Correlation attacks on the Tor network

Dissertation presented by
Henri Crombé , Mallory Declercq

for obtaining the Master's degree in
Computer Science

Supervisor
Olivier Pereira

Readers
Marco Canini, Florentin Rochet

Academic year 2015-2016

Abstract

Internet privacy has become one of the most important concepts of our time. People are worried about the surveillance of their digital communications and are in need of reliable tools to preserve their anonymity as well as the privacy and the integrity of their communications. Researches have been made to design and set up such reliable systems and The Onion Router (Tor) is one of them. Like many other systems, Tor is far from being perfect and may be subject to traffic correlation attacks. Such attacks can be carried out using a variety of methods and approaches.

In this context, we develop the threat-model of an adversary supervising Tor relays and carrying correlation attacks using Tor log files. Three different correlation methods are implemented and their performances are evaluated on reduced-size Tor networks simulated with the Shadow simulation tool. The performances are evaluated considering different client models and reveal what method is more suitable for the adversary. Furthermore, the document develops the procedure an adversary could follow in order to carry out the attacks in a realistic environment.

The results we obtained through simulations suggest that an adversary could conveniently use the Tor log files generated by Tor controller to perform successful end-to-end correlation attacks against the different client models. Furthermore, the results also suggest that among the three correlation methods that were implemented, one was surely more reliable than the other. The results eventually illustrate that the most reliable correlation method performs well against every client models that were simulated. This suggests that Tor users can either do some instant-messaging, web-surfing or even download some large files, they are almost under the same level of threat.

Acknowledgements

First of all, we would like to express our gratitude to our supervisors, Prof. Olivier Pereira and Florentin Rochet, for their availability and their valuable feedback throughout this thesis year. They put their faith in us and always urged us to do better in order to accomplish a meaningful work.

We also would like to thank Prof. Marco Canini, who agreed to take part to this thesis by spending his precious time reading and evaluating our work.

In addition, we are thoroughly grateful to Rob Jansen for his inestimable help and insights regarding the comprehension, the utilization and the customization of the Shadow simulation tool. We can't tell how many times he assisted us when we were struggling setting up proper simulations.

Last but not least, we would like to thank all our friends and families for their encouragement and their support through this year. Particularly Sylviane de Wilde who spent her time reading our work pointing out our English mistakes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, many people agree to say that online anonymity is a fundamental notion to protect privacy and freedom of speech. This concept allows people to express their points of view on sensitive topics without fearing repercussions. Moreover, it prevents ill-intended individuals and unscrupulous marketers from intruding into people's private lives for commercial, political or any other purposes. Therefore, to benefit from online anonymity, more and more people start using sophisticated tools to protect their anonymity on the Internet [45]. The Onion Router, Tor, is one example of such a tool and refers to a low-latency network designed to anonymize the traffic of interactive applications such as web browsing and instant messaging [15]. At this time, it is used every day for a wide variety of purposes by military, journalists, activists, and many others. Obviously, it is also totally suitable for common people that want to protect their privacy from unscrupulous marketers, identity thieves or national and international surveillance.

Basically, Tor provides anonymity to its users by encrypting their communications multiple times and by bouncing them around a vast network of more than 7000 relays ran by volunteers all around the world. Those layers of encryption coupled with the fact that Tor's traffic passes through at least three different relays (each of them only knows the identity of the previous and the next relay in the network) before reaching the final destination are supposed to prevent anyone watching your Internet connection from learning what sites you visit, as well as preventing the sites you visit from learning your physical location.

Nonetheless, as stated by its designers, Tor like any other low-latency anonymity networks fails to provide 100% anonymity in all situations [2]. This is particularly true when an adversary manages to simultaneously control or observe both the entry and the exit side of a communication passing through the anonymity network. Since, in such a case, the adversary, by relying on some correlation techniques and some statistics on the observable traffics, is able to discover with whom a given user interacts. Those kinds of attacks are usually called end-to-end correlation attacks or end-to-end confirmation attacks in the literature. They suggest that Tor users privacy and anonymity are at risk since they permit to link the source and the destination of anonymous communications.

In the context of this thesis, we investigate the feasibility and evaluate the threat of end-to-end correlation attacks performed by a relay adversary. To achieve this objective, we start in chapter 2 by laying the foundation of the Tor anonymity scheme and by providing the reader with a few technical information about Tor's design. We also present in chapter 2 some related works regarding end-to-end correlation attacks. The overall goal of this chapter is to familiarize the reader with Tor and the correlation attacks so that it will be easier for him to understand the implementation of the developed attacks described in section 2.6.3. In Chapter 3, we explain why we have chosen Shadow as simulation tool and we describe how we have generated our topologies to evaluate the efficiency of the developed attack against four distinct user models : web surfing, bulk download, irc and ssh session. Afterwards, in chapter 4, we present three distinct correlation techniques supported by our attack. We compare the efficiency and the accuracy of each of those techniques over two adversary models. Firstly, a model in which the adversary controls all relays of the network and has some prior knowledge about the type of traffics flowing in the network. Then, we consider a more realistic model in which the adversary still controls all relays of the network but doesn't know in advance the type of the transiting traffics. Finally, based on the results obtained for the second adversary model, we identify which correlation technique is best suited to deanonymize users in the live Tor network and we infer how vulnerable is the anonymity of Tor users against end-to-end correlation attacks.

1.1 Contributions

This thesis presents the implementation of three versions of an end-to-end correlation attack performed by a relay adversary. Initially, the efficiency of each version is evaluated over four user models : web surfing, bulk download, ssh and IRc. And an unrealistic adversary model is considered. The purpose of this step is to determine which version of the attack is the best when an adversary controls each relay of the network and knows in advance the type of the transiting traffics. Somehow, this step gives a first insight about how vulnerable each type of traffic is and which version of the attack seems the best suited to break user's anonymity. Then, as this last adversary model is far from the one achievable in the live Tor network, a more realistic one is taken into account to determine which version of the attack should be implemented in the live Tor network in order to increase the chances of an attacker to unveil Tor users anonymity. The results obtained with this last step give an order of idea of the level threat of end-to-end correlation attacks against users anonymity as well as the most efficient correlation technique among the three that were implemented and tested.

Chapter 2

Background

This chapter aims to give an overview of the Tor network to the reader. It briefly describes the set of devices presents in the network, as well as, all technical parts that we find relevant to understand the implementation of our end-to-end correlation attack. In addition to that, a state of the art is depicted to introduce some related works about end-to-end correlation attacks. Finally, the last section of the chapter presents the threat model of the developed attack.

When the Internet Protocol (IP) was first designed, the protocol's designers didn't have privacy and anonymity of the communications in mind. This has resulted in a global network where every communication could be traced down from their source to their destination. Any observers placed on the network could learn who was communicating with whom, and if the communication was not encrypted, he could even understand what the content was about. The observer could be an Internet Service Provider (ISP) that wants to make money on their clients web history or a government that wants to filter or analyse its internal or international communications. The observer could also be an adversary eavesdropping a limited amount of communications. To address these issues of peers anonymity and communication confidentiality, a wide range of protection schemes have been proposed.

The first common way for an Internet user to hide his identity is to use a public proxy. The proxy is used as an intermediary between the communicating peers to prevent malicious observers to know who is really behind a given traffic stream. However, this solution is not completely safe considering that the proxy's provider could leak or reveal the proxy's logs, exposing the user's identity and communications. Furthermore, the logs could be retrieved by hacking or worse, the proxy could be managed by the people the user wants to hide from. Therefore, an Internet user must choose another approach to ensure his privacy. A safer solution would be to encrypt the traffic multiple times, then bounce it through different network nodes, before sending it to the legitimate receiver. By this way, it would be more complicated for an adversary to find the communicating peers. This kind of anonymity schemes can be split into two distinct categories : high-latency and low-latency systems.

With high-latency anonymous networks, communications are forwarded through different nodes and are purposely delayed for a pseudo-random period of time. Considering that the latency in these kind of networks is very high, they are not used for typical web-browsing but rather to send messages that don't suffer from the delay. The goal of adding such delays is to ensure the anonymity against traffic analysis attacks led by large observers. Alternatively, low-latency systems, like the Tor network, aim to reduce the communication delays while still providing peers anonymity. Therefore, low-latency protocols are preferred when the communication needs fast or real-time responses (web authentication, web-browsing, ...) whereas high-latency protocols are preferred for communications tolerating some delay.

The Tor network is one implementation of a low-latency anonymous network. Its objective is to provide anonymity and privacy over Internet for applications running over TCP while preserving a relative good user-experience. To achieve this, it relies on the implementation of the Onion Routing protocol. With Onion Routing, users can choose their path in the anonymous network and build a corresponding circuit where every node is only aware of its successor and predecessor nodes. The traffic goes through the circuit by the mean of encrypted fixed-size cells that are decrypted and forwarded at each node [15]. This Onion Routing protocol makes use of telescopic path-building design to establish reliable circuits. Meaning that the initiator of a connection first negotiates the session keys with each of the successive nodes before adding them in the circuit [15]. Then, when the client tries to reach another party through the network, he will successively encrypt the communication cells with the session keys negotiated before, starting with the session key negotiated with the last node in the circuit. The number of encryption layers depends on the number of nodes in the circuits.

2.1 Tor nodes

The Tor network is based on a set of different kind of nodes that interact with each other to provide a secure and an anonymous environment to its users. The list below gives a brief insight of the role played by each of these nodes within Tor:

Onion proxies, also known as Tor clients, are special proxies that clients must run on their machine to connect to the Tor network. The objectives of an onion proxy is to establish new circuits in the Tor network and to forward the TCP traffic coming from an Internet browser to the anonymous network, and conversely.

Onion routers, also known as Tor relays, form the heart of the Tor network. They handle the Tor traffic and forward each flow of cells to their next destination. By default, Tor establishes circuits using at least 3 relays : an entry, one or more middle and an exit relay. The entry relay is the first node of the circuit. It receives traffic from the client onion proxy and forwards it to the middle relay. Therefore, the first node of the circuit is only aware of the client's identity (IP address) and doesn't know the final destination. The middle relay only forwards traffic between the entry and the exit relays. Consequently, it doesn't know the real identities of the communicating parties. Finally, the exit relay is in charge of forming the gateway between the anonymous network and the Internet. Hence, the exit relay is aware of the identity of the final destination but doesn't know who is the source. This kind of circuit made of 3 relays ensures relatively good anonymity and throughput.

Directory servers are trusted nodes that are responsible of constantly analysing, storing and relaying information about the network state. Each directory servers provides a complete mapping of the network, listing all the nodes and their characteristics (state, flags, bandwidth, ...). When an onion proxy starts, it first downloads a list of available relays from a directory authority (i.e. directory servers) and then, given this list, establishes the most effective/reliable circuit. Therefore, Directory servers must be reliable and active almost all the time. Indeed, an adversary controlling a Directory node could influence onion proxies to choose a given path by advertising a modified network state.

Bridges are onion routers that aren't listed in Tor directory authorities. The goal is to hide some onion routers from the public so that ISPs or governments cannot block the connection to all the relays at one time. Bridges are essential for Tor users who live under censorship since ISPs in some countries are under control of local authorities and they try as much as possible to block Tor traffic. The easiest way for such ISPs to filter Tor connections is to block the traffic going to all the known relays. Therefore, when censored Tor users want to connect to Tor, they need to specify which bridge and what obfuscation technique they want to use.

Hidden servers are anonymous servers connected to the Tor network. They provide usual web services (web/mail server, instant messaging server, ...) without revealing their identities. They are called "hidden" because the real IP addresses of such servers are unknown by clients accessing them. The only way to reach an hidden server is to browse its onion address (16 alphanumeric characters followed by the *.onion* suffix) in the onion proxies. This browsing request leads to the construction of two circuits towards an onion router acting as a rendezvous point. One circuit goes from the user to the rendezvous point and the other from the hidden server to the rendezvous point. Once built, these circuits allow the user to interact with the hidden server. The rendezvous point simply relays messages from client to service and vice versa.

Now that all the components forming the Tor network have been briefly described, let's see how onion proxies build their anonymous circuits.

2.2 Path selection

The path selection algorithm is the strategy followed by Tor to choose an ordered sequence of nodes (*path*) that will be used to build a circuit between a source and a destination. Since its release, the Tor path selection algorithm has evolved many times in order to improve either the performance, the anonymity, the reliability or in response to attacks on the network. As stated in many papers, this algorithm is a key point of Tor design because it has a direct impact on performance and users' anonymity [32, 44, 4, 19]. In its current version, the path selection algorithm mainly chooses three nodes of the circuit according to their bandwidth [13]. This strategy ensures that the network is able to meet the user requirements. Indeed, if Tor was based on an unweighted path selection algorithm, as it was the case in the past, clients would build their paths by choosing relays uniformly at random. As a result, the anonymity provided by Tor would increase since the diversity of paths would be greater but the average performance would suffer since a lot of low-bandwidth nodes would be selected by the algorithm [44]. To overcome this performance issue, the current version of the algorithm uses a strategy based on a weighted random choice where the weight depends on the node's bandwidth. This later strategy ensures good performance but reduces path diversity since a few high bandwidth relays are prioritized making the end-to-end traffic correlation attack easier to conduct.

In order to make end-to-end correlation attack harder to perform, the path selection algorithm follows these constraints for each generated path [13]:

1. The same relay is not chosen twice for the same path.
2. No more than one router is chosen in a given /16 subnet or family.
3. Non-running or non-valid routers are not chosen.
4. If Guard nodes are used then the first node of a circuit must be a Guard.

The first and the second rule have been defined to make end-to-end correlation attacks more difficult to perform. Indeed, by avoiding two nodes on a path being in the same administrative domain, an attacker needs more than one malicious relay to perform an attack [32]. Effectively, if this rule is not enforced, an adversary is able to perform a simple attack which consists in running one or two relays on the same machine or network. Consequently, a client who would have chosen the adversary's nodes as entry and exit nodes would have a high probability of having his anonymity removed since the adversary might observe both entry and exit traffic of this client.

The last rule has been added to the path selection algorithm to prevent well-known attacks against Tor network such as the predecessor and the locating hidden servers attacks [46, 37]. These profiling attacks rely on the fact that a client who always uses new relays to build a circuit has a high probability of picking relays controlled by an adversary. The official documentation confirms this claims [13]:

> "If we choose entry and exit nodes at random, and an attacker controls C out of N relays (ignoring bandwidth), then the attacker will control the entry and exit node of any given circuit with probability $(C/N)^2$. But as we make many different circuits over time, then the probability that the attacker will see a sample of about $(C/N)^2$ of our traffic goes to 1."

To mitigate this issue, the design of Tor has been improved to use *entry guards*. In the next section, we provide some details about what is an entry guard, why is it useful and how Tor manages them.

2.3 Entry guard

An entry guard is a onion router chosen at random by the onion proxy during the creation of a circuit. This relay always acts as entry point in the circuit which means that only the first node of a circuit can be an entry guard. This type of node has been added to the Tor design in order to decrease the opportunities for an adversary to be the first hop of a circuit. As we will see later, the fact that an adversary is able to observe both ends of a communication channel is a serious threat to users' anonymity. Another benefit given by the entry guards is related to the fact that a relay must contribute to the Tor network for several days before having a chance to be selected as an entry guard. This feature allows to delay by several days all attacks which require the control of an entry guard by the adversary. Note that a relay can only be promoted as an entry guard if it fulfills these three requirements [11]:

1. Online longer than 12.5% of relays, or for 8 days.
2. Advertise at least the median bandwidth in the network, or 250KB/s.
3. Have at least the median weighted-fractional-uptime of relays in the network, or 98% WFU.

In the current version of Tor, each client maintains a list of entry nodes (chosen *guards*) persistently stored on disk [13]. When a client builds a new circuit, the first hop of the circuit (*entry guard*) is selected at random from among the first three usable guards on the list. It is important to note that the entry guards list of a client is not fixed. Actually, when an entry guard is used by a client for some time (60-90 days), the entry guard is discarded from the list and another entry guard is selected. This rotation of entry guards is beneficial, because on one hand, it allows a client that would have chosen a compromised guard to recover some privacy and, on the other hand, it ensures a better load-balancing.

In the last few years, some work has been done to show that the way Tor is currently using entry guards is not enough to prevent large-scale attackers from de-anonymizing most users within a relatively short time frame [17, 12]. Based on that finding, a proposal has been formulated to solve the problems of the current guard selection strategy [40]. The key idea of the proposal discussed is to choose a single guard for each client from a set of high-bandwidth relays and then use it for every circuit during a period of up to nine months. Since this strategy is not implemented in the current version of Tor, we will not describe it in details.

