The Pennsylvania State University

The Graduate School

EVALUATING REALISTIC DEPLOYMENTS OF TRUSTED EXECUTION ENVIRONMENTS IN THE TOR NETWORK

A Thesis in

Computer Science and Engineering

by

Rachel King

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science

December 2022

The thesis of Rachel King was reviewed and approved by the following:

Patrick D. McDaniel
Distinguished Professor, School of Electrical Engineering and Computer Science
William L. Weiss Chair in Information and Communications Technology
Thesis Advisor

Sencun Zhu
Associate Professor, School of Electrical Engineering and Computer Science

Chita R. Das
Distinguished Professor, School of Electrical Engineering and Computer Science
Head of the Department of Computer Science and Engineering

Abstract

Tor, the anonymity network, provides privacy to users browsing and communicating over the internet. However, Tor has been demonstrated to be vulnerable to various deanonymizing attacks on its users. In this thesis, we demonstrate how trusted execution environments (TEEs) can be leveraged realistically in the Tor network to mitigate several classes of attacks. As TEEs provide confidentiality and integrity through isolation and attestation, attacks which modify the Tor source code and/or exploit sensitive circuit information violate these security guarantees. We approach this by introducing a framework composed of two parts: (1) we first decompose the attacks to establish a mapping between attacks and the required TEE placements in a circuit to mitigate them, and (2) we model Tor as a graph and introduce an adapted relay selection algorithm to assess the security-performance tradeoff under various deployment scenarios (i.e., TEE availability and placement in a Tor circuit). We find that only *one* attack analyzed requires *every* relay in a circuit to be within a TEE to ensure protection. If, based on `Random` deployment of TEEs, 53% of relays in the network use TEEs, users only see a 32% decrease in performance compared to a non-TEE network, while mitigating all 5 attacks. Our findings show that TEEs provide an effective means to protect users' privacy, with low overhead for even the strictest security requirements (mitigation for all attacks).

Table of Contents

List of Figures

List of Tables

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No's. CNS-1805310 and CNS-1900873. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Chapter 1
Introduction

Anonymity networks have existed for decades to provide users of the internet an enhanced level of privacy when browsing. Tor [1], the onion routing network, is a popular anonymity network used today. The main contributions of Tor include providing privacy while browsing the internet, hosting services anonymously, and bypassing censorship rules [2]. Entirely non-profit, Tor relies on people around the world volunteering their computing resources to host relays, which route users traffic through the network. Tor is an internet browsing solution for users such as journalists, political activists, military professionals, and citizens of countries with geolocation content restrictions to communicate and retrieve information while having their identity and privacy protected. Currently, Tor has over two million users and over 6000 public relays in use [3].

Vulnerabilities, however, exist in the network in attempts to deanonymize users. Many attacks are leveraged through either modifying the source code of Tor, or through exploiting information like circuit IDs, which are used by relays to identify circuits. Examples of deanonymizing attacks include tagging [4–6], fingerprinting [7], and circuit-linking [8]. Another attack that is not explicitly deanonymizing is bandwidth inflation [9, 10]. On its own, it does not deanonymize users, but can increase the likelihood of an adversarial relay being used by a user, then allowing deanonymizing attacks to be leveraged.

Kim *et al.* presents SGX-Tor [11], which utilizes trusted execution environments (TEEs) to prevent attacks that modify code and exploit sensitive data, such as those mentioned above. Trusted execution environments are isolated areas in memory that provide security guarantees of confidentiality and integrity through isolation and attestation. By leveraging TEEs in the Tor network, Kim *et al.* finds that these attacks can be mitigated. However, their analysis is exclusive to a full deployment of TEEs in the network (every relay and user is running Tor within a TEE). While their work is

successful in increasing the security of the Tor network, we find this full deployment assumption to be unrealistic given that the Tor network is entirely volunteer based. To this end, we argue that security guarantees of realistically using TEEs in Tor remain unknown.

In this thesis, we demonstrate how TEEs can be leveraged realistically in the Tor network to mitigate several classes of attacks. We approach this by introducing a framework composed of two parts: (1) we first decompose the attacks to establish a mapping between attacks and the required TEE placements in a circuit to mitigate them, (2) we model Tor as a graph and introduce an adapted relay selection algorithm to assess the security-performance tradeoff under various *deployment scenarios* (the location of TEEs in the network). The attack mapping enables Tor users and operators to understand what role the TEE placement within a circuit (e.g., at an entry or exit relay) has in mitigating different attacks. The model then enables understanding the security-performance tradeoff relative to the *availability* of TEEs in the network (defined as the percentage of nodes that have hardware support for TEEs, ranging from 1-99%) and the the user's *security policy* (defined as the required TEE placement within their circuit).

In evaluating our framework, we aim to understand the practical effects on security and performance when using TEEs. We therefore assess the framework from two perspectives: (1) the baseline attack mitigations and performance offered to users through an incremental rollout of TEEs (without adapted relay selection) and (2) the relative performance impact under concrete user security policies. We explore this space of security and performance under various realistic deployments. In one of our analyzed deployment scenarios, we find that every attack we explore can be mitigated while also meeting the current performance of the Tor network with only 43% of the total relays in the network using TEEs.

Notably, our analysis of the attacks reveals that TEEs are not required in every position in a circuit to mitigate attacks—in fact, 4 out of 5 attacks are mitigated with at most 2 TEEs present in a circuit. This demonstrates that a blanket deployment of TEEs in Tor [11] is not only impractical in the short-term, but is unnecessary. Moreover, our findings show that such incremental deployment of TEEs can mitigate attacks effectively without significantly impacting performance (even for the strictest security policies). This demonstrates that TEEs provide an effective means for protecting user's privacy in the Tor network, and that even incremental deployment of TEEs can have a sizeable impact on user privacy.

We make the following contributions:

- We provide a framework for modeling and analyzing circuit performance and security under varying deployments and TEE circuits.
- We characterize security requirements for circuits in terms of TEE placement in order to mitigate known attacks.
- We perform a security and performance analysis on the effects of using TEEs in the Tor network realistically.

1.1 Thesis Statement

Using partial deployments of trusted execution environments in the Tor network realistically can mitigate known attacks without a significant reduction in performance.

Chapter 2
Background

2.1 Tor

The Tor network [1] is a volunteer based anonymity network that utilizes onion routing [12]. Implemented as an overlay network, Tor traffic is compiled of TCP streams that are encrypted over multiple layers and routed through several servers, known as onion routers, relays, or nodes.

In onion routing, three relays are identified to establish a circuit. Then, an ephemeral key is negotiated between the client and each relay, exclusively. The client will incrementally encrypt the data over each key, beginning with the last relay's encryption, followed by the middle, then the first relay's encryption. Once the data is sent, each relay removes their respective layer of encryption, as if peeling off layers of onion. In this process, a relay is only aware of the prior and subsequent relays. Onion routing's anonymity guarantee is centered around this policy that at no point can an entity in the network see every aspect.

Tor is a popular implementation of onion routing. There are a few main components - onion proxies, onion routers, directory authorities, and hidden services. Below, we will go into detail on the specific nature of each of these components, in addition to more detailed explanations of how relays are selected for circuits and how the routing through a circuit works.

2.1.1 Onion Proxy

An onion proxy, also known as the Tor client, is the Tor software hosted on a user's machine. This client acts on behalf of the user to establish circuits (i.e., path that user's data takes to get from source to destination), encrypt cells (i.e., Tor data being sent

through the network), and begin the transfer of cells through the circuit. Each client maintains an active consensus document which contains an overview of the state of the network and the active relays within it to allow them to connect to relays.

2.1.2 Onion Routers

Onion routers, known as relays from here on, are the relays that are used to route user's data through the network. There are four main types of relays: entry (or guard), middle, exit, and bridge. Each relay has a descriptor which contains information specific to its identity such as its address, identity key, onion key, bandwidth, exit policy, and more. These are all defined as different 'flags' on the relay.

Each relay also maintains a set of keys that are used for encryption processes. An identity key is used to sign its descriptor and sign TLS certificates. An onion key is used to encrypt and decrypt requests to create a circuit and establish ephemeral keys. An ephemeral key (also known as circuit key) is then used to encrypt and decrypt relay cells when sending data through the circuit. These keys are what the client uses to incrementally encrypt the cells, as well as what each relay uses to decrypt at each hop of the circuit.

There are two types of non-exit relays, entry and middle. Entry relays, also known as guard relays, are the first hop in every circuit. These relays are more privileged than others because they have knowledge of the original source of data. Initially, Tor clients will choose three random entry relays. Then, each time the client wants to establish circuit, the entry relay will be one of the three relays chosen. After 30-60 days, the client will throw away their entry list and create a new one with three new entry relays. Middle relays are the second and subsequent hop in a circuit before the final hop. They simply receive and route traffic from one relay to another.

Exit relays are the final relay in a circuit and the only relay which sees the intended destination. These relays actually connect the client to the intended destination. Bridge relays are privately listed entry relays that allow users to hide the fact they are using Tor at all. These relays were created in order to help prevent censorship. As all relays' IP addresses are publicly listed, entities who wish to prevent users from using Tor will block access to the relays; private entry relays are able to prevent this.

2.1.3 Directory Authorities

A directory authority is a specific type of relay that works to maintain the Tor network. As of now, there are 10 directory authorities around the world. They are a set of trusted third parties in the network, run by people explicitly involved in the Tor project [13]. Responsibilities of a directory authority include signing of the directory consensus document with the other directory authorities, as well as keeping up to date information about all relays that are active in the Tor network.

A directory consensus document maintains information on the current state of the network such as what relays exist and their relay descriptors. This document is then advertised to all clients to use when generating circuits. When it comes to signing the consensus document, all authorities must first vote on the general state of the network. This process involves the directories all presenting their personal view of the network in terms of relays. From there, each authority will vote on if they agree with the presented state, and if a majority of the authorities agree, the consensus will be signed and published. It is only valid for a period of time, in which then the authorities need to re-vote and publish an updated consensus.

2.1.4 Hidden Services

Hidden services are special sites and services that can only be able to be accessed within the Tor network. These services benefit users for two reasons. They allow users to host applications or services without exposing their identity, and they allow other users to visit these services without exposing their identities. The key components of hidden services are defined below.

