Measuring the Impact of Certificate Transparency on Scanning Traffic

Thanh-Dat Nguyen

Cybersecurity
TU Delft

Measuring the Impact of Certificate Transparency on Scanning Traffic

by

Thanh-Dat Nguyen

to obtain the degree of Master of Science

at the Delft University of Technology,

to be defended publicly on Wednesday May 11, 2022 at 14:00.

Student number:	5025664	
Project duration:	Mar 1, 2021 – May 11, 2022	
Thesis committee:	Dr.-Ing. Tobias Fiebig,	TU Delft, Daily supervisor
	Prof. Georgios Smaragdakis	TU Delft, Responsible Professor
	Dr. Fernando Kuipers	TU Delft, Committee member

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

When the time came to choose a research topic for my thesis, I asked my supervisor Tobias for guidance. He steered me towards a security technology that I was not aware of: Certificate Transparency. But the focus of the thesis would not be on the effectiveness but on how it could be abused. It never occured to me that a security measure could be used for malicious purposes. Through Certificate Transparency, I got to discover the world of Internet measurement, an endless source of excitement and frustrations.

I want to thank Tobias for guiding me through my thesis. You generously gave me your time to meet and discuss my progress and issues. I always came out from these meetings with new insights and knowledge to tackle my tasks. Thanks to you, I managed to avoid common pitfalls that saved me countless of hours. I really appreciate your patience for sticking with me until the end.

I want to thank Tom for being my friend. Although I left the Netherlands because of the pandemic, we religiously called every week. Our hangouts helped me to retain some sort of student life and gave me motivation for my thesis.

Last, I want to thank my family for supporting me throughout my studies. You were always there to support and cheer me up. I will always remember fondly when you all came to visit. Although we were cramped in my small studio, we had so many laughs.

Thanh-Dat Nguyen
Delft, April 2022

Abstract

Certificate transparency (CT) is a system that publishes all issued certificates so that they can be audited and monitored by any party. This allows to detect mis-issued certificates quickly and catch the misbehaving certificate authorities. CT makes the certificate ecosystem more transparent and less reliant on trust that trusted certificate authorities (CA) do not issue rogue certificates. Events such as the compromise of the CA DigiNotar in 2011 could be easily and quickly detected with CT. CT makes use of CT logs, where issued certificates from trusted CA are stored. CT logs are append-only ledger that can be accessed by anyone. The entire issue with trust and misissuance seems to be solved by this new technology, CT. But are there any downsides to this solution? Albeit, it helps to detect misissuance quickly, but at the same time introduces an additional source to exploit. CT logs are public ledgers of issued certificates. They can be used to discover new domain names, or to look up what domains exist. Given a domain, an attacker could harness the power of CT logs to look up all the subdomains. CT logs expose every domain, even internal ones that need a certificate but do not want to be known to the public. Furthermore, domains can be easily found without the need to guess domain names or exploiting other resources such as DNS, allowing attackers to generate lists of potential targets easily. In this paper, we analyze the security impact of CT on scanning traffic for domains hosted in IPv4 and IPv6. The domains' certificates are appended to CT logs, to leak the embedded domain name, which is acting as a honeytoken, a lure. We conduct passive measurements on the domain hosts, the hosting webserver and the authoritative DNS server, perform data analysis on the measurements and evaluate the security impact of the certificate request.

Thanh-Dat Nguyen
Delft, April 2022

Contents

1

Introduction

Everybody is browsing on the Internet nowadays. Most people assume that what they do is secure. Most of the time, that is thanks to Transport Layer Security, or more commonly called by its abbreviation TLS. Transport layer security (TLS) is the standard for ensuring verified and encrypted traffic on the Internet. It is used everywhere, from a simple blog to a banking site; the internet without TLS would be unthinkable. It would be like sending a letter without an envelope and hoping that the entire post office does not know about your quirky love letter.

The TLS protocol allows applications to communicate while ensuring confidentiality and integrity of the communication. The protocol relies on public-key certificates to establish a secure communication channel between the client and the server. The public key certificate infrastructure relies on trust to function: a client only accepts certificates that were issued by a trusted issuer, commonly called certificate authorities. This all happens without consumers being aware. They do not have to memorize which certificate authorities they should trust; the list is neatly packaged in every device they own.

The caveat here is trust. What if that trust is abused? All the benefits of TLS would be undermined. Take the example of the Certificate Authority (CA) Diginotar, which was hacked and accessed by an attacker in 2011. Unauthorized certificates for Google were issued and used for MITM attacks in Iran. After the incident was revealed, the trust in Diginotar was shattered: most browsers removed Diginotar from their trusted CA list. Subsequently, Diginotar went bankrupt as it couldn't provide the one thing that every CA must provide: trust.

Is there any system that enables auditing issued certificates? To the rescue comes the concept of Certificate transparency (CT), a system that publishes all issued certificates so that they can be audited and monitored by any party. This allows to detect misissued certificates quickly and catch the misbehaving certificate authorities. The issuance of certificates is stored in certificates logs, an append-only ledger publicly accessible. This solution seems to negate the impact of abused trust in the system. Since 2018, Google Chrome requires all domains to have their certificate to be in a CT log to be considered safe [51].

The entire issue with trust and misissuance seems to be solved by this new technology, CT. But are there any downsides to this solution? Albeit, it helps to detect misissuance quickly, but at the same time introduces an additional source to exploit. CT logs are public ledgers of issued certificates. They can be used to discover new domain names, or to look up what domains exist. Given a domain, an attacker could harness the power of CT logs to look up all the subdomains. CT logs expose every domain, even internal ones that need a certificate but do not want to be known to the public. Furthermore, domains can be easily found without the need to guess domain names or exploiting other resources such as DNS, allowing attackers to generate lists of potential targets easily.

1.1. Research questions

The goal of the research is to analyze the security impact of CT on scanning traffic. We will set up experiments in IPv4 and IPv6 address space, where newly generated domain names will be leaked by being appended to CT logs. We will conduct passive measurement on the domain hosts, perform data analysis on the measurements and evaluate the security impact of the certificate request. The primary research question of the paper we want to answer is:

How does certificate transparency affect scanning traffic observed in computer networks?

To tackle the primary research question, we break it up into smaller sub-questions. First, we will evaluate the effect of Certificate Transparency on the host-level, without considering the subnetwork. Once a better understanding of the effect on a single host is established, we will move on to the effect of CT on the network level. With the analysis made on the host and network level in IPv4 and IPv6 address space, we can look for differences between the two address spaces. To summarize, we split the primary research question into the following three sub-questions:

- 1.1 How does Certificate Transparency affect scanning traffic seen by a domain host?
- 1.2 How does Certificate Transparency affect scanning traffic seen by the subnet of a domain host?
- 1.3 How does Certificate Transparency affect scanning traffic in the IPv4 address space compared to the IPv6 address space?

Next to the main research question and its sub-questions, we will further investigate how a standard domain configuration with a PTR record can also lead to domain name leakage. RFC 1912 [7] advises that for every A/AAAA record, there should be a matching PTR record. However, the PTR record leaks the associated domain. Published works can be found using PTR records as a data source. Fiebig et al. [16] studied the state of IPv6 using data gathered from PTR records. Hence, we will attempt to answer the additional research question:

- 2. How does the presence of a PTR record affect scanning traffic of a domain host?

By answering these research questions, we hope to present to you that CT introduces a new weakness in cyberspace. The intention of the research is by no means to condemn CT, but to raise awareness that introducing a new security measure can have security implications.

1.2. Contribution

- This thesis analyzes the effect of CT on scanning traffic at webservers, domain hosts and DNS servers. The effect is studied in both IPv4 and IPv6 address spaces.
- The experiments build upon the works of of Amann et al. [46]. We extend the CT honeypot model and the scope of analysis: we run experiments for over than 150 days and examine traffic recorded at 3 locations.
- We show how the effect of CT on scanning traffic differs depending on the address spaces, IPv4 and IPv6.
- An update on the state of CT in 2021 is given and is compared to the state in 2018 from the findings by Amann et al. [46].
- In the IPv6 address space, we examine how CT affects scanning traffic in a /64 subnet. We record all inbound packets at the unused addresses of the subnet and analyze it.

- We propose 2 filters for isolating CT-induced scanning traffic and show their effectiveness. The first filter uses control experiments to find sources that are not related to CT. The second filter corrects traffic on the AS level given a criterion.
- RFC 1912 [7] strongly recommends that for every A or AAAA record, there should be a matching PTR record. We show how a PTR record in a DNS configuration affects scanning traffic.

The outline of the remainder of the thesis is as follows: Chapter 2 will cover background information and related work on the subject; Chapter 3 will cover the setup of the hardware and software used for the experiments, the architecture of the experiment network, the technologies and infrastructures used and a detailed description of all the experiments performed; Chapter 4 will cover the data analysis of the data gathered during the experiments and derive from the data analysis answers to the research questions of the thesis; And last Chapter 5 will cover the conclusion, the limitations of the research and the possibilities of future work.

2

Background

In this chapter, we provide background information on the topics relevant to the research. We start by covering computer networks and explain how devices are communicating using the OSI model. We follow with an example of a computer network, the Internet, and show how it relates back to the OSI model. We explain how the addressing works by explaining IP addresses and Autonomous Systems (AS). Moving from networks to the Public key infrastructure (PKI), we explain the principal components using HTTPS as a concrete example. We give a high-level explanation of the cryptography in use and the X.509 trust model. We tie the principal components of PKI with Certificate Transparency (CT), the primary focus of this study. We finish with a description of the Domain Name System (DNS) and scanning methods.

2.1. Communication in computer networks

A computer network is a set of interconnected devices. There is a wide range of devices that can be part of a network: computers, network devices or any other smart devices. Devices are interconnected by physical links that allow to forward data. [12] We will explain the communication between devices, also known as hosts, using the OSI model.

2.1.1. OSI model

The Open Systems Interconnection (OSI) model is a layered framework developed in 1984 by the International Organization for Standardization (ISO) to standardize the exchange of data among systems. A system is a set of devices that form an entity capable of transmitting and receiving data. Examples are computer networks such as local area networks (LAN) or autonomous systems (AS), or even single computers. It is defined in ISO 7498. The OSI model abstract the communication process into a seven layer hierarchical model, where the upper layer relies on the functionality of the layer below. The collaboration of all layers allows to transmit data from one location to another. The OSI model does not specify services and protocols for the layers of the framework: it only serves as a standard to facilitate implementation and ensure interoperability between systems. [27]

Figure 2.1 shows how two hosts communicate using the OSI model. It shows how each layer between hosts is logically connected, indicated by dotted arrows. The actual transmission is done on the physical medium. The movement of data is depicted by full arrows. It shows how each layer is connected between the transmitter and the receiver. Each layer, except the physical layer, uses the layer below to create a logical link. The first 3 layers are responsible for moving packets from the transmitter to the receiver, possibly passing through intermediary nodes.

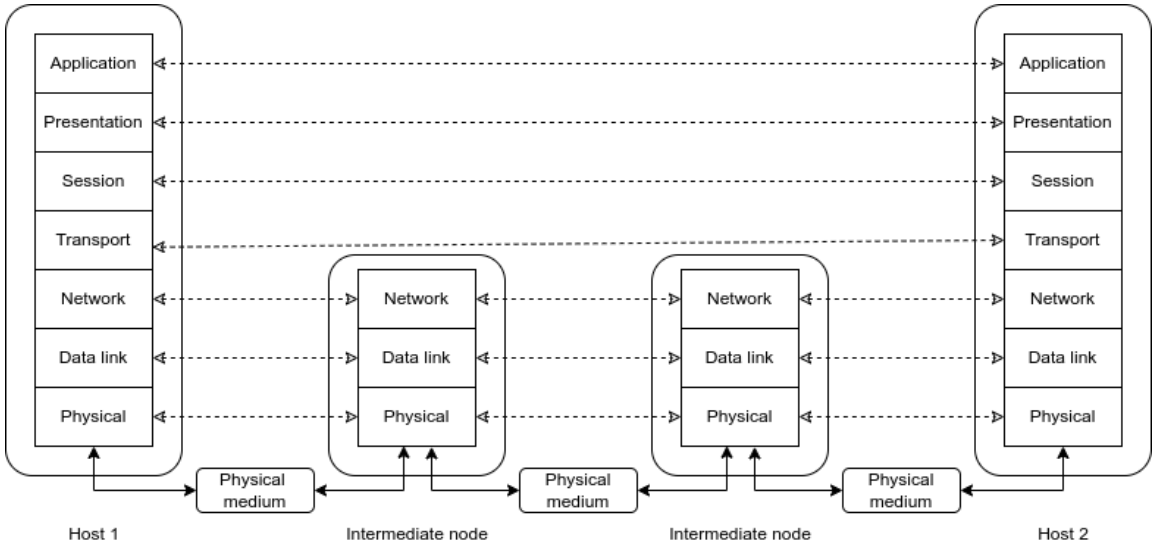

Figure 2.1: Diagram of communication in the OSI model. The diagram shows how each layer between hosts is logically connected, indicated by dotted arrows. The actual transmission is done on the physical medium. The movement of data is depicted by full arrows. It shows how each layer is connected between the transmitter and the receiver. Each layer, except the physical layer, uses the layer below to create a logical link. The first 3 layers are responsible for moving packets from the transmitter to the receiver, possibly passing through intermediary nodes. Based on [49].

Layer 1: Physical layer

The lowest layer of the model is the physical layer, responsible for the physical transmission and reception of raw data between two connected devices on a physical medium. It provides a physical connection and retains the order when delivering the bits of the data stream. The function of the layer is dependent on the transmission medium. [27]

There are two types of transmission media that are used to link two devices: wired links with electrical cables and fiberglass, and wireless links with open space. The physical layer lays out how information, in digital bits, is converted into a signal than can be transmitted on the transmission medium between two devices. Different types of transmission mediums warrant different conversions. For example, copper cables require data to be converted into electrical signals, while Wi-Fi requires data to be converted into radio waves. [12]

Layer 2: Data link layer

The second layer provides a communication channel between devices with a direct connection. We have seen that the lowest layer, the physical layer, handles the transmission of raw bit streams. The data link layer can detect errors and may correct them in the raw bit stream from the physical layer. [27] Whether the channel travels over a physical link or air does not matter, as long as the physical layer provides a way to transmit and receive data [12].

The two ends of a data link are addressed using an addressing scheme for the data-link layer, for example, MAC addresses. The data link layer also coordinates the access to the physical medium. It regulates when a device can transmit and at what rate, and when it has to listen. The management of the link is handled by the quality of service parameters. The communication channel can operate in connection or connection-less mode. The connection-mode adds further functionalities for handling connections. [27]

The data link layer groups data blocks into a logical structure called frames, adds the source and destination in the frame's header, and adds error control. The frame is transmitted to the destination specified in the frame using the physical layer. Reversely, it translates raw abstract bit stream received at the physical layer into frames and, checks for and potentially fixes transmission errors. The received frame is then relayed to the upper layer. [12]

The most common implementation of this layer is the Ethernet protocol. Devices are addressed using media access control address (MAC address), a unique identifier, 48 or 64 bit, assigned to a network interface controller (NIC) of a device by the manufacturer. [12]

Layer 3: Network layer

The network layer provides a communication channel between two devices in different networks. Devices can communicate without sharing a direct connection. The layer ensures that packets are forwarded and relayed to the right destinations. The layer can operate in connection or connection-less mode. The connection mode is an extension of the connection-less-mode by adding functionalities to deal with connections. [27]

As there can be multiple routes that can be taken to deliver information, the network layer may determine an appropriate path for packets from source to destinations. It adapts the transfer of packets to the requirements of the path: if the data is too large to send it as one packet on the data link layer; the data is split into multiple packets, which are reassembled at the destination. This is achieved by negotiating the quality of service between two ends of a network connection. Error detection is used to check whether the quality of service is maintained. [27]

An additional addressing scheme is used to identify hosts from other networks [27]. One of the most common network layer protocols, the Internet protocol (IP), uses IP addresses as network-addresses. The Internet uses the IP to assign IP addresses to hosts. The first generation of IP is IPv4, where 32-bit numbers are used as addresses. Because of the exponential growth of the Internet, the pool of available addresses is depleted. A new version of IP was developed using a tremendously larger address pool. The new version IPv6 uses 128 bit addresses, providing $7.9 * 10^{28}$ more addresses than the older IPv4. [12]

Layer 4: Transport layer

Using the host to host communication channel provided by the network layer, the transport layer connects the hosts with running processes forming a communication channel from process to process. A new addressing scheme is used to differentiate between the communication channels of the process within a host. The transport layer relies on the network layer for routing and relaying the communication channel between processes. [27]

Processes are addressed by ports, identified by a 16 bit number. Ports are virtual endpoints managed by the operating system. A process binds to a port and an IP address to attach its communication channel. Packets arriving at a port are forwarded to the bound process and outgoing packets from the process are transmitted. [12]

The transport layer can establish connection or connection-less communication channels. In connection mode, the layer is responsible for establishing, maintaining and terminating connections between two processes by the transport address, such as ports, and the network address. As the data sent by a process can be larger than the capacity of the communication channel, the transport layer in connection-mode segments data into smaller packets. At the receiving end, the transport layer reassembles the segments back to the original data. Similarly, the layer can block and concatenate data blocks that are smaller than the capacity of the channel. It provides a reliable connection by ensuring that packets are not lost and arrive in the correct order without errors at the destination using flow control mechanisms. In connectionless mode, it provides a mapping between two processes by the transport and network address. The connection-less mode can detect errors. [27]

A transport layer protocol that works with connections is the Transmission Control Protocol (TCP). TCP establishes a stateful connection between two devices using a three-way handshake. It ensures the retransmission of lost and corrupted packets, as well as the arrival of packets in order. The additional overhead of TCP introduces latency in the communication. A simpler protocol that omits flow control and error detection is the User Datagram Protocol (UDP). Unlike TCP, UDP provides a connection-less communication channel, where packets are delivered on a best-effort basis. Such a protocol is used where packet loss is acceptable, such as in video streaming. TCP and UDP comprise most of traffic on the Internet. [12]

Layer 5: Session layer

With the communication channel between processes in the lower layer, the session layer associates both processes into a session. A session is used to manage the dialogue between two processes and close the connection properly. It enables to maintain a connection for longer periods of time. [27] Sessions can be recovered even after a lost connection and resumed at the last state. The session layer is optional and often is merged with the Application layer alongside the presentation layer. The TCP/IP protocol suite used by the Internet is such an example where the three layers are merged. [12]

Layer 6: Presentation layer

Processes in a connection can use different syntaxes for the presentation of data. The presentation layer ensures that the transferred data is translated such that it can be interpreted by the receiver. The required syntax translation is negotiated between the parties of the communication channel. The syntax translation does not change the content of the communication. [27] For example, it translates the sent data into a character encoding scheme accepted by the receiver. The presentation layer is, like the session layer, optional. [12]

Layer 7: Application layer

The application layer is the last and highest layer of the OSI model. It directly interacts with the application processes to handle the communication with other applications. This layer is the only access an application has to access the functionalities of OSI. The functionalities offered in this layer are application specific. [27] An example of an application layer protocol is HTTP. It handles the conversation between the client and the webserver. The client's web browser uses HTTP to request web content. In return, the web server responds with the requested content using the same protocol. The application layer protocol forwards the response to the web browser, who parses the content of the response and displays the results to the user. The HTTP communication between the two parties abstracts over how the communication is transmitted from browser to webserver. It relies on the lower layer to provide that functionality. [12]

2.1.2. Internet

The Internet is a global network of networks spanning all over the world. It uses the TCP/IP protocol suite, a concrete implementation of the OSI model, to provide communication tools for devices. The creation of the Internet dates back to the 1960s by the Advanced Research Projects Agency (ARPA) in the United States. We will explain how the Internet is structured and managed, and devices on the Internet are assigned addresses. [12]

Addressing

As we have seen in the network layer of the OSI model, IP uses IP addresses for the endpoints of the communication. But due to the depletion of the IPv4 addresses, not every device can be assigned one or multiple unique IP address. To combat this lack of addresses, addresses can also be assigned to a network instead of devices. The network's link to the Internet is managed by a router, which carries the assigned addresses. Devices in the network share the same address to communicate with devices outside the network. The router assigns for each of the devices in the network an IP address from the reserved IP list for private use. It takes care to forward traffic to the right destination. An example of such a network is the Local Area Network (LAN), which is used by essentially all households. The Internet Service Provider (ISP) provides an IP address, permanently or temporarily, to a household where all the devices share the same address to access the Internet. [12]

Autonomous System (AS)

An Autonomous System (AS) is a group of networks operated by a single administration. ASes are assigned a unique Autonomous System number (ASN). Every AS has a separate and unified routing policy which specifies which addresses it controls and to what ASes it is connected. This information is used in the Border Gateway Protocol (BGP) for routing packets to the right AS. ASes are the networks which compose the Internet; the Internet is a network of ASes. [23]

Regional Internet registry (RIR)

Regional Internet registries (RIR) are responsible for allocating and registering IP addresses and ASNs. These organisations operate regionally. There are in total 5 RIRs: African Network Information Center (AFRINIC), American Registry for Internet Numbers (ARIN), Asia-Pacific Network Information Centre (APNIC), Latin America and Caribbean Network Information Centre (LACNIC) and Réseaux IP Européens Network Coordination Centre (RIPE NCC). The Internet Assigned Numbers Authority (IANA) provides IP addresses and ASNs to RIRs who allocate and register them to their customers, following their own policies. [26]

2.2. Public key infrastructure (PKI)

In this section, we explain the components that form the Public key infrastructure (PKI). We give a high-level overview of the cryptography in use and the key exchange process. We illustrate the explanations with the protocol HTTPS. We follow with the X.509 standard for the public key certificates.

2.2.1. Cryptography and key exchange

Public key cryptography or asymmetric encryption is a type of a cryptographic scheme with a public and private key. The public key can be distributed and does not need to be hidden. The private must be kept hidden and must not be shared. The public key is used to encrypt a message. As the public key is public, anyone can encrypt. But only the private key can decrypt the message, thus only the owner of the key can read the message. There are many cryptographic algorithms following this scheme. Notable algorithms are El Gamal and RSA. Public key cryptography is used to solve the problem of key distribution in symmetric key cryptography, where a single secret key is used for both encryption and decryption. The secret key needs to be shared with the two parties in the communication. But sending the key over a public communication link would defeat the purpose of encryption, as the key is not private anymore, as any eavesdropper on the communication link can retrieve the key. Instead, a public key scheme can be used to securely share a symmetric key between two parties. The communication can proceed with the new shared key. An example of a protocol is the Transport Layer Security (TLS), which uses asymmetric encryption to securely share a key between the client and the server. HTTPS, the secured version of HTTP, uses TLS. [48][42]

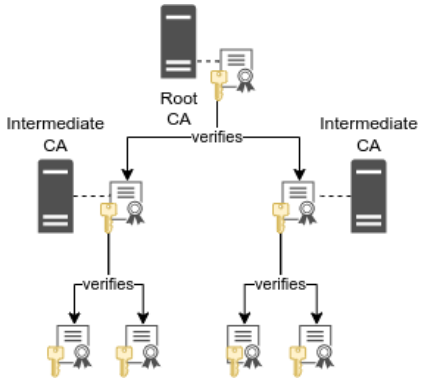

Figure 2.2: Example of a trust model tree, based on [17].

