NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE

DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

MASTER THESIS

Tor Circuit Fingerprinting: Attacks and Defenses

Theodoros D. Polyzos

Supervisor: Konstantinos Chatzikokolakis, Associate professor

ATHENS

February 2021

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Tor Circuit Fingerprinting: Επιθέσεις και Άμυνες

Θεόδωρος Δ. Πολύζος

Επιβλέπων: Κωνσταντίνος Χατζηκοκολάκης, Αναπληρωτής Καθηγητής

ΑΘΗΝΑ

Φεβρουάριος 2021

MASTER THESIS

Tor Circuit Fingerprinting: Attacks and Defenses

Theodoros D. Polyzos

S.N: CS2180011

February 2021

Διπλωματική Εργασία

Tor Circuit Fingerprinting: Επιθέσεις και Άμυνες

Θεόδωρος Δ. Πολύζος

Α.Μ.: CS2180011

Φεβρουάριος 2021

ABSTRACT

This thesis studies tor anonymity network circuit fingerprinting problem focusing on the hidden services scope. Online anonymity and privacy has been based on confusing the adversary by reating indistinguishable network elements. Tor is the largest and most-used deployed anonymity system, designed against realistic modern adversaries. In recent researches, it has been proved that fingerprint Tor’s circuits is possible, simply by capturing and analyzing traffic traces. We study the circuit fingerprinting problem, isolating it from website fingerprinting, and revisit previous findings in this model, showing that accurate attacks are possible even when the application-layer traffic is identical. We then proceed to incrementally create defenses against circuit fingerprinting, using a generic adaptive padding framework for Tor based on WTF-PAD. We present a simple but high-latency defense which can effectively hide onion service circuits. We thoroughly evaluate the defense, discovering new subtle fingerprints, but also showing the effectiveness of the defense.

SUBJECT AREA: Network Anonymity and Privacy

KEYWORDS: anonymity, tor, onion protocol, onion relay, circuit fingerprinting, WTF-PAD framework

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία μελετάμε το δίκτυο ανωνυμίας tor και την διαρροή ιδιωτικότητας μέσω fingerprinting attacks εστιάζοντας στο κομμάτι των hidden services που παρέχει το tor. Η online ανωνυμία και ιδιωτικότητα επιτυγχάνεται με την παρουσία θορύβου στο δίκτυο, αναγκάζοντας τον επιτιθέμενο να μην είναι σε θέση να ξεχωρίσει και κατηγοριοποιήσει την κίνηση του δικτύου όταν και εφόσον είναι σε θέση να την παρατηρήσει. Το tor είναι το μεγαλύτερο και πιο χρησιμοποιούμενο σύστημα ανωνυμίας, σχεδιασμένο κατά ρεαλιστικών σύγχρονων αντιπάλων. Σε πρόσφατες έρευνες έχει παρουσιαστεί η επιτυχία fingerprinting επιθέσεων ενάντια στο tor μέσω την ανάλυσης της ροής των δεδομένων που παράγονται κατά την διάρκεια της ανώνυμης επικοινωνίας. Μελετώντας προηγούμενες έρευνες, θα παρουσιάσουμε ότι τέτοιες επιθέσεις είναι εφικτές ακόμα και όταν το traffic σε application layer επίπεδο είναι ομοιόμορφο χωρίς ιδιαίτερα patterns που μπορούν να οδηγήσουν τον επιτιθέμενο σε άμεσα συμπεράσματα. Σε συνδυασμό με την ανάλυση των επιθέσεων προτείνουμε κάποια defenses που χρησιμοποιούν adaptive padded circuits στην ροή της επικοινωνίας και βασίζονται στο ήδη υλοποιημένο defense framework του tor. Παρουσιάζουμε μια απλή αλλά high-latency άμυνα και παρουσιάζουμε την επίδοσή της άμυνας. Τέλος, δοκιμάζοντας και αξιολογώντας την απόδοση των defenses, περιγράφουμε ένα νέο fingerprinting πρόβλημα που δημιουργείται, σε συνδυασμό ωστόσο με την απόδοση και την αποτελεσματικότητα των defenses μας.

Θεματική Περιοχή: Διαδικτυακή ανωνυμία και ιδιωτικότητα
Λέξεις Κλειδιά: Ανωνυμία, tor, onion protocol, onion relay, circuit fingerprinting, WTF-PAD framework

CONTENTS

FIGURE CATALOG

TABLES CATALOG

1. INTRODUCTION

The Tor network is one of the largest deployed anonymity networks, consisting of thousands of volunteer-run relays and millions of users. In addition to sender anonymity, Tor's hidden services allow for receiver anonymity. Many sensitive services are only accessible through Tor with examples including human rights and whistleblowing organizations (such as Wikileaks) and tools for anonymous messaging (such as TorChat and Bitmessage). Apart from hidden services, many non-hidden services(such as Facebook and DuckDuckGo) have recently started providing hidden versions of their websites to provide stronger anonymity.

1.1 Tor Background

At its basic mode, Tor allows its clients to achieve anonymity when connecting to TCP/IP services. A client can use the Tor network simply by installing the Tor browser bundle, which includes a modified Firefox browser and the Onion Proxy (OP). The OP acts as an interface between client and the Tor network. Before user start communicating and send traffic through the network, the OP builds circuits interactively and incrementally, typically done by extending the circuit to three hops: an **entry guard**, **middle**, and **exit node**. Tor uses 512-byte fixed-sized cells as its communication data unit for exchanging control information between onion relays and for relaying users' data.

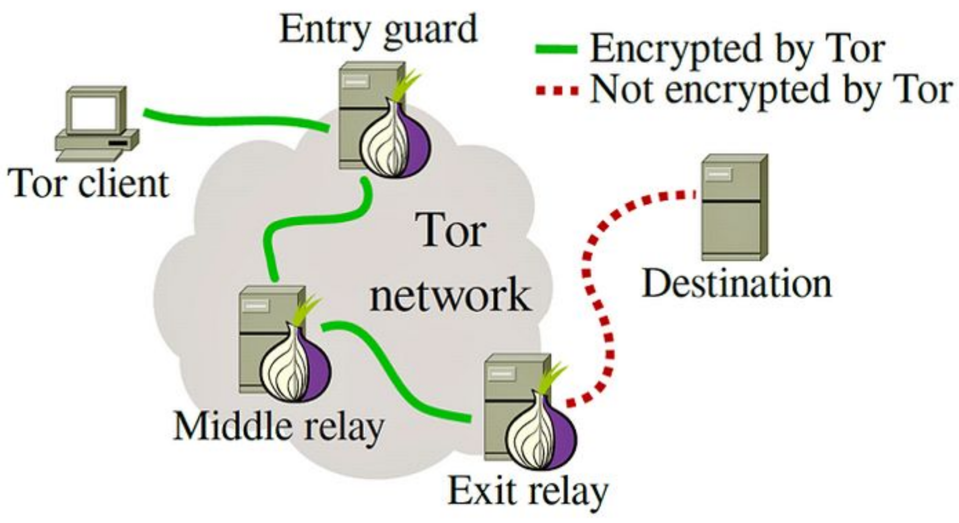


Figure 1: How does Tor work?

Once a circuit is established, the user creates streams through the circuit by instructing the exit node to connect to the desired external Internet destinations. Each pair of relays communicate over a single onion routing connection that is built using the Transmission Control Protocol (TCP). The application layer protocols rely on this underlying TCP connection to guarantee reliability and to deliver application data in the right order, called cells, between each relay. Streams are multiplexed over circuits, meaning that multiple streams can use the same circuits. Moreover, like streams, circuits are multiplexed over connections. In the case of a website, the user asks the exit relay to connect to the destination website and the communication between user and the website happens through the resulting circuit. We call such a circuit that connects to the normal web an exit circuit.

1.2 Tor Onion Services

In addition to client anonymity, Tor allows operators to set up anonymous servers, typically called onion services. A server can serve content behind a user's Onion Proxy without revealing his identity or location. This is achieved by routing communication between the client and the onion service through a rendezvous point which connects anonymous circuits from the client and the server.

In order to be reachable by clients, server's Onion Proxy (OP) must generate a hidden service descriptor. Firstly, the hidden server's OP chooses randomly an Onion Relay (OR) to use as its Introduction Point (IP) and creates a circuit to it. Server's user then sends an established intro message that contains the user's public key (the client can select more than one IP). If the OR accepts, it sends back an intro established to user’s OP. Now a signed descriptor (containing a timestamp, information about the IP, and its public key) is being created and a descriptor-id based on the public key hash and validity duration is being computed. The descriptor is then published to the hash ring formed by the hidden service directories, which are the ORs that have been flagged by the network as “HSDir”. Finally, the hidden service's URL xyz.onion is being advertised (delivered from the server's public key). At this point, the hidden service's communication path has been set up successfully.