- Onion address: The domain name of the service for users to connect to. It is also the encoded identity signing key of the service, which is used to decrypt the hidden service descriptor.
- Hidden service descriptors: Files containing the introduction points for a service. They are encrypted to the private signing key for the specific service, then published in the distributed hash table that is stored at various hidden service directories.
- Hidden service directories (HSDirs): Tor relays that host hidden service descriptors, allowing users to contact them to receive information on the different hidden services available. A relay can become a hidden service directory (has the "HSDir" flag in

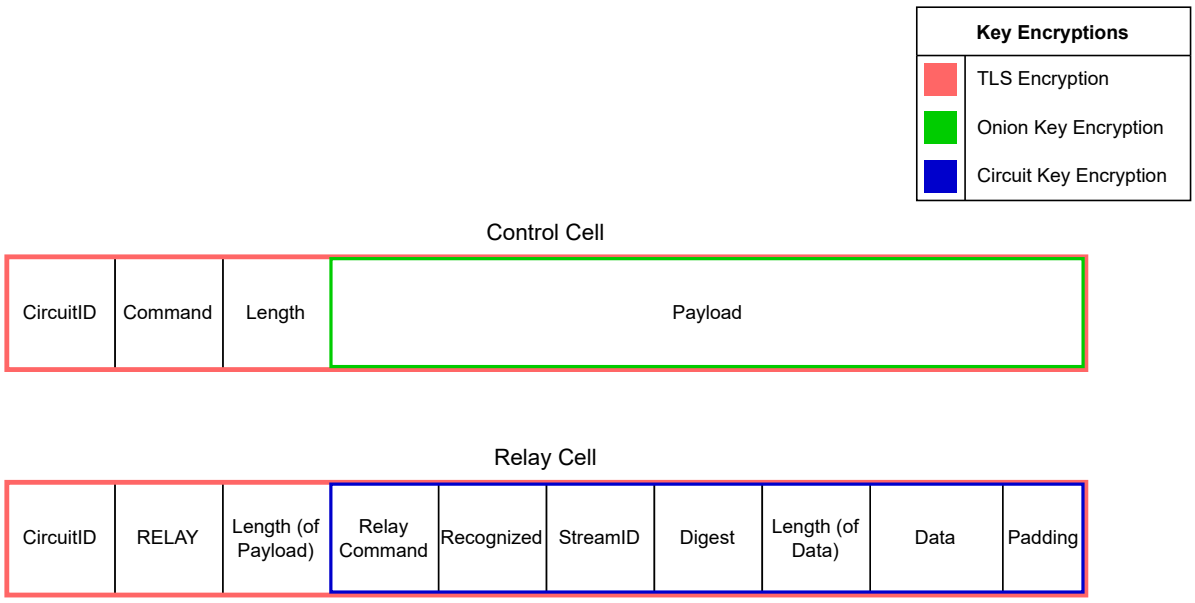


Figure 2.1: Tor cell architecture.

its router descriptor) by being active for a specified number of hours, and also has the "fast" and "stable" flags.

- Introduction points: Tor relays that are designated as the point of connection to request access to a hidden service. This relay passes anonymous requests to the service host containing the address of the rendezvous point.

- Rendezvous points: Tor relays that are designated as the point of connection to a hidden service. Both the service host and the user's client need to have circuits connecting to this point in order to use the hidden service.

2.1.5 Tor Cells

Tor sends and processes messages through a data type called cells. All cells have a header which contains the circuit ID, the command, the length of the payload. After the header is the actual payload of the cell. Note that the header is not incrementally encrypted by the client or relays. There are two types of cells in the Tor design: control cells and relay cells. Control cells are used to establish connections between the client and the relay or between two relays. The main commands of control cells are *create* and its counterpart *created*, as well as *destroy*.

Relay cells are used to send data through the circuit. These cells include additional information in their payload, called the relay header, to include the actual relay command for what to do with the payload (the header command for relay cells is *relay*), a 'recognized' value to determine what relay is to be acting on the cell payload, a streamID to identify what stream this cell is associated with, a digest to be used for integrity checking, and the length of the actionable piece of the cell payload. After the relay header, the actual data is included, along with any padding needed to meet the required cell length.

The architecture of the two cells can be found in Figure 2.1. The contents of each field is labeled, in addition to the difference in encryption between the two cells. Relay cells are always encrypted with the ephemeral key that is negotiated between the client the relay. Control cells, however, can be encrypted with either the onion key of a relay, or the ephemeral key.

2.1.6 Relay Selection

There are a few key factors involved when creating a circuit. When a client begins the process of establishing a circuit, it first needs to select which relays to use. Clients maintain a valid consensus document which they use to download relay descriptors. When a client needs to establish a circuit, it checks the consensus document to find all valid relays. From here, the client chooses its exit relay first, then the rest of the relays from beginning of the circuit to end. The exit relay is chosen based on its exit policy, as this has to correspond with the intended destination of the user. Once the exit is chosen, the entry relay and the remaining relays are chosen, in that order. The entry relay is generally selected from the list of primary relays maintained by the client, unless none of these entries are valid, in which case the client will select an entry relay from a larger subset of all entry relays.

The current approach to relay selection for establishing circuits that Tor uses today is an adjusted weighted bandwidth algorithm [14]. In this algorithm, a relay's weight, or probability of being selected for a circuit, is proportional to their bandwidth. This helps to prevent bandwidth bottlenecks from slowing down user activity by encouraging relays with higher bandwidth to be used more often in circuits than relays with minimal bandwidth.

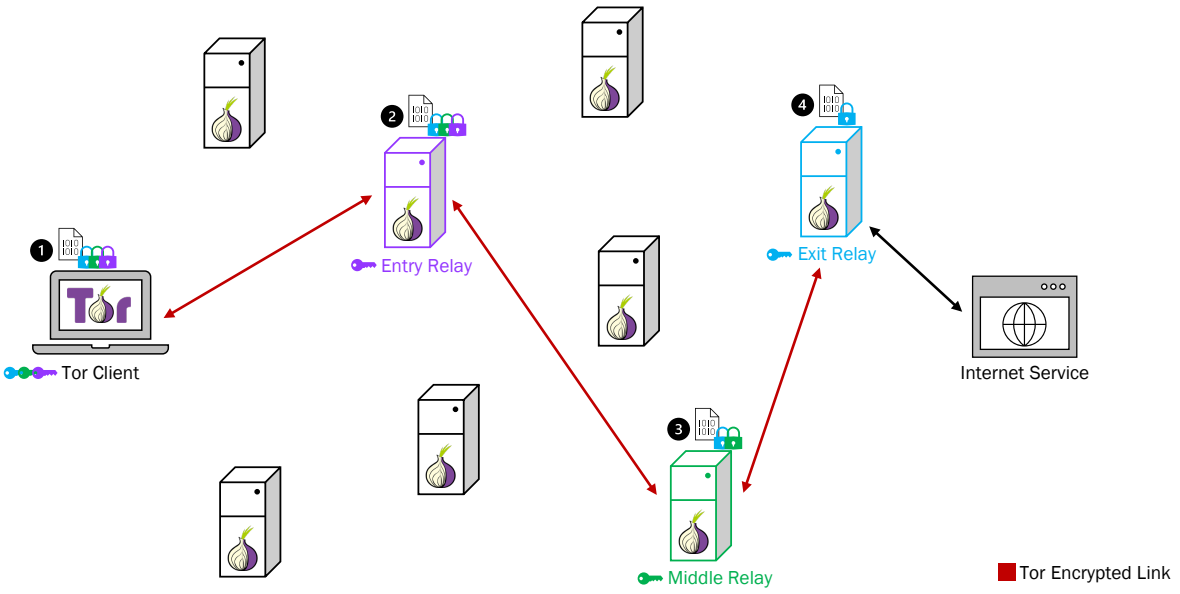


Figure 2.2: The steps for sending Tor data through a circuit.

2.1.7 Tor Circuit Establishment

Once Tor clients have chosen their relays for their circuit, based on the algorithm explained above, they now need to contact each relay to begin setting up their circuit. Once a TLS connection is established between the client and the entry, otherwise known as guard, relay, the Tor client sends a *create* cell with a circuitID unique to this connection, followed by the first half of the handshake used to establish the mutual key encrypted to the entry's public onion key. The entry relay then generates the shared secret and a hash of it with their half of the handshake, which is sent back to the client in a *created* cell. Using AES-CTR, the two then generate their ephemeral keys based on the negotiated shared secret. These ephemeral keys are used to encrypt the payloads of relay cells so that only the intended relay is able to decrypt the payload.

After the connection between the client and entry relay is established, the client now sends a *relay_extend* cell to the entry to extend their circuit by one hop. The entire payload of this cell, containing the address of the new relay, is encrypted to the entry's ephemeral key that was just established, while the data of the payload, containing the first half of the key exchange for the second hop, is encrypted to the second hop's onion key. The entry relay will receive the *relay_extend* cell from the client, establish a TLS connection with the second relay, generate a circuitID to be used for this specific

relay-relay connection, then create a *create* cell with the payload of the *relay_extend* cell to forward on to the new relay. The new relay will generate its half of the key exchange with the client along with the hash to send back to the entry in a *created* cell, which the entry forwards on to the client in a *relay_extended* cell. This exchange allows the client to set up a key specific to each relay while ensuring secrecy with all relays besides the entry.

An overview of the process for sending data through the network can be seen in Figure 2.2. At step 1, we see the Tor client with the data encrypted 3 times by each relay in the circuit's ephemeral key. At step 2, the entry relay receives the data, removing the first layer of encryption using its ephemeral key. The steps at all subsequent relays follow the same procedure.

2.1.8 Modified Tor Relay Selection Algorithms

There have been many published works on the Tor relay selection algorithm and proposed improvements to it. Wang *et al.* propose a latency biased algorithm [15], in which the congestion of Tor relays is calculated and added as a property. Then, when selecting relays for a circuit, a subset of all potential relays is chosen, and the relay with the least congestion is then chosen from there. This is done for each relay in the circuit. Imani *et al.* propose a geographic-aware algorithm [16] which adds an additional weight to relays that takes into account their geographic location. In addition to the weight of the bandwidth, this geographic weight allows for relays with shorter distances from each other to be more likely to be chosen, improving the performance of circuits. Snader *et al.*'s algorithm [17] and Zhang *et al.*'s algorithm [18] take in an additional parameter from the client denoting their desired performance vs anonymity. This parameter is then multiplied by a random variable, then relays are chosen with this additional weight added to them. Both works found that allowing users to specify their performance or anonymity expectations results in little to no performance degradation and meets users preferences. Kiran *et al.* present a selection strategy that is similar to the previous two algorithms in that it takes in a parameter from the client on their expected performance [19]. The relays are separated into two categories based on their bandwidth allowance, and with the client input, potential relays to be chosen for a circuit are based on which category it falls into.

2.1.9 Attacks on Tor

Attacks on the Tor network and its users are constantly being revealed each year. Attacks targeting the network itself include DoS such as the Sniper attack [20] which floods circuits with *sendme* cells, and the CellFlood attack [21], which floods circuits with *create* cells. Sybil attacks [22] admit a large number of malicious relays at once. Attacks targeting Tor users, deanonymizing attacks, can take on many forms. Attacks that are leveraged by the Tor client include [23, 24], which both use the clock skew of the client's computer to deanonymize hidden services. While limited, some attacks are leveraged from only one malicious relay. Attacks requiring only a malicious entry relay include Yang *et al.* and Kwon *et al.*'s fingerprinting attacks [7, 25]. A malicious exit relay attack is Bad Apple Attack [8] which targets BitTorrent users. More commonly, attacks are require two malicious relays, the entry and exit. Examples include tagging attacks such as [4, 6, 26, 27]. Our work focuses on specific classes of attacks that require the modification of Tor source code and the exploitation of circuit details such as circuit IDs and stream IDs. We provide a solution to the mitigation of these various attacks.

