Hardening Tor Hidden Services

Master of Science Thesis
University of Turku
Department of Computing
Cyber Security (M.Sc. Tech)
2022
Pascal Tippe

Supervisors:
Antti Hakkala
University Teacher
University of Turku
Péter Ligeti
Associate Professor
Eötvös Loránd University

UNIVERSITY OF TURKU
Department of Computing

Pascal Tippe: Hardening Tor Hidden Services

Master of Science Thesis, 102 p.
Cyber Security (M.Sc. Tech)
December 2022

Tor is an overlay anonymization network that provides anonymity for clients surfing the web but also allows hosting anonymous services called hidden services. These enable whistleblowers and political activists to express their opinion and resist censorship. Administrating a hidden service is not trivial and requires extensive knowledge because Tor uses a comprehensiveprotocol and relies on volunteers. Meanwhile, attackers can spend significant resources to decloak them. This thesis aims to improve the security of hidden services by providing practical guidelines and a theoretical architecture. First, vulnerabilities specific to hidden services are analyzed by conducting an academic literature review. To model realistic real-world attackers, court documents are analyzed to determine their procedures. Both literature reviews classify the identified vulnerabilities into general categories.
Afterward, a risk assessment process is introduced, and existing risks for hidden services and their operators are determined. The main contributions of this thesis are practical guidelines for hidden service operators and a theoretical architecture. The former provides operators with a good overview of practices to mitigate attacks. The latter is a comprehensive infrastructure that significantly increases the security of hidden services and alleviates problems in the Tor protocol. Afterward, limitations and the transfer into practice are analyzed. Finally, future research possibilities are determined.

Keywords: Hardening Tor Hidden Services, Hardening Tor Onion Services, Vulnerabilities Hidden Services, Security Architecture Hidden Services, Deanonymizing Hidden Services, Hidden Service, Onion Services

Contents

List of Figures

List of acronyms

AS	Autonomous System
BGP	Border Gateway Protocol
DoS	Denial of Service
HSDir	Hidden Service Directory Server
ISP	Internet Service Provider
IXP	Internet Exchange Point
TAP	Tor Authentication Protocol
VPS	Virtual Private Server
WSM	Wall Street Market

1 Introduction

Freedom of speech is essential for every democracy. Everyone must be allowed to express their opinion freely and influence political decision-making processes. Whistleblowers are a prime example: Edward Snowden uncovered illegal behavior from the National Security Agency in America. Instead of punishing responsible executives for breaking laws and invading human rights, the whistleblowers are charged with espionage crimes. Meanwhile, executive powers continue to violate laws or introduce new regulations legalizing invasive actions. [1] Societies must have a corrective allowing people to publish illegal behavior of institutions and express unpopular opinions. Citizens can only make responsible decisions when they have complete information and can access the entire marketplace of ideas. The executive branch and other institutions always have incentives to preserve their power and reputation. Perceptions of opinions also change over time, effectively punishing pioneers.

Due to the digitization of society, the number of networked digital systems is growing immensely. As a result, our society's dependency on these systems is constantly increasing. Citizens communicate electronically with each other and engage in political discussions online. With the new technological possibilities, worries and uncertainties arise that politics and a part of society want to control with totalitarian methods. Opinions and misinformation spread significantly faster, causing increased censorship and privacy invasions. Executive powers and institutions

exploit this situation and establish behavior restricting fundamental rights. Germany has unlawfully practiced data retention and forced telecommunications providers to collect and store communications data without probable cause. While it did not comply with European laws, it took until September 2022 for the European Court of Justice to finally forbid the practice. [2] Meanwhile, they opened the possibility to apply this practice arguing with national security, which effectively allows executive powers to invoke it and let courts check the legitimacy later.

Police and intelligence agencies go one step further and hack suspects' devices. A smartphone incorporates the most intimate thoughts and areas of personal life. In Germany, law enforcement authorities can use malware to observe criminals. Originally, it should target terrorism and other exceptional criminal cases, but practically drug cases make up a large fraction. [3] Even journalists fall victim to malware developed by professional companies for government institutions. In Greece, the government deployed malware on a journalist's smartphone because he reported on corruption scandals. It shows that the executive branch can limit free speech, and national security laws and secrecy limit the power of courts. [4]

Legal measures alone are incomplete since laws and opinions can change quickly. Therefore, technical measures must ensure freedom of speech. Tor is open-source software that allows anonymous communication over the internet. Volunteers provide servers that route connections through them to disguise the origin. It protects users from censorship and ensures privacy. Tor also allows online services to be anonymous with hidden services so that users can share their opinions anonymously and without censorship. Criminals abuse this technology to coordinate crimes or distribute illegal material. At the same time, there are many hidden services dedicated to human rights and freedom of speech, as well as providing information to people in totalitarian regimes. Wikileaks is one of the pioneer organizations. [5] The Tor project and community also use the term onion services for hidden services. Since Tor's official

protocol specifications still use the term hidden service, and academic literature predominantly uses it, the term hidden service is used in this thesis.

However, simply installing Tor is not enough, and hidden service operators, from now on called operators, need to consider more aspects to avoid endangering themselves. This thesis aims to identify the risks specific to hidden services and provide guidelines for operators to harden their hidden services and protect themselves. For this purpose, chapter 2 provides general background information, and chapter 3 explains how Tor fundamentally works. Chapters 4 and 5 collect information about the vulnerabilities of hidden services first. Chapter 4 deals with academic publications since they identify Tor's design limitations and analyze them thoroughly. Well-equipped attackers can exploit these gaps. It includes already fixed vulnerabilities as new techniques could build on them. To model realistic attackers, Chapter 5 examines publicly available internet documents. These provide insight for real-world attackers. Both searches analyze the methods used and classify them into general categories. Based on these, attack vectors for hidden services are derived. Combining academic and practical literature identifies risks comprehensively. Chapter 6 identifies risks for operators based on the previous chapters and provides systematic guidance to determine risk profiles. At the same time, it analyzes existing add-ons for hidden services that increase security. From these results, chapter 7 derives recommendations for operators. First, general guidelines are derived, and later a theoretical technical architecture is developed that significantly increases the security of hidden services. The construction happens step by step to present the advantages and assumptions transparently. It remains theoretical and is not a readymade program. Hence, the implementation must follow later. The following chapter 8 examines the limitations of the research and the security recommendations. In addition, it elaborates on how to transfer the results into practice to support operators and ensure freedom of speech.

2 Theory

2.1 Taxonomy

Multiple concepts are essential for anonymous networks and must be defined to determine security properties. Pfitzmann and Hansen [6] proposed a suitable terminology to standardize several terms:

- Anonymity allows subjects to fit into the anonymity set, which consists of all possible subjects. Actions cannot be traced to an individual but only to the anonymity set, and the possibility to guess the correct one is one divided by the number of all subjects in it. Different actions can have different anonymity sets. For a message, all possible senders could differ from all potential recipients.

- Unlinkability allows subjects to take multiple actions without allowing observers to relate the actions. If they could be linked and have different anonymity sets, an attacker could reduce the anonymity set by intersecting them. For example, an observer cannot attribute messages to one sender or recipient.

- Undetectability refers to the knowledge of existence. An attacker cannot distinguish if the item of interest exists or not. For messages, this means that an observer cannot distinguish them from random noise.

- Pseudonymity allows the identification of subjects via identifiers. These are hard to connect to the real identity and must be protected. The degree of

linkability changes with the context of the pseudonym. A person's pseudonym is used in various contexts by one subject. Role pseudonyms cover only a specific role, i.e., moderating an online forum, and relationship pseudonyms connect to only one communication partner. Role-relationship pseudonyms combine the latter two. While the previous types allow linkability, transaction pseudonyms link only one transaction and qualify for strong anonymity.

2.2 Privacy in Cryptocurrencies

Bitcoin is a peer-to-peer digital currency based on public key cryptography. Addresses are public keys and the owner signs transactions with the corresponding private key. Senders broadcast their transactions into the network, and Bitcoin nodes check their validity. Afterward, they bundle them into a block that references the hash value from the previous block. Finally, Bitcoin nodes need to calculate the block's hash value but must try different nonces to obtain a hash value in a specific format. [7]

While addresses are not directly related to natural persons but are pseudonyms, anyone can access all transaction information and try to combine it to identify the person. To broadcast transactions, users expose their IP addresses. Different attacks exploit the communication protocol to track transactions from a specific node or even replay incorrect transaction information. Some attacks work even when users utilize anonymization techniques. [8] Reid and Harrigan [9] demonstrate transaction analysis determining addresses that might belong to the same users. Additionally, they used online data, for example, posts in Bitcoin forums, to draw further connections.

CoinJoin introduced the concept of mixing Bitcoin transactions to break the clear link between sender and recipient. Multiple users bundle their payments into one transaction to obscure from third parties which input is intended for which output address. A central mix can execute the process, or users organize decentrally. [10] CoinParty is a mixing protocol that shifts payments via multiple sequential

transactions instead of bundling up one large block. Participants can plausibly deny their involvement in the mixing operation. It is tolerant to a small percentage of misbehaving mixing peers. [11]

Monero is a better choice for a private digital currency. It uses a ring signature to prove a sender is part of a transaction without revealing the contribution. Each participant has two key pairs for viewing and sending units, with the concatenated public keys serving as the address. The former allows viewing transaction flows, while the latter allows transferring units. Monero hides the transaction amount while proving that incoming and outgoing sums are equal. Each new transaction includes multiple unrelated old transactions from the blockchain to obfuscate the sender. Users can generate subaddresses from their key pairs to avoid links between posted addresses. Senders transfer units to one-time addresses derived from the recipients' public keys. Thereupon, recipients scan the Monero blockchain for addresses compatible with their private view key and control it with the private spend key. [12]

2.3 Mixnets

Chaum describes a mix network in [13]. It allows sending messages anonymously. A mix collects incoming messages and transfers them simultaneously to prevent leaking information through the chronological sequence. Padding messages to the same size and limiting the number of processed messages prevents observers from linking incoming and outgoing messages. To protect message content and metadata like the recipient's address, senders use asymmetric cryptography.

A sender first encrypts the message M_1 with the recipient's public. The next step is to create a new message M_2. M_2 contains the encrypted message and the recipient's address. Again, the sender encrypts M_2 with the public key of the mix and sends it to the mix. The mix decrypts M_2 and sends M_1 to the recipient's address, which then decrypts it to recover the original messages.

Observers cannot link incoming and outgoing messages because they are indistinguishable. They can only note all senders and recipients. To prevent attackers from encrypting common messages to compare them with the observed data, senders add random data to all messages. Mixes remember incoming messages and senders with hash values to prevent replay attacks.

A dishonest mix knows the identities of senders and recipients and can link them. A mix network combines multiple mixes sequentially. Each one only knows the predecessor and successor so that the first mix knows the sender and the last the recipient. A sender encrypts the message M_1 in layers. First, the sender encrypts it for the recipient, then for the last mix, and its predecessors until the first mix. One compromised mix is not sufficient to link the sender and recipient. An attacker must compromise all involved mixes to uncover this information.

2.4 Onion Routing

Reed, Syverson, and Goldschlag [14] presented onion routing in 1998. It is an overlay network allowing confidential communication over public networks. Connections are bidirectional and long-lasting. During the initialization phase, a client builds up the connection by choosing multiple onion routers from a directory with their public keys. The client sends onions consisting of multiple layers containing the next hop and information for encryption. Onion routers receive and send packets on the client's behalf. The client opens a connection to the first onion router and sends an onion. Afterward, the router decrypts the upper layer and sends the inner onion to the specified address. The second onion router cannot distinguish whether the onion is just forwarded or originates from the first one. This process repeats until the last onion router peels off the inner layer to receive the message and the recipient address.

Dingledine, Mathewson, and Syverson presented 2004 an improved version. [15]

Initially, the client connects to the first router and establishes a shared session key. Afterward, the client extends the circuit by sending packets to the first router that forwards it. The client and second router negotiate a shared session key, and this continues until the circuit reaches the desired length. Afterward, the client sends packets and encrypts them in layers with the shared key. Each router decrypts one layer and forwards the data until the last one in the circuit sends the message to the recipient. The recipient's packages to the sender will be encrypted with each hop and passed the reverse order to the sender.

Onion routers forward the packages and know only the predecessor and successor. After each hop, the packet data changes so that observers cannot link them. After encrypting the initial communication with the onion router's public key, the encryption changes to symmetric algorithms with the shared session keys. It speeds up the transmission significantly. Onion routing enables low-latency bidirectional connections, unlike mix networks.

3 Tor

3.1 Overview

Dingledine, Mathewson, and Syverson [15] developed onion routing into a low-latency anonymous network based on circuits. The Tor network consists of decentral volunteers who provide onion routers. Tor incrementally builds up connections, so-called circuits, instead of using one encrypted data structure with multiple layers. Starting from the last hop, the client extends the circuit to a new onion router and negotiates session keys. While onion routing requires a designated application proxy, Tor uses the SOCKS proxy interface to support TCP-based programs transparently. [16] Previously, onion routing assumed that onion routers permanently flood information about states. Tor uses central directory nodes, called authority nodes, as trustworthy authorities. They vote each hour to create a consensus document describing the state of the network. It includes, inter alia, signed descriptions about onion routers, statistics, and recommended parameters. Major protocol revisions have been made since inception, and the specification is still subject to change. Current protocols still contain legacy options due to the slow adaptation of changes. The following sections in this chapter are based on the latest Tor protocol version according to the specifications in [17].

Onion routers can create a policy to allow only specified traffic types. For example, an onion router can allow all connections or only those inside the Tor network. The

latter avoids legal complaints if clients launch attacks on the internet. Onion routers providing access to the internet can define allowed ports or exclude IP address ranges. One circuit can multiplex multiple TCP streams instead of creating a new one for each stream. ACK cells between client and onion routers prevent network congestion and function as a rudimentary load-balancing function. Moreover, Tor automatically performs integrity checks so that no onion router can change data unnoticed.

The previous mechanisms provide sender anonymity, while the hidden service functionality enables the sender and receiver anonymity. Clients negotiate rendezvous points with hidden services over advertised onion routers. A distributed hash table stores information about hidden services.

Tor's design goal is to provide a low-latency anonymity network that frustrates attackers from linking communication. Deployability enlarges the volunteer base and frees operators from liabilities or thorny installation and maintenance. Usability expands the user base and, therefore, the anonymity set. Flexibility allows the implementation of future improvements without recreating the entire protocol that would break compatibility in a decentral network. Tor does not protect from end-to-end attacks and does not normalize traffic. Instead, clients must ensure that their traffic does not leak undesired information.

The threat model is an adversary who can observe a partial fraction of the traffic, excluding global access, because no low-latency anonymous network can withstand this attack. Active adversaries can compromise onion routers, launch Denial of Service (DoS) attacks impacting the availability, or induce patterns to recognize the stream at a later hop. The Tor project offers a specialized Tor browser preconfigured to avoid browser fingerprinting. Using a standardized browser configuration minimizes the identifying information like installed plug-ins or system language. [18]

3.2 Cell Structure and Cryptography

Tor uses cells with a fixed size of 512 bytes for communication which mitigates traffic correlation attacks. All cells consist of a header and payload. The header fields contain the circuit identifier for connecting related circuits and a command instructing the onion router. Control cells address the onion router directly, while relay cells signal data to forward. The latter contain additional header fields: *stream ID*, *checksum*, *payload length*, and *relay command*. Stream identifiers enable the multiplexing of streams with one circuit. Furthermore, variable-length cells contain a length field indicating their size. Onion routers use them to set up connections and handle authorization and certificates. The payload performs several functions depending on the command values. It can contain padding bytes, handshake data, circuit closing information, or the relay header and body for the next router.

Every onion router has multiple 1024-bit RSA keys: a long-term identity 1024-bit RSA key for signing documents and certificates, a medium-term TAP onion key for decrypting circuit extension attempts, and a short-term connection key for negotiating TLS connections. Besides, onion routers hold a Curve25519 ntor onion key for accepting circuit requests and multiple Ed25519 keys. They contain a long-term master identity key with the sole purpose of signing the medium-term signing key that signs almost everything and, lastly, a short-term link authentication key. The RSA identity key and Ed25519 master identity key uniquely identify an onion router. Connections between onion routers and clients to onion routers are TLS protected. Still, the minimum enforced cipher suite is *TLS_DHE_RSA_WITH_AES_128_CBC_SHA* and suffers from several weaknesses. [19] Participants should negotiate better cipher suites.

Tor offers three ways for TLS handshakes. In the newer two versions, the initiator sends no certificate, and the responder sends a certificate back. Afterward, both parties renegotiate by sending signed certificates for version 2. Both parties exchange

a *VERSIONS* cell to confirm an established connection. For version 3, the responder sends additional cells to prove the identity and a challenge in case the client also wants to authenticate. The initiator uses popular cipher suites in the handshake, and certificates should not leak information to cover the TLS traffic. Finally, the initiator checks the identity key to confirm the identity of the intended onion router. [20]

3.3 Circuit Creation

Circuits are the fundament of Tor. They consist of multiple onion routers that clients can freely choose from the consensus document provided by central authority nodes. The Tor browser and software package use hardcoded references to the nine authority nodes to guarantee integrity. [21] The client connects to the first onion router, called the entry guard, and uses it to extend the connection to a middle relay. The middle relay can only see that the entry guard established a connection. Analogous, the client extends the connection from the middle to the exit relay that forwards traffic to the internet. In the case of a web server, only the exit relay appears in the log files, and the client who initiated the connection remains anonymous. By default, the length of circuits is three hops, but clients can build circuits containing up to 8 onion routers. The current protocol is described in [20].

Initially, traffic from programs is directed to the locally running onion proxy software. It chooses a circuit identifier that is currently unused and initiates a handshake based on the Diffie–Hellman key exchange with the control cell *CREATE*/*CREATE2* to the entry guard. After receiving the cell, the entry guard decrypts the payload, generates a shared key, and sends *CREATED*/*CREATED2* cells back. Afterward, both parties symmetrically encrypt data with the shared key. Tor Authentication Protocol (TAP) and ntor are the two handshake protocols. *CREATE*/*CREATED* should be used for TAP and *CREATE2*/*CREATED2* for ntor handshakes.

For TAP, the client generates the first half of a Diffie-hellman handshake g^x and

a symmetrical secret K. K and a part of g^x are encrypted with the entry guard's onion key. The remaining part of g^x is symmetrically encrypted with K. Now, the onion router creates the Diffie-Hellman part g^y and calculates $g^{y \cdot x}$. Then, it crafts the *CREATED/CREATED2* cell with g^y and derived key data from the shared secret $g^{x \cdot y}$ to prove knowledge.

ntor uses the Curve25519 group. A client needs to know the identity key hash value ID and public ntor onion key B for the onion router. A client generates a Curve25519 key pair x, X. X is the public part, and the client sends it to the onion router OR_j with B and ID. OR_j computes another pair y, Y. Y is the public key. Afterward, ORj computes the value in equation 3.1.

$$\begin{aligned} secret\_input &= EXP(X,y)|EXP(X,b)|ID|B|X|Y|PROTOID \\ KEY\_SEED &= H(secret\_input, t\_key) \\ verify &= H(secret\_input, t\_verify) \\ auth\_input &= verify|ID|B|Y|X|PROTOID|"Server" \end{aligned} \tag{3.1}$$

$PROTOID$, t_key, t_mac, and t_verify are static strings. b is the private ntor onion key, while B is the public one. H is the HMAC function with SHA256 as the underlying hash function. The response contains Y and $AUTH = H(auth\_input, t\_mac)$. The client checks if Y is a member of the group G^* and computes equation 3.2.

$$\begin{aligned} secret\_input &= EXP(Y,x)|EXP(B,x)|ID|B|X|Y|PROTOID \\ KEY\_SEED &= H(secret\_input, t\_key) \\ verify &= H(secret\_input, t\_verify) \\ auth\_input &= verify|ID|B|Y|X|PROTOID|"Server" \end{aligned} \tag{3.2}$$

If $AUTH$ from the response is equal to $H(auth\_input, t\_mac)$, OR_j has successfully proved its identity. Both parties use the shared value KEY_SEED from

which they derive used keys. ntor is significantly faster than TAP.

If both parties have already established an active connection, the client can use the *CREATE_FAST* cell to directly share key material without using new handshakes. The onion router responds with *CREATED_FAST* and additional key material that both parties use to derive keys.

To create a circuit, the client chooses a chain of onion routers $(1, .., N)$ and connects sequentially with a *CREATE/CREATE2* control cell to all onion routers. After establishing a connection to an onion router OR_n analogous to the entry guard, the client extends the circuit to the next onion router OR_{n+1} by creating a *CREATE/CREATE2* cell and encrypting the payload with the public onion key from OR_{n+1}. Following, the client wraps the cell into an *EXTEND/EXTEND2* cell for the last onion router in the circuit. It instructs it to build up a connection to the designated onion router. Again, the client packs the *EXTEND/EXTEND2* into a *RELAY* cell to send it through the circuit to the last hop and encrypts it sequentially with the negotiated symmetric keys for each onion router in it. Each hop decrypts all cells with the shared key. If it cannot identify the payload, it sends it to the next hop until the last one can determine it. The conformation *EXTENDED/EXTENDED2* cell travels the circuit back to the client, and the client can verify the identity and calculate a shared key. While cells travel back in the circuit, each onion router encrypts them with its shared key to shield the content from others. Finally, the TLS connection protects the packets from third parties, and the symmetric encryption covers the cells from participants in the tunnel. Even if an attacker manages to compromise the keys after the connection is terminated, key negotiation provides forward secrecy that results in session keys remaining secure.

On the circuit level, windows control the congestion. The receiver sends a *RELAY_SENDME* cell after 100 cells to the initiator to add 100 to the sending window. For the stream level, the exit relay and onion proxy can indicate their

window also with *RELAY_SENDME* cells. If the cells have a zero stream identifier, they manage the circuit level, otherwise the stream level. [20]

Tor uses a padding mechanism that the client and entry guard implement to impede external observers. It reduces the metadata resolution. A more advanced padding mechanism obfuscates clients connecting to hidden services by matching the cell sequence and count to mimic default circuits for web traffic for the first ten cells. [22]

3.4 Entry Guards

A naive onion router selection algorithm chooses new relays from the consensus for each new circuit. Attackers can inject malicious onion routers, called Sybils, into the Tor network since volunteers run it decentrally. If an attacker controls N out of all k onion routers, the probability of choosing a compromised relay is $\frac{N}{k}$, and for a compromised entry and exit relay is $\frac{N}{k}^2$. With each new circuit, clients have this chance of being compromised due to their selection algorithm. Attackers can confirm if two relays participate in the same circuit with traffic analysis and by comparing the connected IP addresses to the consensus.

Overlier and Syversion [23] described a feasible attack on hidden services. The main idea is that attackers can force a hidden service to create new circuits and check via timing analysis if the target uses a compromised entry guard.

According to the current protocol [24], clients choose a fixed set of onion routers for a fixed time period and use them as entry guards. That decreases the likelihood of Sybil attacks enormously. For the middle and exit relays, clients select randomly from the consensus based on bandwidth and bandwidth-weight. Exit relays must have a permissive exit policy to allow connections to the internet. The authority nodes help clients select appropriate relays by voting about flags of onion routers in the consensus. [25] The following flags represent the most relevant ones:

- Valid - Onion routers run a compatible Tor version and authority nodes did not backlist them.
- Running - Tor authority servers connected in the last 45 min to all advertised ports.
- Stable - Onion routers must fulfill minimum requirements directed to uptime and other faults for a few days.
- Exit - It marks a permissive exit policy to allow connections to the internet.
- Fast - Onion routers must at least offer a certain bandwidth.
- V2Dir - It shows that the onion routers implement at least the version 2 directory protocol.
- HSDir - It signals a hidden service directory server.
- Guard - Onion routers must be fast, stable, V2Dir, and around for a specified time.

