# Efficient Classification of Darknet Access Activity with Partial Traffic

Xuebin Wang[1,2], Meiqi Wang[1,2], Jinqiao Shi[3](✉), Zeyu Li[3], Kexin Zou[3], and Binxing Fang[1,4]

[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, China
shijinqiao@bupt.edu.cn
[4] Institute of Electronic and Information Engineering of UESTC in Guangdong, Dongguan, China

**Abstract.** In this paper we propose a novel approach to classify darknet-access traffic with only partial traffic data, which significantly reduces resource consumption and is as accuracy as prior work. Besides, in order to keep up with the users' real access activity, we simulate new and old user by simply whether delete the cached consensus document before each access and apply our approach. The experiment results confirm that there does exist a window of cell sequence contributes greatly to distinguish darknet-access traffic. With the window size 75 and the start point 67, we can achieve 95.97% accuracy for new user access scenario. Similarly, with the window size 85 and the start point 44, we achieve 94.43% accuracy for old user access scenario.

**Keywords:** Tor · Hidden service · Traffic analysis

## 1 Introduction

Tor [14], a low-latency anonymity network, has emerged as an important privacy-enhancing tool protecting users' online privacy, i.e. hiding the users' IP address while communicating on the Internet. Nowadays, with more than two million users daily [1], Tor is considered to be one of the most popular anonymous communication systems consisting of nearly 7000 volunteer-operated relays, which are run from all around the world.

Besides protecting client's privacy, Tor also allows servers to operate anonymously by offering hidden services (HSs). HSs allow users, in particular those living in oppressive regimes, e.g., human right activists and whistle-blowers, to bypass censorship and to exercise freedom of speech by publishing and offering access to sensitive content without the fear of being targeted, arrested or forced to shut down. As a result, many sensitive contents are hosted and only accessed through HSs, forming a deep-dark cyberspace for criminals [4]. Hence, it is necessary to comprehensively evaluate the level of protection provided by this novel anonymity mechanism.

Unfortunately, attackers can classify whether a user is accessing hidden service and even infer which specific hidden service a user has visited. With malicious nodes controlled, attacker performs circuit fingerprinting attack can easily distinguish hidden service related circuits both at the guard [8] and middle [6] position of a circuit, the attack significantly depends on the number of nodes controlled. What's more, a local observer which eavesdrops traffic between the sender and the first anonymization relay node, can distinguish whether a user is accessing hidden service [11] and guess the user's destination without decryption, called Website Fingerprinting attack (WF attack) [5,10,12,13,15–17]. With the help of machine-learning or deep learning models, prior works treat each whole traffic trace as input to extract features, which is not suitable for online classification scenario.

In this paper we propose a novel approach to classify darknet-access traffic with only partial traffic data, which significantly reduces resource consumption and is as accuracy as prior work. Besides, in order to keep up with the users' real access activity, we simulate new and old user by simply whether delete the cached consensus data before each access. Moreover, we collect direct cell logs by modifying Tor source code to record the basic information of each cell, and use the direct cell logs as ground truth to analysis the nuance between darknet-access and general access activity. By digging into the access process thoroughly, we find that there exists a window of cell sequence contributes greatly to distinguish darknet-access related traffic. With the window size 75 and the start point 67, we can achieve 95.97% accuracy for new user access scenario. With the window size 85 and the start point 44, we can achieve 94.43% accuracy for old user access scenario.

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 darknet-access activity, which is also much more practical and applicable in online manner.
- In order to verify our method, we capture a large and practical dataset by simulating new and old user access activity. Besides, we make the generated dataset publicly available[1], allowing researchers to replicate our results and systematically evaluate new approaches in the future.
- Based on the dataset we collected, we use the direct cell logs to determine the proper window size and the start point of the darknet-access activity. Then, we transform traffic traces into cell sequences and conduct activity classification experiments, the experimental results verify that it does work that with a proper window size and start point can effectively distinguish darknet-access activity from general access activity.

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

**Organization.** The rest of the paper is organized as follows. In Sect. 2, we illustrate the background on Tor and hidden service design as well as the attacker threat model. In Sect. 3, we describe the data collection and processing methodology, generating the dataset for analysis. We next present, in Sect. 4, our observations and experiment results regarding differences between darknet-access activity and general access activity. We introduce the related work in Sect. 5 and the conclusion in Sect. 6.

