¡¯ MEEK-based access to Tor hidden services. Specifically, we investigate the communication fingerprint during the early connection stage of MEEK-based Tor rendezvous establishment, and make use of deep neural network to automatically learn and form a key packet sequence. Unlike most of existing approaches that rely on the entire long communication packet sequence, experiments demonstrate that our key packet sequence enabled scheme can significantly reduce both the time and hardware resource consumption for the identification task by 23%-37% and 80%-86%, respectively, while being able to keep a slightly better accuracy.

Keywords: Tor · Hidden service · Traffic analysis · MEEK

1 Introduction

Tor [17], used by more than two million users daily [1], is one of the most popular anonymous communication systems, aiming to protect users' online privacy. Tor also provides hidden services (HSs), the so-called darknet, to protect server-side anonymity. Therefore, some people use this mechanism to publish sensitive contents on hidden services, making the deep-dark cyberspace a hotbed of crime [4]. Hence, it is necessary to identify Tor hidden service traffic.

In order to enhance the availability of the network in censorship countries, Tor proposes many obfuscation methodologies to bypass censorship, such as Tor bridge mechanisms, MEEK-based [5] obfuscation and Obfs4-based [3] obfuscation. According to Tor project statistics, MEEK is one of the most popular

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

bridges currently. The obfuscation introduced by MEEK protocol brings difficulties for detecting and identifying Tor hidden service traffic.

Many previous work has shown that it is possible to identify whether a user is accessing a hidden service [8,9,11] and even distinguish the specific hidden service the user is accessing [7,10,12,15,19,20]. However, the detection granularity of prior work is in the unit of entire access trace, leading to a relatively lower detection timeliness and much more memory resource consumption, which is not suitable for online identification. Moreover, the obfuscation brings by MEEK protocol has not been taken into consideration, need to be thoroughly analyzed.

In this paper, we present a novel approach to identify Tor hidden service access activity with key sequences under MEEK-based obfuscation scenario, which only uses the specific TCP packet sequence as input and identify the Tor hidden service access behavior in the early stage of the access procedure, effectively improving the identification timeliness as well as reducing the cost of hardware resources. The contributions of this paper are listed as follows:

- verify that there does exist a TCP package sequence contributes significantly to identify Tor hidden service access behavior under MEEK-based obfuscation scenario.
- propose a novel method to identify Tor hidden service access activity under MEEK-based obfuscation scenario only based on key TCP package sequence, which can effectively improve the identification timeliness as well as reduce the cost of hardware resources.
- capture a large and practical dataset on four different MEEK-based obfuscation scenarios to validate our method. Besides, we make the dataset public[5], allowing researchers to replicate our results and evaluate new approaches in the future.
- Based on the collected dataset, compared with the existing method using all data as input, only with the key TCP packet sequence as input, the identification effect is improved by 3%-4%, the timeliness is improved by 23%-37%, and the hardware resource consumption is reduced by 80%-86%.

Organization. The rest of the paper is organized as follows. In section 2, we illustrate the background on Tor hidden service, MEEK protocol as well as the related work. We present the key sequence based identification methodology in section 3. In section 4, we describe the data collection and processing methodology. We next present, in section 5, our observations and experimental results under four MEEK-based obfuscation scenarios. Finally, we draw the conclusion in section 6.

[5] The dataset can be found on the following URL: https://github.com/Meiqiw/meek-mingan/.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

