Towards Comprehensive Analysis of Tor Hidden Service Access Behavior Identification Under Obfs4 Scenario

Xuebin Wang[1,2], Zeyu Li[3], Wentao Huang[3], Meiqi Wang[1,2],
Jinqiao Shi[3], Yanyan Yang[4]
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing
[4] Department of Information Technology and Cyber Security, People's Public Security University of China
wangxuebin@iie.ac.cn, shijinqiao@bupt.edu.cn

ABSTRACT

In this paper, we present a novel approach to identify Tor hidden service access activity with key sequences under Obfs4 scenario. By calculating the key cell signals occurred during Tor hidden service access process, we get the start index and window size of the key TCP package sequence of traffic. In order to verify the effectiveness of this method, we perform comprehensive analysis under nine scenarios of different Obfs4 transmission model. We find through experimental results that there is a TCP package sequence window, which has a great contribution to identifying Tor hidden service access traffic. Only use the key TCP sequence as input, we can achieve more than 90% accuracy as well as recall in nine scenarios.

KEYWORDS

Tor,Hidden service,Traffic analysis,Obfs4

1 INTRODUCTION

Nowadays, Internet users tend to use privacy-enhancing tools to avoid censorship and hide their online activities. Tor[16], used by more than two million users every day[1], is one of the most popular systems, claiming 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[5]. Hence, it is necessary to identify Tor hidden service traffic.

In order to improve the availability and anonymity of the network, Tor introduces many methodologies to bypass censorship, such as Tor Bridge mechanisms, Meek[6] based obfuscation and Obfs4[4] based obfuscation. According to Tor project statistics[2], Obfs4 is the most popular bridges currently. The obfuscation introduced by Obfs4 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[9, 10, 13] and even distinguish the specific hidden service the user is accessing[7, 12, 14, 15, 18–20]. However, the detection granularity of prior works is in the unit of whole flow, which will lead to a relatively lower detection efficiency. Moreover, the obfuscation brings by Obfs4 protocol has not been taken into consideration.

In this paper we propose a novel approach to identify Tor hidden service access activity with key sequences under Obfs4 scenario. By calculating the key cell signals occurred during Tor hidden service access process, we get the start index and window size of the key sequence of traffic. We perform comprehensive analysis under nine scenarios of different Obfs4 transmission model and the experiment results indicates that there does exist a window of TCP package sequence contributes greatly to distinguish Tor hidden service access traffic. Only use the key TCP package sequence as input, we can achieve more than 90% accuracy as well as recall in nine scenarios. The contributions of this paper are listed as follows:

- As far as we know, as a network level attack, we are the first to use partial traffic data to classify Tor hidden service access activity under Obfs4 scenario, which helps to reduce the number of data packets required and effectively improve the real-time detection.
- We capture a large and practical dataset to validate our method. Besides, we make the dataset public[1], allowing researchers to replicate our results and evaluate new approaches in the future.
- We conducted classification experiments on the collected dataset. The experimental results show that by selecting the appropriate window size and start point on the Obfs4 traffic sequences, only the key sequence

[1]The dataset can be found on the following URL: https://github.com/Meiqiw/obfs4-mingan/.

ICEA 2021, December 18-19, 2021, Jinan, China

ACM ISBN 978-1-4503-9160-3/21/12…$15.00
https://doi.org/10.1145/3491396.3506532

needs to be input to the classifier, which can effectively identify Tor hidden service access activities under nine Obfs4 scenario.

Organization. The rest of the paper is organized as follows. We introduce the related work in section 2. In section 3, we illustrate the background on Tor and hidden service as well as Obfs4 protocol. In section 4, we describe the data collection and processing methodology. We next present, in section 5, our observations and experimental results under nine Obfs4 scenarios. Finally, we draw the conclusion in section 6.

2 RELATED WORK

In this section, we describe the related work about Obfs4 identification as well as WFP (Website Fingerprint Attack) on Tor hidden service.