## 2 Background

In this section, we will provide the necessary background on Tor as well as the functionality of the Tor hidden services. Then, we describe the threat model of our attack.

### 2.1 Tor

A user starts the anonymous journey by simply unzipping the Tor browser bundle, which contains the Onion Proxy (OP) and a customized Firefox browser. The OP performs as an bridge between users' applications and the Tor network. Before user sends his application data over the Tor network, the OP must learn about Tor's relays, Onion Routers, by downloading the network consensus document from directory servers. And then select three relays: an entry guard, middle and exit node, creating circuits incrementally and interactively. The OP encapsulate application data into 514-byte fixed-size cells as its communication data unit, forwarded though the created circuit hop by hop. Tor builds circuits hop by hop like an onion, and the details of the circuit construction process as follow. Firstly, the OP sends a *create2* cell to establish the circuit with the guard relay, which responds with a *created2* cell. Secondly, the OP sends an *extend2* command cell to the guard relay, which parses the cell and correspondingly sends a create cell to the middle relay node to establish the circuit on behalf of the user, thus a tunnel between the user and the middle relay has been created. Finally, the OP sends another *extend2* command cell to the middle relay through the tunnel just created, causing the middle relay sends a create cell to the exit node correspondingly. And then the circuit between the OP and the exit relay has been created, then a *begin* cell is relayed the exit node building a TCP connection to the final destination. Figure 1 demonstrates the 3 hop circuit construction process as well as the cells exchanged between OP and the guard relay for general Tor connections.

The TCP connection between each hop of a Tor circuit is secured with TLS. Moreover, Tor multiplexes circuits within a single TCP connection. Precisely, An OP-OR TCP connection multiplexes all circuits from the same user while an OR-OR TCP connection multiplexes circuits for various users simultaneously. An ISP level attacker who monitors the OP-OR TCP connection can not distinguish which TCP packet belongs to which circuit as all circuits exists in the same one TCP connection.

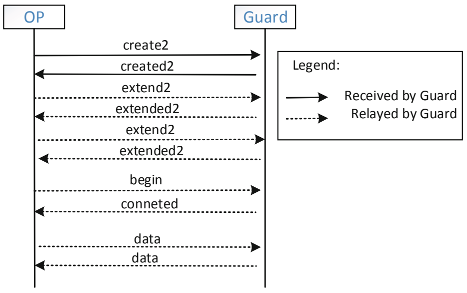


**Fig. 1.** Circuit construction details

## 2.2 Hidden Service Components

Hidden service was introduced in 2004 as a feature of Tor, providing anonymity of responders by hidden the location information while offering service. According to the protocol specification [2], the hidden service architecture consists of the following five components, as shown in Fig. 2:

- ***HS:*** Hidden Server is the information provider which hosts various services, such as WEB, MAIL.
- ***OP:*** User accesses the Tor network by running Tor client instance name as Onion Proxy (OP).
- ***RP:*** Rendezvous Point is the Tor relay which is chosen by the OP randomly, forwarding traffic on behave of OP while concealing the location at the same time.
- ***IP:*** Introduction Point maintains a long-term circuit and forwards the requests from clients to the hidden service.
- ***HSDir:*** Hidden Service Directory is a Tor relay which has the flag **HSDir**, acting as a database for storing and retrieving hidden service information.

Next, we describe the steps to set up a hidden service in Tor and establish a connection to it.

- Firstly, the HS chooses three onion routers as its IPs, and then builds circuits to each IPs by sending a *relay-establish-intro* cell respectively. Upon receiving such a cell, the IPs send back *relay-intro-established* cell with an empty payload to inform that the circuits have been successfully established.
- After establishing circuits to IPs, the HS builds a circuit to the HSDirs chosen to advertise the service descriptor, the cell sequences exchanged is shown in Fig. 3. After that, the HS owner can advertise the onion address in the surface Web with the form z.onion.
- After receiving an onion address, the Tor client creates circuits to HSDirs responsible for the specific HS and retrieves HS descriptor from which the client learns the information about IPs.