Attackers must invest and maintain more resources to deploy entry guards than running middle or exit relays. To choose the concrete entry guards, clients first gather all onion routers from the consensus with the guard flag. Next, clients create a subset *SAMPLED_GUARDS* to limit the maximum number of entry guards a client might connect to in a specified time period. Automatic expiration leads to slow rotations while not accumulating unreachable onion routers. An ordering improves resistance against recently joined Sybils. An additional filter checks for path biases and includes only active, not excluded entry guards in the *FILTERED_GUARDS* subset. The *CONFIRMED_GUARDS* is a subset of *SAMPLED_GUARDS* ordered by priority and contains previously used entry guards. Finally, the set *PRIMARY_GUARDS* stores the entry guards for circuits. It is the intersection of *FILTERED_GUARDS*

and *CONFIRMED_GUARDS*. If this does not yield enough results, the client additionally takes new guards from *FILTERED_GUARDS* according to the ordering.

For a concrete circuit creation, the algorithm chooses the first reachable onion router in *PRIMARY_GUARDS* as an entry guard. If none is active, it picks a new active one from *FILTERED_GUARDS*. The algorithm reorganizes subsets after a new consensus document arrives to handle unreachable onion routers. [24] On November 2, 2022, the authority nodes voted and recommended two primary entry guards. Moreover, authority nodes vote on several parameters for the entry guard selection algorithm that supersede the specification recommendations. [26] The specification currently recommends 120 days to rotate entry guards. [24]

3.5 Hidden Services

3.5.1 Overview and Cryptographic Keys

Hidden services enable servers to stay anonymous instead of solely providing sender anonymity for clients connecting to the internet. For example, operators can provide websites or ssh remote access over Tor to protect the server and client anonymity. The client and hidden service build two circuits connected to a rendezvous point which brings them together according to the current protocol [27]. In contrast to internet domains, hidden services use their base32 encoded master identity key coupled with a version byte and checksum as an identifier. Hidden service addresses end with the suffix *.onion*. Clients fetch the relevant connection addresses from service descriptors.

Hidden services use a variety of keys to communicate securely. The following list mentions them and explains their principal usage:

- Master identity keys are long-term keys for generating blinded keys that operators can store offline.

- Blinded keys are used for signing information for hidden service directory servers (HSDirs) and can be stored offline. Hidden services generate new ones for each time period. They use a nonce to derive the blinded key pair from the master identity key pair. Authority nodes repeatedly generate a new shared random value that hidden services utilize as a nonce. The blinded public key allows no conclusion to the master identity key because it is a one-way function. Operators should cautiously protect the blinded private key because it enables one to infer the private master identity key.

- Descriptor signing keys sign the hidden service descriptor to provide authenticity. It is a Curve25519 key pair, and hidden services must keep them in memory. A signature from the blinded signing key allows clients to authenticate them.

- Introduction point authentication keys ensure the client uses the correct introduction point while the hidden service can check if the contact request came through a specified introduction point. The public Curve25519 key is in the service descriptor and signed with the descriptor signing key.

- Introduction point encryption keys enable clients to encrypt messages for hidden services. They replace the hidden services' public key during some handshake versions. Operators sign them with the blinded private key to prove the authenticity.

- Descriptor encryption keys encrypt hidden service descriptors. The calculation algorithm uses a random value, fixed constants, and variable inputs. The random value in the consensus ensures that a newly encrypted descriptor changes even if the data remains identical.

Hidden services republish new descriptors after 60 to 120 minutes. They create two descriptor versions for clients with an updated consensus and one for the previous time period. [27]

3.5.2 Service Descriptors

Initially, hidden services select between zero and 20 onion routers as introduction points. Hidden services build a circuit to the introduction points and send an *ESTABLISH_INTRO* cell with the newly generated public introduction point authentication key. Also, these cells can include additional parameters to limit the number of *INTRODUCE2* cells. It mitigates DoS attacks. Introduction points report a successful setup with an *INTRO_ESTABLISHED* cell.

Hidden services advertise their introduction points with service descriptors to clients described in the rendezvous specification [27]. These include the introduction point identifier coupled with its ntor onion key and the public introduction point authentication and encryption keys. Afterward, hidden services apply the first layer of encryption with the descriptor encryption key to protect it from unauthorized client access. The algorithm to generate the descriptor encryption key uses the blinded public key and a descriptor cookie if hidden services want to enable client authorization. The latter is a 32-byte secret that clients must know to decrypt the payload. All authorized clients have a common preshared Curve25519 public key and individual credentials that change for each time period. Hidden services encrypt the cookie first for each client with the credentials, and then everything combined with the private Curve25519 key. Then they append this with the encrypted descriptor to a layer1 descriptor. Only authorized clients can decrypt the descriptor cookie to calculate the decryption key for the descriptor.

Hidden services encrypt the layer1 descriptor again with the descriptor encryption key using their blinded public keys to generate the layer2 descriptor. It intends to protect the descriptor from clients not knowing the unblinded onion address, for example, HSDirs hosting it. The final descriptor contains the layer2 descriptor, validity time, protocol version, and a certificate signed with the blinded private key. Additionally, a revision counter specifies the newest version and helps in case multiple

descriptors arrive. If client authorization is not activated, the blinded public key is sufficient to decrypt both encryption layers.

HSDirs store the service descriptors in a distributed hash ring and allow clients to fetch them. The consensus describes HSDirs with a hash value of their Ed25519 identity key, time period information, and the corresponding shared random value. The hash ring orders HSDirs by their hash value. Hidden services can choose a value for *hsdir_n_replicas* from one to 16. It controls how many descriptor replicas hidden services upload. For each *hsdir_n_replicas*, a hash value determines the descriptor position in the hash ring based on the *hsdir_n_replicas* number, time period information, and the blinded public key. Hidden services upload the descriptors to the following four HSDirs in the hash ring. The number is adjustable between 1 and 128, but clients usually rely on the preconfigured values. By default, clients fetch the next three HSDirs for two replicas (*hsdir_n_replicas* = 2). Due to the time period information and shared random value, attackers cannot precisely target a hidden service and deploy HSDirs to block or control the descriptors. [27]

3.5.3 Connection Establishment

Clients can connect to hidden services in multiple steps following the rendezvous specification [27]. First, they must fetch the service descriptor. All onion routers with the HSDir flag and a compatible version (currently 3) can host service descriptors. Every connection attempt requires the blinded public key of hidden services. Clients can derive them from the onion address that must be shared beforehand, for example, via message boards or internet chat messages. Analogous to hidden services, clients choose a random number from 1 to *hsdir_n_replicas*, combine it with the current time period data, and the blinded public key to determine a responsible HSDir. An HTTP GET request to a URL downloads the service descriptor. The URL follows an unambiguous scheme and contains the base64 encoded blinded public key. If the

chosen HSDir is not reachable, clients iterate through the responsible HSDirs. [27]

A rendezvous point connects the hidden service and a client. The client selects a random onion router for this purpose. It connects to it and sends an *ESTABLISH_RENDEZVOUS* cell with a 20-byte rendezvous cookie to identify the hidden service later. The rendezvous point answers with a *RENDEZVOUS_ESTABLISHED* cell. The next step is to contact an introduction point with an INTRODUCE1 cell. Among other fields, it contains the introduction point authentication key. The introduction point checks if it matches an active circuit to a hidden service. In this case, it sends an *INTRODUCE_ACK* cell to the client. Afterward, it relays a copy of an *INTRODUCE1* cell as an *INTRODUCE2* cell to the hidden service. The hidden service checks for a possible replay and compares the introduction point authentication key to the expected introduction point.

An encrypted payload in the *INTRODUCE2* cell contains information about the rendezvous point: the onion key and rendezvous cookie. Furthermore, the client already computed the first part of a ntor handshake and included it in the cell. Now, the hidden service calculates the second part. TAP is also usable but deprecated. Section 3.3 explains a similar ntor handshake. Afterward, the hidden service builds a circuit to the rendezvous point and sends the *REDENZVOUS1* cell containing the handshake reply and rendezvous cookie. In turn, the rendezvous point connects the two circuits according to the rendezvous cookie and relays the handshake data to the client in a *RENDEZVOUS2* cell. The client and server share a secret key and can communicate with each other over the rendezvous point.

4 Academic Literature Review

4.1 Methodology

A systematic literature review of Tor vulnerabilities for hidden services is imperative to understand threat scenarios and how to defend against them. While some academically explored weaknesses might not be directly relevant for currently operating hidden services, they often hint in directions that attackers could exploit already or in the future. The first step is to collect a sufficient number of papers relevant to hidden services. Instead of uncoordinated searching through academic databases and selectively picking references in found publications, this thesis follows a more reproducible approach by collecting all results in the first ten pages of the academic search engine Google Scholar[1]. The results originate from searches on July 2, 2022, for five different phrases that should cover the topic broadly and variate through naming conventions:

- Tor hidden services
- Tor deanonymization
- Tor onion services
- Locating onion services
- Locating hidden services

[1] `https://scholar.google.com/`

Only papers published in journals or conferences will be considered to ensure quality and limit the number of results. It yields 95 unique publications. They require a manual review to assess the relevance and categorize potential threats to hidden services. This analysis disregards surveys due to redundancy and papers unrelated to hidden services. Finally, the 46 remaining publications from 2005 to 2022 will be classified and shortly analyzed to explain the general attack principles.

The Tor project is dynamic and reacts to results from security researchers, attackers, and suggestions from their contributors. Therefore, some research results will be outdated already, sometimes even before publication. Nonetheless, even obsolete results show attack angles that could resurface, or attackers could improve the methods. The categorization helps to abstract from concrete results to general attack vectors that hidden services need to consider.

4.2 Classification

The publications utilize different concrete techniques threatening hidden services. Risk assessments and possible mitigation strategies require generalizing the concrete techniques into categories. Figure 4.1 shows them and how they relate. The boxes with the dashed line represent general categories, while the others contain concrete ones.

Foremost, the distinction ranges from errors in operational security over technical attacks to weaknesses of specific hidden services applications. Insufficient operational security exposes hidden services and operators, for example, by unintentionally leaking metadata via published files or misconfigurations in the webserver revealing the IP address. Applications deployed on a hidden server can unexpectedly interact with Tor and produce undesirable results.

The remaining technical attacks can focus on side-channels like latencies, availability correlation, DoS, HSDirs, or network traffic. Attacks on availability and

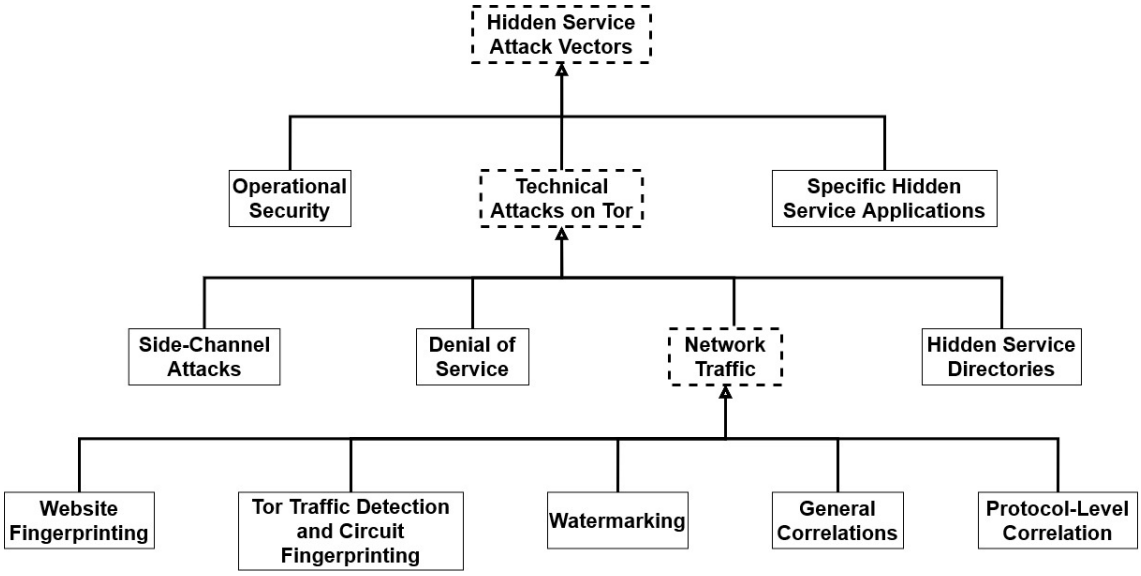

Figure 4.1: Classification of attack vectors from academic literature

HSDirs can overlap. Still, the distinction allows a clear separation of the technical components.

The network traffic offers a plethora of attack vectors. Attackers can guess the visited website with website fingerprinting, identify Tor traffic and circuit types, insert watermarks into circuits or correlate connection traffic. The latter divides into attacks relying purely on general network data or integrating Tor protocol features. The attacker must at least monitor the circuit entry guard and exit relay for watermarking and traffic correlations. Website fingerprinting, identification of Tor traffic, and circuit fingerprinting presuppose weaker conditions: monitoring the hidden service traffic to the entry guard is sufficient.

4.3 Operational Security

Matic et al. present an automized tool, Caronte [28], to extract information about hidden services. They automatically search the website for potentially identifying

information, for example, IP addresses or DNS names on the error page. The content is also relevant as it can contain Bitcoin addresses, unique page titles, or AdSense IDs linking to advertisement accounts. Caronte also examines issued HTTPS certificates as they can reveal IP addresses, DNS names, or the public key. The latter potentially allows attackers to connect different information if it appears elsewhere. A notable misconfiguration is running multiple websites on one host where the hidden service is not the default site. If the client provides a random domain during the HTTP request, the webserver delivers the default website, which might be a standard internet website. Out of 1 974 hidden services, Caronte only exposed 101. But they only counted cases where they could tie an IP address reliably to the hidden service via automatic validation. Real-world attackers have a higher validation rate with manual methods or can even speculate.

Al Jawaheri et al. [29] crawled hidden services to collect Bitcoin addresses combined with a manual collection. With 88 identified addresses, they could link 125 users to 20 hidden services. They identified a few users by combining public data from Twitter and other social media platforms. Maintaining the hidden service infrastructure incurs costs for the operators, especially anonymous journalists and other free speech mediums relying on user donations. Hence, the attackers can try to trace money flows which must take place safely to protect the hidden service and donors. The authors probably underestimate the real threat as they solely assume a passive attacker and renounce test transactions. Their analysis also excludes coin mixing services to only produce reliable results.

PGP is a prevalent encryption suite that offers, for example, encryption and signatures for emails. Without central authorities attributing a public key to a pseudonym or person, actors can sign other keys to express trust. Me et al. [30] scrape public keys from vendors on hidden service markets, extract signatures and consult key servers to conduct a social network analysis showing actors with their

connections. Operators and donors could potentially deanonymize themselves if their keys overlap with information linked to their real persona.

4.4 Hidden Service Directory Server (HSDir)

Steinebach et al. [31] use modified HSDirs to harvest descriptor data and monitor descriptor accesses. An attacker can infer the popularity of a hidden service and opportunistically scan services, even if they did not publish their address publicly. The new version 3 descriptor stores only blinded public keys from onion services. HSDirs cannot link them.

But Eaton et al. [32] mention attackers can calculate the blinded keys and track the hidden service if the onion address is known. They state that attackers can correlate multiple blinded keys via the access and timing distributions. If a hidden service uses a malicious middle relay for uploading the descriptor on a compromised HSDir, the attacker determines one entry guard. If attackers compromise the entry guard, they can deanonymize hidden services. This attack applies directly to the current version.

Hoeller et al. [33] show that attackers can still inject malicious HSDirs in the current Tor version and infer statistics about access rates of a hidden service if the onion address is known. Especially hidden services that serve one specific user and purpose can leak side information like access times that could be critical.

Tan et al. [34], [35] demonstrate an eclipse attack on the distributed hash table that resolves onion addresses on an outdated Tor version. They place malicious HSDirs into it by exploiting the predictable distribution algorithm. Afterward, they will hold the hidden service descriptors and block them selectively. So, no clients can connect to them anymore. Hidden services determine the responsible HSDir deterministically, so attackers can guess the location and generate fitting HSDirs that will be responsible. By blocking only a fraction of the requests, attackers can

significantly decrease the quality of service.

In 2013, Biryukov et al. [36] published a method to harvest directory data and deanonymize hidden services efficiently for an outdated Tor version. Tor used a predictable algorithm to distribute the hidden service descriptors among the HSDirs. Attackers could exploit this by inserting malicious HSDirs with specific public keys that will become the responsible HSDirs for a defined hidden service. It threatens the availability as they can stop serving the correct descriptor.

Currently, Tor uses a nondeterministic way to determine responsible HSDirs and uses blinded public keys. It prevents harvesting directory data and predicting responsible HSDirs for hidden services. Additionally, it shields the descriptor from HSDirs if they do not know the corresponding onion address.

4.5 Website Fingerprinting

Website fingerprinting is a specific traffic analysis applied to Tor connections. Often it is focused on the client. Different websites exhibit unique patterns, for example, packet size, frequency, and destination. It can leak enough information to determine the website from the encrypted traffic. Tor encrypts and pads the traffic to make these attacks harder. This attack also applies to hidden services, where potential attackers spy on the connection link. A data center provider, for example, could try to identify which servers are using Tor and host a hidden service. Attackers can sift through logs to compare against a specific website fingerprint to validate if a server hosts a specific hidden service website.

In [37], Pachenko et al. simulate an attacker who listens to the link between the client and the entry guard. Their approach requires two phases: first, they check if the connection from a client targets a web or hidden service website. In the latter case, they compare the fingerprint against known fingerprints to expose the visited website. While phase one delivers reliable results, the second phase is less successful.

The recognition rate is high in small subsets but drops significantly in larger sets of hidden services.

Yang et al. [38] present an active fingerprinting attack that delays HTTPS requests to improve the detection rate. It targets clients browsing websites on the internet via Tor and delays the packets for the HTTPS request. Actively compromising the connection allows for better accuracy in contrast to the passive attacker just eavesdropping on the client. In an open world setting with 2 000 monitored websites in the list of 10 000 websites, attackers can distinguish the correct monitored website with roughly 91 % true positive rate while keeping the false positive rate below 4 %. The attacks focus on clients, but they could also try to detect servers hosting hidden service websites.

Overdorf et al. [39] further analyze fingerprinting results of different classifiers. The accuracy and precision improve by combining multiple classifiers. In particular, they analyzed website features that lead to high or low-ranking results. Their datasets consist of 482 hidden services from the year 2017. For 47 %, they archive more than 95 % accuracy, while the algorithms classify 16 % with less than 50 % accuracy. Big and static websites seem especially vulnerable to fingerprinting.

Sun et al. [40] develop another fingerprinting approach. By applying Frequency-Domain Analysis and neural networks, they achieve high precision and recall in an open-world setting of 40 000 internet websites. They also considered the WTF-PAD defense technique that delays packet arrival times and inserts dummy traffic. Attackers can tune the proposed fingerprinting methods either for precision or recall. The attacker reaches more than 95 % for the tuned metric, while the other degrades to about 75 %. The attack did not target hidden services but regular web services. With fewer hidden services in the sample and the detection of hidden service traffic, this attack becomes even more dangerous.

4.6 Identification of Tor Traffic and Circuits

Another attack vector is to identify the encrypted connection as Tor traffic. Mayank and Singh [41] show that Tor varies from usual TLS connections by the certificate fields and used ports. The traffic can reveal that a client uses Tor, and attackers could specifically target or block this connection. Sapruta et al. [42] also exploit the TLS certificate fields' subject and issuer information to detect Tor communication in the network.

Gurunarayanan et al. [43] improved existing classifiers to detect Tor traffic inside a network. Especially their random forest classifier had a precision, recall, and accuracy of over 99 %. It indicates a reliable detection that data centers could also apply to detect servers hosting hidden services. Montieri et al. [44] analyze the traffic of the anonymity networks Tor, I2P, and JonDonym from the Anon 2017 dataset that contains traffic from these three anonymity networks. The researchers can reliably distinguish the traffic between them with over 97 % accuracy. Certain types of application layer traffic, for example, streaming or browsing, can only be detected with an accuracy of up to 73.99 %.

Circuit Fingerprinting uses characteristics of Tor circuits to distinguish between different types. Kwon et al. [45] show how duration, initialization sequences, number of cells, and direction leak information about the circuit type. Attackers need circuit-level information visible to the entry guard or, under certain conditions, to the internet service provider (ISP). Subsequently, they can distinguish between normal internet activities and hidden service connections reliably. An attacker can further infer that a user operates a hidden service. In combination with website fingerprinting, the authors achieved notable results among a list of 1 000 collected hidden services.

In another study, Mitseva et al. [46] successfully replicate the results and propose a random forest classifier to make circuit fingerprinting more robust. Platzner et

al. [47] use circuit fingerprinting to determine if the circuit between a targeted hidden service and an introduction point uses a malicious entry guard. First, with a circuit fingerprinting technique, they determine if their controlled onion router is an entry guard between an introduction point and a hidden service. Then they request connections over these circuits with controlled client requests that form a recognizable pattern over time. Their method is outdated, but possible further improvements make this attack vector still relevant.

4.7 Watermarking

Inflow [48] is a watermarking technique that exploits the traffic congestion mechanism in Tor and TCP. It induces a pattern of silent communication periods that entry guards of a hidden service can detect. The attacker needs to control the entry guard of a hidden service and a malicious client. The client connects to the hidden services and stops acknowledging incoming TCP packets until the TCP window fills up and the relays across the Tor circuit to the hidden service stop sending packets. Before the connection times out, the client starts to transmit packets again. The entry guard of a hidden service can detect the pattern of silent periods, but distorting the traffic to cancel the watermark is impractical. With prolonged time between silence periods, the attack provides more stealth. The challenge is to target a specific hidden service with this attack as it needs to connect to the malicious entry guard. Several clients can iterate through many hidden services and build connections with the watermark to decloak unspecified hidden services. Wang et al. [49] show how the Tor padding mechanism allows inserting unique patterns into the traffic to signal to the client entry guard. Principally, it applies to hidden services as well.

Ho et al. [50] exploit the congestion control mechanism in Tor with *SENDME* control cells. The client downloads resources from a hidden service, starts buffering packets after some initialization time and sends a batch of *SENDME* acknowledgment

cells to signal successful reception. Attackers place malicious entry guards and detect if a circuit contains the watermark to link the connected IP address to the hidden service. The detector can recognize the watermark if it is present and the hidden service uses the malicious entry guard.

Chen et al. [51] combine existing strategies to develop an efficient attack called SignalCookie on all hidden services. Instead of targeting just one hidden service, their model allows parallel attacks. Controlled clients connect to different hidden services and pick colluding rendezvous points. The client sends a cookie to the rendezvous point to build up the connection to the hidden service, and the rendezvous point, by default, cannot distinguish which hidden service is involved. But the clients embed information in the cookie used by the rendezvous point to insert a watermark into the circuit stream towards the hidden service. Different malicious onion routers can determine this pattern if they participate in the circuit. If it is an entry guard of a hidden service, then it deanonymizes the hidden service. They can also detect if they are in the middle position and then uncover the entry guard of the hidden service that can be attacked or forced to cooperate. They collected data about entry guards of hidden services in live experiments: more than 83 % of the observed hidden services use 20 % of all entry guards in four months. Worse, the geographic distribution is mainly limited to a few countries. Germany hosts over 26 % of all entry guards and France almost 23 %. Together both provide entry guards for around 50 % of the observed hidden services.