When a client, connects to an onion service, client first creates a directory circuit (we call it HSDir circuit) and fetches the descriptor document of the onion service. The descriptor contains the introduction points of the onion service which are also relays on the Tor network. Following this step, client selects a random relay to use as rendezvous point (RP) and asks it to become the rendezvous point for the current session. After the rendezvous point has been created, client's OP builds a circuit to server's IP and sends information regarding the address of RP. Server's IP then relay's this information back to the server and simultaneously an introduce ack towards the client. At this point, server's OP builds a circuit towards client's RP. Finally, client and server have established a communication path through the 6 relays between them and data can be exchanged. On *Figures* 2,3,4, tor client to hidden service communication is described.

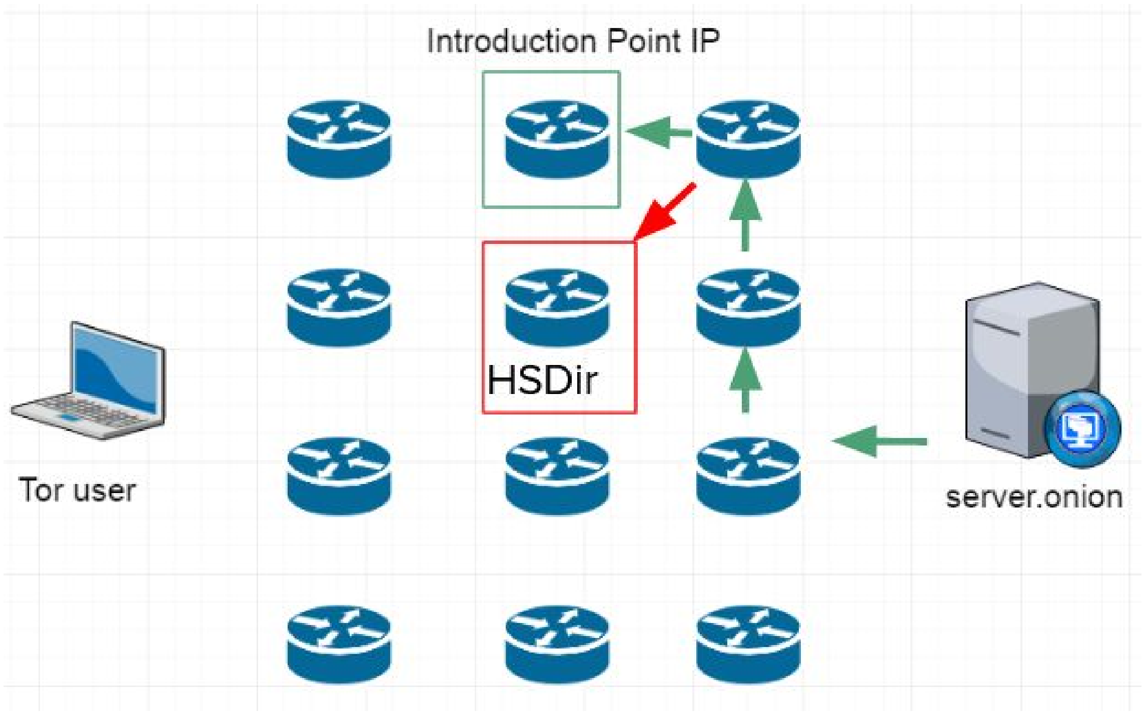


Figure 2: Server picks a random OR to use as his Introduction Point. Then publish the information to the HSDir descriptor flagged relay.

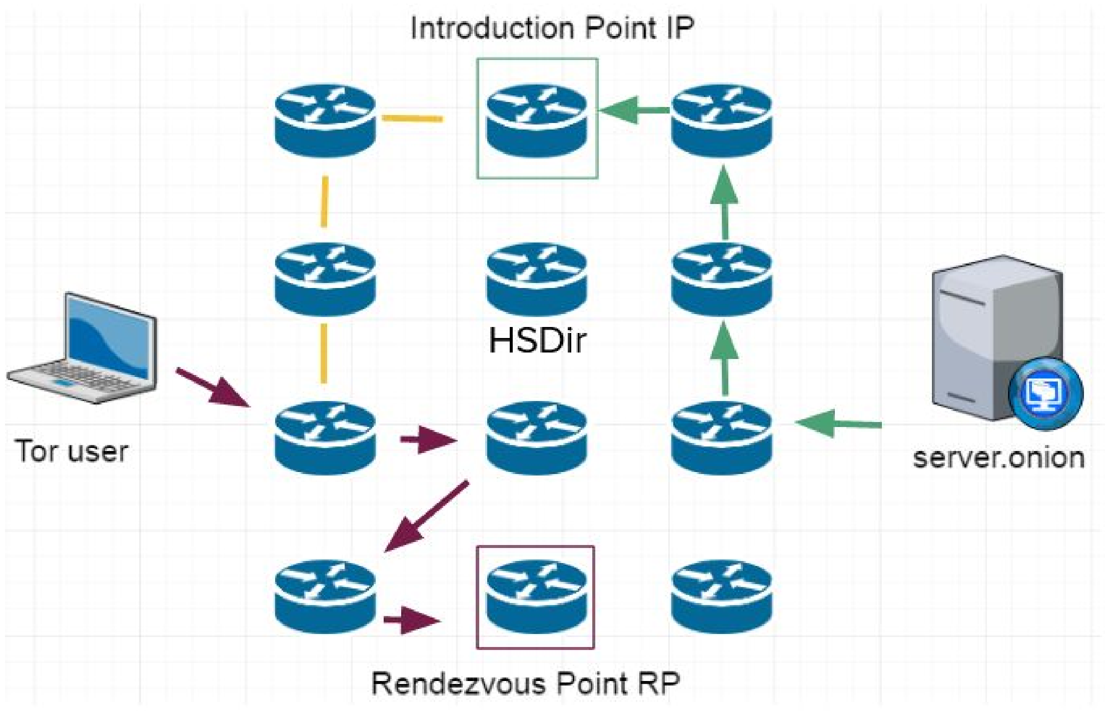


Figure 3: User picks a random OR to use as a Rendezvous Point RP and sends the location of RP to the server through the IP.

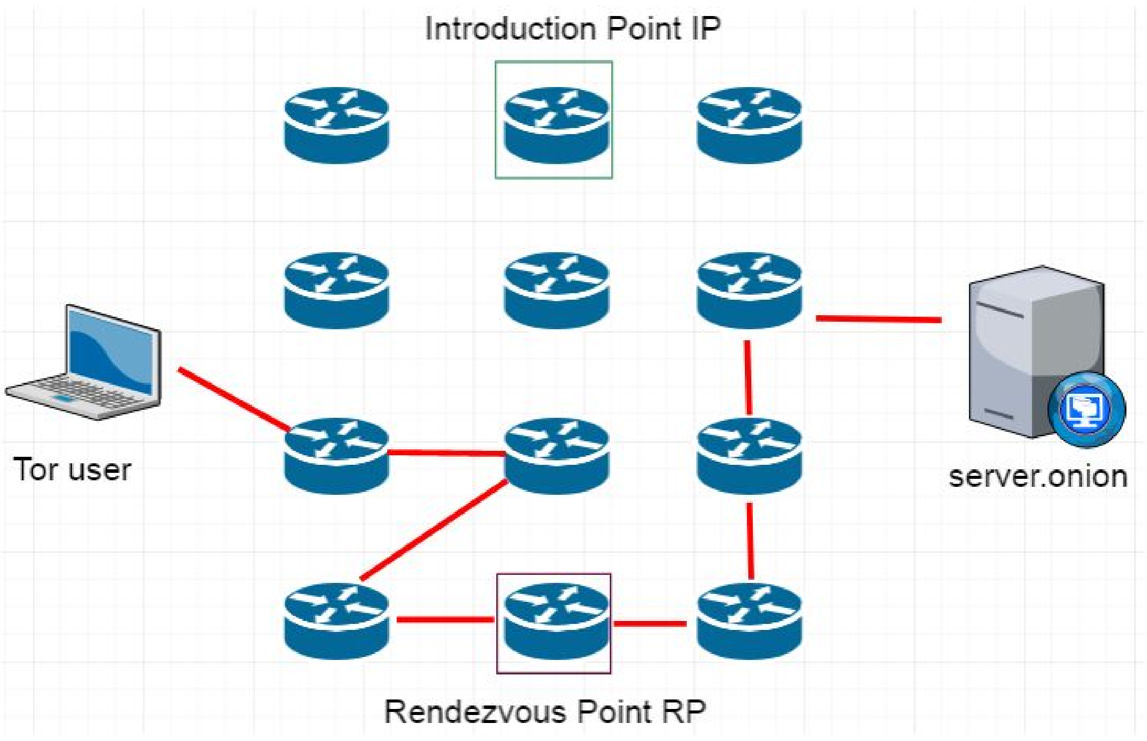


Figure 4: Hidden Service learns user’s RP, connects and data transfer begins.

1.3 Website Fingerprinting VS Circuit Fingerprinting

Website Fingerprinting is a traffic analysis attack that enables a *local passive adversary* to identify the website being visited by a user, by observing the user's encrypted traffic. This can be typically achieved by recognizing patterns in user's network packet exchange, as well as the time characteristics of the packets that also give birth to patterns that can be used as fingerprints.