2.4 Tor cells

In Tor, any traffic flowing between two entities (onion router $\leftrightarrow$ onion router, onion proxy $\leftrightarrow$ onion router) is always divided into fixed-size cells (512 bytes) to provide some resistance to traffic analysis [15]. These units of communication are composed of a header and a payload. The header contains a circuit identifier (circID) that specifies to which circuit the cell refers to and a command identifier (CMD) that describes how to interpret the cell. According to the value of the command identifier, a cell can be interpreted as a control cell (cf. Figure 2.1) or as a relay cell (cf. Figure 2.2).

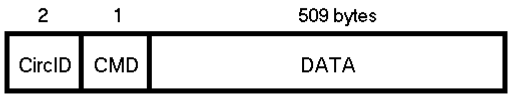


Figure 2.1: Control cell structure.

Control cells are mainly used to set up new circuits and tear down existing ones. The creation of a circuit always requires the exchange of two control cells. First of all, a *create* cell which informs an onion router that a new circuit plan to add it as one of its three hops.

Then, a *created* cell which is returned by the receiver of the *create* cell to acknowledge it. When it comes to release an existing circuit, a *destroy* cell is transmitted along network's entities being part of the circuit to destroy.

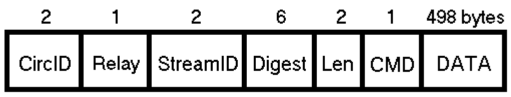


Figure 2.2: Relay cell structure.

Regarding relay cells, they are used to carry end-to-end stream data. Compared to control cells, relay cells have an additional header (called the relay header) located in front of the payload block (called the relay payload). This header contains useful information about the cell such as the *streamID* field indicating to which stream the cell belongs (many streams can be multiplexed over a circuit); The *digest* field containing the result of the checksum; The *len* field holding the size of the relay payload, and finally, the *relay command* field specifying the task assigned to the cell. This later field can take many values[1]:

- *Begin*: to open a new stream.
- *End*: to close a stream cleanly.
- *Connected*: to notify the onion proxy when the stream is established.
- *Extend*: to extend the circuit by a hop.
- *Extended*: to acknowledge the circuit extension.
- *Data*: to carry some data.

As you will see in the next section, the entire content of the relay header and the relay payload are encrypted (resp. decrypted) together whenever the cell reaches an entity (OR/OP) along the circuit.

2.5 Circuit construction

When a Tor user wants to interact with a third party (user, website, hidden service, ...), Tor has the responsibility to attach the TCP stream between these parties to an appropriate open circuit [13]. By default, Tor constructs a set of circuits preemptively in order to avoid delays related to the establishment of a new circuit between peers. Obviously, a circuit connecting two parties does not always exist, and thus, it is a necessity for Tor to build it in order to handle user requests. The remainder of the section will describe Tor's circuit construction mechanism.

As illustrated by figure 2.3, the creation of a circuit is always initiated by the user's onion proxy (OP) which selects three onion routers (OR1, OR2, OR3) according to the

[1]Note that this list is not exhaustive. An interested reader may refer to this link [14] to get a detailed view of all correct values.

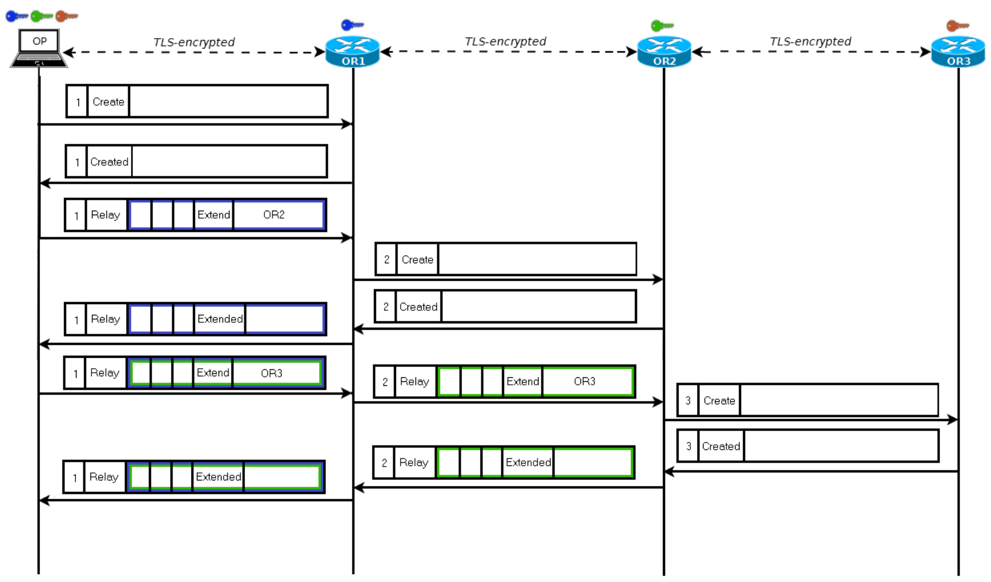


Figure 2.3: Three-hop circuit construction in the Tor network.

Tor's routers selection algorithm (cf. section 2.2). Then, given the three chosen relays, the user's OP is responsible of incrementally building a three-hops circuit towards a given destination [14]. To achieve such a goal, the user's OP and the chosen ORs communicate with each other by sending fixed-size messages (cf. section 2.4). Basically, a circuit construction requires four types of cells. The first type of cell sent by the user's OP to initiate the construction of a circuit is the *create* cell. The purpose of this cell is to inform its receiver (e.g. OR1) about the cryptographic parameters that will be later used to generate a shared encryption key. In the current version of Tor, the shared keys are generated with a key agreement algorithm called *ntor*. This algorithm replaces the previous TAP handshake to reduce the computation overhead of circuit construction. Indeed, compared to TAP, ntor eliminates the use of RSA encryption-decryption and relies on Curve25519 to reduce the Diffie-Hellman computational complexity [29, 6]. As soon as an onion router receives a successful *create* cell, it responds to the sender of this cell with a *created* cell that serves as an acknowledgement and that allows the receiver to inform the sender about its own cryptographic parameters. Once this mutual exchange of cells is done, the first portion of the circuit is established (one hop from the user's OP to OR1). To extend the current circuit up to the next onion router (OR2), the user's OP has to send an *extend* cell to the last relay of the current circuit (OR1). This cell must indicate to OR1 the identity of the next onion router involved in the circuit (OR2). Upon arrival of this message at OR2, a new shared key is negotiated between the user's OP and OR2, then OR2 has to acknowledge the *extend* cell by sending an *extended* cell towards the user's OP. The above process is repeated to extend the circuit from OR2 to OR3. Note that, by default, each circuit has a lifetime of ten minutes. Once this limit is reached, no new streams can be attached to the circuit.

2.6 Attacks on the Tor network

2.6.1 State of the Art

This section aims to give an overview of the different researches realized in the past and which are related to the security and the anonymity of the Tor network. The reader will be introduced to various threat models used against Tor and to the related attacks developed in the literature.

As previously described, the Tor network is a low-latency anonymous network providing both user anonymity and network performance. In order to provide a good user experience, Tor focuses on providing low communication delays while preserving the anonymity of its users. However, low Round Trip Time (R.T.T.) comes with a price. Indeed, since Tor attempts to optimize the communication delays between peers, the traffic characteristics such as the R.T.T and throughput at one end of the anonymous path gets reproduced at the other end of the path. This makes the Tor network vulnerable to traffic analysis attacks for an adversary who can eavesdrop the communication channels. Traffic analysis attacks could be used to derive a wide range of information. It could be used to extract information about the identity of the communicating peers. It could also be used to profile the status or the habits of a specific target. And eventually, it could be used to extract the information transferred in a specific communication. Therefore, traffic analysis is a serious threat to the main objective of Tor which is preserving users and services anonymity.

Traffic analysis attacks against Tor network can be conducted in several ways and by different kinds of adversaries. In such attacks, the attacker usually has some sort of view on the network and can observe the traffic entering and leaving different network links and/or relays. By analysing and correlating traffic statistics in these different links, the attacker is, in theory, able to associate some network connections that seems correlated. By doing so, he could be able to identify the source of some anonymous communications. Traffic analysis against Tor generally aims to retrieve the identity of some anonymous user(s) or to retrieve the source and/or target of anonymous connections. This deanonymization process typically involves identifying an anonymous user among several competing connections.

In the past years, researchers have designed, developed and implemented some attacks to use against low-latency anonymous networks such as Tor. Some of the attacks described in the literature are dedicated to low-latency anonymity systems in general and some others are specifically designed to be carried against Tor. As the attacks designed for general low-latency systems are more theoretical than real attacks, they have not really been implemented or tested against real anonymous networks. Nevertheless, other attacks have been specifically designed for the Tor network and that is the kind of attack that will be developed in the remainder of this section.

As explained above, a powerful adversary that can observe each link of the anonymous network could identify the sources of anonymous communications. However, such an adversary doesn't exist since it is impossible to monitor all the network links and hosts at the same time. In the last decade, researchers have been working on reducing the set of network infrastructure to observe in order to perform a successful traffic analysis attack. As an omnipotent adversary doesn't exist, researchers focused on attacks that could be carried out by a more realistic one.

A first approach considers a powerful adversary being able to observe the traffic entering and leaving vantage Autonomous Systems (AS) or Internet Exchange Points (IXP) [33, 16, 25]. In the paper written by Murdoch *et al.*, researchers showed that IXPs are a single vantage point where traffic analysis can be performed. As the IXPs are in general designed to connect multiple ASes, the traffic leaving and entering the IXP generally comes from different ISPs generally located in different countries. And since Tor partially creates anonymous circuits based on the geographical diversity of the relays, IXPs are an excellent location to collect data traffic and to carry out a successful traffic analysis attack. In their results, they revealed, by the mean of a simulation, that a small set of IXPs are able to observe traffic passing through several Tor entry and exit nodes located in the UK. This means that an adversary which is able to observe the traffic passing through one or a small set of IXP is able to conduct successful traffic analysis attacks. They also showed that an adversary could complete a traffic analysis attack by analyzing the logs of the traffic monitoring systems installed on high-speed routers.

Two years later, Edman and Syverson studied Autonomous System adversaries and their potential threat against the anonymous network [16]. As the Tor network grows over time, with more and more volunteers operating new relays, it is more and more difficult and unlikely for a single AS to be able to look at the both ends of an anonymous connection. However, this last assertion was only an intuition and the researchers wanted to prove if it was correct or not. The results of their experiments, carried out on the Tor network, showed that a significant percentage of Tor circuits originate from a small number of ASes and a significant percentage of the traffic is destined to a small number of ASes. This means that a significant amount of Tor's traffic enters and leaves from a small number of ASes. Therefore, Tor is still vulnerable to AS-level adversaries despite the growing number of relays. They also showed that the /16 subnet separation enforced by the Tor path selection algorithm was not enough to imply ASes independence for the anonymous circuits. Indeed, they demonstrated that for 22% of the Tor circuits originating from different subnets, there are ASes which can observe traffic going towards a Tor entry node and from exit nodes to some popular destinations. They conclude that the growth of the Tor network has only a moderate impact on the AS-level path independence, meaning that, if indeed the growth of the Tor network makes more difficult for a single AS adversary to conduct a successful correlation attack, it doesn't make the attack unfeasible. Indeed, as revealed in study published by Wacek *et al.*, there were still, in 2013, 18% chances that a single AS appears in both sides of an anonymous connection [44]. As the number of relays have more than doubled from 2009 and 2013, the probability of seeing the same AS on both side of an anonymous connection didn't change that much [39].

In a more recent study, and following the steps of Edman and Syverson, Johnson *et al.* made a realistic and exhaustive analysis of traffic correlation attacks against Tor network [25]. In their new study, they proposed different attack scenarios where the adversary is either controlling one or more ASes, IXPs or malicious relays. In the first case, they showed that some users have over 95% chance of being deanonymized within three months against a single AS or IXP. They argue that some users are safer than others due to their geographical locations. In this case, safer locations means that an adversary would need more time to deanonymize a user hiding in these locations. Nevertheless, they also claimed that an adversary controlling more than one ASes or IXPs has a much higher compromising speed, including the safest locations. In their second attack scenario, the adversary commands malicious relays providing no more bandwidth than a substantial volunteer could do today. They revealed that this kind of adversaries could deanonymize regular Tor user within three months with over 50% probability and the odds raise to over 80% within six months. These results are quite pessimistic and worse than previously thought considering that, in the case of a AS-level adversary, a single company could own multiple IXPs or ASes and may have malicious intents against users passing through their networks.

The most recent paper regarding AS-level adversaries proposed an empirical study that measures the threat faced by Tor against traffic correlation attacks [35]. The researchers work with up-to-date maps of the Internet in addition with an algorithm that can predict which ASes are in a good position to conduct correlation attacks on the forward and reverse path of anonymous connections. Furthermore, they identified ASes that belongs to the same organization providing a clearer picture of today's network-level threats and adversaries. The results of their experiments show that up to 40% of all Tor circuits are exposed to traffic correlation attacks from AS-level adversaries. Moreover, 42% of all the circuits are vulnerable to organizations that own several ASes or IXPs and 85% of the circuits are vulnerable to state-level adversaries which can observe the traffic passing through their regional ASes.

At the moment, Tor doesn't implement proper protection to address the threat of AS-level adversaries who operate a single or multiple ASes or IXPs. However, researchers recently proposed refined relay selection algorithms to reduce to threat of AS-level attackers [3, 35]. These new algorithms aim to build safer anonymous circuits by avoiding situations where the circuit is vulnerable to a single or colluding AS(es) adversary.

In the remainder of this section, some end-to-end confirmation attacks are described to highlight the fact that in many attack scenarios two relays controlled by an adversary are enough to deanonymize communications.

In 2006, Øverlier and Syverson proposed a passive attack to reveal the location of a hidden server just by having a single hostile Tor node under control [37]. Their attack relies on the fact that any node of the network that claims to be stable can be used by the hidden server to build an anonymous circuit to the rendezvous point. Hence, if an attacker manages to take control of the first hop of the circuit from the hidden server to the rendezvous point, he can immediately determine the real IP address of the hidden server. Therefore, to carry out this attack, the attacker has to establish many circuits to the hidden server until one of his malicious Tor's relays is chosen as first hop of the circuit from the hidden server to the rendezvous point. In older versions of Tor, this attack was a serious threat to anonymity. Now, such an attack is much more complicated to set up since Tor chooses the adjacent node of the hidden server from a set of entry guard nodes.

In a paper written by Fu and Ling in 2009, an active end-to-end confirmation attack is presented to break Tor's anonymity [21]. The purpose of their attack is to confirm the fact that a given user (say Alice) is talking to another given user (say Bob). To demonstrate that, the authors made use of a malicious entry and a malicious exit onion router. The goal of the malicious entry node is to manipulate (duplicate, modify, insert or delete) data sent through a circuit while the malicious exit node is in charge of detecting the manipulation performed on the data. In all cases, an error is raised by the exit node during the decryption of the stream containing modified data. This behavior helps an adversary to confirm the fact that traffic entering the network at the entry node leaves the network at the exit node. Furthermore, since the entry node knows the IP address of the source and the exit node knows the IP address of the destination, the attacker can easily learn who is talking to whom.

Three years later, another active end-to-end confirmation attack has been followed to confirm the fact that it was possible to deanonymize two users (Alice and Bob) interacting with each other through Tor [27]. In this approach, the attacker performs a cell counter based attack which consists in two distinct steps. During the first step, the malicious entry node transmits data cells through the circuit between Alice and Bob according to a well-defined sending pattern (e.g. send three consecutive cells to simulate the binary digit "1" and simulate the binary digit "0" by only sending a single cell). During the second step, the malicious exit node tries to recognize the pattern. Once a match between the pattern at the entry and the exit node is found, the attacker is able to confirm that Alice and Bob communicate with each other.

In 2013, A. Biryukov, I. Pustogarov and R-P. Weinmann describe an attack capable of deanonymizing hidden services protected by the concept of entry guard [7]. In order to carry out their attack, the adversary needs to control at least two non-exit relays. One of these relays is intended to play the role of malicious rendezvous point and the other one is intended to serve as malicious entry guard between the hidden service and the rendezvous point. Then by generating a traffic pattern at the rendezvous point and correlating it at the malicious entry guard node, the attacker is able to retrieve the IP address of the hidden service. As for the attack discussed by Øverlier and Syverson, the tricky part is to build a circuit passing through the relays controlled by the attacker. Consequently, it may require many circuit creations to achieve such a configuration.