In another publication, Chen et al. [52] used the data to classify hidden services that probably belong to the same operators. Most hidden services rely on the default Tor configuration. Using specific onion routers as entry guards for an exceptionally long time for multiple hidden services can hint at a unique administrator. If multiple hidden services run over one onion proxy, the guard selection is identical, and over a prolonged time, this delivers a precise correlation to relate them. Or using private

bridges is also a strong indicator and even ties the hidden service operator to the bridge operator. The content and address prefixes for onion addresses also reveal clues. It is problematic for hidden services as an attacker only needs to deanonymize one hidden service to endanger all connected hidden services. It can happen if the hidden services leak information about the operator or an attacker poisons all entry guards.

4.8 Side-Channel Attacks

Murdoch [53] discovered a side-channel attack on hidden services. The idea is to generate traffic on hidden services that leads to different CPU temperatures resulting in slightly different clock skews. If attackers know a candidate set of potential servers, they can probe timestamps via TCP and link the physical device to the hidden service. Zander and Murdoch [54] improve the method by utilizing HTTP timestamps and synchronizing the probes with the traffic load. Attackers can even possibly estimate the hosting location of servers by comparing daily temperature changes to the clock skew.

In case a hidden service resides on an onion router, attackers can use the attack described by Murdoch and Danezis [55]. The idea is to insert traffic patterns and measure the latencies of all onion routers to discover the circuit relays. Subsequently, the latencies on the hidden service circuit relays will also measurably change. Hopper et al. [56] show that an attacker can infer information in the Tor network traffic by using the same method to measure the latencies of onion routers to correlate circuit relays. The authors aim to deanonymize clients by generating many HTTP requests to a malicious hidden service. This method also applies to hidden services if enough traffic flows from client to server (for example, file uploads).

Simioni et al. [57] correlate the uptime of a server with public availability information. They deployed a server in the I2P network and artificially took the

server offline. I2P is a different anonymization network, but general results can also be transferred to Tor. They measured the availability of nodes in intervals and compared the hamming distance. After four days, they could reliably identify their server. A similar approach is viable for the Tor network: Attackers could potentially influence the availability of several suspected servers and try to access targeted onion addresses. If the target hidden service is among the candidate servers, then the deanonymization is successful.

Another attack, shown by Shebaro et al. [58], in this category is to send a fixed number of requests in intervals to leave fingerprints in the log files. If an attacker gains access to a machine, the log entries can reliably identify that a specific server hosted the hidden service.

Cangialosi et al. present a framework called Ting [59] that builds circuits to measure the latencies of individual onion routers. This information allows attackers to infer additional information. For example, if an attacker connects via a colluding rendezvous point to a hidden service, the exit relay from the hidden service circuit to the rendezvous point is known. The attacker also measures the overall circuit latency from the rendezvous point to the hidden service. Therefore, the combined entry and middle relay latency must be less than the total circuit latency minus the known exit relay latency. It decreases the set of possible onion routers involved in the circuit. If the attacker determines a second involved onion router in the circuit, the remaining candidate set is small due to the Ting latency measurement precision.

4.9 Specific Hidden Service Applications

Gao et al. [60] carry out a more specific attack against Bitcoin nodes that conceal their identity as a hidden service. Bitcoin is a decentralized system connecting different Bitcoin nodes to spread transaction information, and administrators can protect their identity by running a Bitcoin node as a hidden service. The attack

proceeds in two stages: first, an attacker determines the IP address of the hidden service, and in the second step, connects the transaction to it. In the first step, the hidden service needs to use a malicious entry guard, and a client will insert a watermark in the connection. The entry guard detects the watermark and then knows the IP address if the hidden service uses this entry guard. The entry guard delays packets from the hidden service to tie the corresponding transaction. If the malicious Bitcoin node receives the transaction information from the target node first, it must come from the exposed hidden service since other nodes insert a random delay that is less than the induced delay.

Hellebrandt et al. [61] mention a possible attack on the Tor authority nodes to manipulate a majority vote. Following this, an attacker can create a forged list of compromised onion routers. These could lead to mass deanonymization, but this attack is noisy when large parts of the consensus document change.

4.10 Denial of Service (DoS)

Janes et al. [62] discovered a protocol flaw in Tor enabling DoS attack on arbitrary onion routers. The attacker must control a client that builds circuits in the basic version. Furthermore, the attacker must manage the circuit exit relay or an external server. The idea is to send packets into the victim's packet queue by continuously sending packets that violate the signaled TCP window boundaries while the client (or malicious server) stops reading from the connection. An advanced setup requires only a single client that downloads large files, stops reading from the TCP connection to the entry guard (except keep-alive signals to maintain the connection), and signals the exit relay to continue sending packages by crafted flow control packets. The entry guard's packet queue overflows and consumes memory until the process terminates. It becomes delicate in combination with the entry guards: the attack may crash the hidden service's entry guards, forcing the hidden service to select new entry guards.

It increases the chance that the hidden service chooses malicious ones. While Tor fixed this concrete vulnerability, DoS attacks remain relevant and offer novel attack surfaces, for example iterating through different entry guards for hidden services.

Döpmann et al. [63] mention the computational asymmetry during the connection initialization between the client and hidden service. It is costly for a hidden service to build circuits and calculate the initial handshake. An attacker could use bogus data and directly connect to the introduction point to send a connection request to the targeted hidden service. Subsequently, it needs to conduct expensive calculations that attackers can exploit to launch DoS attacks. Barbera et al. [64] describe the computational asymmetry initially by just calculating costs for creating connection requests for clients and hidden services. A client needs a fraction of the computational power incurring a tremendous resource consumption on targeted onion routers.

Jansen et al. [65] explore other DoS attacks combined with cost estimation. The naïve approach overloads an onion router with traditional packet flooding, which is comparatively expensive at more than 7 200 000 U.S. Dollars for the entire Tor network needed during their 2019 analysis. The other attack utilizes Tor protocol features. By building up longer circuits with eight hops, they included targeted relays multiple times in a circuit to amplify attacks. An attacker can severely limit the connection quality of targeted relays even with limited resources. In particular, if the entry guards of a hidden service are known, they can be accurately attacked so that the hidden service selects new entry guards. Another compelling approach they presented is to attack the Tor bandwidth measurement system. For around 2 800 U.S. Dollars, they can severely impact the Tor measurement system for a whole month. Meanwhile, attackers can insert malicious onion routers and inflate their offered bandwidth by claiming false information. It increases the chances that clients will choose them.

4.11 General Network Traffic Correlation

In 2007, Bauer et al. [66] demonstrated how onion routers can lie about their bandwidth to increase their chances of getting selected. When they compromise the entry guard of Tor clients or hidden services, they correlate traffic and check if a malicious exit relay is involved. If this is not the case, the entry guard destroys the circuit, and the client selects new onion routers until the attacker pushes the selection of the compromised exit relays.

2011, Zhang et al. [67] used clients to connect to hidden services with the HTTP 1.0 protocol that builds a new TCP connection for every object to load. A malicious entry guard of a hidden service can try to correlate the requests from the attackers' clients with the connected relays to see if the pattern matches. If the hidden service directly connects to the malicious entry guard, then it is deanonymized.

Li et al. [68] exploit the selection process. During the time of the publication, Tor maintained a list of entry guards for 30 to 60 days to make attacks on the entry guard selection process harder. It contained three entry guards by default. If an entry guard becomes unavailable, the hidden service chooses a new one to ensure availability. An active attacker observing the network connections of Tor clients and hidden services can learn the entry guard list from the connection logs and selectively block connections. The attacker will wait for the reselection of entry guards, and if the client or hidden service does not choose a colluding entry guard, the attacker repeats the process until it selects a controlled entry guard. The attacker can increase stealth by blocking connections when there are no ongoing connections and only blocking two of the three guards to allow the Tor proxy to work undisturbed.

The internet consists of Autonomous Systems (ASs) that internally route packets individually but advertise standardized interfaces to other ASs to ensure smooth routing. An AS can capture packets from clients. If an AS can observe the communication between a client and entry guard and exit relay and destination, the AS

could correlate traffic to determine the destination and source. Internet Exchange Points (IXPs) are locations to connect different ASs and could potentially wiretap the connections. Johnson et al. [69] modeled typical user behavior and attackers running relays, ASs, or IXPs to measure Tor's security. Using data monitored between 2012 and 2013, they simulated Tor's path selection for users with different behaviors. The users' locations also influence how fast an attacker can successfully deanonymize at least one circuit. Assuming the worst-case user location: after 90 days, almost 100 % were connected to a compromised circuit by an AS attacker. In the best user location, between 38 % and 67 % after 90 days by an AS attacker. For IXP organizations (controlling multiple IXPs), around 95 % in the worst case after 90 days and between 15 % to 17 % for the best user location after 90 days. Juen et al. [70] measure with *traceroute* in which ASs onion routers reside and how vulnerable the path selection is. Even with their improved path selection algorithm, the average client would have a mean probability of between 5.3 % and 11 % for a vulnerable path in one week. While they used the browser behavior of an average internet user and ran simulations, the results also concern hidden services.

Sun et al. [71] systematically analyze how ASs can deanonymize Tor circuits with traffic correlation attacks. ASs need to wiretap the traffic between the client and entry guard and the exit relay and destination. Interestingly, the internet path can differ for incoming and outgoing packets. It increases the ASs' chances of a successful attack as they could wiretap the connection between the client and entry guard on the forward route and exit relay and destination on the reverse. They correlate the cleartext TCP sequence number and TCP acknowledgment number to determine the client and destination. They achieved a 95 % accuracy without false positives in a limited real-world study. 21.3 % of all Tor circuits during their experiment in 2015 were vulnerable as a single AS could deanonymize clients, considering churn in the routes over time, even 31.8 %. ASs use the Border Gateway Protocol (BGP)

to distribute routing information across different ASs. Attackers can hijack IP prefixes by issuing unauthorized routing updates, and neighboring ASs will update their routing tables accordingly. With more specific prefixes than the target range, the malicious routing information gains more relevance than the authentic route information. The authors successfully demonstrated an attack and showed that previous BGP attacks affected Tor.

4.12 Protocol-Level Traffic Correlation

Ling et al. [72] present a protocol-level attack on hidden services. They need to control the entry guard of a hidden service, a rendezvous point, and a client. The client chooses the malicious rendezvous point to connect to the hidden service. If their entry guard receives packets to build up a circuit to a rendezvous point, the attacker's rendezvous point inserts a corrupt package that leads the hidden service to close the connection. A central server logs times and packet information about network packets. Furthermore, the server logs packets signaling the closing of the circuit at the entry guard, rendezvous point, and client. The hidden server can detect the corrupt packets and notify the operator about the ongoing attack.

Ma and Xu [73] present a scheme for deanonymizing clients connecting to a malicious hidden service via a colluding entry guard. The hidden service injects specially crafted cells into the circuit that a compromised entry guard will detect. This attack utilizes protocol features. Similarly, another attack variation inserts conspicuous cells at the entry guard of a hidden service and detects these at the exit relay or client. Problematic is that according to Tor's specification, protocol modifications lead to the termination of streams, but bugs or vulnerable protocol designs can lead to different behavior. Moreover, even the termination reveals information. Still, protocol feature flaws often remain temporary.

Yoshiura and Koizumi [74] show that the number of network packets varies

for an entry guard depending if a client or hidden service connects. Therefore, a malicious entry guard can collect the IP addresses of potential hidden services without associating them with their hosted content. They also mention that there are only two entry guards in Japan at the time of publication, indicating a skewed geographical distribution. Hidden services connected to these might stand out.

Tan et al. [75] combine different attack vectors to deanonymize Tor clients even faster. Initially, the attacker needs to monitor the entry guards of a client by wiretapping the network connection. ISPs, ASs, or data center providers have this capability. Then they push the client to select a compromised entry guard by selectively blocking connections. Afterward, the attacker must ensure that the client selects a compromised exit relay. Then the entry guards collect a fingerprint of the initiated circuit and compare it with colluding exit relays. If the client chooses a benign exit relay, the compromised entry guard breaks the circuit. Afterward, the client initiates a new one with a new exit relay. The attacker repeats until the user selects a compromised exit relay. Following, a traffic correlation attack on the entry and exit relay will reveal the source and destination of Tor clients. Hidden services also initiate circuits to the rendezvous points, and an attacker can apply this attack to deanonymize them.

Greubel et al. [76] measure the load distribution in the Tor network. The Tor proxy chooses onion routers randomly, but the chances increase with higher offered bandwidth that onion routers can self-report. The mechanism that checks actual bandwidth does not work reliably for onion routers with fast bandwidth. An attacker can insert malicious ones and inflate their bandwidth to increase the likelihood of clients connecting. They also mention a strong recentralization of Tor to a few onion routers in the study. Attackers can focus resources on high-priority targets to deanonymize mass Tor traffic and hidden services.

5 Internet Literature Review

5.1 Methodology

Next to the theoretical attack vectors determined by an academic literature review, strategies from the real-world attacker are also relevant. An internet search supplies information about previous attacks on hidden services. Each source must be critically examined since anyone can freely publish information online. Initial search attempts yielded poor results, like multiple newspaper articles rephrasing press releases from law enforcement. Blog posts, newspaper articles, and YouTube videos often neglect technical investigative details, and the variety of authors coupled with little background information makes it difficult to assess the reliability of this information. Naturally, hackers will usually not publicize their attacking techniques, and interview answers cannot be verified. Therefore, this analysis concentrates on court documents because they partially describe how investigators located criminals and involved law enforcement agents face punishment if they intentionally make false statements. While law enforcement must respect legal barriers, their budget and persistence should make them more dangerous than hackers and make them comprehensible model attackers.

Many countries restrict access to court documents, and law enforcement has no intention to reveal their methods. Otherwise, criminals can adapt their cover

strategies. The United States Department of Justice provides a website[1] with a search function for court documents and press releases. The number of records is comparably low and resembles more highlighted cases with public interest than a representative collection. The analysis includes all results from the first ten pages on July 10, 2022, for the following three key phrases:

- Darknet
- Hidden Service
- Tor

An initial search showed that these three keywords were commonly used in criminal cases that involved the Tor network. The overview does not include incidents with coincidental Tor involvement, but only cases with strong Tor technology involvement. Investigators leverage different techniques depending on the offenses. All described incidents fall under 3 major categories that later help to determine the various investigative angles: drug offenses, hidden service platforms, and others. During the research, three court cases emerged as particularly relevant to operators of hidden services, each of which a separate section analyzes.

5.2 Drug Offenses

Operation Dark Gold targeted drug vendors on marketplaces by undercover agents posing as money launderers. In one case, they offered to exchange cryptocurrencies for banknotes mailed via Post. Agents then obtained a warrant to surveil the address and exposed the drug dealer. [77]

A pharmacist sold drugs over a hidden service marketplace. Through multiple undercover purchases, undercover agents posed as active users and got placed on

[1] `https://www.justice.gov/`

a mailing list. They received regular updates directed to all customers via email. The PGP key tied the vendor and email address. Investigators determined that the pharmacist used an online marketing service to manage email communication with customers. Subsequently, the requested record from the service showed logins from an unprotected IP address, billing address, and telephone number. Surveillance teams captured the pharmacist sending drug packages, and the Bitcoin payments were tied to his accounts on a cryptocurrency exchange platform via transaction analyses and access times. [78]

A comprehensive blockchain analysis provides hints for another drug case. While investigators tracked the stamps of intercepted drug packages to identify the responsible person, a blockchain analysis showed a connection between undercover purchases and cryptocurrency exchange accounts registered in the suspect's name for more than 100 transactions. The defendant likely used a cryptocurrency mixer, but some clues incriminate him in court. [79]

US law enforcement apprehended a drug vendor in 2021 operating under multiple pseudonyms. Initially, investigators became attentive after a man received a drug package at a post office in 2018. He denied wrongdoing, but prior relevant convictions led to a search of his home. His computer was powered on and indicated "darknet vendor activity". They tracked various drug packets and confirmed with surveillance camera footage that the man had committed crimes. [80]

A notable organized action called HunTor to fight online drug trafficking resulted in 150 arrests worldwide. Again, the press release does not mention any explicit investigation methods but links to the relevant court documents. [81] An indictment for an alleged drug dealer reveals that an undercover agent used the given Bitcoin address to link transactions. The monitored Bitcoin wallet sent Bitcoins to a cryptocurrency exchange account and was used to book a hotel. It linked the person to the Bitcoin address and thus the pseudonym. From another involved

cryptocurrency exchange, the investigator analyzed records showing even the number of accesses and IP addresses. [82] Another interesting affiliated case shows how the FBI linked a drug vendor to different profiles by comparing pseudonyms, PGP keys, and messages from seized servers. Several hidden purchases determined common characteristics to connect the various profiles. Afterward, investigators traced the Bitcoins and discovered a transaction to a Bitcoin exchange involving a personal email address. Provided records showed access times and the IP addresses, which linked the defendant. The FBI determined that the defendant used the Tor network by wiretapping his internet connection. [83]

5.3 Hidden Service Platforms

The Hydra market appeared in 2015 and was shut down in 2022 by the combined effort of U.S. and German law enforcement. The platform laundered cryptocurrencies and enabled criminals to sell illicit items and services. According to the press release and indictment, undercover agents used the money laundering service to follow the money flow. Moreover, they used ordered drugs from vendors. The administrator ran a hosting company to cover the servers for the Hydra market. The indictment lists some of his invoices and how often he logged into an affiliated email account. [84]

A Canadian citizen in Thailand hosted AlphaBay, the most popular marketplace at the time, which authorities seized in 2017. In 2014, new users received an introductory email that included a standard internet email address in the header. The password recovery process included the same email address. The records from the email provider led to the administrator's identification, who posed as a successful entrepreneur with technical background on LinkedIn. Before the marketplace launched in 2014, a user post under the same pseudonym appeared in a French forum in 2008. The pseudonym is unique and was also the administrator's alias on AlphaBay. The forum post also

included his real name, and the email later appeared in the email header. Law enforcement artificially influenced the servers' availability to lure him into restarting it from his laptop to catch him red-handed. The police confiscated the laptop quickly to get an unencrypted view of the marketplace's infrastructure and assets. At least five different servers handled the cryptocurrency wallets in encrypted containers. [85]

Freedom Hosting was a Tor-based hosting provider allowing users for free to set up their hidden services. Without further explanation, U.S. law enforcement surveilled two websites containing illegal pornography that leaked an IP address connected to a hosting company in France. After receiving a search warrant, the servers forcibly shut down to copy the disks, but the containers were inaccessible due to encryption. The billing address led to a mailbox in the U.S. that redirected the post to an Irish address. The administrator paid the server with his debit card registered in his real name. The access logs from the hosting provider showed a single IP linking to a virtual private server (VPS) that, in turn, revealed the administrator's residence. The server kept log files for remote connections that referenced his VPN account and previous Irish IP addresses. Authorities cracked the password and gained access to the encrypted containers giving insights into the infrastructure. In chat logs, administrators of the hosted websites incriminated the Freedom Hosting administrator for knowingly tolerating their actions. [86]

Helix was a cryptocurrency mixer laundering Bitcoins from 2014 to 2017. The administrator cooperated with drug dealers and other marketplaces to conceal the money flow. Simultaneously, the administrator ran a search engine called Grams. Undercover agents used the Bitcoin laundering service. The document gives no further information. [87]

A notorious child exploitation hidden service leaked its IP address due to misconfigurations in the page source code. Investigators also traced Bitcoin payments from undercover agents to cryptocurrency exchange accounts from the administrator.

They stored identifying information about him. He set the server up in his private house, and during the police search in 2018, they confiscated further incriminating evidence on his computer. [88]

Law enforcement seized a platform offering illicit pornographic material operating on the Tor network. Previously, the website had a standard internet version, and investigators tracked the payments for the hosting server to an account registered in the administrator's name. The given email address also showed frequent access from a dutch IP address and content about administrating the website. [89]

A case that attracted a great deal of attention was the child pornography website PlayPen. Undercover agents surveilled activity on the website and gathered intelligence about users and hierarchical structures. Police forces from another country seized a similar hidden service promoting child pornography, and during the investigation, they identified a moderator account linking to PlayPen. Private and public text messages established this connection. Law enforcement sent a link to a streaming site to the moderator on the seized website. After the administrator opened this link and the video file, the file transmitted the IP address and other identifiers over the internet to law enforcement. After receiving information from the ISP, police searched the administrator's house and arrested him. [90]

Rather than shutting down PlayPen immediately, the FBI moved the server image to their servers and continued hosting the hidden service for about two weeks. They exploited a security vulnerability and included malware to expose users down- and uploading content. The exploit generated a unique identifier for each computer and collected the hostname, operating system username, MAC address, IP address, time, software versions, and whether it was already running on the target system. [91]

5.4 Other Offenses

A user sold insider stock trading tips on a marketplace and later on his hidden service. Undercover agents subscribed to his service and gained trust over multiple conversations. By referring to a new wealthy potential customer, the FBI lured the criminal into encrypted voice calls that later helped to confirm the suspicion. They identified him by tracing the Bitcoin payments, some of which he directly transferred to exchange platforms and cashed out. He tried to conceal other transactions through dummy transactions ending at other exchange platforms. There he needed to identify himself. Law enforcement used these records that directly led to the insider trader. [92]

A woman paid a hidden service to have her husband killed by a hitman. A hacker gained access to chatlogs and transaction data from the website and provided this information to law enforcement. They, in turn, traced the Bitcoin payments to an exchange platform account that revealed information about the client ordering the hit. Investigators tied her pseudonyms to her kids' names and car registration numbers. Private emails from her account aligned with the messages to the hitman. [93]

A U.S. government employee tried to sell confidential data to a foreign government by initiating contact via post. The package contained a sample and instructions for electronic communications with keys. The intended receiver turned it over to the FBI, which, in turn, set up an undercover operation attempting to lure the criminal into serving a surveilled dead drop. At first, he insisted on using electronic communication and the anonymous Monero currency but later agreed to deliver an SD card physically. The FBI monitored and successfully identified the perpetrator and continued the encrypted conversations and payments for a while before apprehending the criminal. [94]

In 2021, a U.S. court sentenced a resident to prison for charges including admin-

istering a child pornography forum. The press release mentions that he came to the attention of law enforcement due to incriminating material during a search for unrelated online offenses. [95]

Law enforcement exposed a marketplace moderator by meticulously connecting information. In the first step, they linked accounts on the marketplace and the discussion forum. Secondly, they compared similar slang, posting times, content, and written languages to establish a link. During a covert operation where police ran a darknet marketplace, a user with the same username registered and asked for employment. The police carefully extracted background information and lured him into providing a postal address. The given recipient name refers to a person in Brazil and a registered company. The company's contact email address links to an online hacking forum and social media accounts. The uploaded profile photos matched, and one shows a book in the background. The moderator mentioned this book in a discussion in the marketplace forum. Furthermore, investigators discovered his eBay account buying credit card skimming equipment. [96]

5.5 Extensive Court Cases

5.5.1 Silk Road

The notorious Silk Road hidden service operated from 2011 to 2013 and was the first major marketplace. In the official version of events, the FBI identified the Silk Road servers in Iceland because a captcha leaked the IP address unassociated with any onion router. After connecting directly to it, a part of the Silk Road login page appeared that incriminated this IP address. The responsible agent claims it is a configuration error bypassing the Tor proxy. Afterward, the hosting provider monitored the servers' network connection which indicated a high volume of Tor traffic. Furthermore, they imaged the server, and the FBI analyzed the content. Investigators discovered forum

messages, records of sales, and the source code. Configuration files lead to the IP addresses of two backup servers. The FBI decided to surveil their suspect, Ulbricht, by wiretapping routing data from the ISP, including IP addresses, ports, and MAC. The suspicion was corroborated by correlating the monitored ISP information with the online times of the operator. [97]

In an interview with a news agency, investigators stated that after locating the Silk Road servers, they tracked a person logging into administrative areas with the username frosty in an internet café. Another investigative breakthrough was the cooperation of a Silk Road staff member. Undercover agents took over his account and gathered intelligence. Another agent searched online for clues about the Silk Road operators. Subsequently, he connected an early post advertising Silk Road and a hiring request on another forum by the same username. The latter contained an email address with Ulbricht's full name. They could also tie another post offering coding contracts for hidden services to that username, but the name changed later to frosty. The U.S. Customs and Border Protection intercepted fake passports ordered by Ulbricht. After being confronted, Ulbricht mentioned Silk Road and abruptly moved to another state afterward.