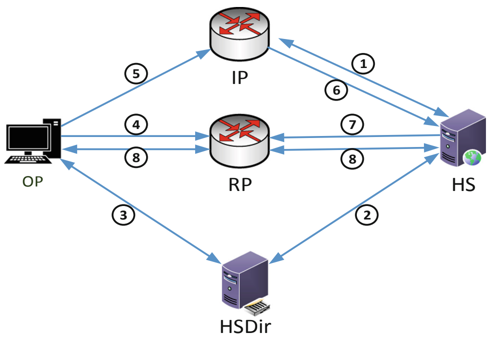


**Fig. 2.** Hidden service architecture

- The Client randomly chooses a pre-created circuit and picks the last hop as the RP and builds a circuit to that onion router by sending a *relay-establish-rendezvous* cell which carries the rendezvous cookie, and the RP replies with an empty *relay-rendezvous-established* cell, indicating that the rendezvous circuit has been successfully built.
- The Client then builds separate circuits to one of the IPs extracted from the HS descriptor. The Client sends a *relay-introduce1* cell contains the rendezvous cookie, the fingerprint of RP and the hash of the public key of the HS along the introduction circuit.
- Once the IP receiving *relay-introduce1* cell, it sends the *relay-introduce2* cell to corresponding HS, according to the hash of the public key. The *relay-introduce2* cell also contains the rendezvous cookie generated by Client and the fingerprint of RP.
- Upon receiving the *relay-introduce2*, HS decrypts it with the private key and extract rendezvous cookie and RP's fingerprint. Then, the HS extends a circuit to RP according to the fingerprint and sends a *relay-rendezvous1* cell containing rendezvous cookie.
- At last, RP binds two circuits which have the same rendezvous cookie, so as to deliver relay cells from each of the two circuits to the other, and sends a *relay-rendezvous2* cell to Client which denotes the beginning of communication between Client and HS.

In this way, the OP and HS communicates successfully without any leakage of their anonymity. Figure 4 demonstrates the cell exchange process in detail. From the description introduced above, there does exist significant differences between hidden service related activity and general access activity, indicating that it is possible to distinguish hidden service related activity from others.

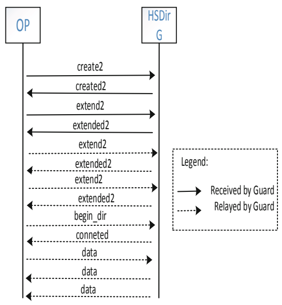


**Fig. 3.** Client-HSDir circuit cell exchanges

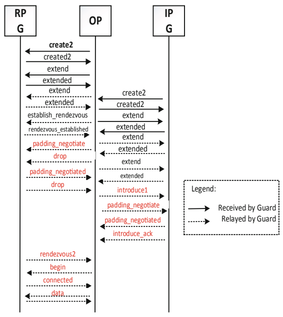


**Fig. 4.** Client-HS circuit cell exchanges

## 2.3 Threat Model

In this work, we assume a network level attacker which is: local, meaning that he has access only to the connection between the OP and the guard relay node, and passive, i.e., he can only collect the network packets and can not delay, drop, modify or decrypt even them. Precisely, we assume the attacker is the Internet Service Providers (ISP). Figure 5 illustrates the attack scenario: the user access both general web service and hidden service over the Tor network and intercepts the traffic between the user and the Tor network. We assume that the attacker knows the user's identity and only aims at distinguishing the darknet-access activity from numerous connections.

Within this attack scenario, we make several assumptions about the attacker goals and capabilities.

*Traffic Parsing:* The ISP attacker has access to all OP-OR TCP connections built by huge amount users concurrently, and able to record the traffic packet meta-data of the both direction, including timestamp, srcIP, srcPort, dstIP, dstPort, packetSize, direction. As what has been mentioned above, the ISP attacker can only distinguish each OP-OR TCP connection but can not identify each circuit multiplexed in one TCP connection while a node level attacker does.

*Goals:* In this work, we assume the ISP level attacker only focus on identifying that a user is connected to hidden service (darknet-access activity) within huge amount TCP connections as effective as possible, or even the real time scenario. Identifying which website the user surfs is out of the coverage of this work.