Tor clients use circuits for anonymous communication, which are multi-hop paths through the Tor network carrying the application data. There are various types of circuits, some of them for navigating the regular Internet, others for fetching Tor directory information, connecting to onion services or simply for measurements and testing. Although all traffic is encrypted, it is still possible for certain types of adversaries (such as relay and LAN adversaries) to distinguish Tor circuit types from each other using a wide array of metadata and distinguishers. These attacks are known as *circuit fingerprinting*. In circuit fingerprinting, an adversary typically emula-tes tor network conditions of the monitored clients. The attacker deployes a client who visits through the live network the websites that are going to be used for classi-fication. During this process, the attacker collects the network traces of the clients. Then, he trains a supervised classifier with many identifying features of a network traffic of a website, such as the sequences of packets, size of the packets, and inter-packet timings. This model built from the samples is then used for classification of the user's live network traces.

Website fingerprinting can be considered the "*easiest problem*" for an adversary because of the "boundless" world space of web pages and due to the fact that there is no centralized way to make websites look similar to each other. On the other hand, *circuit fingerprinting* is a "harder" problem for the adversary because the Tor's onion websites world is significantly smaller and due to the fact that Tor protocol can be controlled and shaped easily through upgrades.

Leaving any of those two problems unsolved allows an adversary to leverage them to solve the other problem. For example, an adversary who can solve the *website fingerprinting* problem can use its distinguisher to make the *circuit fingerprinting* easier by classifying all websites found as a specific type of circuit. Similarly, an adversary who can solve the *circuit fingerprinting* problem can use its distinguisher to make the *website fingerprinting* easier, by classifying non-website circuits as irrelevant for the purposes of website fingerprinting.

1.4 Adversary Model

For the purposes of this research, we assume that the attacker is either a LAN or a relay adversary. The attacker needs to be able to extract circuit level related features out of the client's traffic and in particular to distinguish cells from each other.

A *relay adversary* can be part of Tor network by setting up relays as part of a *Sybil attack.* In the Sybil *attack*, a reputation system is subverted by creating a large number of identities, and using them to gain a disproportionately large influence in the network. This allows the attacker to spectate or even influence the user's traffic.

During this work, the main adversary class we are concerned about is the guard adversary since they are in position to know the location of the user and hence have more power if they manage to carry out a circuit fingerprinting attack. A guard adversary can perform traffic analysis attacks on the user's traffic since they have full visibility on the Tor circuit and the cells transferred within. While they cannot see the types or contents of Tor cells, they can still see the exact number of incoming and outgoing cells.

A *LAN adversary* can be the network administrator of the user, their ISP, or any other intruder between the user and their guard node. A LAN adversary can collect TCP traces from the user and then use heuristic algorithms to turn those traces into cell sequences. For the purposes of this research, we assume this is indeed possible with high accuracy [7].

In this research, we analyze the circuit fingerprinting problem and separating it from *website* and *traffic* fingerprinting. While these problems are connected, circuit fingerprinting problem needs to be analyzed isolated from the other.

- The *traffic fingerprinting* problem is a LAN adversary distinguishing between Tor traffic and the rest of the traffic.

- The *circuit fingerprinting* problem is a LAN or relay adversary distinguishing circuit types and purposes in a Tor session. It involves the Tor protocol traffic.
- The *website fingerprinting* problem is a LAN or relay adversary distinguishing websites from each other within a Tor session. It involves the *application-layer* traffic.

Separating these problems from each other allows us to formalize them and gain a greater understanding on how they interact with each other.

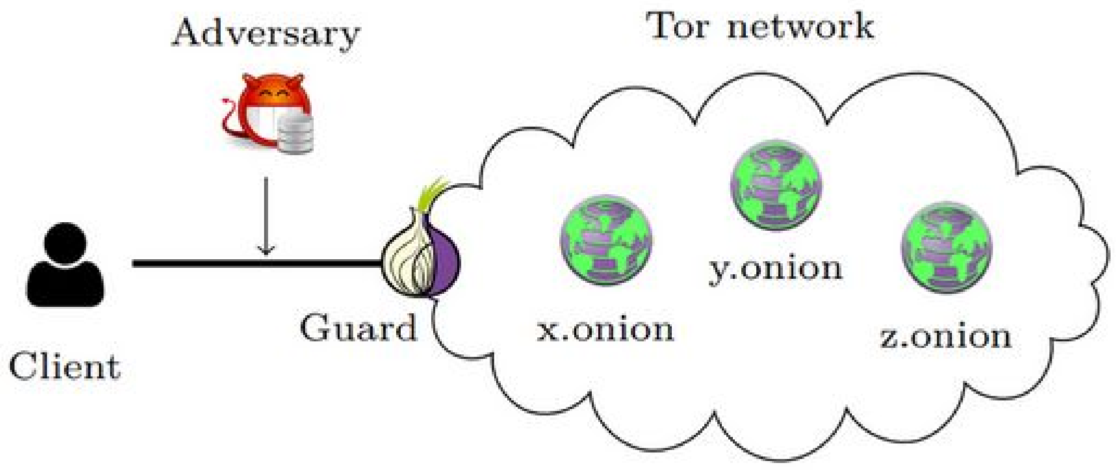


Figure 5: LAN adversary model

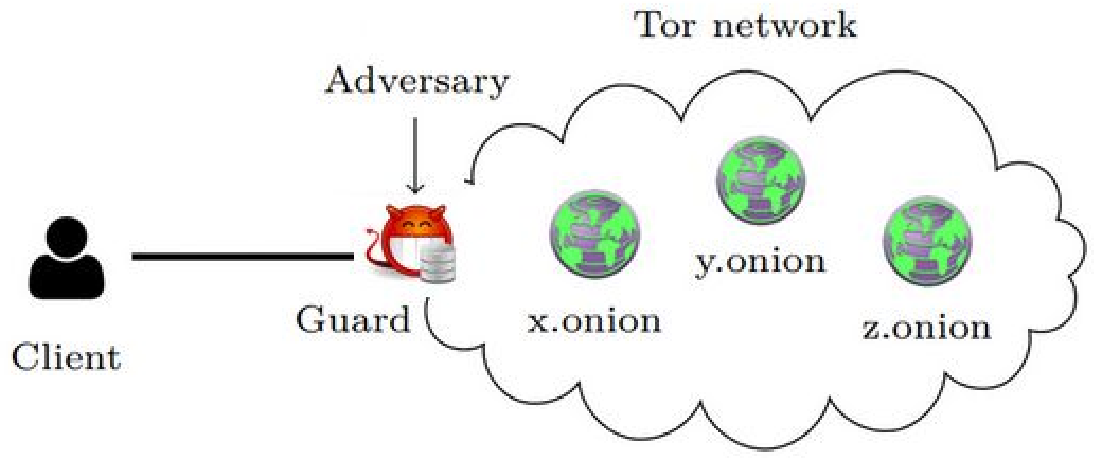


Figure 6: Guard adversary model

1.5 About this Thesis

In an ideal setting, adversaries should not be able to distinguish circuit types by inspecting traffic. Tor traffic should look amorphous to an outside observer.

As an example of a malicious scenario that results from a circuit fingerprinting attack, consider a source user who wants to submit information to a portal anonymously. The specific user uses Tor for most of his communication, but only uses an onion service to submit information anonymously to the portal. An attacker who can distinguish between user's clearnet Tor communication and its onion service connections can correlate the onion connections with the arrival of user's information to the portal. As another example, consider an ISP or a cloud hosting provider using circuit fingerprinting attacks to distinguish onion services from regular Tor clients within its network.

In this research, we study the circuit fingerprinting problem, isolating and analyzing it on its own, separately from other network traffic characteristics that can be used as fingerprints. Under this model, we revisit results from previous researches (*Kwon et al.* [1]), and show that distinguishing onion service circuits is possible with high accuracy, even when the application-layer traffic is identical, by exploiting fingerprints of the onion service protocol itself.

We then proceed to analyze and evaluate a padding framework for Tor [8], based on WTF-PAD, an adaptive padding system proposed by *Juarez et al.* [3] designed against website fingerprinting and has been shown to be effective and also versatile enough to be used for multiple purposes. This framework has already been implemented and deployed to the Tor network. We evaluate this defense, showing that it provides in fact little defense against realistic attacks, revealing the persistence of the fingerprinting problem.

We then proceed in designing a padding strategy, which is simple but has high-latency overhead. Through the evaluation of this strategy, important previously unknown behaviors that can act as circuits fingerprints are being revealed. Despite the fact that the aforementioned strategy is against the low-latency philosophy of the tor network, its evaluation reveals that the defense is successful, based on the classification results that we present.

The rest of the thesis will follow the structure described below.

Chapter 2: We describe the procedure of the data collection , process and classification. We illustrate the structure of the datasets, as well as the classification scenarios that we applied.

Chapter 3: We analyze the existence of the circuit fingerprinting problem on the current state of the tor network.

Chapter 4: We analyze and evaluate the Padding Framework, a defense already implemented on Tor [8].