The undercover agent posing as staff could connect the phrase "yea" frequently used by the Silk Road administrator with Ulbricht's YouTube account. Furthermore, the access times of his private email and the operator's online times correlated strongly. To gain unencrypted access to his laptop, agents surveilled him waiting for him to contact the undercover agent on Silk Road. During the chat, agents staged a fight while others grabbed his laptop. Cybersecurity experts immediately took care of it. Ulbricht's girlfriend also knew partially about his involvement in Silk Road [98]

Ulbricht's version differs significantly from the official version. In his motion submitted to the court [99], he included documents showing that law enforcement initially suspects other individuals as the responsible operators by applying linguistic

profiles. Specific words like “lemme”, “intellectual laziness” or “agorist” combined with general spelling contribute to a linguistic profile. Ulbricht also includes chats with a claimed law enforcement employee offering him information about the investigation. The proclaimed source said investigators tried to target vendors and staff members. Also, the investigation used a psychological profile based on login times, speech, ideologies, and linguistics. Analysts permanently crawled Silk Road, and undercover agents tried to impersonate the operator’s account by similar names or claimed false identities when contacting potential contacts to resume an old conversation. Furthermore, the source speculated that law enforcement started a DoS to incite a response from the Silk Road operator that could lead to mistakes. The motion also includes allegations that U.S. law enforcement wiretapped Tor exit relays.

Ulbricht’s lawyers released documents showing how investigators captured the IP addresses of the Silk Road servers. According to security specialists, the publicized logs are inconsistent with the official version of events. Silk Road used two servers as a front and back end. The back end could only communicate with the front end, which, in turn, acted as the hidden service for clients. Direct contact with the IP address could not lead to the mentioned login page but to an authorization error. The recorded log files also hint at a phpMyAdmin page rather than a captcha or Silk Road login page. [100]

Another internet blog concludes from the files that the back-end server denied requests from the internet but established a TCP connection to deliver the unauthorized access page. A systematic internet scan could reveal information from the SSL Certificate. Even worse, the PHP configuration section overrides the general rules for .php files, which means that the phpMyAdmin was accessible straight from the internet, and the phpMyAdmin page used HTTP. An adversary could snoop on the transmitted password and log into the page. The log file showed successful access to the phpMyAdmin page, which, according to the blogger, was either falsified or

the FBI knew the correct password. [101] This fuels suspicions that law enforcement used a "parallel construction" that builds up a separate investigation line to present at court to conceal initial evidence.

Ulbricht required users working for him to scan and send their identification documents to him. Law enforcement retrieved them and subsequently arrested participating persons. An advisor helped Ulbricht to develop a cover story for people he talked about Silk Road. By changing the username, the consultant also suggested creating the illusion of multiple operators under a single account and that he possibly sold the marketplace to someone else. Additionally, he advised Ulbricht to enforce encrypted messages between vendors and buyers. [102] A rouge agent misused the account of an arrested user to divert Bitcoin into his possession during the Silk Road investigation. [103]

5.5.2 Silk Road 2.0

Five weeks after the Silk Road shut down, Silk Road 2.0 launched as a successor. Again, before the investigation experienced a breakthrough, undercover agents crawled through the website and gathered intelligence. The founder invited several people to a Tor discussion forum to work on Silk Road 2.0. Law enforcement placed an informant in this inner circle that continuously forwarded the information to staff members.

The breakthrough came when investigators revealed the IP address of the server. The affidavit used in court does not mention how they discovered it. In a separate case against a Silk Road 2.0 drug vendor, Brian Farrell, the affidavit refers to the "FBI NY Source of Information" that provided IP addresses for the Silk Road 2.0 servers, 17 other hidden service marketplaces, and 78 IP addresses that accessed the Silk Road 2.0 vendor portal. Silk Road 2.0 maintained different hidden services for the marketplace, vendors, forum, and support interfaces. It reflects a compartmentalized

server architecture. [104]

The documents do not directly state that law enforcement agencies broke Tor anonymization. The fact that the various hidden service pages and even Tor users were deanonymized indicates this. The IP identification took place from January 2014 to July 2014, which perfectly aligns with the security notification from the Tor project. The Tor project identified malicious relays operating from January 2014 to July 2014 engaging in traffic confirmation attacks. [105] In 2016 after the administrator was identified and charged, additional documents proved that Carnegie Mellon University uncovered his IP address. [106]

After identifying the Silk Road 2.0 servers, the authorities imaged the server and analyzed the content. During the process, the provider took the server temporarily offline. Simultaneously, Silk Road 2.0 was not reachable anymore. It further confirmed to investigators that they imaged the correct server. Configuration files and in-depth messages between users and operators revealed the inner working to investigators. The operator used a personal email address linking to his IP address to register the servers. Also, support tickets in the hosting company logged the same IP address. The hosting provider kept logs of a second IP address for issuing tickets leading to a hotel where Blake Benthall resided at that time. Similarly, a third IP address was linked to him again during another hotel stay.

Emails in his account incriminated him further, for example containing staff-only messages. Investigator found his tweets mentioning the Silk Road returning and noted his steady financial income from Bitcoins. At the same time when Silk Road 2.0 appeared, his account at the cryptocurrency exchange started to receive large Bitcoin transfers. Investigators showed that Blake Benthall used the same software in the same version to access the Silk Road 2.0 support interface and the cryptocurrency exchanger. His browser and operating system version hint at him. At the last stage, law enforcement surveilled him physically and correlated his account being online on

Silk Road 2.0 with his physical activities. Months before law enforcement arrested Blake Benthall and shut Silk Road 2.0 down, he wrote to his staff members that an attacker hacked the servers and stole all Bitcoin funds. He used his personal financial resources to cover the loss and aimed to recoup the losses with new commission profits. [107]

5.5.3 Wall Street Market

Comprehensive court documents from a recent criminal case reveal with what methods attackers can deanonymize hidden services. Wall Street Market (WSM) offered a marketplace for drugs, malware, stolen data, and other illicit goods. The FBI purchased drugs from vendors covertly to target drug dealers and gain insights about WSM. In 2019, a joint force of U.S., Dutch, and German law enforcement agencies managed to apprehend the operators and shut the market down. Initially, Dutch investigators imaged a server that hinted to WSM with code variable names, SQL queries, and comments. The document does not mention how investigators identified the servers. The description suggests that this server was only responsible for processing cryptocurrency transactions. WSM divided the infrastructure into multiple servers fulfilling only part of the complete functionality. The configuration files hinted at various IP addresses in Germany.

The German authorities copied a WSM database referencing the names of the previously imaged server. They redacted concrete details from the public document of how they identified the servers initially. An infrastructure analysis yielded a link to the development server in the Netherlands and a database containing communication between the operators. Subsequently, Dutch investigators obtained an image and linked it to WSM because it contained the source code. All three configured administrator accounts have similar names to the operators' names on WSM. The operators used VPN providers to access the development server.

The first operator made a fatal mistake when the connection to the VPN provider ceased, but his connection to the development server continued, thus exposing his IP address. The IP address led to a mobile UMTS Stick registered under a false name. Investigators tracked the locations and identified a person based on the login location at the residence and employment place. A similar GitHub name, information about cryptocurrencies, and personal preferences related to the username led to the sentencing of this person.

German investigators correlated connection timings between the second operator to the VPN provider and the VPN provider to the WSM server infrastructure that led to a person. Later the person in question confessed his role in WSM. The third operator reused a PGP key on another marketplace for an account. This account used a Bitcoin wallet that paid for services from a media marketing agency registered with personal details. Also, he bought a video game for an account registered in his name. The administrator also bought video games from an unrelated Bitcoin wallet and refilled it later from WSM wallets. All three operators could be linked by tracing Bitcoin transactions to a previous marketplace German Plaza Market that shut down in 2016. [108]

5.6 Classification

The mind map in figure 5.1 depicts the seven fundamental categories for investigative approaches. All these present attack vectors that hidden service operators must bear in mind and actively anticipate. The overview below presents concrete methods for the seven attack vectors:

- Vulnerable Tor Specifications
  - Malicious entry guards continuously try to correlate hidden services to connected IP addresses

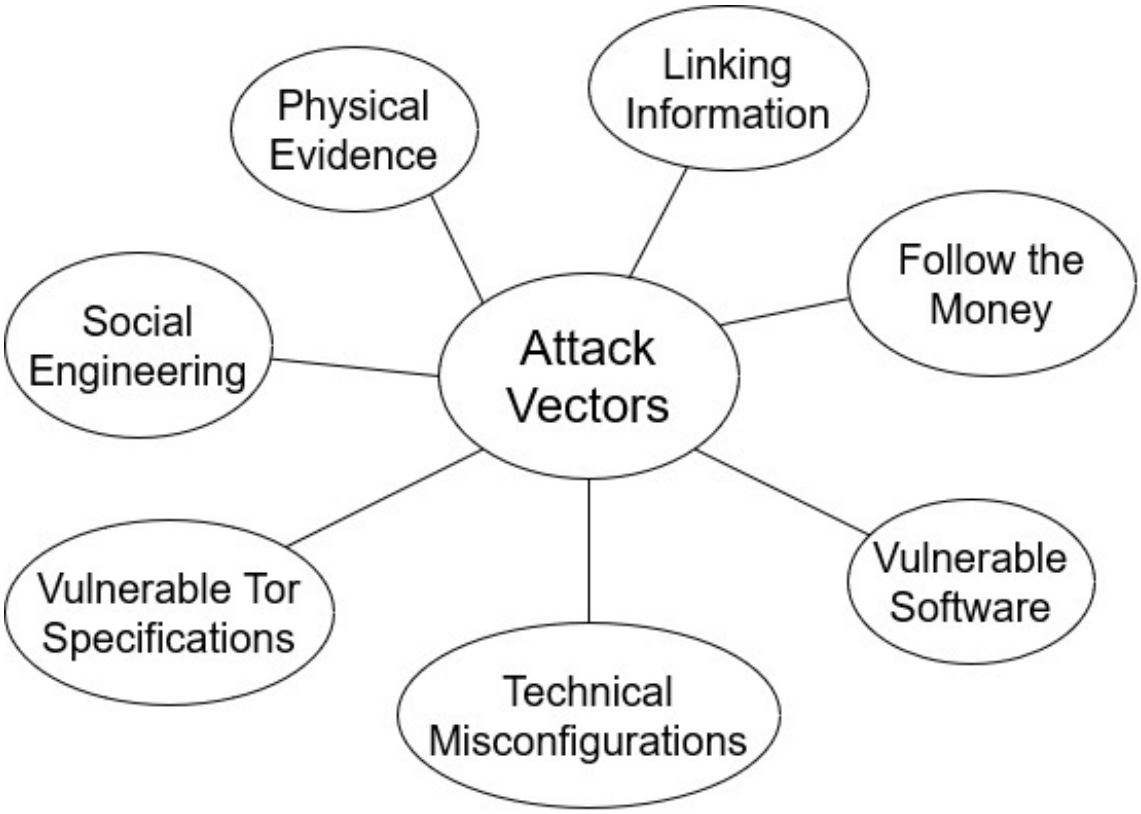

Figure 5.1: Classification of attack vectors from internet literature

  - Network traffic correlations indicate Tor usage in local networks

- Technical Misconfigurations

  - Weak passwords allow attackers to take over accounts or access administrative areas
  - An IP scan of the server can expose a hidden service
  - Configuration files and the source code provide insight into the remaining hidden service infrastructure
  - Embedded resources or the hidden service itself can leak IP addresses or other metadata

- Vulnerable Software

  - Program libraries used and self-programmed software may have vulnerabilities
  - Attackers utilize exploits against the Tor Proxy or Tor browser

- Follow the Money
  - Some cryptocurrencies have a blockchain that publicly stores all transactions that eventually lead to a wallet associated with a person
  - Payments of involved wallets give further hints about involved persons

- Linking Information
  - Electronic signatures uniquely identify persons across pseudonyms
  - The username, date, and content of posts can indicate that a person might be involved in a hidden service
  - Linguistics, login times, and expressed ideology facilitates the creation of a personal profile
  - Public information, for example, from social media, allows attackers to compare data against an operator to reduce the number of suspects
  - The hosting provider or other affiliated services (i.e., email address) deliver additional angles

- Physical Evidence
  - Searched devices reveal files referring to the usage of hidden services
  - Local network eavesdropping makes it possible to correlate Tor traffic usage with activity patterns of operators
  - Physical surveillance can catch suspects red-handed or at least build strong correlations

- Social Engineering
  - Impersonating accounts can trick the victim into divulging additional information
  - Phishing attacks can grab relevant information or even passwords
  - Inciting stress on hidden services via availability attacks or server restarts can push the operators to make errors
  - Undercover agents, arrested users, and infiltrated hidden services persistently gather intelligence and can manipulate operators

The number of attack vectors is significantly higher than the list. It only highlights methods used by investigators to catch operators and decloak hidden services. Therefore, this list focuses more on attack vectors specific to hidden services. Chapter 4 details the vulnerable Tor specifications, and many technical vulnerabilities of hidden services also apply to default web services. Generally, the documents give the impression that non-technical methods or simple misconfigurations usually expose a hidden service. Economically, an attacker initially tries to exhaust cheaper and faster methods before dedicating tremendous resources to crawl the internet to compare linguistics or slowly infiltrate the organization.

6 Risk Assessment and Security Add-Ons for Hidden Services

6.1 Risk Assessment Process

Every hidden service and operator faces multiple risks that damage assets. For hidden services, the most relevant assets are anonymity and data because this is a core function of the Tor network. Hidden services differ from companies because they often act in sensitive fields like whistleblowing. Deanonymization is the dominant risk, and faults entail severe consequences for real people. Therefore, it is imperative to deal with risks methodically in order to implement sufficient protective measures and emergency plans.

Every hidden service is also a web service that utilizes the Tor software as an additional layer. Therefore, by default, all properties of web services also apply to hidden services. Like companies, operators should use a structured process to identify and counter potential risks. It helps to cover them comprehensively and adopt appropriate measures. The National Institute of Standards and Technology from the U.S. offers a detailed guide. [109]

A risk describes potential harmful effects on an entity and consists of the impact and likelihood of occurrence. Subsequently, a risk model incorporates threats, vulnerabilities, predisposing conditions, likelihood, and impact. Threats can be

human errors, natural disasters, or vulnerabilities. Adversaries differ highly in their financial, personal, and technical resources. Vulnerabilities describe weaknesses in information systems and organizational structures and, in a broader view, also include human errors or vulnerable hardware. Predisposing conditions describe existing factors that affect the impact of risks. Likelihood states the probability that a threat realizes and impacts the assets in a defined timeframe. The likelihood of threat realization is adapted depending on the adversary's motivation, resources, and targeting. Impact predicts the damage to assets.

Operators should assess their adversaries carefully. Hobbyists and petty criminals usually pose a low risk. Organized criminal networks and hacktivists have significantly more resources at their disposal. Security researchers and governmental institutions can spend enormous resources over a prolonged period on attacking hidden services. While known vulnerabilities with readymade exploit scripts are usually easy to mitigate, advanced attacks on the Tor specification require special attention and are not entirely preventable. [110]

6.2 Risks for Hidden Services

Common threats apply to all servers. Next to software security, physical security also plays a role. An attacker can try to access the machine, install malware via the console and extract data. Hardware faults or infrastructure failures impact integrity and availability. This thesis focuses mainly on risks that particularly affect hidden services. The sensitive nature of hidden services increases the threat level.

The main objective is to keep the server location hidden. Secondly, when attackers discover a server, operators should not be directly affiliated with servers. Only an outline is given for the second part because it involves endless possibilities that potentially can be illegal. Trying to conceal post addresses is an example that falls under this category. The protection measures emphasize server location obfuscation.

Figure 4.1 in chapter 4 shows Tor's vulnerabilities in academic literature. The Tor project continuously develops the Tor software and protocol. The Tor community, as well as attackers, react quickly to publications. Many concrete approaches mentioned in papers have already been thwarted, while the general ideas provide starting points for discovering new attack vectors. Other methods, such as amplified DoS attacks with long circuits, are still highly relevant.

Operational security focuses on humans instead of technical components or software. Therefore, all approaches are still highly relevant. The four coarse categories capture various methods in this area: Follow the money, linking information, physical evidence, and social engineering. Technical aspects require a manual examination of the attack vector to compare it with the current Tor version. All categories shown in figure 5.1 remain relevant. Only specific methods became obsolete, especially those utilizing previous implementation errors. Also, the hidden service directory protocol got a major change in version 3 deprecating the predecessor in October 2021. [111] Chapter 7 examines concrete strategies to reduce or mitigate the classified vulnerabilities from chapters 4 and 5. Before, the following three sections examine security add-ons for hidden services.

6.3 Vanguards

Vanguards [112] is an add-on for hidden services and consists of three components: Vanguards, Rendguard and Bandguards. The Vanguards main subsystem uses the Tor control port to communicate with the local Tor process. The idea is to fix the middle and exit relay to subsets of possible onion routers instead of rotating randomly through all possibilities. Limiting the number of available middle and exit relays slows down guard discovery attacks. Without any security measures, attackers can initiate many connections over a malicious rendezvous point until the hidden service chooses a malicious middle relay, resulting in the guard discovery. Subsequently, the

attacker either compromises the guards or disrupts the availability.

The exit relay subset contains eight onion routers, the middle relay four and the entry guards remain two. Automatically rotating through the subsets requires attackers to start Sybil and compromisation attacks. The outer layer changes after one up to 48 hours with an average value of 31.5 hours. Effectively, compromising these relays is not valuable for an attacker because they change too fast. Instead, attackers have to launch a Sybil attack. Middle relays rotate after one day up to 45 days with an average value of 29.5 days. Rotation durations follow the idea to require a Sybil attack for exit relays and a compromisation attack on the guard and middle relay. In some situations, clients and hidden services use an additional fourth relay. Hidden services use the fourth relay when connecting to a rendezvous point, making attacks more difficult. Clients choose an onion router as a rendezvous point. By default, attackers can discover the exit relay instantly. If the hidden service uses Vanguards, they can only observe a relay that differs with each new connection.

Vanguards eases circuit requirements. By default, Tor cannot establish paths if multiple relays are from the same /16 subnet or relay family, a tag provided voluntarily by the administrators. Depending on the HSDir, introduction, or rendezvous point, the default configuration forbids the circuit's construction or the entry guard changes. In the Tor circuit with Vanguards enabled, entry guards can also appear as the HSDir, exit relay, introduction, or rendezvous point. Otherwise, attackers can enumerate through different rendezvous points to observe which onion routers are not chosen as a hop before or to force connections to the second entry guard.

Software programs simulate the deanonymization success of customized attackers. The recommended parameters are two entry guards, three middle, and eight exit relays. They are based on an attacker with a 5 % probability of a client choosing a controlled onion router combined with an attacker that compromises an onion router with a 50 % success rate after 2 to 14 days. [113] It is a decent estimation, but

the linearly increasing success probability for compromising onion routers and the undifferentiated probability of placing a Sybil for the guard, middle, and exit relays are unrealistic. Subsection 7.5.6 will deal with this issue.

Rendguard counts how often clients use various rendezvous points. The variation should be high for legitimate requests, while malicious clients can choose colluding onion routers as a rendezvous point. A prime example is the SignalCookie [51] attack aiming at discovering the entry guards explained in section 4.7. Rendguards determines the overuse of onion routers acting as a rendezvous point compared to their consensus weight because not all onion routers have the same selection probability. The authors recommend setting the value to 5. It blocks rendezvous points used five times more than expected. Onion routers not contained in the current consensus document of the hidden service receive an adjustable value of 1 % consensus weight. It allows unknown ones to act as a rendezvous point. The reason is that the local consensus might be outdated and natural churn appeared. Attackers can abuse this system to artificially initiate numerous connections over popular onion routers as a rendezvous point that effectively impacts legitimate users if the hidden services block their requested rendezvous point. In these cases throwing warnings might be more reasonable than entirely blocking connections.

Bandguards monitors the Tor circuits to detect side-channel attacks. The attack from Ling et al. [72], see section 4.12, aims at recognizing that a colluding onion router is involved in a Tor circuit. It works by choosing a malicious rendezvous point and injecting cells to signal the relay if they are using the same Tor circuit. While an update addressed this concrete problem, attackers can still access packet volume and timing information for traffic correlations. Active attackers can insert watermarks as described in section 4.7. Unexpected cells will be dropped in the circuit but might leak information if attackers intentionally insert them. If the hidden service detects a dropped cell, it sends the *DESTROY* cell to the entry guard to immediately close

the circuit and emits a warning message for the hidden service operator. A malicious entry guard might ignore the *DESTROY* cell and continue delivering packets. [114]

Malicious onion routers can append additional data to hidden service descriptor submissions and the corresponding responses that parsers potentially ignore. A non-standard option enables hidden services to close circuits to introduction points after a fixed traffic volume. It helps to mitigate DoS and traffic confirmation attacks preventing potential spikes in traffic volume. As it impacts legitimate users, operators must individually adjust the option to become active. Limiting the total traffic volume for a Tor circuit is also possible to prevent detectable traffic patterns. With larger volumes, the available packet number and timing increase to significantly improve traffic correlation attacks. This option also needs to be actively activated to come into effect. Using circuits too long negatively affects privacy as the old TLS connection stays open while new Tor circuits will use a new TLS connection after a while. Missing traffic multiplexing makes traffic correlations easier and leaks information about the entry guard via uptime correlations. It defaults to 24 hours.

Generally, uptime is an attack vector. Attackers can passively monitor the availability of a hidden service and try to correlate other parallel events to infer information about the location. An active attacker can even disrupt network parts to determine if the availability changes. Bandguards cannot prevent uptime correlation but emits a warning if the hidden service cannot establish new circuits or the entry guard connection fails.

6.4 Onionbalance

Availability is crucial for hidden services. Attackers could try to suppress inconvenient information by launching DoS attacks. The default setup does not provide any load balancing, and one machine answers all client requests. Congesting it with numerous circuits is comparatively easy with medium-sized resources. Therefore, hidden

services require load-balancing mechanisms. Currently, Onionbalance [115] offers load balancing for hidden services with a round-robin approach.

The central idea is to use a front-end server and multiple back-end servers called instances. Each instance runs as an independent hidden service with a unique address. The front end regularly fetches the descriptor of all instances and assembles a superdescriptor. It contains introduction points from multiple hidden services. Clients only know the front-end address and fetch the corresponding descriptor. It includes a list of introduction points and the necessary information to contact the corresponding instance. Operators need to configure these to accept requests destined for the front end.