2.2 Trusted Execution Environments

2.2.1 Overview

Trusted execution environments (TEEs) are hardware primitives that are used to provide additional security to computer users. They use hardware isolation to provide confidentiality and attestation to provide integrity in remote computing. TEEs provide a secure container in memory that is entirely isolated from the host operating system. Any data and computations placed within this TEE are encrypted to all areas outside of the TEE. Interactions between the TEE and anything outside of it must go through secure function calls each time to ensure data isn't being leaked to the untrusted portion of memory. The architecture of a TEE can be seen in Figure 2.3. As can be seen, the TEE memory is completely separate from the the rest of the memory.

TEEs provide two main security guarantees: confidentiality and integrity. Confidentiality is guaranteed through hardware isolation and sealing. Hardware isolation is from the memory of the TEE being distinctly separate from the OS memory. Any data inside of the memory region of the TEE can only be accessed via the TEE, meaning the OS cannot access this data. Sealing is used to encrypt data within the TEE. Any time data needs to be stored out of the TEE, a private key that is specific to the TEE being used

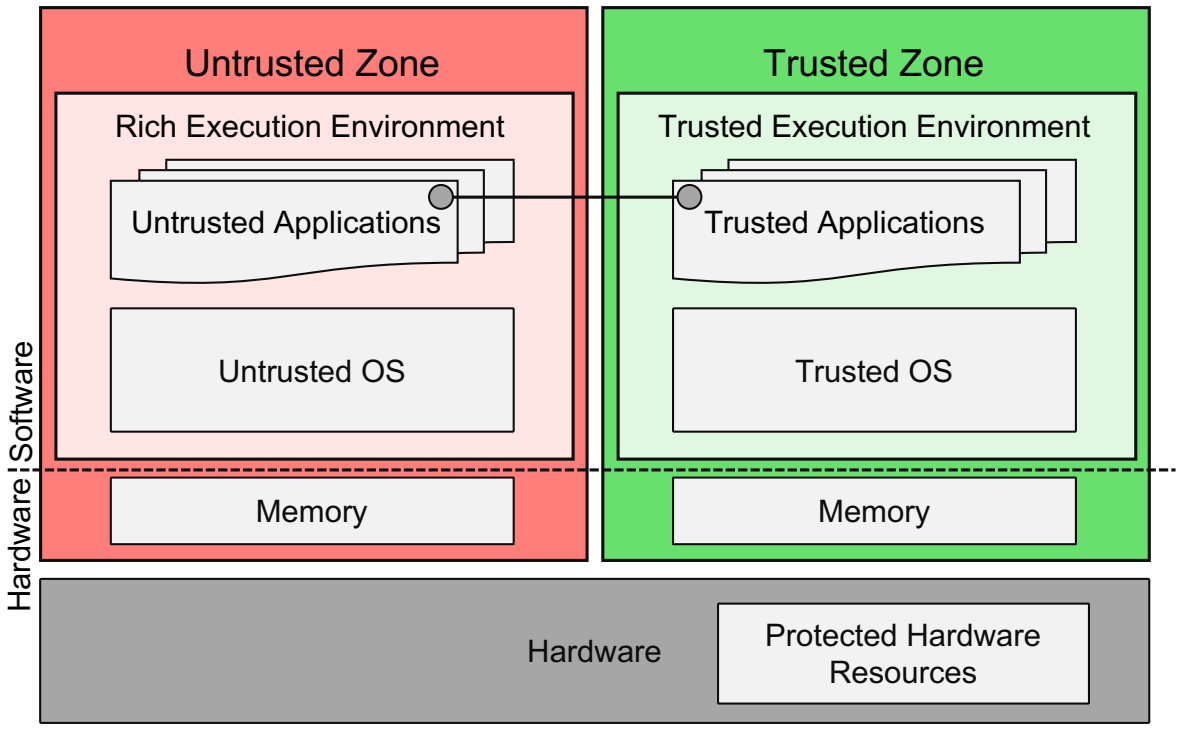


Figure 2.3: The architecture of trusted execution environments.

encrypts the data. This key is only known to the TEE being used, as it is written to to the hardware of the TEE [28]. This prevents the OS from being able to access the plaintext data, ensuring the confidentiality of it.

Integrity is provided through attestation. Attestation is used when communicating with another TEE or when using a TEE from a remote location. When a TEE intends to verify the identity of another TEE on the same system, the two exchange messages containing their identities to then allow for a secure channel to be established for the two to interact. Remote attestation is the process of verifying a remote party is using a TEE for computations [29].

2.2.2 TEE-Based Applications

As the effort of integrating applications with TEEs is generally nontrivial, recent works have been published allowing for easier integration. Once approach to simplifying the porting of applications is by using Library OSes. Essentially, all core functions of an operating system (eg., network functions) are placed within a module which then runs within a TEE, increasing the size of the TCB (trusted computing base)

but allowing for unmodified code to reap the security benefits of TEEs with minimal modification. Haven [30] was one of the first library OSes to leverage TEEs in cloud computing. Panoply [31] presented a smaller TCB compared to Haven, while still allowing for applications to run within micro-containers which abstract the OS functionality. Graphene-SGX [32], motivated by the critiques of library OSes, presented a practical TEE system with performance exceeding both Haven and Panoply. These works all present systems which provide for easy portability of applications into TEEs. Our work presents how the security guarantees of TEEs can be leveraged to improve the security of the Tor network.

2.3 Securing Tor with TEEs

A few notable works have explored the union of TEEs and the Tor network in various ways. Panoply [31], the library OS for integrating applications with TEEs, presents a case study on Tor. Shinde *et al.* integrate all Tor Directory Authority protocol into their system in order to prevent adversaries from manipulating the consensus of the network to allows malicious relays to be accepted. Jain *et al.* present OpenSGX [33], a system that emulates SGX at an instruction level. In their evaluation, they are motivated in preventing attacks that exploit the private keys of directory authorities and exit nodes, in which they integrate the cryptographic functions of the Tor protocol into their system. Specifically, all identity key generation and storage for directory authorities, as well as exit relays, is moved into the TEE. Additionally, the onion key of the exit node and the consensus documents are stored in the TEE. These works look at increasing the security of select Tor components, while our work analyzes how the specific position of TEEs in a Tor circuit can mitigate attacks.

Kim *et al.* present SGX-Tor [11] where they place the majority of the Tor protocol within an SGX enclave. The addition of TEEs ensures the integrity and confidentiality of Tor. Adversaries are no longer capable of modifying the Tor source code, and are not able to gain access to sensitive information such as circuit IDs and stream IDs. We expand on the relevance of this work further below.

2.3.1 Side Channel Attacks

Side channel attacks on Tor are a class of attacks that TEEs do not mitigate. We detail a some examples below.

Cell counting attacks, in general, and more specifically the attack presented by Ling *et al.* [6], rely on recognizing how many Tor cells are being sent through a circuit at specific times. This information can be used a few different ways, Ling *et al.*'s work uses the number of cells as a tagging attack. Through collusion, one end of the circuit sends a signal to another by holding the cells in a queue, before pushing them all out at once. TEEs do not provide mitigation of this attack. Despite the encryption of Tor cells occurring within the TEE, the Tor protocol delivers cells to the OS, which forms the IP packets and forwards them, revealing the information that is critical to performing this attack.

Arp *et al.* present Torben [34], another side channel attack. This attack involves a signal being implanted at a server which can be recognized by a malicious entry relay in a circuit. When a user visits a website, adversarial content such as Javascript code, instigating data to be sent of distinct sizes seen at the entry relay. TEEs cannot mitigate this attack. Even if the entry relay is using a TEE, the OS of the relay processes the IP packets meaning it will also recognize the signal being sent.

2.4 SGX-Tor

Kim et al. present SGX-Tor, a system which integrates TEEs with the Tor network. SGX-Tor's solution is motivated by the vulnerabilities that exist in the Tor network and the deanonymizing attacks that can be leveraged against users. By placing all security-sensitive components of Tor within an SGX enclave, the threat model of SGX-Tor is significantly reduced compared to that of original Tor. SGX-Tor protects against the modification of the Tor source code and the release of security sensitive information like circuit IDs, cell commands, router descriptors, and private keys. These protections are guaranteed through the enclave's attestation procedures and isolation from the underlying host system, with the assumption that the enclave behavior can be trusted.

Attestation in SGX-Tor ensures the integrity of all Tor components. When a Tor relay attempts to be admitted to the network, their code is attested first by the directory authorities, guaranteeing that the Tor code running on the relay has not been modified by an adversary. If the code of the relay has been modified, attestation will fail and they will not be admitted into the network.

Isolation and sealing in SGX-Tor ensure the confidentiality of Tor components. This includes identifying information in the network such as circuit IDs, cell commands, and more, since they are now only processed within the enclave. This means that outside of

the enclave, all of this information is encrypted, preventing the host of the relay from processing revealing information. All private keys and any other information that needs to be stored is sealed prior to being written outside of the enclave. Sealing encrypts the data that is being stored in untrustworthy locations. Therefore, in the event of an adversary attempting to maliciously use these keys which are stored outside the enclave, they are only able to access encrypted versions.

Kim et al. find through their evaluation that under a full deployment of SGX-Tor in the network, many known attacks on Tor are mitigated. They effectively reduce the adversarial capabilities in Tor to that of network level. This means that adversaries can still enact timing attacks and view the encrypted packets being sent over the network. In their followup work [35], a preliminary analysis of which portions of the circuit are responsible for mitigating each attack is also presented. We extend on their work by providing a comprehensive mapping of TEE requirements for the entire circuit to mitigate each attack discussed.

Chapter 3
Threat Model and Assumptions

We are evaluating the security of the Tor network with the addition of TEEs. With this, we assume the adversary is capable of running entry, middle, and exit relays in the network, whether they admit them themselves, or compromise existing relays. While running these relays, the adversary can extract private Tor information as well as modify the Tor source code. As with common TEE threat models, we do not trust the operating system, or any other hardware of the relay outside of the TEE. We assume the system is capable of manipulating and modifying any data accessible to it.