In a more recent article from 2014, a traffic analysis attack using flow records (NetFlow[2]) is described to identify the source of an anonymous communication [8]. Basically, the first step of the attack is to lure a victim to access a particular server controlled by the adversary while some NetFlow data about the traffic between the exit node and the server, as well as between Tor clients and the victim's entry node, are collected. Since the attacker has control of the malicious server, she knows from which exit node the traffic of the victim originates. Therefore, the big challenge for the adversary is to determine the identity of the anonymous client that interacts with the malicious server. To solve such a challenge, a pattern is injected by the server to make the victim flow distinguishable from all other flows. Then by computing the correlation among all entry-node-to-client flows with the server-to-exit-node flow, the adversary is able to detect who is the source of the anonymous communication since the flow of source will carry the same traffic pattern than the one induced by the malicious server.

Finally, a thesis has been written by K. Müller to evaluate the cost in terms of performance and network load of the existing protections against end-to-end confirmation attacks [34]. In that thesis, a timing attack is implemented to validate the fact that the

[2]A protocol designed by Cisco for collecting and monitoring network traffic.

traffic flow of a controlled client (i.e. a client that always chooses the same entry and exit nodes) can be identify with high accuracy. Like most end-to-end confirmation attacks, the timing attack described by Müller requires the adversary to control (or observe) the entry and the exit relays of a circuit. Once under control, the correlation among traffic flows passing through these nodes can be computed by the mean of a cross-correlation formula (cf. section 2.6.2). The key idea of the formula reused by Müller is to divide the time of observation into adjacent time windows and then count the number of packets received during each window size. By repeating this process for the duration of the attacker's stream, the adversary is able to recognize, for each flow, the sequences of packet count at both malicious end points of the circuit, and hence, she can distinguish her traffic flow from any other user's one.

2.6.2 Traffic correlation techniques

As seen in the previous section, lots of effort has been made in the last years to either deanonymize Tor's traffic or to mitigate the threat of AS-level or relay adversaries. In most of the papers regarding traffic correlation attacks, the researchers implemented a two-step process to carry them out successfully. The first step of the process mostly consists of getting into a situation where the attacker has some view on the anonymous network and is able to gather interesting information from the transiting traffic. The second step generally involves analysing and correlating the obtained traffic to unveil, totally or in a part, user's anonymity. As described in the preceding section, there exist many ways for an attacker, whatever its size and resources, to set up a situation where he is able to observe anonymous traffic. However, many of these studies rely on the same techniques to eventually correlate some traffic in the interest of revealing the sender and receiver of some communications. This section aims to introduce the most used correlation techniques in the literature as well as their respective benefits and drawbacks.

A first approach has been described in 2005 by Murdoch *et al.* to conduct a traffic analysis at low cost against Tor [31]. In their paper, Murdoch and al. proposed an attack set-up where the adversary manages a corrupted server that inject a pattern in the traffic flowing to the victim. The shape of the injected pattern changes in function of the server sending rate. The adversary has also a latency probe under control that allows him to check the load of a given relay at a given time. The goal of the experiment was to check whether or not the load of a given relay is correlated with the injected traffic pattern sent by the corrupted server. The researchers proposed a simple correlation technique to unveil if a relay was carrying the anonymous modulated stream. Their correlation formula can be defined as the sum of the multiplications of the template formed by the modulated traffic and the probe data for every sampled time t. The template formed by the modulated traffic, S(t), is known and defined as :

$$S(t) = \begin{cases} 1 & \text{If the corrupted server is sending at sample number t} \\ 0 & \text{Otherwise} \end{cases}$$

The probe data is expressed as $L(t)$ and measures the latency (in μs) of a given relay at sample time t. The normalized version of the probe data is expressed as $L'(t)$ and is calculated by dividing $L(t)$ by the mean of all samples. They finally defined c as the correlation between the injected pattern and the latency of a given relay :

$$c = \frac{\sum S(t) \times L'(t)}{\sum S(t)} \tag{2.1}$$

The correlation index c reflects the strength of the correlation between the two data streams. A high correlation index indicates a strong relationship between the two streams whereas a low correlation index reveals that the two streams are quite different from one another. An obvious drawback of this formula is related to the fact that it doesn't take into account the effect of latency. As the modulated traffic travels through the network, it is more than likely that it will be reshaped due to transmission delays or packet loss. Therefore, if an attacker tries to correlate the reshaped traffic with the initial one, she will get distorted results if she doesn't take into account the effect of network latency.

As the last correlation formula suffers from a lack of precision when it comes to correlate delayed traffic, researchers put some efforts in finding more powerful ones. Thankfully, O'Gorman *et al.* wrote an article that presents and compares the most used correlation techniques in the literature [36]. The remainder of this section is dedicated to introduce the correlation techniques presented in their article.

The first approach known as *packet counting* is a simple technique that consists of counting the number of packets in a traffic flow. By comparing the amount of packets obtained for a given stream with the amount of packets presents into each stream from a set, an adversary is able to find which stream from the set is potentially the same stream as the initial one. As already stated above, some factors (e.g. latency, throughput) may influence the number of packets observed for the same stream entering and exiting the network. Therefore, simply checking the equality between the number of packets of two streams could be inaccurate. Consequently, in practice, to measure the similarity between the packet count x from stream X and the packet count y from stream Y, a distance measure is preferred.

$$d(x, y) = \sqrt{(x - y)^2} \tag{2.2}$$

The formula 2.2 implies that the more the difference between packet count x and y is small, the greater the similarity between stream X and Y is. This formula can also be used to evaluate the similarity between two streams of Tor cells. Indeed, instead of counting the packet flowing on the two streams, the adversary in control of Tor relays could count the cells transiting on Tor circuits to estimate the similarity of two circuits.

The second approach described in the paper written by O'Gorman is the *cross-correlation coefficient technique*. This technique is used to measure the similarity between two series as a function of the delay of one relative to the other. More precisely, this technique consists in extracting a sequence of values from a data stream, to afterwards comparing it with the sequences obtained by other streams in the network. The sequence of values for a given stream is extracted by slicing the stream into small windows of size W, and then by counting for the duration of each window, the number of packet received. Hence, the correlation between two streams is computed as follow:

$$r(d) = \frac{\sum_i((x_i - \mu)(x'_{i+d} - \mu'))}{\sqrt{\sum_i(x_i - \mu)^2}\sqrt{\sum_i(x'_{i+d} - \mu')^2}} \tag{2.3}$$

In the above formula, “The two streams being compared are x and x' with delay value d. This delay value is the time required for the stream to transit the network. x_i is the ith packet count of stream x and x'_i is the ith packet count of stream x'. μ is the average of packet counts in stream x and μ' is the average of packet counts in stream x'.” (O’Gorman, 2009, p. 2025)

The result of $r(d)$ ranges from -1 to 1. When 1 is returned, it means that a linear equation perfectly describes the relationship between the two streams being compared (all data points lie on a line inclined at 45^o). On the contrary, a value of -1 involves that all data points lie on a sloped line of -45^o, meaning that the increase of the value of one variable results in a decrease of the value for the other variable. Finally, a value of 0 indicates that there is no linear correlation between the two streams. In practice, it is very rare to see such values (-1, 0, 1). Most of the time, you will get a result somewhere in between those values (cf. figure 2.4). The following table provides some guidelines for interpreting the cross-correlation coefficient results [1] :

	Coefficient, r	
Correlation strength	Positive	Negative
High	0.5 to 1.0	-0.5 to -1.0
Medium	0.3 to 0.5	-0.3 to -0.5
Low	0.1 to 0.3	-0.1 to -0.3

Table 2.1: Categories of cross-correlation coefficient

Three graphical examples of cross-correlation coefficient are displayed below. The first graph depicts a high positive correlation between two streams, the second depicts a medium positive correlation and the third represents a situation where there is no correlation at all.

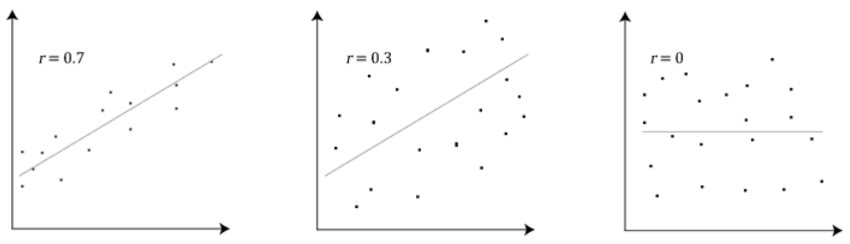


Figure 2.4: Examples of cross-correlation coefficient

During their experiments on the real Tor network and on simulation environments, the authors have confirmed that such correlation techniques offer a reasonable degree of accuracy in ideal conditions, even with the simple byte counting strategy [36]. Moreover, their results have also showed that an addition of extra "noise" in the network considerably decreases the efficiency of the correlation. The "noise" in the network can be due to network latency, packet loss or reordering. Therefore, it could be interesting to investigate what kind of noise and how much noise is needed to mitigate the feasibility of these kinds of attacks.

2.6.3 Developed end-to-end correlation attack

As explained in 2.5, clients send EXTEND cells to notify the current last hop of the circuit to extend the circuit to another relay. Upon arrival of such a cell, a relay checks if a TCP connection towards the next relay has already been established. If the TCP connection doesn't exist yet, the relay creates a new one with the next hop. Tor cells associated with the circuits that transit between the two relays are transferred over this (new) connection. Tor makes use of a single TCP connection to connect two Tor entities that need to communicate. This means two things. First, there exists only one TCP connection at given time to connect two Tor entities. Secondly, by using TLS, the connection provides both encryption and authentication for all the cells passing through it.

In Tor implementation, different types of connections are used to connect the different types of network entities. OR (Onion Router) connections are dedicated to connect Tor relays while Edge connections are used to connect Exit relays to external peers. The OR connections represent the TCP connection over TLS described above. The Edge connections between Exit relays and remote servers could either use TLS (HTTPS) or not. One connection between two Tor entities multiplexes all the circuits that the two entities directly share together. Therefore, all the traffic that goes through these circuits travels through a unique TCP/TLS connection.

The internal cell processing for Tor relays is depicted on figure 2.5. Relays listen for incoming data on their connection (called channel in Tor implementation) and store them in an internal input buffer. Once the data received on a channel makes full TLS record, the record is decrypted and the extracted cells are sent to their respective internal circuit queues. There exist as many circuit queues as there are distinct circuits on a relay. Once a channel is ready to send, it selects one circuit queue from all the circuit travelling through that channel and writes the content of the queue into the output buffer of the channel. When the output buffer of a channel has gathered enough data, it sends all the data to the next relay.

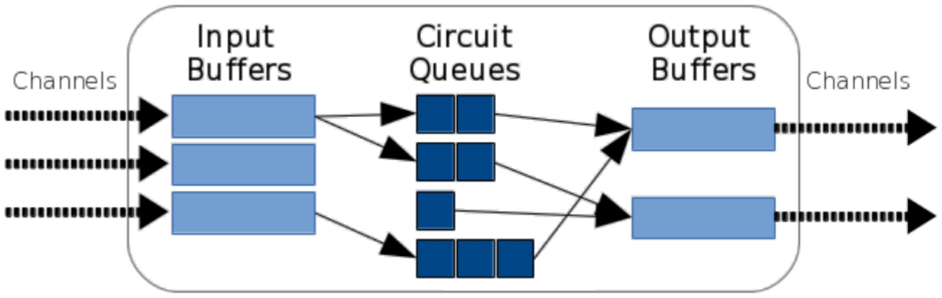


Figure 2.5: *Internal architecture of cell processing inside a Tor relay. All cells from a given relay arrive on a single channel, and are then de-multiplexed to circuit queues for processing. Each circuit queue is emptied into the channel associated with the next hop relay.* Figure and caption taken from the paper written by Rob Jansen *et al.* [24]

In order to correctly forward the cells from one entity to another, Tor relays rely on the circuit IDs it reads in each cells. As the circuits are incrementally extended one relay at a time, all the relays from a given circuit are aware of its predecessor and its successor in the chain. The figure 2.6 gives an example of a client sending an HTTP GET request over the Tor network. In the context of this figure, the client has already created a circuit with the 3 Onion Routers and is ready to send. Therefore, the connections between the different routers have already been established and the encryption keys have already been distributed. The circuit and connections IDs are also known from the concerned parties and are shown on top of each entity (OP and OR). Now, imagine that the client process behind the Onion Proxy wants to send an HTTP GET request to the remote website. To do so, the client process needs to ask the Onion Proxy to create a remote connection with the website. Upon reception of such a request, the Onion Proxy generates a new random stream ID (e.g. StreamID=10) and attaches the stream to the most recent circuit (e.g. CircID=1) it has created. The OP then creates a new relay cell with the command bits set to Begin. The cell's circuit and stream IDs are respectively set to 1 and 10. The IP address and the port of the remote website are written in the data bits of the cell. The OP finally adds some padding if necessary, encrypts three times the last 509 bytes of the cell and sends it to the Entry relay on the shared connection (ConnID=48). Note that, while transiting on the TCP connection, the 512 bytes cell is encrypted and wrapped in a TLS record. Upon arrival of the cell at OR1, the relay extracts the cell's circuit ID (it's not encrypted) and decides what to do with it. As the cell is not intended to him, the relay decrypts one layer of encryption using its circuit key (Blue key) and changes the cell's circuit ID to 2. It then forwards the cell to the next hop of the circuit (OR2). It is interesting to note that OR1 doesn't use the same circuit ID to communicate with the OP and OR2. Indeed, if the same circuit ID is used along all the circuit, an attacker which controls both the Entry and the Exit would be able to directly match the two flows. In the same way, Tor uses at least 3 relays to create a circuit because if only two were used, an attacker would have no difficulty in finding the identities of the peers since it would know the corresponding circuit IDs and remote IP addresses of the clients. When the cell arrives at OR2, the same transformations as for OR1 are applied and the cell is forwarded to OR3. When the cell finally reaches OR3, the relay decrypts the last layer of encryption and starts executing the Begin command. It then establishes an exit TCP connection with the chosen website. If the TCP handshake went well, the OR3 creates a new relay cell with the command bits set to Connected and sends it to the OP. When this last cell arrives at the OP, it notifies the user process that the connection has been established and that the process can start sending data. Upon arrival of the cell containing the GET request at OR3, the relay knows what to do with it regarding the stream ID (=10) attached to the cell. The response data are forwarded backward in the circuit using the strategy described before.

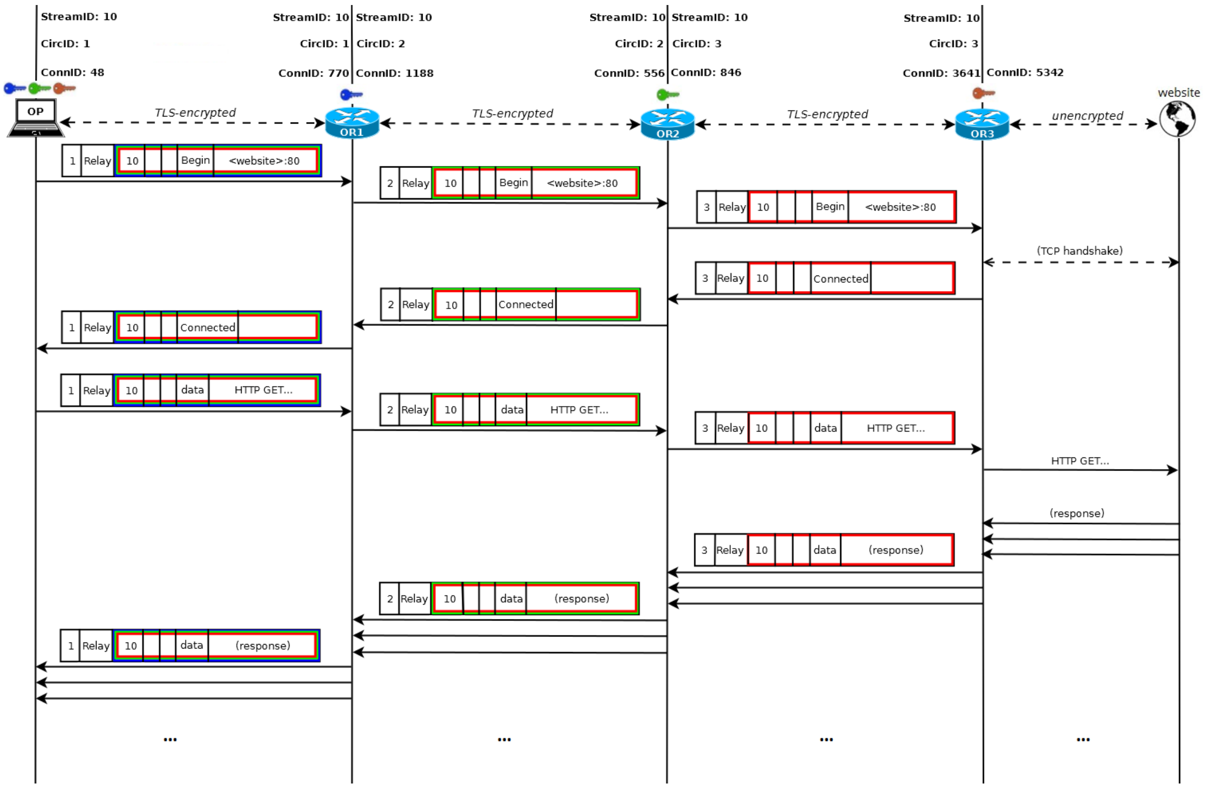


Figure 2.6: Example of HTTP GET request sent over Tor using the Onion Proxy.