While everyone can access the Onionbalance repository with source code, there is a conspicuous lack of documentation. The repository description provides only a basic overview and, for example, does not mention the maximum instance number. A manual source code analysis shows that Onionbalance allows up to 8 instances. Responsible is the *get_num_instances* function in the *config_generator.py* file. The parameter file *params.py* defines that the superdescriptor contains at least two introduction points for each instance (*N_INTROS_PER_INSTANCE = 2*). The front end uploads the superdescriptor with the parameters for two replicas (*HSDIR_N_REPLICAS = 2*) and spreads it to the following four HSDirs (*HSDIR_SPREAD_STORE = 4*), see section 3.5.2 for more details.

6.5 Bridges and Pluggable Transports

Bridges are onion routers that the central consensus document does not include. They provide access to the Tor network for users in censored regions. Users request them manually to prevent censors from simply iterating over all bridges and their IP addresses. Writing an email or solving a captcha is sufficient, and users will obtain information to connect to a bridge. Analogous to the authority nodes, bridge

authority nodes maintain a list of all bridges. Instead of broadcasting it publicly, users have to request information with an identifier. [116]

Pluggable transports obfuscate traffic between clients and their bridges. The goal is to prevent adversaries from detecting Tor traffic. By default, the TLS handshake indicates Tor traffic with the chosen ciphers. Tor offers an interface to flexibly switch between different protocols. One approach is to randomize traffic to deter blacklisting classifiers. ScrambleSuit [117] uses a pseudo-random payload to make traffic indistinguishable from random noise. Additionally, it obfuscates traffic flow patterns like the packet length distribution and uses a preshared secret. The latter prevents censors from scanning through suspected IP addresses and checking if they respond according to a bridge protocol. Clients only receive the expected answer if they provide the secret. Protozoa [118] relies on a different approach: adding covert data into a WebRTC stream while users engage in video calls. It mimics the underlying protocol and embeds data into the video and audio feed. Rebound [119] implements decoy routing. Clients in censored regions connect to uncensored, cooperating hosts and include a signal in the connection request. The host recognizes the signal and acts as a proxy for a censored service. To prevent time delays, Rebound requires users to constantly send requests that the host answers with answers to previous requests.

Format-Transforming encryption uses characteristics from known protocols to imitate them. Simple pattern-matching algorithms can be confused by simply wrapping traffic into HTTP requests. Simply inserting strings from known protocols in between also irritates basic algorithms. [120] Flash proxies are short-term proxies. Volunteers host scripts on a website so that every browser visit will start a new proxy. Simultaneously, the host distributes the information to censored users about temporary proxies via a rendezvous protocol. Following this, they can use short-term proxies to access censored content. Similar to Rebound, a popular website can react

to specially crafted strings in HTTP requests to deliver secret messages. A prime example is an encrypted IP address returned as a session cookie. An adversary cannot distinguish it from random data. [121] According to the Tor metrics, obfs4 is the most common pluggable transport for connected bridge users. [122] obfs4 disguises the traffic to prevent adversaries from identifying the obfs4 protocol. It is based on ScrambleSuit and has two phases to exchange keys and encrypted traffic. The key exchange uses Curve25519 keys and works similarly to the ntor handshake, see section 3.3. Clients must know the Node ID and public key to initiate communication. Padding and time obfuscation contribute to resembling random noise. [123]

7 Hardening of Hidden Services

7.1 Operator Security Practices

Several information pieces combined can expose operators. Therefore, every operator should implement strict operational security measures. It is imperative to avoid links to the real identity. Moreover, an additional computer helps to separate the identities instead of mixing files with work or leisure activities. Operators should generate a new identity for each hidden service. Among other things, the identity includes passwords, keys, cryptocurrency wallets, and usernames. It decreases the chances of accidentally leaking information. Creating a cover story can help to confuse attackers. Operators should intentionally use the cover story and document any personal information they disclose through messages, content, or other text. Otherwise, they might be inconsistent and remain consistent only if the information is accurate.

Operators should encrypt their devices as random controls can occur, or they could lose devices. Audited software with secure algorithms is well suited for this. In addition to full disk encryption, operators should encrypt sensitive files with separate passwords. In the case of targeted attacks, operators might lose access to the device while it is active. Also, simply powering the device off is insufficient because the memory retains data for a while. Experts can read the content and even prolong the time window by freezing the memory chips. [124] By solely decrypting extra

sensitive files in safe physical environments, operators can minimize their risk. It is a trade-off between usability and security. Attacks become more complicated with higher degrees of compartmentalization, but operators might make mistakes.

Passwords remain dominant for authentication, and operators should choose strong passwords. Attackers can use dictionaries adopted to cultural backgrounds and apply simple substitutions. Markov models [125], and probabilistic context-free grammars [126] exploit common password structures and allow attackers to guess passwords efficiently. A strong password contains unpredictable patterns and a sufficient length. The entropy should be similar to cryptographic standards, typically 120 bits. The zxcvbn library [127] can help to assess password strength, but caution is needed. The password should take the attacker's perspective into account. Password reuse is dangerous, and a cracked password can help the attacker make assumptions about the other passwords used.

Password managers help operators to sort credentials and store encrypted notes. Different password databases reflect compartmentalization. Offline, open-source, and well-known software reduces the risk of implementation errors and information leakage. Secure encryption and hashing algorithms fend off attackers. Passwords require designated hashing algorithms that intentionally slow down legitimate users and attackers. The penalty is not equal but affects attackers with specialized hardware significantly more. Memory-hard functions artificially utilize a lot of memory that is equally expensive for attackers and legitimate users. Argon2 [128] is a standardized memory-hard hashing function designed for passwords. Operators should choose Argon2 in combination with suitable parameter values adapted to their computational resources.

7.2 Hidden Service Security Practices

The first approach for attackers is information provided directly by hidden services. Even open ports with limited functionality can reveal sensitive details in the header. SMTP services might respond with a header containing the time zone, hostname, and software version. Operators should minimize connection information. Another alternative is to wrap vulnerable protocols by utilizing protocols that stay silent when clients do not know preshared information. Protocols like HTTP or SMTP transmit documents that can include personal identifiers. Hidden services should avoid all references to the operators, software configuration, or server location.

Noticeable linguistic features in documents can form a unique profile that leads to concrete suspects. Operators should alter used expressions but also vary word and sentence lengths. Forensic approaches reliably detect authors but struggle with obfuscated language. [129] Generally, operators should also keep a small footprint with their real identity on the internet to minimize available material for comparisons. Frequently emptying log files and deleting unused files delivers attackers less material.

Social engineering is a widespread issue that also affects hidden services. First, they should verify the identities of users. Otherwise, attackers can easily impersonate them. Second, operators should distrust anonymous people and consider that attackers might hijack established accounts. Different circumstances require different levels of trust, but operators should keep it to a minimum.

Especially if a hidden service is load-balanced over multiple servers with Onionbalance, attackers can identify and compromise one out of all hosting instances. Stored billing addresses, payment information, or other personal details expose the hidden service operator. Also, archived support tickets and connection logs pose a threat. Depending on the anticipated adversary, the server location should vary. Different jurisdictions protect against various legal requests and require stricter data protection. Whistleblowers from developing countries might find suitable server

locations in the U.S., while European whistleblowers can seek refuge in countries with different political goals. A physical server is preferable to shared hosting, as the latter makes it easier for attackers to eavesdrop. For example, a VPS usually runs inside a virtual machine, and the host system has full access to all resources, while the VPS cannot even reliably detect that the host accessed files. Full disk encryption protects server data if someone shuts it down to copy files. Sensitive files like passwords should exclusively reside in the memory. Operators should prefer a hosting provider that does not offer direct access to memory areas or indicates this to the customer. The hosting provider should manage the server professionally and ensure physical security.

Operators should always provide data warily, choose hosting services with anonymous payment methods and use Tor for buying and administrating the servers. Operators should only connect to the servers using Tor to protect themselves in case attackers have already compromised the servers. Trimming or pseudonymizing log files minimizes data that can prove ownership in case of a seizure or threatens users. Alternatively, hidden services can transfer log files to separate hidden services immediately to increase the burden for attackers to acquire them.

Donations or other payment forms enable attackers to investigate money flows. At least, hidden services should rotate payment addresses. Providing one address per user and transaction is preferable. Otherwise, attackers can determine the overall amount quickly and even examine the users who donated. [9] Section 7.6 investigates the money aspect thoroughly.

7.3 Hidden Service Software Configuration

Tor provides strong anonymity for hidden services. Still, software configurations can decrease anonymity by not using Tor properly, leaking information, or creating vulnerabilities. Therefore, next to choosing and updating software carefully, operators

should manually configure deployed software to protect sensitive data.

Foremost, hidden services should only allow connections from Tor and not directly from the internet. Else, attackers can scan IP address ranges and discover the hidden service. Especially in the case of a website or other delivered documents, the attacker can easily relate it to the scanned IP address. If the Tor proxy redirects all requests locally to the software, then operators should ensure that programs do not assign supplementary trust due to the local origin. Besides software configurations allowing only access over Tor, a strict firewall should prevent unexpected connections. Section 7.5 details a suitable architecture.

Operators should remove or relocate preconfigured administrative interfaces, default ports, and status pages. Error messages can leak information, and operators should only activate them in separate development environments. By deploying a separate content filter software or firewall, it can filter out suspicious requests and answers. Exploits in Tor could potentially deanonymize circuits, overload resources, or execute commands on the server. The content filter can use whitelisting techniques to minimize the attack surface for exploits, but it could also negatively affect usability if the filter is too strict.

Programmers exhibit distinct coding and variable naming styles. Attackers can infer the author by evaluating the page source code or examining configuration files. Machine learning algorithms automatically identify code authors by comparing fragments to public repositories. Operators should alter the code. It includes, for example, indentations, variable and function names, type declarations, and control structures. Type declaration might show preferences towards certain data types, and control structures also indicate preferences. Simple lexical changes and altering control structures render current attribution algorithms useless. These changes include turning a for loop into a while loop or calling different APIs. [130]

Hidden services should not respond to connection probing and remove or alter

timestamps to limit side-channel attacks. If possible, TCP timestamps should be deactivated. Additional measures are necessary depending on the specific application, for example, Bitcoin nodes.

Databases are particularly at risk from SQL injection attacks. Every software must check user input extensively and notably special characters. For SQL, parametrized SQL queries fix the semantics and interpret special characters as strings. [131] Likewise, the software should check all text inputs for scripting attempts and scan uploaded files for file types. An advanced approach is to add honey passwords to databases. If an attacker uses these passwords, they trigger an alarm because it indicates that attackers accessed internal data. [132]

7.4 Tor Configuration

Tor has vulnerabilities that attackers can cleverly use to deanonymize hidden services. Operators should avoid misconfigurations, use additional security plugins and harden the default configuration. Some misconfigurations allow attackers to decloak hidden services quickly, while many vulnerabilities require time and coincidences. Due to Tor's decentralized nature and code complexity, attackers will discover new deficiencies. A secure configuration can reduce the chances and increase the required resources to launch a successful attack.

It is possible to host multiple hidden services on one server. But as all of them share the same entry guard rotation, latency, and downtime, attackers can link them. Also, in case of successful deanonymization or compromise, all files are simultaneously exposed. The same applies to relays and hidden services on the same host. If the service targets a limited number of clients and sharing out-of-band data is possible, the authorization feature allows control of connecting clients. Attackers would first need to acquire the preshared data to attack the service.

Operators should use the Vanguards add-on. Services with high-security require-

ments can manually specify middle and exit relays in Vanguard. Onion routers with a long history without providing connection data and known administrators are ideal candidates. The selection should blend in and not use exotic onion routers. Otherwise, attackers can target these, and they might react differently from established, experienced non-profit organizations providing onion routers. Optimally, the selected routes should lead through different ASs and IXPs.

Vanguards also mentions the scenario for operators to host a bridge that their hidden services use. [112] It enables pluggable transports that impede local eavesdropping. The hosting company, local ISP, or other local network devices cannot easily recognize that the server uses Tor. Moreover, website fingerprinting is hampered and protects the server from this attack category. Still, hidden services should not include too large files and serve dynamic content. Obfs4 is currently the most popular pluggable transport. [122] Local eavesdroppers can still confirm with induced traffic spikes or connection resets that a machine is running a specific hidden service. Just by rejecting traffic above a threshold, induced traffic is not a problem. Intended connection resets are harder to mitigate as they directly affect entry guard selection and uptime. Entry guards and middle relays should not rotate too quickly, and operators with high-security requirements and expertise should preferably select the onion routers for the middle and exit relay subsets manually. It still affects the uptime, and attackers can correlate this. Operators should set warnings in case of disconnects and frequently evaluate log files for possible attacks.

Hosting a bridge can increase security but requires careful configurations. Other Tor clients should also use it to blend into the overall traffic. Otherwise, attackers can directly infer that bridge and hidden service operators are the same entities. Sufficient flags and traffic bandwidth attract enough Tor clients. Also, the location and other properties should not be exotic. Due to the high number of relays in Germany and France, these are attractive locations. Operators can distance themselves from the

bridges, which, in turn, raises suspicions, and attackers can leverage this angle. On the contrary, operators can also openly present them and use plausible deniability, even though attackers can still use illegal observations, torture, and other methods to punish suspected operators without evidence.

Rendguards can detect rendezvous point overuse and hinder quick discoveries by malicious entry guards. Likewise essential is Bandguards to limit circuit bandwidths. It mitigates induced high traffic volume on single circuits. Introduction points should rotate, and circuits should expire after a limited time. Even though the default values are reasonable, operators should customize them according to live usage. Dropped or unexpected cells, closed circuits, and general downtime hint at side-channel attacks and should alert operators.

7.5 Security Architecture for Hidden Services

7.5.1 Improving Availability

Large traffic volumes and targeted DoS attacks require operators to perform additional efforts and implement an extensive infrastructure. Onionbalance can manage up to 8 instances. It also protects the instances when they connect to malicious HSDirs. If the circuit to the HSDir contains a colluding middle relay, the attacker can correlate connection data to discover the entry guard of a hidden service. If the entry guard is under the attacker's control, then the hidden service's IP address is exposed. But as the front-end server assembles a superdescriptor for all instances, HSDirs cannot draw conclusions to instances as long as operators do not leak their onion addresses. Each instance has its unique onion address, and the front-end server downloads them over Tor from the HSDirs. Afterward, it publishes the superdescriptor for all instances under a separate onion address. Therefore, malicious HSDirs primarily affect the front-end server running Onionbalance. Operators should keep all onion addresses

secret and only share the onion address from the front-end server. Operators should use multiple synchronized servers for the front end to handle server failures.

The front-end servers should regularly check if the descriptors are still available. If not, then the HSDirs block the descriptor selectively. Even though it is unlikely due to the rotating number of responsible HSDirs, operators can use multiple front-end servers with different onion addresses that will have other responsible HSDirs. Subsequently, users should know about the various onion addresses for the hidden service and can manually connect to addresses. The current implementation does not enable instances to accept connections from front-end server descriptors with different onion addresses. Therefore, operators must split the instances into distinct groups and assign the responsible front-end onion address separately.

The current Onionbalance implementation is not well documented and lacks some features. Besides using multiple onion addresses for the front end, the Tor specifications allow the integration of more instances. Each descriptor can contain up to 20 introduction points: consequently, it can include up to 20 instances with one introduction point for each. In this case, introduction points could become targets of DoS attacks. Moreover, hidden services can generate multiple valid descriptors with different introduction points. Instead of uploading one descriptor as described in section 6.4, hidden services can upload distinct descriptors for each responsible HSDir. Currently, clients use the value 2 for *hsdir_n_replicas* and 3 for *hsdir_spread_fetch*. Therefore, they choose one out of six HSDirs, and hidden services can exploit this to distribute six distinct descriptors. It allows the publication of up to 120 introduction points. With each instance occupying two, clients can reach 60 instances. Operators should adjust the number of introduction points per instance depending on live traffic data and must consider that the descriptor size limitation of 50 000 bytes might limit the numbers.

Hidden services can use parameters like *DOS_INTRODUCE2_RATE_PER_SEC*

and *DOS_INTRODUCE2_BURST_PER_SEC* when they connect to their introduction points to mitigate DoS attacks. Both parameters instruct the introduction points to limit the number of relayed *INTRODUCE2* cells and help to improve availability. *HiddenServiceMaxStreams* limits the number of simultaneous connection streams to a rendezvous point. Hidden services can also try to detect patterns in malicious clients like user agents, but the Tor browser does not allow them to distinguish between well-adapted bots and real users. Operators can deploy separate servers responsible for captchas before forwarding requests to the main servers. Section 7.5.4 details this concept.

7.5.2 Transferring and Storing Sensitive Files

Operators can transfer and distribute sensitive data between servers to minimize traces and improve anonymity. Figure 7.1 shows an architecture with several advantages. It includes three different control servers that separate files from hidden services: management, log, and key servers.

Section 7.5.1 mentions load balancing with front-end servers publishing superdescriptors and instances for client connection processing. All instances choose their entry guards independently by default. Hence, the probability of malicious relays in circuits increases which accelerates deanonymization attacks. The management server centrally distributes configuration files containing entry guards, relays, and other parameters. It connects to the attached servers over Tor with their distinct onion address and uses simple copy commands over SSH. Preshared keys and strong ciphers ensure security. Attackers need to deanonymize instances or front-end servers before discovering a connection to the management server. Management servers should use different entry guards and relays to force attackers to launch separate attacks on the new relays. If the front-end servers use different entry guards and relays than the instances, attackers do not automatically discover the circuit relays

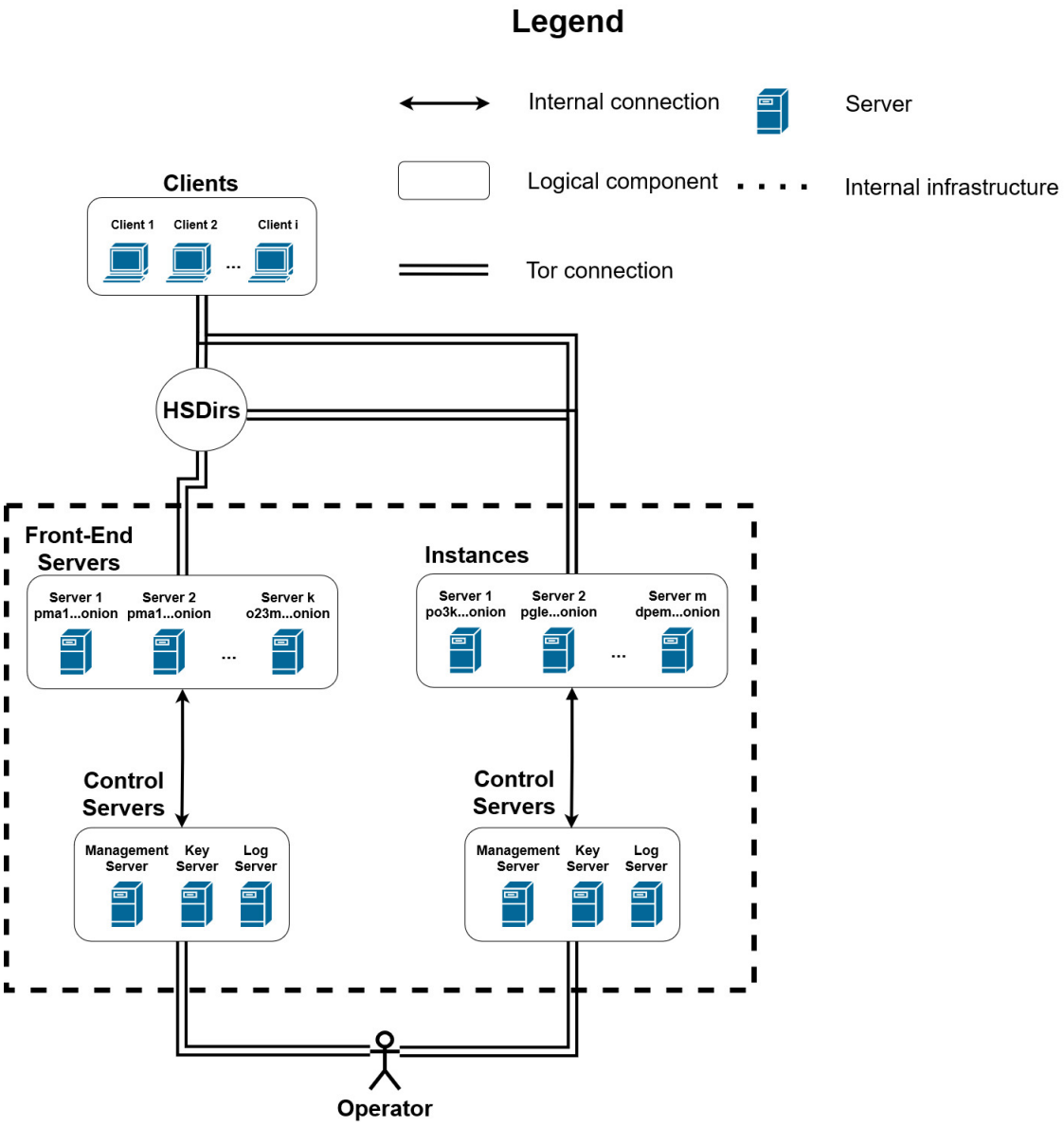

Figure 7.1: Security architecture for hidden services

for them in case they decloak front-end servers during descriptor uploads.

It is possible to use separate management servers for the front-end servers and the instances to increase compartmentalization. If attackers managed to breach a front-end server and then the connected management servers, the instances and their management servers remain secure. Otherwise, attackers could use the compromised

management servers from the front end to induce poisoned configurations into instances. Still, this solution increases the complexity, and attackers need to break Tor two times for this threat scenario instead of directly targeting the instances.

Log files include sensitive information, and attackers can place hints that prove ownership analogous to section 4.8. Generally, log files should only protocol relevant events instead of collecting everything. Operators should determine their use case and use plugins to reduce the accuracy of log information. Exact timestamps or IP addresses are seldom relevant but equip attackers with new angles after a compromise. A central logging server offers a better overview for the operator and shifts sensitive log files away to a secured location. Instances periodically connect over Tor via a stored onion address to the logging server, send their log files, and cleanse present ones. Before attacking the logging server, attackers initially need to compromise attached servers to retrieve the log server's onion address. If attackers get in control of the log server, they only gather the onion addresses of all connected servers and the log files. To not endanger the remaining infrastructure, log files should not include sensitive information about it, or at least all attached servers should pseudonymize sensitive information before sending it. Operators further minimize risks by encrypting all log files with a rotating key. Operators can fetch encrypted log files, delete them from the logging server and rotate the encryption key. Afterward, they can evaluate them locally. It minimizes files that attackers can access by comprising the logging server.

Additional key servers safeguard the master identity and blinded signing keys. Key servers deliver prepared keys, so hidden services do not need to store these but rely on temporary ones, as described in section 3.5.1. If the latter ones become known, attackers can only take over the identity for a limited time. With those keys, they can conduct man-in-the-middle attacks, DoS, or impersonate the target entirely. If the master identity keys get exposed or lost, only generating new keys with a different onion address helps. It erodes the reputation, and attackers can trick users

until someone notifies them about the breach.

Key servers can distribute descriptor signing keys to the front-end servers and instances for each time period. Immediately when operators detect a breach, they should stop key distribution to affected parts and set up new servers to continue operation. Concurrently, they can investigate the cause and consequences. Operators should consider all communication to affected onion addresses as potentially contaminated and avoid them for the validity period. Management servers can modify configuration files to exclude this onion address if the software is programmed accordingly.