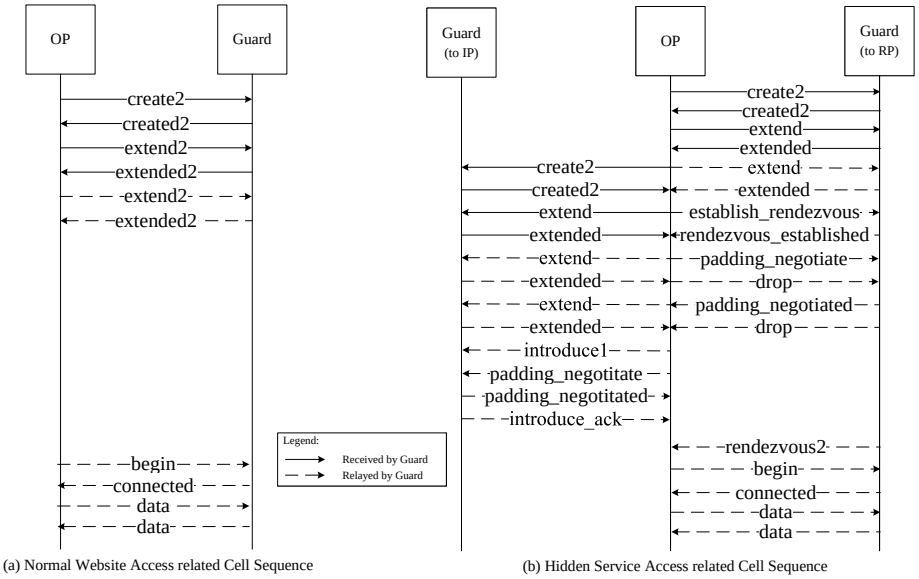


Fig. 1. Different Cell Sequences Between Hidden Service and Normal Website Access Behaviors

2 Background and Related Work

In this section, we introduce the necessary background for our work, including Tor hidden service mechanism and MEEK obfuscation protocol. Besides, we describe the related work about MEEK detection as well as WFP (Website Fingerprint Attack) on Tor hidden service.

2.1 Tor Hidden Service

Hidden service was launched in 2004, it can hide the location information for the service provider, thus can anonymous the responder. According to the specification [2], there are obvious differences between hidden service and normal website access activities. Fig. 1 shows the detail different cell sequences exchanged when accessing normal website and hidden service from the OP(Onion Proxy) side. There are less control cells when accessing the normal website than that of accessing the hidden service, like *relay-establish-rendezvous* and *relay-rendezvous2*. Standing between the OP and the entry relay, we can see that the cell sequences is significantly different when accessing the hidden service and accessing the normal website. Accordingly, the different cell sequences will cause the difference of traffic sequences between two accessing behaviors as Tor uses TLS as the transport layer protocol which is based on TCP, and that provides the basis for our work.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

2.2 Meek

MEEK is one of the most popular pluggable transports commonly used by users in censorship areas. MEEK uses domain fronting [5] technology to avoid censorship, so as to access a domain name prohibited by the censor. The structure of MEEK mechanism mainly include three different parts: MEEK client bounded with Tor client, fronted server with an allowed domain name provide by cloud service provider and MEEK server bounder with Tor router. Firstly, MEEK client encapsulates the data from Tor client into the payload of an HTTP post request and sends it to the fronted server within a HTTPS request. The domain name in the DNS query and the SNI field of the TLS in the request are both **allowed.example**, and the host header field in the HTTP part of the request is **forbidden.example**. Because the HTTP part has been encrypted by TLS, the censor can only get **allowed.example**, but can't resolve the domain name(forbidden.example) that user really wants to access. Secondly, after receiving the request, the fronted server extracts the internal HTTP request and sends it to the MEEK server. What's more, as the MEEK server is not allowed to push data to the MEEK client actively, the MEEK client needs to continuously poll the fronted server to check whether the MEEK server forwards data back and finally extracts the response content.

2.3 Related Work

Many researches use machine learning methods to detect tor pluggable transport traffic [6,13,16], and have achieved good results. MEEK is one of Tor's pluggable transports that has a wide audience, more and more work has been done on MEEK traffic recognition [14, 18, 21, 22], which all got state-of-the-art detection performance.

Another research area related to our work is website fingerprinting attacks. WFP needs to collect client-side traffic, and then use traffic classification method to train the classifier. The difference is that WFP needs to answer the question of which website the user has visited, while we need to answer whether the user has visited Tor hidden service. Many outstanding works [7,10,12,15,19,20] in the WFP field have proposed classification algorithms with great reference value.