Recent years, researchers pay more attention on traffic identification of Obfs4 protocol and propose many outstanding methods. He et al.[8] used coarse-grained filtering according to the randomness and timing of communication handshake messages and SVM algorithm to analyze and accurately identify Obfs4 traffic with over 99% accuracy. Liang et al.[11] put forward a multi-feature fusion method to identify Obfs4, which extracts random features, sequence features, length features of handshake packets, statistical features of communication packets, etc., and the accuracy can reach 93.82%. Wang et al.[17] proposed an method based on convolutional neural network, extracted the hidden features of the data packet through deep learning, achieving 99.9% accuracy.

Another research area related to our work is website fingerprinting attacks. Similar to our method, WFP also need to collect client-side traffic, and then use the traffic classification method to train the classifier. The difference is that WFP need to answer the question of which website the user has visited, while we need to answer whether the user has visited Tor's hidden service. Many outstanding works[7, 12, 14, 15, 18–20] 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.[10] 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.[9] control the middle nodes to execute the circuit fingerprinting attach, which proves that the attack from the middle nodes is as 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.[13] 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.

As far as we know, little research work pay attention on identification of Tor hidden service access behavior under Obf4 scenario. In our paper, we use Obfs4 TCP partial sequences and explore the effectiveness of our method through comprehensive analysis of nine different scenarios.

3 BACKGROUND

In this section, we will provide the background on Tor hidden services. Then, we describe the principle of Obfs4.

3.1 Tor Hidden Service

As a function of Tor, it can hide the location information for the service provider, thus can anonymous the responder. According to the protocol specification[3], the hidden service architecture is composed of five components in the Fig. 1:

- ***HS:***Hidden Server is the information provider on which various services such as web and mail can be deployed.
- ***OP:***Onion Proxy is the client of users accessing Tor network.
- ***RP:***Rendezvous Point is a Tor relay randomly selected by OP, which is responsible for forwarding traffic according to the behavior of OP while hiding the location.
- ***IP:***Introduction Point maintains a long-term circuit with HS and forwards the requests of OP to HS.
- ***HSDir:***Hidden Service Directory is a Tor relay with the flag **HSDir**, which is responsible for storing and searching hidden service information.

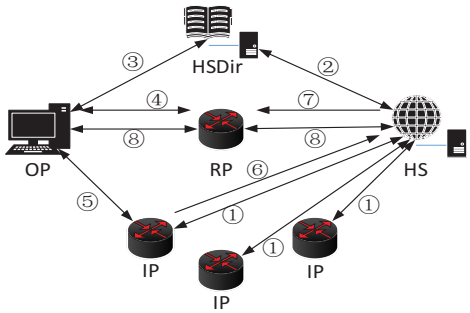


Figure 1: Hidden Service Architecture

We describe the steps of deploying a hidden service and users accessing the hidden service below.

- Firstly, HS selects three ORs as its IPs, and sends a *relay-establish-intro* cell to each IP to establish the circuit. After receiving the cell, IPs respond to a *relay-intro-established* cell with an empty payload to inform HS that the circuit has been established.
- After the circuit is established between HS and IPs, HS establish the circuit to the HSDir which is responsible for announcing the service descriptor. After that, the owner of HS can advertise the onion address in the form of 'z.onion' on the web.
- After a user receives an onion address, he uses OP to establish a circuit to the HSDir, and searches the service descriptor of the corresponding HS in the HSDir. OP can learn the IPs information corresponding to the HS in the service descriptor.
- OP randomly select a previously created circuit and take the last hop in the circuit as the RP. The OP sends a *relay-establish-rendezvous* cell carrying the rendezvous cookie generated by the client to the RP to establish the rendezvous circuit, and the RP responds to an empty *relay-rendezvous-established* cell to inform the OP that the establishment of the circuit is complete.
- OP then creates a circuit to IP, which is one of IPs in HS service descriptor. OP sends a *relay-introduce1* cell carrying the rendezvous cookie, the fingerprint of the RP and the hash of the public key of corresponding HS to IP.
- After receiving the *relay-introduce1* cell, IP finds the corresponding HS through the hash value of the public key. Then it sends the *relay-introduce2* cell containing the fingerprint of the RP and the rendezvous cookie to the HS.
- After receiving the *relay-introduce2* cell, HS decrypts the cell with its own private key to obtain the fingerprint of the RP and the rendezvous cookie. Then HS finds the corresponding RP by fingerprint, establishes a circuit to the RP, and sends a *relay-rendezvous1* cell containing the rendezvous cookie to it.
- Finally, RP binds two circuits with the same rendezvous cookie together, forwards cells from both ends, and sends a *relay-rendezvous2* cell to OP to inform OP that it can communicate with HS.