Figure 2.3: Diagram of a certificate chain. Users follow the certificate chain by verifying the signature with the public key of the upper level until they reach a trusting CA, based on [17].

2.2.2. X.509 Trust model

The X.509 standard was created by the International Telecommunication Union (ITU) to specify the standards for Public Key Infrastructure. One of the protocols that makes use of the X.509 standard is HTTP over TLS (HTTPS). The standard defines the structure of certificates. Using HTTPS as a concrete example, the certificate consists of a domain name, acting as the identity, a public key and the digital signature computed on the certificate. The content of the certificate cannot be modified other than by the signer, who owns the private key from the signing process: The domain name and the public key are bound by the signature. The certificate can be used to secure a communication path between the client and the domain if the issuer of the certificate, here called certificate authority (CA), is trusted by the client. When faced with a certificate, the client can check whether the signer of the certificate is trusted. The trust guarantees that the certificate was issued to the rightful domain owner. The client can then use the public key embedded in the certificate to establish a secure connection with the domain. [28]

When exposed to a certificate that is signed by an untrusted CA, the client can still establish a connection, but it might not be secure. Since the certificate is not trusted, the client cannot know whether the certificate was issued for the owner of the domain, i.e. that the public key is bound to the domain. The lack of trust does not prove that the client established a secure communication channel with the domain. [28]

For example, a client connects to the login page of his mailbox, when in reality, the page is a phishing page controlled by a malicious attacker. The client will try to establish a secure communication channel with the mailbox. Since the attacker is not the owner of the domain of the mailbox, he should not be able to request a certificate from a trusted CA. There are untrusted CAs that can issue the certificate, for example, the attacker himself. But when the client attempts to establish a secure communication channel, the client will be presented with a certificate from an untrusted CA. Ignoring the distrust, the client establishes a communication channel with the attacker instead of the mailbox. Such an attack is called a man-in-the-middle attack. Only by establishing connections with trusted certificates mitigates this attack.

The trust model is structured in a tree model consisting of two types of certificates. Figure 2.2 illustrates the trust model with the certificate types. Leaves are end-entity certificates and nodes are CA certificates. CA certificates are issued to CAs. They can be used to issue other certificates. The CA certificate at the root of the tree is also called Root CA certificate. The remaining are called intermediary

CA certificates. End-entity certificates are issued using CA certificates and bind to an identity. Unlike CA certificates, it cannot be used to sign certificates. To verify whether a certificate truly belongs to the owner of the domain, the certificate chain is checked. First, the signature of the certificate is verified using the public key of the issuing CA. If the CA is not trusted, then the signature of the CA certificate is verified with the certificate of the issuing CA. This chain is followed until a trusted CA is encountered or the root of the tree is reached. If no trusted CA is found, then the certificate is not trusted. In case that a certificate becomes invalid, a CA can revoke it. The X.509 standard specifies how CAs can revoke issued certificates. Revoked certificates are made public in certificate revocation lists. Before establishing a secure connection using a certificate, a client needs to check the revocation list to determine if the certificate can be trusted. [28]

2.3. Certificate Transparency

In 2011, several Certificate Authorities (CA) were compromised and issued rogue certificates. These certificates allowed for man-in-the-middle-attacks (MITM), where an attacker can eavesdrop on a seemingly secure communication. As a countermeasure to such attacks, Certificate Transparency (CT) was created in order to provide transparency and verifiability of certificates in the Web Public Key infrastructure (PKI). The goal of CT is to make PKI, the underlying technology on which HTTPS is built upon, more robust to provide a safer cyberspace.

Before the deployment of CT, HTTPS solely relied on a list of trusted CAs, determined by the browser or the manufacturer. CAs were audited and their track records were analysed to decide whether they were trustworthy. However, trusted CAs could still issue rogue certificates – for various reasons – exposing Internet users to MITM attacks. There was no technology that could allow to verify the legitimacy of a certificate, i.e. whether it was issued for the rightful domain owner. Even after being detecting, rogue certificates would remain in use until the issuing CA would retract it. CT removes this reliance on CA by providing a public mean to verify certificates.

Issued certificates are appended in taper-evident logs, called CT logs. CT logs are independently operated and can be verified cryptographically by any parties. Anyone can monitor issued certificates. Misbehaving CAs are easily detected by monitoring CT logs for rogue certificates, which result from compromised CAs or malpractices. CT logs hold CAs accountable for their actions. As of Mar 7, 2022, 6195762645 certificates were logged in CT logs. [30] [56] [55]

2.3.1. CT log

CT logs are append-only and taper-evident ledgers of certificates. Added certificates cannot be removed or modified without detection. CT logs are operated independently so that no single organization has complete control of the certificate ledger. The cryptographic mechanism used by a CT log is the Merkle hash tree, whose properties facilitate audits, monitoring and insertion of certificates. [30]

Merkle hash tree

A Merkle hash tree is a binary tree, where leaves are logged certificates and nodes are hashes of the child nodes or the leaves. The root node of the tree is signed with the private key of the CT log. The signed root node is called signed tree head (STH). To append one or more certificates, they are combined into a separate Merkle tree. The new Merkle tree is merged with the current Merkle Tree by taking the root nodes and computing their hash, which becomes the new root node with the old root nodes becoming child nodes. The CT log signs the new root node creating a new STH. Therefore, any appending of certificates changes the STH. Changing a logged certificate also changes the STH, as the change trickles up the tree through the hashes. Any modifications to the Merkle tree result in a different STH, therefore making it tamper-proof. [30][56]

Merkle Consistency Proofs

Consistency proofs allow to verify the consistency between two views of the Merkle Tree from different dates. The more recent tree must include all the certificates in the same order than in the older view. Certificates that are in the new view are not present in the old view. Satisfying those 2 requirements proves that the Merkle tree is consistent with the two views. The proof does not require checking all the certificates and hashes of the two views. The two first-level nodes of the old Merkle tree can be recovered from the new view. Computing the hash of these two nodes yields the root hash of the old tree. Taking the hash values of the subtrees that both have in common and the subtrees with the new certificates, the root hash of the more recent view can be computed. The Merkle tree is consistent if the computed hashes correspond each to the trees and the SCT signature is valid to the log's public key. The proof is to verify whether CT logs are behaving properly. Practically, an auditor only requires storing the STH of a view. They can request a consistency proof from a CT log, who will return the necessary hashes. The auditor computes the hashes of the view it owns and the new view of the Merkle tree. [30][56]

Merkle Audit Proofs

Audit proofs allow to check whether a certificate is present in a CT log. The auditor requires all the missing nodes required to compute the root hash. With all the nodes and hash of the certificate in question, he can compute the root hash and compare it to the value advertized by the CT log. If both match and the signature of the STH is valid, then the certificate is logged in CT. To retrieve all the missing nodes of the auditor, the CT log starts from bottom of the tree at the requested certificate. Since the auditor is in possession of the requested certificate, its hash does not need to be sent. To compute the parent node, the sibling of the known node is needed. In the same way, to compute the parent of the parent, the sibling node is needed. Moving up the tree until the root, we can find all the nodes that are needed for the verification. The number of nodes needed is bound to log(height of tree). [30][56]

2.3.2. Clients

There are multiple types of clients interacting with CT [30]

- **Submitters** are entities that submit certificates or precertificates to one or multiple CT logs. In response, CT logs return signed certificate timestamps (SCT).
- **TLS clients** are not directly interacting with the CT infrastructure. When connecting to a domain, they are presented with the trusted certificate and the SCTs. The usual procedure is taken for the certificate by verifying the chain of certificate. For the SCT, the signature is verified with the public key of the responsible CT log.
- **Monitors** verify that CT logs are behaving correctly. This is achieved with Merkle Consistency Proofs. They can also fetch all the certificates from a CT log and recompute the entire tree to verify the root hash.
- **Auditors** provide audits to TLS clients. Using Merkle audit proofs, they can check whether a certificate is included in CT log. Auditing can be separate entities but can also be part of a monitors function.

2.3.3. Certificate Transparency in TLS ecosystem

In this section, we show how CT is integrated into the TLS ecosystem. Figure 2.4 shows all the steps, from certificate issuance to certificate logging and auditing. The domain owner sets up a domain and (1) requests a certificate from a CA. The CA generates a precertificate and (2) sends it to one or more CT logs. The CT log promises to add the certificate within the maximum merge delay (MMD)

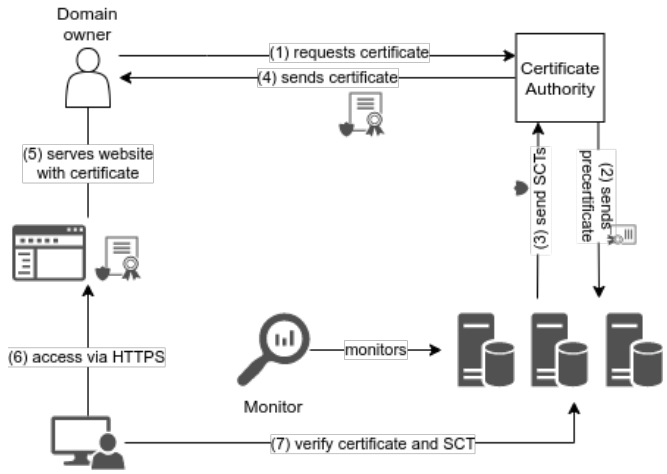

Figure 2.4: Diagram of the interactions of Certificate Transparency, based on [56]

specified in the SCT, which is (4) sent back to the CA. The CA uses the precertificate and the SCTs to issue the final certificate and (4) gives it to the domain owner. The CT log also adds the final certificate to the CA. The reason for sending a precertificate instead of the certificate is because of a deadlock: the CA needs to embed the SCT in the certificate, but only logged certificate can get a SCT. The domain owner (5) can use the new certificate for his website running under the certified domain name. Clients (6) connect to the website via HTTPS. During the handshake to establish the connection, they are presented with the certificate and the SCTs of the website. They (7) verify the certificate by following the certificate chain and the SCTs with the public key of the CT logs. Depending on the browser, CT logs can be audited to verify that the presented certificate is logged. After the verification, the handshake is finalized and the user can access the contents of the website. In parallel, monitors are verifying that CT logs are behaving correctly. There are different ways to integrate CT in TLS, but Figure 2.4 shows by far the most common method. [56]

2.4. Transport Layer Security (TLS)

The Transport Layer Security protocol is situated between the transport and the application layer in the OSI model [12]. The protocol ensures confidentiality by encrypting the application data with a secret key shared by the two ends of the channel. Both parties can communicate on the secure channel. The communication channel is duplex, allowing both parties of the communication to send and receive messages. The integrity of the application data is guaranteed with a message authentication code (MAC) that is computed for each data packet sent. Authentication is obligatory for at least one of the parties in the communication, most often for the server. It is achieved by providing a valid and trusted certificate to the other party. [42] The MAC is computed before encrypting it with the message; Inverting the order of authentication and encryption has severe security implications [12].

A TLS connection first needs to be established between two parties before any data can be sent. The negotiation of the connection is composed of three subprotocols forming the handshake protocol. The handshake is initiated by the client. The client shares a list of encryption, authentication and compression algorithm and TLS versions supported, a session ID, a timestamp and a random nonce. This first message is called Client Hello. The server responds with a Server Hello and shares a random nonce, a session ID and chooses for each algorithm and protocol a supported version of the client. The random nonce and the timestamp help to prevent replay attacks, where an eavesdropper replays the messages in a handshake. In this phase, the client and the server set up random numbers and agree on a common set of algorithms for the connections. Next, the server authenticates himself by sending the server certificate. If the client accepts the certificate as valid authentication, the handshake can continue. Optionally, the client can also authenticate to the server. The following step is the key exchange for encryption and authentication. The key exchange protocol depends on what was agreed

on in the first phase. There are 3 types of key exchanges: Diffie-Hellman, RSA and Fortezza KEA. The certificate can send parameters for the key generation using the key exchange protocol. The parameters are signed to ensure authenticity. Now both parties have all the necessary information to derive the master key of the TLS session. For the key exchange protocol, a premaster key was negotiated to share the parameters of the master key. The reason the initial key is not used is that of the weakness of the key. The key exchange protocol relies on asymmetric encryption, which is inherently weaker than its counterpart symmetric encryption since some information is leaked through the public key, whereas the key in symmetric encryption is kept private. It is only used for a short time to establish a master key for symmetric encryption. [42]

The master key is compromised of 2 types of keys: for confidentiality and integrity. The exact contents of the master key depend on the chosen algorithms for encryption and integrity. The master key contains multiple key blocks used by the client or the server for encryption and MAC computing. Using the master key, application data can be sent securely through the communication channel. First, the data is fragmented into blocks of at most 2^{14} bytes. If required by the client, the data is compressed before the MAC is computed. The data, the MAC and padding are encrypted, and a header with the sequence number and the length of the data block is added. The message is passed down the transport layer to send it. Upon receiving the message, the receiver reverses the process. He extracts the payload from the packet and decrypts it, removes the padding and verifies the MAC using the master key. If the verification fails, the TLS session is aborted. As TLS builds upon TCP to provide a reliable connection, any detected failures could indicate an ongoing attack. [42]

2.5. DNS

The Domain Name System (DNS) is a distributed database that translates domain names to physical IP addresses. It is easier to remember a name than it is to remember an IP address. DNS allows to easily lookup the address of a host using a name, like a telephone book. The client can look up a name by making a DNS query to translate the name to an address. This system simplifies the navigation on the Internet. [4]

2.5.1. Domain names

Domain names are structured in a hierarchal tree structure in DNS. The domain name consists of the assigned label, concatenated with a '.' and the label of the parent node on the right recursively until the root node. For example, take mail.example.com, test.example.com and website.nl. Each string, separated by '.', is a node in the tree. The label to the right of a label is the parent node. The root domain is '.', which is omitted in domain names, encompasses all three example domains. Below are the top-level domains. In this example, we have .nl and .com. .nl only has one domain and .com two. Within the .com domain, we have one second-level domain example.com, which has two domains with labels test and mail. [35]

2.5.2. Name server

DNS is a distributed database of name servers (NS) or DNS servers. There are two types: authoritative Name servers and DNS resolvers. An authoritative NS is responsible for delegated domain names. It stores the DNS records of the domain and responds to queries. It can also delegate the translation of part of the domain to other NSes. Queries for delegated domains will either be redirected to the responsible NS or the address of the NS will be returned. Each domain needs to be logged in at least one authoritative NS. DNS resolvers have a different function, they help clients to resolve a DNS query by returning the response if they have already stored it in their cache, by pointing them towards an NS that might carry the desired information (non-recursive) or by performing all the intermediary queries until the desired information is found and returned to the client (recursive). [4]

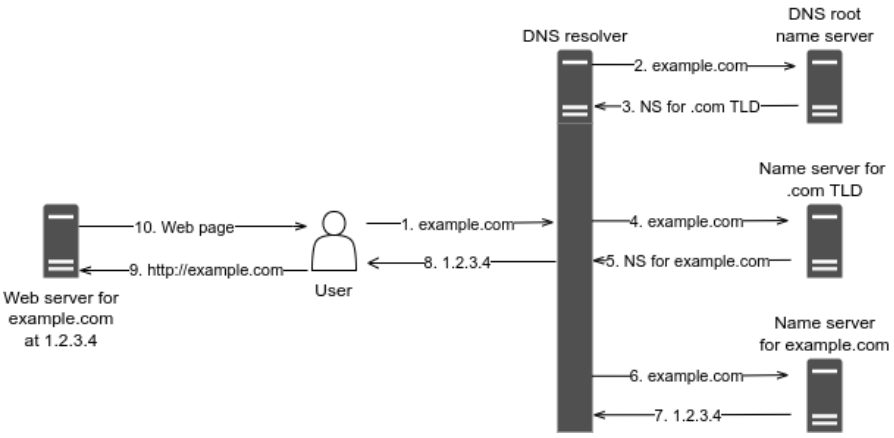

Figure 2.5: Example of a recursive DNS query resolved by a DNS resolver. The user makes a DNS request for www.example.com at the DNS resolver (1)(2). From (3) to (5) the DNS resolver starts by contacting the top-level NS and gets routed to lower level NS until reaching the authoritative NS for example.com. In (6) the address of the requested website is returned from the NS server to the resolver, who relays the information back to the user (8). With the IP address, the user can access the site (8)(9). Based on [4].

2.5.3. Address resolution

A domain name is resolved by starting at the top-level domain, the right most label in the domain name. A query for the domain name is sent to the server responsible for the top-level domain name. If the query is stored at the NS, it will return it to the querier, else it returns the NS of the one level lower domain. This cycle continues until the authoritative name server of the domain name is found and the answer to the query returned. [25]

Figure 2.5 is an example of a recursive DNS query resolved by a DNS resolver. The user makes a DNS request for www.example.com at the DNS resolver (1)(2). From (3) to (5) the DNS resolver starts by contacting the top-level NS and gets routed to lower level NS until reaching the authoritative NS for example.com. In (6) the address of the requested website is returned from the NS server to the resolver, who relays the information back to the user (8). With the IP address, the user can access the site (8)(9). [4]

2.5.4. Necessary DNS records for a domain

The Name Server (NS) record is necessary for a domain to indicate which authoritative server is responsible for the domain. The record holds the name of the NS, which can be translated to a physical address. It indicates to the querier where the NS of the desired domain name can be found. [35]

Every domain name is part of a DNS zone, which can include the domain and also its subdomains. Each DNS zone is managed by a specific entity. Management protocols can differ from one zone to another. Each zone requires a start of authority' (SOA) record which contains the information of the zone and the managing entity. [35]

The address of the domain is stored in A or AAAA records, that map the domain name with its IP address. The A record stores an IPv4 address while the AAAA record stores an IPv6 address. Without knowing in which address space the domain is hosted, both A and AAAA record might need to be queried.

RFC 1912 requires that every IP address has a pointer record (PTR) [7]. They map an IP address to domain names. They are used in reverse lookups to retrieve which domains point to an IP address. PTR can store both IPv4 and IPv6 addresses. IP addresses are not stored as is in the record, but reversed with either ".in-addr.arpa" or ".ip6.arpa" added, for IPv4 and IPv6 addresses, respectively. [7]

2.5.5. DNS security mechanisms

We briefly explain two security features in DNS: DNSSEC and 0x20 bit encoding. The Domain Name System Security Extensions (DNSSEC) provides authentication for the responses to queries. It protects queriers from spoofing and poisoning attacks. It ensures that the response comes from the expected NS and not from a malicious actor who is impersonating the NS to feed false information. The drawback of DNSSEC is the added overhead and, therefore, latency. The header of the DNS protocol has a flag that can be set to indicate whether DNSSEC is supported. The 0x20 bit encoding adds entropy to the query to make poisoning attacks more difficult. The encoding lies in the domain name, where the letter case is altered according to the encoding value. Since domain names are case insensitive, the encoding does not corrupt it. The querier ignores responses where the encoding of the domain name does not match the encoding that was sent. Attackers who want to send a malicious response to the querier will need to spoof the contacted NS, send the crafted response to the querier before the NS and guess the encoding that was used. [44][59]

2.6. Scanning

Scanning is a powerful tool for gathering information on a target. This section will focus on the scanning of hosts. The results of a host scan can reveal details about the device name, operating system, running applications, etc. The information gathered from the target scan can be used in various ways, for benign activities such as troubleshooting, but also for malicious activities, such as reconnaissance, to develop an attack. Nmap is a popular tool that automates scanning of hosts and networks. We will explain some common scan types that are used in the wild.

The simplest form of a scan is host discovery. Using protocols such as ICMP, we can check whether running hosts are listening on the targeted IP addresses. For example, an ICMP echo request packet is sent to an IP address. If an ICMP reply is received, then there is a host behind the address. Such types of scanning occur in the network layer. [34]

Another type of scan is the port scan. The port scan is used to probe for open ports on a host. The port number can show what type of service is listening on it. There are three types of ports scans. Scanning multiple ports of a single IP address is called a vertical scan. This scan is used to analyse a single host for vulnerable services. A horizontal scan scans multiple hosts on the same port. Such a scan is employed when looking for a specific vulnerable service. Block scans are hybrid scans that share properties of both vertical and horizontal scans. [31]

Scanning ports exploit that hosts comply with RFC 793: ports, open or closed, should always return a specific reply, which reveals the state of the port. Ports that are closed will return an ICMP port unreachable message. Open ports will return a response depending on the TCP packet sent. [41] The most basic TCP scan is a SYN scan, which sends a TCP packet with the SYN flag set to establish a connection with the process listening on the target port. An open port will respond with a SYN-ACK packet to acknowledge the connection process. This packet has revealed to the scanner that the port is active. [34] However, the connection is never finalized and therefore wastes resources of the target, since it needs to keep track of the connection. Excessive scanning of the target can overwhelm its resources, essentially turning the scan into a Denial of Service (DoS) attack, called SYN-attack [12]. Such a scanning procedure can alert the target of an ongoing scan, making it block any traffic coming from the scanner. To not exhaust the resources of the target, we can make use of the half-open SYN scan, where the connection is broken by the scanner with a RST packet, thus freeing the resources assigned for this connection. Other types of TCP scan exist using less common combinations of TCP flags to avoid detection, stealth scans. Though the benefits is avoiding detection and potentially circumventing firewalls, there are some caveats to such scans as there are dependent on the code. The response to such scans can be unpredictable, as these flag combinations have not a specified response. Examples are FIN, XMAS and NULL scan. [34]

A service scan is used to determine the service running on a port. It works by sending packets with data specific to an application protocol and check for the response. A valid response indicates that the guess was correct. For example, send an HTTP GET request to the target port. If the target responds with a valid HTTP response, then it is revealed that a web server is running behind the target port. Investigating the response might even indicate what type of software is running the web server. [36]

2.7. Related work

In this chapter, we explore research studies around CT and methodologies used in our paper. We present the findings from published research on the state of CT, on leveraging CT logs as a data source, on cyberattacks involving CT and on using CT honeypots.