On Figure 2.6, the entry node knows the identity of the client (i.e. its IP address) and the exit node knows the identity of the remote server. But there's no way for the entry node to discover the final destination of the cells as their content is encrypted. The exit node, on the other hand, is not able to discover the origin of the request. However, as Tor tries to provide low-latency communication, the shape and the timing of the traffic flowing on the connection between the client and the entry node are susceptible to look like the same as the traffic flowing on the connection between the exit and the remote server. In the same way, the traffic flowing from the exit to the middle relay should be similar to the traffic observed at the entry relay to the OP.

Based on those behaviours, two approaches can be followed by the adversary to perform a traffic analysis attack. One approach consists of analyzing the traffic of the TCP/TLS connections between the OPs and the entry relays and between the exit relays and the remote servers. Such a scenario is depicted on Figure 2.7.

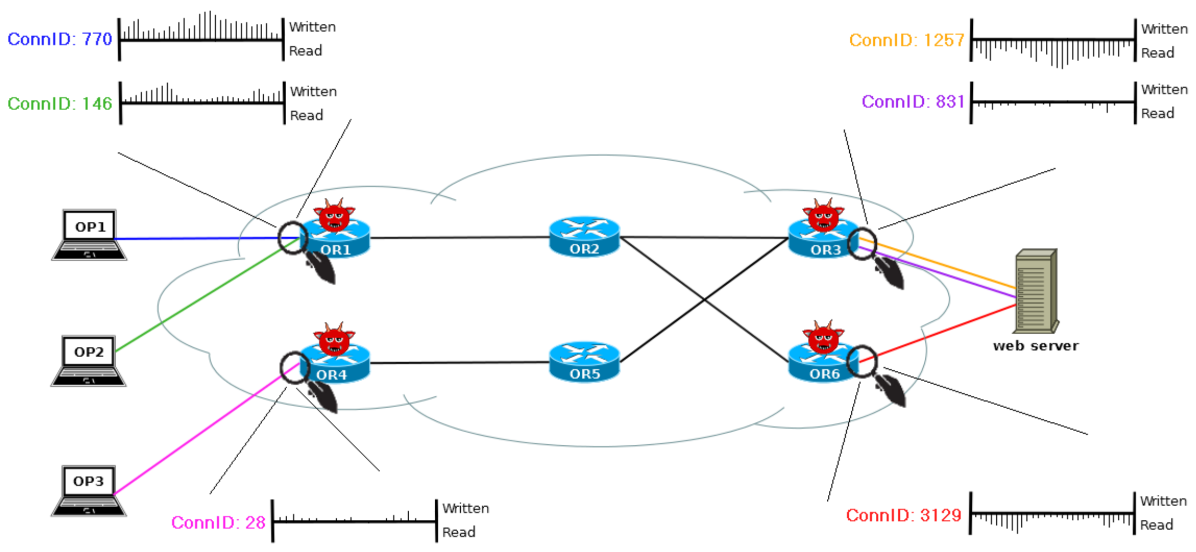


Figure 2.7: Example of TCP connections observable by a relay adversary who controls entry and exit relays in the Tor network.

Figure 2.7 sketches an attack in which an adversary controls some entry nodes (OR1, OR4) and some exit nodes (OR3, OR6). Considering that the blue flow corresponds to a bulk download from the web server, the green flow to an SSH session between OP2 and the web server and the pink flow represents an IRC session. Then, a possible objective for the adversary would be to recognize that the blue flow at the entry nodes matches with the orange flow at the exit nodes. If she succeeds then it will mean that the adversary has broken the anonymity provided by Tor since she will know with whom a given client interacts (e.g. OP1 $\leftrightarrow$ web server). To reach such an objective, the adversary has to compare the pattern of the blue connection with the pattern of all observable exit connections (in this case, the purple, orange and red ones). The pattern of each TCP connection is built by looking at the amount of data transferred per unit of time (bytes/sec). In Tor, the generation of such a pattern can efficiently be obtained by using a built-in module named the Tor controller. This module allows users to interact with the Tor process using the Tor Control Protocol. It can be used, for example, to track the bandwidth consumed by every connection on a relay at a given time. By default, the Tor controller creates a log file on the relay to which it is connected. This log file contains all relevant information about specific events occurring in the network. In the case of this approach, the CONN_BW event is the interesting one since it provides the bytes read and written per second per connection.

It is important to point out that this kind of attack can be applied even if the adversary doesn't control Tor relays. Indeed, the adversary could eavesdrop TCP traffic outside of the Tor network and could be able to retrieve the pattern of each connection without the help of the Tor controller. However, this approach is quite difficult to put into action for several reasons. First, Tor implements obfuscation techniques that can be used to dissimulate Tor traffic flowing over the TCP/TLS connection. With such obfuscation techniques, it becomes almost impossible for an adversary to determine whether an observed TCP connection is carrying normal or Tor traffic. Another drawback is that the TCP connections observed at the entry nodes carry cell traffic and can multiplex several circuits. Therefore, the traffic observed at the entry won't have the same distribution of TCP packet than the connection that connect the exit relay and the remote peer. Meaning that the two traffic flows won't have the same patterns and will be more difficult to match. Furthermore, even if the entry connection only carried the cells of a unique circuit, the exit connection should last long enough in order to be able to efficiently correlate the flows. Indeed, if a client requests a web-page, the exit connection will at worst last several seconds which is not enough for an adversary to accurately determine the corresponding flows. This is not the case for the next approach that is discussed below. This new approach relies on circuits, an abstraction defined by Tor, to retrieve the patterns to compare during the correlation step.

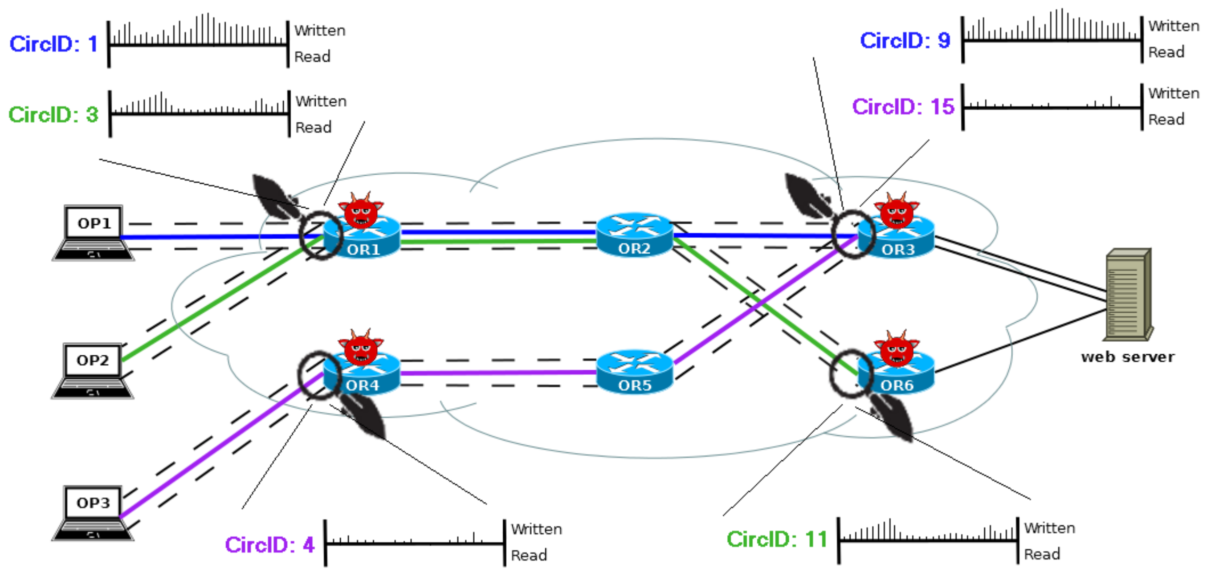


Figure 2.8: Example of cell traffic patterns observable by an adversary who controls entry and exit relays in the Tor network. The colored lines represent Tor circuits and the doted lines represent the TCP connections that carry the circuits. Black lines represent connections to a remote peer, a web server in this case.

Figure 2.8 depicts the second approach to remove the anonymity of communications within the Tor network. In that scenario, the adversary defines a new type of pattern by counting, for each entry and exit relays under her control, the number of RELAY cells sent per circuit per unit of time. Knowing that the lifetime of a circuit is roughly 10 minutes, each pattern built indicates every second and for a total duration of approximately 10 minutes the amount of RELAY cells exchanged between two peers. As all entities (i.e. OR and OP) communicate with each other by exchanging cells over previously established circuits, once the adversary has access to the pattern of each of those circuits, she can use, as in the above approach, some correlation techniques to bring out a fairly strong correlation between a pattern belonging to a circuit used by a given client at the entry side (i.e. the pattern of the circuit between the OP and the OR) and the pattern of another circuit at the exit side (i.e. the pattern of the circuit between the exit relay and the middle relay). In the figure below, the straight color lines represent circuits established by the user's OP. Each circuit is included between two dashed lines representing a TCP connection. The straight black lines from the exit relays to the web server describe TCP connections carrying the data from the web server (at this point of the network, no circuit abstraction exists). Like in the approach described above, a possible attack scenario might be to compare the correlation value of the blue flow with

CircID 1 at the entry side with all observable patterns of all circuits at the exit side (i.e. CircID 9, CircID 15 and CircID 11). If she succeeds to highlight the fact that CircID 1 is highly related to CircID 9 then she is able to infer that OP1 is interacting with the web server (the source and destination of the blue flow are unveiled). Compared to the other approach, this method requires the adversary to control at least one entry and one exit relay since the patterns are built by analyzing the traffic flowing from the circuit point of view which is a particular abstraction defined in the Tor network.

The end-to-end correlation attacks developed in this document are based on the second approach described above. In order to get the best results, the developed attack only focuses on the correlation of flows on the backward channel because this channel carries the responses to client requests. Consequently, it will contain much more data than the forward channel. As for the first approach, the Tor controller provides a convenient way to build the pattern of the traffic flowing through each circuit. Indeed, the CELL_STATS events, which is emitted every second by the Tor controller, can be used by the adversary to retrieve the cell traffic patterns for each circuits on each relays. Then, by accessing and analyzing the Tor controller logs, the adversary is able to recover all the information she needs to perform correlation attacks against Tor circuits. By accurately correlating and matching related circuits, the adversary reveals which Onion Proxy (IP address) is linked to which remote exit connections. This is conceivable considering that the adversary can determine which circuits are associated to which TCP/TLS connections. This is the threat model that is considered and examined for the remainder of this document.

Chapter 3

Simulating the Tor network

This chapter begins by presenting a brief overview of some existing Tor experimentation tools that can be used to set up an attack and evaluate its performance against the Tor network. As explained in the paper written by Shirazi *et al.* [41], different tools exist and they all have their own specifications (e.g. maximum number of relays supported, resources requirements, ...). Section 3.2 describes in more details the specifications of the chosen tool while section 3.3 presents the contribution to the Shadow project. Section 3.4 contains a complete description of the topologies and the types of traffic that were simulated to evaluate the efficiency of the attacks depending on the user model. Section 3.5 explains how the circuit construction protocol has been altered to easily verify the results. In other words, the circuit construction protocol has been altered to unveil which circuits observed at the entry matches which circuit at the exit. Finally, the last section of the chapter compares the performances of our simulations with the performances of the live Tor network.

3.1 Tor experimentation tools

When it comes to carry out researches on the Tor network, an obvious approach is to directly apply the changes induced by the research topic on the live Tor network. However, directly working on the live Tor network is not recommended since it might inadvertently harm live users performance and anonymity. Therefore, a better approach consists of using experimentation tools to build small-scale private Tor networks. At time of writing, five well-known experimentation tools are available to help researchers to simulate or emulate downscaled Tor networks [41]:

Shadow is a discrete-event simulator designed to run real applications like Tor on a single machine. According to its developer, it combines the accuracy of emulation with the efficiency and control of simulation, achieving the best of both approaches [23].

ExperimenTor is a network emulator designed by Bauer *et al.* in 2011. Despite being a valuable testbed for Tor experimentation, it is no longer maintained [5].

SNEAC (Scalable Network Emulator for Anonymous Communication) is a network emulator dedicated to run unmodified Tor code and that can scale to thousands of nodes. It has a very high hardware requirements since it runs all applications including Tor in real time [43].

TorPS (Tor Path Simulator) is a simulator designed by Johnson *et al.* to simulate the relay selection process during the circuit construction. It has only been developed to provide results for the circuit construction process. Therefore, it is only suitable for researches on improving or changing the path selection algorithm in Tor [25].

COGS (Changing of the Guards) is a simulation framework that helps to study the effects of Tors entry guard selection on user privacy. COGS is no longer maintained [18].

Two of the tools described above have been developed for specific-purpose. TorPS has been implemented to evaluate the circuit construction process and COGS for evaluating the effects of Tor entry guard selection on user privacy. As this thesis does not focus on the circuit construction process nor on the entry guard selection, TorPS and COGS can immediately be removed from the list of suitable experimentation tools. Among the remaining tools, two of them are categorized as being emulators (SNEAC and ExperimenTor) and one of them as being a simulator (Shadow). Knowing that emulators require much higher hardware resources and are more difficult to set-up than simulators, Shadow emerged as the best experimentation tool to run, on a single machine, a downscale version of the live Tor network and to set up the developed attack presented in the previous chapter. The next section introduces the reader to the main features of the chosen experimentation tool, Shadow.

3.2 The Shadow simulator

As previously mentioned, Shadow is a discrete-event simulator that allows to run a downscaled version of the live Tor network on a single machine [22, 23]. It provides accurate results by natively executing the real Tor software within a virtual network topology. Furthermore, simulations ran with Shadow can be tuned to model the negative effect of the latency, the packet loss and the jitter that happen on the links connecting different network entities. To take those factors into account, Shadow relies on geographic clustering by country. In this clustering approach, there exists one network vertex per country. Each vertex is assigned to an upstream and downstream bandwidth and is connected with several or all the other vertex with network edges. Each edge is assigned to a latency, a packet lost and a jitter rate. The whole thing forms a complete graph that tries to approximate the real Internet topology. When a node is simulated in a Shadow simulation, it is assigned a geo-location (i.e. a country code like "US") and when two nodes communicate with each other, their communication is affected by the latency, packet loss and jitter that exist between the two locations. This allows a reasonable compromise regarding the fact that replicating the real Internet topology (ASes, backbone, etc...) would be extremely difficult and inefficient. Finally, Shadow is provided with a set of python scripts that allow researchers to easily generate their own Tor network topology as well as parsing and plotting some important statistics from the Shadow and the traffic generator messages (e.g. network throughput over time, statistics of downloads performed by clients, etc...).