Key servers are an attractive target. To spread the risk, operators can deploy different key servers analogous to the management servers. Two key servers could separately provide temporary keys for the front-end servers and instances. In case of a successful compromise, attackers can only impersonate front-end servers or instances. Because the descriptor signing keys derive from the blinded keys and a randomly generated value in the consensus: either the operators have to regenerate them manually for each time interval, or key servers store the blinded keys. The former approach makes the key servers redundant as the operator could directly distribute them to the front-end servers and instances.

The architecture enhances security enormously if used correctly. In practice, this leads to many potential errors for operators who make mistakes or misconfigure programs while managing diverse servers. Higher complexity can compensate for systematic weaknesses but requires impeccable calibration and operation. Depending on the operators' security requirements and budget, the control servers can also run on one server that handles all three control functionalities. Front-end servers and instances can also use the same control servers. However, the configuration is more complex if the functionalities are not clearly separated.

7.5.3 Mitigating Side-Channel Attacks

Operating systems and programs usually depend on various underlying libraries, each containing even more lines of code. Consequently, operators must anticipate software vulnerabilities and deploy multiple security layers to reduce the likelihood of attackers exploiting stacked vulnerabilities. Bypassing the Tor connection is one of the primary attack vectors. Attackers can utilize numerous possibilities to trick hidden services into leaking information. DNS requests, injected scripts, or even zero-day exploits endanger them.

Apart from hardening configurations and using up-to-date software versions, operators should use hardened operating systems. With fewer libraries installed, the attack surface generally decreases. Whonix [133] is a free and open-source operating system and offers several advantages out of the box. The central idea is to use two virtual machines: a workstation and a gateway. The workstation runs all programs while only communicating with the gateway. In turn, the gateway tunnels all traffic from the workstation over Tor and hides Tor from the workstation. It allows configuring the hidden service software and Tor proxy separately, which improves the overview. To launch side-channel attacks, attackers must initially deceive the workstation and, afterward, the gateway. Because the latter only focuses on tunneling traffic through Tor, the attack surface is significantly lower. Whonix requires a hypervisor that can be a complete operating system like Windows or, in the case of KVM, a lightweight hypervisor. Whonix is based on a modified Debian operating system with hardened security features. Using a different host system can increase the security against malware that would need to contaminate the Debian and host system.

Qubes OS [134] is a privacy-friendly operating system that integrates well with Whonix. Users operate in multiple environments isolated by the Xen hypervisor preventing single compromised areas from contaminating the entire system. The

combination of Whonix with Qubes OS as the host deactivates TCP and ICMP timestamps and provides a strict security configuration by default. Qubes OS supports full disk encryption and allows backups of single environments. [135] Operators can easily migrate the Whonix environment to different physical servers to leave as few traces as possible for attackers. Generally, rotating to new physical servers fends off attackers that decloaked or compromised servers. Also, if the guard configuration changes simultaneously, attackers lose all their enumeration information about the Tor configuration. Therefore, operators should regularly rotate their servers and Tor guard configurations.

All front-end servers and instances in the architecture depicted in figure 7.1 should use Qubes OS as the operating system and run Whonix. The gateway virtual machine needs a program that interacts with the management and key server and tunnels the communication over Tor. The workstation virtual machine requires no extensive configuration to communicate with the log server as the gateway tunnels the traffic. If the log server contacts the workstation and not the other way, then the workstation needs to transfer the log files to the gateway that, in turn, responds to the log server.

7.5.4 Extensions

The architecture from the previous sections uses several instances that constitute the entire web application. Synchronization problems might occur between them in the case of shared data. Additionally, when operators divide the architecture further, they are more flexible and simplify configuration because each server only takes care of a part. Figure 7.2 shows the improved architecture for hidden services.

EndGame [136] is an open-source software composition protecting hidden services from DoS attacks by deploying captcha servers. Silk road used a similar captcha system as subsection 5.5.1 outlines. The idea is to mitigate DoS attacks by requiring

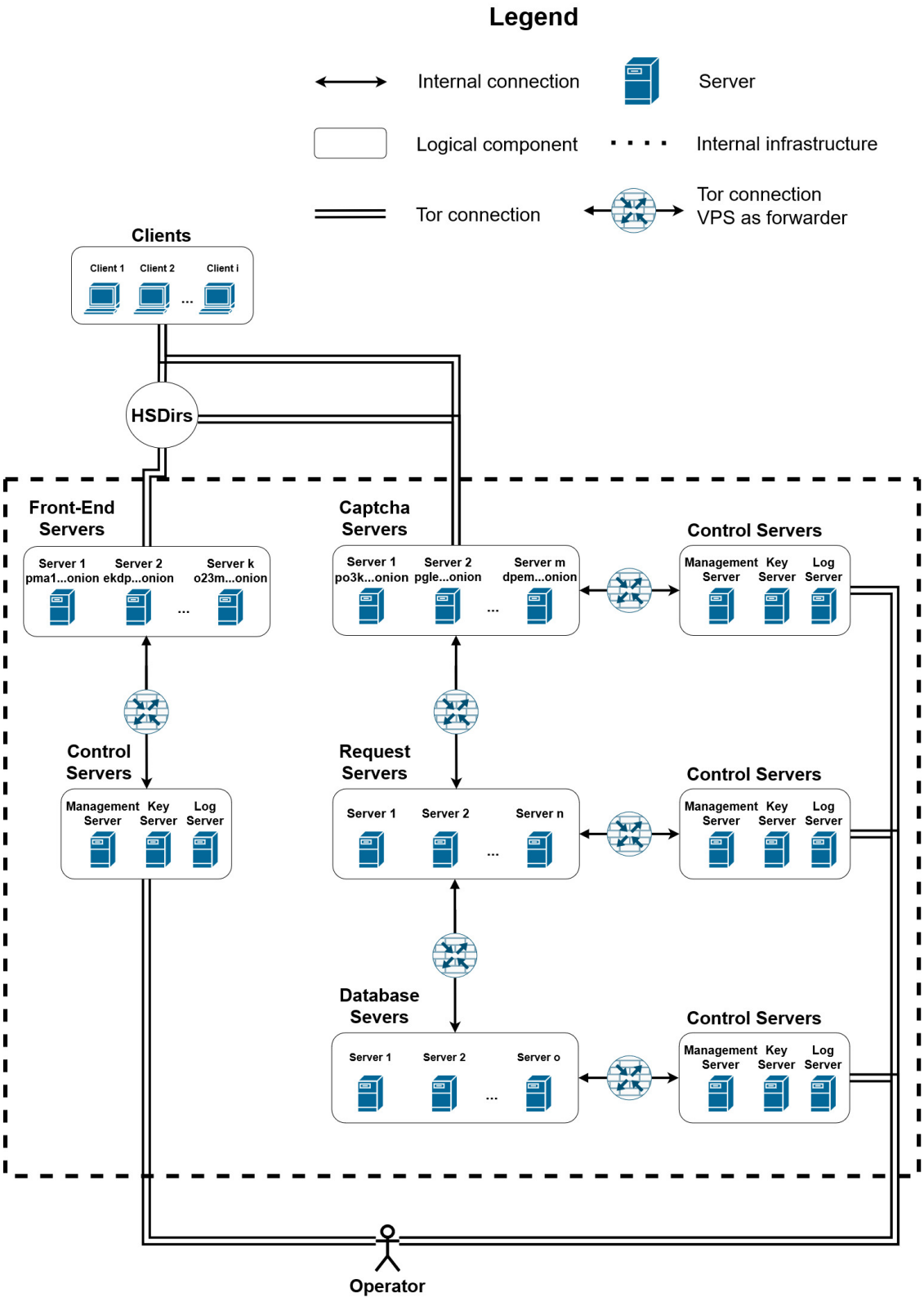

Figure 7.2: Extended security architecture for hidden services

human interaction. Moreover, captcha servers allow the separation of DoS protection from the remaining infrastructure and assess all client requests. The new architecture splits the instances into the captcha and request servers. The former ones receive client requests and redirect legitimate requests to the latter. After clients solve the captcha, the captcha server places a cookie to remember clients and directly forwards session traffic. Bots cannot circumvent the captcha servers, so they can only try to exhaust their resources. Simultaneously, operators can deploy more captcha servers to handle increased traffic load while the remaining infrastructure remains unaffected. In contrast, the previous architecture in figure 7.1 only allows rescaling instances with complete functionality. Besides the increased flexibility, the separation enables filtering all client requests to the request servers. Captcha servers can discard suspicious characters or unexpected API calls before they reach the request servers.

Many applications need to store and read synchronized files. If multiple request servers operate, they must use a shared database. Therefore, operators should further split the functionality into several servers. For example, one server can be responsible for processing client requests or operating as a database server. Various other tasks are separable into additional servers, like CPU-intensive calculations or payment handling. If a single server performs a function, a secondary one should mirror all files. If the primary one is not reachable, the secondary continues the functionality. Other servers must know the onion address from the primary and secondary server to redirect requests in case of unavailability. When multiple servers perform a function collaboratively, they must synchronize data. In the case of databases, a cluster system that automatically distributes data among several nodes is favorable. For simplicity, the architecture in figure 7.2 assumes that there is only one database cluster, but operators can add additional functions analogously.

Again, all connections between the servers should be encrypted and anonymized. Tor can perform the latter very well but increases the overall delay for requests. First,

user requests to the captcha server have an inevitable delay, then the connection between the captcha and the request server introduces additional delay. If the request server connects to a database cluster, then the overall response time includes connection delays from three individual Tor circuits. While this offers suitable protection, it might deteriorate usability significantly. Operators can use different anonymization techniques between servers to balance security, performance, and usability.

Request and other servers deep inside the architecture require attackers to compromise multiple other servers before attackers can reach them. It only holds if the attacker cannot exploit vulnerabilities to compromise them immediately. Operators need to assess the application security and make decisions for the control servers. Using different control servers for the front end and captcha servers is reasonable, while separate control servers for the database cluster might cause too much overhead compared to the security gain.

Figure 7.2 mentions that operators can deploy a VPS between the server connections. The VPS can analyze the traffic to detect and block conspicuous activity. Furthermore, the VPS can increase the security of the servers in the background because attackers need to compromise the VPS before reaching them. Operators can secure both connections with Tor, but it increases the delay. Additional delay is not problematic for non-time-critical operations like connections to the control servers. Contradictorily, users will not tolerate prolonged response times. The VPS could send alarms to the management server but should use different guards than the attached servers. Otherwise, the anonymity remains unchanged as both sides share the same guards. Operators should carefully consider for which connections the VPS is beneficial.

7.5.5 Security Enhancements

Operators should expect that attackers will expose hidden services over time, even if all security measures have been taken. Dividing the functionalities into different servers forces the attacker to conduct multiple attacks in a row successfully. A critical weakness in Tor could enable attackers to compromise multiple circuits fast, allowing them to penetrate the entire infrastructure as the servers communicate. Consequently, operators can use other anonymization techniques internally to diversify the software. For example, I2P [137] is a peer-to-peer anonymization network that offers a similar hidden service function. The principle differs from Tor and should be immune to Tor-specific zero-day exploits. Regardless, all servers should encrypt the traffic with preshared keys or certificates that operators can distribute with the management server or install manually for all servers. Even if an eavesdropper listens in, the encryption protects the content. SSH or similar protocols with low overhead fit for this purpose.

Rotating servers regularly frustrates attackers further. Long-planned Sybil attacks and continuing investigations will decloak server locations. The layering approach shields the inner infrastructure even from compromised border components like the captcha servers in the proposed architecture from subsection 7.5.4. Switching to new servers cleans potentially contaminated parts, altogether with the attacker's foothold in the architecture. The servers should change simultaneously with the entry guards and relays. Otherwise, a successful Sybil attack on previous servers deanonymizes the new ones. Components closer to the user, like captcha servers, should rotate faster than inner components like the database cluster. It is essential to variate this process, or else sophisticated attackers could monitor where a predefined number of newly set up servers exhibit a comparable amount of Tor traffic. Payment and customer information should also vary over time to avoid observable patterns.

If operators let the infrastructure communicate internally via Tor, they should

use client authorization whenever possible. Even if attackers know the onion address of the inner infrastructure, they cannot build up circuits as they cannot decrypt the necessary information from the descriptor. An attack requires the onion address and preshared keys. Without client authorization, attackers can continue to build up circuits to the inner infrastructure to launch further attacks even after compromised servers rotate. With each server rotation, the client authorization secrets from adjacent logical components should change too. Only then do attackers have to start from scratch again. Either management or key servers could automatically distribute client authorization secrets to all infrastructure servers. Because it can take some time to propagate the new information in the distributed hash table, old secrets should be valid for a few more hours before new ones render them invalid.

Operators should automatize the server setup to prevent mistakes during the process. For example, Ansible [138] is an easy administration tool that uses SSH and python to bring servers into a reproducible state. Operators should cautiously define the setup commands for the different logical components, while servers from the same logical component should have an equivalent configuration. Preshared keys and certificates require special attention, while operators only need to replace IP addresses in the Ansible configuration for the remaining configuration to set up running servers. Before, these must allow SSH connections and have python libraries installed. Automatizing the setup of a hardened system mentioned in section 7.5.3 can prove difficult with complex software configurations. Sensitive information in configuration files, like private keys, should be encrypted with built-in tools, then operators can decrypt them on the fly during the setup execution.

If attackers control a substantial number of HSDirs and know an onion address, they can track access times as outlined in the section 4.4. It leaks potentially sensitive information as operators operate alone on the control servers or VPS. Because only the operator accesses the control servers, attackers cannot know about the addresses

from the infrastructure configuration. The log server is an exception as other servers push their log files, but the log server could contact them instead to avoid storing an onion address in their configuration files. Operators could use a script to rotate onion addresses regularly and then the management servers distribute the new addresses accordingly. Consequentially, this renders information about addresses invalid after the rotation period, and attackers must acquire it again to repeat the tracking attack. As this change requires extensive administrative effort and mitigates a side-channel leak, operators should thoroughly consider the trade-off.

User requests ripple through the infrastructure and induce traffic throughout the infrastructure. If attackers know about the multilayered structure, they can send requests and listen to colluding Tor relays or observed networks for traffic patterns. If a logical component uses servers from different networks, a local eavesdropper only sees a part of the traffic as it should uniformly distribute among all servers. Malicious Tor relays involved in circuits can still observe traffic spikes. Operators can artificially create independent traffic between servers to obfuscate traffic fluctuations. Potentially, management servers can distribute scripts that connect to attached servers at random intervals via Tor to produce additional traffic. Also, servers could contact external hidden services outside of the infrastructure, but it introduces the possibility for further deanonymization attacks if the contacted hidden service cooperates with attackers.

7.5.6 Determining Rotation Times

Operators should regularly rotate the guards and relays for the hidden service components and relocate the servers. Each rotation resets ongoing guard discovery and compromisations attacks. At the same time, they increase the possibility of picking a malicious Sybil relay. If the entry guard is conspiring with rendezvous points chosen by an attacker, protocol features could reliably detect the correlation.

Principally, the approaches from section 6.3 apply. Vanguards picks two entry, four middle, and eight exit relays. As section 6.3 describes, the outer relays change on average after 31.5 hours, while middle relays change after 29.5 days on average. The rationale is that attackers should not actively attack the outer guards but wait until a rotation includes their Sybil. Contrary, middle relays rotate slower to mitigate Sybil attacks and force attackers to attack them.

The Vanguards developer calculated optimal relay numbers and rotation times in simulations. They assumed an attacker with a 5 % chance to place a Sybil into any relay chosen by a target. Moreover, the attacker can successfully attack any relay with a 50 % chance after 2 to 14 days with a linear increase. If the entry guards rotate after 90 to 120 days, then it takes an attacker 146 hours on average to discover an entry guard and 900 hours to deanonymize the server. [113] The attacker model determines the whole setup, and each operator should create an individual threat model. Activists from developing countries with low technical resources will face attackers with a significantly lower success probability. Contrary, whistleblowers in Europe or the United States of America face well-equipped attackers. An illegal expression of opinion or leaking secret documents that do not threaten national security will not attract significant attention. For comparison, section 5.5 describes several notorious criminal enterprises that continued to run for years: the infamous Silk Road for more than 2.5 years, the successor Silk Road 2.0 for more than one year, and Wall Street Market for almost two years. Not every whistleblower or political activist will face extensive attacks.

Without a concrete threat model, operators should use the proposed default parameters from Vanguards. In the case of customized parameters, the process should start with the exit relay rotation time. The value should disincentivize attackers from launching active attacks but rather deploy Sybils. For an attacker with a Sybil probability of 2 % and a hidden service with eight exit relays, the

likelihood of picking at least one Sybil is $1 - (1 - 0\,02)^8$. One Sybil is sufficient to learn about the middle relays. If the attacker has a 100 % probability of compromising a target relay after 1 - 14 days with a linear increase, then operators can calculate the rotation time X in hours after the first day with inequation 7.1:

$$1 - (1 - 0\,02)^8 > \frac{1}{13 \cdot 24} \cdot X \tag{7.1}$$

In this exemplary case, attackers cannot compromise relays during the first 24 hours, so the rotation time includes these. 46 hours after the first day, the compromise probability starts to exceed the Sybil one. Therefore, the exit relays should rotate at the latest after $24 + 46 = 70$ hours to make Sybil attacks more attractive. From there, calculating the cumulative probability of a successful attack targeting the second layer is necessary. It also depends on the likelihood of a successful attack on the exit relays. The compromise probability for the second layer should be higher than the Sybil one to avoid unnecessary rotations increasing the attack surface. The trade-off between Sybil and compromising attacks demarcates the rotation times.

The advanced architecture from subsection 7.5.4 describes that all logical components should rotate their server regularly. Determining rotation times depends on various factors. Attackers first need to pass the captcha servers to learn about the onion addresses of the request servers. If they also rotate regularly but not simultaneously, attackers have a short window to attack them before operators completely replace the compromised servers and alter the client authorization secret. Without the latter, attackers cannot build up circuits to continue the attack. Logical components should rotate with a grace period for the previous servers. Instead of instantly shutting them down, they should continue to run for a few extra hours to ensure a smooth transition to the new ones. Otherwise, outdated descriptors and consensus files might still point to the previous ones and make the server unavailable. Also, ongoing connections should slowly time out to not interrupt users during their

interaction with the hidden service.

Client authorization codes and onion addresses from internal servers could change periodically too. The former should rotate when adjacent components rotate because they allow attackers to continue attacks on an onion address as long as the shared secret remains unchanged. It is also advisable to change the onion addresses simultaneously. Thus, it prevents malicious HSDirs from correlating access times.

7.6 Handling Cryptocurrencies

Operators should use privacy-preserving cryptocurrencies like Monero and trade them in decentral and anonymous exchanges into fiat currency. They often require authentication, and report, together with banks, suspicious money flows to authorities. It can cause government institutions to target specific persons and surveil them. Many expenses, for example, leasing servers, can be paid directly in cryptocurrencies. Since Bitcoin is still the dominant cryptocurrency with a market cap of over 315 billion U.S. Dollars, it is more accessible to users than Monero, with a market cap of just over 2 billion U.S. Dollars. [139] Swapping the Bitcoin into Monero at a decentral exchange or applying mixing protocols first helps to preserve anonymity. Operators should use exchange marketplaces for cashing out cryptocurrencies that do not cooperate with the adversary. For example, whistleblowers should prefer marketplaces with jurisdiction in countries that support them. Otherwise, an online business can serve as a legitimate front to justify profits, or operators can exchange cryptocurrencies locally in foreign countries with other users. Optionally, operators can provide trustworthy entities viewing keys for specified addresses or transaction histories to receive further support and strengthen the reputation for their political engagement.

Trusted mixing servers and exchanges are imperative. Otherwise, operators might expose themselves or lose all transferred funds. Technically, operators should very

carefully protect payment servers because they are a profitable target. Splitting payment operations among multiple servers and various wallets reduces traceability. Still, attackers can conduct payments to observe the transaction flows. If operators support transparent cryptocurrencies, they should not merge incoming donations but directly process them to gain anonymity. Otherwise, operators endanger users by linking all donations. Payment servers should minimize the number of incoming connections to limit deanonymization attacks in Tor. Using a VPS as an additional proxy, as described in section 7.5.4, increases network anonymity and makes the server harder to locate.

Depending on the size of incoming and outgoing transactions, operators should transfer as much as possible to separate inactive addresses. The servers do not access these permanently but only the necessary assets. Operators can manage inactive addresses independently from the servers without haste. In case of a breach, they only lose the assets managed by the servers and can use the offline addresses to restore the hidden service. Sometimes it is enough to provide servers with the private viewing key for billing purposes instead of giving full access. From time to time, the management servers may generate new addresses to replace those currently in use. Initially, operators should fill these with anonymous cryptocurrency units or, at least, mixed ones from inactive resources. The servers should delete the private keys from inactive wallets after transferring them to operators to mitigate the effects of a compromise.

8 Discussion

8.1 Determining Risks for Hidden Services

The systematic academic literature review and analysis of court documents provide an extensive overview of vulnerabilities for hidden services. Both help hidden service providers to assess the risks of using Tor.

Academic literature primarily focuses on vulnerabilities in Tor, and only three out of 46 publications from chapter 4 explicitly deal with operational security. The impact of many vulnerabilities is strenuous to quantify as the authors research them without practical experiments. These are not carried out for ethical reasons limiting the analysis to theoretical ideas, small-scale field tests, or simulations. Side-channel attacks and website fingerprinting pose a real threat but require additional conditions. For example, the former can compare hidden service availability to correlated events to determine an approximate location, while the latter requires a local eavesdropper. Quantifying risks requires a reliable estimate if involved hosting providers, local ISPs, or ASs track and link traffic information. On the other hand, Sybil, protocol-level traffic correlation, and watermarking attacks are an immediate danger that all hidden services operators should extensively consider before launching their projects. However, many publications do not mention how to defend against these attacks effectively, only general ideas to improve a specific vulnerability.

The internet literature review analyzes court documents on relevant cases that

provide insight into how law enforcement works. Due to limited availability, this thesis only examined U.S. court cases. Because U.S. law enforcement cooperates internationally, as the case in section 5.5.3 shows, and has extensive resources, they represent this specific attacker type well. The Justice Department's website mainly provides extraordinary investigations that attract the public interest. How much resources law enforcement can spend on less relevant cases remains unclear, therefore the methods in chapter 5 represent a higher bound. Court documents will not include all investigative approaches from law enforcement to protect them. Otherwise, criminals can specifically avoid them. The Silk Road case from subsection 5.5.1 and especially the recorded log files cast some doubt that law enforcement might have resorted to other technical methods to identify the servers and later brought up an alternative explanation in court. This method enables law enforcement to exceed their authority and still use collected evidence in court as a cover story.

The methods mentioned in court documents deviate significantly from vulnerabilities in academic publications. Practically, any rational attacker will iterate through methods costing less time and resources. It suggests that many operators make mistakes that are not necessarily directly related to Tor, such as exposing them via cryptocurrency transactions. Also, the academically proposed attacks might not work as intended, alert hidden service operators, and will be more problematic to use in court instead of well-known methods. Many methods also apply to other fields besides Tor and hidden services. Because hidden services share many risks with traditional web services, operators must mitigate these too. The academic literature review does not cover risks to general web services, and the court documents only partially cover them as a side effect. This thesis mainly analyzes and mitigates risks that apply specifically to hidden services. Therefore, operators must also consider the risks for web services to get a complete overview.