2.7.1. State of CT

After the introduction of CT, studies have been done on the deployment rate and effectiveness. Amann et al. [2] monitors traffic on an uplink and cross-matches the measurements with data from CT logs to evaluate the adoption of CT logs. Furthermore, they perform active measurements on domains gathered from their experiment to check if the certificates of these domains are valid. The goal is to see whether any improvements have been made in terms of security in the TLS ecosystem since the incident involving DigiNotar in 2011. Their findings show the adoption rate of CT in 2017, where they find that popular domains have more of a tendency to adopt CT. From domains in the Alexa top 1K dataset, 38.57 % of the domains have at least one SCT. Expanding the dataset to the Alexa top 1M, the adoption rate drops to 19.48 %. At their vantage point, less than 7.5 % of the TLS connections include an SCT. The SCT tends to be embedded in the certificate as opposed to shared in the TLS extension. They find that 67.16 % of certificates with embedded SCTs come from one CA: Symantec. They speculate that the CA was forced to adopt CT because of previous incidents where rogue certificates were issued. Out of all the detected certificates, they only find one where the SCTs are invalid. It was due because of a rare edge case in the implementation of SCT issuance. They investigate the inclusion status of SCTs and find that every encountered SCT is also logged in CT. [2]

Scheitle et al. [46] investigate the adoption rate of CT by monitoring TLS connections at an uplink for one year. They inspect connections to see whether SCTs are sent during the TLS handshake. The SCT is the promise of a CT log that a certificate will be logged in due time. Hence, a TLS connection shows that the sender of the SCT has adopted CT. The SCT can occur at 3 locations: it can be embedded in the certificate, it is sent via TLS extension or OCSP reply. They find that in 32.61 % of the TLS connections, at least one SCT could be found. The most common way to share SCTs was by embedding in the certificate and accounted for 65.62 % of the TLS connections with SCT. Scheitle et al. [46] further examine the adoption rate of CT at servers by performing active scans to servers via TLS and verifying whether a SCT is shared by servers during the TLS handshake protocol. 68.7 % of the 42.8M encountered certificates had an embedded SCT. Looking at the CT logs that issued the SCTs, they find a disparity between the most frequent CT logs found during the passive measurements at the uplink and the active scans of servers. They presume that the disparity is because of the popularity of some services. Out of all the collected certificates, they find 16 certificates with an invalid SCT. They attribute this mistake by CAs to be because of rare edge cases and because of the early adaptation period to CT. [46]

Gasser et al. [18] investigate whether certificates from CT logs comply with the Baseline Requirements (BR). They find that in 2017, 1.3 % of the certificates violated the BR. The BR can be categorized in 4 requirements: identity, signature, key and validity time. They find that violations of all requirements, except the signature requirements, have decreased since 2014 to 2017. Signature violations have been the predominant cause of violations in 2017. The study of Gasser et al. [18] further investigates if certificates in CT logs are consistent with certificates found in the wild. 85 % of the detected certificates from active scans were logged in a CT log and 99 % of the detected certificates were consistent with the logged certificate in CT. [18]

Gustafsson et al. [21] analyses the properties of CT logs and the logged certificates. They examine 11 CT logs and check the maximum merge delay, the update interval (UI) and the time to publish (TTP). They find sizeable differences between the CT logs in the TTP and UI. They can range from less than 1 minute to 12 hours. They use CT logs to investigate properties of certificates as for example, find certificates that use weak hashing algorithms such as SHA1. [21]

2.7.2. Leverage CT logs as a data source

CT logs provide an easy way to retrieve issued certificates. Conventional methods require collecting certificates from each domain individually or retrieve datasets from domain threat intelligence. With CT, certificates can be downloaded in bulk directly from CT logs. Gasser et al. [18] use CT-logged certificates to check whether these certificates adhere to the baseline requirements. Aertsen et al. [1] use CT-logged certificates to determine how LE democratized encryption. Scheitle et al. [46] use CT-logged certificates to investigate on the deployment rate of CT. Gustafsson et al. [21] used CT-logged certificates to analyse the properties of certificates. CT logs are not an uncommon source for researchers.

CT logs are not only useful for retrieving certificates, but can also be used to compile a hitlist of domain names. Gasser et al. [18] use CT logs to expand their hitlist of domain names by extracting the domain names embedded in the CT-logged certificates. Using CT logs increased their hitlist by 58.7 %. VanderSloot et al. [57] find even more impressive results. They investigate how different data sources of certificates accurately represent the certificate ecosystem. Their findings show certificates retrieved from CT logs covered 90.5 % of certificates from seven certificate data sources combined. Scheitle et al. [46] used domains from certificates in CT log to create a hitlist of subdomains. Popular subdomain labels were concatenated with the domains embedded in the certificates. They discover 18.8M new and active subdomains.

2.7.3. CT in cyberattacks

CT at its core is a security measure to improve the security in the cyberspace. However, public ledgers, or CT logs, on which CT relies, have been criticized for privacy reasons [22]. Domains are indiscriminately exposed by the CT-logged certificate. This especially poses a problem for sensitive domains that point, for example, to sensitive management interfaces. Solutions to this problem are not adding the certificate of the sensitive domain to a CT log. However, popular browsers such as Chrome require certificates to be logged in trusted CT logs to work as intended [51]. Hall [22] proposes a new feature to CT that allows to obfuscate the subdomain label in order to not partially hide the sensitive domain. The CT log named Deneb and operated by Symantec implements a similar idea by truncating subdomain labels. However, it is not trusted by the popular browser Chrome. The exploitation of CT logs for discovering domains is not only theoretical.

Scheitle et al. [46] have shown that CT logs are being monitored and used for discovering domains to scan. The domain name can be extracted from the Common Name and Subjective Alternative Name fields. The domains leaked can reveal which subdomains exist and reveal which services are running behind it. The most common subdomain label is www, which accounts for 95 % of all labels. Interesting subdomains in the top 10 were webdisk, cpanel and whm, which likely point to management interface and could be attractive targets for malicious attacks. Not only can the domain found in the certificate be scanned, but it can also be used in subdomain enumeration where the domain name, extracted from CT, is concatenated with common subdomain labels and scanned. Subdomains used for running sensitive services might not be logged in CT to avoid their discovery. The goal of subdomain enumeration is to scout for these sensitive domains for further attacks. Using popular tools for enumeration, they discover 18.8M new and active subdomains, from which only 1.1 M was known by the Sonar database from April 27, 2018. The construction of new domain names from CT has shown to be effective in drastically extending existing databases.

Scheitle et al. [46] examined how effective CT is in combating phishing domains. They look for domains in CT logs that mimic domains from Apple, PayPal, Microsoft, Google and eBay. In total, 117K phishing domains were found mimicking these popular services. They believe that CT logs are a promising countermeasure to phishing campaigns.

CT logs can also act maliciously: they can show different but valid views of the logs to different users. This type of attack is classified as a partition attack. Gossiping protocols have been designed as a countermeasure against partition attack [37]. In the proposed gossiping protocols, CT logs share information about their state. Seemingly effective against partition attacks, gossiping protocols raise the question of privacy of the shared information [37].

2.7.4. CT honeypot

Scheitle et al. [46] introduces the notion of CT honeypot to catch third parties that monitor CT logs to gather knowledge about potential new targets. They try to determine how the information in CT is used. The CT honeypot creates a random domain name that acts as a unique and hard-to-guess identifier called honeytoken. It requests a certificate from a trusted CA for the domain and logs the certificate in CT logs, where the domain name, embedded in the certificate, will be leaked. The attacker model assumes that there are third parties that monitor CT logs and collect domain names from the logged certificates. The honeypot monitors requests at the authoritative DNS server for A or AAAA requests for generated domain name honeytoken. If the honeytoken is detected, it shows that CT logs are being monitored and the information in the certificates is used. Their findings show that it takes 73 seconds to 3 minutes for an A/AAAA request to appear at the DNS server after the domain name was leaked in CT. They identify 6 ASes that made DNS requests for multiple of their honeytokens and 4 machines that attempted to connect to the honeypot over IPv4. Honeypots running on IPv6 detected no inbound packets. [46]

Our research builds upon the works of Scheitle et al. and their honeypot. This method of can be classified as inverse surveillance [43] to detect third parties monitoring CT logs. Monitors are being monitored.

3

Methodology

In this chapter, we cover our methodology for evaluating the effect of CT on scanning traffic. First, we give an overview of the methodology, describe the interaction with third parties of the methodology and define an attacker model. Next, we break down the methodology into two components – passive and active – and explain the inner workings of the components, followed by a description of external data sources used in the data analysis. Then, we cover the experiments run within the methodology, give their description and purpose. We end the section with a discussion on the limitations and on the ethics of our methodology.

3.1. Overview

Figure 3.1 shows how attackers can exploit CT logs to perform targeted scanning. We are assuming the following attacker model: an attacker monitors CT logs for new certificates, (6) from which he extracts the embedded domain name. With the domain name, (7)(8) he queries the IP address of the domain name via a DNS query. (9) He specifically targets the discovered IP address for scanning. Amann et al. [46] proposes a similar attacker model.

Following the scanning strategy of our attacker model has its advantages over randomly choosing addresses to scan. When requesting a certificate for a domain name, the necessary DNS entries need to be available and the domain name must be hosted on the IP address specified in the DNS entry. Thus, any IP address discovered via CT logs must have been active when the certificate was requested. Therefore, when targeting those IP addresses, they are likely to still be active, giving an advantage against randomly targeting IP addresses, for which it is unknown whether they have been active in the past. The scanning strategy allows for more efficient scanning results, as less time is spent on scanning inactive addresses. The scanning strategy also provides an advantage over scanning the entire address space: scanning the entire address space is time-consuming, because time is wasted on inactive addresses. These advantages are amplified in the IPv6 address space due to its sheer size and sparsity of active addresses.

To measure the effect of the attacker model – how CT logs affect scanning traffic on computer networks – we split our methodology in an active and a passive component. The purpose of the active component is to disseminate domain names via CT logs, so that they can be leaked to an attacker. The passive component is responsible for measuring scanning traffic – traffic stemming from attackers monitoring CT logs – and storing the data. Both components form a CT honeypot, according to the definition given by Amann et al. [46]. The domain names act as honeytokens – coined in 2003 [52] – which are unique identifiers intended to lure attackers into scanning the virtual hosts who act as hon-

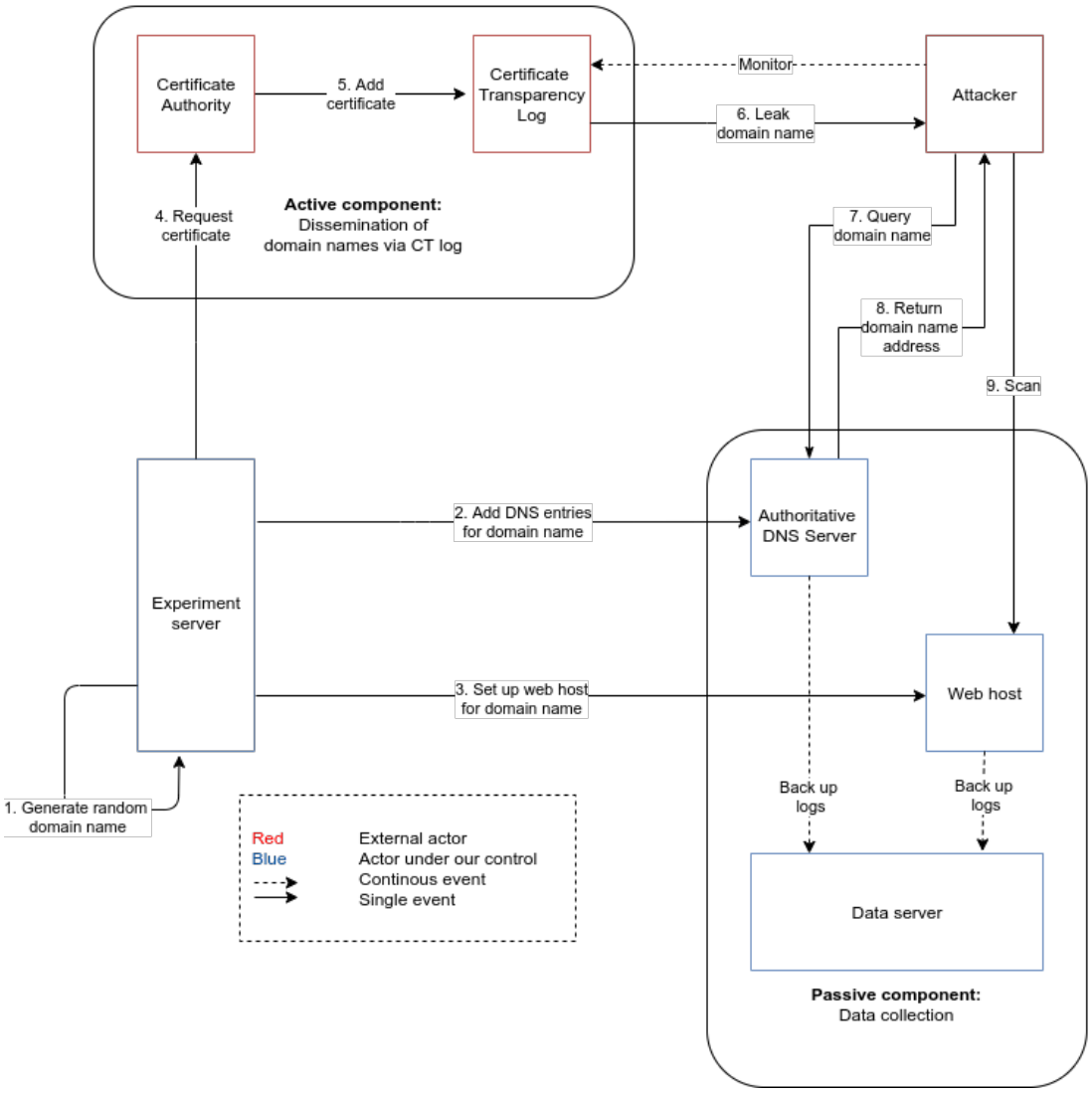

Figure 3.1: Overview of the interaction between the attacker with the active and passive component

eypots. Figure 3.1 shows the interaction between the active and passive component with the attacker model.

3.2. Active component: Domain name dissemination

The active component is responsible for disseminating domain names via CT log. Given the attacker model defined in Section 3.1, we need to add domain names to CT logs. Domain names cannot be directly added to CT logs, but instead indirectly through certificates in which the domain names are embedded. The domain names are randomly generated to ensure that they cannot be guessed. In our methodology, we do not add certificates manually to CT logs; in Figure 3.1, (4) the CA, in this case, Let's Encrypt (LE) [13], issues a certificate and (5) automatically adds it to CT logs [14]. As a result, the domain name is publicly accessible through a CT log and thus can act as a honey token.

3.2.1. Domain name generation

Domain names for dissemination are randomly generated to ensure that they cannot be guessed. The domain names are generated using the content of /dev/urandom file. The content of the file is converted to a string of lowercase letters and numbers of length 13. /dev/urandom is a file on UNIX-like operating systems that acts as a pseudo-random number generator. The PRNG generates numbers using the environmental noise of the hardware of the machine [33]. The full domain name is the concatenation of the randomly generated string and our DNS zone "example.com", for example, abcdefghijklm.example.com. The randomly generated strings act as a honeytokens in our experiments.

3.2.2. Experiment server specification

Technical specifications	
OS	Ubuntu 18.04.5 LTS
CPU clock speed	2.3 GHz
CPU cache	4 MB
CPU core count	1
RAM	2 GB
Storage	120 GB

Table 3.1: Technical specifications of the experiment server

Network configuration	
Dedicated IPv4 address	203.0.113.0
Dedicated IPv6 address	2001:DB8::
Tunnelled IPv4 networks	192.0.2.0/24
	198.51.100.0/24
Tunnelled IPv6 networks	2001:DB8:0:1::/64
	2001:DB8:1::/48

Table 3.2: Network specifications of the experiment server. The actual IP prefixes were replaced by documentation prefixes.

Table 3.1 holds the technical specification of the experiment server. The server is configured in a dual-stack network configuration, with one IPv4 and one IPv6 address. These addresses are used each as an IP tunnel endpoint. Two IP tunnels are connected to the experiment server to provide more addresses for the experiments. The experiment server has access to an additional two /24 IPv4 networks, one /64 and one /48 IPv6 networks. An overview of the available addresses of the server is given in Table 3.2.

3.3. Passive component: Data collection

The passive component is responsible for providing a target for attackers to scan and collect the incoming traffic. There are 2 types of hosts in the passive component: the Authoritative DNS server responsible for the generated domain names and virtual hosts. After (1) generating a domain name, the necessary [7] DNS entries for the domain name are added to the DNS server and (2) the virtual host of the domain name is set up. Any queries received by the DNS server and all incoming traffic to the virtual host is recorded and backed up into the data server for later analysis, hence passive measurement. Table 3.3 shows an overview of the collected data.

Data source	Time period	Entries	Size
DNS	Mar. 3 - Sep. 5, 2021	3.8 M	525.9 MB
PCAP files	Mar. 3 - Sep. 5, 2021	14.5 G	1.6 TB
Webserver	Mar. 3 - Sep. 5, 2021	2.1 M	1.8 GB

Table 3.3: Overview of the internal data sources

Given the defined attacker model, after discovering a new domain name via a CT log, he needs to retrieve the address of the domain name to be able to scan the host. The (8) IP address is retrieved via an (7) A or AAAA DNS query. The domain name is only discoverable in a CT log or in the PTR record present in the DNS server. Thus, any A or AAAA DNS query with one of the generated domain names implies that either a CT log or the PTR record leaked the domain name to the attacker.

3.3.1. Authoritative DNS server

By logging DNS queries of the generated domains of our research, we can gain valuable insight into the actors that try to gather additional information on the domain names leaked via CT. Not only is the source of the query logged but also the type of the query, time of the query and additional flags from the DNS packet. Since we control the authoritative DNS server responsible for the domain zone of the generated domains, we can log all DNS queries made to our domain. The authoritative DNS server runs with PowerDns 4.1.1. All the DNS records are stored in a private MySQL database.

3.3.2. Web server

Analysing HTTP traffic can reveal the behaviour and identity of attackers. For example, we can detect whether attackers are trying to compromise the virtual host. Therefore, all HTTP(S) requests to the virtual hosts are recorded. Since the website is only accessible via its domain name as specified in Section 3.2.1, the domain name must have been leaked and collected by the attacker.

The experimental server is running an Nginx web server (version 1.14.0). A virtual host is created for every generated domain name. Using virtual hosts allows for different network configurations and HTTP configurations per domain name. The same HTML page is published by all virtual hosts. Figure 3.2 shows the configuration of the webserver with its virtual hosts in the experiment server.

3.3.3. Server

On the experiment server, all packets, incoming and outgoing, are captured using tcpdump. SSH service is set up listening on every IP address in use by the webserver. All SSH connection attempts are recorded.

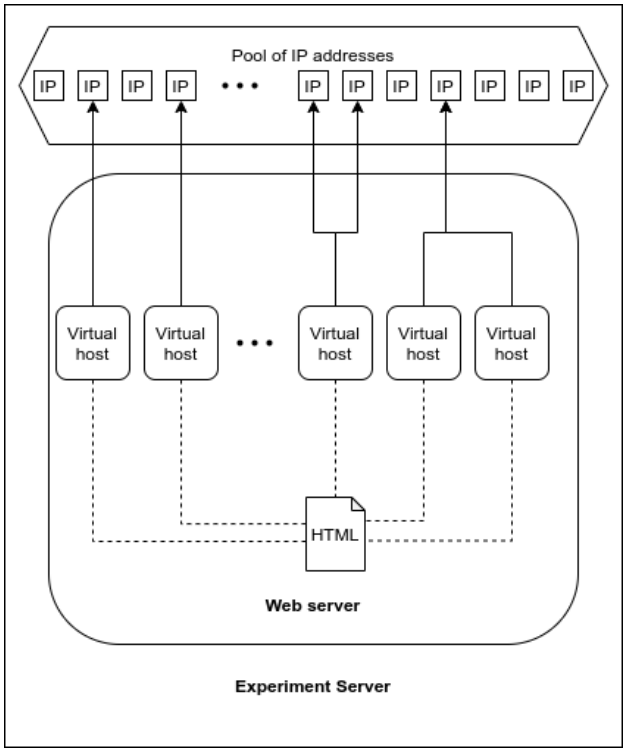

Figure 3.2: Diagram of the web server setup in the experiment server

3.3.4. Data Collection Infrastructure

The experiment setup consists of two servers: the experiment server and the data server. The experiment is responsible for running the experiments, which includes running the web servers, SSH server, the authoritative DNS server, and the data collection. The data server is responsible for backing up the data collected on the experiment server. The experiment server has low data storage, therefore backing up the data to the data server is necessary. Table 3.4 gives the technical specifications of the server. The data server has one IPv4 address.

Technical specifications	
OS	Ubuntu 20.04.2 LTS
CPU clock speed	2.4 GHz
CPU cache	28 MB
CPU core count	1
RAM	2 GB
Storage	3 TB

Table 3.4: Technical specifications of the data server

3.4. External data sources

In the data analysis, we also make use of external data sources besides the data stemming from the DNS server and the virtual hosts. Table 3.5 shows an overview of the collected data.

3.4.1. CT logs

From CT logs, we retrieve all the certificates of the generated domain names using the API of crt.sh [9]. We extract the timestamp when the certificate was appended to the CT log. The timestamp indicates when the domain name is present in the CT log, ie. when the domain name is being leaked via CT. Furthermore, we check which CT logs have appended the certificates. In total, we retrieve 1442 certificate entries.

3.4.2. Threat intel

To analyze the source of incoming traffic, we retrieve threat intel on the sources from VirusTotal [58] and from Team Cymru [11]. We make use of their API to automate the retrieval of data. We do not use their geolocalization data because geolocalization of IP addresses is deemed to be inaccurate [32]. In total, we download threat intel for 0.8M IP addresses.

Data source	Entries	Size
VirusTotal	458804	28MB
BGP	326257	41MB

Table 3.5: Overview of the external data sources

3.5. Experimental Evaluation

Name	Address space	Network configuration	Certificate type
610a	IPv6	1 virtual host listening on 1 IP	issued by LE
610b	IPv6	1 virtual host listening on 1 IP	self-signed
410a	IPv4	1 virtual host listening on 1 IP	issued by LE
410b	IPv4	1 virtual host listening on 1 IP	self-signed
4p0c	IPv4	2 x 1 virtual host listening on 1 IP	none
6p0c	IPv4	2 x 1 virtual host listening on 1 IP	none

Table 3.6: Overview of all the experiments

The experimental evaluation consists of 16 experiments with different configurations. The difference between them lies in steps (3) and (4) of Figure 3.1. Table 3.6 gives a complete overview of all the experiments.

In Step (3), different configurations for virtual hosts are available, allowing us to test the effect of CT logs on scanning traffic in different scenarios. There are 3 variables in the configuration of the virtual hosts: the type of address space, the network configuration, and the website configuration. Experimenting in both IP address spaces – IPv4 and IPv6 – shows how the effect of CT differs depending on the address space (see RQ 2.1). The purpose of the different network configurations is to evaluate the effect of CT on the host-level and the network level (see RQ 1.2, RQ 1.2 and RQ 1.3).

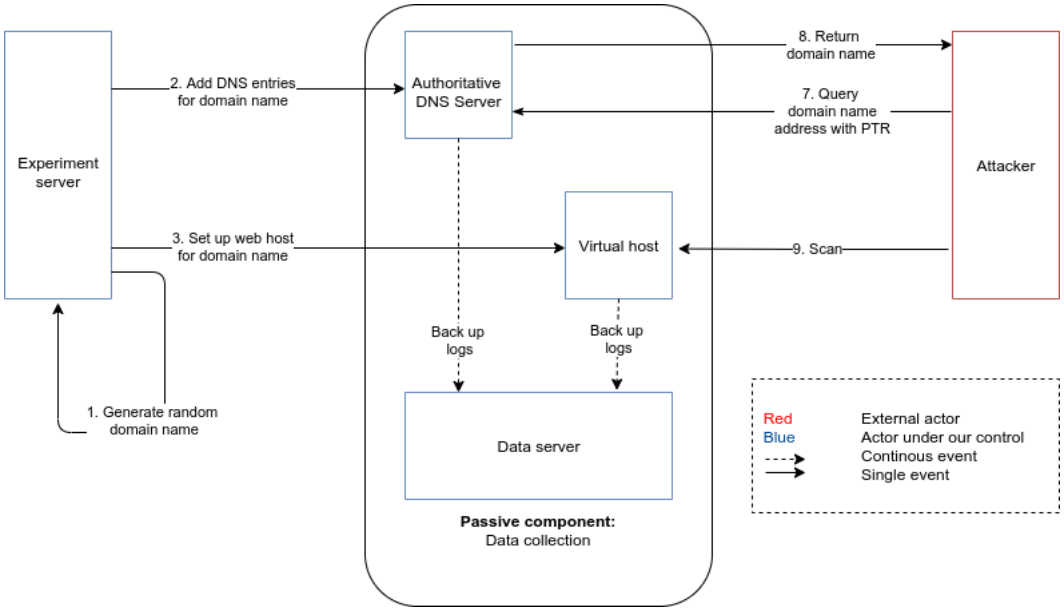

Figure 3.3: Overview of the interaction between the attacker and the virtual host without CT involvement

The experimental evaluation can be split into two different sets of experiments: experiments with the active component and ones without. Experiments without the active component are not disseminating the generated domain names via CT logs. The data collected from these experiments provide a baseline with no CT log involvement. We refer to such experiments as control experiments. Experiments with the active component are referred to as main experiments. The control experiments help to isolate the impact of CT in the main experiment. The step that differs between the two sets of experiments is step (4): request a certificate from a CA following the description in Section 3.2 or generate a self-signed certificate in the experiment server without appending the certificate to a CT log. Figure 3.3 shows how the process changes with the altered step (4).