3.3 Developed plugin

By default, Shadow comes with a basic traffic generator, called **tgen**, that allows to simulate simple users traffic behaviours such as basic bulk download and web surfing. Unfortunately, this traffic generator is not really suitable to generate more complex traffic behaviour such as IRC and SSH communications. To overcome this issue, a new plugin was designed to directly replay IRC and SSH traffic traces within the simulated Tor network. The plugin is in charge of creating two distinct peers that replay TCP traffic saved in pcap files. One peer acts as a client and the other one as the remote peer of the connection. The client replays the traffic intended to the remote peer and, conversely, the remote peer replays the traffic intended to the client according to the content of the pcap files. Any TCP traffic that involves only two peers can therefore be reproduced in Shadow. Moreover, the Socks proxy handshake has been implemented to allow the peers to send their traffic over the simulated Tor network. More detailed information and implementation are available on the Shadow plugin extras Github repository [10]

3.4 Simulations set-up

In order to study the feasibility and the performance of the developed attack depending on the user model (i.e. bulk, web, SSH, IRC), four topologies, one per traffic type, were generated using a modified version of the script **generate.py** included with Shadow. This script ensures that each simulation mimics as closely as possible the behavior of the live Tor network. To achieve this objective, it relies on the following archived Tor data:

- The Alexa top 1000 list of one million most popular web sites.
- The Tor network consensus, relay descriptor and extra information.
- Some Tor metrics related to clients.

Running this script with all mandatory arguments (i.e. number of authorities, relays, clients, servers wished) outputs an XML file specifying the structure of the topology. More precisely, the XML file specifies when each virtual node is created and what software each virtual node runs. It also indicates the proportion of entry, exit and middle relays in the simulation.

As previously said in the chapter, Shadow runs each simulation on a unique machine. As a result, the size of the simulated network is directly limited by the hardware resources available. Therefore, in order to keep the simulations results consistent and, also to keep a reasonable execution load and time for the simulations, an estimation of the number of relays, servers and clients per topology has to be evaluated such that the total network load is the same between all topologies (bulk, web, IRC, SSH) and the memory requirement for a topology simulation never exceeds the memory available on the machine.

To satisfy the above constraints, the amount of relays and servers has been fixed to 50 for each topology and the bandwidth of each server has been set to 100 MiB/s to reduce the risk for the servers to become the bottleneck of the communications. Although the generated topologies were different, they follow the same distribution of relay flags :

- 9 Guard relays
- 2 Exit-Guard relays
- 33 Middle relays
- 4 Exit relays
- 2 Directory Authorities

Subsequently, the amount of clients per topology has been set in order to generate the same total network load in all topologies. This has been made by first creating a web topology with 500 bulk clients. This is the amount of bulk clients that consumes approximately all the available RAM of the machine. Then, based on the total throughput in the bulk topology, a web topology with 500 web clients, an IRC topology with 500 IRC clients and an SSH topology with 500 SSH clients have been set up. As Web, Irc and ssh clients generate less traffic than bulk clients, a few highly loaded bulks have been added in those topologies to compensate the lack of network load. In order to obtain reliable results, two topologies of each kind have been generated and simulated, giving 8 simulations in total and accumulating 1000 clients of each kind.

In the remainder of this section, each client behavior is described to give an idea of the behavior of each topology. Note that the web surfing and the bulk traffic were simulated using the Shadow traffic generator while the IRC and SSH ones have been generated using the plugin discussed before (see Section 3.3).

The behavior of the web clients is set in a way that they request web-pages from a set of servers, wait for the responses and then they randomly wait from 1 to 70 seconds before requesting another page again. This behavior is in accordance with researches assessing that almost 80% of web users don't stay longer than 70 seconds on the same web-page before requesting a new one [28]. As shadow does not allow the clients to request files of random size, they are configured to request 320 KiB files from the remote servers. Regarding the bulk clients, they behave quite differently from the web clients, since, when started, they choose a server at random and start downloading a single big file of $\approx$ 15 MiB from the chosen server.

In order to reproduce messaging and remote-control traffic in the simulations, some IRC and SSH traces were recorded using Wireshark and replayed in Shadow using the plugin described in section 3.3. The IRC traces are used to simulate peers communicating over Tor with instant-messaging application while the SSH traces records typical remote-control behavior such as sending commands or files, editing text, compiling programs on a distant server.

Furthermore, in the interest of comparing the simulations performance with the live Tor network, each simulation includes some special clients who act as performance measurement probes. They are configured to randomly and repeatedly download either 50 KiB, 1 MiB or 5 MiB files from the different servers. Each probe records some statistics about its download statistics (e.g. time to first byte, time to last byte). These statistics are compared to the ones of the Tor network to ensure that the simulations adopt a consistent behavior. This subject is examined in the simulation analysis section.

All simulations described above have been generated with Tor metrics (consensus, relay descriptor, . . .) from February 2016. Moreover, each simulation has been configured to simulate one hour of Tor traffic. In all topologies, the clients are started after half an hour as it takes roughly 30 virtual minutes for the network to be fully bootstrapped. To give an order of idea, it takes approximately 7 hours to generate one virtual hour of Tor traffic on a machine having 16 GB of DDR4 RAM coupled with an Intel i7 CPU. Please also note that each of the four topologies (web, bulk, IRC, SSH) has been simulated twice, and for each kind of topology, the results of the first simulation have been aggregated with the results of the second in order to form four distinct topologies of 1000 identical clients (1000 web clients, 1000 bulk clients, ...). All results presented in section 3.6 and chapter 4 are based on those topologies.

3.5 Circuit mapping

To the extent of analyzing and validating the results of the developed attacks, the Tor implementation employed in the simulations has been modified to unveil the clients anonymity. The circuit construction protocol implementation has been altered to adjoin the IPv4 address of the circuit initiator to the CREATE and EXTEND cells that are sent to create a new circuit. Likewise, the unique circuit ID generated and used by the initiator of the circuit is appended next to the IP address. Upon arrival of a CREATE or an EXTEND cell at a relay, this one extracts and saves the information about the initiator and pursue the rest of the protocol as normal.

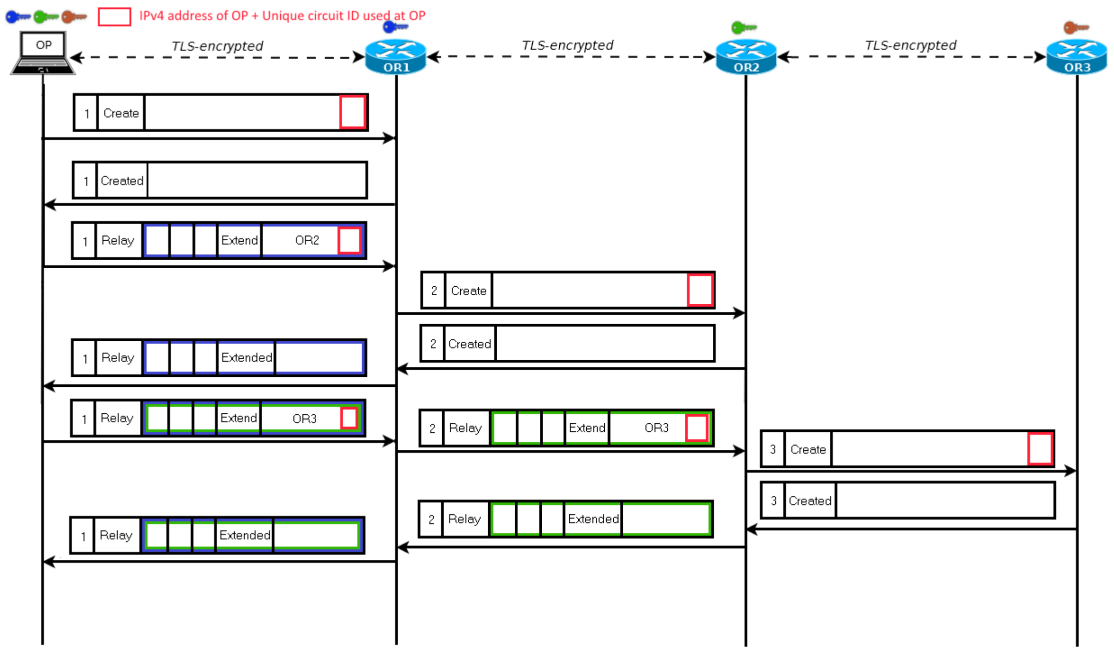


Figure 3.1: Modified Tor circuit construction protocol. The red square represents the IPv4 address of the Onion Proxy and the unique circuit ID that will be used by the OP to identify this circuit.

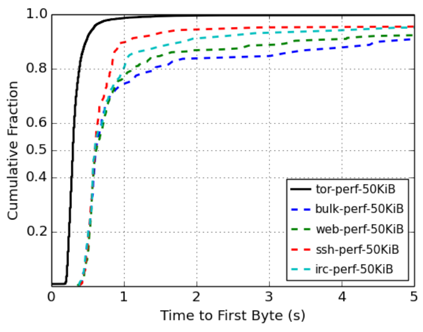
Figure 3.1 depicts how the new circuit construction protocol carries the client information from one relay to another. As the combination of the initiator's IPv4 address and circuit ID is unique for a given simulation, it is possible to map all of the circuit to their origin. The mapping will allow to verify and validate the results of the correlation attacks. The complete implementation of the modified Shadow Tor plugin is available at [9].

3.6 Simulation analysis

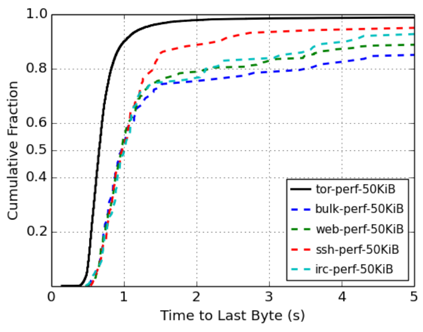
This section is dedicated to examine the performance of the simulated networks compared to the live Tor network. As the efficiency of the developed correlation attack is directly affected by the performance of the network, it is crucial to ensure that the Shadow simulations behave as closely as possible to the live network. Indeed, in a way, the more the simulations are accurate, the more the results of the attacks are reliable.

The simulations have been configured to model a reduced Tor network using real Tor metrics collected in February 2016. Therefore, the client performance measured in the test environment must be compared with the performance measured in the live Tor network at that time. The performance of the live Tor network are given by torperf, a tool that monitors Tor's performance by downloading files of different sizes (50 KiB, 1 MiB and 5 MiB) from randomly chosen server located all around the world. The torperf tool acts as typical Tor client and is used, in part, to measure the time to download files through Tor. The torperf clients also measure the time to download the first and last byte of payload for a given request. Lots of torperf clients monitor the Tor network everyday and the complete history of their measurements is publicly available on the CollecTor web-site [38]. The aggregation of all these measurements gives a good indication of the network throughput and responsiveness over time. In the same way, torperf clients have been included in the simulations and their measurements are compared to the performance of the live Tor network.

The time to download the first and last byte of payload are respectively a good indicator of the network latency and throughput. Figures 3.2, 3.3, 3.4 expose the time to download the first and the last byte of payload when downloading 50 KiB, 1 MiB and 5 MiB files. The dark lines represent the measurements of the live Tor network and the dotted lines represent the simulations measurements for the different simulations. As can be seen on the graphs, the time to download the first byte of payload in the live network is quite similar to the one observed in the simulations. This means that the simulations approximate adequately the latency of the live network. On the other hand, it seems that it takes longer in the simulations to download the files. This is the sign that the simulated network was too heavily loaded in traffic. However, all the simulations behaved quite the same way whatever the type of client that was simulated. This ensures that the results of the correlation attacks for a given simulation will be coherent with the results obtained with other simulations.

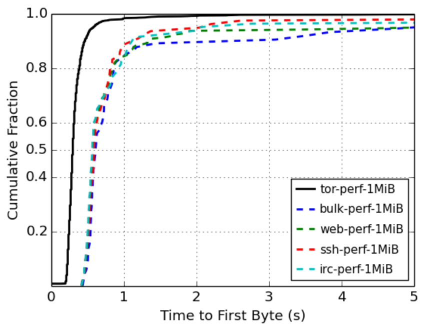


(a) 50 KiB - Time to first byte

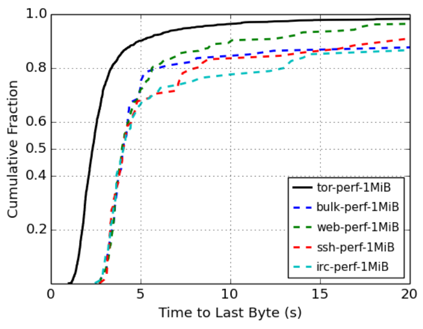


(b) 50 KiB - Time to last byte

Figure 3.2: Performance comparison between the live Tor network and the Shadow simulations when downloading 50 KiB files.

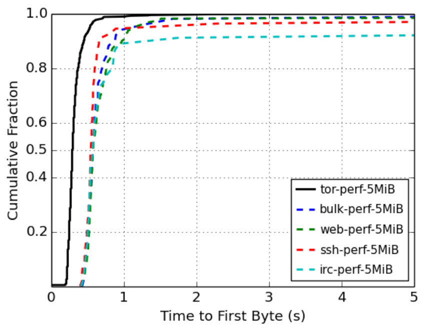


(a) 1 MiB - Time to first byte

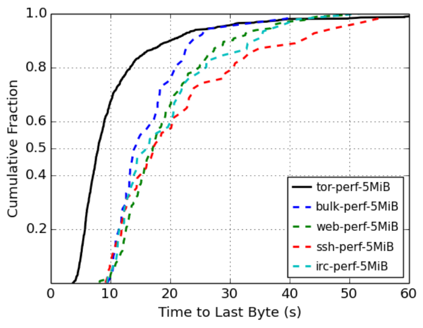


(b) 1 MiB - Time to last byte

Figure 3.3: Performance comparison between the live Tor network and the Shadow simulations when downloading 1 MiB files.



(a) 5 MiB - Time to first byte



(b) 5 MiB - Time to last byte

Figure 3.4: Performance comparison between the live Tor network and the Shadow simulations when downloading 5 MiB files.

The conducted analysis reveals that the reduced size Tor networks simulated with Shadow behave quite like the live Tor network, at least until the 80th percentile. More precisely, the simulated networks seem to approximate very closely the latency observed in live network. However, the simulations seem to be slightly too loaded in traffic but it is not a big deal considering that the simulated communications need to be negatively affected at least as in the live network. This ensures that the results of the developed attacks will be consistent and reliable and that they should approximate as closely as possible the behaviour that could be observed in the live network. In addition to that, the performance analysis gives to the potential adversary some insights on how to conduct her attack. Indeed, as the time-to-first-byte gives a good approximation of the average RTT observed in the network, an attacker could conveniently predict the network delay she needs to use for a more successful correlation attack. If it takes half second for the request to reach the remote server and to receive the first byte of payload at the client, it takes roughly 250 ms for the first bytes to travel from the server to the client. Therefore, the bytes streams observed at the exit relays are delayed, on average, for approximately 250 ms from the streams seen at the entry relays. The attacker could use this value to refine her attack and to raise the correlation accuracy.

Chapter 4

Threat analysis

This chapter aims to evaluate and measure the threat of an adversary that conduct end-to-end correlation attacks against the Tor network using malicious relays. The feasibility and the effectiveness of the attacks are assessed against the reduced size Tor networks described in chapter 3.

4.1 Experimental set-up

The second threat model introduced in section 2.6.3 assumes an adversary that controls or observes the traffic of at least one entry and one exit node inside the Tor network. If a user communicates to a remote server outside the Tor network using a Tor circuit that passes by the malicious entry and exit nodes, the adversary should be able to determine if the observed cell flow at the entry matches any of the flows observed at the exit relays. In the case the adversary finds two matching circuits and that the circuits are indeed related, then, the adversary has correctly unveiled the user's anonymity. As described in previous sections, the cells that flows to and from the Tor nodes can be observed and recorded using the Tor controller. With the modified Tor implementation, the Tor controller also records the IPv4 address of the initiator as well as the circuit ID used by the circuit initiator.

The implemented attacks make use of the cell traffic recorded by the Tor controller on both the entry and exit nodes to correlate Tor circuits. The Shadow simulations have been configured to save the Tor controller records of every entry and exit nodes, and, it is based on these records that the correlation attacks are performed and evaluated. The operation of extracting, analyzing and correlating the cell traffic patterns saved in the Tor controller logs is actually performed by a program written in Python. The program is first in charge of parsing Tor controller logs to make a complete mapping of the simulated network. The mapping consists of extracting and storing in memory all the data necessary to conduct and validate the attacks. This includes :

- Client and relay identities : Name and IPv4 addresses
- Traffic pattern for each circuit and each relay.
- Circuits identification at relays : IPv4 address and the unique circuit ID generated and used by the initiator.