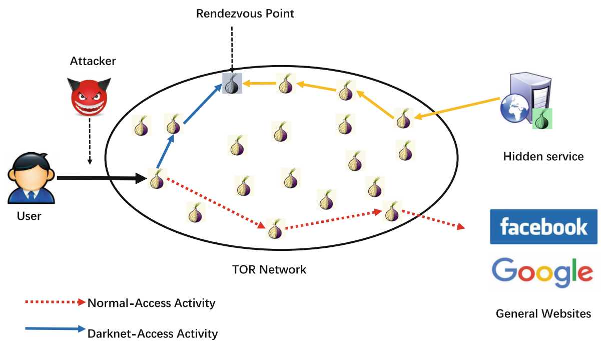


**Fig. 5.** The threat model

## 3 Data Collection and Processing

In this section, we propose our new data collection method and describe our experimental setup. Then we describe the data extraction procedure as well as giving a brief introduction of our dataset.

### 3.1 Data Collection

In order to simulate users' access activity over the Tor network more realistically, we define two scenarios: delete Tor cached documents and not delete. The former scenario aims at simulating a user who use Tor network for the first time, while the latter scenario for simulating a user who use Tor network access the Web just now. The reason is that new user needs download Tor consensus document before building circuits while the old user not needs, making totally difference in the number connection built as well as packet pattern transferred.

We use a distributed setup, utilizing 3 Virtual Machines (VMs) on cloud environment provided by Vultr[2]. These virtual machines are located in different countries including Singapore, France and the United States, so as to ensure the diversity of traces. Each VM is configured with 2 CPUs and 8 GB of RAM. On each VM, 10 docker instances are deployed, and each docker with a separate Tor process (version 0.4.4.6). To access the Tor network, we use `Selenium`[3] (version 3.12.0) to control headless browser `Firefox` (version 60.0.2), utilizing a SOCKS5 proxy listened by Tor. We recorded the traces of web pages leveraging `tcpdump`[4]. Web pages are given 120 s to load, and upon loading the page, it was left open for an additional 10 s, after which the browser is closed and the Tor process is killed. As for new user scenario, the cached data in the DataDirectory such as cached consensus, server descriptors, is deleted automatically each time. 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.

[2] http://vultr.com.
[3] http://www.seleniumhq.org/.
[4] http://www.tcpdump.org/.

**Table 1.** Data collection for both scenarios

| | Website | WebsiteTrace | Onion | OnionTrace |
|---|---|---|---|---|
| Delete cache | 8155 | 13504 | 8755 | 14594 |
| Not delete | 7754 | 20622 | 8267 | 22255 |

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. What to be mentioned is that one trace may contains multi connections, we split all connections and build dataset for our attack as the network level attacker does. While recording the traffic trace, we also record the connection creation, circuit construction, stream info, cell sequences into the notice log file by modifying Tor source code, aiming at showing light on the real activity Tor instance occurs during darknet-access as well general-access activity. By performing statistics on the connection, circuit, stream as well as cell, we reveal the difference between darknet-access activity and general-access activity in two scenarios described above. Those statistic results have theoretical significance for the attack approach we proposed.

Following our data collection method, we use Alexa Top websites and Tor hidden services[5] as our target website for both scenarios, each with 10,000 websites. Similar to the previous work, after data collection, we filtered out invalid traces and outliers, which caused by timeout or crash of the browser or Selenium driver. Eventually, we obtained huge amount of traces as shown in Table 1, each trace accomplished with one corresponding notice log file.

### 3.2 Data Extraction and Processing

In general, as in many previous work [12,13,17], we represent the data as a sequence of [+1, −1], where each +1 or −1 represents a cell, which is the most basic communication unit of Tor, and the sign indicates whether the direction of the cell is from the client to the Tor entry node or vice versa. As a result, an input instance of our model is a series of 1 and −1. In the experiment, we truncated the input sequences to a fixed length, and filled the sequences less than this length with 0. As asserted in [12], neural networks generally work with real numbers from the compact interval [−1, 1] due to the nature of the mathematical operations they perform. Moreover, by providing the input data in such a format, we avoid having to rescale and/or normalize the values and thus mitigate a possible information loss coupled with the preprocessing step.

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

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, TLS record sequences as well as cell sequences. The detail processing procedure described as follows respectively.