We assume we can trust everything within the TEE memory region. We assume the Tor client is not malicious and can be trusted. We assume the Tor client is configured to use the default of 3 hop circuits. We also assume that all directory authorities are running within a TEE and not malicious, as these relays are operated by trusted parties in the Tor community. Every relay using a TEE must go through the attestation process to establish they can be trusted. This process is enacted by the directory authorities. To this end, when a relay chooses to use a TEE, the directory authorities will attest it first, before allowing it to claim its use of a TEE. By attesting, we confirm that all Tor data of this relay will abide by the confidentiality and integrity guarantees of TEEs.

Attacks targeted on the Tor client being malicious are not considered, including [20, 36, 37]. If the client is malicious, as in the user's host machine has been compromised, the user is already in a compromised state.

Chapter 4
Mapping Attacks

In this section we present how TEEs can be integrated into the Tor network, and what security benefits they will provide to Tor users. We detail how the specific position of TEEs in a Tor circuit can mitigate known attacks.

4.1 Overview

As TEEs provide both confidentiality and integrity, integrating TEEs into the Tor network will mitigate known attacks on the network. TEE isolation provides confidentiality of Tor components within the TEE. This includes identifying information in the network such as circuit IDs and cell commands, since they are now only revealed within the enclave. Outside of the enclave, all of this information is encrypted, preventing the OS of the relay from accessing the revealing information. In the original Tor network, circuit IDs and cell commands are unencrypted besides their TLS encryption. TEE isolation and attestation provide the guarantee of integrity in the Tor network. All components in the network that are contained in a TEE will be attested first. This ensures that Tor source code of the TEE relay is unmodified, verifying the behavior of the relay is to be expected.

We provide a security analysis of how various attacks on the Tor network can be mitigated with the application of TEEs based on their security guarantees, expanding on prior work [35] in this area, as their analysis is preliminary. To do so, we characterize each attack in terms of its threat model and adversarial capabilities. We identify the steps of the adversary to initiate and complete the attack, and then determine how each attack is mitigated in terms of which relays in a Tor circuit are required to have a TEE. We consider this specification of which relays as a concept of configuration. Our full circuit configurations to attack mappings can be found in Table 4.1. Each attack we consider requires either the modification of Tor source code and/or knowledge of Tor

sensitive information related to the cells and circuits of a user.

4.2 Replay Attack

4.2.1 Description

Tagging attacks can take on many forms. The general idea is a malicious circuit edge, say entry relay, sends a signal through the use of Tor cells for the exit relay to notice. If the two relays are colluding, they can now share the information each have related to the user and the destination, effectively deanonymizing the user.

Pries *et al.* present the Replay Attack [4] which exploits the decryption error that occurs through cell duplication. Since Tor uses AES-CTR encryption, duplicating cells results in the counter being incorrect, throwing an error. This attack can take two approaches: both the entry and exit relays to be malicious, or the entry is malicious and an adversary is eavesdropping on the connection between the exit relay and the destination. The initial premise is a malicious entry relay duplicates a relay cell it receives before forwarding the cell down the circuit. Once the duplicated cell reaches the exit relay, and the integrity check fails due to the decryption error, a malicious exit relay working with the entry can confirm the user and the destination.

If the adversary is controlling the entry relay and eavesdropping on the connection between the exit relay and the destination, the attack steps are similar. Once the entry duplicates the cell which results in a decryption error, the circuit will be torn down immediately. The adversary eavesdropping the exit relay connection will notice the TCP stream being cut off unexpectedly, confirming the connection between the entry and the sniffer.

4.2.2 Mitigation

The Replay Attack [4] requires the modification of Tor source code. Specifically, whichever relay is initiating the attack (this attack can be initiated from either end of the circuit) must duplicate the target cell to cause the decryption error. The guarantee of integrity with a TEE prevents this modification of code, as a modified Tor relay will not pass attestation. Mitigation of this attack requires at least the entry relay to be within a TEE, with the exit relay being optional. It is not necessary to require both to be within a TEE. This is due to the guarantee of integrity on at least the entry of the circuit. In the event the entry relay is within a TEE, we can guarantee this relay will not be duplicating any

cells, meaning any adversary at the exit relay will have no decryption error to recognize. If the exit relay is duplicating a cell, the entry relay's behavior is trusted, so despite the decryption error at the entry relay, the entry will not do anything malicious with it.

However, consider the event the attack is leveraged from the entry relay, with a sniffer on the connection between the exit and the destination. TEEs cannot prevent the sniffer from inspecting this connection. For this reason, mitigation requires the entry relay to be within a TEE, to prevent the duplication of the cell from ever taking place at the entry relay.

4.3 Hidden Services Attack

4.3.1 Description

Biryukov *et al.* present a hidden services attack [26] (we recognize as the Hidden Services Attack from here on) is another tagging attack which relies on a malicious entry or middle relay, rendezvous point, and another relay to reveal the location of hidden services. The goal of this attack is for an adversary to confirm they are running the entry relay for a hidden service by recognizing a pattern of cells sent by the rendezvous point. When the adversary requests to be introduced to the hidden service, they provide their malicious rendezvous point to the hidden service. The hidden service then constructs a circuit to the rendezvous point, where the adversary sends 50 *padding* cells down the circuit, followed by a *destroy* cell.

If the adversary is the entry relay in this circuit to the rendezvous point, they will receive 53 cells in total, 2 *extended* cells from circuit establishment, 50 *padding* cells, then 1 *destroy* cell. This scenario confirms the adversary is the entry relay of the hidden service, allowing them to reveal the identity of the hidden service as the hop prior to them. If the adversary is only the middle relay, it will receive 52 cells total (one less *extended* cell), confirming the hop prior to is the entry of the circuit.

4.3.2 Mitigation

The Hidden Services Attack [26] requires the modification of Tor source code to send padding cells down the circuit. TEEs' guarantee of integrity prevents modified code from being run within a TEE. It also requires the entry relay to be malicious in order to recognize the padding cells and count how many cells total have been sent. Determining the number of cells being sent is not protected with a TEE, as the size of IP packets can

expose how many Tor cells are being sent. Only requiring the entry relay to be within a TEE will not mitigate this attack since malicious OS on the TEE entry relay can still determine how many packets are sent, confirming the link between the rendezvous point and the entry.

Therefore, in order for this attack to be mitigated, the act of sending the padding cells must be prevented. This requires the rendezvous point to be within a TEE, which translates to the exit relay in the circuit. This will require the hidden service to confirm that the provided rendezvous point is within a TEE, otherwise it will have to deny the user's request for connection to the hidden service.

4.4 Fingerprinting Attack

4.4.1 Description

Kwon *et al.* present a fingerprinting attack [7], which we denote as the Fingerprinting Attack from here on, which was able to exploit circuit level identifying information to reveal if users are visiting hidden services or not. This attack requires the adversary to be acting as the entry relay, passively.

The adversary makes note of three different properties of the traffic to and from hidden services: incoming and outgoing cells, duration of activity, circuit construction sequences. Based on these properties, introduction point circuits are first sought out, meaning circuits between a client and introduction point for a hidden service and introduction point. Once evidence of these circuits is found, the adversary monitors the users of these circuits further to determine rendezvous point circuits, either between a client or a hidden service. The activity monitored can effectively determine if the adversary is an entry relay for a hidden service or a user visiting a hidden service. In the event of the relay being for a hidden service, the adversary is now able to identify the hidden service.

4.4.2 Mitigation

The Fingerprinting Attack [7] requires the entry relay to be malicious to recognize patterns of cells being sent across circuits. Additionally, the authors claim the attack can be implemented by someone eavesdropping on the connection between the user and the entry relay. This attack does not require any source code modification, but does require circuit level identifying information such as circuit IDs to distinguish between

different circuits. TEEs' guarantee of confidentiality from hardware isolation prevents this information from being revealed to a malicious entity on the relay.

Without the circuit ID, a malicious host or entity eavesdropping the connection will only see IP packets being sent between the user's client and the entry relay. This does not distinguish different circuits though, because of Tor's entry relay strategy. Since all circuits of a user go through the same 3 relays, there will be a significant amount of traffic between a user and one of the entry relays, by design. Therefore, recognizing the circuitID is critical to determining recognizing different circuits.

Mitigation of this attack requires the entry relay to be within a TEE. This ensures the entry cannot recognize any patterns and is not able to distinguish different circuits through the use of the circuit ID. However, TEEs only reduces the adversary to a network level. This attack could still potentially be possible through analysis of the IP packets in an attempt to distinguish between circuits.

4.5 Bad Apple Attack

4.5.1 Description

As explained above, Tor multiplexes multiple TCP streams into one circuit. So, in the event a user is visiting an onion service, which may require multiple streams to fetch all the objects, all these streams will be sent over the same circuit. Subsequently, if an exit relay is able to identify the source for one of the streams, it now knows the source of all the other streams along that circuit. The Bad Apple Attack [8] exploits this Tor design choice.

Targeting users of peer-to-peer (P2P) file sharing applications like BitTorrent, this attack requires the adversary to host malicious exit relays, monitor users of P2P applications, and host a malicious peer for the applications. Blond *et al*'s attack evaluation focuses on the P2P application BitTorrent. This attack exploits the fact that 70% of Tor users accessing BitTorrent only use Tor to request peers, as found by the authors. After this, users will connect directly to the peer outside of the Tor network.

When a user's circuit contains one of the malicious exit relays, and the user is requesting a list of peers to contact that have the requested files, the exit relay can modify the returned list to include their malicious peer. Then, when the user connects to the malicious peer outside of Tor, the user exposes their IP address (by design of P2P applications). As all exit relays are public, the malicious exit relay can first confirm that

the request originated from the Tor network, and then can link the peer request through Tor to the actual user. Furthermore, the source of all other multiplexed streams of this particular circuit are now exposed.

4.5.2 Mitigation

The Bad Apple Attack [8] Mitigation is straightforward. This attack relies on the exit relay in a user's circuit to be malicious with modified Tor source code. Additionally, no collusion is required for this attack, but the knowledge of circuit level information such as circuit and stream IDs is required to be able to distinguish between different circuits and different streams.

TEEs protect against the modification of Tor source code through the integrity guarantee from attestation. TEEs also hide circuit identifying information such as circuit and stream IDs, through hardware isolation, providing confidentiality. This prevents the adversary from recognizing the different circuits and streams through their IDs. For these reasons, requiring only the exit relay in a user's circuit to be within a TEE ensures mitigation of this attack under TEE capabilities. However, the adversary is reduced to a network level, as they can still attempt to recognize different circuits and streams through IP packet inspection.

4.6 Bandwidth Inflation

4.6.1 Description

In the current Tor design, relays are selected for circuits weighted proportionally to their bandwidth. This means that there is an incentive for having higher bandwidth, as that relay is now more likely to be used in circuits. From an adversarial perspective, this is beneficial in their task of controlling entry and exit relays of a circuit [9, 10]. Tor adopts the approach of bandwidth scanners to then validate the reported bandwidth. A portion of the directory authorities are considered bandwidth authorities and will periodically scan the bandwidth of the Tor relays by sending traffic to and from a relay and measuring the size of the data and the time it took. From there, the actual published bandwidth of the relay is the median of at least three of the bandwidth authorities' measurements [38].