3.5.1. Experiment naming scheme

The experiments are assigned a 4 character code to denominate the type of experiment. Every character position corresponds to an experiment characteristic allowing one to easily determine the nature of the experiment only with the code.

First character: Address space
The first character corresponds to the address space of the experiment. The majority of the experiments are conducted exclusively in one address space: IPv4 or IPv6, denoted respectively by the number ”4” or ”6”.

In summary:

- 4xxx: experiment in IPv4 address space
- 6xxx: experiment in IPv6 address space

Second character: Host configuration
The second character corresponds to the configuration of the hosts. There are 2 configurations denoted by 1 and p. In experiments of configuration 1, 1 web host is set up on 1 IP with one website. The PTR configuration denoted by ”p” is used for the PTR experiments to determine whether PTR records leak domain names. The PTR configuration is both used for IPv4 and IPv6 experiments. In this configuration, we have 2 web hosts, each hosting a website with a unique domain name on separate IP

addresses. For both domain names, we create an A or AAAA DNS record and also the associated NS record to that zone. Let's name one domain, domain 1, and one domain, domain 2. We craft a PTR record pointing to the IP address of domain 1, but with the name of domain 2.

In summary:

- x1xx: 1 website on 1 web host
- xpxx: PTR experiment

Third character: Website page
The third character corresponds to the website page. We only use one configuration for the website: a simple HTML page with text hosted at the root path. It is denoted by xx0x.

Fourth character: Certificate type
The fourth character corresponds to the certificate type associated with the domain name of the website in the experiments. In configuration a, certificates are issued by LE; in configuration b, certificates are generated locally on the experiment server and self-signed. For configuration c, there is simply no certificate.

In summary:

- xxxa: Certificate from the CA LE
- xxxb: Self-signed certificate generated on the experiment server
- xxxc: No certificate

3.5.2. IP address allocation

For every experiment, various amounts of IPs are allocated. The IPs allocated to the experiments have not been used previously. Due to the abundance of IPv6 addresses at our disposal, every experiment is assigned one whole /64 subnet, regardless of how many addresses will be used. However, due to the lack of IPv4 addresses, we do not assign an IPv4 subnet to an experiment.

3.5.3. Certificates

The experiment's domains are relayed to CT logs by requesting certificates from a trusted CA. After issuing the certificate to our domain, the CA appends the certificate to one or more CT logs. Since the domain name is embedded in the certificate, it is publicly accessible. Thus, we have leaked the domain name via CT.

For the experiments, we are using the CA LE for requesting certificates. Since the CA imposes a rate limit on certificate issuance at 50 certificates per week, we request at a frequency of at least 210 minutes to prevent from hitting the rate limit. With an interval of 210 minutes, at most 50 certificates are requested in 7 days.

The certificates are requested using the Certbot tool, which automatically requests certificates from LE. The tool automatically renews certificates every 60 days, even though LE certificates are valid for 90 days.

To establish a control group, we are repeating every experiment twice, with the only difference being the certificate. Instead of requesting a certificate from a CA, we generate locally self-signed certificates using OpenSSL, an open-source cryptographic, on the experiment server. The certificates follow the X.509 standard, have a validity of 1 year and use the cryptographic algorithm RSA with a key size of 2048 bits.

3.5.4. Experiments

Experiment 610a

The purpose of this experiment is to determine the effect of CT on scanning traffic for a single host and its subnet (RQ 1.1 and RQ 1.2). To understand the impact on the subnet, all the traffic destined for the subnet is examined. The results of the experiment are compared to the result of the baseline experiment 610b where CT is not involved. Next, the results of the experiment are used in the comparison between IPv4 and IPv6 to determine how CT affects scanning traffic differently depending on the address space (RQ 1.3).

We set up a virtual host listening on an IPv6 address in an unused /64 subnet dedicated to the experiment. The virtual host serves one domain name. The website behind the domain name serves a simple HTML page at the root page with only text. For the domain name, we create a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an AAAA and PTR record with the domain name and the IPv6 address. After setting up the domain name, we request a certificate from LE to leak the domain name into a CT log.

Experiment 610b

The purpose of this experiment is to form a baseline, where CT is not involved. The results of the experiment are compared to the result of the equivalent experiment 610a, where CT is involved, to determine the effect of CT on scanning traffic for a single host and its subnet (RQ 1.1 and RQ 1.2).

We set up a virtual host listening on an IPv6 address in a /64 subnet dedicated to the experiment. The virtual host serves one domain name. The website behind the domain name serves a simple HTML page at the root page with only text. For the domain name, we create a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an AAAA and PTR record with the domain name and the IPv6 address. After setting up the domain name, we generate locally a self-signed certificate.

Experiment 410a

The purpose of this experiment is to determine the effect of CT on scanning traffic for a single host in IPv4 (RQ 1.1). The results of the experiment are compared to the result of the baseline experiment 410b, where CT is not involved. Next, the results of the experiment are used in the comparison between IPv4 and IPv6 to determine how CT affects scanning traffic differently depending on the address space (RQ 1.3).

We set up a virtual host listening on an IPv4 address. The virtual host serves one domain name. The website behind the domain name serves a simple HTML page at the root page with only text. For the domain name, we create a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an A and PTR record with the domain name and the IPv4 address. After setting up the domain name, we request a certificate from LE to leak the domain name into a CT log.

Experiment 410b

The purpose of this experiment is to form a baseline, where CT is not involved. The results of the experiment are compared to the result of the equivalent experiment 410a, where CT is involved, to determine the effect of CT on scanning traffic for a single host (RQ 1.1).

We set up a virtual host listening on an IPv4 address. The virtual host serves one domain name. The website behind the domain name serves a simple HTML page at the root page with only text. For the domain name, we create a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an A and PTR record with the domain name and the IPv4 address. After setting up the domain name, we generate locally a self-signed certificate using.

Experiment 4p0c

The DNS setup of the domains in all the experiments follows RFC 1912, which states that for every IP, there should be a matching PTR record. However, a PTR record can leak a domain name. For example, a scanner randomly reverse queries an IP address to which also points a domain name. It will then learn the domain name of the website that is hosted on the queried IP. The purpose of the experiment is to act as a baseline for the experiments involving CT; the results of the experiment are used to control the defect via reverse pointers (RQ 1 and RQ 3). Furthermore, this experiment aims to figure out whether scanners are using the leak domain names through PTR records for scanning purposes (RQ 2).

We set up 2 virtual hosts with each an IPv4 address. Each virtual host is hosting a website with a unique domain name on its IP address. The websites behind the domain names serve each an HTML page at the root page with only text. For the domain names, we create for each a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an A record with the domain name and the IP address. Let's name one domain, domain 1, and one domain, domain 2. We craft a PTR record pointing to the IP address of domain 1, but with the name of domain 2.

Experiment 6p0c

The purpose of the experiment is the same as of experiment 4p0c, except that the experiment is run in the IPv6 address space: it acts as a control to correct the defect of reverse pointers (RQ 1 and RQ 3) and to determine the effect on scanning traffic (RQ 3).

We set up 2 virtual hosts with each IPv6 address. Each virtual host is hosting a website with a unique domain name on its IP address. The websites behind the domain names serve each an HTML page at the root page with only text. For the domain names, we create for each a new zone within the 'example.com' zone by creating an NS record and adding it to the authoritative DNS server. Furthermore, we add an AAAA record with the domain name and the IP address. Let's name one domain, domain 1, and one domain, domain 2. We craft a PTR record pointing to the IP address of domain 1, but with the name of domain 2.

3.6. Limitations

Our experimental evaluation of CT is not without its limitations. Ideally, the experiments are mutually independent: for deterministic experiments, the outcome should always be the same no matter the order of the experiments. However, the outcome of our experiments is not deterministic, hence verifying whether they are mutually independent is more difficult. Our experiments rely on 3 assumptions, which also become limitations to our study. We discuss the limitations of the study and the measures taken to minimize them.

In the attacker model, we have an attacker who constantly monitors CT logs. If he observes in a CT log that, in a short amount of time, a large number of certificates for domains all coming from one domain zone are being appended, he might interpret that phenomenon and modify his scanning behaviour. For example, he could interpret the burst of certificates as an indication that the certificates are issued for experimental purposes and therefore, he would cease scanning for these domains. To minimize the risk of this happening, we never request certificates in quick succession. Instead, we scatter certificate issuances with at least a 210-minute interval between them. The 210-minute interval was not arbitrarily determined: its second purpose is to ensure that we always respect the rate limit imposed by the CA LE. However, it is not certain that the time interval of 210 minutes is a sufficient measure, nor that requesting a lot of certificates in quick succession can arouse suspicion. We also note that using a fixed time interval could also arouse suspicion, though we deem it less likely to be detected compared to the alternative.

Besides the issue of rapid certificate issuance, the use of a single domain zone for our study could pose a problem. In the attacker model, if the attacker notices a large number of certificates are being issued for one domain zone, he might become suspicious and adapt his scanning behaviour. However, due to the limited budget of the study, using a new domain zone for every experiment is unfeasible.

Another concern is the proximity of the IP addresses used in the experiments. This is less of a concern for experiments in the IPv6 address space, as we have an abundant amount of IP addresses at our disposal and therefore we can separate each experiment into their own /64 subnet. However, for experiments in the IPv4 address space, all experiments are in the same /24 subnet. Possibly, experiments in close proximity can affect each other. For example, an attacker discovers a virtual host via CT and decides to not only scan that virtual host but also its network for other virtual hosts. We are specifically testing for such network-wide scanning patterns in the IPv6 address space, but it is not possible for the IPv4 address space. Hence, the interpretation of scanning traffic observed for IPv4 experiments should take this limitation into account.

Since we are adhering to RFC 1912 for the DNS configuration of the generated domain names, we introduce a second source, besides CT logs, where domain names can be leaked, namely the Authoritative DNS server via PTR records. To account for the defect via reverse pointers, we perform two additional experiments (4p0c and 6p0c) to determine the effect of PTR on scanning traffic in IPv4 and IPv6 address space. Published works can be found using PTR records as a data source: Fiebig et al. [16] studied the state of IPv6 using data gathered from PTR records. So, the attacker model assumes that the domain names are only leaked via CT or PTR. However, they could also be learned from other sources that collect DNS requests, such as FarSight's DNSDB. But the first leakage of the domain is from CT or PTR. Distinguishing between other sources and CT/PTR is outside the scope of this research.

The network configuration of the IPv4 addresses at our disposal pose a limitation to the network traffic capture: for security reasons, we do not receive traffic for certain port numbers. We acknowledge that the network traffic analysis will not be complete. Below you can find which network ports are blocked by the by the firewall of our network uplink:

- TCP: 23, 25, 42, 111, 135, 137, 138, 139, 161, 162, 427, 445, 524, 593, 16987, 16988, 16994, 31386, 31387, 31388, 61301, 61317, 61330, 61331
- UDP: 42, 67, 111, 135, 137, 138, 139, 161, 162, 427, 445, 524, 593, 1434, 3702, 6784, 16988, 16989, 31387, 61301, 61316, 61317, 61333

3.7. Ethical considerations

Regarding the experimental evaluation of the study, we follow the guidelines proposed by the Menlo Report [5]. We identify the subjects of the study and assess whether the experiments can cause any harm. For the external data sources, we follow the instructions and respect the rate limit imposed by the organisations, therefore, it is unlikely that we overwhelm their data services. As for the web servers, we do not send any malicious data to the querier, but only a simple HTTP page which should not cause any harm to the receiver. In the synthesis of the passive measurements, we are minimizing any potential harm to the subjects by anonymizing identifying information such as IP addresses. The measures taken to ensure that the subjects of the study are only incurring minimal risk.

4

Results

In this chapter, we analyse the amassed data to answer the research questions in Chapter 1. In the first part of the chapter, we analyse every experiment in isolation for each dataset collected. The chapter opens with the analysis of the DNS dataset in Section 4.1, followed by the analysis of the network traffic dataset in Section 4.2 and finally, the analysis of HTTP traffic in Section 4.3. These first 3 sections share the same structure: first, we give an overview of the collected data, then we analyze each experiment using the dataset in question. Examining the experiments in isolation gives us a foundation for the next section, where we will synthesize the results in an attempt to answer the research questions.

4.1. DNS data analysis

In this section, we will analyse data collected from the DNS server. We start by describing the collected DNS data from the Authoritative DNS server. From the collected DNS data, we extract the queries relevant to the experiments and analyse them. We investigate the domain names queried, the arrival time, the properties and the originators of the queries.

4.1.1. Data description

We collect a total of 525.9 MB of logs from our authoritative DNS server, with the first query recorded on Mar. 03, 2021 at 06:25:35 and the last query on Sep. 05, 2021 at 06:24:57. The collected data consists of 3.7M DNS queries, where 30.2 % of the queries point to a domain generated in one of the

Experiment	Time period	# DNS queries
410a	03 Mar, 21 - 05 Sep, 21	5208
410b	03 Mar, 21 - 05 Sep, 21	4001
4p0c	03 Mar, 21 - 05 Sep, 21	5120
610a	12 Mar, 21 - 05 Sep, 21	5971
610b	12 Mar, 21 - 05 Sep, 21	0
6p0c	12 Mar, 21 - 05 Sep, 21	0

Table 4.1: Overview of DNS queries logged in authoritative DNS server

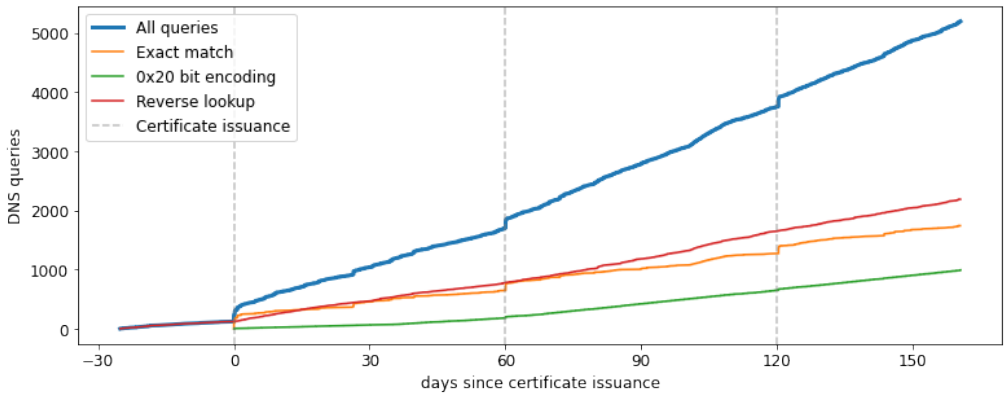

Figure 4.1: Cumulative number of DNS queries in experiment 410a. The DNS queries are classified by what is queried. Queries with exact match contain the exact domain name. Queries with 0x20 bit encoding contain the domain name with 0x20 encoding. Reverse lookups contain the address of the domain in ".in-addr.arpa" format. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

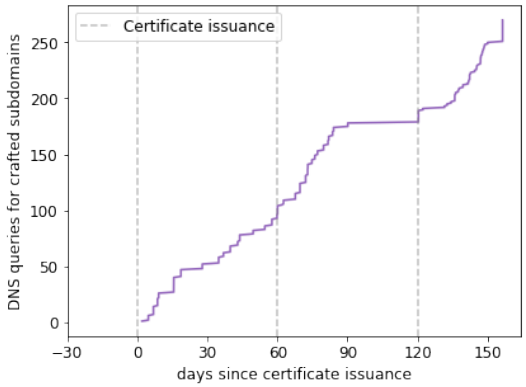

Figure 4.2: Cumulative number of DNS queries containing the honeytoken but not matching the domain name in experiment 410a. The queried domains are subdomains of the experiment and are of the form "subdomain.example.com". The grey dotted lines indicate when the certificate of the domain was issued or reissued.

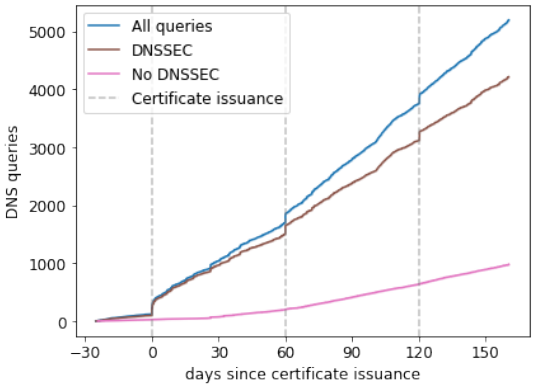

Figure 4.3: Cumulative number of DNS queries with and without DNSSEC support in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

experiments. Table 4.1 summarizes the data collected from the authoritative DNS server, giving the number of DNS queries and the time period per experiment.

4.1.2. Experiment 410a

In experiment 410a, we are hosting a domain on one address in IPv4 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv4, via CT affects DNS traffic pertaining to the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. We collect 5208 DNS queries in total. During the certificate issuance process, we detect 12 DNS queries from the CA. The following analysis will ignore these DNS queries from the CA, as they do not stem from a leak via CT.

Figure 4.1 shows the cumulative number of DNS queries received for the experiment's domain. Queries containing domain names exactly matching the domain name of the experiment account for 22.5 %, followed by queries with domain names only matching without letter case distinction, ie. where

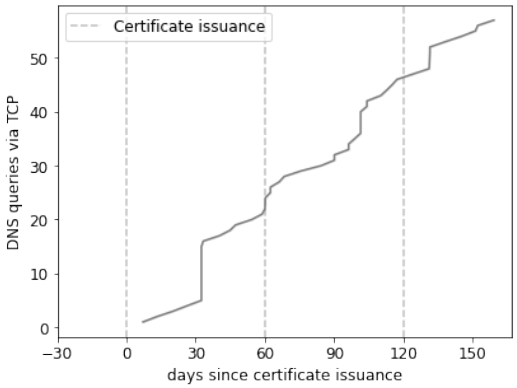

Figure 4.4: Cumulative number of DNS queries made via TCP in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

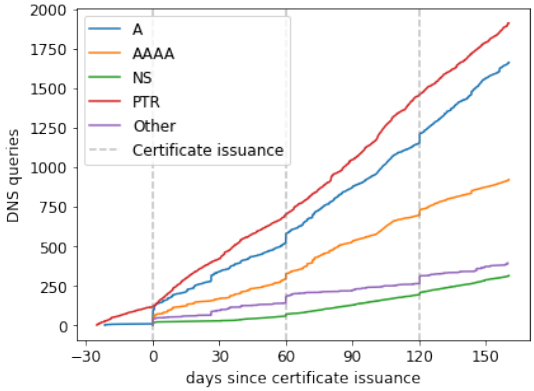

Figure 4.5: Cumulative number of DNS queries per DNS query type in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

0x20 encoding [59] is used, make up 13.0 % of DNS queries of the domain. Queries with reverse lookups are responsible for the largest part of the domain's DNS traffic at 54.9 %. Before the certificate issuance, only queries with the reverse lookup of the experiment's host are detected. These queries are received at a constant rate. It is impossible to receive requests for the experiment's domain name, because it was not created yet. At time 0, the certificate of the newly set up domain is issued and appended to a CT log during the certificate issuance process. We begin to detect the first queries containing the domain name and the rate at which queries with reverse lookups are received increases. The overall DNS traffic pertaining to the experiment surges, mainly because of queries where the requested domain name matches the domain name of the experiment. Similar jumps in traffic occur on days 60 and 120, which corresponds to the reissuance dates of the certificate. When differentiating between the different types of queries, we find that these jumps in traffic are only present for queries, exactly matching the domain of the experiment. In between the issuances, the number of DNS queries is growing at a linear rate for all three types of queries.

In Figure 4.2, we depict the cumulative number of DNS queries containing a domain name with the generated honeytoken of the experiment, but the complete domain name does not match the one of the experiment. The requested domain names are all non-existent subdomains of the experiment domain. These queries only account for 5.2 %, a small percentage of the DNS traffic in the experiment. As expected, we only detect this type of query after the leak of the domain name. The traffic increases approximately at a linear rate. Querying non-existent sub domain names can discover hosts, also ones that are not indexed in any CT logs. Such an attack is called subdomain enumeration [3]. Such queries show that there is more elaborate scanning on the experiment host.

Figure 4.4 depicts the cumulative number of DNS queries performed via TCP. The recorded DNS traffic is predominantly sent via UDP, making up 99.1 % of the traffic, while TCP queries are only responsible for 0.9 %. Since the DNS traffic is mostly composed of UDP traffic, the cumulative count of UDP queries follows nearly the same growth as the overall traffic as seen in Figure 4.1. In Figure 4.4, we detect the first TCP queries only after the certificate was leaked and none before. TCP traffic grows at a near linear rate except for a small jump at around day 30. Though issuing the certificate affects DNS traffic via TCP, it remains unaffected at reissuances.

Figure 4.3 depicts the cumulative number of DNS queries with and without DNSSEC support. For 80.8 % of the queries, the originator supports DNSSEC. Here, we see the same surge in DNS traffic at reissuances as in the overall DNS traffic. The difference between the two plots is that the overall DNS traffic has a stronger linear growth between issuances. Queries without DNSSEC support are not affected by reissuances; there is not a burst of requests when the domain is leaked again via CT. When comparing Figure 4.1 and Figure 4.3, both figures seem to be identical: DNS queries with exactly

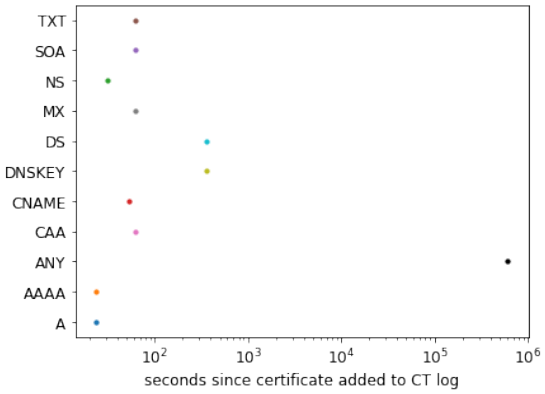

Figure 4.6: First DNS query containing honeytoken of experiment for every DNS query type in experiment 410a. X-axis is log-scaled.