Chapter 5: We describe a high-latency but effective defense, whose evaluation revealed new network characteristics that can be used as fingerprints.

Finally, some conclusions are going to be drawn based on the results extracted from the experiments. Moreover, we will introduce a different low-latency defense and mark it for future work.

2. METHODOLOGY

In this section we describe the steps for data collection, processing and classification that are used in order to feed our experiments.

2.1 Data Collection

As a dataset, we used a collection of 100 real onion addresses, selected based on web site traffic popularity and most importantly site availability. Availability of the sites has great significance for data gathering since the data collection procedure performed multiple times during the experiment. Initially we locally fetched every home page of each site including all the elements of the site. Following the fetching procedure, we alter each local site page by stripping off any external resource that the site requests during load. Since our experiment focuses on tor circuits analysis, having multiple requests towards external hosts would complicate the traffic analysis and pattern feature extraction for the initial target.

At this point, we hosted each cached website once on a clearnet server and an identical version on an onion service. Since the circuit fingerprinting issue exists on the Tor protocol layer, application layer traffic should not affect the classification process. Serving the same content over both clearnet and onion world was a key factor for the experiments, because application layer traffic remains the same and does not change during the requests.

After the sites were collected, gathering actual tor traffic traces was achieved using a Tor Browser instance combined with Selenium in order to automate the procedure. Moreover, the browser was instructed to use a custom patched Tor binary that exploits more detailed information regarding its operation. Specifically, our patch emits logs every time a circuit is created and for every incoming and outgoing cell. One of the problems that arose during the collection process on the client side was the possible distortion on timing accuracy of the cells. This could occur by many factors with the simplest being the network delays on the client side and the software integration (Tor browser and Selenium script) on the client's machine. However, due to the fact that the classification process and features that were used do not utilize timing information and that the cell order is preserved intact during the collection, timing accuracy issue was not further analyzed and currently was accepted as it is.

During data collection we instructed Selenium to fetch pages off our dataset multiple times. The amount of times and the set of pages to fetch depends on the experiment. Between each page fetch we forced Tor to avoid reusing already used circuits and instead create a new set of circuits. With this tactic tor does not reuse the existing circuits and reconducts the circuit creation procedure for every request.

Based on the sections of this research, we will mention the total number of circuits on each experiment. These circuits are the result of the following section, *2.2 Data Processing.*

2.2 Data Processing

We created a custom script in order to process the output log of tor data collection. The script receives the raw log as input alongside with multiple parameters that are dependent on the each classification scenario. The script reads the tor log and keeps track of all the circuits and their cells. Following the reading process, it creates an output file and separates each circuit to a different line. Each line includes the label of the circuit, indicating whether it is an onion circuit or not, and a vector of features for that circuit. This vector provides multiple circuit characteristics that are being used as features by the classifiers in order to classify a circuit as onion-related or not.

The characteristics that are being extracted are the following:

- ***Cell Sequences**:* A list containing all circuit cells alongside with the cell's direction. This is represented as +1 for incoming cells and -1 for outgoing cells.
- ***Duration of Activity (DoA)***: The total time that the circuit was active, calculated by the first and the last cell that produced for each circuit.

Moreover, each experiment uses a different classification scenario and is described in the respective sections.

2.3 Data Classification

Based on each experiment, the classifier splits the set of circuits into a train set and a testing set, and then trains itself using the train set. All results in this project are produced using Support Vector Classification (SVM) and Decision Tree classifiers. We used scikit-learn python library in order to create our classification models. In terms of parameter optimization for our models, we tried to keep the architecture simple and mostly focused on choosing the right features for the classification.

Similar to the website fingerprinting world, our experiments also carry the concept of open and closed world. We call an experiment open world when the hosts in the train URL dataset and the test URL dataset are disjoint. In this context, disjoint means that URLs in the train dataset are not to be found in the test dataset. On the other hand, we call an experiment a closed world when the train URL dataset and the test URL dataset are not disjoint. This means that both the train and test logs contain sessions to the same destinations. It’s important to note here, that in closed world scenarios the exact same session is never shared between train and test sets. Instead we share different requests but to the same destination.

Traditionally, due to the nature of open world scenarios, they are harder to deal with as a website fingerprinting attacker. However, as we previously mentioned, our research focuses on tor circuit fingerprinting. Fingerprints for circuits exist in the Tor protocol and not the application layer content, making open world scenarios on circuit fingerprinting problem more manageable though still tough.

In addition to the open and close world scenarios, we also create two scenarios based on the number of the websites that they involve.

Single-site scenario contains a single destination hosted in onion and not onion world. The classifier on this scenario attempts to determine whether a connection performed over onion or over clearnet. An easier to deal with scenario that implies that the application layer traffic is identical and only circuit fingerprints exist. Single scenario is better for providing a defense success rate since it is easier for the classifier to identify a single website.

Multi-site scenario contains multiple destinations hosted in onion and not onion world. Again, the classified attempts to determine whether the connection to a destination performed over onion or clearnet, however, without knowledge of the actual destination. This presents a real world scenario where most of the websites are different causing confusion to the adversary. Multi site scenario is better for providing attack success rate since the classifier identifies a website over multiple.

Eventually, we provide valuable metrics that present an overall picture of the classification procedure success. Among these statistics are *Accuracy*, *Precision*, *True Positive Rate* (*TPR*) and *False Positive Rate* (*FPR*) for each classification result.

Table 1: Classification scenarios

Open World	The Adversary has no prior knowledge of the websites that he attempts to identify.
Closed World	The adversary has prior knowledge of the websites.

Single-Site	The adversary has a single site as a knowledge base.
Multi-Site	The adversary has multiple sites as a knowledge base.

3. FINGERPRINTING TOR

In this section we will analyze and test the current tor state in order to verify that fingerprinting can be achieved. We will use multiple features for the classifiers for this evaluation. In particular, we will examine whether introduction and rendezvous circuits can be identified over all other circuits. As we previously mentioned, application layer traffic is identical for both onion and clearnet connection since we want to ensure that information leak exists on the onion service protocol. We will refer to this section as Experiment 1 for the rest of the project.

3.1 Features and Classifiers

For this experiment, we will break down two parts of onion service protocol, introduction and rendezvous circuits. The classifiers are splitted on these two types of circuits. More particularly, the first part will separate introduction circuits from all other types and the second part will separate rendezvous circuits from all other types.

The features used in this experiment are the following:

- ***Cell Sequences*:** An array that contains all circuit cells direction represented as -1 for outgoing cells and +1 for incoming cells.
- ***Duration of Circuit Activity*:** The total time that the circuit was active, calculated by the first and the last cell that produced for each circuit.

Moreover, we tested the following features that were used on previous research [1]. Using these features we can simulate and compare previous results with our findings. The features that were described on [1] were *Incoming and Outgoing Cells of Circuit*, including the number of the incoming and outgoing cells, *Duration of Circuit's Activity*, meaning the total time that each circuit was active, and *Circuit Construction Sequences*, meaning the circuit's cell direction.

For this experiment we used all 100 websites from our dataset. In contrast to [1], we use the same websites served over both clearnet and onion connections, in order to restrict classification to the circuit fingerprinting problem alone. For the *closed scenario* the training and the testing datasets are intersecting as previously described. Additionally, for the *open scenario* the disjoint datasets structure contains 75 websites in the training and 25 websites in the testing dataset. After data processing, we present the number of circuits in each dataset for each experiment on *Table 2*. These circuits describe the multi site scenarios of our tests.

Table 2: Number of Circuits in dataset for Experiment 1

Introduction	Rendezvous	Other
2835	2749	13511

3.2 Classification Results

In both cases classifiers performed very well having an average accuracy of 98-99% as can be seen in *Table 3*. In addition to our features, we identified that using the features from the research [1] also resulted in high classification accuracy.

Table 3: Accuracy Results (*Experiment 1*)

	Other-vs-Intro		Other-vs-Rend	
	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Close	0.99	0.99	0.99	0.98
Multi-Open	0.99	0.98	0.99	0.99

Based on the experiment, we found that circuit classification is possible without relying on application layer traffic, meaning that the fingerprint lies on the traffic pattern that produced that tor protocol. An important assumption that came up by these findings is that since the website traffic pattern is not exploited in this attack scenario, it is possible that an open world model may lead to high accuracy results. Summing up with Experiment 1, an adversary, that has no prior knowledge of a website traffic format, can identify whether the website is accessed via an onion service or via clearnet.

4. PADDING FRAMEWORK (PROPOSAL 302)

After verifying that client-side onion circuits fingerprinting is possible, we evaluate "Padding Framework (Proposal 302)" that has been implemented in Tor (*since 0.4.0.1-alpha release*) [8]. This padding framework has been inspired by *"Website Traffic Fingerprinting Protection with Adaptive Defense (WTF-PAD)"*, a probabilistic link-padding defense based on Adaptive Padding proposed by *Juarez et al.*[3], designed against website fingerprinting and has been shown to be effective and also versatile enough to be used for multiple purposes.