This bandwidth scanning is an improvement in preventing misreporting of bandwidth, however, it doesn't defeat the attack entirely. As relays are able to recognize the directory authorities that scan for bandwidth, relays are then able to provide more bandwidth

to these streams by reducing(throttling) the bandwidth they allow for the rest of its streams. This can effectively convince the directory authorities that the relay is capable of higher bandwidth which in turn increases the probability it will be chosen for circuits, as can be shown by Biryukov *et al.* [26].

4.6.2 Mitigation

As this attack is specific to each relay, mitigation requires all relays in a circuit to be within a TEE, due to the integrity guarantee from attestation. However, requiring any two of the three relays in a circuit would be an effective mitigation if considering bandwidth inflation as a means to leverage more attacks. Almost all attacks discussed require collusion of at least two relays, except for Fingerprinting [7] and Bad Apple [8]. Fingerprinting only required a malicious entry relay to be successful, whereas Bad Apple Attack only required a malicious exit. Therefore, requiring both the entry and exit relays in the circuit to be within a TEE would be effective in preventing all attacks discussed when using bandwidth inflation as leverage.

Attack	**Adversary Relays**	**Adversarial Goal**	**TEE Position Requirement**
Replay Attack	Entry and Exit	Deanonymize users	Entry
Hidden Services Attack	Entry and Exit	Deanonymize hidden services	Exit
Fingerprinting Attack	Entry	Deanonymize users and hidden services	Entry
Bad Apple Attack	Exit	Deanonymize users	Exit
Bandwidth Inflation	Entry, Middle, and Exit	Increase relay's usage in circuits	Entry, Middle, and Exit

Table 4.1: Minimum required TEE placement in circuit configurations to mitigate attacks against Tor, assuming a circuit consisting of an entry, middle, and exit relay.

Chapter 5
Modeling Deployments

As we presented in chapter 4, TEEs can provide mitigation of various attacks in the Tor network in realistic deployments. In fact, circuits do not need a TEE in every position to provide mitigation. In this section, we present how the TEE circuit mappings can be realistically used in various deployments through simulation of the Tor network using Tor relay data from the directory consensus document [39]. More details can be found in chapter 6.

5.1 Overview

Prior work in this area has analyzed the integration of TEEs in the Tor network [11]. However, this work provides an incomplete analysis given the assumptions that are made on the deployment. More specifically, Kim *et al.* assume in their work a full deployment of TEEs. This reflects as every relay in the Tor network using a TEE. This assumption, however, a greenfield deployment, in which the existing Tor network will be entirely exchanged for the proposed SGX-Tor network. This scenario is unrealistic, as we can't expect all of Tor to transition immediately to a brand new configuration. Tor is a volunteer-based network, and TEEs require specific hardware that not everyone will have to run the Tor software. Instead, a realistic application of TEEs in Tor would be a brownfield deployment. This presents itself as some Tor relays using TEEs, while others remaining as they are with no change to their behavior. Given this brownfield deployment, Tor circuits can be created with both TEE relays, as well as non-TEE relays.

In order to understand the realistic impact of TEEs, we first model the Tor network as a graph [40]. Each node in our model has a few identifying properties: IP address, bandwidth, and TEE status. The TEE status of a relay is defined by the deployment scenario provided. We implement the Tor relay selection algorithm for circuit establish-

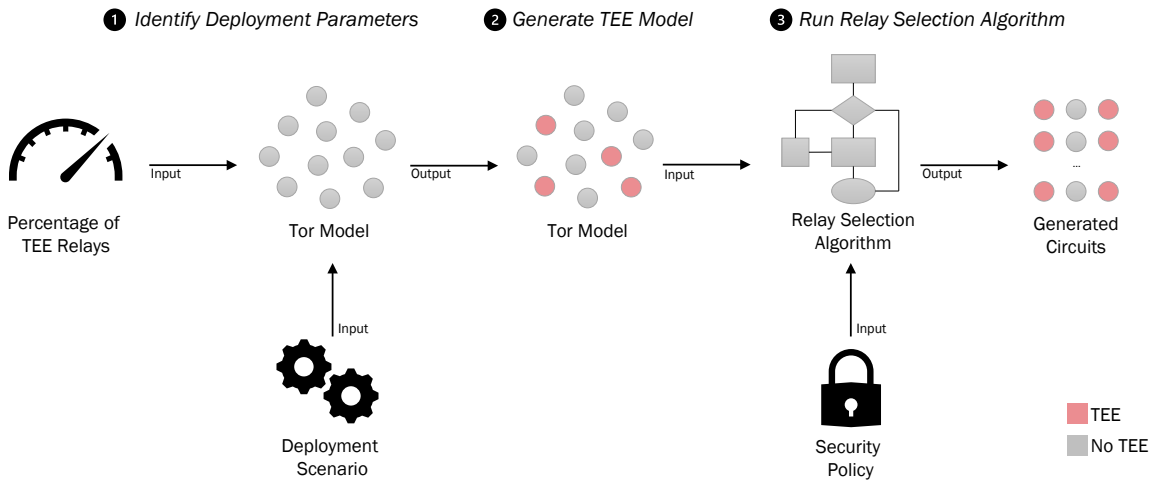


Figure 5.1: Steps for modeling realistic deployment scenarios of TEEs in the Tor network.

ment, with an additional parameter for the security policy of the circuits, defined by the placement of TEEs in the circuit. We generate potential circuits that could be chosen for a user, and analyze the performance of the circuits in terms of bandwidth, and security of the circuits. The deployments and security policies we explore are expanded on below. The steps of our methodology for modeling realistic TEE deployments can be seen in Figure 5.1.

5.2 Deployment Scenarios

To represent how TEEs could be deployed in the Tor network under realistic brownfield scenarios, we incrementally assign relays in the network to be TEE. We randomly select nodes in the model to be TEE under four different deployment scenarios: fully randomized, entry-exit biased, bandwidth weighted, and inverse bandwidth weighted. These scenarios represent how nodes are chosen to be TEE in the model based on different probabilities and characteristics of the nodes. Motivation and implementation is described further below.

5.2.1 Random Deployment

`Random` deployment weights each node in the graph equally, then randomly selects the nodes. No relay-specific characteristics are taken into account in this deployment. We represent this deployment due to the fact that there is no way to truly know what relays

in the Tor network are compatible with TEE hardware. CPU information is private, as this could be identifying information that could potentially deanonymize users. Therefore, we cannot know how TEEs would actually be deployed in the Tor network.

Additionally, this deployment represents the middle ground between the best case deployment `Bandwidth Weighted` and worst case `Inverse Bandwidth Weighted`. This is likely the average scenario for TEEs being present in the network.

5.2.2 Entry-Exit Biased Deployment

We implement an `Entry-Exit Biased` deployment which weights the likelihood of a relay being selected to be TEE around the relays capabilities in the circuit. Relays with a "Guard" flag or "Exit" flag are assigned weight `w`, whereas all other relays are assigned weight `1-w`.

Motivation for the `Entry-Exit Biased` deployment comes from the fact that the majority of attacks on Tor, including those covered in SGX-Tor [11], rely on the entry and exit relays being compromised. For this reason, there is more incentive for entry relay and exit relay volunteers to host TEE relays. We chose not to analyze middle-biased deployments because middle relays are much less significant in enacting attacks on Tor, so there would be little incentive for middle-only relays to be TEE.

5.2.3 Bandwidth Weighted Deployment

`Bandwidth Weighted` deployment weights each relay's probability of being chosen as TEE based on its bandwidth. This approach is much like the relay selection of Tor, relays with a bandwidth two times that of another's are twice as likely to be chosen as TEE.

This deployment is motivated by the relay selection algorithm of Tor. As relays with higher bandwidth are more likely to be chosen for circuits, they are also going to be chosen more often because they can withstand more traffic. This was our intuition behind weighting the selection of TEE relays by their bandwidth; these relays are used more often for users so there is more incentive to be TEE, as this increases the likelihood an average Tor user will be more secure. We consider this approach a best case scenario, in that requiring TEEs will likely increase the performance of a user.

5.2.4 Inverse Bandwidth Weighted Deployment

`Inverse Bandwidth Weighted` deployment weights each relay's probability of being chosen as TEE based on the inverse of its bandwidth. This approach is the opposite of the *Bandwidth Weighted* deployment.

We are motivated for this deployment to understand the worst case scenario in terms of performance when requiring TEEs. The maximum performance degradation is presented in this deployment.

5.3 Circuit Security Policy

The parameter of security policy represents which relays are able to be used in circuits, and in what place in the circuit. More specifically, based on a relay's TEE status, and the position this relay is going to be in the circuit (i.e. entry, middle, or exit), only a select number of relays will be acceptable. If a TEE requirement is provided, a relay then must have the equivalent TEE status in order to be chosen.

For example, if the security policy specified requires only the entry relay to be TEE, with the middle and exit relays having no requirement, the potential circuits generated will be limited compared to that of no TEE requirement specified. As the entry relay is now required to be TEE, only the subset of all entry relays that are TEE are valid for the circuits generated.

In our simulation, we analyze 5 different security policies (i.e., positional TEE requirements): the 4 policies that represent the different combinations of attack mitigations as specified in Table 4.1, as well as the lack of a security policy which represents baseline Tor.

5.4 Extended Relay Selection Algorithm

We implement a weighted bandwidth algorithm to apply to relay selection for establishing circuits, shown in algorithm 1. This algorithm is the algorithm used by Tor today [14], as described in chapter 2. We extend this algorithm slightly, however, to include additional parameters specifying what relays in our circuits are required to be TEE as a concept of security policy, as described above. The algorithm we use to select relays is defined in algorithm 1. We only select relays for positions in which they are able to be in, so only relays with the "Guard" flag in their descriptor are chosen for the entry relay position in

the circuits.

```
Function CircuitEstablishment (G = (V, E), R = (P, T)) :
    circuit = [ ];
    for position, TEEreq ∈ R do
        relaylist = {};
        for v ∈ V do
            if position ∈ v.positions then
                if v.TEE or not TEEreq then
                    add v to relaylist;
                end
            end
        end
        totalBW = ∑_{∀r∈relaylist} r.bandwidth;
        select relay r with probability r.bandwidth / totalBW;
        add relay to circuit;
    end
    return circuit
end
```

Algorithm 1: Bandwidth weighted relay selection for circuit establishment algorithm. $G = (V, E)$ is the graph representing the Tor network and R represents the configuration for the circuit, which contains the relay types (default is Entry, Middle, Exit) and security policy of TEE requirements for the circuit.