At present, there are few researchers who pay attention to distinguishing whether users are accessing a hidden service. In the existing researches on distinguishing hidden services, attackers can be divided into node level attackers and network level attackers according to their capabilities and locations. **Node level attackers** control relays of Tor network, passively record cell sequences and traffic data, and can infer whether users are accessing hidden services by analyzing the traffic of the collected information. Kwon et al. [9] showed that by controlling the entry nodes, the traffic accessing hidden services can be distinguished from the traffic accessing normal services with an accuracy rate of over 90%. Recently, Jansen et al. [8] control the middle nodes to execute the circuit fingerprinting attach, which proves that the attack from the middle nodes is as

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

effective as the attack from the entry nodes. However, the effectiveness of node level attacks largely depends on the number of controlled nodes. While **network level attackers** are located in the network path between users and entry nodes, and they can only collect some widely known traffic data. Hayes and Danezis [7] detected the onion site in 100,000 sites, and distinguished the onion site from other conventional web pages. The true positive rate was 85%, and the false positive rate was only 0.02%. Panchenko et al. [11] used machine learning to identify hidden service related traffic, with an precision rate of over 90% and a recall rate of over 80%. These works are all about taking the whole traffic flow as the input of machine learning or deep learning, and then extracting features for recognition, which is not suitable for online classification scenario.

Besides, the obfuscation brings by MEEK protocol has not been taken into consideration, need to be thoroughly analyzed. In this paper, we present a novel approach to identify Tor hidden service access activity with **key sequence** under MEEK-based obfuscation scenario, which only uses the specific TCP packet sequence as input and identify the Tor hidden service access behavior in the early stage of the access procedure, effectively improving the identification timeliness as well as reducing the cost of hardware resources.

3 METHODOLOGY

3.1 Threat Model

In this work, we assume a network-level adversary, which means the adversary can not process traffic by modifying, dropping or delaying packets. Besides, we assume that the adversary knows the client's identify so that the position of adversary is between the OP and MEEK Server, aiming at distinguishing normal website or hidden service access behavior without decrypting packets. The adversary scenario can be shown in Fig. 2.

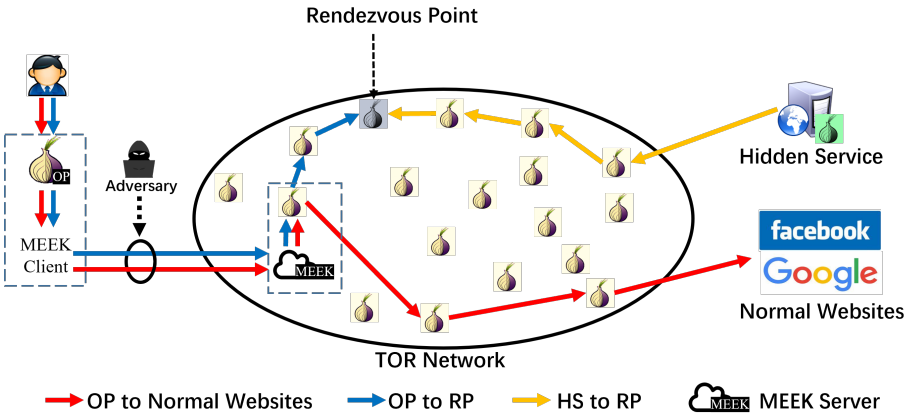


Fig. 2. The Adversary Scenario of This Approach

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

3.2 Key Sequence Definition

According to the Tor hidden service protocol, there exists some particular cells representing the start point and ready status for accessing Tor hidden service. The **establish_rendezvous** cell which is send by the OP along the OP-RP circuit indicating the start point of accessing Tor hidden service, while the **rendezvous2** cell which is received from HS along the OP-RP circuit indicating the ready status of accessing Tor hidden service. In this paper, we define the TCP sequences between sending **establish_rendezvous** and receiving **rendezvous2** as the **key sequence** for distinguishing Tor hidden service access behavior from public website access activity.

3.3 Key Sequence Based Identification Methodology

According to the Tor specification, many circuits with different purposes are multiplexed in one single Tor TLS connection, resulting that a network level attacker can not distinguish each circuit from others. What's more, according to the MEEK protocol, MEEK client only persists one HTTPS connection with MEEK server simultaneously, indicating that all data communicating between OP and HS are encapsulated and transferred in one single HTTPS connection. Additionally, the particular polling mechanism of MEEK protocol generates numerous heartbeat packages obfuscating the location distribution and sequence length of the **key sequence** defined above.