When the program has generated the mapping, it computes the correlation value between the interesting circuits observed at the entry relays with all the circuits observed at the exit. The interesting circuits are the ones originating from a potential victim (Web, Bulk, ssh, Irc) and which have been used to transfer data. In order to know if a circuit have in fact been used, the program examines if the client have marked the circuit as dirty in its log file. Note that only an global (unrealistic) adversary could determine if a circuit is marked as dirty since this information is only detained by the clients. This means that a blind adversary would need another method to find out if a circuit observed at the entry should be analyzed or not. In the same way, the program determines which exit circuits are worth correlating with the entry circuit being analyzed by determining if there is some traffic activity at the exit circuit during the interval of correlation. In addition, the program implements three different correlation techniques inspired by the methods described in section 2.6.2. The goal of implementing these different correlation techniques is to compare their performance and to observe if one technique is more suitable than another regarding the different client models.

The last step of the program consists of analyzing the results and establishing the performance of the attack given the network mapping previously generated. Section 4.2 addresses this subject.

The correlation attacks are carried out using three different techniques. The first method is inspired from the work of Murdoch *et al.* which describes a basic timing analysis attack using latency probes [31]. The sampling method has been reviewed to fit the current threat model resulting in a basic timing analysis attack that is performed using the observed cells traffic. In the following sections, this first method will be called “the basic approach”. The second and the third methods that are implemented and tested are respectively the “packet counting” and “cross-correlation” technique described in section 2.6.2. The following sub-sections explain how these techniques have been utilized to conduct the attacks.

4.1.1 Basic approach

As previously described, the basic approach consists of a timing analysis attack inspired from the work of Murdoch *et al.* [31]. The correlation attack has been reviewed in order to be applicable to the current threat model. With the revisited attack, the adversary does not rely on latency probes to determine the traffic patterns. Instead, she relies on the cell traffic histories generated by the Tor controller to obtain the traffic patterns. With the new method, the traffic patterns observed at the entry and exit relays are respectively defined as $E(t)$ and $E'(t)$:

$$E(t) = \begin{cases} 1 & \text{If one ore more RELAY cells are sent on the entry circuit at time t} \\ 0 & \text{Otherwise} \end{cases} \tag{4.1}$$

$$E'(t) = \begin{cases} 1 & \text{If one ore more RELAY cells are sent on the exit circuit at time t} \\ 0 & \text{Otherwise} \end{cases} \tag{4.2}$$

Both $E(t)$ and $E'(t)$ acts as activity probes and defines the patterns at both the entry and exit nodes in function of the time. The adversary determines if the two circuits are related using formula defined by Murdoch *et al.* 4.3:

$$c = \frac{\sum E(t) \times E'(t)}{\sum E(t)} \tag{4.3}$$

The resulting correlation value c reveals if the two circuits were active at the same moment. The closer to one the value c is, the more the circuits are active at the same time. Meaning that they are more related, at least in a timing point of view. Therefore, this approach is considered a pure timing analysis attack since the effective throughput (i.e. cell/s) is not taken into account. It is also relevant to point out that the Tor controller records the traffic of Tor cells once every second. Therefore, the traffic patterns of cell flows are sliced into time windows of one second. This shouldn't be problem considering that this size of window should give accurate results [42].

4.1.2 Packet counting

The second technique that is implemented consists in calculating the number of RELAYS cells that transits on the entry and exit circuits during a period of time and to compute the distance between the two cell counts using Formula 2.2. The counts of RELAYS cells observed at the entry and exit relays during an interval of time t are respectively defined as x and y. The distance between two cell counts x and y is defined as $d(x, y)$ and can be used to measure the similarity of the flows. However, the resulting distance $d(x, y)$ between two cell counts cannot be used as such and must be normalized with the maximum distance observed among the correlated circuits to obtain a consistent measurement. The correlation value c between the cell flows x and y is given by the following formula :

$$c = 1 - \frac{d(x, y)}{d_{max}} \tag{4.4}$$

The more the distance is close to zero, the more the two cell counts are related. A distance of zero means that cell counts x and y are equal. The distance $d(x, y)$ is normalized with the maximum distance observed in order to obtain a normalized distance varying from 0 to 1. Thus, the correlation value c varies from zero to one, one being the best correlation value (i.e. when cell counts x and y are equal). In contrary to the basic approach, the packet counting strategy only relies on the amount of Tor cells transferred during a period of time to compare the similarity of two flows. And the method does not take into account the exact time at which the cells are sent.

Considering that Tor circuits are designed to be used for maximum 10 minutes by default, the interval t during which the cell are counted is also set to 10 minutes. This long interval should allow to distinguish the related circuits for the other.

4.1.3 Cross-correlation

The last tested technique consists in applying the cross-correlation formula (equation 2.3) as correlation measurement for the attacks. The formula requires the adversary to fix a network delay d and a time window W to be applicable. In the context of the experiments, and as the RTT have been evaluated to approximately 500 ms, the network delay d is set to 0 second. And the window size W is set to one second as suggested in a past article [42]. As for the other approaches, the formula is applied against the cell flows observed at the entry and exit relays. The obtained correlation values vary from -1 to 1, one being the best value (i.e. the two cells flows are highly related). This formula should give the most accurate results as it takes both timing and throughput into account to correlate two flows.

4.2 Evaluation methodology

This section is dedicated to define how the accuracy and the effectiveness of the correlation attacks can measured and evaluated.

As the attacks are carried out in the Shadow environment and that Tor implementation has been altered to withdraw circuits anonymity, it is possible to determine if an attack has given correct results. Indeed, for each correlation value c computed between an entry and an exit circuit, and given a fixed threshold t, the adversary can determine if the correlation is a :

True-positive (TP) The correlation value is higher than the threshold ($c \geq t$) and the two circuits are indeed related. It is a **correct match**.

True-negative (TN) The correlation value is lower than the threshold ($c < t$) and the two circuits are indeed not related. It is a **correct non-match**.

False-positive (FP) The correlation value is higher than the threshold ($c \geq t$) and the two circuits are not related. It is an **incorrect match**.

False-negative (FN) The correlation value is lower than the threshold ($c < t$) and the two circuits are related. It is an **incorrect non-match**.

The threshold value t is set by the adversary and determines the performance of the attack. If the threshold t is set too high, there is a risk of not finding any correct matches. On the other hand, a low threshold emphasizes the risk of finding false-positive matches. Therefore, the threshold should be set to maximize the number of true-positive while keeping the number of false-positive as low as possible. Two measurements, the *false-positive rate* and the *false-negative rate*, can be used to evaluate the effectiveness of the attacks for a given threshold [20]. The false positive rate, also known as the false-alarm rate, is defined as follow :

$$FPR = \frac{FP}{FP + TN} \tag{4.5}$$

The FPR measures the proportion of false-positives found among all the unrelated circuits. A low FPR is desirable considering that the adversary is less susceptible to find false-positives in the results. The second measurements that helps evaluating the performance of the attacks is the false-negative rate :

$$FNR = \frac{FN}{FN + TP} \tag{4.6}$$

The FNR measures the proportion of related circuits that have not been correctly matched (c < t). As for the FPR, a low FNR is desired, since in that case, the attacker is more susceptible to find correct matches among the results.

The performance of an attack can be evaluated using the FPR and the FNR as both measurements reveal the efficiency and the accuracy of the conducted attack for a fixed threshold t. One manner to determine the best accuracy achievable for an attack is to compute the *equal error rate* (EER), which is obtained by adjusting the threshold t to make the FPR and the FNR equal [26]. The more the EER is low, the more the attack is accurate. The EER can be used as a point of reference to compare the accuracy of the conducted attacks against the different client models. Furthermore, this evaluation method can be applied regardless of the correlation method used. Then, the accuracy of the different correlation techniques can also be compared.

The following sections aim to detail and discuss the results of the attacks obtained in the case of two different adversaries. In the first scenario, the adversary is global and knows which circuits are used by which clients. She is also aware of the time when the circuits were first used and she carries out the correlation attacks using these information. Consequently, the results should be optimal and should reflect the efficiency of the different correlation methods, as well as their effectiveness against the different client models. In the second case, the adversary is not almighty, she just has access to the Tor logs of all relays and needs to determine how to set up the attacks to achieve the best results. One of the objective of the first scenario is to gather results that will give some insights on how to maximize the performance of the attack in the case of a blind attacker. The second scenario places the adversary in a more realistic situation and is dedicated to illustrate how an adversary could set-up an attack in the live Tor network.

4.3 Global adversary

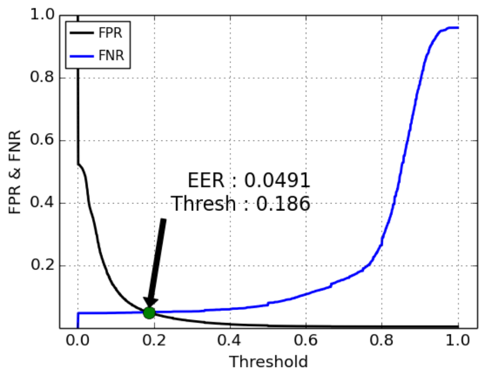
This section details and discusses the results of the end-to-end correlation attacks performed by a global adversary that can monitor the entire Tor network. In this scenario, the opponent has access to all the information she needs to achieve the most accurate attacks. She knows the complete network mapping and knows which entry circuits are worth correlating. Moreover, she can determine when to start correlating the cell flows since she knows when the circuits have been used for the first time with the circuit_dirty timestamp. The time interval of correlation is set to ten minutes and starts when the victims circuits have became dirty. The interval is set to ten minutes as it is the maximum lifetime of a circuit in the simulation (Tor default settings).

The three correlation techniques described in the above sections are tested against 1000 clients of each kind (web, bulk, SSH and Irc). The following sections 4.3.1, 4.3.2 and 4.3.3 will first describe and analyze the results for each technique individually. Then, a comparison is made between the techniques to determine which one is most accurate and efficient. The focus is placed on finding which method will best suit the needs of the blind adversary and to obtain some insight on how to achieve the best results in that scenario.

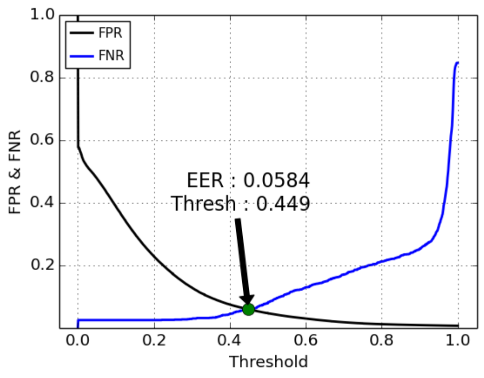
4.3.1 Basic approach

As previously described in section 4.2, the accuracy of a correlation attack can be established using the *equal error rate* (EER) which is obtained by finding the threshold t such that the false positive and negative rates are equal. The lower the equal error rate is, the more accurate the attacks are. Calculating the EER helps to determine if a correlation technique has a chance of achieving good results. Furthermore, it gives an adequate measurement to compare the accuracy of different attacks.

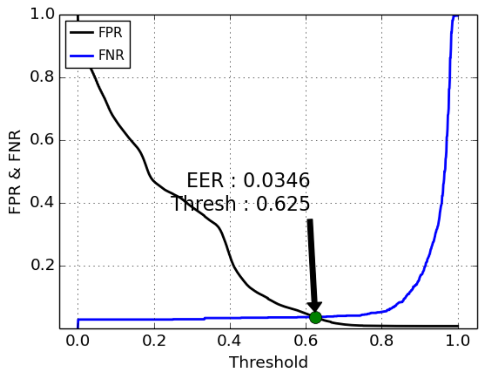
Figure 4.1 illustrates the false positive and negative rates obtained for each kind of client when varying the threshold t between 0 and 1. The EER for each type of traffic is given when the FPR and FNR intersect.



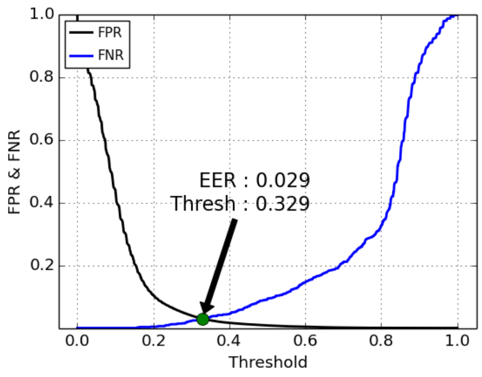
(a) Web clients



(b) Bulk clients



(c) Ssh clients



(d) Irc clients

Figure 4.1: Graphs of the false positive and negative rates obtained with the basic approach for every type of client when varying the threshold between 0 and 1. The equal error rate is shown where the FPR and FNR curves intersect.

The graphs in Figure 4.1 reveal that the basic approach seems accurate against all kind of traffic. Indeed, the EER is quite constant and only varies from 2,9% (Irc) to 5,8% (Bulk). It seems that the basic approach achieves better accuracy when correlating ssh and Irc clients. This can be explained by the fact that the traffic observed on the circuits belonging to ssh and Irc clients are more unique since they carry little traffic with lots of random gaps in the transfers. On the other hand, one might expect that bulk traffic, with the lack of gaps in the traffic patterns, would be more difficult to correlate (EER = 5,8%).

Nevertheless, the obtained EERs are given for different thresholds. This means that a blind adversary, who doesn't know what kind of traffic an entry circuit is carrying, must fix a unique threshold t for its attacks. Fixing the threshold to the most efficient value is discussed later. Still, the problem of fixing a unique threshold can already be observed graphically. Imagine that an adversary fixes the threshold to 0.5, then she should obtain good results for every kinds of clients since the FPRs will be low in each case (i.e. less incorrect matches in the results). However, by fixing this threshold, she would obtain higher FNRs leading to less correct matches in the results. Since having less correct matches in the results is less damaging than having false positives, the adversary would prefer a high threshold to maximize its chance of successfully matching related flows. Still, one can observe that the FNRs for each types of client do not evolve in the same way. Then, fixing the threshold too high, even if it is acceptable for one type of traffic, could decrease the chance of matching another type of client. Let's take the Web and SSH clients of Figure 4.1 as example. Imagine that the adversary fixes the threshold to 0.8 because she thinks that it will maximize the probability of achieving good results for the ssh clients (i.e. low FNR and FPR). With that threshold, she would obtain a relatively high FNR with the web clients, leading to miss more web clients than ssh clients.

4.3.2 Packet counting

This section develops the results obtained with the packet counting method. As for the previous approach, the accuracy of the method is given by the equal error rate obtained when the FPR and FNR are equal.

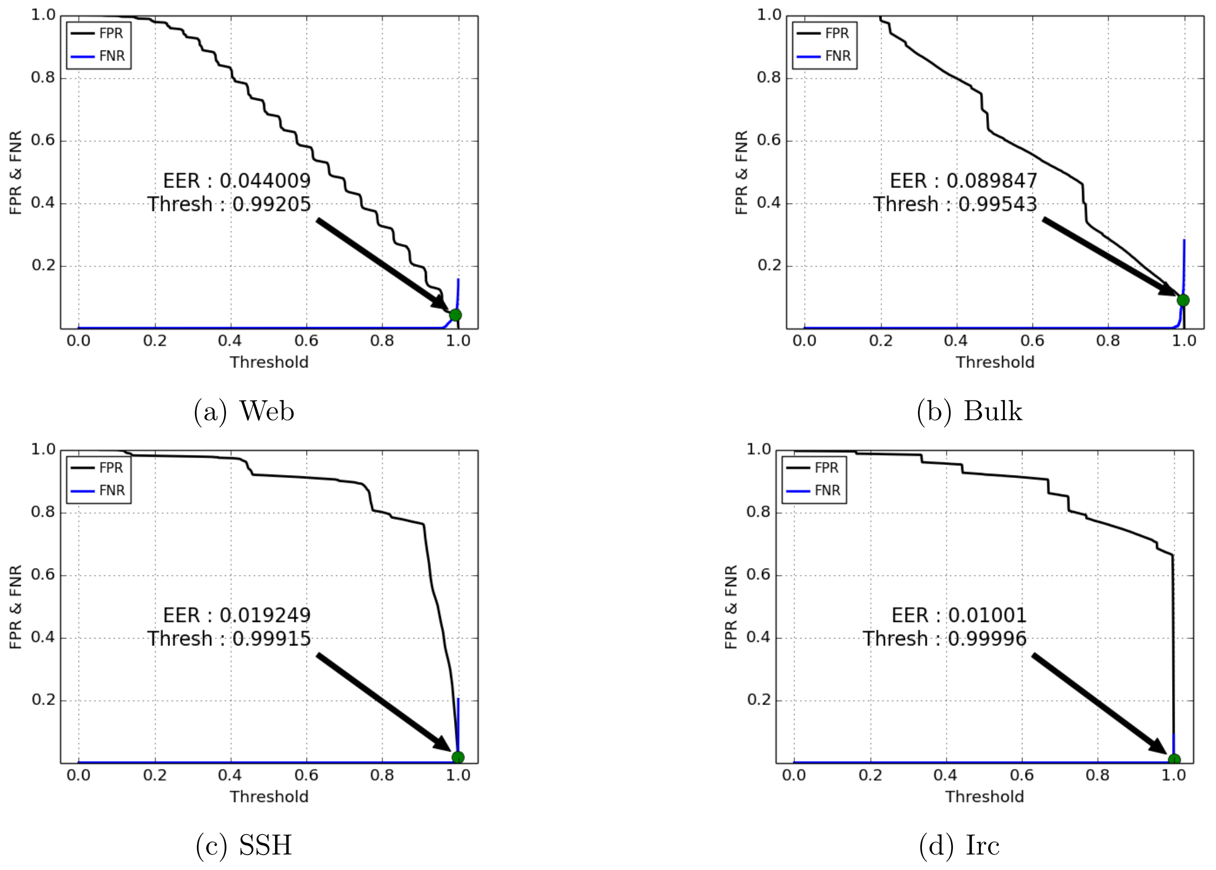
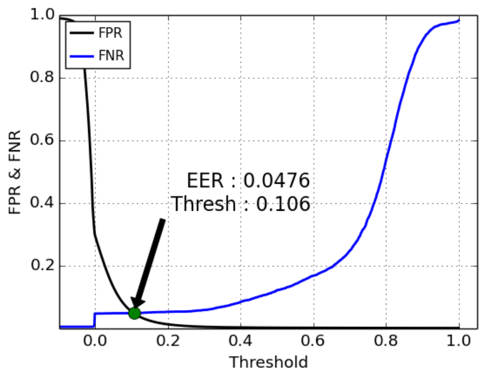


Figure 4.2: Graphs of the false positive and negative rates obtained with packet counting technique for every type of client when varying the threshold between 0 and 1. The equal error rate is shown where the FPR and FNR curves intersect.