In this way, the communication between OP and HS can be anonymous. Fig. 2 shows the detailed process of cell exchange on OP side. It can be seen that there are obvious differences between activities related to hidden services and general access activities, and activities related to hidden services can be distinguished from other activities.

3.2 Obfs4 Potocol

Obfs4 is a protocol confusion layer for TCP protocol. Tor client uses Obfs4 plugin to communicate with Obfs4 server, and the Tor node running Obfs4 service plays the role of entry node as Obfs4 bridge node. The communication of

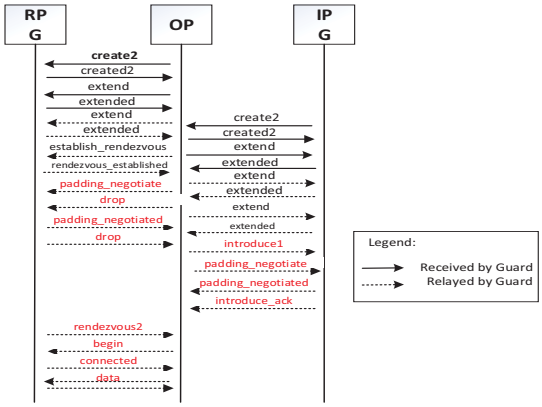


Figure 2: Client-HS Circuit Cell Exchanges

Obfs4 protocol consists of handshake phase and encrypted data transmission phase.

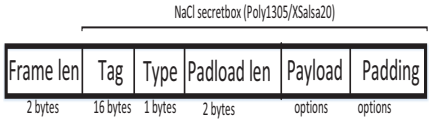


Figure 3: The structure of the Obfs4 frame

In the data transmission phase, Obfs4 will pack the data from Tor process into the format of frame as shown in Fig. 3, and may pad the data. The frame len field is not encrypted, but the true value is confused by the key, and the other fields are encrypted by the key. Obfs4 has three iat-mode modes when sending data packets, which have values of 0, 1 and 2. When iat-mode=0, Obfs4 does not process the data packet except padding, and directly delivers it to TCP. When iat-mode=1, Obfs4 splits the data packet with a maximum of 1448 bytes after padding, when iat-mode=2, Obfs4 splits the data packet with a random size and pads the data; in these two modes, Obfs4 randomly delays for a period of time after sending a data fragment.Because the iat-modes of the client and the server are independent of each other when sending data packets, there are nine iat-mode combinations.

4 DATA COLLECTION AND PROCESSING

4.1 Data Collection

Pluggable transmission Obfs4 provides 3 different modes for obfuscating data. According to our observation, the client and server side can use different modes, which means that a total of 9 different transmission modes are generated during the communication between the client and the server side, we call this as 9 scenarios. ***Obfs4 Server Setup*** Firstly, we rented vps to build private Obfs4 servers and use different processes to build Obfs4 bridges with different iatmode. We set up the vps so that they could provide tor and Obfs4proxy services, then modified the tor configuration file torrc to command tor to use Obfs4, configuring the port for Obfs4proxy

and the type of iatmode used, depending on what was required. We then restarted tor and verified the validity of the pluggable transport server built. For example, the following configuration (1) controls tor's use of Obfs4's iatmode2.

$$\begin{aligned} &BridgeRelay\ 1 \\ &ServerTransportPlugin\ obfs4\ exec\ /usr/bin/obfs4proxy \\ &ServerTransportListenAddr\ obfs4\ 0.0.0.0:51439 \\ &ServerTransportOptions\ obfs4\ iat-mode=2 \end{aligned} \tag{1}$$