4.1 Padding Framework overview

With Tor padding framework, arbitrary circuit level padding patterns can be added and overlay non-padding traffic. This padding can occur at any hop of client's circuits between client and relay communication. However, for this framework to be functional, both sides, client and relay, need to support the same padding mechanism. A special padding negotiation cell in Tor protocol can be used by clients to negotiate with relays and determine which padding patterns should be used. Padding is performed by padding machines that apply it in terms of lapse of time between packets and total packet counts. Due to the fact that one of Tor advantages is that it is a low-latency network model, padding should not have a negative effect on this characteristic. Delaying packets could be a method to obfuscate traffic and confuse an attacker, however, this could lead to low-latency degradation. For this reason, the Padding framework achieves its purpose only by adding padding traffic taking into consideration the bandwidth cost that this activity generates.

4.2 Defense Analysis

Padding Framework works by using two padding machines that aim to hide *introduction* and *rendezvous* circuits. The padding is initiated by the client up to the middle relay that replies with its own padding. In particular, the machines intended to hide rendezvous and introduction circuits by making them look like exit and directory circuits respectively.

The padding attempts to hide the following features:

Circuit Construction sequence

The machines add padding cells during the initial circuit handshake trying to make them look like most general tor circuits until the end of the construction sequence. Specifically, the obfuscation attempts to make rendezvous look like *exit circuits* and introduction circuits look like *directory circuits*.

Number of incoming and outgoing cells

The number of a circuit’s incoming and outgoing cells could be used to distinguish circuit types. More specifically, client side introduction circuits have the same amount of incoming and outgoing cells, while rendezvous circuits have more incoming than outgoing cells. This framework aimed to change the number of cells that are sent in introduction circuits, while the rendezvous circuits remain intact as their flow resembles normal Tor circuits.

Duration of Activity

The period of time during which circuits send and receive cells can be used to distinguish circuit types. Client side introduction circuits are short lived, while service side introduction circuits are very long lived. Rendezvous circuits on the other hand, have the same median lifetime as general Tor circuits which is 10 minutes. By keeping client side introduction circuits open for 10 minutes, the framework attempts to imitate the duration of general Tor circuits.

4.3 Padding Strategy

Most general Tor circuits used to browse the web and retrieve directory information, start with the following *6-cell* relay sequence in *Figure 7*.

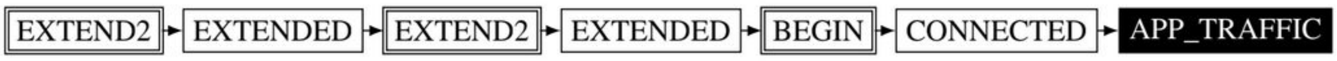


Figure 7: Exit Circuit construction sequence (outgoing cells are double outlined)

When this is completed, the 3-hop circuit has been established and a data stream has been opened. After this, fetching site content or directory information and establishing SSL connections usually occurs by a stream of DATA cells. After the sequence of the first 6 cells, 2 outgoing and 2 incoming DATA cells follow the circuit construction.

Focusing on "*Introduction circuit padding*", the machines are being activated when the INTRODUCE1 cell has been sent out. Based on this, the sequence before tha activation of the padding is presented in *Figure 8*.

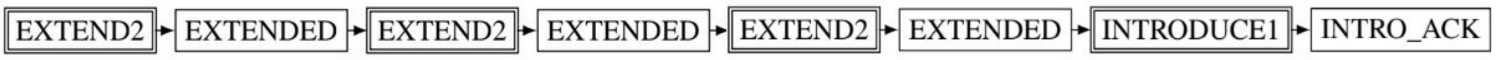


Figure 8: Introduction Handshake cell sequence

The above circuit matches the previously described on *Figure 7* on the first 7 cells, meaning the first 6 cells and the first outgoing DATA cell that matches INTRODUCE1 outgoing cell. The rest of real introduction circuits is followed by **INTRODUCE_ACK** cell and then it terminates. Padding framework adds an outgoing and an incoming cell to match the 10 cell sequence that is described in *Figure 7*.

Focusing on "*Rendezvous circuit padding*", the machines apply when the rendezvous point has been established (*REND_ESTABLISHED* has been received). Until then, the following cell sequence, presented on *Figure 9*, has been observed.

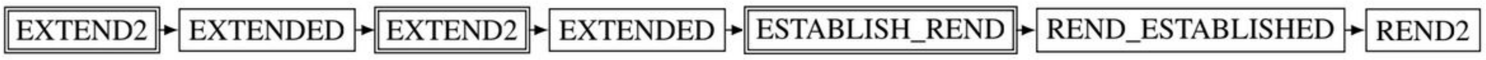


Figure 9: Rendezvous Handshake cell sequence

This matches the general circuit construction for the first 6 cells until the normal rendezvous circuit receives RENDEZVOUS2 cell followed by a BEGIN and CONNECTED cells. Since this pattern does to fit in the circuit construction sequence of general circuits, the machine initiates its padding after REND_ESTABLISHED is received and sends a PADDING_NEGOTIATE and a DROP cell before receiving PADDING_NEGOTIATED and a DROP cell.

4.4 Evaluation of Padding Framework

This experiment evaluates the effectiveness of the Padding Framework [8]. For this experiment, we performed a new round of data collection using the padding machines this time. Two classifiers were implemented, one that separates *padded introduction* circuits from all other types and a second that separates "*padded rendezvous"* circuits from all other types. After data processing, we present the number of circuits on *Table 4*. These circuits describe the multi site scenario of our Experiment 2 classification approach.

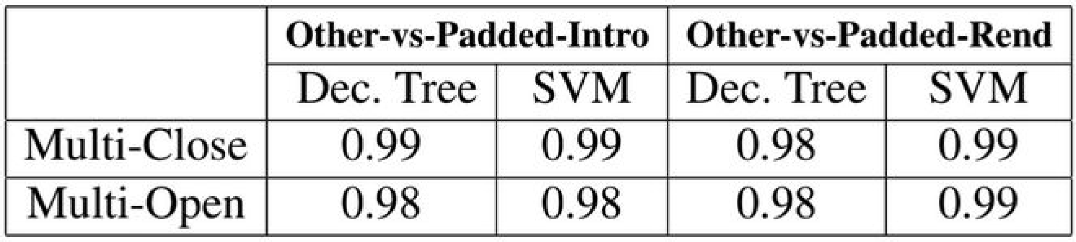
Table 4: Number of Circuits in dataset for Experiment 2

Introduction	Rendezvous	Other
2255	2122	14683

The features that we used for this process, similarly with the first experiment, are ***"Cell Sequences"*** and ***"Duration of Circuit Activity"***.

The classification results are presented on *Table 5*. Based on the relevant findings, classifiers still have very good performance, with an average accuracy of ***97-99%***. These results indicated that the Padding Framework defence did not provide any notable protection and there were still strong fingerprints that could be used for classification.

Table 5: Accuracy results (Experiment 2)



	Other-vs-Padded-Intro		Other-vs-Padded-Rend	
	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Close	0.99	0.99	0.98	0.99
Multi-Open	0.98	0.98	0.98	0.99

The following observations regarding the *Rendezvous* and *HSDir* circuits need to be noted.

First, *rendezvous circuits* can not be shaped to look like *exit circuits*. The initial approach was the attempt to shape rendezvous circuits into exit circuits by adding cells in the correct stage of the sequence. Since rendezvous circuits have a longer circuit handshake that exit circuits, trying to fix this disparity by just adding padding cells did not work. As previously reported, rendezvous circuits use 9 relay cells before application layer traffic while exit circuits use 6 relay cells. The attempt of padding the rendezvous circuits is not possible since the sequence of rendezvous circuits is larger and because of that, the two circuits can not look similar by padding. We can compare rendezvous and exit circuits cell sequence by observing *Figure 9* (*Rendezvous Circuit*) and *Figure 7* (*Exit Circuit*).

In addition, after the experiment, we identified that *HSDir* circuits are completely distinct from the rest of the circuits, hence they can be used as a fingerprinting factor. During our first experiment and also based on the previous work, *HSDir* circuits were not considered as a part of the onion service circuit fingerprint and for this reason the padding machines do not offer any obfuscation for these circuits.

Summarizing experiment 2, by implementing the framework, we do observe that the Padding Framework (Proposal 302) works as expected, even if the protection against fingerprinting it offered had minor improvements. The framework impleme-ntation was important for the defence that will be described in the following sections of this research.

5. ADVANCED PADDING STRATEGIES

The negative results of the previous experiment show that the Padding Framework offered no privacy improvements at all since the adversary did not face any significant difficulties in circuit separation and recognition. In this section we will analyze a padding strategy that can increase protection against circuit fingerprinting. We will evaluate and compare the results of this strategy and describe the advantages and disadvantages of this approach.

5.1 The dummy requests approach

As previously discussed, onion connections are different from clearnet connections as they include three extra requests between Tor nodes. These are an HSDir fetch, an introduction handshake (both run on their individual circuits) and a rendezvous handshake (run in the circuit that will host the application data). Preventing an adversary from distinguishing onion from clearnet connections can be achieved by generating dummy HSDir, introduction and rendezvous requests that will be very similar to the corresponding real ones.