In this paper, from the perspective of a network level attacker, we propose a lightweight method to identify Tor hidden service access behavior based on key sequence under MEEK-based obfuscation scenario. As shown in Fig. 3, the identification framework contains Data Collection, Data Preprocess and Prediction three phrases. In data collection phrase, we collect traffic record and cell information as our raw data, we describe the data collection process in next section in detail. While in data preprocess phrase, we extract TCP package sequence as well as cell sequence and build our dataset which we will describe in next section. At last, in the prediction phrase, we find the Key Sequence and use deep learning networks to perform identification task. Firstly, we perform statistics on all the cell sequences, finding the start time point and end time point of the key sequence which is shown in the green box. In detail, we count the absolute position of establish_rendezvous and rendezvous2 in the unit of one connection which is shown in the yellow box. Then, we find the corresponding TCP package index and window size with the help of start time point and end time point respectively, which is shown in the red box. Moreover, we extract the **key sequence** as input of identification model to distinguish Tor hidden service access activity from public website access activity.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

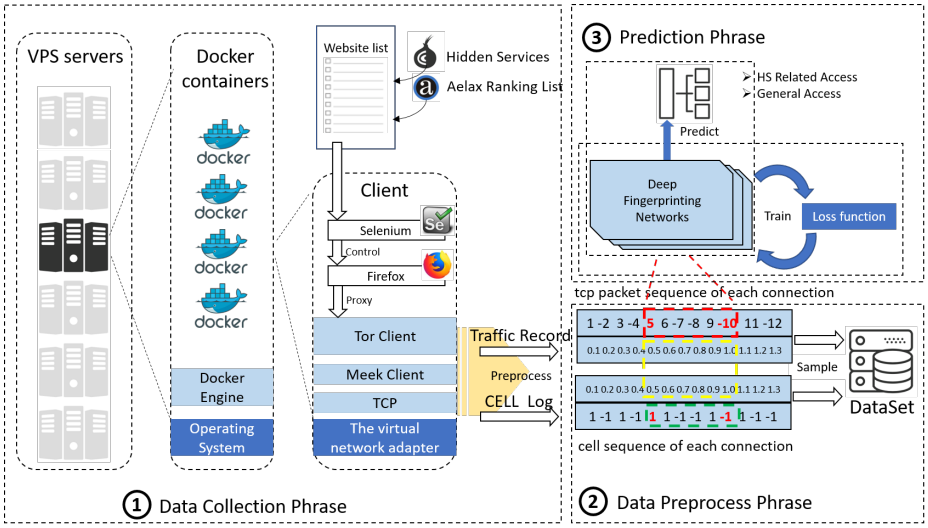


Fig. 3. KEY Sequence Based Hidden Service Access Identification Framework

4 DATA COLLECTION AND PROCESSING

4.1 MEEK Access Scenarios Configuration

Despite of the public MEEK bridge node extracted from the Tor browser, user can setup private MEEK bridge on various CDNs(cloud service providers). The forwarding mechanism of different CDNs may have impact on the location and window size of key sequences used for Tor hidden service access identification. Thus, in order to simulate users using different CDNs for Tor network access, four scenarios are defined: **public, cdn77, fastly, stackpath**. In all four scenarios, users use the MEEK plugin to browse the web in the Tor network, where public refers to the public MEEK bridge node setup on azure cloud service by Tor official, and the other three are private MEEK bridges built on different cloud platforms respectively.

4.2 Data Collection

To increase the diversity of traffic, we use different country networks for different scenarios: **public** and **cdn77** within China, which use Virtual Machines (VMs) of 8-core CPU, 31G of RAM and 1T of disk; **fastly** scenario leases two VPS nodes in the US and **stackpath** scenario leases two VPS nodes in the UK, each VPS is provided by Vultr, with a 4-core CPU, 8G of RAM and 160G of disk. In each VPS or VMs, multiple dockers are used for distributed data capture which is shown in the data collection phrase of Fig. 3. In each docker, we use `Selenium` (version 3.12.0) to control headless browser `Firefox` (version 60.0.2) to access the Tor network, utilizing a SOCKS5 proxy listened by Tor. At the same time, we require the client to access the Tor network via the `MEEK` plugin with corresponding configuration in the torrc file.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