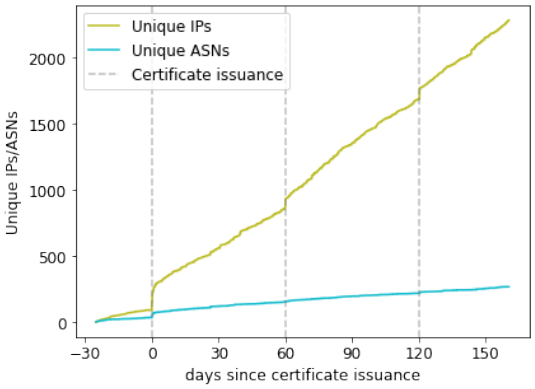

Figure 4.7: Cumulative number of unique IPs and ASes making queries for the domain in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

matching domain names tend to support DNSSEC while DNS queries with 0x20 encoding tend to not support DNSSEC. Only 4 % of DNS queries do not follow these combinations.

Figure 4.5 shows the average cumulative number of DNS queries per DNS query type. There is a tendency for traffic to increase at certificate issuances. Queries for ANY and PTR records do not follow this trend. Before the initial certificate issuance, we only detect A and PTR records. Receiving A records before the issuance is peculiar, since the domain name has not yet been created. When looking at the A queries, we see that they request the reverse lookup and not a domain name.

The first DNS query containing the honeytoken of the experiment was detected 23.3 seconds after the certificate was issued and leaked in a CT. Figure 4.6 shows the detection time of the domains for every record type on a logarithmic time scale. The lowest detection times are detected for type A and AAAA records, which share very similar detection times. TXT, SOA, MX and CAA records have near identical detection times. Same goes for DS and DNSKEY queries, which share the same detection times. ANY records have the longest detection times.

Figure 4.7 shows the total number of unique IPs and ASes. Disregarding the surges on days 0, 60 and 120, the number of unique IPs and ASes is increasing at a constant rate. The largest surge for both IPs and ASes happens at the initial issuance. Subsequent reissuances only trigger a small jump in traffic. For the AS count, the jump is barely detectable. Even though more IPs are detected at reissuances, the number of unique ASes is not surging. The newly detected IPs come from ASes, from which we have already detected other IPs. This could indicate that the actors querying the domain names at the initial leak are also querying at subsequent leaks from reissuances. We suspect these actors have a pool of addresses at their disposal and that these addresses belong to the same AS. These actors are rotating the addresses used for scanning, hence causing these surges in newly detected addresses but not in ASes.

Figure 4.8 shows the average cumulative number of DNS queries per registry. Before the leak of the domain name, we detect traffic from all 5 registries, but only in small quantities compared to after the leak. Most of the DNS traffic comes from queriers in the ARIN registry, accounting for 74.8 % of the traffic, followed by the RIPENCC registry with 17.0 %. Only traffic from these two registries jumps slightly at the issuances. Traffic from APNIC is increasing after the leak, but unpredictably. Traffic from AFRINIC and LACNIC is nearly non-existent.

Using the threat intel from VirusTotal on the IP sources, we can differentiate between traffic from suspicious and benign sources. Figure 4.9 shows only total traffic from suspicious IPs. Before the leak, we only detect few queries from suspicious addresses. Traffic from suspicious sources grows

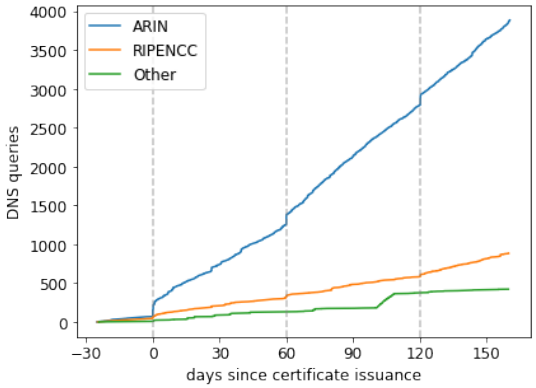

Figure 4.8: Cumulative number of DNS queries per registry 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

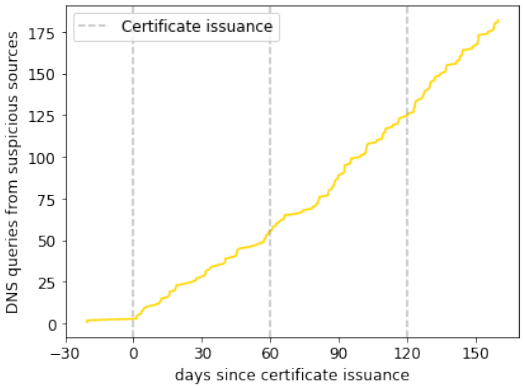

Figure 4.9: Cumulative number of DNS queries from suspicious sources in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

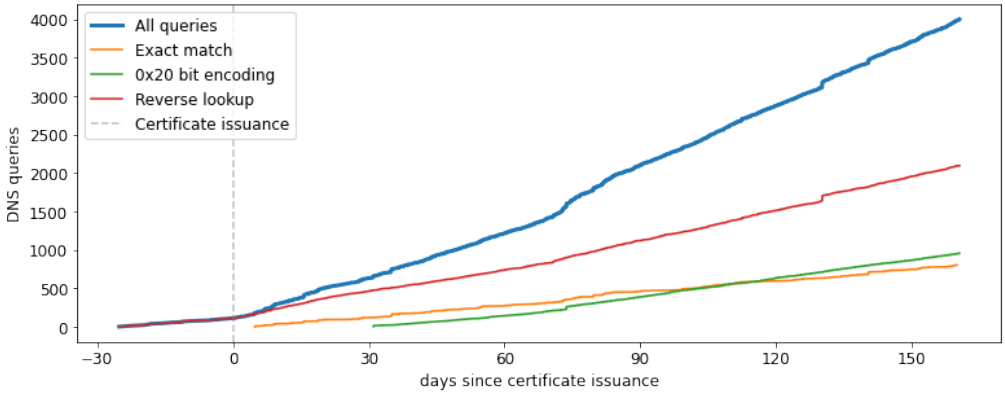

Figure 4.10: Cumulative number of DNS queries in experiment 410b. The DNS queries are classified by what is queried. Queries with exact match contain the exact domain name. Queries with 0x20 bit encoding contain the domain name with 0x20 encoding. Reverse lookups contain the address of the domain in ".in-addr.arpa" format. The grey dotted line indicates when the certificate was issued.

linearly after the leak. But at reissuances, traffic is not affected. An explanation for the linear growth of suspicious traffic could be that suspicious scanners disregard any subsequent leaks of domain names and therefore they keep track of domain names they collected from CT logs.

4.1.3. Experiment 410b

In experiment 410b, we are hosting a domain on one address in IPv4 and generate a self-signed certificate locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv4, affects DNS traffic pertaining to the domain of the experiment. This experiment will be used as a baseline for comparison with experiment 410a. Experiment 410b was run once and ran from Mar 3, 2021 to Sep 5, 2021. In total, we collect 4001 DNS queries.

Figure 4.10 shows the cumulative number of DNS queries received for the experiment's domain. Queries containing domain names matching the domain name of the experiment account for 20.1 %, followed by queries with domain names only matching without letter case distinction (0x20 encoding) make up 24.0 % of DNS queries of the domain. Queries with reverse lookups handle the largest part of the DNS traffic at 45.5 %.

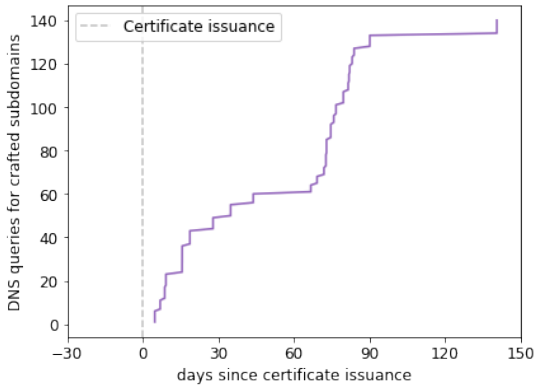

Figure 4.11: Cumulative number of DNS queries containing the honeytoken but not matching the domain name in experiment 410b. The queried domains are subdomains of the experiment and are of the form ”subdomain.example.com”. The grey dotted line indicates when the certificate was issued.

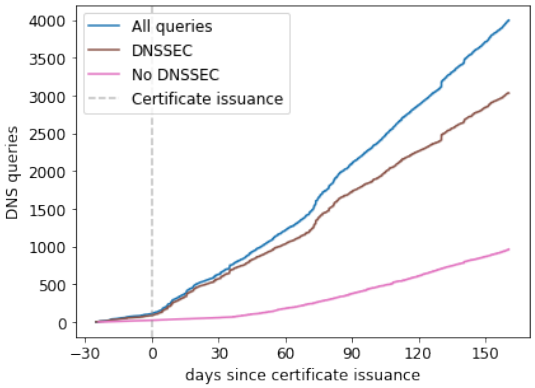

Figure 4.12: Cumulative number of DNS queries with and without DNSSEC support in experiment 410b. The grey dotted line indicates when the certificate was issued.

Before the certificate issuance, only queries with a reverse lookup are received. Since the domain has not yet been created, it is impossible to receive requests with the domain name. At time 0, the certificate for the experiment is generated locally, and the required DNS entries are appended to the DNS server. Shortly after, we detect an increase in PTR queries for the domain name. The rate of PTR queries is relatively constant, except for a small jump at around day 130. After 5 days, we receive the first DNS query for the domain name of the experiment. The only public place where the domain name can be found is in the PTR record of the experiment host available on the DNS server. After 30 days, we detect the first query for the domain name with 0x20 encoding. The rate at which we detect these types of queries is higher than queries with exactly matching domain name.

In Figure 4.11, we depict the cumulative number of DNS queries containing a domain name including the generated honeytoken of the experiment, but where the complete domain name does not match the one of the experiment. The requested domain names are all non-existent subdomains of the experiment domain. These queries only account for 3.5 %, a small percentage of the DNS traffic in the experiment. We only detect this type of query after the leaking the domain name via the PTR record. Two periods of detection can be observed: from day 5 to day 45, we detect about 60 such queries and from day 75 to 90, about 70 queries. Such as in experiment 410a, queries for such domain names are probably because of brute force scanning of subdomains.

Figure 4.13 depicts the cumulative number of DNS queries performed via TCP. The recorded DNS traffic is predominantly sent via UDP, making up 99.1 % of the traffic, while TCP queries are only responsible for 0.9 %. Since the DNS traffic is mostly composed of UDP traffic, the cumulative count of UDP queries follows nearly the same growth as the overall traffic Figure 4.10. Before the leak of the domain name, no queries were made via TCP; they are detected at a linear rate from day 5 onwards.

Figure 4.12 depicts the cumulative number of DNS queries with and without DNSSEC support. For 75.8 % of the queries, the originator supports DNSSEC. After day 5, the rate of queries with DNSSEC support increases and grows at a linear rate with a small jump at around day 75 and 130. We only see a slight increase in queries without DNSSEC support after day 30.

Figure 4.14 shows the average cumulative number of DNS queries per DNS query type. PTR queries are the most prevalent query accounting for 45.5 % of the traffic, followed by A queries for 26.8 %. These two types of queries also appear before the certificate creation, though A queries contain a reverse DNS lookup address instead of a domain name. Five days after the certification creation, we start to detect other types of DNS queries. For all query types, the incoming query rate is approximately constant with some minor deviations.

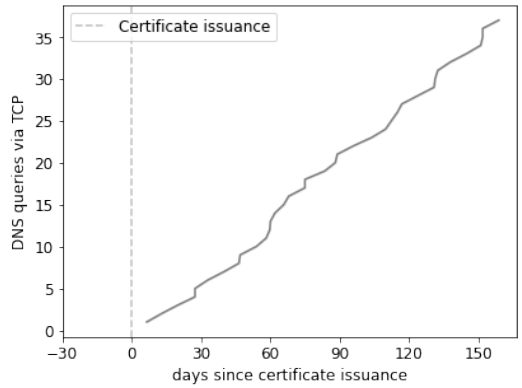

Figure 4.13: Cumulative number of DNS queries made via TCP in experiment 410b. The grey dotted line indicates when the certificate was issued.

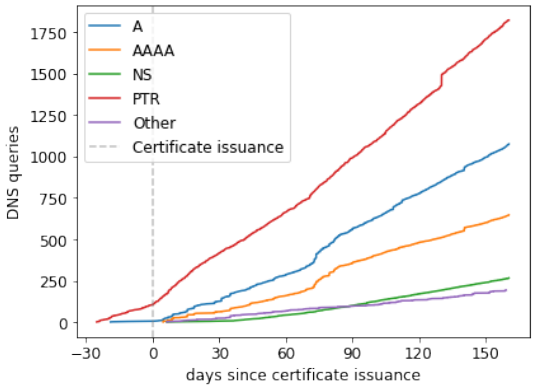

Figure 4.14: Cumulative number of DNS queries per DNS query type in experiment 410b. The grey dotted line indicates when the certificate was issued.

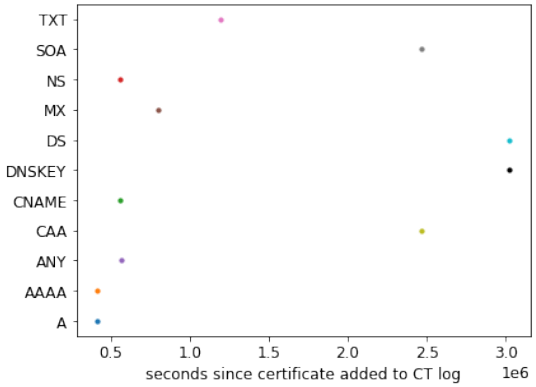

Figure 4.15: First DNS query containing honeytoken of experiment for every DNS query type in experiment 410b.

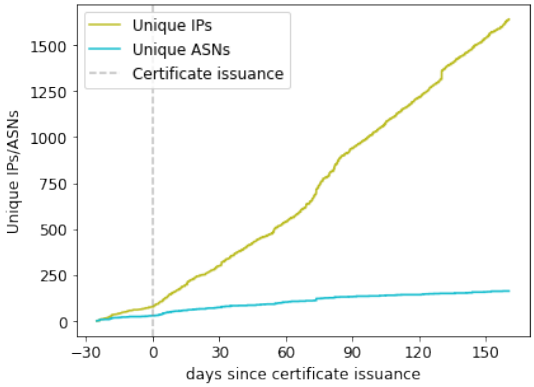

Figure 4.16: Cumulative number of unique IPs and ASes making queries for the domain in experiment 410b. The grey dotted line indicates when the certificate was issued.

The first DNS query containing the honeytoken of the experiment was detected 4.7 days after the certificate and DNS entries were created. Figure 4.15 shows the detection time of the domains for every record type on a logarithmic time scale. The lowest detection times are detected for type A and AAAA records, with the first query containing the honeytoken being of type A. Queries of type DS and DNSKEY have the longest detection time of 34.7 days.

Figure 4.16 shows the total number of unique IPs and ASes from which we receive DNS queries. The number of unique IP addresses, compared to the number of unique ASes, grows at a significantly higher rate. After generating the certificate, the rate at which we detect new IP addresses increases, but not for the number of unique ASes, which stays constant. The number of unique ASes grows at the same rate before and after the certificate generation. Assuming that an AS represents an actor, no new actors are starting to scan the experiment host because of the certificate generation and the experiment domain creation. However, the number of unique IP addresses significantly increases, showing that actors that are scanning the host, regardless of whether a domain exists, increase their scanning effort and their resources, such as IP addresses, upon discovering a domain.

Figure 4.17 shows the average cumulative number of DNS queries per registry. Most of the traffic originates from the ARIN registry with 78.2 %. After the certificate generation, we detect a higher rate of DNS queries coming from the top 3 registries: ARIN, RIPENCC, and APNIC. For AIRN and APNIC, the rate changes significantly, whereas for RIPENCC, the range change is subtle. This result shows that,

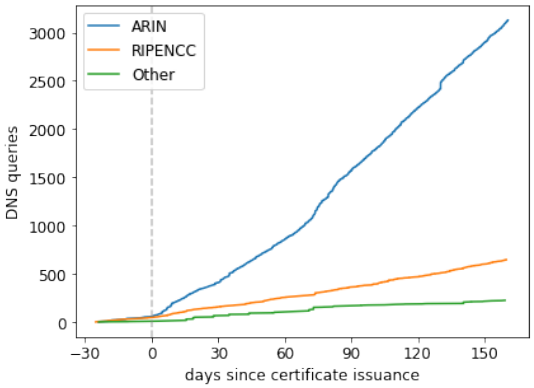

Figure 4.17: Cumulative number of DNS queries per registry 410b. The grey dotted line indicates when the certificate was issued.

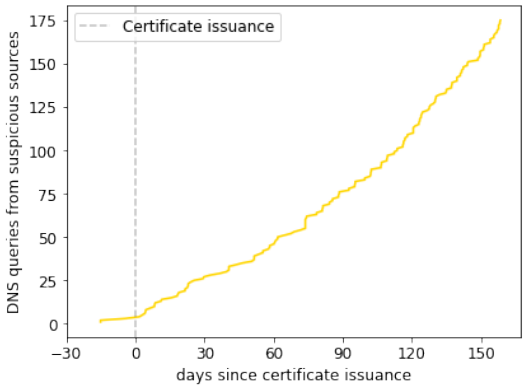

Figure 4.18: Cumulative number of DNS queries from suspicious sources in experiment 410b. The grey dotted line indicates when the certificate was issued.

proportionally to the incoming traffic of the registries, more actors in ARIN and APNIC are monitoring for new domains.

Using the threat intel from VirusTotal on the IP sources, we can differentiate between traffic from suspicious and benign sources. Figure 4.18 shows only total traffic from suspicious IPs. Before the certificate generation, we only detect few queries from suspicious addresses. Only after does the traffic from suspicious sources grow.

4.1.4. Experiment 610a

In experiment 610a, we are hosting a domain on one address in IPv6 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv6, via CT affects DNS traffic pertaining to the domain. Experiment 610a was run 3[1] times and ran from March 12, 2021 to September 5, 2021. In total, we collect on average 1990 DNS queries per domain, or 5971 DNS queries. During the certificate issuance process, we detect 12 DNS queries per domain from the CA. The following analysis will ignore these DNS queries from the CA, as they do not stem from a leak via CT.

Figure 4.19 shows the average cumulative number of DNS queries received for a domain. After a CT log appends a certificate, i.e., it leaks the domain name associated to the certificate, the domain experiences a surge in DNS traffic. The same phenomenon occurs on days 60 and 120, which correspond to the reissuance dates of the certificate. In between issuances, the number of DNS queries is increasing at a linear rate. Before the initial issuance, we detect no DNS queries for the domains of the experiment.

In Figure 4.19 and Figure 4.20, we also depict the average cumulative number of DNS queries with different types of matching. Queries containing exactly matching domains are the most frequent, accounting for 56.2 %. These queries experience the same surge at certificate issuances. Queries containing strictly case-insensitive matching domain names come in second with 36.3 %. For these types of queries, we do not see a surge in traffic at the initial certificate issuance or at reissuances; these types of queries are detected at a constant rate. Queries with reverse lookup only represent 1 % of DNS queries of the experiment.

The remaining 5.5 % of the queries point to unexisting domain names, though they contain the honeytoken of the domain name, see Figure 4.20. Traffic increases significantly on days 60, 120 and 150. At reissuances, traffic is increasing sharply and in between the reissuances, traffic is barely

[1]The Experiment 610a was run 5 times, but 2 runs failed because of an issue in the certificate issuance process

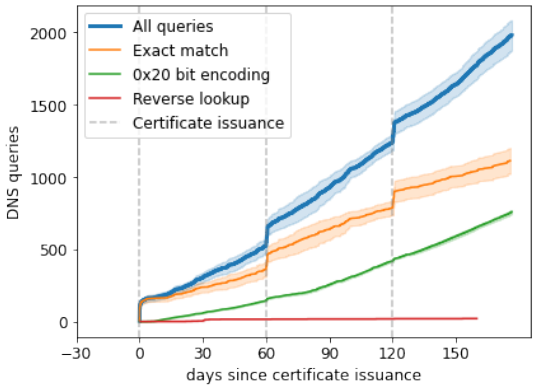

Figure 4.19: Average cumulative number of DNS queries of the 3 runs in experiment 610a. The DNS queries are classified by what is queried. Queries with exact match contain the exact domain name. Queries with 0x20 bit encoding contain the domain name with 0x20 encoding. Reverse lookups contain the address of the domain in ".in-addr.arpa" format. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

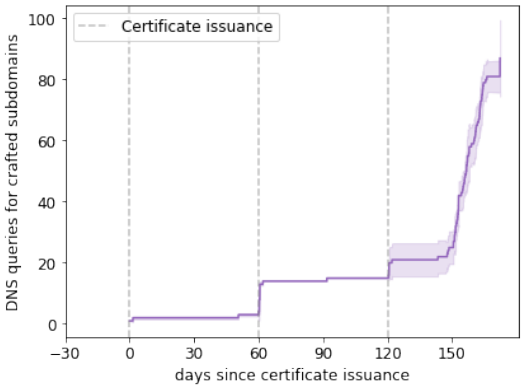

Figure 4.20: Average cumulative number of DNS queries containing the honeytoken but not matching the domain name of the 3 runs in experiment 610a. The queried domains are subdomains of the experiment and are of the form "subdomain.example.com". The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

increasing — between 1 to 5 requests in 30 days. The increase in traffic at reissuances is expected, as in Figure 4.19. But we also notice a surge on day 150, which does not coincide with a reissuance date: Traffic is not increasing sharply as in the previous surges, but it increases over 20 days. The requested domain names are all non-existent subdomains of the experiment domain.

Figure 4.22 depicts the average cumulative number of DNS queries performed via TCP. The recorded DNS traffic is predominantly sent via UDP, making up 98.2 % of the traffic, while TCP queries are only responsible for 1.8 %. Since the DNS traffic is mostly composed of UDP traffic, the accumulative count of UDP queries follows nearly the same growth as the overall traffic. In Figure 4.22, we detect on average 1 TCP query within the first day of the leak, but we detect no further queries until day 20. On day 20, TCP traffic grows linearly, with no signs of variations at reissuances. Though TCP traffic starts after issuing a certificate, it remains unaffected at reissuances.

Figure 4.21 shows the average cumulative number of DNS queries with and without DNSSEC support. For 61.3 % of the queries, the originator supports DNSSEC. Here, we see the same surge in DNS traffic at reissuances as in the overall DNS traffic. The difference between the two plots is that the overall DNS traffic has stronger linear growth between the issuances. Queries without DNSSEC support are not affected by reissuances as the counterpart with DNSSEC support. There is not a burst of requests when the domain is leaked via CT. When comparing Figure 4.19 and Figure 4.21, both figures seem to be identical: DNS queries with exactly matching domain names tend to support DNSSEC while DNS queries with case-insensitive matching domain names tend to not support DNSSEC. Only 4 % of DNS queries do not follow these combinations.