Chapter 6
Evaluation

We evaluate the effects of TEEs in the Tor network with our model by answering the following questions: *(1) What is the probability a user is protected from each attack if a TEE security policy is not specified? (2) How do varying deployment settings impact the performance of a user with a required TEE security policy?*

6.1 Experimental Setup

Our strategy simulator was written in Python, using the `networkx` library to model relations between relay nodes. Simulations were performed on a Mac M1 CPU with 16 GB of ram and 3.2 GHz max clock speed. We collect our data of current Tor nodes from the directory consensus document published on April 22, 2022 at 3:00pm [39]. We assume that Internet connections between nodes have unlimited bandwidth (i.e., the bandwidth between two nodes is only limited by the minimum of their individual bandwidths, not by any other link between them).

The vertices each have the properties of bandwidth as reported from the directory authorities, and a set of flags associated with the node's relay descriptor, specifically the 'Exit' and 'Guard' flags. We then add an additional property of trust to each node. This trust label represents the node's TEE capability, set to either true or false based on the different deployment settings that are applied.

We run our modified relay selection algorithm to generate 1000 potential circuits per trial for each security policy. We run each trial 10 times and take the average to normalize our results. For each deployment scenario, we specify the number of TEE relays in a range from 1% to 99%. Depending on the specific deployment, the distribution of TEE relays will vary. For example, in the case of `Bandwidth Weighted`, we selected relays randomly with probability of TEE proportional to the relay's bandwidth. In the

case of `Entry-Exit Biased`, we fix weight `w` of entry and exit relays being selected for TEE status.

In order to understand the effect on security and performance of weight `w` used in `Entry-Exit Biased` deployment, we also iterate over a range of weights from .55 to .95, while fixing the percentage of TEEs in the network.

A summary of the deployments we evaluate, as described in section 5.2, is below:

- Random: fully randomized placement of TEEs in the network
- Entry-Exit Biased: a higher probability is placed on entry and exit relays having a TEE
- Bandwidth Weighted: the probability of a relay having a TEE is based on the its bandwidth
- Inverse Bandwidth Weighted: the probability of a relay having a TEE is based on the inverse its bandwidth

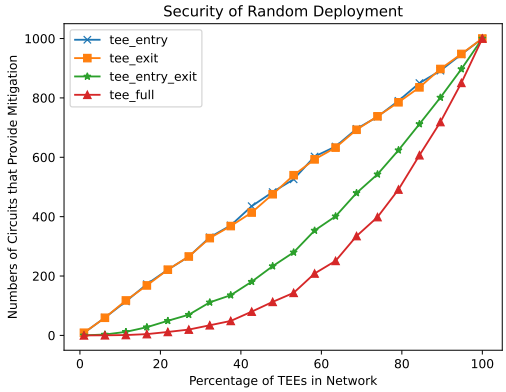
6.2 Security

To quantify how the average Tor user would be more secure in the presence of TEEs in the Tor network, we examine the circuits that were generated with no specific TEE requirements for each deployment setting. We determine how many circuits generated have TEEs and where the TEEs were placed in the circuits. These results reflect the how TEEs would be useful in the Tor network that we see today, with no modification to the relay selection algorithm. Results are found in Figure 6.1.

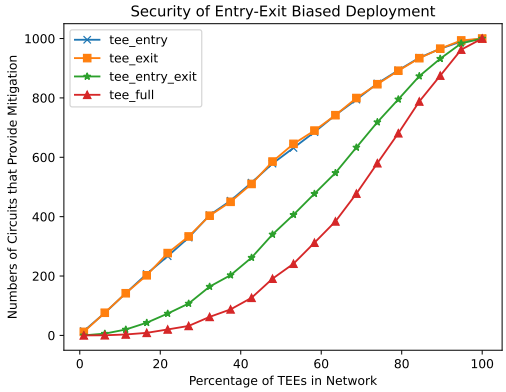
Across all deployments, we see that more circuits generated have only a TEE entry relay, or TEE exit relay, but not both. Full TEE circuits are the least likely to be generated, which makes sense. Middle relays generally have the lower bandwidths out of all relays. If middle relays have TEEs, unless they have a competitive bandwidth, they are not likely to be chosen as often.

6.2.1 Random

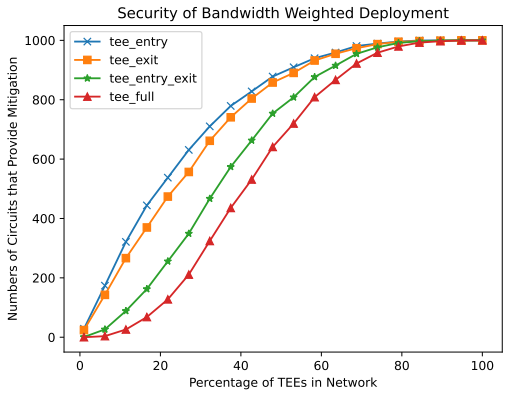
The percentage of circuits generated with a TEE entry relay is almost identical to the percentage of TEE exit relay circuits. This is interesting, but can be attributed to the fact that many exit relays are also entry relays.



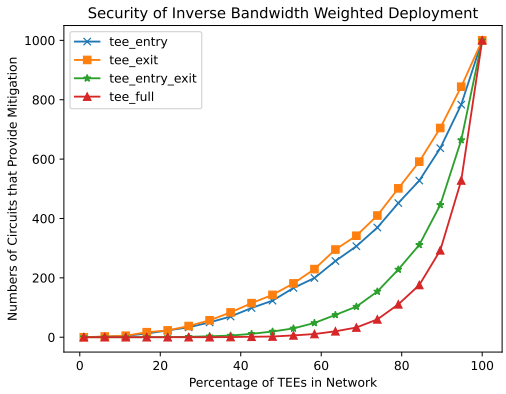
(a) Random Relay



(b) Entry-Exit Biased



(c) Bandwidth Weighted



(d) Inverse Bandwidth Weighted

Figure 6.1: TEE presence in circuits when no required TEE security policy is specified. This represents the security that Tor users can receive in the event the relay selection algorithm cannot be modified. The number of circuits generated is out of 1000.

We see with a TEE presence of 53% in the network, 52.5% of Tor users are assigned a circuit with an entry relay TEE, protecting them from both the Replay attack and Fingerprinting attack. With the same TEE presence, 53.9% of users are assigned a circuit with an exit relay TEE, protecting them from both the Hidden Services attack and Bad Apple attack. 27.9% of users are assigned a circuit that is protected from the Replay attack, Fingerprinting attack, Hidden Services attack and Bad Apple attack. Only 14.3% of users. are protected from all 5 attacks. In order to have more than 50% of Tor users protected from every attack, a TEE presence of more than 80% is required.

6.2.2 Entry-Exit Biased

Iterating over the percentage of TEEs in the network (weight `w` fixed to .7), we see, the results for TEE entry relay circuits and TEE exit relay circuits are nearly identical. Results for circuits with both entry and exit relays TEE are improved from `Random` deployment. This makes sense, as this deployment places a higher emphasis on entry and exit relays having a TEE.

At 53% TEE presence, 63.1% of users are protected from the Replay attack and Fingerprinting attack, while 64.5% of users are protected from the Hidden Services attack and Bad Apple attack. 40.6% of users, though, are protected from all attacks but Bandwidth Inflation. Only 24.1% of users receive protection from all attacks. In order for at least 50% of users to be protected from all attacks, more than 80% of relays in the network need a TEE.

Iterating over weight `w` of entry and exit relays having a TEE, seen in Figure 6.2a (percentage of TEEs in the network fixed to 40%), we see a generally slow, but steady increase in the percentage of circuits generated with TEEs. Ranging `w` from 55-95, the percentage of circuits increases by 43.3% for TEE entry relay circuits, and 43.1% for TEE exit relay circuits. Circuits with both entry and exit relays TEE increases by 104%, whereas full TEE circuits increase by a drastic 168%.

6.2.3 Bandwidth Weighted

`Bandwidth Weighted` deployment provides the best security results out of all the deployments. This is to be expected because of Tor's relay selection algorithm. This deployment scenario increases the likelihood of a TEE relay being used.

At 53% TEE presence, 90.9% of user's circuits had an entry relay TEE, protecting them from both the Replay Attack and Fingerprinting attack. 89% of circuits had an exit relay TEE, protecting against the Hidden Services and Bad Apple attacks. 80.8% of user's circuits have both an entry and exit relay TEE, mitigating all attacks but Bandwidth Inflation, whereas 71.9% of user's circuits are protected from every attack.

These results are significant in showing how many users can realistically benefit from the presence of TEEs in the Tor network today. Despite the best case scenario that these results present, there is a large incentive for high bandwidth relays to use TEEs, meaning this deployment scenario is quite realistic.

6.2.4 Inverse Bandwidth Weighted

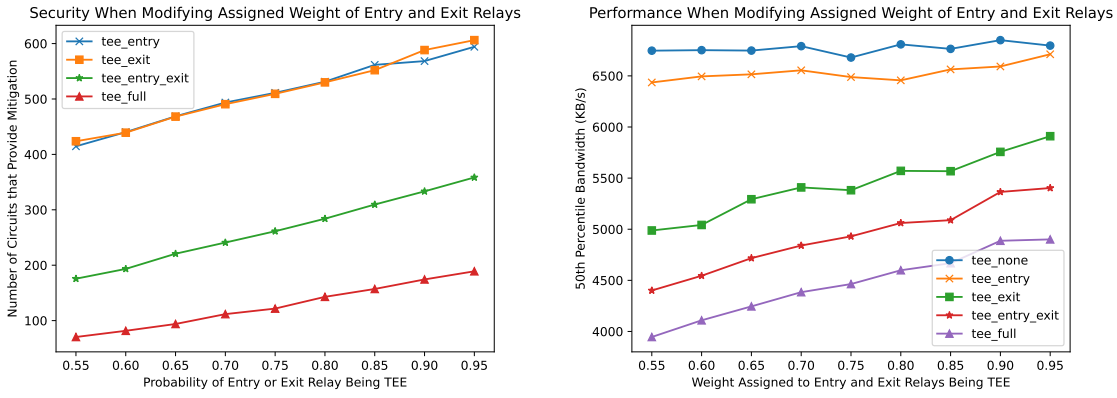
The `Inverse Bandwidth Weighted` deployment presents the worst security results. This is expected, once again, because of Tor's relay selection algorithm. The likelihood of receiving a circuit with a TEE is significantly reduced in this deployment because of the minimal bandwidth they offer the network.

With a TEE presence of 53%, 16.5% of user's are protected from the Replay attack and Fingerprinting attack, while 18.1% are protected from the Hidden Services attack and Bad Apple attack. Only 2.9% of circuits are protected from all attacks but Bandwidth Inflation, while 0.6% of circuits were full TEE, protecting every attack. In order for 50% of users to be protected from all attacks but Bandwidth Inflation, a TEE presence of more than 90% is required. A TEE presence of 95% is required in order for over 50% of users to be protected from every attack.