Table 1. DATA COLLECTION of FOUR SCENARIOS

Scenario	Position	Time	WebsiteTrace	OnionTrace	Total
public	China	2021.4-2021.5	15071	10622	25693
cdn77	China	2021.12-2022.1	15370	20984	36354
fastly	US	2021.12-2022.1	31174	27494	58688
stackpath	UK	2021.12-2022.1	32294	28129	60423

For each access activity, we record traffic trace of web pages leveraging `tcpdump`. Each Web page is given 120 seconds to load, and upon loading the page, it is left open for an additional 10 seconds, after which the browser is closed and the Tor process is killed. Next, `tcpdump` and Tor process are restarted. A script to monitor the bootstrap status of the Tor process is deployed, ensuring Tor is ready before each visit. With this setup, new connections and circuits are established each time as the client visits a website, ensuring that we never used the same circuit to download more than one instance of a single page. Besides, in order to find out the location and window size of key sequence, we modify the source code of Tor OP and record the connection creation, circuit construction, stream info, cell sequences into the notice log file, aiming at showing light on the real activity Tor instance occurs during each access trace.

Following our data collection method, we use Alexa Top websites and Tor hidden services[6] as our target website for four scenarios, each with 10,000 websites. After data collection, we filter out invalid traces and outliers, which caused by timeout or crash of the browser or Selenium driver. Eventually, we obtain huge amount of traces as shown in TABLE 1, each trace accomplishes with one corresponding notice log file.

4.3 Data Extraction and Processing

Our dataset contains two type of data: **traffic traces** and **cell records**. With the cell records, we split the cell sequences according to different connectionID, generating cell sequences of one specific connection, which commonly multiplexed with multiple circuits. For the traffic traces, we split each traffic trace into different flows. And then transfer each flow into TCP packet sequences. The detail processing procedure described as follows respectively.

Cell Record Processing By parsing the notice log file, much basic information about the connection, circuit and cell are extracted, including connection creation time, connectionID, circuitID, cell command and direction etc. Firstly, we order cells of each circuit with timestamp and tag the circuit with different flags according to the circuit purpose. We divide circuit into two categories: **clientdata**, meaning that this circuit is built to access non-hidden service related data, **clientip, clientrp, clienthsdir**, those three are hidden service related.

[6] As the prior work, we chose hidden services based on the list provided by the .onion search engine http://www.ahmia.fi/.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

Table 2. RESULTS of DF VS CUMUL

	input_size	DF				CUMUL			
		acc	pre	recall	f1	acc	pre	recall	f1
public	857	0.9684	0.9578	0.98	0.9687	0.9036	0.9042	0.9036	0.9035
cdn77	1893	0.986	0.9742	0.9984	0.9862	0.9234	0.9244	0.9234	0.9234
fastly	793	0.9845	0.9797	0.9894	0.9845	0.9192	0.9195	0.9192	0.9191
stackpath	3217	0.9872	0.9792	0.9956	0.9873	0.9295	0.931	0.9295	0.9294

Secondly, we select circuits belongs to the same connection and put corresponding cells together, generating the cell sequence of one specific connection. At last, we tag each connection according to the circuits categories multiplexed in the same connection.

Traffic Trace Processing As for collected traffic traces, our process performs as follows: Firstly, we tag each connection the same category as the connection recorded in the notice log file which processed in the prior subsection. What's more, we parse the single flow pcap files into TCP packet sequences, with the help of `Tshark` (version 1.12.1), an industrial grade widely used tool for network traffic analysis.

DATASET MEEK21 After completing the above operation, we eventually obtain *MEEK21*, consisting of two subsets: 10,000 instances of hidden service related and general website related for each scenario, each instance contains packet size, direction and time information of each packet.

5 Evaluation and Discussion

In this section, we first select the state-of-the-art classification model used in this paper. Next, we verify the existence of the key sequence. Then, we analyze the location distribution and window size of the key sequence under four MEEK scenarios. At last, we perform iterative experiments to learn the best choice of the start point and window size to perform our attack.