Figure 4.23 shows the average cumulative number of DNS queries per DNS query type. There is a tendency for traffic to increase at certificate issuances, with NS records only showing a very slight increase in count. ANY and PTR records do not follow this trend. We only detect ANY records at around day 15 and thereafter the count increases linearly. We detect the bulk of PTR records in a short time between days 20 and 35. Outside this range, few PTR records are detected. This delay in detection is also found for TCP traffic, as in Figure 4.22. Looking at DNS traffic over TCP, we find that the type of the queries is always ANY type. Hence, the traffic surge happens at the same moment in both figures.

The first DNS queries logged for the domains are at (19.4, 23.7, 12.2) seconds, with a mean of 18.4 seconds. They are of record type (A, TXT, AAAA) and bufsize (1680, 1452, 1680). The originators of the

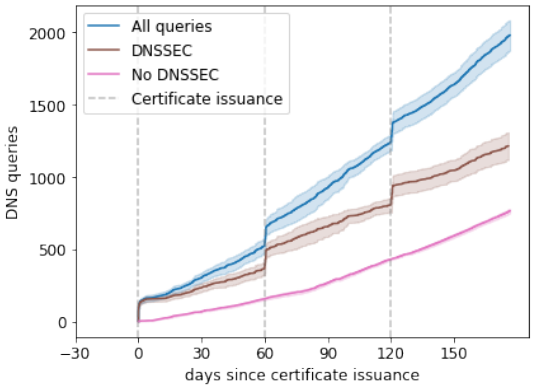

Figure 4.21: Average cumulative number of DNS queries with and without DNSSEC support of the 3 runs of the 3 runs in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation. The shaded area shows the standard deviation.

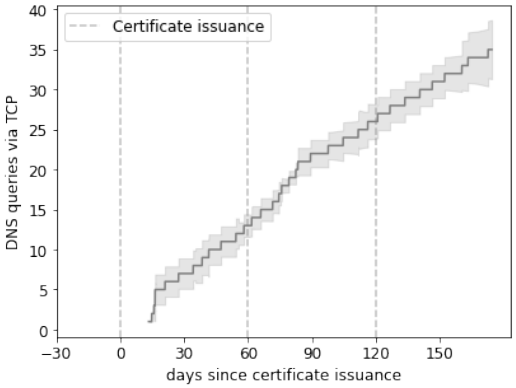

Figure 4.22: Average cumulative number of DNS queries made via TCP of the 3 runs in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

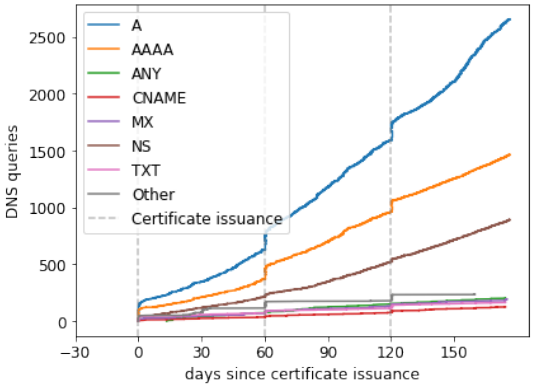

Figure 4.23: Average cumulative number of DNS queries per DNS query type of the 3 runs in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

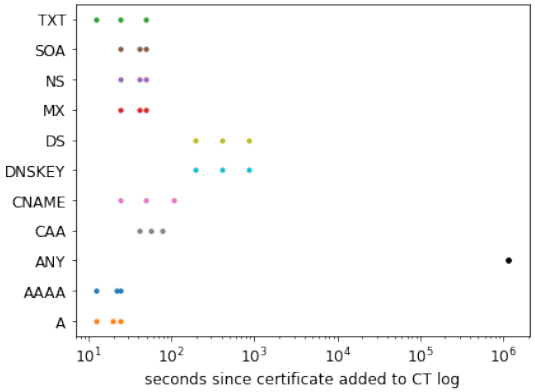

Figure 4.24: First DNS query containing honeytoken of experiment for every DNS query type of the 3 runs in experiment 610a. X-axis is log-scaled.

queries support DNSSEC and queried via UDP. The queried domain name matches exactly the domain name of the experiment. Figure 4.24 a) shows the detection time of the domains for every record type on a logarithmic time scale. The lowest detection times are detected for type A and AAAA records, which share very similar detection times. SOA, NS and MX records have near identical detection times, therefore showing that these types of records are queried together. Same goes for DS and DNSKEY queries, which share the same detection times. ANY records have the longest detection times, while PTR records have the most variance in detection times.

Figure 4.25 shows the total number of unique IPs and ASes. Disregarding the surges on days 0, 60 and 120, the number of unique IPs and ASes is increasing at a constant rate. The largest surge for both IPs and ASes happens at the initial issuance. Subsequent reissuances only trigger a small jump in traffic. For the AS count, the jump is barely detectable. We find the same result as in the previous experiments 410a and 410b.

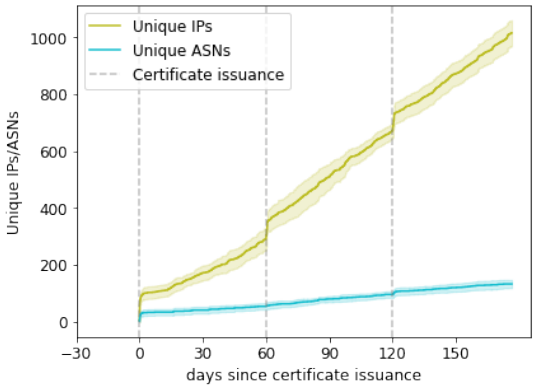

Figure 4.25: Average cumulative number of unique IPs and ASes making queries for the domain of the 3 runs in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

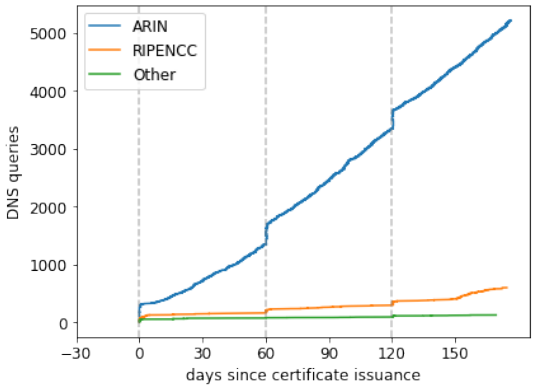

Figure 4.26: Average cumulative number of DNS queries per registry 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

Most of the DNS traffic comes from queriers in the ARIN registry, accounting for 86.1 % of the traffic. We see for the top 3 registries that traffic surges at the certificate reissuances. Traffic from AFRINIC and LACNIC is too low to detect any surges. DNS traffic from RIPENCC registry increases outside of reissuances starting on day 150. The increase in traffic can also be found in Figure 4.20.

Since we allocate a whole /64 subnet to the experiment, we examine whether DNS reverse lookups are detected for the unused addresses in the subnet. We do not detect any queries for an unused address in the allocated /64 networks of the experiment. Exposing a host via CT logs does not trigger DNS traffic for the rest of the subnetwork in IPv6.

4.1.5. Experiment 610b

In experiment 610b, we are hosting a domain on one address in IPv6 and generate a self-signed certificate for the domain locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv6, affects DNS traffic pertaining to the domain of the experiment. This experiment will be used as a baseline for comparison with experiment 610a. Experiment 610b was run 5 times and ran from March 12, 2021 to September 5, 2021. During the time period of the experiment, we do not detect any DNS traffic pertaining to the experiment at the authoritative DNS server.

4.1.6. Experiment 4p0c

In experiment 4p0c, we are hosting two domains on two IPv4 addresses in IPv4. Let's name one domain, domain 1, and one domain, domain 2. We craft a PTR record pointing to the IP address of domain 1, but with the domain name of domain 2. In this context, the goal of the experiment is to determine whether the PTR record of a domain, hosted in IPv4, affects DNS traffic pertaining to the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. We detect 1623 queries for domain 1 and 3497 queries for domain 2.

Figure 4.27 shows the cumulative number of DNS queries received for the two domains of the experiment. We observe 2 types of queries: queries for a domain name and for a reverse DNS lookup.

For domain 1, we only detect queries with reverse DNS lookups; we do not find the domain name in any queries. The domain name of domain 1 is kept private on the host: it cannot be retrieved outside the host. Since the domain is running on a web server, it is possible to guess its name by using the web

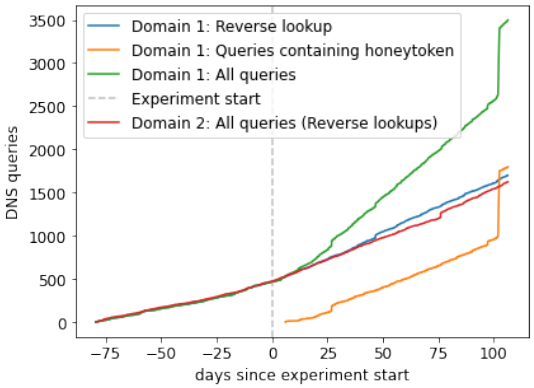

Figure 4.27: Cumulative number of DNS queries of the two domains in experiment 4p0c. The grey dotted line indicates the start of the experiment. For domain 1, we distinguish between reverse lookups and queries that contain the honeytoken in the requested domain. The domain can either be the exact domain name or a crafted domain name containing the honeytoken. 0x20 bit encoding is ignored. For domain 2, only reverse lookups were detected.

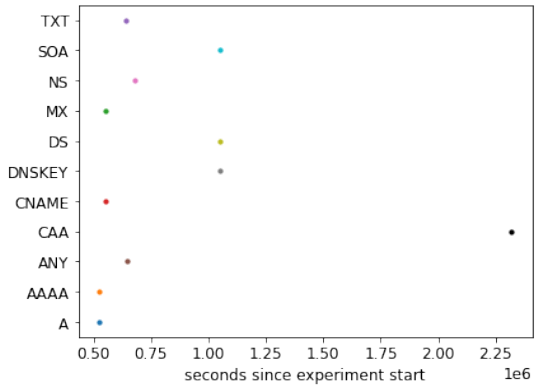

Figure 4.28: First DNS query containing honeytoken of experiment for every DNS query type of the two domains in experiment 4p0c. X-axis is log-scaled.

server as an oracle. The domain name is found when the web server successfully returns a webpage to the guesser. In Figure 4.27, DNS queries for domain 1 is growing at a linear rate and are not affected by the start of the experiment — the setup of the domains.

For domain 2, we detect the 3 types of queries. In Figure 4.27, the growth for reverse DNS lookups is nearly identical to the one for domain. The two lines diverge slightly after day 50. What differs between the two domains is the detection of queries for a domain name. After 6 days, we detect the first query with the domain name of domain 1. After the first query, the number of this type of query grows linearly until day 100, where it surges, nearly doubling. These queries are responsible for 52 % of the queries for domain 1. Within these queries, we find not only the domain name but also unexisting subdomain names. The subdomain names contain a subdomain, besides the honeytoken of the domain name. 37 % of the queried domain names are subdomains.

From these results, we find that domain names are leaked when the corresponding PTR record exists. The consequence is an increase in DNS queries, but only for queries with a domain name. Queries for reverse DNS lookups remain unaffected by the existence of the PTR record.

The first DNS query containing a domain name was detected 6 days after the start of the experiment. The query is of type A. Figure 4.28 shows the detection time of the domains for all the record types. Type CAA has the longest detection time of over 25 days. Type SOA, DS and DNSKEY have near identical detection times at 12 days. The remaining types are detected within 5 and 10 days. Note that all these queries are for domain 1, as domain 2 does not leak its domain name and therefore we do not detect any queries for it either.

Figure 4.29 shows the total number of unique IPs and ASes querying the domains of the experiment. Before the start of the experiment, we count the same number of ASes and IPs for both domains. After 5 days or equivalently after the first query with a domain name, the count between the domains start to differ. We observe a significant increase in new IPs and a minor increase in ASes for domain 1 compared to domain 2, where the count is only increasing linearly.

Using the threat intel from VirusTotal on the IP sources, we can differentiate between traffic from suspicious and benign sources. Figure 4.30 shows only traffic from suspicious IPs. We only detect few queries from suspicious addresses before the start of the experiment. After setting up the domains, we start to detect more queries from suspicious sources; we detect more for domain 1 than domain 2.

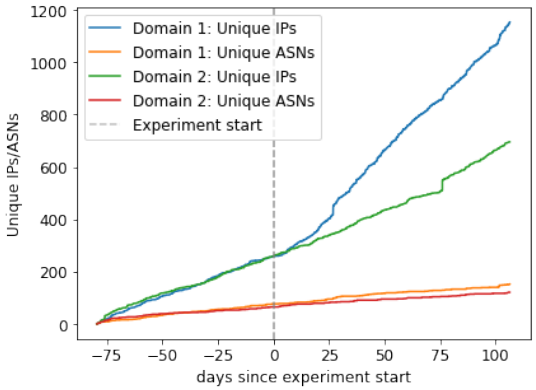

Figure 4.29: Cumulative number of unique IPs and ASes making queries for the domains in experiment 4p0c. The grey dotted line indicates the start of the experiment.

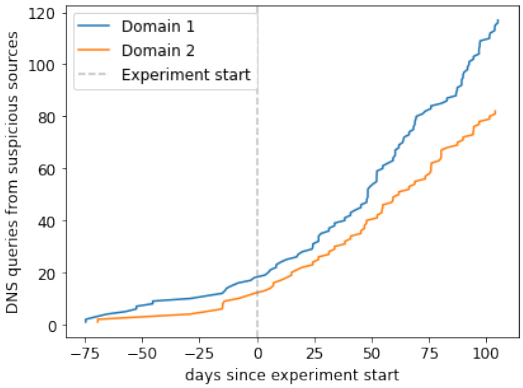

Figure 4.30: Cumulative number of DNS queries from suspicious sources of the two domains in experiment 4p0c. The grey dotted line indicates the start of the experiment.

The only thing that changed before and after the start is that for 2 IPs, the DNS server replies to the queries with a valid response. Therefore, adding entries in a DNS server significantly increases DNS traffic from suspicious traffic.

The findings show that setting up the DNS configuration of a domain following the guidelines [7] exposes the domain name to the public. The DNS configuration becomes a source of leakage for the domain: by performing reverse lookups, domain names can be discovered. We will take into consideration that, in IPv4, PTR records are a source of leakage in the synthesis in 4.4.1.

4.1.7. Experiment 6p0c

In experiment 6p0c, we are hosting two domains on two IPv6 addresses in a dedicated /64 subnetwork. Let's name one domain, domain 1, and one domain, domain 2. We craft a PTR record pointing to the IP address of domain 1, but with the domain name of domain 2. In this context, the goal of the experiment is to determine whether the PTR record of a domain, hosted in IPv6, affects DNS traffic pertaining to the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. During the time period of the experiment, we do not detect any DNS traffic pertaining to the experiment at the authoritative DNS server. The experiment shows that even when setting up the DNS configuration of a domain following the guidelines [7], where for every domain a PTR record must be created, no reverse lookups were detected and therefore the domain was not leaked. The lack of domain leak via PTR records simplifies the analysis of IPv6 experiments.

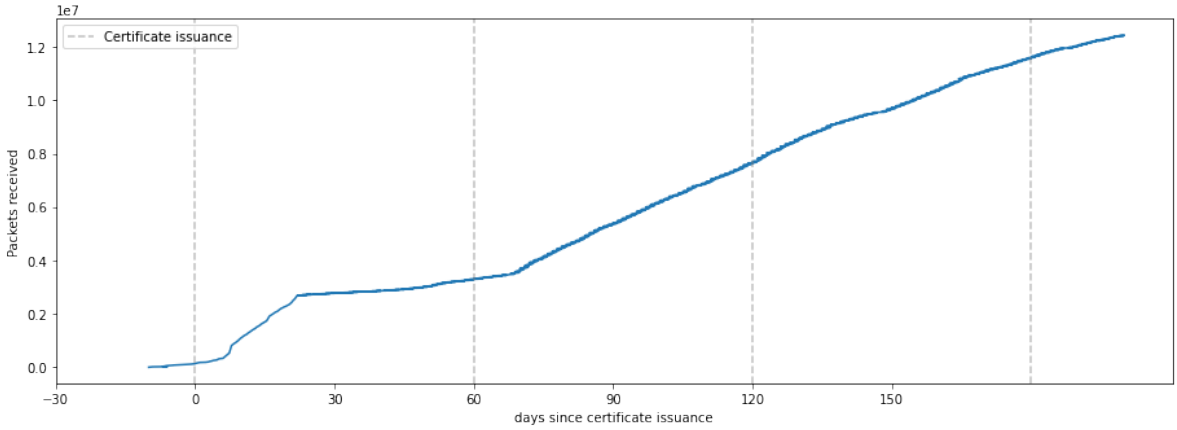

Figure 4.31: Cumulative number of packets in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

4.2. Network traffic analysis

In this section, we will analyse the packets received in the experiment networks. We start by describing the collected packet capture and we will extract the relevant packets to the experiments. We will analyse the packets to determine the effect of CT logs on scanning traffic.

4.2.1. Data description

We capture a total of 1.4 TB of packets in the IPv4 networks and 421MB in the IPv6 network. The packet capture spans from January 16, 2021 to October 31, 2021. The packets are stored in the PCAP file format.

4.2.2. Experiment 410a

In experiment 410a, we are hosting a domain on one address in IPv4 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv4, via CT affects scanning traffic at the host of the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. We collect 13M packets for a total of 424 MB at the host of the experiment.

Figure 4.31 shows the cumulative count of packets received at the host on the relative timeline, with day 0 being the time when the certificate of the experiment domain is issued and leaked to a CT by the CA. There are 3 major increases in traffic: on day 10, traffic sharply increases until day 20; for the next 50 days, traffic is slow; from day 70, traffic increases at a similar rate as on day 10, but for a much longer period of 140 days; at the end of the long slope, traffic surges at a considerably higher rate than observed before. At 60 day intervals, the certificate of the host is renewed. The first 2 increases in traffic occur about 10 days after the initial and second issuance. It is not clear whether the increase was caused by the issuance: we will further investigate in Section 4.4.1, where we compare the figure with the control experiment.

Packets are skewed towards the TCP protocol, accounting for 97.9 % of the whole traffic. Only 1.5 % of the packets are sent via UDP and 0.5 % of the packets are ICMP. The remaining combined are less than 1 % of the traffic; we find protocols GRE, SCTP, HOPOPT, DCCP.

In Figure 4.32, we show the distribution of received packets at the destination ports of the experiment host. Note that the host of the experiment ran behind a firewall and therefore, we do not detect packets for some port numbers, see Section 3.6 for more information. We find the most hit ports are within

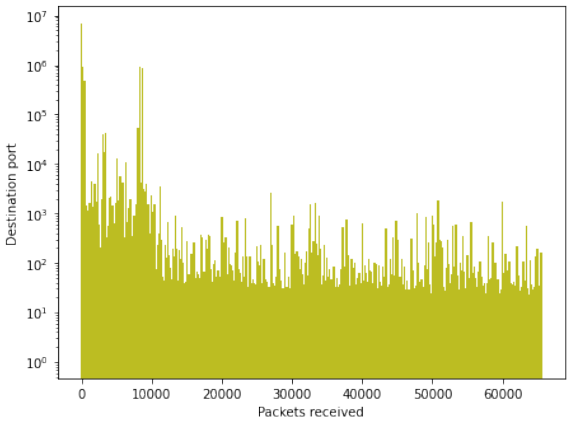

Figure 4.32: Cumulative number of packets per destination ports in experiment 410a. We do not distinguish between TCP and UDP. Y-axis is log-scaled.

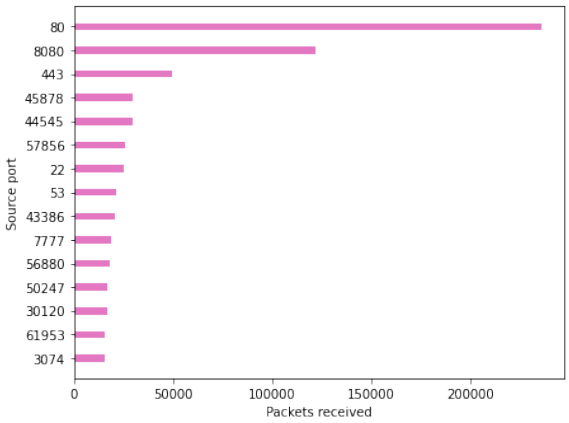

Figure 4.33: Number of packets received at the top 15 destination ports in experiment 410a. Red bars show UDP ports, the remaining yellow bars are TCP ports. X-axis is log-scaled.

the 1 to 10000 port range. In the remaining port range, the packet count varies between 10 and 1000. Figure 4.33 gives us more detail on the highest hit ports. The most hit port is port 22 with more than 1M hits. The next most hit ports are port 80, 443, 8291 and 8728, in that order, but the packet count for these ports is smaller than for port 22 by a factor of 10 or more. SSH, running on port 22, comprises 68.1 % of the traffic. Out of the top 15 most hit ports, only 6 are from the well-known port range from 1 to 1024. Traffic to port 80, ie. HTTP traffic is more prevalent than its secured version HTTPS on port 443. Since the experiment domain is leaked via CT log and therefore a certificate was issued, the domain needs to be accessible via HTTPs. However, we still see more HTTP traffic than HTTPS. Currently, we are not looking at the content of the HTTP(s) traffic: even if packets are detected on ports 80 or 443 of the experiment host, it does not indicate that the packets were directed to the experiment's domain. We will further investigate HTTP(s) traffic in the following Section 4.3. Among the top 5 most hit ports, we find 2 port numbers outside the well-known ports: 8291 and 8728. These ports are commonly used as GUI and API access point by the operating system RouterOS — an OS for routers.

Figure 4.34 shows the number of bytes received at the destination ports receiving the most traffic. We find that packets to port 5060 are considerably larger (350 bytes) compared to packets at the other ports. On this port, we commonly find the Session Initiation Protocol (SIP), which is used by VoIP software. When comparing to the packet count in Figure 4.33, we find that some ports in the lower range – port 389 and 1900 – now appear in the top 15 ports with most bytes received because of their large packet size. These ports are commonly used for the protocols LDAP and UPnP/SSDP, respectively.

We perform the same analysis on the source ports. In Figure 4.36, we show the complete distribution of packets received on the source port range. We find multiple clusters of ports. Two clusters, centered on 25000 and 50000, are in the ephemeral port range specified by RFC 6056, from 1024 to 65535. The most used source ports are in the cluster from 1-10000. In this cluster, traffic is more concentrated on a small number of ports, compared to the cluster centered on 50000. In Figure 4.37, we find port 80, 443, 22 and 123 ports — usually reserved for common services (HTTP, HTTPS, SSH and NTP) — used for sending packets. To send packets from ports in the reserve range, elevated privileges are needed, unlike for the ports outside the reserved range. Out of the 15 ports with the most traffic, 2 ports, 56880 and 55438, are from the Dynamic/Private port range. The remaining are in the registered port range.

With Figure 4.38, we tie destination and source ports to show their relation. For the most hit destination ports, we detect the full range of source ports. We find numerous vertical lines, which show full port scans using the same source port. There are some curves, the largest one starting on day 30000, showing a full port scan but where the source port was incremented after each scan. Using deterministic strategy for the source port is easily detectable compared to randomized source ports.