These three circuits that will be generated have specific cell patterns, displayed in *Figure 10*. An HSDir fetch consists of 37 cells (3 outgoing, 34 incoming), an introduction handshake consists of 4 cells (2 outgoing, 2 incoming) and a rendezvous handshake consists of 3 cells (1 outgoing, 2 incoming). Having the padding machines injecting the aforementioned cell sequences in a circuit, we can create dummy requests that are indistinguishable from the real ones to an adversary that only observes the shape of the cells.

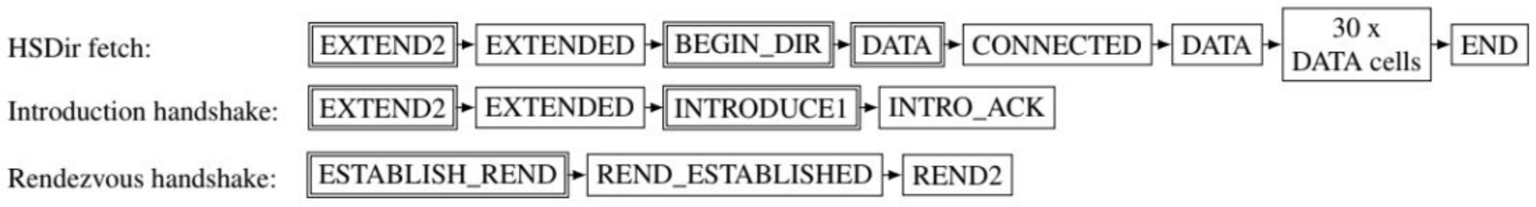


Figure 10: Padding patterns of dummy HSDir / Intro / RV requests

The main question is what is the exact time to perform these dummy requests and on which circuits. The following approach described on section *5.2* consists of the idea of injecting dummy requests only on clearnet traffic. Despite the simplicity of this idea, it requires to delay the actual data, which eventually will lead to high latency.

5.2 A High-latency but effective solution

Since the three requests discussed in *5.1* are missing in the clearnet connection when compared to the onion connection, the first approach is to inject a dummy *HSDir*, *Introduction* and *Rendezvous* circuit to the clearnet traffic, so that it matches the onion traffic. This simple solution can be achieved with the following steps. When a user asks to open a clearnet connection, Tor client "delays" the initial request and first creates two new circuits, extended to two random middle nodes inside the network. Then the padding machine injects a dummy *HSDir* request to the first circuit and a dummy *introduction* request to the second. Finally, the actual *exit* circuit for the clearnet connection is created, then the padding machine injects a dummy *rendezvous* request to the circuit before the actual traffic and finally the application data is sent as usual. This procedure creates three circuits whose traffic looks identical to a real *HSDir*, *Introduction* and *Rendezvous* circuit handshake.

Although simple, this strategy is referred to as a *high latency* strategy, since it has a major drawback: it delays all clearnet connections. This delay can not be avoided since the dummy requests involve several round trips to the middle node. Such a delay is against *low latency* philosophy that characterizes tor network, hence it is an experimental proposal for discussion and no immediate application to the network.

Still, this strategy is interesting from the point of view of "*circuit fingerprinting*", since it can potentially provide full privacy, making onion connections completely indistinguishable from clearnet ones, since the traffic is identical. As a consequence, in the following section we perform a thorough evaluation of this approach, which reveals new aspects of the network that act as fingerprints.

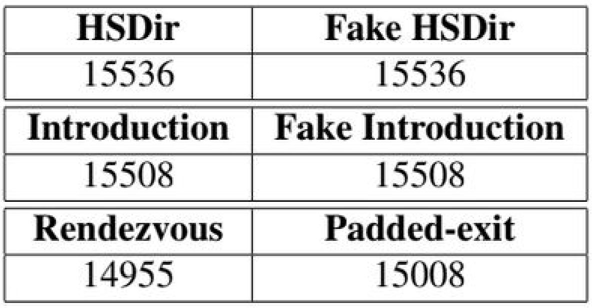
5.3 Evaluation of High-latency strategy

As previously discussed, the need to delay traffic makes this defense unimpleme-ntable in the currently deployed framework. As a consequence, we evaluate the defense by simulating its padding as part of our experiment as follows. First, for each *exit circuit* in our dataset, we create two new circuits, a *fake HSDir* and a *fake introduction* one, simulating those in the defense. We inject a dummy *HSDir* request to the first and a dummy *introduction* to the second and add the in the dataset. Note that this defense only adds padding to clearnet connection, having the onion traffic intact.

5.3.1 Initial Evaluation

In contrast to the previous experiments, in this approach both clearnet and onion connections have three circuits. In this experiment, we evaluate the adversary’s ability to distinguish each of the three pairs of circuits independently. We will use three classifiers in this experiment. First, a classifier which separates *fake HSDir* from *real HSDir* circuits. Second, a classifier which separates *fake introduction* from *real introduction* circuits, and third a classifier which separates *padded exit* from *rendezvous* circuits. Total number of circuits that were used for classification displayed on *Table 6.*

Table 6: Number of Circuits in dataset for Experiment 3 - *5.3.1*



HSDir	**Fake HSDir**
15536	15536
Introduction	**Fake Introduction**
15508	15508
Rendezvous	**Padded-exit**
14955	15008

The first attempt involves two datasets, a *multi-open* and a *multi-closed* dataset consisting of 10 different websites. Classification results are presented on *Table 7* and *Table 8*. These results indicate that the defense was effective in completely hiding *HSDir* and *Introduction* circuits in all scenarios. For rendezvous circuits, the defense was also successful in the *open world* scenario, where the classifier had not seen the same sites during training. However, in *closed world* scenario, even if the classifier struggled more to identify the rendezvous circuits compared to the almost perfect results of the previous experiments, average accuracy and precision were still at high levels, approximately between *70-80%* (*Table 7*, *Table 8*).

Table 7: Accuracy results (Experiment 3 - *5.3.1*)

	Fake-HSDir vs HSDir		Fake-Intro vs Intro		Padded-Exit vs Rend	
	Dec. Tree	SVM	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Closed	0.51	0.51	0.49	0.49	0.62	0.79
Multi-Open	0.52	0.52	0.50	0.50	0.50	0.49

Table 8: True Positive and False Positive Rates in comparison with Precision (Experiment 3 - *5.3.1*)

		Fake-HSDir vs HSDir		Fake-Intro vs Intro		Padded-Exit vs Rend	
		Dec. Tree	SVM	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Close	TPR	0.03	0.03	0	0	0.91	0.98
	FPR	0	0	0	0	0.31	0.72
	Precision	1.00	1.00	1.00	-	0.73	0.56
Multi-Open	TPR	0.04	0.04	0	0	1.00	0.49
	FPR	0	0	0	0	0.99	0.50
	Precision	1.00	1.00	-	-	0.49	0.49

Based on these results, we performed the same experiment in the *single-site* case, for each one of the same 10 websites. The results displayed in *Figure 11*, reveal the same behavior. *HSDir* and *Introduction* circuits are hidden, but *rendezvous* circuits can be distinguished from *padded exit* ones for most websites with high accuracy.

These findings were not expected, since, in general, the pattern of *padded exit* circu-it is in theory identical to those of *rendezvous* circuits. By examining the fingerprints used by our classifiers we managed to identify the following findings that caused this behavior. In all experiments we tried to keep the *application-layer* data identical, as described in previous sections, by having identical content served from web servers with identical configurations. Still, the actual *application-layer* traffic did not seem to match in terms of cell sequences.