Obfs4 Client Setup For each scenario, we deployed 3 groups of dockers (10 per group, 30 in total) for browsing trace collection, each group having the same list of target urls. We use multiple dockers to achieve diversity in the data we collect, in order to be able to better support our proposed approach. For each docker, we used the private Obfs4 bridge built above as the first hop of the tor, so that the server-side iatmode could be diversified. Similar to the server-side configuration, the tor configuration file torrc in the client-side docker is configured as (2)

$$\begin{aligned} &BridgeRelay\ 1 \\ &ClientTransportPlugin\ obfs4\ exec\ /usr/bin/obfs4proxy \\ &Bridge\ obfs4\ <obfs4bridgeinfo>\ iat-mode=2 \end{aligned} \tag{2}$$

Meanwhile, we modified the Tor source code, so that we could obtain control information such as Tor's creation of connections and construction of circuits, as well as data information such as streams and sequences of letter elements. This information is recorded in the Log files when collecting data to show the real activity of Tor while performing web browsing. We have used the Tor logs to count connections, circuits, streams and primitives, in order to find the differences between Tor accessing hidden services and ordinary websites, which is of crucial importance to our proposed approach.

For each trace, open the client's Tor and Obfs4proxy processes before starting the capture, and close the client's Tor and Obfs4proxy processes after a successful capture. The server-side Tor and Obfs4proxy processes keep running. Therefore, in our experiment, when a client visits a website, new connections and circuits will always be established. That is, every access activity of the client uses new unused circuits. It is worth noting that the traffic trace obtained by a visit may contain multiple connections. As a network-level adversary, we split these connections based on the IP address and port number of the sender and receiver. Next, we select the website to be visited to form a website visit list. For Tor's hidden service, we randomly select 15,000 websites based on the .onion search engine http://www.ahmia.fi/; for other services, we use Alexa's top 15,000 websites. Similar to the previous work, after collecting the data, we filtered out invalid traces and outliers caused by browser or Selenium driver timeouts or crashes. In the end, we got a large number of traces as shown in TABLE 1, and each trace has one corresponding Tor notice log file. It should be pointed out that the scenarios

Table 1: DATA COLLECTION OF NINE SCENARIOS

	Website	WebsiteTrace	Onion	OnionTrace
c0_s0	7754	20622	8267	22255
c0_s1	11165	27741	11235	27825
c0_s2	7340	11926	9117	15669
c1_s0	8439	13539	12625	48530
c1_s1	11048	26588	11161	26917
c1_s2	3794	10614	3692	10678
c2_s0	8386	13373	12650	48747
c2_s1	10963	26441	11216	27027
c2_s2	3854	11676	3772	12009

of `client0_server0` means that the iatmode on the client and server is 0 and 0 side respectively.

4.2 Data Extraction and Processing

Since Tor uses fixed size cells for data transfer, there is a lot of work[14, 15, 20] that points to the possibility of representing tor traffic sequences as a sequence of [+1, -1]. Since pluggable transport Obfs4 uses TCP for transmission, as well as confusing and delaying packets in different iatmodes, the approach in the traditional work is not applicable. We have modified the above method to represent Obfs4 traffic as a sequence of [+package_size, -package_size], where a positive sign indicates that the client sends a TCP packet to the server side, a negative sign indicates that the server sends a TCP packet to the client side, and package_size indicates the payload size of the packet. In the experiment, we truncated the input sequences to a fixed length, and filled the sequences less than this length with 0.

As mentioned above, we log the Tor cells while the client visits a website. Therefore, every traffic trace we collect has a corresponding cell log file. In this way, our dataset consists of two types, **traffic traces** and **cell records**. The detail processing procedure are described as follows respectively.