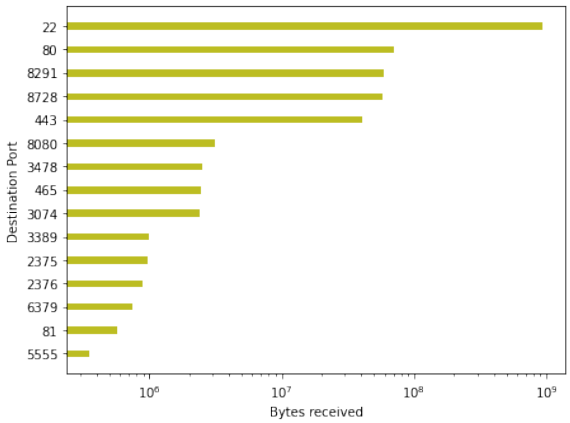

Figure 4.34: Bytes received at the top 15 destination ports in experiment 410a. X-axis is log-scaled.

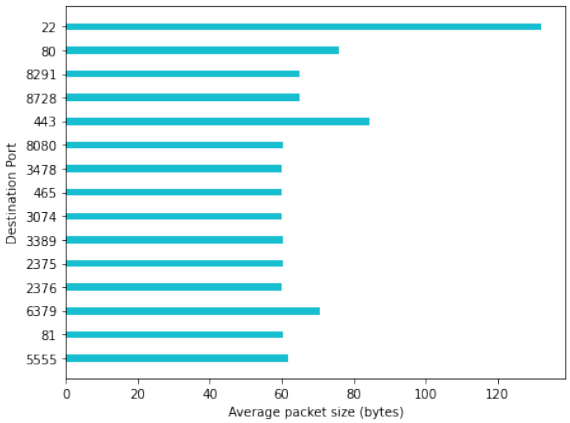

Figure 4.35: Average packet size at the top 15 destination ports in experiment 410a.

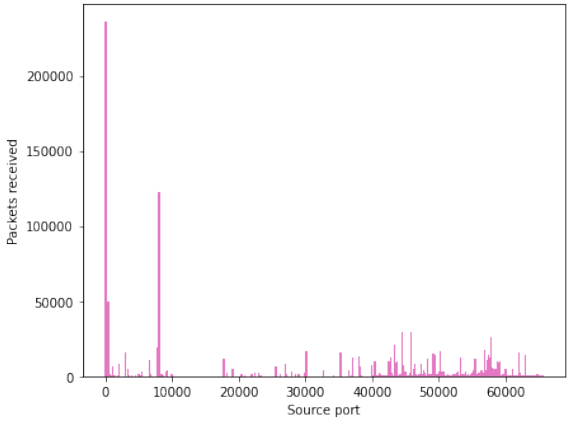

Figure 4.36: Cumulative number of packets per source port in experiment 410a.

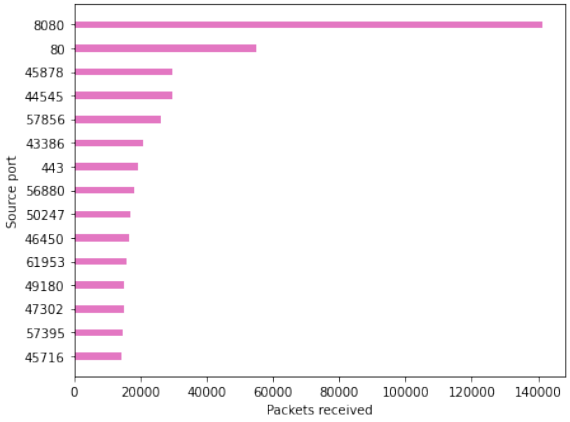

Figure 4.37: Number of packets received from the top 15 source ports in experiment 410a. X-axis is log-scaled.

We find a high concentration in the source port range 40000 to 60000. These are ephemeral ports used for short term connections. We find numerous full port range scans using these ports.

We perform the same analysis on the UDP traffic in Figure 4.39. Most data points are concentrated between source ports 30000 and 60000, also ephemeral ports. We do not detect as many full port scans as for TCP in Figure 4.38. For the most hit UDP ports, source ports are distributed over the whole range with bias in the 30000 to 60000 range.

Figure 4.40 shows the distribution of packets in the IPv4 address space. The address space is divided into 256 /8 subnets. In Figure 4.40, we see that the traffic is not uniformly distributed in the address space. We find a couple of subnets that generate a lot of traffic, notably bins 5, 223, 45 and 61 accounting for 37.1 %, 7.0 %, 4.9 % and 4.6 % of the traffic, respectively. Similarly, we find in Figure 4.41, that the volume of traffic is not uniformly distributed but concentrated in a couple of subnets. When comparing both figures, we find that traffic from subnet 45 comprises mainly smaller packets, as is traffic coming from the subnets 64 to 200. We will use these results in conjunction with the control experiment 410b to determine whether the originator of the traffic at the hosts changes with the involvement of CT logs.

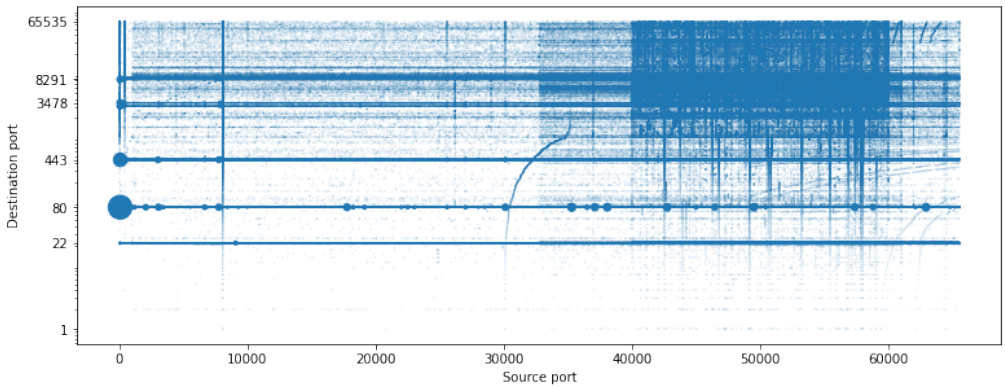

Figure 4.38: Relation between TCP source and destination ports in experiment 410a. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

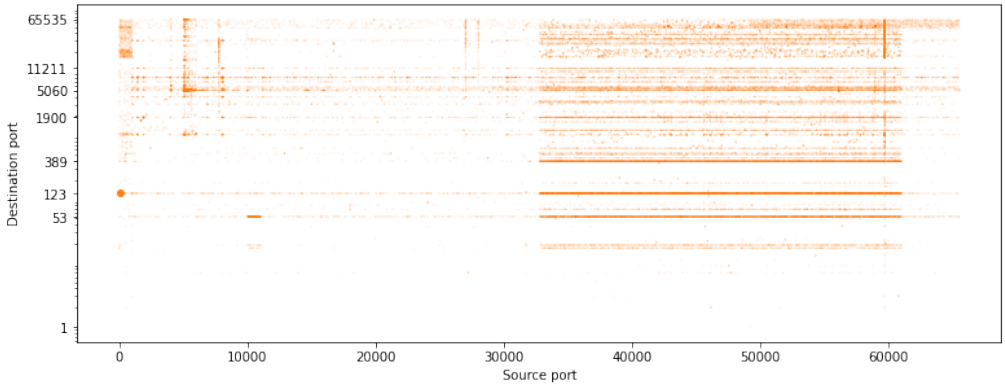

Figure 4.39: Relation between UDP source and destination ports in experiment 410a. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

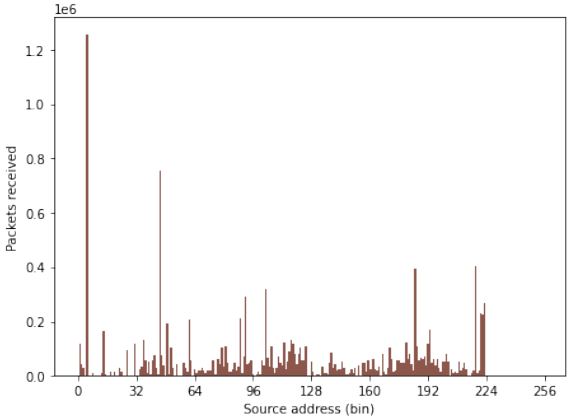

Figure 4.40: Number of packets received from source address bins in experiment 410a. The address space is split into 256 bins of /8 subnets.

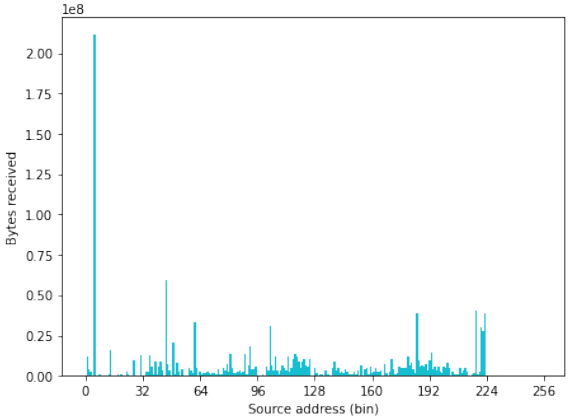

Figure 4.41: Bytes received from source address bins in experiment 410a. The address space is split into 256 bins of /8 subnets.

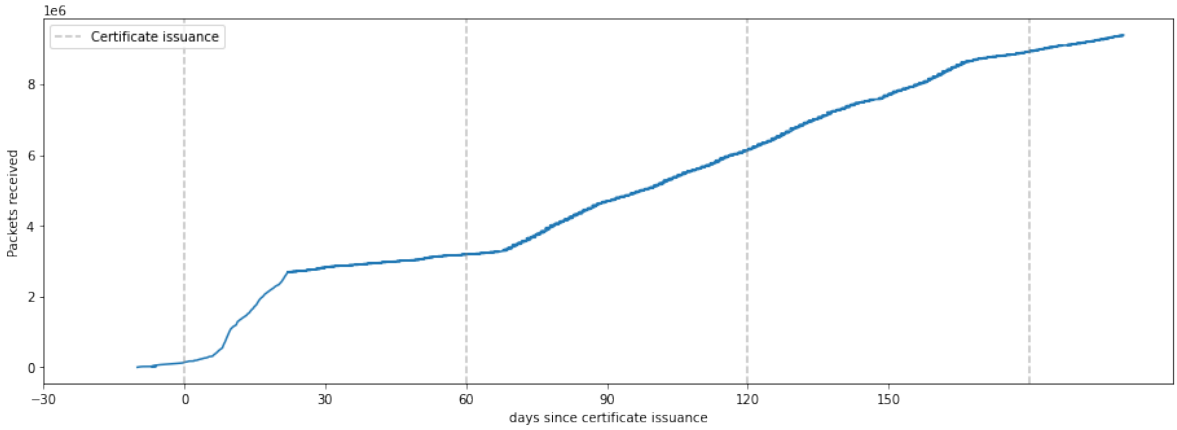

Figure 4.42: Cumulative number of packets in experiment 410b. The grey dotted line indicates when the certificate was issued.

4.2.3. Experiment 410b

In experiment 410b, we are hosting a domain on one address in IPv4 and generate a self-signed certificate locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv4, affects scanning traffic at the host of the domain. This experiment will be used as a baseline for comparison with experiment 410a. Experiment 410b was run once and ran from Mar 3, 2021 to Sep 5, 2021. We collect 9M packets for a total of 419 MB at the host of the experiment.

Figure 4.42 shows the cumulative count of packets received at the host on the relative timeline, with day 0 being the time of certificate generation on the host. There are 3 major increases in traffic: on day 10, traffic sharply increases until day 20; for the next 50 days, traffic is slow; from day 70, traffic increases at a similar rate as on day 10, but for a much longer period of 140 days; at the end of the long slope, traffic surges at a considerably higher rate than observed before. The shape of the traffic in experiment 410b strongly resembles the shape in experiment 410a.

Packets are skewed towards the TCP protocol, accounting for 96.7 % of the whole traffic. Only 1.5 % of the packets are sent via UDP and 0.5 % of the packets are ICMP. The remaining combined are less than 1 % of the traffic; we find protocols GRE, SCTP, HOPOPT and DCCP.

In Figure 4.43, we show the distribution of received packets at the destination ports of the experiment host. Note that the host of the experiment ran behind a firewall and therefore, we do not detect packets for some port numbers, see Section 3.6 for more information. We find slight differences in Figure 4.44, which gives us more detail on the highest hit ports. The difference lies in the lower 10 ports in the top 15. We find a different order, a slightly different subset of ports and slightly different magnitudes.

Figure 4.45 shows the number of bytes received at the destination ports receiving the most traffic. Similarly to Figure 4.34, we obtain similar results to experiment 410a: there are slight differences in order and magnitude. Additionally, we find port 1433, which is used by Microsoft SQL servers. We will go into further detail in Section 4.4.1, where we compare the main experiment 410a with the control experiment 410b.

We perform the same analysis on source ports. In Figure 4.47, we show the complete distribution of packets received on the source port range. We find multiple clusters of ports. Two clusters are centered under port 10000; they each have the highest peaks in packets received compared to the rest of the port space. We find a smaller cluster centered on 30000 and a larger cluster centered on 50000. The latter cluster has a more uniform distribution compared to the other clusters. Three of the clusters are in ephemeral port range specified by RFC 6056. Figure 4.48 gives more information on the most used source ports: we find port 80, 443, 22 and 123, ports — usually reserved for common services (HTTP, HTTPS, SSH and NTP) — used for sending packets. To send packets from ports in the reserve

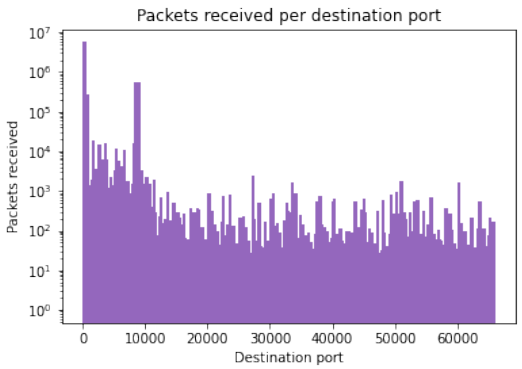

Figure 4.43: Number of packets received per destination port in experiment 410b. Y-axis is log-scaled.

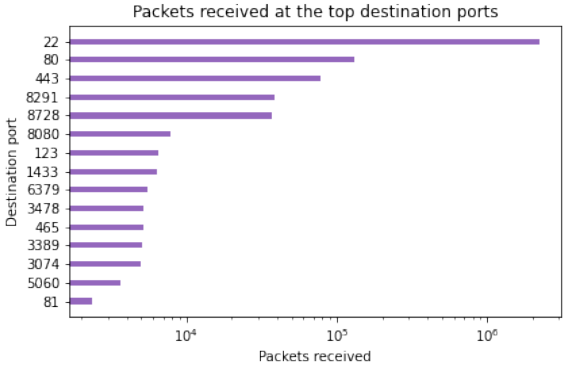

Figure 4.44: Number of packets received at the top 15 destination ports in experiment 410b. X-axis is log-scaled.

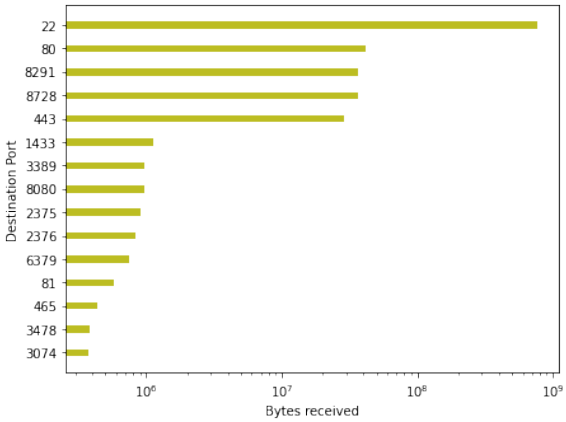

Figure 4.45: Bytes received at the top 15 destination ports in experiment 410b. X-axis is log-scaled.

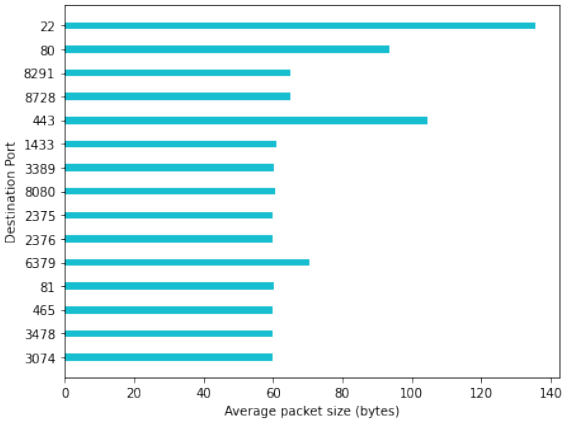

Figure 4.46: Average packet size at the top 15 destination ports in experiment 410b.

range, elevated privileges are needed, unlike for the ports outside the reserved range. Out of the 15 ports with the most traffic, 2 ports, 56880 and 55438, are from the Dynamic/Private port range. The remaining are in the registered port range.

With Figure 4.49, we tie destination and source ports to show their relation. For the most hit destination ports, we detect the full range of source ports. We find numerous vertical lines, which show full port scans using the same source port. There are some curves, the largest one starting on day 26000, showing a full port scan but where the source port was incremented after each scan. Using deterministic strategy for the source port is easily detectable compared to randomized source ports. We find a high concentration in the source port range 40000 to 60000. These are ephemeral ports used for short term connections. We find numerous full port range scans using these ports.

We perform the same analysis on the UDP traffic in Figure 4.50. Most data points are concentrated between source ports 30000 and 60000, also ephemeral ports. We do not detect as many full port scans as for TCP in Figure 4.49. For the most hit UDP ports, source ports are distributed over the whole range with bias in the 30000 to 60000 range.

Figure 4.51 shows the distribution of packets and bytes in the IPv4 address space. The address space is divided into 256 /8 subnets. In Figure 4.51, we see that the traffic is not uniformly distributed in the address space. We find a couple of subnets that generate a lot of traffic, notably bins 5, 134, 45 and 61. If we disregard the subnets that generate the most traffic, we find that from subnet 32 to 224, the distribution looks uniform. Similarly, we find in Figure 4.52, that the volume of traffic is not uniformly distributed but concentrated in a couple of subnets. When comparing both figures, we find that traffic

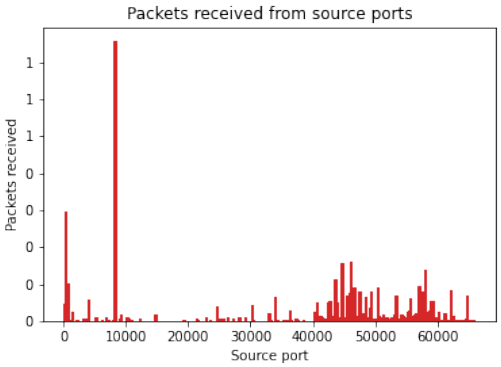

Figure 4.47: Cumulative number of packets per source port in experiment 410b.

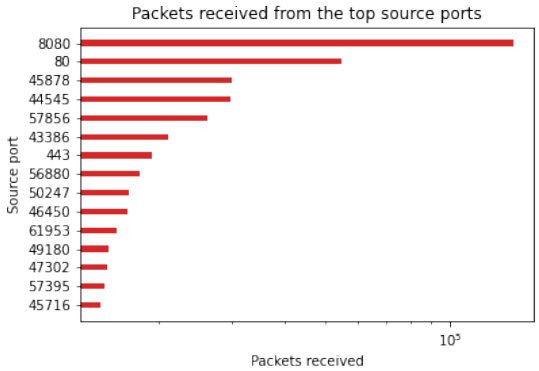

Figure 4.48: Number of packets received from the top 15 source ports in experiment 410b. X-axis is log-scaled.

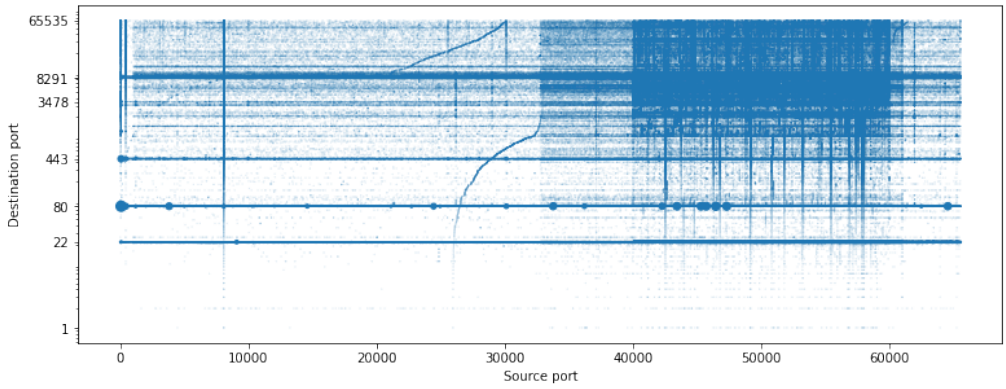

Figure 4.49: Relation between TCP source and destination ports in experiment 410b. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

from subnet 128 mainly sends smaller packets, as we do not see any peak in Figure 4.52. Subnet 221 shows the opposite behaviour, as in not many packets are sent, but the packets are of larger size. We will use these results in conjunction with the control experiment 410b to determine whether the originator of the traffic at the hosts changes with the involvement of CT logs.

4.2.4. Experiment 610a

In experiment 610a, we are hosting a domain on one address in IPv6 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv6, via CT affects network traffic at the host of the domain. Experiment 610a was run 3[2] times and ran from March 12, 2021 to September 5, 2021. We collect 8113 packets for a total of 812 MB at the 3 hosts of the experiment.

Figure 4.53 shows the average cumulative number of packets received at the three hosts for the different types of IP protocols. We detect the first packets at the 3 hosts on day 20 after the certificate issuance, i.e. after the domain leak via CT. The standard deviation is depicted in the shaded area. We see that the traffic is near identical up to day 95, where one run experiences a small surge. Disregarding the surge, we find that the traffic grows linearly and is nearly identical in all three runs. On day 210, we detect a surge in all three runs at the same time, but with each with different magnitudes. The

[2]The Experiment 610a was run 5 times, but 2 runs failed because of an issue in the certificate issuance process

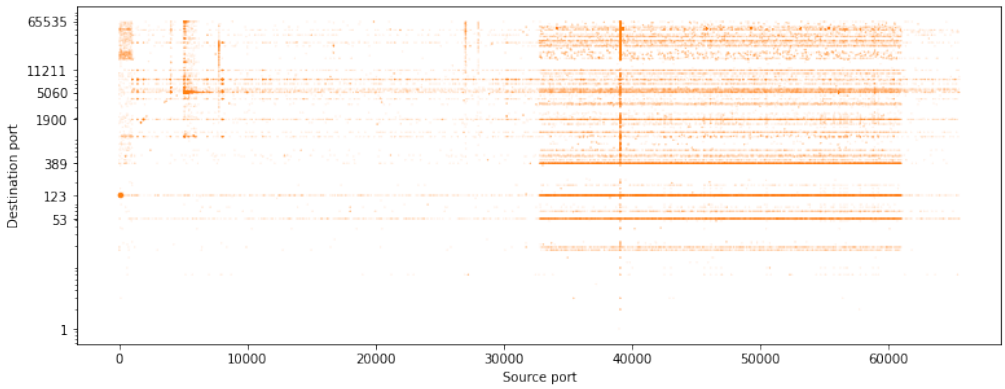

Figure 4.50: Relation between UDP source and destination ports in experiment 410b. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

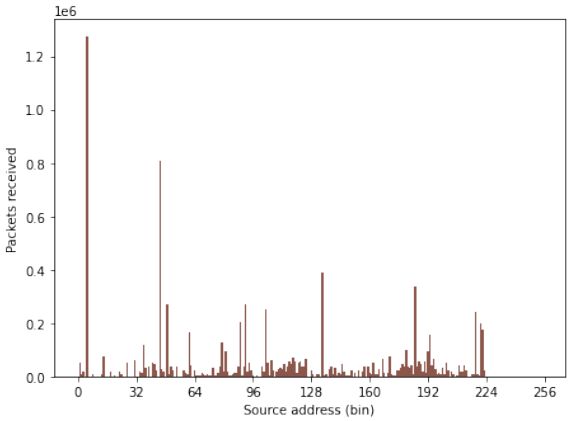

Figure 4.51: Number of packets received from source address bins in experiment 410b. The address space is split into 256 bins of /8 subnets.