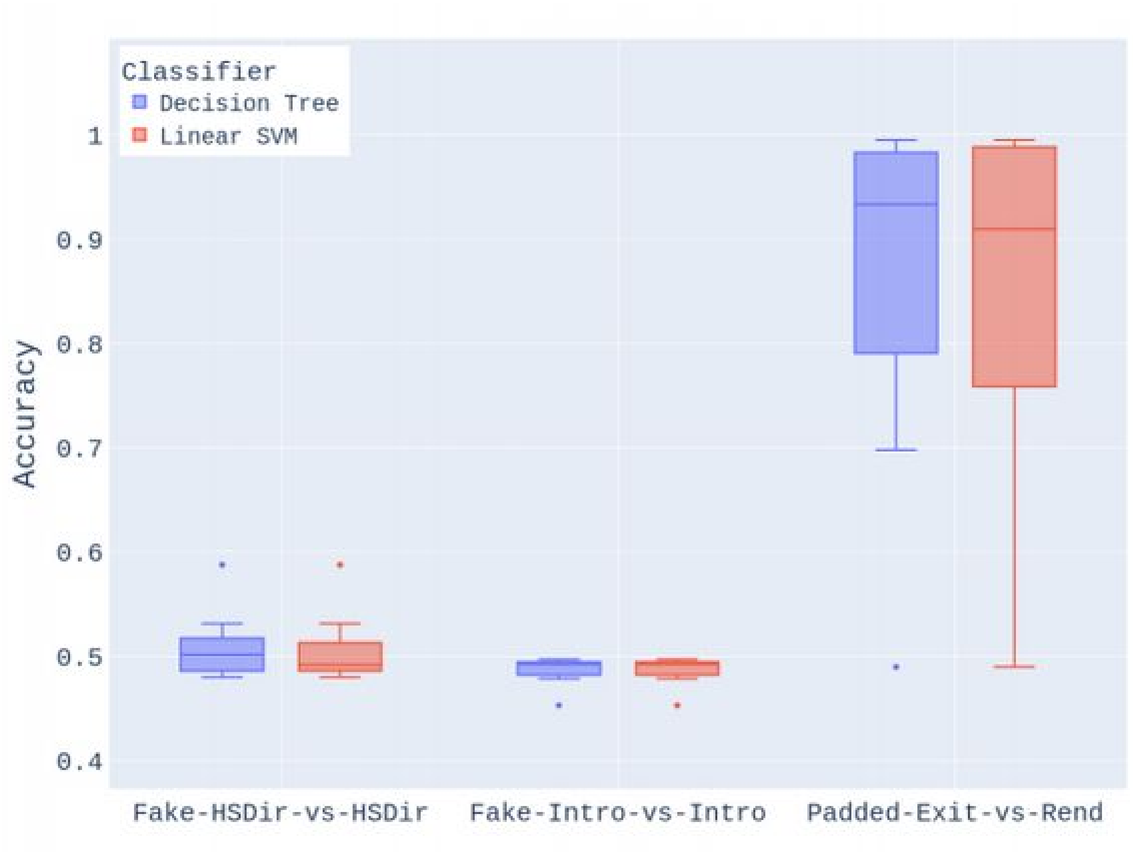


Figure 11: Single site accuracy results (Experiment 3 - *5.3.1*)

We found the following two major issues that caused this unexpected behavior:

Cell packing difference

The biggest fingerprint exploited by the classifiers is the fact that *cell packing* is different between clearnet and onion connections. By *cell packing* we refer to the way that an *exit node* arranges the data in order to transmit them back to the client. In the onion case, the cell packing is done by the *onion server*, who typically receives the data from a web server running on the same machine. This means that the web server transmits *application-layer* data to the onion server via a high- bandwidth localhost connection, allowing the provider to optimally package this *application-layer* traffic to cells.

However, in the clearnet case, the cell packing is done by the *exit node*, who recei-ves the data from the destination web server over a remote connection. As a conse-quence, unpredictable network delays are added to the *application-layer* traffic before it reaches the *exit node*, causing cells to be packed more loosely, but also a higher *variance* in the resulting number of cells.

The number of cells observed in our experiments for a specific website are shown in *Figure 12*. Rendezvous circuits are clearly packed more densely, but also have less variance in their cell count.

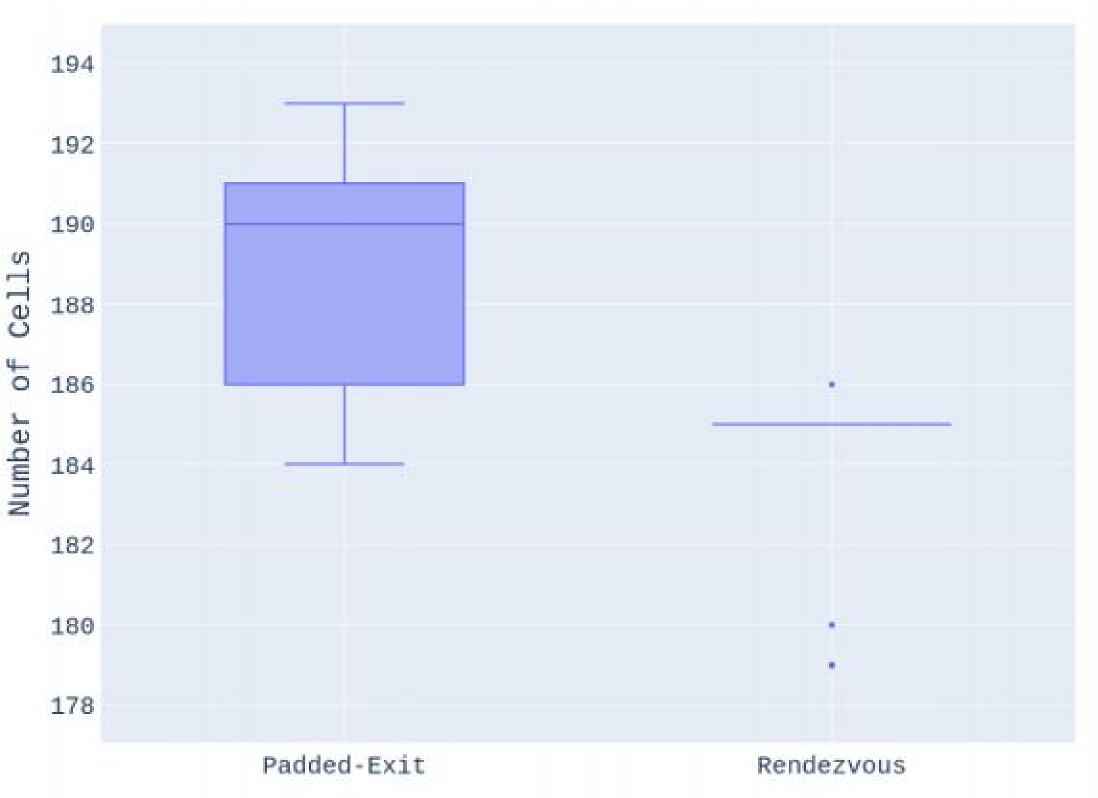


Figure 12: Cell packing difference between rendezvous and exit circuits

Latency affects the application-layer data

Another fingerprint was the fact that the inherent latency difference between *exit* and *rendezvous* circuits (due to the different number of hops) was creating patterns on *application-layer* cell sequences. In particular, we found that, for complex websites, the Tor Browser's scheduling of the various page resources (images, scripts, etc.) was sensitive to latency differences, causing, for instance, images to be predictably loaded in a different order over *clearnet* connections that over *onion* connections. Note that Tor Browser opens multiple TCP connections to fetch resources, which is important for the existence of this fingerprint. However, the exact reason why the scheduling order is so predictably different is unclear and left as future work.

Summarizing, it should be noted that these two fingerprints lie in the application data, not the onion service protocol. They depend on the traffic patterns of individual websites, which is why the defense was effective in the *open-world* scenario, but ineffective in the *closed-world* scenario (*Table 7*, *Table 8*).

5.3.2 Overcoming obstacles

The two fingerprints that were described on the previous section (*5.3.1*) are fundamental and affect the majority of onion service setups. Regarding *cell packing* fingerprint, we believe that onion servers can work around it by placing the underlying service endpoint on a remote host or emulating a virtual remote host. It might also be possible to reconstruct the padding machines so that they emit additional padding cells on *rendezvous* circuits to simulate the cell packing of *exit* circuits. Regarding Tor Browser's scheduling fingerprints, limiting the number of concurrent connections to 1 or reconstructing the scheduling algorithm to make it insensitive to latency differences could fix this issue. In order to work around these obstacles and test these defenses, we employed a Tor network running completely on a single machine, using the *Chutney Tor Emulator*, avoiding the problem of cell packing differences and ensuring that the *application-layer* cell length is the same in both clearnet and onion connections. Although this work around is not a defense proposal, it will help us to eliminate the cell packing fingerprint, in order to discover any other fingerprint that exists on the *onion protocol* side. Moreover, in order to bypass Tor Browser's scheduling issue, we decided to use *wget* instead of the Tor Browser for the remainder of the experiment. Total number of circuits that are used for this experiment are displayed on *Table 9.*

Table 9: Number of Circuits in dataset for Experiment 4 - *5.3.2*

HSDir	**Fake-HSDir**
15000	15000
Introduction	**Fake Introduction**
15000	15000
Rendezvous	**Padded-exit**
15000	15000

With these workarounds applied, we repeated the experiment. The results of the multi-open and multi-closed scenarios, displayed in *Table 10*, show that the defense is now effective in hiding all types of circuits, with accuracy close to *0.5* even in *multi-closed* case. Similarly, the results for the 10 individual *single-site* scenarios, displayed in *Figure 13*, also show low classification accuracy for all circuit types.