Other attackers, such as hackers, act secretly, and their methods are not directly

assessable. They will utilize entirely different approaches as they cannot use legal instruments to gather information from involved third parties and will preferably exploit software or human weaknesses. Hackers can more easily execute DoS attacks on hidden services because Tor makes the clients indistinguishable and requires extensive calculations from hidden services during the circuit initialization. Otherwise, the attack surface for hackers does not differ significantly from web services. The possibilities of secret agencies are even harder to assess but are likely to be remarkably more potent than those of law enforcement.

The Tor community reacts to discovered weaknesses, and some researchers inform them before publishing their papers. Additionally, Tor had considerable developments. Therefore, many previously discovered vulnerabilities became obsolete, but these can still provide valuable hints for undiscovered weaknesses. Often, similar attack vectors are reusable if applied differently.

8.2 Limitations of Hardened Hidden Services

Section 7.5 provides a comprehensive architecture to enhance the security of hidden services. Redundant captcha servers ensure availability, while the layered structure mitigates the impact of attacks. Even if attackers compromise the captcha and request servers, the hidden service recovers after rotating to new servers. Attackers must decloak several Tor connections with different entry guards and relays to take over the entire infrastructure. Attacks become less likely, and operators can even diversify their defenses by deploying various additional anonymization techniques for communication between their servers. Attackers must chain multiple attacks to succeed in infiltrating the infrastructure. The analyzed court documents from section 5.5 show that investigated hidden services use less effective architectures, and the proposed one would have impeded the technical investigations.

In addition to the improvements, the underlying programs must also be secure.

Otherwise, attackers can infiltrate infrastructure directly through user input or other security gaps. Because this applies to all web services, operators must consider them even without using Tor. The guidelines also contain a contingency plan for operators: even if attackers get to their servers, the hosting provider should not be able to provide information linking to the operator.

The general practices for hidden services and their operators are not fully detailed but rather general. On the one hand, operators have to actively think about how to deal with it and can only use them as a rough guide. On the other hand, this covers a larger spectrum since not every small new attack technique leads to modifications. Furthermore, operational security is a broad field that does not exclusively affect hidden services but also web services. Every activity that generates data visible to attackers can give them new angles. Since the number of potential actions is enormous and attackers derive knowledge creatively, this thesis cannot provide a systematic way to mitigate all possible attacks.

The proposed practices and architecture do not entirely cover all Tor-related issues. The proposals focus on the specified attack vectors from the prior academic and internet literature review. Distinct vulnerabilities might appear in other criminal cases or academic publications but not in the reviewed sources. For example, it remains open how operators determine trustworthy guards or publish their onion addresses. In both cases, the operators must reckon with severe consequences if they make mistakes. If they share onion addresses in unprotected environments, attackers can skip all attacks on the hidden service and target the person instead.

The proposed architecture offers additional protection also by adding extra components. Each component increases the overall complexity of a hidden service, and more involved programs need monitoring and regular updates. Operators must manage a large number of servers and corresponding configurations. Mistakes can lead to deanonymization, take the service offline or introduce other risks. While the

log servers collect log files, operators should evaluate them accordingly. Moreover, the log files are not linked by default because multiple servers from different components create them separately. Without rigorous management, operators will lose the overview, and attacks might go unnoticed.

The scope of the architecture can be off-putting to inexperienced operators, as many components are intertwined, and numerous servers are required. The latter ones also increase the organizational effort with associated risks such as payment information that can be traced back to the operators and burden the financial budget. As Tor constantly evolves, the architecture needs constant improvements to keep up with changes. Operators should follow the Tor mailing list to anticipate the latest developments and emerging vulnerabilities. Hidden services need to install updates fast and shut down their service in case of vulnerabilities until a hotfix is published.

8.3 Transfer into Practice

While some parts of the proposed architecture already exist, many are only theoretical. Vanguards, Onionbalance, and EndGame for the captcha servers already exist. The software for the control servers does not exist. Ideally, it should not only be a script but also use graphical user interfaces to simplify the handling for operators. Existing software can fulfill some functions. For example, squid [140] is suitable software for the proxy functionality of the VPS and HAProxy [141] for the web application firewall to filter potentially dangerous user input before the application parses it.

Initially, a community should determine suitable programs to implement the architecture. These should be open source, have a track record with few vulnerabilities, and maintain continuous development. It simplifies integration, avoids license problems, and increases security. In the next step, all programs require a pondered configuration to ensure compatibility between components. Each operator must execute some operations manually, for example, providing an onion key or files for

the hidden service content. All other tasks, for example, installing and configuring software on the servers, should happen automatically and take settings from the user input. Consequently, less experienced operators can also run a secure hidden service, and mistakes occur less often during setups. For manual steps like buying servers or storing keys safely, multiple short guidelines could aid operators and serve as cheat sheets.

Ansible or similar programs install and configure servers automatically. With existing readymade scripts, operators only need to modify input variables and can abstract from manual configurations. The number of components carries the risk that parts no longer behave as intended due to errors and paralyze the hidden service. Several developers should work on this project, while others should test the programs under diverse scenarios to find bugs. A stable version requires extensive tests and experience from practical use. The organization as an open source project and some known hidden services as initial partners can give the project the necessary publicity to gain adequate support.

The developers should not implement the entire architecture entirely at once, but step by step analogously to the structure of section 7.5. It has the advantage that the development becomes visible and functional faster. The gained experience for the first components from live usage improves the ongoing process. Empirical values can also flow directly into the development. The architecture contains some general parts without an optimal solution. One example is communication between servers, as servers can communicate with each other via Tor or other anonymization solutions. Operators should make their own choices, and a modular structure allows this. A modular composition enables developers to modify and integrate functionalities easier. External auditors can analyze single components and their connectivity without understanding the entire code base. Newly discovered vulnerabilities and updates to Tor will require the architecture to adapt. Modules can change without altering the

whole architecture as long as the interfaces remain identical.

Depending on the content, hidden services protect themselves against different attackers. Some require very high security that justifies the proposed architecture. Others operate under lower security requirements and only need a few components. A systematic risk assessment guideline can help operators to analyze their risks and determine appropriate measures. Instead of implementing all measures, operators can also realize parts and thereby already significantly improve security. The project could create three different security levels for operators. Lower security levels require fewer resources and manual configuration, while the highest implements the entire architecture. Project ambassadors and comprehensive documentation intended for less tech-savvy people can help operators without deep knowledge to exercise their freedom of speech securely. In general, graphical user interfaces support operators to configure hidden services, and visualized data can produce a transparent overview. With a good overview and usability, operators will save time and make fewer mistakes.

Despite all measures, operators should create contingency plans in case of security vulnerabilities in Tor or other deployed software. Safety should always have priority. Over time, the community can gather experience to establish robust default values, for example, rotation times in Vanguards or the number of introduction points in descriptors.

9 Conclusion

When reviewing the academic literature, no publication explicitly examined hardening individual hidden services but generally improving the Tor protocol. While these are positive steps, many vulnerabilities have no apparent solution or require lengthy changes. This thesis contributes to improving the current situation for operators and shows long-term improvement possibilities for the Tor protocol.

The vulnerability specifications and classification provide a comprehensive overview of risks for hidden serves. These are the foundation for operators to make informed decisions and assess the risks they face. Furthermore, future research can build upon the classification and extend the categories to cover risks even broader. Additionally, the common classification supports the research community in standardizing terms and avoiding misunderstandings. Possible attack methods could utilize several techniques, which may question the current categories or create new super categories. The classifications in the figures 4.1 and 5.1 from the academic and internet literature review can be combined to include other approaches from real attackers in the academic research that have been neglected so far.

Combining academic publications and investigative methods from law enforcement is a novel approach in this field and enables a differentiated risk assessment for hidden services. Practically, attackers will use less sophisticated methods to decloak hidden services, while in the case of failure, potent attackers can resort to more complex and less tested techniques. Many journalists, political activists, and activists do not

necessarily require the highest security standards. Still, simple attacks endanger their freedom and restrict freedom of speech, especially in totalitarian regimes, even if these are not academically relevant.

Although the reviews do not include the entire literature, the systematic approach covers a significant portion and can serve as a starting point for further additions. At the same time, the overview also includes already fixed vulnerabilities, as these can serve as a starting point for new vulnerabilities. However, future audits of the Tor software and practical experiments should analyze the underlying mechanism to explore unknown attack vectors or determine if the updates fundamentally solve the issues. Due to the time constraint and limited accessibility, this thesis does not include techniques from hackers or intelligence services. Interviews with operators from popular hidden services or hackers could provide further information in the future. Although, they are likely to keep a low profile to protect their anonymity. In this case, law enforcement reports about hacker groups and security companies dealing with them can provide information about techniques hackers use. However, this information is inherently more difficult to verify than court documents and academic papers.

While there are a few projects or shorter guides on the internet for securing hidden services, these do not build on each other and refer to specific issues. They do not meet academic standards, nor do they support operators extensively. In contrast, this thesis systematically derives practices from the classified risks and covers them comprehensively. Operators can use them as a central source of information, and they can guide future projects and modifications. The devised practices and configuration recommendations protect hidden services and their operators, as these are particularly easy for attackers to exploit. The academic literature on hidden services deals primarily with technical problems, while attackers first use more straightforward methods. At the same time, however, this thesis focuses specifically

on hidden services. Therefore, the reviews, derived practices, and the proposed architecture only partially cover risks that also apply to web services. Another possible research direction is to investigate how often attackers utilize classified methods. Simultaneously, it is clear that sophisticated attackers, like U.S. law enforcement, already apply them, as the Silk Road 2.0 case in subsection 5.5.2 shows. It is imperative to communicate this limitation to operators. Future works can integrate existing research for web services to provide operators with a single starting reference for hardening hidden services in all aspects.

This thesis also develops a theoretical architecture reducing technical vulnerabilities specific to hidden services and massively hampering attacks. It focuses more on academic and technical attacks and effectively mitigates methods presented in academic publications and court documents. However, it remains a future task to implement this architecture. Some design decisions have been intentionally left open for operators and future developers to determine, as no optimal solution is apparent. Further research and empirical experiences are required to expand the architecture and address the gaps.

A modular design for the architecture supports future modifications and changes if new vulnerabilities arise or better ideas emerge. The added complexity also introduces additional risks, as the code can contain vulnerabilities. At the same time, the clear separation of functionalities into multiple servers also leads to a simplified configuration of each server. Furthermore, this thesis describes in section 8.3 how the architecture can be transferred into practice. An active community, widespread use, and appropriate communication with operators are crucial points. In the future, risk profiles can be created to assist operators in selecting suitable measures. Depending on the security level, a subset of the proposed hardening measures is already sufficient, while the entire set comprehensively increases security even for hidden services with extraordinarily high-security requirements.

Further research can focus on isolated attack methods or combine and adapt results in a larger context. New watermarking methods and countermeasures can be developed for the former. Tor currently uses no circuit-wide padding procedure that would hinder traffic correlations. New padding mechanisms could provide better protection by default instead of artificially creating cover traffic with scripts, as subsection 7.5.5 mentions. Website fingerprinting attacks have so far focused on clients browsing the internet or contacting hidden services. Additional research can show whether ISPs or data centers also can carry out the attack and whether it is even more powerful. Because hidden services usually have more traffic than Tor clients, and malicious clients can send specific requests with unique patterns, website fingerprinting could be significantly more effective for hidden services. As the academic research focuses on Tor, comparisons with other anonymization networks can yield new improvements for Tor, and extensive infrastructures like the proposed architecture could benefit from using several different ones. Privacy in cryptocurrencies, secure software, and social engineering do not relate directly to Tor, but the far-reaching significance makes attacks particularly severe, and hidden services rely on them.

References

[1] G. Greenwald, E. MacAskill, and L. Poitras, *Edward Snowden: the whistle-blower behind the NSA surveillance revelations*, Jun. 2013. [Online]. Available: https://www.theguardian.com/world/2013/jun/09/edward-snowden-nsa-whistleblower-surveillance (visited on 12/03/2022).

[2] Deutsche Welle, *ECJ rules against mass data retention in Germany*, Sep. 2022. [Online]. Available: https://www.dw.com/en/german-data-retention-rules-not-compatible-with-eu-law-says-top-court/a-63178438 (visited on 12/03/2022).

[3] A. Meister, *Polizei setzt Staatstrojaner alle zwei Wochen ein*, Aug. 2022. [Online]. Available: https://netzpolitik.org/2022/justizstatistik-2020-polizei-setzt-staatstrojaner-alle-zwei-wochen-ein/ (visited on 12/03/2022).

[4] Media Freedom Rapid Response, *MAPPING MEDIA FREEDOM MONITORING REPORT January – June 2022*, Sep. 2022. [Online]. Available: https://ipi.media/wp-content/uploads/2022/09/MFRR-Monitoring-Report_2022.pdf (visited on 12/03/2022).

[5] A. Biryukov, I. Pustogarov, F. Thill, and R.-P. Weinmann, "Content and Popularity Analysis of Tor Hidden Services", in *2014 IEEE 34th International Conference on Distributed Computing Systems Workshops*, IEEE, Jun. 2014. DOI: 10.1109/icdcsw.2014.20.

[6] A. Pfitzmann and M. Hansen, *Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management – A Consolidated Proposal for Terminology*, Feb. 2008. [Online]. Available: `http://dud.inf.tu-dresden.de/literatur/Anon_Terminology_v0.31.pdf` (visited on 12/03/2022).

[7] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, Oct. 2008. [Online]. Available: `https://bitcoin.org/bitcoin.pdf` (visited on 12/03/2022).

[8] A. Biryukov and I. Pustogarov, "Bitcoin over Tor isn't a Good Idea", in *2015 IEEE Symposium on Security and Privacy*, Oct. 2015, pp. 122–134. DOI: `10.1109/SP.2015.15`.

[9] F. Reid and M. Harrigan, "An Analysis of Anonymity in the Bitcoin System", in *2011 IEEE Third International Conference on Privacy, Security, Risk and Trust and 2011 IEEE Third International Conference on Social Computing*, Oct. 2011, pp. 1318–1326. DOI: `10.1109/PASSAT/SocialCom.2011.79`.

[10] gmaxwell, *CoinJoin: Bitcoin privacy for the real world*, Aug. 2013. [Online]. Available: `https://bitcointalk.org/index.php?topic=279249.0` (visited on 12/03/2022).

[11] J. H. Ziegeldorf, F. Grossmann, M. Henze, N. Inden, and K. Wehrle, "CoinParty: Secure Multi-Party Mixing of Bitcoins", in *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '15, San Antonio, Texas, USA: Association for Computing Machinery, Mar. 2015, 75–86. DOI: `10.1145/2699026.2699100`.

[12] koe1, K. Alonso, and S. Noether, *Zero to Monero: Second Edition*, Apr. 2020. [Online]. Available: `https://www.getmonero.org/library/Zero-to-Monero-2-0-0.pdf` (visited on 12/03/2022).

[13] D. L. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms", *Commun. ACM*, vol. 24, no. 2, 84–90, Feb. 1981. DOI: 10.1145/358549.358563.

[14] M. Reed, P. Syverson, and D. Goldschlag, "Anonymous Connections and Onion Routing", *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 482–494, May 1998. DOI: 10.1109/49.668972.

[15] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router", in *Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04, San Diego, CA: USENIX Association, Jun. 2004, p. 21.

[16] Tor Project, *Tor's extensions to the SOCKS protocol*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/socks-extensions.txt (visited on 12/06/2022).

[17] Tor Project, *Tor's protocol specifications*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/ (visited on 12/03/2022).

[18] Tor Project, *About Tor Browser*, Dec. 2022. [Online]. Available: https://tb-manual.torproject.org/about/ (visited on 12/04/2022).

[19] N. J. Al Fardan and K. G. Paterson, "Lucky Thirteen: Breaking the TLS and DTLS Record Protocols", in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 526–540. DOI: 10.1109/SP.2013.42.

[20] R. Dingledine and N. Mathewson, *Tor Protocol Specification*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/tor-spec.txt (visited on 12/06/2022).

[21] T. Wilson-Brown, *Tor Directory List Format*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/dir-list-spec.txt (visited on 12/06/2022).

[22] M. Perry and G. Kadianakis, *Tor Padding Specification*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/padding-spec.txt (visited on 12/06/2022).

[23] L. Overlier and P. Syverson, "Locating Hidden Servers", in *2006 IEEE Symposium on Security and Privacy (S&P'06)*, May 2006, 15 pp.–114. DOI: 10.1109/SP.2006.24.

[24] I. Lovecruft, G. Kadianakis, O. Bini, and N. Mathewson, *Tor Guard Specification*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/guard-spec.txt (visited on 12/06/2022).

[25] Tor Project, *Tor directory protocol, version 3*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/dir-spec.txt (visited on 12/06/2022).

[26] Tor Project, *Consensus Health*, Nov. 2022. [Online]. Available: https://consensus-health.torproject.org/ (visited on 11/02/2022).

[27] Tor Project, *Tor Rendezvous Specification - Version 3*, Nov. 2022. [Online]. Available: https://gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt (visited on 12/06/2022).

[28] S. Matic, P. Kotzias, and J. Caballero, "CARONTE: Detecting Location Leaks for Deanonymizing Tor Hidden Services", in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, ACM, Oct. 2015. DOI: 10.1145/2810103.2813667.

[29] H. Al Jawaheri, M. Al Sabah, Y. Boshmaf, and A. Erbad, "Deanonymizing Tor hidden service users through Bitcoin transactions analysis", *Computers & Security*, vol. 89, p. 101 684, Feb. 2020. DOI: 10.1016/j.cose.2019.101684.

[30] G. Me, L. Pesticcio, and P. Spagnoletti, "Discovering Hidden Relations Between Tor Marketplaces Users", in *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, IEEE, Nov. 2017. DOI: 10.1109/dasc-picom-datacom-cyberscitec.2017.93.

[31] M. Steinebach, M. Schäfer, A. Karakuz, K. Brandl, and Y. Yannikos, "Detection and Analysis of Tor Onion Services", in *Proceedings of the 14th International Conference on Availability, Reliability and Security*, ACM, Aug. 2019. DOI: 10.1145/3339252.3341486.

[32] E. Eaton, S. Sasy, and I. Goldberg, "Improving the Privacy of Tor Onion Services", in *Applied Cryptography and Network Security*, Springer International Publishing, Jun. 2022, pp. 273–292. DOI: 10.1007/978-3-031-09234-3_14.

[33] T. Hoeller, M. Roland, and R. Mayrhofer, "On the state of V3 onion services", in *Proceedings of the ACM SIGCOMM 2021 Workshop on Free and Open Communications on the Internet*, ACM, Aug. 2021. DOI: 10.1145/3473604.3474565.

[34] Q. Tan, Y. Gao, J. Shi, X. Wang, and B. Fang, "A closer look at Eclipse attacks against Tor hidden services", in *2017 IEEE International Conference on Communications (ICC)*, IEEE, May 2017. DOI: 10.1109/icc.2017.7996832.

[35] Q. Tan, Y. Gao, J. Shi, X. Wang, B. Fang, and Z. Tian, "Toward a Comprehensive Insight Into the Eclipse Attacks of Tor Hidden Services", *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1584–1593, Apr. 2019. DOI: 10.1109/jiot.2018.2846624.

[36] A. Biryukov, I. Pustogarov, and R. Weinmann, "Trawling for Tor Hidden Services: Detection, Measurement, Deanonymization", in *2013 IEEE Symposium on Security and Privacy*, IEEE, May 2013. DOI: 10.1109/sp.2013.15.

[37] A. Panchenko, A. Mitseva, M. Henze, F. Lanze, K. Wehrle, and T. Engel, "Analysis of Fingerprinting Techniques for Tor Hidden Services", in *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society*, ACM, Oct. 2017. DOI: 10.1145/3139550.3139564.

[38] M. Yang, X. Gu, Z. Ling, C. Yin, and J. Luo, "An active de-anonymizing attack against tor web traffic", *Tsinghua Science and Technology*, vol. 22, no. 6, pp. 702–713, Dec. 2017. DOI: 10.23919/tst.2017.8195352.

[39] R. Overdorf, M. Juarez, G. Acar, R. Greenstadt, and C. Diaz, "How Unique is Your .onion?", in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Oct. 2017. DOI: 10.1145/3133956.3134005.

[40] Y. Sun, X. Luo, H. Wang, and Z. Ma, "A Method for Identifying Tor Users Visiting Websites Based on Frequency Domain Fingerprinting of Network Traffic", *Security and Communication Networks*, vol. 2022, W. Meng, Ed., pp. 1–12, Jan. 2022. DOI: 10.1155/2022/3306098.

[41] P. Mayank and A. K. Singh, "Tor traffic identification", in *2017 7th International Conference on Communication Systems and Network Technologies (CSNT)*, IEEE, Nov. 2017. DOI: 10.1109/csnt.2017.8418516.

[42] F. A. Saputra, I. U. Nadhori, and B. F. Barry, "Detecting and blocking onion router traffic using deep packet inspection", in *2016 International Electronics Symposium (IES)*, IEEE, Sep. 2016. DOI: 10.1109/elecsym.2016.7861018.

[43] A. Gurunarayanan, A. Agrawal, A. Bhatia, and D. K. Vishwakarma, "Improving the performance of Machine Learning Algorithms for TOR detection", in

2021 International Conference on Information Networking (ICOIN), IEEE, Jan. 2021. DOI: 10.1109/icoin50884.2021.9333989.

[44] A. Montieri, D. Ciuonzo, G. Aceto, and A. Pescape, "Anonymity Services Tor, I2P, JonDonym: Classifying in the Dark (Web)", *IEEE Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 662–675, May 2020. DOI: 10.1109/tdsc.2018.2804394.

[45] A. Kwon, M. AlSabah, D. Lazar, M. Dacier, and S. Devadas, "Circuit Fingerprinting Attacks: Passive Deanonymization of Tor Hidden Services", in *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C.: USENIX Association, Aug. 2015, pp. 287–302.

[46] A. Mitseva, A. Panchenko, F. Lanze, M. Henze, K. Wehrle, and T. Engel, "POSTER: Fingerprinting Tor Hidden Services", in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Oct. 2016. DOI: 10.1145/2976749.2989054.

[47] F. Platzer, M. Schäfer, and M. Steinebach, "Critical traffic analysis on the tor network", in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ACM, Aug. 2020. DOI: 10.1145/3407023.3409180.

[48] A. Iacovazzi, S. Sarda, and Y. Elovici, "Inflow: Inverse Network Flow Watermarking for Detecting Hidden Servers", in *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, IEEE, Apr. 2018. DOI: 10.1109/infocom.2018.8486375.

[49] R. Wang, Q. Wen, H. Zhang, S. Qin, and W. Li, "Transparent Discovery of Hidden Service", *IEICE Transactions on Information and Systems*, vol. E99.D, no. 11, pp. 2817–2820, Nov. 2016. DOI: 10.1587/transinf.2016edl8100.

[50] A. Iacovazzi, D. Frassinelli, and Y. Elovici, "The DUSTER Attack: Tor Onion Service Attribution Based on Flow Watermarking with Track Hiding", in *22nd*

International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), Chaoyang District, Beijing: USENIX Association, Sep. 2019, pp. 213–225.

[51] M. Chen, X. Wang, T. Liu, J. Shi, Z. Yin, and B. Fang, "SignalCookie: Discovering Guard Relays of Hidden Services in Parallel", in *2019 IEEE Symposium on Computers and Communications (ISCC)*, IEEE, Jun. 2019. DOI: 10.1109/iscc47284.2019.8969639.