5.1 Select Classification Model

To find the best performing model for the method proposed in this paper, we compare the best performing CUMUL [10] method for machine learning and the best performing DF [15] method for deep learning. To check the robustness and accuracy of the models, we divide the dataset into a training set, a validation set and a test set according to 1:1:2. In particular, as deep learning methods require uniform inputs, we sort the length of each trace in the dataset and select trace length at 95% loci as model inputs to ensure that the vast majority of trace information can be fed into the model. As shown in TABLE 2, in each scenario, the DF method achieves better identification performance compared to the CUMUL method, which indicates that the deep learning approach works better in identifying the access behavior of Tor hidden service under MEEK

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

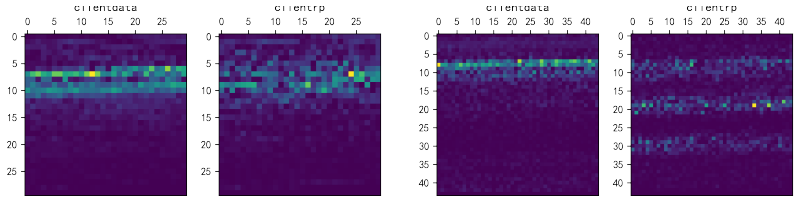
Table 3. Identification Results Based on Key Sequence

scenario	start_index	window_size	accuracy	precision	recall	f1
public	186	101	0.9379	0.9052	0.9608	0.9321
cdn77	309	156	0.9666	0.9568	0.9648	0.9607
fastly	278	114	0.9593	0.9461	0.9558	0.9509
stackpath	338	181	0.9652	0.9368	0.9756	0.9558

obfuscation scenario. Hence, we subsequently validate our proposed approach using the DF model.

5.2 Key Sequence Verification

To verify the existence of key sequence in Tor hidden service identification task, we put the whole trace as input with DF model and calculate the gradients value of each packet, aiming at finding out which part of the trace contributes the most to the identification task. As shown in Fig. 4, each square in the diagram rep-



(a) Public Scenario (b) CDN77 Scenario

(c) Fastly Scenario (d) Stackpath Scenario

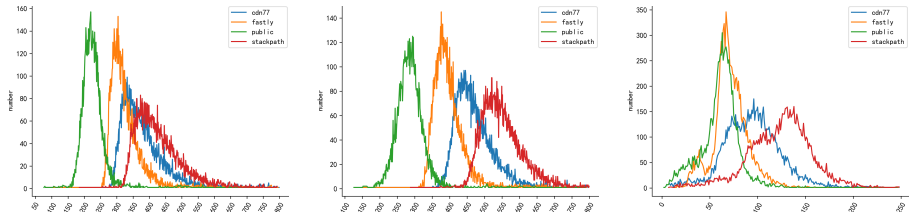
Fig. 4. Gradients Value of Each Packet to the identification Task in Four Scenarios

resents the gradient value of the packet for distinguishing public website access activity(**clientdata**) and Tor hidden service access activity(**clientrp**). Brighter colors indicate that the packet at that position has more weight to identification task. From the result, we draw the conclusion that there does exists a **key sequence** which contributes significantly for identifying Tor hidden service in each MEEK scenario. But the location and window size for key sequence varies for each scenario.

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

5.3 Key Sequence Location Distribution Observation

To find the location distribution of key sequence, for each scenario, we count the absolute position of the **establish_rendezvous** and **rendezvous2** cells, the time relative to the start of the trace(we use seconds as the unit), and the number of packets between the two cells. Then, we count the time of **establish_rendezvous** and **rendezvous2** in the cell logs relative to the connection and use that time as the start and end time point to locate the location of key sequence in TCP packet level. We note the index of to **establish_rendezvous** as the start position, and the index of **rendezvous2** as the end position.



(a) Start Index Distribution (b) End Index Distribution (c) Window Size Distribution

Fig. 5. Key Sequence Location and Window Size Distribution in TCP Sequences

As shown in Fig. 5, the location of the key sequence varies from each scenario while all accord with normal distribution. In the **public** scenario, the key sequence is the earliest, starting at the 184th TCP packet and ending at the 352ed TCP packet, with an interval of about 90 TCP packets; followed by the **fastly** scenario, where the key sequence starts at about the 277th TCP packet and ends at the 491st TCP packet, with an interval of about 104 TCP packets; followed by the **cdn77** scenario starts at about the 306th TCP packet and ends at the 629th TCP packet, with an interval of about 146 TCP packets, followed by the **stackpath** scenario, which starts at the 343rd TCP packet and ends at the 713th TCP packet, with an interval of about 176 TCP packets.