6.2.5 Takeaways

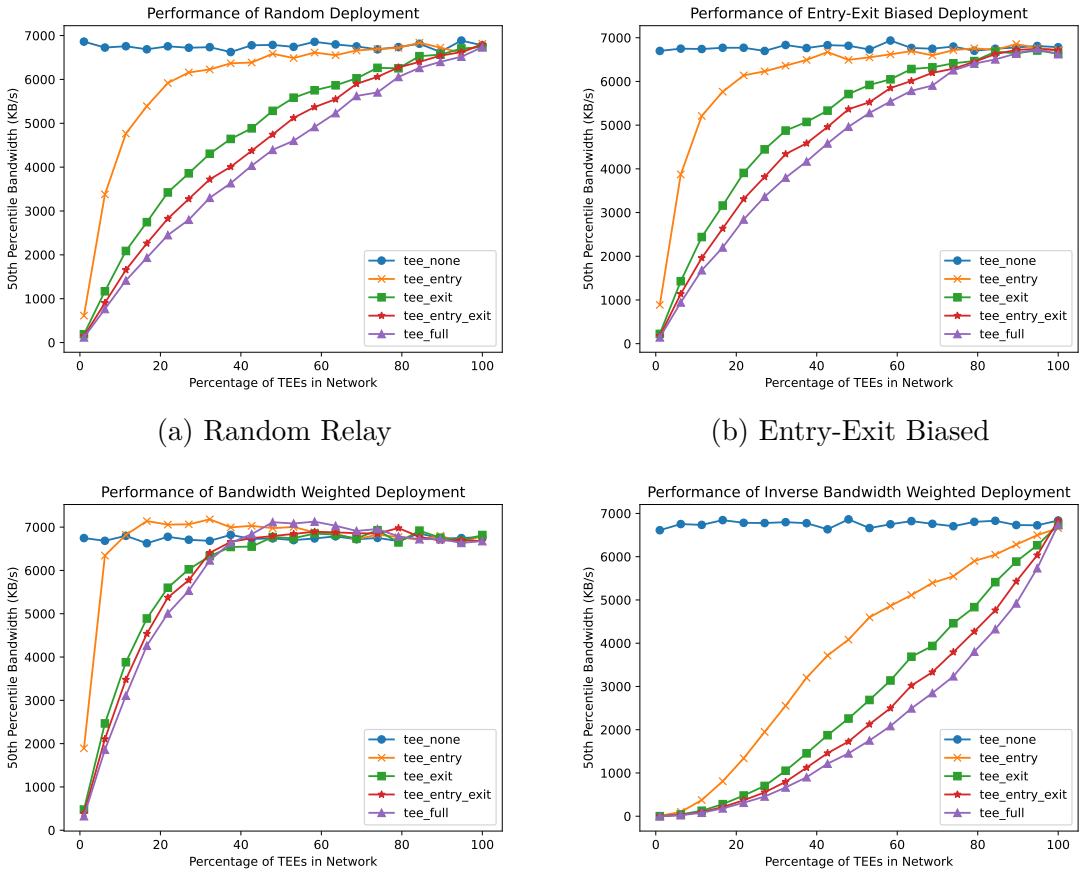
Our results find that the `Bandwidth Weighted` deployment provides the best security for Tor users. This is to be expected, as this is our best case scenario, yet the realism of it exists. With just over 50% of relays in the network using a TEE, more than 70% of Tor users are protected from all 5 attacks.

With a more average deployment such as `Random`, more than half of Tor users are protected from 2 attacks if there is just over a 50% TEE presence. Overall, the presence of TEEs in the Tor network can improve the security of many Tor users.



(a) Security (b) Performance

Figure 6.2: Iterating over the weight that Entry or Exit relays will be running within a TEE. Both the security and performance results are represented here.



(a) Random Relay

(b) Entry-Exit Biased

(c) Bandwidth Weighted

(d) Inverse Bandwidth Weighted

Figure 6.3: The median percentile of bandwidth (KB/s) of circuits generated, when incrementing the percentage of TEEs present in the network.

6.3 Performance

To quantify the cost on performance of placing TEEs in the Tor network, for each deployment, we calculate median bandwidth of all the circuits, which can be found in Figure 6.3. In each figure, 'tee_none' represents no TEE requirement, so any relay can be chosen with no relevance to its TEE status, which represents the expected performance of user in the Tor network today. To normalize this value, we take the average across all the deployments, getting a final baseline bandwidth of 6757.4 KB/s.

Across all deployment scenarios, out of all circuits with a TEE security policy, the policy requiring only the entry relay to be TEE ('tee_entry' in each figure) achieve the highest performance. These results are likely due to the fact that entry relays are required to meet a minimum bandwidth, while other types of relays do not [14]. Only requiring

entry relays to be TEE, then, results in the least reduction (if any) of performance, and guarantees mitigation of both the Replay attack and Fingerprinting attack.

The rest of the TEE security policies have more of a linear increase in bandwidth, as circuit bandwidth is now limited to that of what TEE relays are capable of. Since there is no minimum bandwidth relays must be capable of (besides entry relay's required minimum), performance is reduced more than we saw with entry relay security policy.

6.3.1 Random

We see a steep increase in performance immediately before tapering off with TEE entry relay security policy. For example, the bandwidth at 6% TEE presence is 5.5 times more than that of 1% TEE presence, with bandwidths of 3379.1 KB/s and 613.1 KB/s, respectively. From 22% to 27% TEE presence, though, we only see an increase of 4% in bandwidth.

More notably, at only 22% TEE presence in the network, circuits with a TEE entry relay requirement have a median bandwidth of 5917.8 KB/s. This bandwidth is only a decrease of 12% from the baseline bandwidth. At a higher TEE presence of 53%, there is only a 4% decrease in performance, with a bandwidth of 6486.7 KB/s. As a result, expected performance of users is only minimally reduced, while guaranteeing protection from the Replay attack and Fingerprinting attack. Continuing with a TEE presence of 53%, circuits with exit relay TEE have a bandwidth of 5583.3 KB/s, a decrease by 17% from the baseline, protecting from the Hidden Services and Bad Apple attacks. Circuits with both entry and exit relays TEE have a bandwidth of 5122.6 KB/s, a decrease by 24%. Full TEE circuits had a bandwidth of 4600.8 KB/s, a 32% decrease from the baseline.

These results are critical in showing that with a fully random deployment, which we can consider the average case scenario for deployment, specifying a security policy which requires TEEs only minimally degrades performance of a user.

6.3.2 Entry-Exit Biased

While iterating over the percentage of TEEs in the network (weight `w` fixed to .7), similar to `Random` deployment, we see a steep increase in performance before slowing with the TEE entry relay security policy. A 6% TEE presence had 4.4 times the bandwidth that 1% TEE presence had. From 22% to 27% presence, there is only an increase in bandwidth by 2%.

At 22% TEE presence in the network, circuits requiring a TEE entry relay had a median bandwidth of 6138.3 KB/s, only a 9% reduction from our baseline performance. At 53% TEE presence, TEE entry relay circuits had a median bandwidth of 6551.7 KB/s, a decrease by only 3% from the baseline. Once again, this shows how performance is only minimally reduced, but users are protected from both the Replay and Fingerprinting attacks. TEE exit relay circuits have a bandwidth of 5919.4 KB/s, a decrease in bandwidth by 12%, protecting from the Hidden Services and Bad Apple attacks. Circuits with both entry and exit relays TEE have a bandwidth of 5524.8 KB/s a decrease by 18%. A full TEE circuit had 5278.7 KB/s bandwidth, which is a only a 22% decrease from the baseline and protects users from all 5 attacks.

Iterating over weight `w` of entry and exit relays having a TEE, seen in Figure 6.2b (percentage of TEEs in the network fixed to 40%), we find once again that TEE entry relay circuits have the highest bandwidth out of all TEE security policies, with an average of 6534.6 KB/s. For all other security policies, from 55% to 95% probability, we see an increase in bandwidth by 18.5% for TEE exit relay circuits and an increase by 22.8% for circuits with TEE requirements of both entry and exit. There is an increase by 24.2% for full TEE circuits. This is significant in showing that modifying the probability of Entry and Exit relays having TEE status has does impact the overall performance of a user.

6.3.3 Bandwidth Weighted

The results for `Bandwidth Weighted` deployment are significantly different than the previous two deployments, as this is considered the best case for deploying TEEs in the network. All security policies have a dramatic increase in performance early on before slowing.

We see that at only 11% TEE presence in the network, TEE required entry circuits achieved a bandwidth of 6819.8 KB/s, a higher bandwidth than the baseline by 1% and ensuring protection from the Replay and Fingerprinting attacks. In fact, at only 43% TEE presence, full TEE circuits have a median bandwidth of 6836.5 KB/s, surpassing the baseline by 1% while allowing for mitigation of every attack.

By requiring TEEs in a circuit, this deployment essentially guarantees you will receive the highest performance, while also increasing your security. The baseline, however, reflects the potential for receiving lesser performance under Tor's relay selection algorithm.

6.3.4 Inverse Bandwidth Weighted

We see a significant reduction in performance with `Inverse Bandwidth Weighted` deployment. For the most part, though, all security policies have a more steady increase in performance compared to the other deployments. Circuits with TEE requirements on only the entry relay perform the best, as per all other deployments as well.

For TEE entry relay circuits, a TEE presence of 53% results in a 32% reduction in performance from the baseline, with a bandwidth of 4598.7 KB/s. For comparison, at the same TEE presence, this is the same reduction in performance that is a result of full TEE circuits in `Random` deployment. TEE exit relay circuits result in a bandwidth of 2687.8 KB/s, which is a 60% reduction in performance from the baseline. Circuits with both entry and exit relays TEE provide a bandwidth of 2125.5 KB/s, a 68.5% reduction from the baseline. When requiring a full TEE circuit, a bandwidth of 1751.6 KB/s is expected, presenting a 74% decrease from the baseline.

Ultimately, this deployment scenario yields the worst results. This is to be expected, based on Tor's relay selection algorithm being weighted by bandwidth. Relay operators with little bandwidth would have much less incentive to use a TEE because of their limited use in circuits.

6.3.5 Takeaways

As per the results for security, we see the `Bandwidth Weighted` deployment achieving the best results in terms of performance across all TEE security policies, which we expect considering the best case scenario this deployment presents. If just over 40% of the network uses TEEs, users exceed the baseline performance and have a defense from 5 attacks.

With an average deployment such as `Random`, with just over half of the network using TEEs, users only see a 32% decrease in performance, while receiving protection from 5 additional attacks that are not currently mitigated in Tor. Ultimately, we find that users can achieve increased security with TEEs while only sacrificing minimal performance.