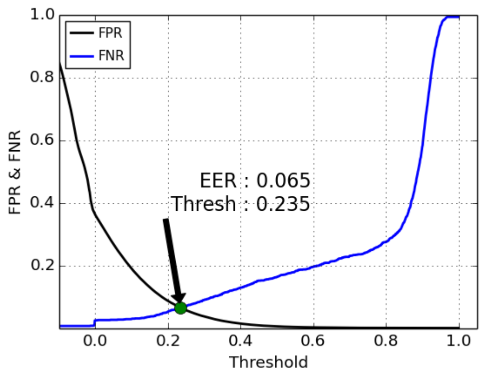
Figure 4.2 reveals that using the packet counting strategy based on a distance measurement is quite challenging. As can be seen on the graphs, the optimal threshold for every type of traffic must be chosen with precision to obtain reliable results. If the attacker does not fix the threshold very precisely, the results will be either full of false positive or false negative as the FPR and FNR tend to grow abruptly around the EER. This is particularly true for the ssh and Irc traffic. Imagine that the adversary fixes the threshold to 0.99 for the ssh clients. In that case, the false negative rate drops to zero, meaning that there will be no false-negative in the results. However, with this threshold, the false positive rate is very high, leading to an inaccurate attack. This is even more true for the Irc traffic. Thus, it is difficult for the adversary to fix a threshold that would be optimal for all types of traffic as one threshold could fit one type of client but would be disastrous for the others.

4.3.3 Cross-correlation

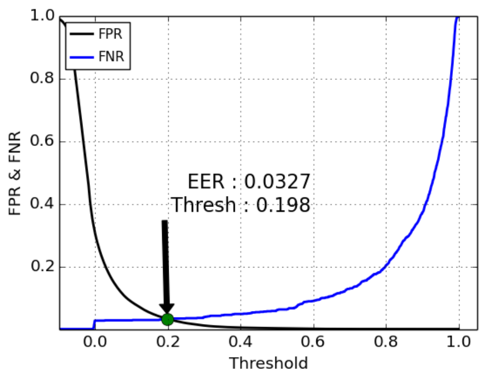
This section details the results obtained with the cross-correlation method. As for the last techniques, the performance of the attacks against the different client models are established using the equal error rate. Figure 4.3 depicts the false positive and negative rates obtained for each kind of client when varying the threshold t between -1 and 1.



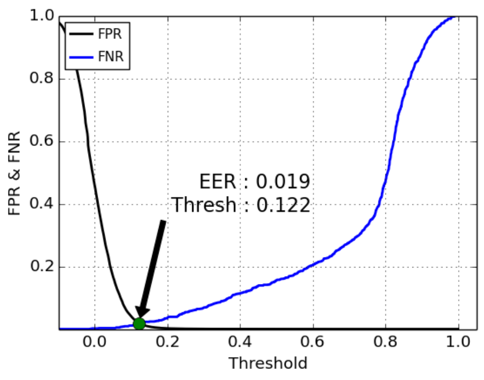
(a) Web



(b) Bulk



(c) SSH



(d) Irc

Figure 4.3: Graphs of the false positive and negative rates obtained with cross-correlation technique for every type of client when varying the threshold between -1 and 1. The equal error rate is shown where the FPR and FNR curves intersect.

Results presented on Figure 4.3 show that the cross-correlation formula gives accurate results for every client model. The EER varies from 1,9% (Irc) to 6,5% (Bulk) and the thresholds obtained with the EER for each type of traffic are close to each other ($0.106 < t < 0.235$). Furthermore, the FNR and FPR curves have quite the same shape suggesting that it could be easy for the adversary to set a threshold t providing satisfying results whatever the type of traffic observed.

4.3.4 Comparison of the correlation techniques

This section aims to compare the performance between the different correlation techniques in the case of a global adversary. As explained previously, in this scenario, the adversary is all-knowing and can differentiate the type of traffic that is flowing on each circuits. Therefore, the adversary can conduct distinguishing attacks and determine the optimal threshold for each type of clients. However, this scenario is only conceivable in an experimental environment and the results obtained with the global adversary are only intended to find the most appropriate technique for the blind attacker. The technique must be both accurate and polyvalent. Meaning that the adversary, by fixing a unique threshold t, should be able to precisely correlate all types of traffic. The remainder of this section is committed to determine which technique and thresholds will best suit the needs of a blind attacker.

Table 4.1 summarizes the results obtained with the three correlation techniques. The table recaps the thresholds and the resulting equal error rates obtained for every techniques and client models.

		Threshold	EER
Basic approach	*Bulk*	0.449	0.0584
	Web	0.186	0.0491
	SSH	0.625	0.0346
	IRC	0.329	0.0290
Packet counting	*Bulk*	0.99543	0.089847
	Web	0.99205	0.044009
	SSH	0.99915	0.019249
	IRC	0.99996	0.01001
Cross correlation	*Bulk*	0.235	0.065
	Web	0.106	0.0476
	SSH	0.198	0.0327
	IRC	0.122	0.019

Table 4.1: Summary of EERs obtained for each type of correlation technique and traffic.

As one can observe, all approaches provide pretty much the same EERs when the same type of traffic is compared. Indeed, the highest difference among the EERs for a fixed type of traffic is $\approx 3\%$ (when the EER of the bulk traffic of the packet counting approach is compared with the results obtained for the bulk traffic of the two other approaches). In all other cases, the EERs differ from less than 3%. Therefore, from the global attacker's point of view, each approach appears to be as accurate as the two others.

However, even if the *equal error rate* allows to compare the accuracy of the different correlation techniques, it doesn't give any clue about the probability of matching two cell flows that are indeed related. Consider an entry flow I and an exit flow J, the probability that the adversary matches flow I with flow J when the two flows are related is defined as $P(I = J|I \sim J)$. The probability $P(I = J)$ is equal to $\frac{1}{n}$, where n is the number of concurrent flows at the exit relays that exist in the time interval of the correlation. The probability $P(I = J|I \sim J)$ can be evaluated using the equation 4.7 defined by Levine *et al.* in their paper concerning timing analysis attacks [26] :

$$P(I = J|I \sim J) = \frac{(1 - FNR) * P(I = J)}{(1 - FNR - FPR) * P(I = J) + FPR} \tag{4.7}$$

Equation 4.7 defines the probability that flow I and J are correctly matched when they have been correlated. As the equation illustrates, the probability depends on the FNR and the FPR as well as the number of concurrent exit flows existing during the interval of correlation. The lower the FNR and the FPR are, the more the attack has chances of being successful. Additionally, the more there are concurrent exit flows existing at the time of the correlation, the less the attack is reliable. Indeed, as n increases, $P(I = J)$ decreases, decreasing in same time the probability of finding a correct result.

As stated above, the probability that the adversary matches the two related flows among n concurrent exit flows is defined by $P(I = J) = \frac{1}{n}$. In the developed attack, a Python script is in charge of counting the average number of concurrent cell flows detected at the exit relays. The number of concurrent cell flows observed at the exit relays slightly varies from one simulation to another and from one correlation interval to another. Thus, the number of concurrent flows, n, is set to the average number of concurrent flows observed during all attacks. The average number n has been evaluated to ≈ 1000 ($855 < n < 1247$). Then, when the adversary correlates an entry flow, she observes on average 1000 concurrent flows at the exit relays. The evaluated number n gives the probability $P(I = J)$ for the experiments.

The probability $P(I = J|I \sim J)$ gives to the adversary a convenient measurement to evaluate the performance of an attack for a fixed threshold. Furthermore, by varying the threshold, the adversary can find the optimal threshold that maximizes the probability of matching two related flows. Table 4.2 illustrates the optimal threshold that gives the best chance of successfully matching two related flows for each technique and client model.

		Optimal Threshold	FNR	FPR	$P(I{=}J \mid I \sim J)$
Basic approach	*Bulk*	0.932	0.2797	0.0072	0.09
	Web	0.715	0.1630	0.0034	0.195
	SSH	0.832	0.0664	0.0062	0.131
	IRC	0.842	0.4914	0.00013	0.79
Packet counting	*Bulk*	0.99998	0.2804	0.0019	0.265
	Web	0.999936	0.147	0.0027	0.24
	SSH	0.999975	0.2030	0.0003	0.693
	IRC	0.999978	0.0890	0.0102	0.082
Cross correlation	*Bulk*	0.879	0.445	0.00004	0.928
	Web	0.745	0.3307	0.0003	0.686
	SSH	0.934	0.499	0.00001	0.975
	IRC	0.761	0.349	0.0	1.0

Table 4.2: Summary of FPR, FNR and $P(I = J|I \sim J)$ for every correlation techniques when fixing the threshold such that it maximize the probability $P(I = J|I \sim J)$ for a given type of client.

As can be seen on Table 4.2, the results for the basic approach suggest that, even if the optimal threshold does not vary that much from one client model to another, the technique is not that efficient. The best probabilities of matching two related circuits that belong to bulk, web and ssh clients varies from 9% to 19%. However, the approach gives satisfying results for the Irc clients. Then, it seems that the technique does well when correlating irregular, intermittent and slightly loaded traffic and is less efficient when it comes to correlate traffic with less interleaving.

By having a look at the results for the packet counting approach, it seems that this approach is more efficient than the basic one to match two related bulk, web and ssh circuits. Indeed, for each of those traffic types, $P(I = J|I \sim J)$ is higher. However, when it comes to match IRC traffic, it is clear that the packet counting approach is less reliable than the previous one. For such a type of traffic, the matching probability severely decreases (79% $\rightarrow$ 8.2%). Concerning the optimal thresholds in the approach, one can observe that they barely moves from one traffic type to another. Consequently, it is easier for an adversary to set a threshold that is close to the optimal ones and suitable for each type of traffic (e.g. 0.9999).

Finally, the cross-correlation technique seems to be the approach providing the best matching probabilities regardless of the client model. With this technique, the matching probability is above 92% for three out of four traffic types. Only the web surfing traffic achieves a lower probability, while remaining much better in this approach than in the two previous ones. However, the cross-correlation technique is far from perfect since the FNR is relatively high for each type of traffic. As a result, the risk of not detecting a correct match is not negligible.

Furthermore, as already stated, the probability of matching two related flows is directly affected by the amount of concurrent exit flows. In the live Tor network, this amount can vary a lot depending on the amount of active Tor users. Logic dictates that the concurrent exit flows are expected to be higher during the rush hour and lower during off-peak. Consequently, the matching probability can drastically change depending on that factor. For example, if the matching probably is computed with the parameters of the cross-correlation web clients from table 4.2 (FNR=0.3307, FPR=0.0003) and a rush hour scenario (n=10000) is considered then the resulting $P(I = J|I \sim J)$ is about 18%. If on the other hand, an off-peack scenario (n = 100) is taken then the matching probability roughly jumps to 96%. Therefore, when the number of active Tor users is large, it is of the first importance for the adversary to choose an accurate correlation technique that minimizes the FNR and the FPR.

Finally, based on the results from table 4.1 and table 4.2, it would be interesting to determine which correlation technique is best suited when an adversary has no prior knowledge about the types of traffic flowing in the Tor network. In such a case, a unique threshold must be set per approach. Consequently, it is likely that some techniques prove to be less efficient or more difficult to put in place than others. This subject is covered in more detail in the next section.

4.4 Blind adversary

This section develops the scenario of a blind adversary carrying correlation attacks as in the live Tor network. In this situation, the adversary is not all-knowing and only have access to the Tor controller logs of all relays. She has no information regarding the type of traffic flowing on each circuit and does not know which circuits are indeed used by the victims (i.e. which circuits are dirty). Therefore, several problems stand before the adversary.

One of the first difficulty for the blind adversary is that she needs to determine which circuits observed at the entry relays are worth correlating. One way to achieve this could be to determine if the cells that are received on a circuit come from a victim and if the circuit indeed carries some RELAY cells. Imagine that the adversary has a list of IP addresses of potential victims. Then, if the adversary receives some cells at an entry relay on a TCP/TLS connection originating from a victim IP address, she can determine which circuits belong to the victims. Moreover, she just needs to count the RELAY cells that transit on a circuit to find out which circuits have been really used. Finally, she can determine when to start the correlation by finding when the first RELAY cell containing data is sent. In the case of a 3-hop circuit, the cells carrying data are the ones that are sent after the circuit construction protocol (Figure 2.3). Thus, when the attacker observes that an EXTENDED and a RELAY cell are first sent back to the OP (i.e. the construction protocol is achieved), she knows that the following cells, if they exist, will carry some data traffic. Meaning that the circuit has been used by the client and should be analyzed. As for the global adversary, the blind one should choose to fix the interval of correlation to 10 minutes as it is the average lifetime of dirty circuit in Tor.

Another complication for the blind adversary is that she must choose which is the best correlation method to employ and must determine how to fix a unique efficient threshold allowing to accurately correlate diversified kind of traffic. The previous section partially answers to the question by revealing that the cross-correlation is the most efficient technique regardless of the type of traffic. Nonetheless, it remains to determine what is the most effective threshold to use. In the context of the experiments, the adversary correlates the circuits of 1000 clients of each kind. Therefore, in this situation, the blind adversary has one chance out of four to stumble upon a given type of client. So, the adversary should fix the unique threshold to the average optimal value. However, in the live Tor network, the client distribution is not the same. There are approximately 93%, 3.5% and less than 1% chance to respectively encounter a web, a bulk or a ssh/irc client [30]. Therefore, the adversary that operates on the Tor network would prefer to fix a threshold that maximizes the chance of correctly correlate web clients. Table 4.3 illustrates that situation by showing the performances of the three correlation methods when the adversary fixes a unique threshold that maximizes the probability of matching web clients. The performances of each method are compared by evaluating the probabilities of correctly matching the flows for each type of clients. The obtained performances are to be compared to the best performances achievable which have been determined by the global adversary (see table 4.2).

As can be observed on Table 4.3, the results obtained for the basic approach are quite unsatisfying. Fixing the threshold in order to get the best probability of catching web clients greatly reduces the chance of catching the other types. Indeed, the probabilities of matching the other types of client are almost divided by two considering the best probabilities achievable. Furthermore, the overall performance of the method are not astonishing as the probability of matching two related circuits only varies from 4 to 42%.