[52] M. Chen, X. Wang, J. Shi, Y. Gao, C. Zhao, and W. Sun, "Towards Comprehensive Security Analysis of Hidden Services Using Binding Guard Relays", in *Information and Communications Security*, Springer International Publishing, Feb. 2020, pp. 521–538. DOI: 10.1007/978-3-030-41579-2_30.

[53] S. J. Murdoch, "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, Oct. 2006.

[54] S. Zander and S. J. Murdoch, "An Improved Clock-Skew Measurement Technique for Revealing Hidden Services", in *Proceedings of the 17th Conference on Security Symposium*, ser. SS'08, San Jose, CA: USENIX Association, Jul. 2008, 211–225. DOI: 10.5555/1496711.1496726.

[55] S. Murdoch and G. Danezis, "Low-Cost Traffic Analysis of Tor", in *2005 IEEE Symposium on Security and Privacy (S&P'05)*, Jun. 2005, pp. 183–195. DOI: 10.1109/SP.2005.12.

[56] N. Hopper, E. Y. Vasserman, and E. Chan-TIN, "How much anonymity does network latency leak?", *ACM Transactions on Information and System Security*, vol. 13, no. 2, pp. 1–28, Feb. 2010. DOI: 10.1145/1698750.1698753.

[57] M. Simioni, P. Gladyshev, B. Habibnia, and P. R. Nunes de Souza, "Monitoring an anonymity network: Toward the deanonymization of hidden services",

Forensic Science International: Digital Investigation, vol. 38, p. 301 135, Oct. 2021. DOI: 10.1016/j.fsidi.2021.301135.

[58] B. Shebaro, F. Perez-Gonzalez, and J. R. Crandall, "Leaving timing-channel fingerprints in hidden service log files", *Digital Investigation*, vol. 7, S104–S113, Aug. 2010. DOI: 10.1016/j.diin.2010.05.013.

[59] F. Cangialosi, D. Levin, and N. Spring, "Ting: Measuring and Exploiting Latencies Between All Tor Nodes", in *Proceedings of the 2015 Internet Measurement Conference*, ACM, Oct. 2015. DOI: 10.1145/2815675.2815701.

[60] Y. Gao, J. Shi, X. Wang, R. Shi, C. Zhao, and C. Li, "A Two-Stage Deanonymization Attack Towards Bitcoin Hidden Service Nodes", in *2021 IEEE 23rd Int Conf on High Performance Computing & Communications; 7th Int Conf on Data Science & Systems; 19th Int Conf on Smart City; 7th Int Conf on Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, IEEE, Dec. 2021. DOI: 10.1109/hpcc-dss-smartcity-dependsys53884.2021.00095.

[61] L. Hellebrandt, I. Homoliak, K. Malinka, and P. Hanacek, "Increasing Trust in Tor Node List Using Blockchain", in *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, IEEE, May 2019. DOI: 10.1109/bloc.2019.8751340.

[62] R. Jansen, F. Tschorsch, A. Johnson, and B. Scheuermann, "The Sniper Attack: Anonymously Deanonymizing and Disabling the Tor Network", in *Proceedings 2014 Network and Distributed System Security Symposium*, Internet Society, Jan. 2014. DOI: 10.14722/ndss.2014.23288.

[63] C. Döpmann, V. Franck, and F. Tschorsch, "Onion Pass: Token-Based Denial-of-Service Protection for Tor Onion Services", in *2021 IFIP Networking Conference (IFIP Networking)*, IEEE, Jun. 2021. DOI: 10/gpvk89.

[64] M. V. Barbera, V. P. Kemerlis, V. Pappas, and A. D. Keromytis, "CellFlood: Attacking Tor Onion Routers on the Cheap", in *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, Sep. 2013, pp. 664–681. DOI: 10.1007/978-3-642-40203-6_37.

[65] R. Jansen, T. Vaidya, and M. Sherr, "Point Break: A Study of Bandwidth Denial-of-Service Attacks against Tor", in *28th USENIX Security Symposium (USENIX Security 19)*, Santa Clara, CA: USENIX Association, Aug. 2019, pp. 1823–1840.

[66] K. Bauer, D. McCoy, D. Grunwald, T. Kohno, and D. Sicker, "Low-resource routing attacks against tor", in *Proceedings of the 2007 ACM workshop on Privacy in electronic society - WPES '07*, ACM Press, Oct. 2007. DOI: 10.1145/1314333.1314336.

[67] L. Zhang, J. Luo, M. Yang, and G. He, "Application-level attack against Tor's hidden service", in *2011 6th International Conference on Pervasive Computing and Applications*, IEEE, Oct. 2011. DOI: 10.1109/icpca.2011.6106555.

[68] Q. Li, P. Liu, and Z. Qin, "A Stealthy Attack Against Tor Guard Selection", *International Journal of Security and Its Applications*, vol. 9, no. 11, pp. 391–402, Nov. 2015. DOI: 10.14257/ijsia.2015.9.11.36.

[69] A. Johnson, C. Wacek, R. Jansen, M. Sherr, and P. Syverson, "Users get routed: traffic correlation on tor by realistic adversaries", in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*, ACM Press, Nov. 2013. DOI: 10.1145/2508859.2516651.

[70] J. Juen, A. Johnson, A. Das, N. Borisov, and M. Caesar, "Defending Tor from Network Adversaries: A Case Study of Network Path Prediction", *Proceedings on Privacy Enhancing Technologies*, vol. 2015, no. 2, pp. 171–187, Jun. 2015. DOI: 10.1515/popets-2015-0021.

[71] Y. Sun, A. Edmundson, L. Vanbever, *et al.*, "RAPTOR: Routing Attacks on Privacy in Tor", in *24th USENIX Security Symposium (USENIX Security 15)*, Washington, D.C.: USENIX Association, Aug. 2015, pp. 271–286.

[72] Z. Ling, J. Luo, K. Wu, and X. Fu, "Protocol-level hidden server discovery", in *2013 Proceedings IEEE INFOCOM*, IEEE, Apr. 2013. DOI: 10.1109/infcom.2013.6566894.

[73] Y. Ma and X. Xu, "Locating tor's hidden service clients based on protocol feature", in *2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, IEEE, Dec. 2017. DOI: 10.1109/itnec.2017.8284989.

[74] N. Yoshiura and K. Koizumi, "A Method of Collecting the IP Addresses of Hidden Server in Tor Networks", in *Proceedings of the 2020 9th International Conference on Software and Computer Applications*, ACM, Feb. 2020. DOI: 10.1145/3384544.3384589.

[75] Q. Tan, X. Wang, W. Shi, J. Tang, and Z. Tian, "An Anonymity Vulnerability in Tor", *IEEE/ACM Transactions on Networking*, pp. 1–14, May 2022. DOI: 10.1109/tnet.2022.3174003.

[76] A. Greubel, S. Pohl, and S. Kounev, "Quantifying measurement quality and load distribution in Tor", in *Annual Computer Security Applications Conference*, ACM, Dec. 2020. DOI: 10.1145/3427228.3427238.

[77] United States Department of Justice Office of Public Affairs, *Darknet Fentanyl Dealer Indicted in Nationwide Undercover Operation Targeting Darknet Vendors Who were Selling to Thousands of U.S. Residents*, May 2019. [Online]. Available: https://www.justice.gov/opa/pr/darknet-fentanyl-dealer-indicted-nationwide-undercover-operation-targeting-darknet-vendors (visited on 12/04/2022).

[78] United States District Court for the Eastern District of Virginia, *William Anderson Burgamy Iv Affidavit*, Jul. 2020. [Online]. Available: https://www.justice.gov/usao-edva/press-release/file/1317991/download (visited on 12/04/2022).

[79] United States District Court for the Western District of Tennessee, *Kevin Ombisi And Eric Russell, Jr Affidavit*, Sep. 2021. [Online]. Available: https://www.justice.gov/opa/press-release/file/1367096/download (visited on 12/04/2022).

[80] United States Attorney's Office, Eastern District of California, *Sacramento Man Indicted for Drug Distribution Via the Darknet*, Oct. 2021. [Online]. Available: https://www.justice.gov/usao-edca/pr/sacramento-man-indicted-drug-distribution-darknet (visited on 12/04/2022).

[81] United States Department of Justice Office of Public Affairs, *International Law Enforcement Operation Targeting Opioid Traffickers on the Darknet Results in 150 Arrests Worldwide and the Seizure of Weapons, Drugs, and over $31 Million*, Oct. 2021. [Online]. Available: https://www.justice.gov/opa/pr/international-law-enforcement-operation-targeting-opioid-traffickers-darknet-results-150 (visited on 12/04/2022).

[82] United States District Court for the District of Columbia, *Luis Miguel Teixeira-spencer And Olatunji Dawodu Indictment*, Jan. 2021. [Online]. Available: https://www.justice.gov/opa/page/file/1444551/download (visited on 12/04/2022).

[83] United States District Court for the Eastern District of Virginia, *Albie Pagan Criminal Complaint*, Feb. 2021. [Online]. Available: https://www.justice.gov/opa/page/file/1444536/download (visited on 12/04/2022).

[84] United States Department of Justice Office of Public Affairs, *Justice Department Investigation Leads to Shutdown of Largest Online Darknet Marketplace*, Apr. 2022. [Online]. Available: https://www.justice.gov/opa/pr/justice-department-investigation-leads-shutdown-largest-online-darknet-marketplace (visited on 12/04/2022).

[85] United States District Court for the Eastern District of Virginia, *Alexandre Cazes Forfeiture Complaint and Exhibits*, Jul. 2017. [Online]. Available: https://www.justice.gov/opa/press-release/file/982821/download (visited on 12/04/2022).

[86] United States District Court for the District of Maryland, *Eric Eoin Marques Criminal Complaint*, Aug. 2013. [Online]. Available: https://www.justice.gov/opa/press-release/file/1147691/download (visited on 12/04/2022).

[87] United States Department of Justice Office of Public Affairs, *Ohio Resident Charged with Operating Darknet-Based Bitcoin "Mixer," which Laundered Over $300 Million*, Feb. 2020. [Online]. Available: https://www.justice.gov/opa/pr/ohio-resident-charged-operating-darknet-based-bitcoin-mixer-which-laundered-over-300-million (visited on 12/04/2022).

[88] United States District Court for the District of Columbia, *Jong Woo Son Indictment*, Aug. 2018. [Online]. Available: https://www.justice.gov/opa/press-release/file/1210441/download (visited on 12/04/2022).

[89] United States District Court for the District of Columbia, *Michael Rahim Mohammad Indictment*, May 2019. [Online]. Available: https://www.justice.gov/usao-dc/press-release/file/1257641/download (visited on 12/04/2022).

[90] United States District Court for the Western District of North Carolina, *David Lynn Browning Criminal Complaint*, Jul. 2015. [Online]. Available: https://

storage.courtlistener.com/recap/gov.uscourts.ncwd.78069.3.0.pdf (visited on 12/04/2022).

[91] United States District Court for the Eastern District of Virginia, *Application For a Search Warrant*, Feb. 2015. [Online]. Available: https://storage.courtlistener.com/recap/gov.uscourts.ncwd.78069.80.1.pdf (visited on 12/04/2022).

[92] United States District Court for the Southern District of New York, *Apostolos Trovias Criminal Complaint*, Feb. 2021. [Online]. Available: https://www.justice.gov/usao-sdny/press-release/file/1410806/download (visited on 12/04/2022).

[93] United States District Court for the Eastern District of California, *Kristy Lynn Felkins Criminal Complaint*, Sep. 2020. [Online]. Available: https://www.justice.gov/media/1093581/dl?inline= (visited on 12/04/2022).

[94] United States District Court for the Northern District of West Virginia, *Jonathan Toebbe and Diana Toebbe Criminal Complaint*, Oct. 2021. [Online]. Available: https://www.justice.gov/opa/press-release/file/1440946/download (visited on 12/04/2022).

[95] United States Department of Justice Attorney's Office Eastern District of Virginia, *Fairfax Man Sentenced for Receiving, Soliciting and Promoting Child Pornography*, Aug. 2021. [Online]. Available: https://www.justice.gov/usao-edva/pr/fairfax-man-sentenced-receiving-soliciting-and-promoting-child-pornography (visited on 12/04/2022).

[96] United States District Court for the Eastern District of California, *Marcos Paulo De Oliveira-Annibale Criminal Complaint*, May 2019. [Online]. Available: https://www.justice.gov/media/1002611/dl?inline= (visited on 12/04/2022).

[97] United States District Court for the Southern District of New York, *Declaration Of Christopher Tarbell*, Sep. 2014. [Online]. Available: https://upload.wikimedia.org/wikipedia/commons/e/e4/Tarbell_declaration.pdf (visited on 12/05/2022).

[98] C. Sommers and E. Bernstein, *Inside the FBI takedown of the mastermind behind website offering drugs, guns and murders for hire*, Nov. 2020. [Online]. Available: https://www.cbsnews.com/news/ross-ulbricht-dread-pirate-roberts-silk-road-fbi/ (visited on 12/05/2022).

[99] United States Court of Appeals for the Second Circuit, *On Appeal from the United States District Court for the Southern District of New York (New York City)*, Jan. 2016. [Online]. Available: https://freeross.org/wp-content/uploads/2018/02/Doc_34_Jan_12_Vol_IV_Appendix_A769-A1050.pdf (visited on 12/05/2022).

[100] B. Krebs, *Silk Road Lawyers Poke Holes in FBI's Story*, Oct. 2014. [Online]. Available: https://krebsonsecurity.com/2014/10/silk-road-lawyers-poke-holes-in-fbis-story/ (visited on 12/05/2022).

[101] R. Graham, *Reading the Silk Road configuration*, Oct. 2014. [Online]. Available: https://blog.erratasec.com/2014/10/reading-silk-road-configuration.html (visited on 12/05/2022).

[102] United States District Court for the Southern District of New York, *Roger Thomas Clark Criminal Complaint*, Apr. 2015. [Online]. Available: https://www.justice.gov/usao-sdny/file/797251/download (visited on 12/05/2022).

[103] United States Department of Justice, *Former Secret Service Agent Sentenced To 71 Months In Scheme Related To Silk Road Investigation*, Dec. 2015.

[Online]. Available: https://oig.justice.gov/sites/default/files/2019-12/2015-12-07.pdf (visited on 12/04/2022).

[104] Superior Court of King County, Washington, *Affidavit for Search Warrant*, Jan. 2015. [Online]. Available: https://cdn1.vox-cdn.com/uploads/chorus_asset/file/3326868/2015-01-07-kingcountysuperiorcourt-brianfarrel-searchwarrantaffidavit.0.pdf (visited on 12/05/2022).

[105] arma, *Tor security advisory: "relay early" traffic confirmation attack*, Jul. 2014. [Online]. Available: https://blog.torproject.org/tor-security-advisory-relay-early-traffic-confirmation-attack/ (visited on 12/05/2022).

[106] United States District Court for the Western District of Washington, *Order On Defendant's Motion To Compel*, Feb. 2016. [Online]. Available: https://s3.documentcloud.org/documents/2719591/Farrell-Weds.pdf (visited on 12/05/2022).

[107] United States District Court for the Southern District of New York, *Blake Benthall Criminal Complaint*, Oct. 2014. [Online]. Available: https://www.justice.gov/sites/default/files/usao-sdny/legacy/2015/03/25/Benthall%2C%20Blake%20Complaint.pdf (visited on 12/05/2022).

[108] United States District Court for the Central District of California, *Tibo Lousee, Klaus-Martin Frost, and Jonathan Kalla Criminal Complaint*, May 2019. [Online]. Available: https://www.justice.gov/opa/press-release/file/1159706/download (visited on 12/05/2022).

[109] National Institute of Standards and Technology, *Guide for Conducting Risk Assessments*, Sep. 2012. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-30r1.pdf (visited on 12/03/2022).

[110] Bundesamt für Sicherheit in der Informationstechnik, *Cyber-Sicherheits-Exposition*, Jul. 2018. [Online]. Available: https://www.allianz-fuer-cybersicherheit.de/SharedDocs/Downloads/Webs/ACS/DE/BSI-CS/BSI-CS_013.pdf?__blob=publicationFile&v=1 (visited on 12/03/2022).

[111] dgoulet, *Onion Service version 2 deprecation timeline*, Jul. 2020. [Online]. Available: https://blog.torproject.org/v2-deprecation-timeline/ (visited on 12/06/2022).

[112] G. Kadianakis, M. Perry, I. R. Learmonth, E. Betts, Baccount, and onionltd, *The Vanguards Onion Service Addon*, Jul. 2021. [Online]. Available: https://github.com/mikeperry-tor/vanguards (visited on 12/03/2022).

[113] G. Kadianakis, M. Perry, and andathan, *Vanguards simulation*, Sep. 2021. [Online]. Available: https://github.com/asn-d6/vanguard_simulator (visited on 12/03/2022).

[114] F. Rochet and O. Pereira, "Dropping on the Edge: Flexibility and Traffic Confirmation in Onion Routing Protocols", *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 2, pp. 27–46, Feb. 2018. DOI: 10.1515/popets-2018-0011.

[115] G. Kadianakis, *Onionbalance*, Jul. 2021. [Online]. Available: https://gitlab.torproject.org/tpo/core/onionbalance/ (visited on 12/03/2022).

[116] Tor Project, *Get Bridges for Tor*, Dec. 2022. [Online]. Available: https://bridges.torproject.org/ (visited on 12/03/2022).

[117] P. Winter, T. Pulls, and J. Fuß, "ScrambleSuit: A Polymorph Network Protocol to Circumvent Censorship", *CoRR*, vol. abs/1305.3199, May 2013. eprint: 1305.3199. [Online]. Available: http://arxiv.org/abs/1305.3199.

[118] D. Barradas, N. Santos, L. Rodrigues, and V. Nunes, "Poking a Hole in the Wall: Efficient Censorship-Resistant Internet Communications by Parasitizing on WebRTC", in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, Virtual Event, USA: Association for Computing Machinery, Nov. 2020, 35–48. DOI: 10.1145/3372297.3417874.

[119] D. Ellard, C. Jones, V. Manfredi, *et al.*, "Rebound: Decoy routing on asymmetric routes via error messages", in *2015 IEEE 40th Conference on Local Computer Networks (LCN)*, Oct. 2015, pp. 91–99. DOI: 10.1109/LCN.2015.7366287.

[120] K. P. Dyer, S. E. Coull, T. Ristenpart, and T. Shrimpton, "Protocol Misidentification Made Easy with Format-Transforming Encryption", in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, Berlin, Germany: Association for Computing Machinery, Nov. 2013, 61–72. DOI: 10.1145/2508859.2516657.

[121] D. Fifield, N. Hardison, J. Ellithorpe, *et al.*, "Evading Censorship with Browser-Based Proxies", in *Privacy Enhancing Technologies*, S. Fischer-Hübner and M. Wright, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, Jul. 2012, pp. 239–258. DOI: 10.1007/978-3-642-31680-7_13.

[122] Tor Project, *Tor Metrics - Users*, Dec. 2022. [Online]. Available: https://bridges.torproject.org/ (visited on 12/03/2022).

[123] meskio, *obfs4 (The obfourscator)*, Oct. 2022. [Online]. Available: https://gitweb.torproject.org/pluggable-transports/obfs4.git/tree/doc/obfs4-spec.txt (visited on 12/03/2022).

[124] J. A. Halderman, S. D. Schoen, N. Heninger, *et al.*, "Lest We Remember: Cold-Boot Attacks on Encryption Keys", *Commun. ACM*, vol. 52, no. 5, 91–98, May 2009. DOI: 10.1145/1506409.1506429.

[125] A. Narayanan and V. Shmatikov, "Fast Dictionary Attacks on Passwords Using Time-Space Tradeoff", in *Proceedings of the 12th ACM Conference on Computer and Communications Security*, ser. CCS '05, Alexandria, VA, USA: Association for Computing Machinery, Nov. 2005, 364–372. DOI: 10.1145/1102120.1102168.

[126] M. Weir, S. Aggarwal, B. d. Medeiros, and B. Glodek, "Password Cracking Using Probabilistic Context-Free Grammars", in *2009 30th IEEE Symposium on Security and Privacy*, May 2009, pp. 391–405. DOI: 10.1109/SP.2009.8.

[127] D. L. Wheeler, "Zxcvbn: Low-Budget Password Strength Estimation", in *Proceedings of the 25th USENIX Conference on Security Symposium*, ser. SEC'16, Austin, TX, USA: USENIX Association, Aug. 2016, 157–173.

[128] A. Biryukov, D. Dinu, and D. Khovratovich, "Argon2: New Generation of Memory-Hard Functions for Password Hashing and Other Applications", in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*, Mar. 2016, pp. 292–302. DOI: 10.1109/EuroSP.2016.31.

[129] P. Juola and D. Vescovi, "Empirical Evaluation of Authorship Obfuscation Using JGAAP", in *Proceedings of the 3rd ACM Workshop on Artificial Intelligence and Security*, ser. AISec '10, Chicago, Illinois, USA: Association for Computing Machinery, Oct. 2010, 14–18. DOI: 10.1145/1866423.1866427.

[130] E. Quiring, A. Maier, and K. Rieck, "Misleading Authorship Attribution of Source Code Using Adversarial Learning", in *Proceedings of the 28th USENIX Conference on Security Symposium*, ser. SEC'19, USA: USENIX Association, Aug. 2019, 479–496.

[131] A. Sadeghian, M. Zamani, and A. A. Manaf, "A Taxonomy of SQL Injection Detection and Prevention Techniques", in *2013 International Conference on Informatics and Creative Multimedia*, Sep. 2013, pp. 53–56. DOI: 10.1109/ICICM.2013.18.

[132] A. Juels and R. L. Rivest, "Honeywords: Making Password-Cracking Detectable", in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13, Berlin, Germany: Association for Computing Machinery, Nov. 2013, 145–160. DOI: 10.1145/2508859.2516671.

[133] ENCRYPTED SUPPORT LP, *Whonix ™ Documentation*, Dec. 2022. [Online]. Available: https://www.whonix.org/wiki/Documentation (visited on 12/12/2022).

[134] The Qubes OS Project, *Qubes OS Introduction*, Dec. 2022. [Online]. Available: https://www.qubes-os.org/doc/ (visited on 12/12/2022).

[135] ENCRYPTED SUPPORT LP, *Qubes-Whonix ™ Overview*, Dec. 2022. [Online]. Available: https://www.whonix.org/wiki/Qubes (visited on 12/12/2022).

[136] onionltd, *EndGame V2 - Onion Service DDOS Prevention Front System*, Jul. 2021. [Online]. Available: https://github.com/onionltd/EndGame.

[137] The Invisible Internet Project (I2P), *Intro - I2P*, Dec. 2022. [Online]. Available: https://geti2p.net/en/about/intro (visited on 12/05/2022).

[138] Red Hat Inc., *OVERVIEW How Ansible works*, Dec. 2022. [Online]. Available: https://www.ansible.com/overview/how-ansible-works (visited on 12/12/2022).

[139] CoinMarketCap, *Today's Cryptocurrency Prices by Market Cap*, Dec. 2022. [Online]. Available: https://coinmarketcap.com/ (visited on 12/05/2022).

[140] squid-cache.org, *Squid: Optimising Web Delivery*, Dec. 2022. [Online]. Available: http://www.squid-cache.org/ (visited on 12/05/2022).

[141] HAProxy, *HAProxy The Reliable, High Performance TCP/HTTP Load Balancer*, Dec. 2022. [Online]. Available: `http://www.haproxy.org/` (visited on 12/05/2022).