***Cell Record Processing.*** By parsing the notice log file, many basic information about the connection, circuit, 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 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**, those three are hidden service related, and others. 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 shown in Fig. 6, at the application layer, Tor embeds the encrypted data into a fixed-size (514-byte) packet, which is called a cell. And the cell is further embedded into the TLS record. Multiple cells can be grouped into a single TLS record. Finally, in the transport layer, TLS records are typically fragmented into multiple TCP packets, the size of which is limited by the Maximum Transmission Unit (MTU). Note that several TLS records can be within a single TCP packet. As for collected traffic traces, our process performs as follows: Firstly, we cut each visit traffic trace file into multi flows according to four-meta tuple (srcIP, srcPort, destIP, destPort), ensuring each flow contains and only contains one connection. Secondly, 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 and TLS record sequences, with the help of `Tshark`[6] (version 1.12.1), an industrial grade widely used tool for network traffic analysis. At last, we extract cell sequences from the TLS record sequences following the method proposed by [17], thus translating each connection traffic packet traces into cell sequences tagged with corresponding categories. A slight difference from [17] is that we divide the length of the TLS record by 514 instead of 512, because with the upgrade of the Tor protocol, the length of the cell has changed from a fixed 512 bytes to 514 bytes. Therefore, we believe that this treatment is much more closer to the real situation.

***DATASET MingAn21.*** After completing the above operation, we eventually obtained *MingAn21*, consisting of two subsets: (i) 15,000 instances of hidden service related, general website related and others each, corresponding

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

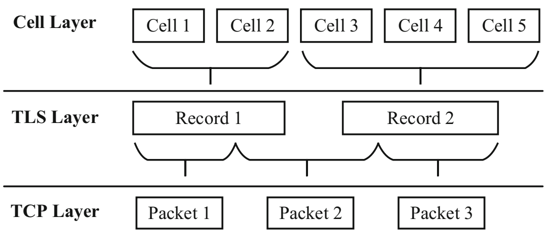


**Fig. 6.** Layers of data transport in Tor

to the delete cache scenario (ii) 10,000 instances of hidden service related and general website related each, corresponding to the not delete cache scenario. Our dataset contains both direction and time information of each cell.

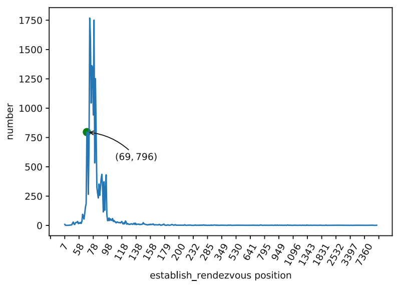
## 4 Evaluation and Discussion

In this section, we firstly perform location statistic of hidden service related cell commands with the real cell record parsed from the notice log file, revealing the possible position of **establish_rendezvous** and **rendezvous2** cell in one single connection. Next, we provide an evaluation of the different classification methods of prior work with proper length of cell sequences, finding the state-of-the-art as the one we use in this paper. At last, we perform iterative experiments to learn the best choice of the start point and window size to perform our attack.

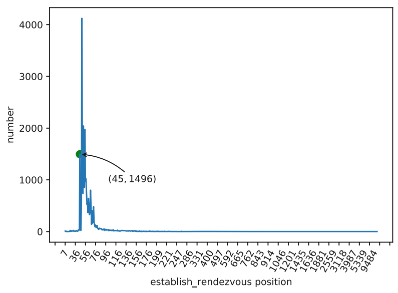
### 4.1 Position Distribution Observation

According to our data collection method, we record two type of data for every connection: raw traffic packets and real cell sequences. We record some basic cell data including cell command name, cell direction and timestamp. With the cell command name, 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. In order to have a clear understanding the position of those two functional cells within one single connection, we perform statistics on all the record cell sequences parsed from the notice log file for both two scenarios. In detail, we statistic the absolute position of establish_rendezvous and rendezvous2 as well as the interval distance in the unit of one connection.

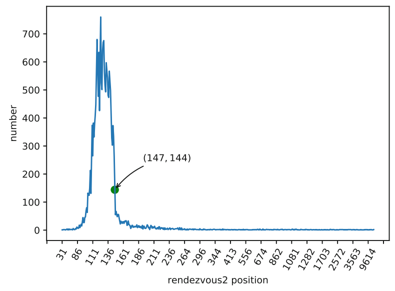
As shown in Fig. 7, in most cases establish_rendezvous cells are send after 69 and receive rendezvous2 cell before 147 with a window size of 72 in the delete cache document scenario. However, in not delete cache document scenario, most establish_rendezvous cells are send after 45 and receive rendezvous2 cell before 130 with a window size of 73. It is oblivious that, in most cases, hidden service related functional cell signals of not delete document scenario occur much earlier