		Fixed Threshold	FNR	FPR	P(I=J | I $\sim J$)	Best probability achievable
Basic approach	*Bulk*	0.715	0.1782	0.0160	0.0489	0.09
	Web	0.715	0.1630	0.0034	0.195	0.195
	SSH	0.715	0.0395	0.0097	0.0902	0.131
	IRC	0.715	0.2202	0.0011	0.4218	0.79
Packet counting	*Bulk*	0.999936	0.0	0.4734	0.0021	0.265
	Web	0.999936	0.147	0.0027	0.24	0.24
	SSH	0.999936	0.0	0.8970	0.0011	0.693
	IRC	0.999936	0.0	0.8520	0.0012	0.082
Cross correlation	*Bulk*	0.745	0.2330	0.0007	0.5172	0.928
	Web	0.745	0.3307	0.0003	0.686	0.686
	SSH	0.745	0.1357	0.0002	0.8144	0.975
	IRC	0.745	0.2883	0.0000	0.9980	1.0

Table 4.3: Summary of FPR, FNR and $P(I = J|I \sim J)$ for each type of traffics and correlation techniques when fixing the threshold that maximizes the chance of correlating Web clients.

The case of the packet counting strategy, although similar to the basic approach, is more extreme. Notwithstanding the web clients, the probabilities for the other types of clients are roughly divided by 100 considering the best achievable probabilities. Even if this method achieves better results than the basic approach for the web clients, it gives poor results in overall as the probability of matching related circuits only varies from 0,11 to 24%. The best performance are achieved with the cross-correlation method. With this last method, the probability of catching bulk client is almost divided by two. However, the probabilities do not decrease that much for the ssh and irc clients and remain quite satisfactory (respectively 81 and 99.8%). The particularly good results for this last method can be explained by the relatively low FPR for each client model. Indeed, if there are almost no false-positives in the results, they should only contain matching results (i.e. $P(I = J|I \sim J)$ is close to one). Nevertheless, one should notice that the FNRs for this method are quite high and vary from 13 (ssh) to 33% (web). This means that, even if the results are almost filled with correct matches, the attacker has also a great chance of missing a correct results. Let's make an example with the Web clients of the cross-correlation method. If the adversary encounters an entry circuit carrying web traffic, she has 33% chance that the correlation with the related exit circuit will be less than the threshold ($c < t$). Then, she will put 1 correct match out of 3 in the negative results by mistake. Thereby, even if she gets very satisfying probabilities that the results will be correct, she will miss quite a lot of correct matches. Again, as explained earlier, the adversary prefers missing correct matches than having results filled with false positives.

Now that the adversary has determined what threshold she should use, she ultimately wants to determine what is the probability of matching related circuits when she has no information about the traffic that is flowing (blind adversary). If she operates on the live

Tor network, she can determine that she has approximately 93%, 3.5% and less than 1% chance to respectively encounter a web, a bulk or a ssh/irc client. Then, she fixes the threshold such that it maximizes her chance of catching web clients and computes the average probability in function of the client ratios to determine her probability of matching related circuits. The resulting probabilities have been calculated for each correlation technique and are shown on Table 4.4. The table reveals the chance of correctly matching one entry and one exit circuit that are related, when the entry circuit is picked at random and that the threshold is fixed to maximize the chance of catching web clients. While considering that the client distribution respects the following ratios : 93%, 5%, 1% and 1% of chance to respectively encounter a web, a bulk, a ssh or a irc client.

	Fixed Threshold	P(I=J | I $\sim J$)
Basic approach	0.715	0.189
Packet counting	0.999936	0.223
Cross-correlation	0.745	0.682

Table 4.4: Probability of correctly matching related circuits when the client distribution respects the following ratios : 93%, 5%, 1% and 1% of chance to respectively encounter a web, a bulk, a ssh or a irc client

The results shown on Table 4.4 are not surprising considering the previous results but give some insight to the blind adversary. Indeed, even if the packet counting approach is challenging to set up because the adversary needs to find a very precise threshold, if the threshold is accurately fixed, this approach can be more effective than the basic approach. However, the first two methods achieve clearly worst performance than the cross-correlation as this last technique obtains a chance of 68% that two correctly matched circuits ($c > t$) are indeed related.

4.5 Discussion and Further work

The results show that end-to-end correlation attacks are a real threat against the Tor anonymity network. However, the results must be put into perspective. As explained, the probabilities of correctly matching two circuits that are indeed related depends on the number of concurrent circuits observed at the exit relays. For the simulation, the number of concurrent circuits have been evaluated to ≈ 1000. However, a realistic adversary that operates on the Tor network may encounter far more concurrent circuits. This situation would hugely reduce her chance of correctly matching related circuits. Moreover, if the adversary has only compromised few relays, her odds drop even more. Yet, as identified by Jhonson *et al.*, the adversary can significantly raise her chance by performing a long-term attack campaign. By doing so, they showed that an adversary with only few resources could deanonymize any given user within six months with over 80% probability [25]. In response to that paper, the Tor developers reduced the default number of guards and raised the time before a guard expires. Even with these changes, the anonymity network is still at risk and no real and effective defense strategy has been adopted yet. Most of the defenses proposed in the literature impact negatively the network performance and the user experience. However, promising researches showed that multiple-path circuits could be used to improve both the performance and the anonymity of the network at the cost of increasing the cryptographic load at the relays [19]. Splitting user's traffic among several circuits at the same time has some advantages. The bandwidth load is more evenly distributed across the network resulting in a better user experience. Furthermore, increasing the number of circuits that are used across the network at the same time decreases the chance of correctly correlating two related circuits. At the time being, it is the only known defense strategy that could improve both the performance and anonymity of the Tor network at the same time. An implementation of such multi-path routing algorithm has been presented in a recent paper [47]. The authors demonstrated the reliability of the new routing algorithm, mTor, with the Shadow simulator. They revealed that the new algorithm leads to better network performance and makes traffic analysis attacks harder to perform. Then, an interesting subject of further researches could be to measure the difference in performance and anonymity between the single and the multi-path algorithm. Another interesting subject could be to measure the performance of the correlation attacks against different client models when using multi-path routing.

Another area of research, which could benefit the adversary this time, would be to evaluate the performance of the attacks when correlating the cell traffic going both in the backward and forward directions. By doing so, the adversary could obtain more unique traffic patterns that could be easier to distinguish from the others. This would result in lowering the false-positive and negative rates, making the attacks more precise and efficient. Furthermore, it could be interesting to see what happens to chance of correctly matching related circuits when the adversary has only compromised a few relays and has only a partial view of the network. As for now, the Tor bandwidth is known to be quite unbalanced between the different relays. As relays advertising high bandwidth are more susceptible to be chosen by the client to create circuits, it could be interesting to identify how much relays and/or bandwidth an adversary needs, to obtain a relatively good chance of matching related circuits. The Shadow simulation tools could be used to assess these subjects.

Chapter 5

Conclusion

Over the course of this document, we developed and analyzed the threat of a relay adversary carrying end-to-end correlation attacks against the Tor network. For this purpose, we first laid the foundation of the Tor anonymity scheme and defined in details how to conveniently set-up correlation attacks using the Tor log files. Then, the document describes how we utilized and adapted the Shadow simulation tools to test and assess the severity of the threat. We tested three different correlation methods. The first method, called the basic approach, which consists of a timing analysis attack inspired from the work of Murdoch *et al.* has given the worse results considering a blind adversary operating on the live Tor network. The packet-counting method almost gave the same poor results as the basic approach and is more problematic to calibrate. The cross-correlation was identified as the method giving the most satisfying results. With this last method, a blind adversary has a great chance to successfully deanonymize circuits carrying any type of traffic. The cross-correlation method has been identified as the most multipurpose among the three methods implemented and should be used instead of the two others. The research that was conducted illustrates that an adversary, given enough time and some access to Tor relays log files, could deanonymize users circuits regardless of their online activity. Users can either do some web-surfing, instant-messaging or even download large files over Tor, they are almost taking the same risk. At least, this is true if the adversary uses the cross-correlation method.

The above situation demonstrates another time the urge for an efficient and reliable defense mechanism to counteract end-to-end correlation attacks. After a decade of research, the problem is still unsolved but there exist some promising study showing that using multi-path circuits with Tor could increase both the performance and the anonymity of the Tor network. Therefore, it could be interesting to evaluate the benefit in term of performance and anonymity that multi-path circuits can provide considering different kind of client models. Researches going towards this objective would encourage the Tor community to eventually improve and adopt a more efficient system providing better performance and anonymity to Tor users.

Bibliography

[1] Pearson product-moment correlation. https://statistics.laerd.com/statistical-guides/pearson-correlation-coefficient-statistical-guide.php.

[2] What attacks remain against onion routing. https://www.torproject.org/docs/faq.html.en#AttacksOnOnionRouting, 2016.

[3] M. Akhoondi, C. Yu, and H.V. Madhyastha. Lastor: A low-latency as-aware tor client. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 476–490, May 2012.

[4] Michael Backes, Aniket Kate, Sebastian Meiser, and Esfandiar Mohammadi. (nothing else) mator(s): Monitoring the anonymity of tor's path selection. *IACR Cryptology ePrint Archive*, 2014:621, 2014.

[5] Kevin Bauer, Micah Sherr, Damon McCoy, and Dirk Grunwald. Experimentor: A testbed for safe and realistic tor experimentation. In *Proceedings of the 4th Conference on Cyber Security Experimentation and Test*, CSET'11, pages 7–7, Berkeley, CA, USA, 2011. USENIX Association.

[6] Daniel J. Bernstein. Curve25519: New diffie-hellman speed records. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 207–228. Springer, 2006.

[7] 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, pages 80–94, Washington, DC, USA, 2013. IEEE Computer Society.

[8] Sambuddho Chakravarty. Traffic analysis attacks and defenses in low latency anonymous communication. Columbia University, 2014. Graduate School of Arts and Sciences.

[9] Henri Crombé and Mallory Declercq. Github of the modified version of the tor shadow plugin used for the simulations. https://github.com/HenriCrombe/shadow-plugin-tor-enhanced, 2016.

[10] Henri Crombé and Mallory Declercq. Github of the pcap-replay plugin. (pulled to shadow-plugin-extras). https://github.com/shadow/shadow-plugin-extras/tree/master/pcap_replay, 2016.

[11] Roger Dingledine. Better guard rotation parameters. In *Tor Tech Report 2011-08-001*, 2011.

[12] Roger Dingledine. Improving tor's anonymity by changing guard parameters., 2013.

[13] Roger Dingledine and Nick Mathewson. Tor path specification. `https://gitweb.torproject.org/torspec.git/tree/path-spec.txt`.

[14] Roger Dingledine and Nick Mathewson. Tor protocol specification. `https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt`.

[15] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. In *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, SSYM'04, pages 21–21, Berkeley, CA, USA, 2004. USENIX Association.

[16] Matthew Edman and Paul Syverson. As-awareness in tor path selection. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, pages 380–389, New York, NY, USA, 2009. ACM.

[17] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, pages 43–54, New York, NY, USA, 2012. ACM.

[18] Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, WPES '12, pages 43–54, New York, NY, USA, 2012. ACM.

[19] O. Pereira F. Rochet and O. Bonaventure. Moving tor circuits towards multiple-path: Anonymity and performance considerations. 2015.

[20] Tom Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[21] Xinwen Fu and Zhen Ling. One cell is enough to break tor's anonymity. *Proceedings of Black Hat Technical Security Conference*, 2009.

[22] Rob Jansen and Nicholas Hopper. Shadow: Running tor in a box for accurate and efficient experimentation. In *NDSS*. The Internet Society, 2012.

[23] Rob G. Jansen. The shadow simulator. `https://github.com/shadow/shadow`, 2015.

[24] Rob Jansen John Geddes and Nicholas Hopper. Tor IMUX: Managing connections from two to infinity, and beyond. In *Proceedings of the 12th Workshop on Privacy in the Electronic Society (WPES)*, November 2014.

[25] Aaron Johnson, Chris Wacek, Rob Jansen, Micah Sherr, and Paul Syverson. Users get routed: Traffic correlation on tor by realistic adversaries. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer &#38; Communications Security*, CCS '13, pages 337–348, New York, NY, USA, 2013. ACM.

[26] Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright. Timing attacks in low-latency mix-based systems. In Ari Juels, editor, *Proceedings of Financial Cryptography (FC '04)*, pages 251–265. Springer-Verlag, LNCS 3110, February 2004.

[27] Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia. A new cell-counting-based attack against tor. *IEEE/ACM Trans. Netw.*, 20(4):1245–1261, August 2012.

[28] Chao Liu, Ryen W. White, and Susan Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 379–386, New York, NY, USA, 2010. ACM.

[29] Nick Mathewson. Improved circuit-creation key exchange. `https://gitweb.torproject.org/torspec.git/tree/proposals/216-ntor-handshake.txt`, 2011.

[30] Damon Mccoy, Kevin Bauer, Dirk Grunwald, Tadayoshi Kohno, and Douglas Sicker. Shining light in dark places: Understanding the tor network. In *Proceedings of the 8th International Symposium on Privacy Enhancing Technologies*, PETS '08, pages 63–76, Berlin, Heidelberg, 2008. Springer-Verlag.

[31] Steven J. Murdoch and George Danezis. Low-cost traffic analysis of tor. In *Proceedings of the 2005 IEEE Symposium on Security and Privacy*, SP '05, pages 183–195, Washington, DC, USA, 2005. IEEE Computer Society.

[32] Steven J. Murdoch and Robert N. M. Watson. Metrics for security and performance in low-latency anonymity networks. In Nikita Borisov and Ian Goldberg, editors, *Proceedings of the Eighth International Symposium on Privacy Enhancing Technologies (PETS 2008)*, pages 115–132. Springer, July 2008.

[33] Steven J. Murdoch and Piotr Zieliński. *Privacy Enhancing Technologies: 7th International Symposium, PET 2007 Ottawa, Canada, June 20-22, 2007 Revised Selected Papers*, chapter Sampled Traffic Analysis by Internet-Exchange-Level Adversaries, pages 167–183. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[34] Konstantin Müller. Defending end-to-end confirmation attacks against the tor network. Gjøvik University College, 2015. Department of Computer Science and MT-DMT Technology.

[35] Rishab Nithyanand, Oleksii Starov, Adva Zair, Phillipa Gill, and Michael Schapira. Measuring and mitigating as-level adversaries against tor. *CoRR*, abs/1505.05173, 2015.

[36] Gavin O'Gorman and Stephen Blott. Improving stream correlation attacks on anonymous networks. In *Proceedings of the 2009 ACM Symposium on Applied Computing*, SAC '09, pages 2024–2028, New York, NY, USA, 2009. ACM.

[37] Lasse Øverlier and Paul Syverson. Locating hidden servers. In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*. IEEE CS, May 2006.

[38] Tor project. Data-collecting service in the tor network portal. `https://collector.torproject.org/`, 2016.

[39] Tor project. Tor metrics portal. `https://metrics.torproject.org`, 2016.

[40] Nick Mathewson Roger Dingledine, George Kadianakis and Nicholas Hopper. One fast guard for life (or 9 months). In *7th Workshop on Hot Topics in Privacy Enhancing Technologies*, HotPETs, 2014.

[41] Fatemeh Shirazi, Matthias Goehring, and Claudia Diaz. Tor experimentation tools. In *Proceedings of the 2015 IEEE Security and Privacy Workshops*, SPW '15, pages 206–213, Washington, DC, USA, 2015. IEEE Computer Society.

[42] Vitaly Shmatikov and Ming-Hsiu Wang. Timing analysis in low-latency mix networks: Attacks and defenses. In Dieter Gollmann, Jan Meier, and Andrei Sabelfeld, editors, *Computer Security – ESORICS 2006: 11th European Symposium on Research in Computer Security, Hamburg, Germany, September 18-20, 2006. Proceedings*, pages 18–33, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[43] Sukhbir Singh. Large-scale emulation of anonymous communication networks. University of Waterloo, 2014.

[44] Christopher Wacek, Henry Tan, Kevin Bauer, and Micah Sherr. An Empirical Evaluation of Relay Selection in Tor. In *Proceedings of the Network and Distributed System Security Symposium - NDSS'13*. Internet Society, February 2013.

[45] Zeadally Sherali Winkler Stephanie. An analysis of tools for online anonymity. University of Kentucky, 2015.

[46] Matthew K. Wright, Micah Adler, Brian Neil Levine, and Clay Shields. The predecessor attack: An analysis of a threat to anonymous communications systems. *ACM Trans. Inf. Syst. Secur*, 7:2004, 2004.

[47] Lei Yang and Fengjun Li. mtor: A multipath tor routing beyond bandwidth throttling. In *Communications and Network Security (CNS), 2015 IEEE Conference on*, pages 479–487, Sept 2015.