5.4 Classification with Key Sequence

In this section, we try to search the best value of the start point and window size for the DF classification method for four scenarios. We denote the search space as S*W, which S indicates the space of start TCP index and W indicates the window size. According to the observation described above, we set S belongs to [start index-5, start index+6] and W belongs to [window size-10, window size+11]. Then, by setting the radio of training, validation and testing as 1:1:2, we perform experiments with DF classification method iteratively by increase the S and W parameter with a step by 1 for four MEEK scenarios.

As shown in TABLE 3, in each scenario, only with key sequence can achieve 3%-4% better performance compared with the results of existing research work

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

Table 4. Results of Full TCP Sequence without Key Sequence

scenario	input_size	accuracy	precision	recall	f1
public	756	0.8883	0.8592	0.9288	0.8926
cdn77	1737	0.9181	0.872	0.98	0.9228
fastly	679	0.8871	0.8517	0.9374	0.8924
stackpath	3036	0.8612	0.7866	0.9914	0.8772

Table 5. Method Effectiveness in Timeliness and Resource Consumption

scenario	ADKSF	ANPKSF	ADC	ANPC	Time Saving	Memory Saving
public	92.74	101	137.74	511	0.33	0.8
cdn77	76.76	156	120.89	971	0.37	0.84
fastly	68.09	114	88.57	566	0.23	0.8
stackpath	70.15	181	105.13	1304	0.33	0.86

CUMUL [10] using full TCP sequence which are shown in TABLE 2, indicating that with the key TCP package sequences as input, a network level attacker can distinguish whether a user is accessing Tor hidden service with a high accuracy without decrypting the packets.

Besides, we remove the key sequence in each trace and use DF model for classification again, the results are shown in TABLE 4. Compared the identification results of using only key sequence, without key sequence, full TCP packet sequence as input, which results are shown in TABLE 3, TABLE 4 and TABLE 2 respectively, indicating that the key sequence proposed in this paper contributes tremendously in Tor hidden service access identification task. The advantage of using key sequence is that only a portion of the TCP sequence needs to be observed, rather than the entire trace, effectively improving the identification timeliness as well as reducing the cost of hardware resources.

Moreover, we calculate the TCP index at the end of the key sequence fragment and the average time duration, and compare it with the average number and time duration of entire access trace to obtain the percentage of time and memory consumption saved by our method. In the TABLE 5, **ADKSF** means average time duration of the key sequence, **ANPKSF** means average number of packets of the key sequence, **ADC** means average time duration of one access trace, **ANPC** means average number of packets of one access trace. As shown in TABLE 5, compared with the existing methods using all data as input, the timeliness is improved by 23%-37%, and the hardware resource consumption is reduced by 80%-86%, indicating that our method has high feasibility and good universality, which is much suitable for online identification scenario.

6 Conclusion

In this paper, we verify that there does exist a TCP package sequence contributes significantly to identify Tor hidden service access behavior under MEEK

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

scenario. Moreover, we present a novel approach to identify Tor hidden service access activity with key sequence under MEEK scenario. What's more, we perform comprehensive experiments and thorough analysis in both public and private MEEK scenarios, the results show that our method can identify the Tor hidden service access behavior in the early stage of the access procedure, effectively improving the identification timeliness as well as reducing the cost of hardware resources, indicating that our method has high feasibility and good universality, which is much suitable for online identification scenario.

The approach we present is to identify Tor hidden service access behavior under MEEK scenario, and is useful for supervisor to monitor the violation of criminal activity. However, it is true that the approach can identify which user uses Tor to access hidden service, but it is not clear which hidden service the user accesses. Therefore, the approach is a pre-work for recognizing hidden service for fine-grainedness.

The MEEK establish one connection from OP to MEEK server to transmit packets. In some cases, the user will access both hidden services and normal website at the same time. Whether the approach can identify the access behaviors which users access multiple hidden services or access boss hidden services and normal websites by using multiple tabs or not, and whether this approach can apply to the different status of network, needs further study.

Acknowledgements. This work is supported by the Fundamental Research Program(JCKY2019211B001), the Key Research and Development Program for Guangdong Province under grant(No.2019B010137003) and the Strategic Priority Research Program of the Chinese Academy of Sciences with No. XDC02030000.