Traffic Trace Processing Since the traffic trace may contain multiple connections, we cut each visit traffic trace into flows according to four-meta tuple(srcIP, srcPort, destIP, destPort), ensuring each flow contains and only contains one connection. Then we label each connection with one of the five types of labels based on the cell log corresponding to each connection. We will describe the labels in detail in **Cell Record Processing** part. Finally, we use the `Tshark`[2] (version 1.12.1) tool to extract the TCP packet length of each connection and generate a TCP packet sequence.

Cell Record Processing We parse the notice log files and extract helpful information including connection creation time, connectionID, circuitID, cell command and direction, etc. From the Tor hidden service communication principle introduced above, we can see that different circuits will be established during access and have different circuit purposes. Therefore, we distinguish the circuits according to the ID number of the circuit, and arrange the cells of each circuit in the order of the timestamp. Since we can read the command of each cell in the log file, we can get the purpose of every

[2]https://www.wireshark.org/docs/man-pages/tshark.html

circuit. We divide circuit into five categories: **create-fast**, meaning that this circuit is built for downloading consensus documents at bootstrapping process, **client-data**, meaning that this circuit is built for access non-hidden service related data, **client-ip, client-rp, client-hsdir**, meaning that this circuit is hidden service related, and **others**. Next, we select circuits belongs to the same connection and put corresponding cells together, generating the cell sequence of one specific connection. At last, we label each connection according to the circuits categories multiplexed in the same connection.

DATASET Obfs4-MingAn After completing the above operation, we eventually obtained *Obfs4-MingAn*, consisting of 9 subsets: each subset contains 10,000 instances, corresponding to one transmission scenario. For each subsets, it contains two parts: TCP data and cell data. TCP data contains both direction and time information of each TCP packet, and cell data contains cell direction, timestamp and circuit purpose.

5 EVALUATION AND DISCUSSION

5.1 Comparison of Different Classification Methods

The most relative work to us is Panchenko et al.[13], it uses the whole traffic as input to extract features called CUMUL[12]. Additionally, DF[15] achieves better performance in WFP task. Hence, we reproduce the classification methods of prior work on our dataset **Obfs4-MingAn**, finding the state-of-the-art as the method we use in this paper. In order to check the robustness and accuracy, by setting the radio of training, validation and testing as 1:1:2, we increase the TCP sequences from 90 to 360 with a step by 10 iteratively for DF while the whole traffic as input for CUMUL.

As shown in Fig. 4, the accuracy of DF increases with the length of input packages and achieves stable at the 300. Besides, as the iat-mode of both client and server changes from 0 to 2, the performance of DF decreases correspondingly, indicating that the obfuscation introduced by Obfs4 protocol has a significant impact on the result. What's more, we can also draw the conclusion that the iat-mode server side has more impact than that of client side. As shown in TABLE 2, the DF classification method performs better than that of CUMUL in all Obfs4 scenarios with excellent robustness. Thus, we take DF as our classification method used in this paper.

5.2 Select Key TCP Sequences

Based on our dataset, with the cell command name and timestamp, we can clearly notice the activity the Tor instance is doing. As for hidden service related activity, **establish_rendezvous** and **rendezvous2** cell are import functional cells during the OP-HS connection construction procedure, indicating the start and success signals correspondingly. Firstly, we perform statistics on all the record cell sequences find the start time point and end time point. In detail, we statistic the absolute position of establish_rendezvous and rendezvous2 in the unit of one connection. Then, we find the

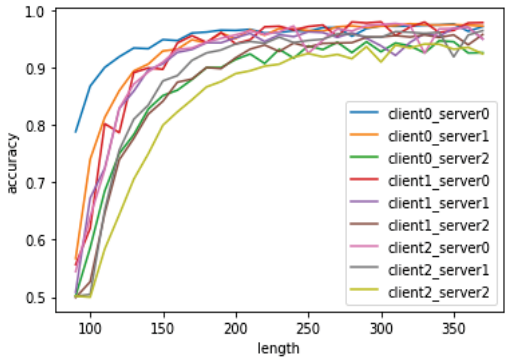
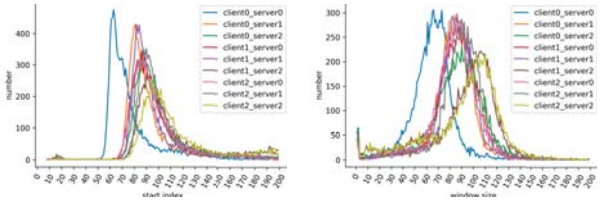