Chapter 7
Conclusion

The Tor network is vulnerable to a wide variety of attacks on its users. Many of these attacks require either the source to be modified, or the exploitation of sensitive user information. Attacks that exploit these vulnerabilities in the network include bandwidth inflation [9, 10], as well as deanonymizing attacks like tagging [4–6], fingerprinting [7], and circuit-linking [8]. SGX-Tor [11] presents a mitigation solution to these attacks by integrating Tor components with Intel's SGX, a trusted execution environment. Their work successfully mitigates the attacks, but in practice, is unrealistic as they assume a full deployment which equates to every Tor relay and user running Tor within a TEE. We presented a security analysis on the effects of using TEEs in realistic deployment settings. Our results found that TEEs do not need to be present in every position of a user's circuit in order to mitigate attacks. Our performance analysis provides an evaluation on the impact on bandwidth when requiring TEEs in a circuit. We found that in a random deployment of 53% of TEEs in the Tor network, a user's performance is only degraded by 31.9%. Ultimately, realistically using TEEs in the Tor network can increase the privacy of users with only a minimal impact on performance.

Bibliography

[1] Dingledine, R., N. Mathewson, and P. Syverson (2004) "Tor: The Second-Generation Onion Router," in *13th USENIX Security Symposium (USENIX Security 04)*, USENIX Association, San Diego, CA.
URL https://www.usenix.org/conference/13th-usenix-security-symposium/tor-second-generation-onion-router

[2] "The Tor Project 2020-2021 Annual Report | Tor Project," https://blog.torproject.org/2020-2021-annual-report/.

[3] "Users – Tor Metrics," https://metrics.torproject.org/userstats-relay-country.html.

[4] Pries, R., W. Yu, X. Fu, and W. Zhao (2008) "A New Replay Attack Against Anonymous Communication Networks," in *2008 IEEE International Conference on Communications*, IEEE, Beijing, China, pp. 1578–1582.
URL http://ieeexplore.ieee.org/document/4533341/

[5] Ling, Z., J. Luo, W. Yu, X. Fu, W. Jia, and W. Zhao (2013) "Protocol-Level Attacks against Tor," *Computer Networks*, **57**(4), p. 869–886.
URL https://doi.org/10.1016/j.comnet.2012.11.005

[6] Ling, Z., J. Luo, W. Yu, X. Fu, D. Xuan, and W. Jia (2012) "A New Cell-Counting-Based Attack Against Tor," *IEEE/ACM Transactions on Networking*, **20**(4), pp. 1245–1261.

[7] Kwon, A., M. AlSabah, D. Lazar, M. Dacier, and S. Devadas (2015) "Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services," in *24th USENIX Security Symposium (USENIX Security 15)*, USENIX Association, Washington, D.C., pp. 287–302.
URL https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/kwon

[8] Blond, S., P. Manils, C. Abdelberi, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous (2011) "One Bad Apple Spoils the Bunch: Exploiting P2P Applications to Trace and Profile Tor Users," *4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET '11)*.

[9] Bauer, K., D. McCoy, D. Grunwald, T. Kohno, and D. Sicker (2007) "Low-Resource Routing Attacks against Tor," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, Association for Computing Machinery, New York, NY, USA, p. 11–20.
URL https://doi.org/10.1145/1314333.1314336

[10] ——— (2007) "Low-Resource Routing Attacks against Tor," in *Proceedings of the 2007 ACM Workshop on Privacy in Electronic Society*, WPES '07, Association for Computing Machinery, New York, NY, USA, p. 11–20.
URL https://doi.org/10.1145/1314333.1314336

[11] Kim, S., J. Han, J. Ha, T. Kim, and D. Han (2017) "Enhancing Security and Privacy of Tor's Ecosystem by Using Trusted Execution Environments," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, USENIX Association, Boston, MA, pp. 145–161.
URL https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/kim-seongmin

[12] Goldschlag, D. M., M. G. Reed, and P. F. Syverson (1996) "Hiding Routing Information," in *Proceedings of the First International Workshop on Information Hiding*, Springer-Verlag, Berlin, Heidelberg, p. 137–150.

[13] "Circumvention and Anonymity | Tor Project," https://blog.torproject.org/circumvention-and-anonymity/.

[14] github (2022), "torspec," https://github.com/torproject/torspec.

[15] Wang, T., K. Bauer, C. Forero, and I. Goldberg (2012) "Congestion-Aware Path Selection for Tor," in *Financial Cryptography and Data Security* (A. D. Keromytis, ed.), Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 98–113.

[16] Imani, M., M. Amirabadi, and M. Wright (2016) "Modified Relay Selection and Circuit Selection for Faster Tor," *CoRR*, **abs/1608.07343**, 1608.07343.
URL http://arxiv.org/abs/1608.07343

[17] Snader, R. and N. Borisov (2008) "A Tune-up for Tor: Improving Security and Performance in the Tor Network." in *ndss*, vol. 8, p. 127.

[18] Zhang, Y. and Y. Xia (2021) "A Dynamic Selection Algorithm of Tor Relay Based on Client Bias," in *2021 22nd Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pp. 41–44.

[19] Kiran, K., B. Vignesh, P. D. Shenoy, K. R. Venugopal, T. V. Prabhu, and M. S. E. Prasad (2017) "Client requirement based path selection algorithm for Tor network," in *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pp. 1–6.

[20] Jansen, R., F. Tschorsch, A. Johnson, and B. Scheuermann (2014) "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network," in *Proceedings 2014 Network and Distributed System Security Symposium*, Internet Society, San Diego, CA.
URL https://www.ndss-symposium.org/ndss2014/programme/sniper-attack-anonymously-deanonymizing-and-disabling-tor-network/

[21] Barbera, M. V., V. P. Kemerlis, V. Pappas, and A. D. Keromytis (2013) "CellFlood: Attacking Tor Onion Routers on the Cheap," in *Computer Security – ESORICS 2013* (D. Hutchison, T. Kanade, J. Kittler, J. M. Kleinberg, F. Mattern, J. C. Mitchell, M. Naor, O. Nierstrasz, C. Pandu Rangan, B. Steffen, M. Sudan, D. Terzopoulos, D. Tygar, M. Y. Vardi, G. Weikum, J. Crampton, S. Jajodia, and K. Mayes, eds.), vol. 8134, Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 664–681, series Title: Lecture Notes in Computer Science.
URL http://link.springer.com/10.1007/978-3-642-40203-6_37

[22] Winter, P., R. Ensafi, K. Loesing, and N. Feamster (2016) "Identifying and Characterizing Sybils in the Tor Network," in *25th USENIX Security Symposium (USENIX Security 16)*, USENIX Association, Austin, TX, pp. 1169–1185.
URL https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/winter

[23] Murdoch, S. J. (2006) "Hot or not: Revealing hidden services by their clock skew," in *Proceedings of the 13th ACM conference on Computer and communications security - CCS '06*, ACM Press, Alexandria, Virginia, USA, pp. 27–36.
URL http://dl.acm.org/citation.cfm?doid=1180405.1180410

[24] Zander, S. and S. J. Murdoch (2008) "An Improved Clock-skew Measurement Technique for Revealing Hidden Services," in *17th USENIX Security Symposium (USENIX Security 08)*, USENIX Association, San Jose, CA.
URL https://www.usenix.org/conference/17th-usenix-security-symposium/improved-clock-skew-measurement-technique-revealing-hidden

[25] Yang, M., X. Gu, Z. Ling, C. Yin, and J. Luo (2017) "An active de-anonymizing attack against tor web traffic," *Tsinghua Science and Technology*, **22**(6), pp. 702–713.

[26] Biryukov, A., I. Pustogarov, and R.-P. Weinmann (2013) "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization," in *2013 IEEE Symposium on Security and Privacy*, pp. 80–94.

[27] Rochet, F. and O. Pereira (2018) "Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols," *Proceedings on Privacy Enhancing Technologies*, **2018**(2), pp. 27–46.
URL https://petsymposium.org/popets/2018/popets-2018-0011.php

[28] Intel, "Introduction to Intel® SGX Sealing," https://www.intel.com/content/www/us/en/developer/articles/technical/introduction-to-intel-sgx-sealing.html.

[29] ———, "Intel® Software Guard Extensions Tutorial Series: Part 1," https://www.intel.com/content/www/us/en/developer/articles/training/intel-software-guard-extensions-tutorial-part-1-foundation.html.

[30] Baumann, A., M. Peinado, and G. Hunt (2015) "Shielding Applications from an Untrusted Cloud with Haven," *ACM Transactions on Computer Systems*, **33**(3), pp. 1–26.
URL https://dl.acm.org/doi/10.1145/2799647

[31] Shinde, S., D. Le, S. Tople, and P. Saxena (2017) "Panoply: Low-TCB Linux Applications with SGX Enclaves," in *NDSS*.

[32] che Tsai, C., D. E. Porter, and M. Vij (2017) "Graphene-SGX: A Practical Library OS for Unmodified Applications on SGX," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, USENIX Association, Santa Clara, CA, pp. 645–658.
URL https://www.usenix.org/conference/atc17/technical-sessions/presentation/tsai

[33] Jain, P., S. Desai, S. Kim, M.-W. Shih, J. Lee, C. Choi, Y. Shin, T. Kim, B. Byunghoon Kang, and D. Han (2016) "OpenSGX: An Open Platform for SGX Research," in *Proceedings 2016 Network and Distributed System Security Symposium*, Internet Society, San Diego, CA.
URL https://www.ndss-symposium.org/wp-content/uploads/2017/09/opensgx-open-platform-sgx-research.pdf

[34] Arp, D., F. Yamaguchi, and K. Rieck (2015) "Torben: A Practical Side-Channel Attack for Deanonymizing Tor Communication," in *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, Association for Computing Machinery, New York, NY, USA, p. 597–602.
URL https://doi.org/10.1145/2714576.2714627

[35] Kim, S., J. Han, J. Ha, T. Kim, and D. Han (2018) "SGX-Tor: A Secure and Practical Tor Anonymity Network With SGX Enclaves," *IEEE/ACM Transactions on Networking*, **26**(5), pp. 2174–2187.

[36] Overlier, L. and P. Syverson (2006) "Locating hidden servers," in *2006 IEEE Symposium on Security and Privacy (S P'06)*, pp. 15 pp.–114.

[37] Evans, N. S., R. Dingledine, and C. Grothoff (2009) "A Practical Congestion Attack on Tor Using Long Paths," in *Proceedings of the 18th Conference on USENIX Security Symposium*, SSYM'09, USENIX Association, USA, p. 33–50.

[38] (2019), "How Bandwidth Scanners Monitor The Tor Network," https://blog.torproject.org/how-bandwidth-scanners-monitor-tor-network/.

[39] "Directory — Stem 1.8.0 documentation," https://stem.torproject.org/api/directory.html.

[40] Jansen, R., K. S. Bauer, N. Hopper, and R. Dingledine (2012) "Methodically Modeling the Tor Network." in *CSET*.