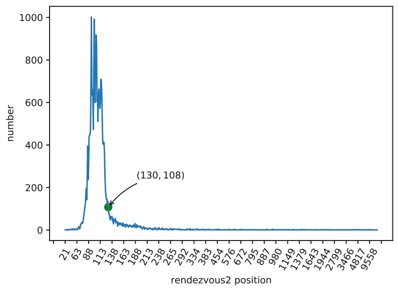
(a) Establish_Rendezvous Cell Position Distribution of Delete Scenario



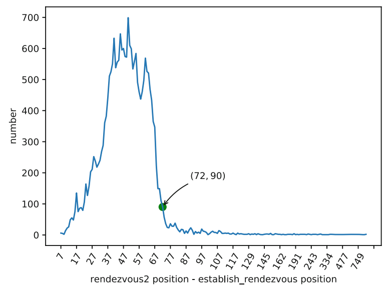
(b) Establish_Rendezvous Cell Position Distribution of Not Delete Scenario



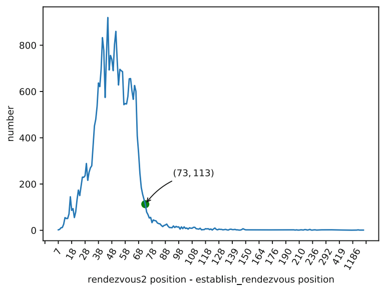
(c) Rendezvous2 Cell Position Distribution of Delete Scenario



(d) Rendezvous2 Cell Position Distribution of Not Delete Scenario



(e) Interval Distance Distribution of Delete Scenario



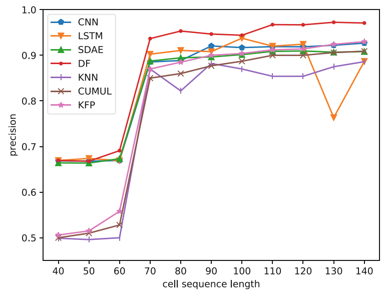
(f) Interval Distance Distribution of Not Delete Scenario

**Fig. 7.** Position statistic results on both scenarios

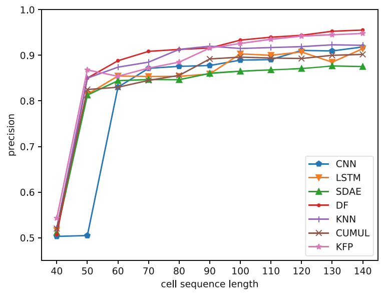
than that of delete cache document scenario. From the statistic results, we draw the conclusion that there does exist a fragment of cell sequences contribute significantly on distinguishing hidden service related activity and that it is possible to filter hidden service related activity only with partial cell sequences.

### 4.2 Comparison of Different Classification Methods

In this subsection, we reproduce the classification methods of prior work on our dataset **MingAn21**, finding the state-of-the-art as the method we use in this paper. In order to check the robustness and accuracy, we increase the cell sequences from 40 to 140 with a step by 10 iteratively by setting the radio of training, validation and testing as 1:1:2. With the above setting, we perform our experiments on both scenarios with different classification methods, including CNN [12], LSTM [12], SDAE [12], DF [13], k-NN [16], CUMUL [10] and k-FP [5].