Figure 4: DF Classification Result

Table 2: CUMUL VS DF on NINE SCENARIOS

	accuracy		precision		recall	
	df	cumul	df	cumul	df	cumul
c0_s0	0.9651	0.9148	0.9552	0.9163	0.976	0.9148
c0_s1	0.9688	0.9175	0.96	0.9186	0.9784	0.9175
c0_s2	0.9262	0.8818	0.8876	0.8832	0.976	0.8818
c1_s0	0.9664	0.9111	0.944	0.9131	0.9916	0.9111
c1_s1	0.9547	0.9157	0.9302	0.9165	0.9832	0.9157
c1_s2	0.9436	0.8968	0.9452	0.8998	0.9418	0.8968
c2_s0	0.9733	0.913	0.9768	0.9151	0.9696	0.913
c2_s1	0.9443	0.9152	0.9173	0.9157	0.9766	0.9152
c2_s2	0.9187	0.8974	0.9042	0.8986	0.9366	0.8974

corresponding TCP package index and window size with the help of start time point and end time point respectively.



(a) Start TCP Package Index (b) Sequence Window Size

Figure 5: Key Sequence Metric Statistics

As shown in Fig. 5, each Obfs4 scenario has its start index and window size respectively, which accord with normal distribution. As the iat-mode change from 0 to 2, the distribution scatters from left to right to some extend, the start index as well as the window size increase correspondingly. Taking client0_server0 scenario as example, most of the start index begins from and the window size less than 100. Thus, we get the corresponding key TCP sequences identifying the Tor hidden service access activity. From the statistic results, we draw the conclusion that there does exist a fragment of cell sequences contribute significantly on distinguishing hidden service related activity and that it is possible to filter hidden service related activity only with the key TCP package sequences.

5.3 Identification by Key TCP Sequences

In this section, we try to search the best value of the start point and window size for the DF classification method for

Table 3: DF WITH KEY TCP SEQUENCES

iatmode	startindex	windowsize	accuracy	precision	recall
c0_s0	60	89	0.9456	0.9527	0.9378
c0_s1	73	110	0.9552	0.9559	0.9544
c0_s2	78	121	0.9182	0.9089	0.9296
c1_s0	74	114	0.9656	0.9767	0.954
c1_s1	78	113	0.9506	0.9407	0.9618
c1_s2	82	130	0.9359	0.9434	0.9274
c2_s0	74	110	0.9539	0.9475	0.961
c2_s1	82	120	0.9461	0.9559	0.9354
c2_s2	84	134	0.9153	0.9249	0.904

both scenarios. We refer 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-1, start_index+1] and W belongs to [window_size-1, window_size+1]. 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 nine Obfs4 scenarios.

The final results are illustrated in TABLE 3, for all Obfs4 scenarios, with proper start index and window size, it is possible to identify Tor hidden service access activity as efficient as prior work while significantly reduce the resource cost. All scenarios achieves better performance than prior work, 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. Because of using much less traffic data, our approach can identify timely without waiting the end of this access activity, which can be applied on online identification scenario.

6 CONCLUSION

In this paper we propose a novel approach to identify Tor hidden service access activity with key sequences under Obfs4 scenario, which significantly reduces resource consumption and is as accuracy as prior work. By calculating the key cell signals occurred during Tor hidden service access process, we get the start index and window size of the key TCP package sequence of traffic. In order to verify the effectiveness of this method, we perform comprehensive analysis under nine scenarios of different Obfs4 transmission model. The experiment results show that there does exist a window of TCP package sequence contributes greatly to identify Tor hidden service access traffic. Moreover, our approach performs as well as state-of-the-art methods with respect to classification accuracy as well as recall in nine scenarios.

Acknowledgements. This work is supported by the Key Research and Development Program for Guangdong Province under grant(No.2019B010137003) and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040400.