References

1. Tor project, users – tor metrics. `https://metrics.torproject.org/userstats-relay-country.html?start=2021-11-20&end=2022-01-20&country=all&events=off`, accessed January 2022
2. Tor specification, `https://gitweb.torproject.org/torspec.git/tree/rend-spec-v2.txt`
3. Angel, Y., Winter, P.: obfs4 (the obfourscator), may 2014. URl: https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt
4. Christin, N.: Traveling the silk road: A measurement analysis of a large anonymous online marketplace. Archives of Neurology **2**(3), 293 (2012)
5. Fifield, D., Lan, C., Hynes, R., Wegmann, P., Paxson, V.: Blocking-resistant communication through domain fronting. Proc. Priv. Enhancing Technol. **2015**(2), 46–64 (2015)
6. Guan, Z., Gou, G., Guan, Y., Wang, B.: An empirical analysis of plugin-based tor traffic over ssh tunnel. In: MILCOM 2019-2019 IEEE Military Communications Conference (MILCOM). pp. 616–621. IEEE (2019)
7. Hayes, J., Danezis, G.: k-fingerprinting: A robust scalable website fingerprinting technique. In: USENIX Security Symposium. pp. 1187–1203 (2016)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40

8. Jansen, R., Juárez, M., Galvez, R., Elahi, T., Díaz, C.: Inside job: Applying traffic analysis to measure tor from within. In: 25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018. The Internet Society (2018)
9. Kwon, A., Alsabah, M., Lazar, D., Dacier, M., Devadas, S.: Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In: usenix security symposium. pp. 287–302 (2015)
10. Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., Wehrle, K.: Website fingerprinting at internet scale. In: NDSS (2016)
11. Panchenko, A., Mitseva, A., Henze, M., Lanze, F., Wehrle, K., Engel, T.: Analysis of fingerprinting techniques for tor hidden services. In: Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017. pp. 165–175. ACM (2017)
12. Rimmer, V., Preuveneers, D., Juarez, M., Van Goethem, T., Joosen, W.: Automated website fingerprinting through deep learning. In: Network & Distributed System Security Symposium (NDSS) (2018)
13. Shahbar, K., Zincir-Heywood, A.N.: An analysis of tor pluggable transports under adversarial conditions. In: 2017 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–7. IEEE (2017)
14. Sheffey, S., Aderholdt, F.: Improving meek with adversarial techniques. In: 9th {USENIX} Workshop on Free and Open Communications on the Internet ({FOCI} 19) (2019)
15. Sirinam, P., Imani, M., Juarez, M., Wright, M.: Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. pp. 1928–1943 (2018)
16. Soleimani, M.H.M., Mansoorizadeh, M., Nassiri, M.: Real-time identification of three tor pluggable transports using machine learning techniques. The Journal of Supercomputing **74**(10), 4910–4927 (2018)
17. Syverson, P., Dingledine, R., Mathewson, N.: Tor: The secondgeneration onion router. In: Usenix Security (2004)
18. Wang, L., Dyer, K.P., Akella, A., Ristenpart, T., Shrimpton, T.: Seeing through network-protocol obfuscation. In: Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security. pp. 57–69 (2015)
19. Wang, M., Li, Y., Wang, X., Liu, T., Shi, J., Chen, M.: 2ch-tcn: A website fingerprinting attack over tor using 2-channel temporal convolutional networks. In: 2020 IEEE Symposium on Computers and Communications (ISCC). pp. 1–7 (2020). https://doi.org/10.1109/ISCC50000.2020.9219717
20. Wang, T., Goldberg, I.: Improved website fingerprinting on tor. In: Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society. pp. 201–212. ACM (2013)
21. Xie, H., Wang, L., Yin, S., Zhao, H., Shentu, H.: Adaptive meek technology for anti-traffic analysis. In: 2020 International Conference on Networking and Network Applications (NaNA). pp. 102–107. IEEE (2020)
22. Yao, Z., Ge, J., Wu, Y., Zhang, X., Li, Q., Zhang, L., Zou, Z.: Meek-based tor traffic identification with hidden markov model. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). pp. 335–340. IEEE (2018)

ICCS Camera Ready Version 2022
To cite this paper please use the final published version:
DOI: 10.1007/978-3-031-08751-6_40