(a) Delete Cache Document



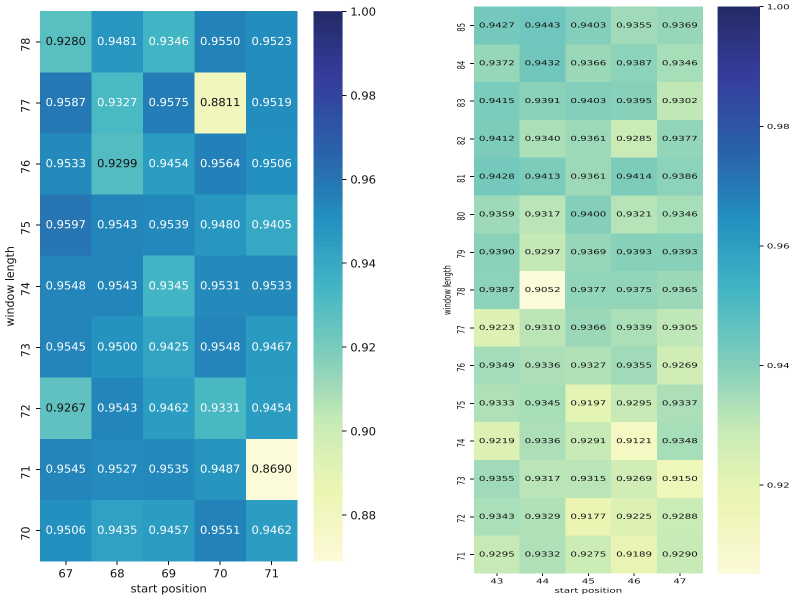
(b) Not Delete Cache Document

**Fig. 8.** Different classification on both scenarios

As shown in Fig. 8, the DF classification method performs the best with excellent robustness as well as accuracy for both scenarios. As for the delete cache document scenario, with the cell sequences length at 40, deep learning related classification methods achieve much better accuracy than that of machine learning methods do. As the increase of the cell sequence length more than 70, the performance of DF method increases rapidly and keeps stable. As for the not delete cache document scenario, KFP and DF both achieve better accuracy and stability as the cell sequence increases. As DF achieves better performance in both scenarios, we take DF as our classification method used in this paper.

### 4.3 Classification with Partial Cell Fragment

In this section, we try to search the best value of the start point and window size for the DF classification method for both scenarios. We refer the search space as $S * W$, which S indicates the space of start point and W indicates the window size. According to the observation described above, we set S belongs to [start_point-2, start_point, start_point+2] and W belongs to [window_size-2, end_point-start_point]. In delete cache document scenario, S belongs to $[67, 68, 69, 70, 71]$ and W belongs to $[70, 71, 72, 73, 74, 75, 76, 77, 78]$. And in not delete cache document scenario, S belongs to $[43, 44, 45, 46, 47]$ and W belongs to $[71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85]$. 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. The final results are illustrated in Fig. 9, with the window size 75 and the start point 67, we can achieve 95.97% accuracy for delete cache document scenario. With the window size 85 and the start point 44, we can achieve 94.43% accuracy for not delete cache document scenario. The result verifies that it is possible to classify hidden service access and general service access activity as efficient as prior work while significantly reduce the resource cost. With 75 and 85 cell sequences respectively, in both scenarios, a network level attacker can distinguish whether a user is accessing hidden service or general service with a high accuracy without decrypting the packets.



(a) Delete Cache Document (b) Not Delete Cache Document

**Fig. 9.** Different classification on both scenarios

The most relative work to us is Panchenko et al. [11], in contrast to this work, our approach only use 75 or 85 cell sequences of each trace while achieving as good performance as the prior work. Moreover, in terms of training time, with much less data for training, our approach is also much effective than the prior work.

## 5 Related Work

Many attacks have been proposed to challenge the security of Tor hidden services. Most of these threat models assume that the attacker is active, that is, the adversary has the ability to modify the monitored traffic, or influence the routing of relays. For example, Biliukov et al. [3] proposed that when a malicious RP receives *relay-introduce1* sent by the hidden service, it sends a message to the hidden service consisting of 50 padding ucells. This signal allows another malicious OR along the circuit from the hidden service to the RP to identify the hidden service or its entry guard on the circuit. Another example, Jansen et al. [7] proposed a memory-based DoS attack. The attacker identifies and disables the entry node of the target HS, thereby forcing the server to choose a new guard. Chen et al. [9] proposed a novel approach discovering the hidden service or its entry guard in a parallel manner by embedding numerous hidden services identification into rendezvous cookie.

In contrast, the adversary in our threat model is passive, that is, the attacker can only record the traffic he monitors, but cannot modify or drop them. It is worth mentioning that a study similar to our research method is Website Fingerprint Attack, which uses traffic classification to identify which website Tor users have visited. The difference is that we need to answer whether Tor users have accessed hidden services, or just used Tor to access a website that can be accessed through a normal browser. Although we have different granularities of traffic classification, many outstanding works [5,10,12,13,15–17] in the WFP field are also of great reference value.

Unfortunately, only few prior studies pay attention to how to distinguish whether a user is accessing a hidden service. According to the ability and location of attackers, they can be divided into node level attackers and network level attackers. With malicious entry node controlled, node level attackers record the circuit creation signals as well as cell sequences silently and traffic data in a passive manner. With the information collected, the node level attacker perform traffic analysis attacks to infer whether the user has accessed a specific hidden service. However, network level attackers located on the path between user and the first anonymization node, can collect and only collect the traffic data with much more widely visibility.