REFERENCES

[1] [n. d.]. Tor Project, Users – Tor Metrics. https://metrics.torproject.org/userstats-relay-country.html?start=2021-08-16&end=2021-11-14&country=all&events=off. ([n. d.]). Accessed November 2021.

[2] [n. d.]. Tor Project, Users – Tor Metrics. https://metrics.torproject.org/userstats-bridge-transport.html?start=2021-08-16&end=2021-11-14&transport=!⟨OR⟩&transport=obfs4&transport=meek. ([n. d.]). Accessed November 2021.

[3] [n. d.]. Tor Specification. ([n. d.]). https://gitweb.torproject.org/torspec.git/tree/rend-spec-v2.txt

[4] Yawning Angel and Philipp Winter. [n. d.]. obfs4 (the obfourscator), May 2014. *URl: https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt* ([n. d.]).

[5] Nicolas Christin. 2012. Traveling the Silk Road: A measurement analysis of a large anonymous online marketplace. *Archives of Neurology* 2, 3 (2012), 293.

[6] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proc. Priv. Enhancing Technol.* 2015, 2 (2015), 46–64.

[7] Jamie Hayes and George Danezis. 2016. k-fingerprinting: A Robust Scalable Website Fingerprinting Technique.. In *USENIX Security Symposium*. 1187–1203.

[8] Yongzhong He, Liping Hu, and Rui Gao. 2019. Detection of Tor Traffic Hiding Under Obfs4 Protocol Based on Two-Level Filtering. In *2019 2nd International Conference on Data Intelligence and Security (ICDIS)*.

[9] Rob Jansen, Marc Juárez, Rafa Galvez, Tariq Elahi, and Claudia Díaz. 2018. 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.

[10] Albert Kwon, Mashael Alsabah, David Lazar, Marc Dacier, and Srinivas Devadas. 2015. Circuit fingerprinting attacks: passive deanonymization of tor hidden services. In *usenix security symposium*. 287–302.

[11] Di Liang and Yongzhong He. 2020. Obfs4 Traffic Identification Based on Multiple-feature Fusion. In *2020 IEEE International Conference on Power, Intelligent Computing and Systems (ICPICS)*.

[12] Andriy Panchenko, Fabian Lanze, Jan Pennekamp, Thomas Engel, Andreas Zinnen, Martin Henze, and Klaus Wehrle. 2016. Website Fingerprinting at Internet Scale.. In *NDSS*.

[13] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, Klaus Wehrle, and Thomas Engel. 2017. 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*. ACM, 165–175.

[14] Vera Rimmer, Davy Preuveneers, Marc Juarez, Tom Van Goethem, and Wouter Joosen. 2018. Automated website fingerprinting through deep learning. In *Network & Distributed System Security Symposium (NDSS)*.

[15] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. 2018. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 1928–1943.

[16] Paul Syverson, R Dingledine, and N Mathewson. 2004. Tor: The secondgeneration onion router. In *Usenix Security*.

[17] Han Wang, Xiangyang Luo, and Yuchen Sun. 2020. An Obfs-based Tor Anonymous Communication Anline Identification Method. In *2020 6th International Conference on Big Data and Information Analytics (BigDIA)*. IEEE, 361–366.

[18] Meiqi Wang, Yanzeng Li, Xuebin Wang, Tingwen Liu, Jinqiao Shi, and Muqian Chen. 2020. 2ch-TCN: A Website Fingerprinting Attack over Tor Using 2-channel Temporal Convolutional Networks. In *2020 IEEE Symposium on Computers and Communications (ISCC)*. 1–7. https://doi.org/10.1109/ISCC50000.2020.9219717

[19] Tao Wang, Xiang Cai, Rishab Nithyanand, Rob Johnson, and Ian Goldberg. 2014. Effective Attacks and Provable Defenses for Website Fingerprinting.. In *USENIX Security Symposium*. 143–157.

[20] Tao Wang and Ian Goldberg. 2013. Improved website fingerprinting on tor. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society*. ACM, 201–212.