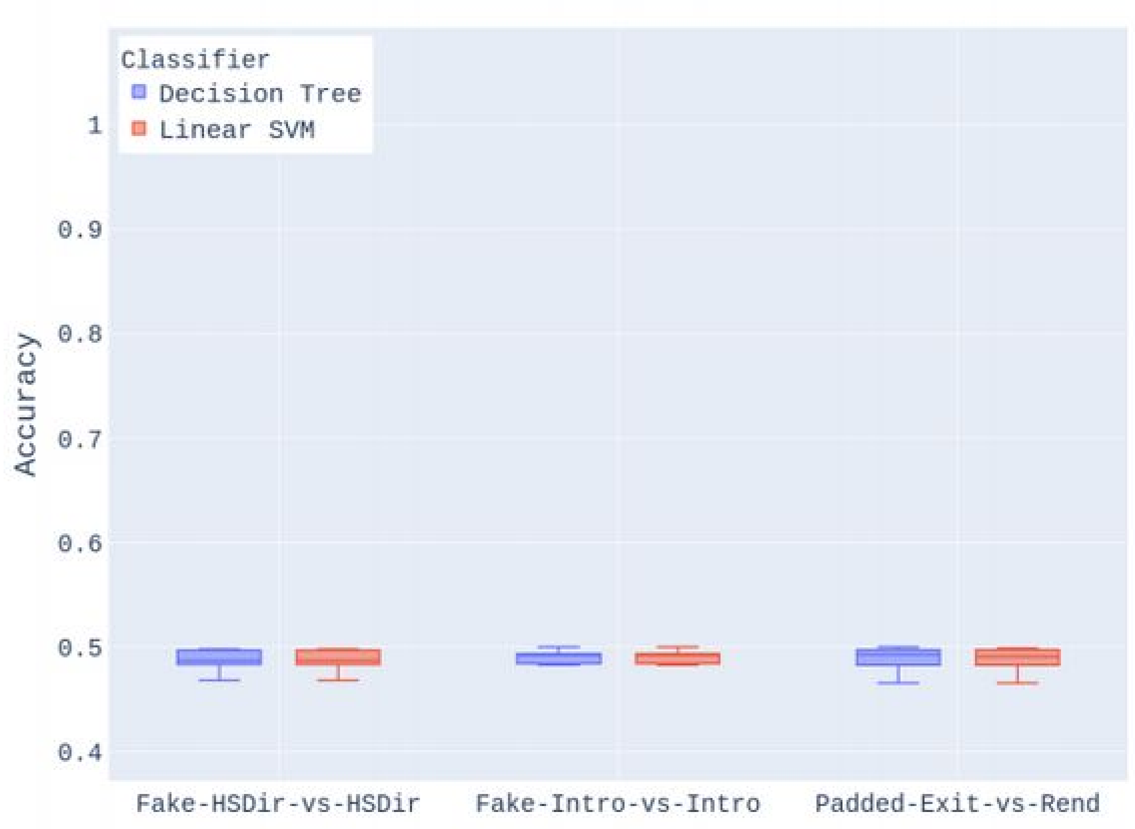


Figure 13: Single site accuracy results (Experiment 4 - *5.3.2*)

The effectiveness of this defense is also displayed on *Table 11*, with more classification metrics. Precision is close to *0.5* in all cases (or undefined in the case of Fake-HSDir vs HSDir, in which the classifier labels all circuits as negative), while TPR is almost identical to FPR, meaning that the classifier's outcome is in fact independent from the circuit's type, hence useless for the attack.

Table 10: Accuracy results (Experiment 4 - *5.3.2*)

	Fake-HSDir vs HSDir		**Fake-Intro vs Intro**		**Padded-Exit vs Rend**	
	Dec. Tree	SVM	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Closed	0.49	0.49	0.49	0.49	0.48	0.48
Multi-Open	0.50	0.50	0.50	0.50	0.50	0.50

Table 11: True Positive and False Positive Rates in comparison with Precision (Experiment 4 - *5.3.2*)

		Fake-HSDir vs HSDir		**Fake-Intro vs Intro**		**Padded-Exit vs Rend**	
		Dec. Tree	SVM	Dec. Tree	SVM	Dec. Tree	SVM
Multi-Close	TPR	0	0	1.00	1.00	0.61	0.86
	FPR	0	0	1.00	1.00	0.64	0.88
	Precision	-	-	0.49	0.49	0.47	0.48
Multi-Open	TPR	0	0	1.00	1.00	1.00	1.00
	FPR	0	0	1.00	1.00	1.00	1.00
	Precision	-	-	0.50	0.50	0.50	0.50

The second evaluation of the high-latency attack proved that the defense can be successful provided that the application-layer traffic issues are mitigated. However, on the scope of Tor protocol, this defense can manage to completely obfuscate the circuits, hence confuses the attacker on separating the circuits. Despite the fact that this defense is against the low-latency strategy and philosophy of the Tor network, it can help to introduce and develop a more robust and easier to apply defense against Circuit Fingerprinting attacks.

6. CONCLUSION AND FUTURE WORK

In this research we analyzed Tor Circuit Fingerprinting attacks and how they can be used by an adversary to exploit valuable information that reveal whether a website was requested either via clearnet or via a hidden service. In particular, we evaluated this attack on the current Tor network and proved that Circuit Fingerprinting is an existing issue on the Tor protocol. We evaluated Padding Framework from Proposal302 [8], a defense that has already been implemented on Tor, revealing that the defense did not provide any significant protection against the attack. We introduced a High-Latency defense that performs well against Circuit Fingerprinting, by providing enough obfuscation on the traffic so that the classifier can not determine the circuit's purpose. During its evaluation, we also revealed new application-layer characteristics that can be used as fingerprints. Based on the defense's evaluation, we can come to the conclusion that the defense can be successful given that the application-layer does not provide any more fingerprintable characteristics.

We intend to develop and evaluate a padding framework that uses Tor's network preemptive circuit's capabilities. To optimize performance, the Tor client continuously builds preemptive circuits, that are extended to only two hops and then stay dormant until a proper use for them is found. Moreover, we believe that the investigation of a cell's timing characteristics is important and could reveal more information and patterns that could be used as fingerprints. That said, the "time difference between cells" analysis could be used to improve current and future defenses against Tor Circuit Fingerprinting.

TABLE OF TERMINOLOGY

Website Fingerprinting	Αποτύπωση ιστοσελίδας
Circuit Fingerprinting	Αποτύπωση κυκλώματος
Eavesdrop	Υποκλέπτω
Adversary	Αντίπαλος
Exploit	Εκμεταλεύομαι
Cell Sequense	Ακολουθία πακέτων
High Latency	Υψηλή καθυστέρηση
Low Latency	Χαμηλή καθυστέρηση
Classifier	Ταξινομητής

ABBREVIATIONS - ACRONYMS

Tor	The onion router
TCP	Transmission Control Protocol
OP	Onion Proxy
OR	Onion Relay
IP	Introduction Point
RP	Rendezvous Point
HSDir	Hidden Service Directory
WF	Website Fingerprinting
CF	Circuit Fingerprinting
LAN	Local Area Network
DoA	Duration of Activity
ISP	Internet service provider
SVM	Support Vector Classification

REFERENCES

[1]. Albert Kwon, Mashael AlSabah, David Lazar, Marc Dacier, and Srinivas Devadas. Circuit fingerprinting attacks: Passive deanonymization of Tor hidden services. In *24th USENIX Security Symposium, USENIX Security 15*, pages 287–302. USENIX Association, 2015.

[2] Marc Juarez, Sadia Afroz, Gunes Acar, Claudia Diaz, and Rachel Greenstadt. A critical evaluation of website fingerprinting attacks. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 263–274, New York, NY, USA, 2014. Association for Computing Machinery.

[3] Marc Juarez, Mohsen Imani, Mike Perry, Claudia Diaz, and Matthew Wright. Toward an efficient website fingerprinting defense, 2015.

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

[6] Alex Biryukov, Ivan Pustogarov, and Ralf-Philipp Weinmann. Trawling for Tor hidden services: Detection, measurement, deanonymization. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, SP '13, page 80–94, USA, 2013. IEEE Computer Society.

[7] Tao Wang and Ian Goldberg. On realistically attacking Tor with website fingerprinting. *Proceedings on Privacy Enhancing Technologies*, 2016, 02 2016.

[8] George Kadianakis and Mike Perry. Proposal 302: Hiding onion service clients using WTF-PAD, https://lists.torproject.org/pipermail/tor-dev/2019-May/013822.html

[9] Roger Dingledine, Nicholas Hopper, George Kadianakis,and Nick Mathewson. One fast guard for life (or 9 months). In *HotPETs*, 2014.

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

[11] Andriy Panchenko, Asya Mitseva, Martin Henze, Fabian Lanze, Klaus Wehrle, and Thomas Engel. Analysis of fingerprinting techniques for Tor hidden services. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, pages 165–175. ACM, 2017.

[11] Andriy Panchenko, Lukas Niessen, Andreas Zinnen, and Thomas Engel. Website fingerprinting in onion routing based anonymization networks. In *Proceedings of the 10th Annual ACM Workshop on Privacy in the Electronic Society*, WPES '11, page 103–114, New York, NY, USA, 2011. Association for Computing Machinery.

[12] Tao Wang and Ian Goldberg. Improved website fingerprinting on Tor. In *Proceedings of the 12th ACM Workshop on Workshop on Privacy in the Electronic Society*, WPES '13, page 201–212, New York, NY, USA, 2013. Association for Computing Machinery.

[13] Payap Sirinam, Mohsen Imani, Marc Juarez, and Matthew Wright. Deep fingerprinting: Undermining website fingerprinting defenses with deep learning, 2018.

[14] Tor Browser automation with Selenium. https://github.com/webfp/tor-browser- selenium