***Node Level Attacker.*** Kwon et al. [8] showed that hidden services' traffic can be distinguished from regular websites with more than 90% accuracy from a malicious entry node perspective. Recently, Jansen et al. [6] performed circuit fingerprinting attach from the middle relay position, demonstrating that traffic fingerprinting techniques are as effective as prior works shown from a guard relay perspective. However, the result of this kind of attack significantly depends on the number of nodes attacker controlled.

***Network Level Attacker.*** Hayes and Danezis [5] find that the onion sites can be discriminated from other regular web pages with 85% true positive rate and only 0.02% false positive from a dataset of 100,000 sites. Panchenko et al. [11] use machine learning methods to distinguish hidden service related traffics accurately and scales well, with a precision more that 0.9 and a recall at least 0.8. With the help of machine-learning or deep learning models, prior works treat each whole traffic trace as input to extract features, which is not suitable for online classification scenario.

In our paper, we innovatively examine the existence of the fragment of cell sequences and explore the effectiveness of only use partial traffic data for distinguishing darknet-access activity from general access activity.

## 6 Conclusion

In this paper we have analyzed the susceptibility of darknet-access activity to the traffic analysis attack. To this end, we proposed a novel approach to classify darknet-access traffic with only partial traffic data, which significantly reduces resource consumption and is as accuracy as prior work. In order to verify the effectiveness and applicable for practical scenario, we conduct experiments both on new and old user scenario. The results depict that there does exist a window of cell sequences contribute greatly to distinguish darknet-access traffic. Moreover, our approach performs as well as state-of-the-art methods with respect to classification accuracy. Thus, with only partial traffic data we can distinguish darknet-access activity effectively with much less resources, which can be applied in online classification scenario.

**Acknowledgements.** This work is supported by the Key Research and Development Program for Guangdong Province under grant (No. 2019B010137003), Fundamental Research program of National Defence (JCKY2019211B001) and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02040400.

## References

1. Tor project, users - tor metrics. https://metrics.torproject.org/userstats-relay-country.html?start=2021-02-25&end=2021-05-26&country=all&events=off. Accessed May 2021
2. Tor specification. https://gitweb.torproject.org/torspec.git/tree/rend-spec-v2.txt
3. Biryukov, A., Pustogarov, I., Weinmann, R.P.: Trawling for tor hidden services: detection, measurement, deanonymization. In: 2013 IEEE Symposium on Security and Privacy, pp. 80–94 (2013). https://doi.org/10.1109/SP.2013.15
4. Christin, N.: Traveling the silk road: a measurement analysis of a large anonymous online marketplace. Arch. Neurol. **2**(3), 293 (2012)
5. Hayes, J., Danezis, G.: k-fingerprinting: a robust scalable website fingerprinting technique. In: USENIX Security Symposium, pp. 1187–1203 (2016)
6. 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, 18–21 February 2018. The Internet Society (2018)
7. Jansen, R., Tschorsch, F., Johnson, A., Scheuermann, B.: The sniper attack: anonymously deanonymizing and disabling the tor network. In: 21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, 23–26 February 2014 (2014)
8. 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)

9. Muqian, C., Wang, X., Liu, T., Shi, J., Yin, Z., Fang, B.: SignalCookie: discovering guard relays of hidden services in parallel, pp. 1–7, June 2019. https://doi.org/10.1109/ISCC47284.2019.8969639
10. Panchenko, A., et al.: 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, 30 October–3 November 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 and Distributed System Security Symposium (NDSS) (2018)
13. 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)
14. Syverson, P., Dingledine, R., Mathewson, N.: Tor: the second-generation onion router. In: USENIX Security (2004)
15. 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
16. Wang, T., Cai, X., Nithyanand, R., Johnson, R., Goldberg, I.: Effective attacks and provable defenses for website fingerprinting. In: USENIX Security Symposium, pp. 143–157 (2014)
17. Wang, T., Goldberg, I.: Improved website fingerprinting on tor. In: Proceedings of the 12th ACM Workshop on Privacy in the Electronic Society, pp. 201–212. ACM (2013)