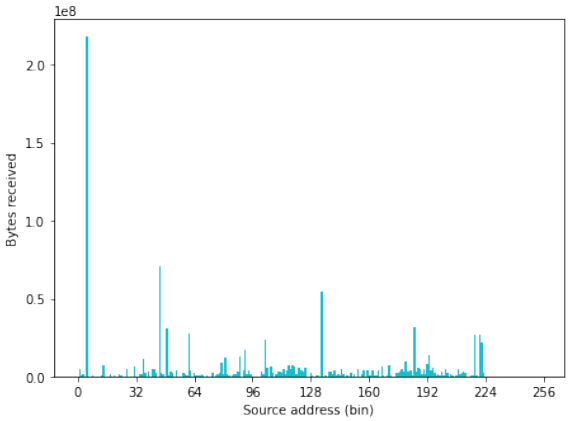

Figure 4.52: Bytes received from source address bins in experiment 410b. The address space is split into 256 bins of /8 subnets.

deviations seen in the overall traffic are attributed to IP protocol 6, also called TCP protocol. Packets are skewed towards the TCP protocol, accounting for 75.7 % of the whole traffic. ICMP (IP protocol 58) packets make up 18.1 %. The remaining packets are sent via UDP (IP protocol 17) and account for 6.1 %. Traffic from the two latter protocols grows linearly with very little deviations between the runs. We do not detect any other protocols. The low deviation between the runs shows that the setup of the experiment results in a rather predictable outcome.

We investigate the source of the packets by looking at the traffic generated per AS. In Figure 4.54, we show the average number of packets received per AS. The black bar indicates the standard deviation. 47 % of the traffic is originating from 1 AS and 90 % of the traffic is only generated by 6 ASes. Among these ASes, we find two that show a higher deviation: AS 6939 and AS 38283. AS 6939 is responsible for the difference in magnitude of the surge on day 210. AS 38283 is responsible for the surge on day 95 that only occurred in one run. These two ASes caused nearly all the deviation between the runs, as seen in Figure 4.53.

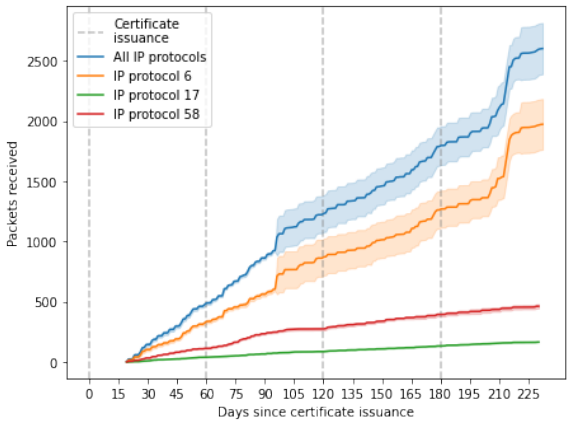

Figure 4.53: Average number of packets received per protocol in the 3 runs of experiment 610a. The shaded area represents the standard deviation. The grey dotted lines indicate when the certificate of the domain was issued or reissued. IP protocol 6 is TCP; IP protocol 17 is UDP; IP protocol 58 is ICMP.

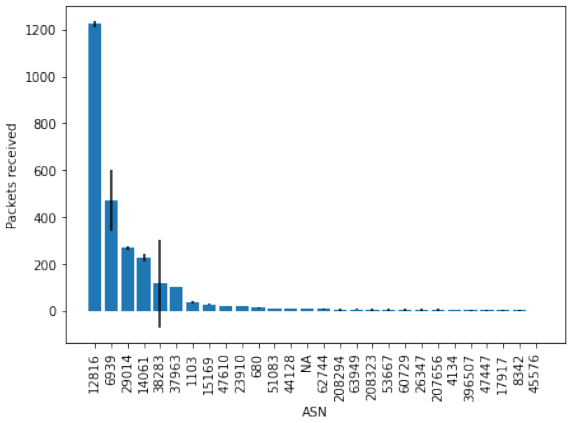

Figure 4.54: Average number of packets received per AS in the 3 runs of experiment 610a. The black lines represent the standard deviations. The ASN denoted by 'NA' groups all traffic where no information on the AS was found.

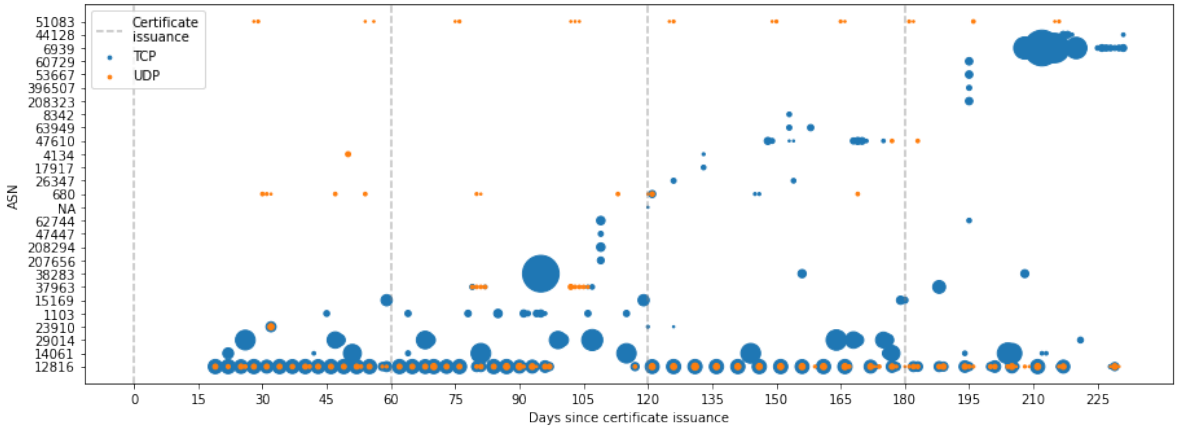

Figure 4.55: Number of daily packets per AS in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.55 shows the number of daily packets per AS in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. We can see how each AS behaves: some scan our hosts periodically, some have irregular patterns, others only scan once. The bottom AS (12816) shows the most predictable scanning pattern, scanning both TCP and UDP ports at a constant interval. There are 2 phases in his scanning separated by a minor break between days 100 and 115 and differentiated by the periodicity. We find two groups of 4 ASes that scan the host exclusively on the same day, day 107 and 195. That might be an indication that the source addresses generating the traffic in these ASes are controlled by a single actor.

We move our focus from the originator of the traffic to the properties of TCP and UDP traffic by looking at the source and destination ports used by the incoming traffic. In Figure 4.56, we show the average number of packets received per source port in the 3 runs of experiment 610a. The black lines represent the standard deviations. Only the 10 source ports generating the most traffic are displayed. There is a lot of variability between the runs, as we detect that most standard deviations are greater than the mean. It shows that the sources of the traffic do not use the same source ports to send traffic to the experiment hosts. Furthermore, the most detected source ports are all within the ephemeral ports, which are usually used for short-lived communications. It is likely that the sources use randomized

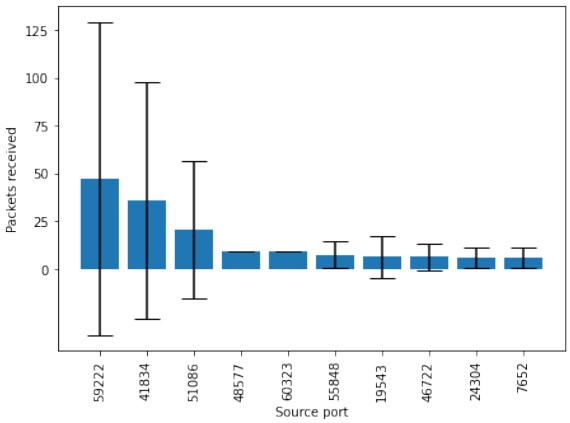

Figure 4.56: Average number of packets received per source port in the 3 runs of experiment 610a. The black lines represent the standard deviations. Only the 10 source ports generating the most traffic are displayed.

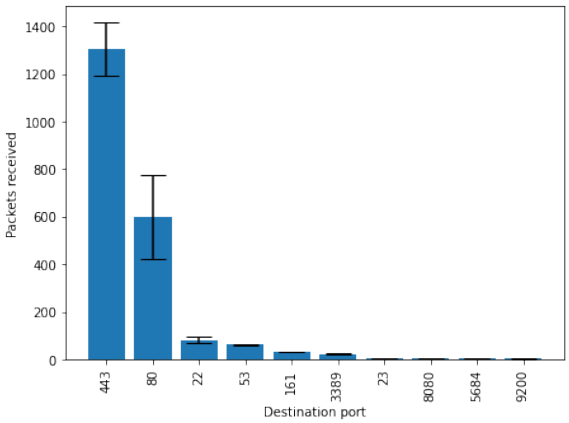

Figure 4.57: Average number of packets received per destination port in the 3 runs of experiment 610a. The black lines represent the standard deviations. Only the 10 destination ports generating the most traffic are displayed.

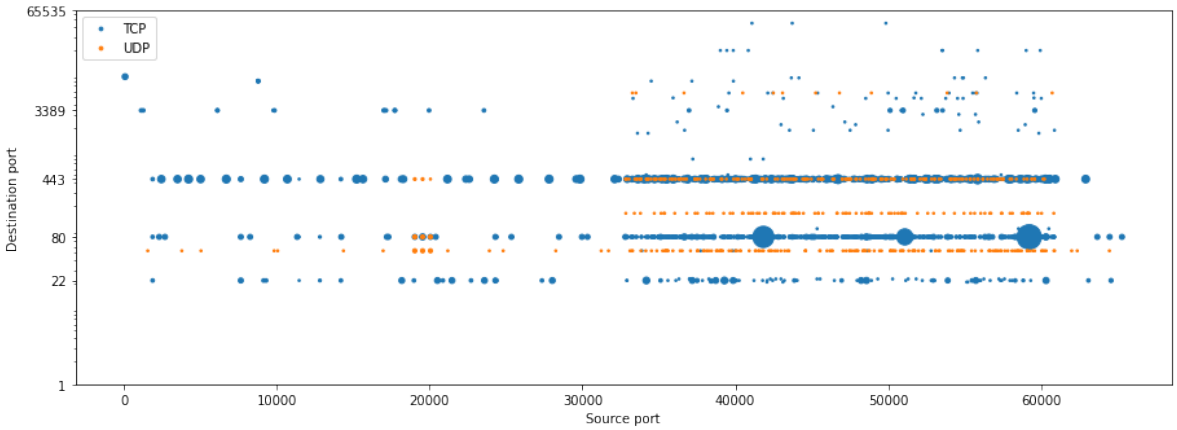

Figure 4.58: Relation between source and destination ports in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

port numbers within the ephemeral ports to establish connections with our hosts. Therefore, we do not detect a consistent set of source ports used in each of the runs.

Figure 4.59 shows the number of daily packets per source port in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. We notice that most data points are within a block, in the 35000 to 60000 port range, which is part of the ephemeral ports. The points within that block are uniformly distributed in the port range, confirming that source ports are randomly selected. We notice small clusters of UDP ports around port number 20000 on days 80 and 100. Cross-referencing with Figure 4.55, we see that they belong to AS 38283. No other AS shows such a distinct source port selection.

Figure 4.57 shows the average number of packets received per destination port in the 3 runs of experiment 610a. The black lines represent the standard deviations. Only the 10 destination ports generating the most traffic are displayed. The 3 most targeted ports, ports 443, 80 and 22, attract 92 % of the traffic. Note that these are also the only ports that are open on the experiment hosts. On these ports, full TCP connections can be established and therefore generate more traffic than closed ports. A similar argument can be made for ports 443, HTTPs, and 80, HTTP, because HTTPS connections have

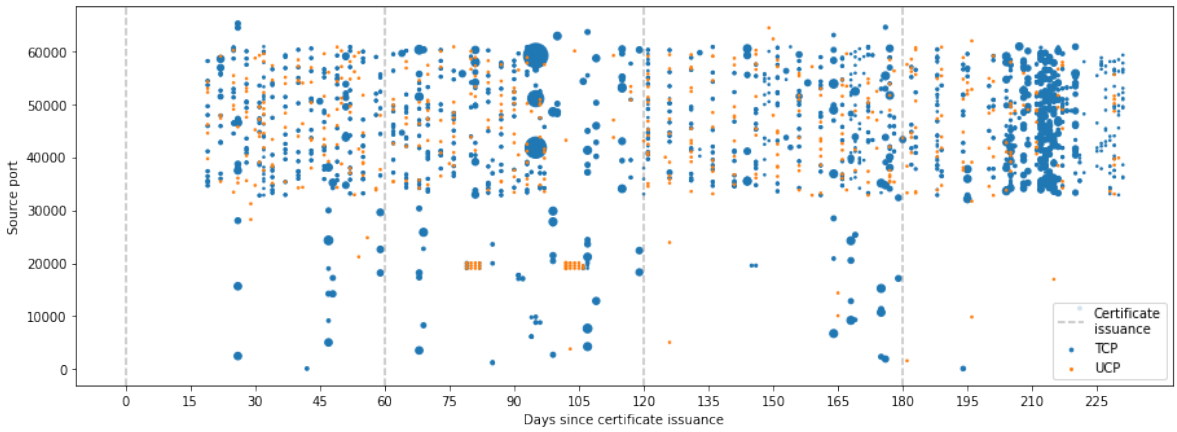

Figure 4.59: Number of daily packets per source port in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

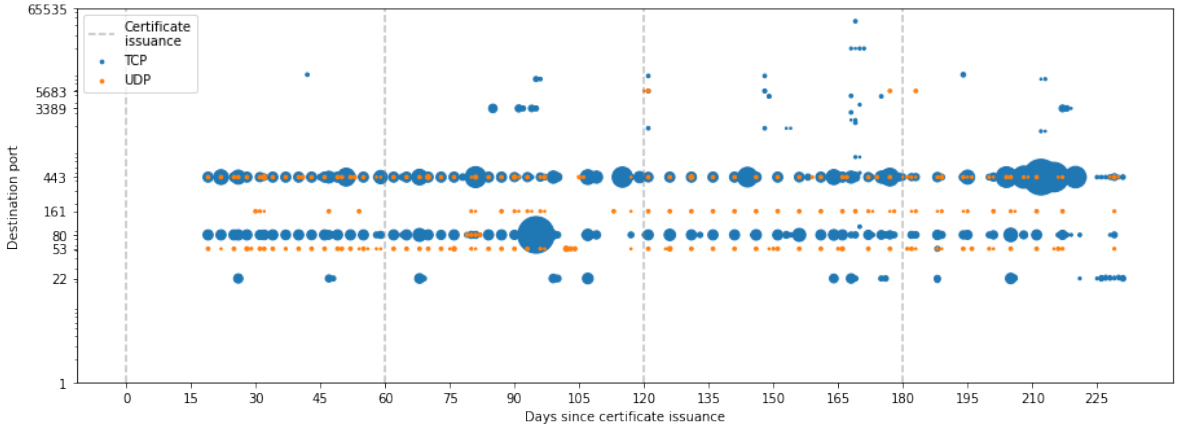

Figure 4.60: Number of daily packets per destination port in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

an additional overhead with TLS protocol to secure the communication. We will further investigate the HTTP(s) traffic without the TLS overhead in Section 4.3.4.

Figure 4.60 shows the number of daily packets per destination port in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports. We find consistent traffic for ports 53, 80, 161 and 443. For the HTTP(S) ports, consistent traffic is expected compared to ports 53 and 161, which are not even open on the hosts. Traffic to the SSH port appears at an irregular interval, even though the port is open. Even though HTTP(S) ports are heavily targeted, SSH ports do not experience the same. Around days 90 and 220, we find a cluster of packets to port 3389, scanning for vulnerable MikroTik routers.

With Figure 4.58, we tie destination and source ports to show their relation in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. We differentiate between UDP and TCP with different colours. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports. For the most hit destination

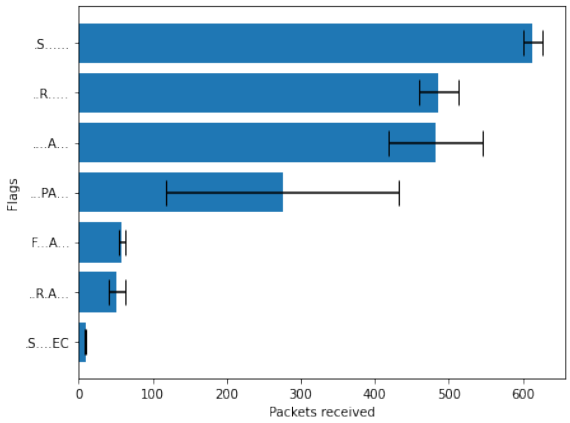

Figure 4.61: Average number of packets received per TCP flag configuration in the 3 runs of experiment 610a. The black lines represent the standard deviations. The '.' in the Y-label indicates that the flag is not set. For the set flags, we use the first letter of the flag name as abbreviation.

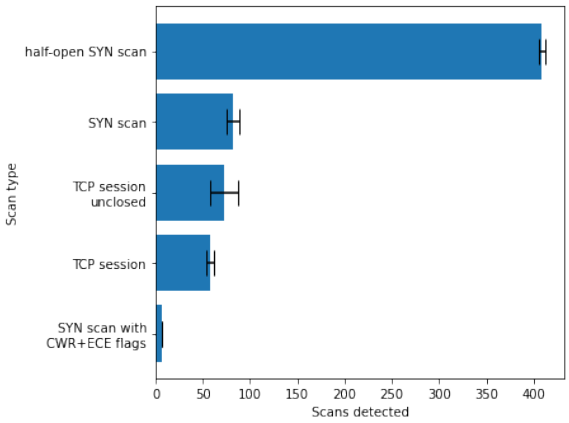

Figure 4.62: Average number of packets received per scan type in the 3 runs of experiment 610a. The black lines represent the standard deviations.

ports, we find source ports from the whole port range, but with an emphasis in the 30000 to 60000 port range. Destination ports above 500 tend to receive packets only from that range too.

We extract the flags from TCP packets in Figure 4.61, where we show the average number of packets received per TCP flag configuration in the 3 runs of experiment 610a. The black lines represent the standard deviations. The '.' in the Y-label indicates that the flag is not set. For the set flags, we use the first letter of the flag name as abbreviation. The most common packets are SYN packets accounting for 31 % of the TCP traffic, followed by RST and ACK packets accounting each for 24 %. We find the highest deviation in ACK-PSH, which is detected mostly in HTTP(S) traffic, to push the request to the web server. Note that we detected the highest deviations in the HTTP(S) ports in Figure 4.57. All first 6 flag configurations are common in a standard TCP session. The least detected flag configuration with 0.5 % is the EWE-CWR-SYN configuration. The first two flags are used for TCP congestion control. However, these packets were not detected as part of a TCP session. The congestion flags might have been set as an evasion method to hide that a SYN scan was performed.

Figure 4.63 shows the number of daily packets per TCP flag configuration in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. The size of dots gives a qualitative representation of the number of packets. SYN and RST packets have a relatively uniform distribution over the whole experiment duration. We notice that FIN-ACK, RST-ACK and PSH-ACK often occur on the same days. Since these flags are commonly used in TCP connections, it might indicate full TCP connections.

Instead of examining the TCP flags individually per packets, we classify each TCP connection based on the flags detected. Figure 4.62 shows the average number of TCP connection in the 3 runs of experiment 610a. The most common is the half-open SYN scan, accounting for 65 % of all the connections. Half-open SYN scan consists a SYN packet followed by a RST packet. Half-open SYN scans are used to probe for open ports and finish by closing the connection with RST to not keep the target waiting and waste its resources. This scanning method arouses less suspicion than the more simple SYN scan that does not close the connection. The target does not need to keep track of the connection created by the scan, as they are immediately closed by the follow-up RST. There is less risk that the resources of the target are overwhelmed, leading to a DoS attack. The simple SYN scan accounts for 13 %. 20 % of the connections are fully established TCP sessions at the open ports. Approximately half of the TCP sessions are not properly closed. In last place, accounting for 1 %, are SYN scans with congestion flags set. Since they occur isolated from other packets, we classify them as SYN scans. Looking at the deviations for the connections, we notice that the deviations are significantly lower than in Figure 4.61, where we looked at the TCP flags of packets individually. The high deviation that was detected is due

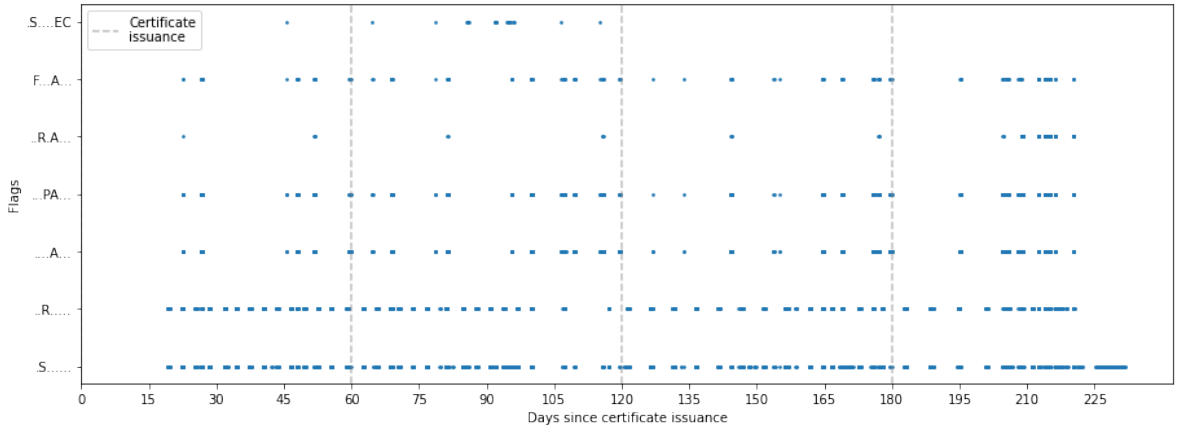

Figure 4.63: Number of daily packets per TCP flag configuration in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. The size of dots gives a qualitative representation of the number of packets. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The '.' in the Y-label indicate that the flag is not set. For the set flags, we use the first letter of the flag name as abbreviation.

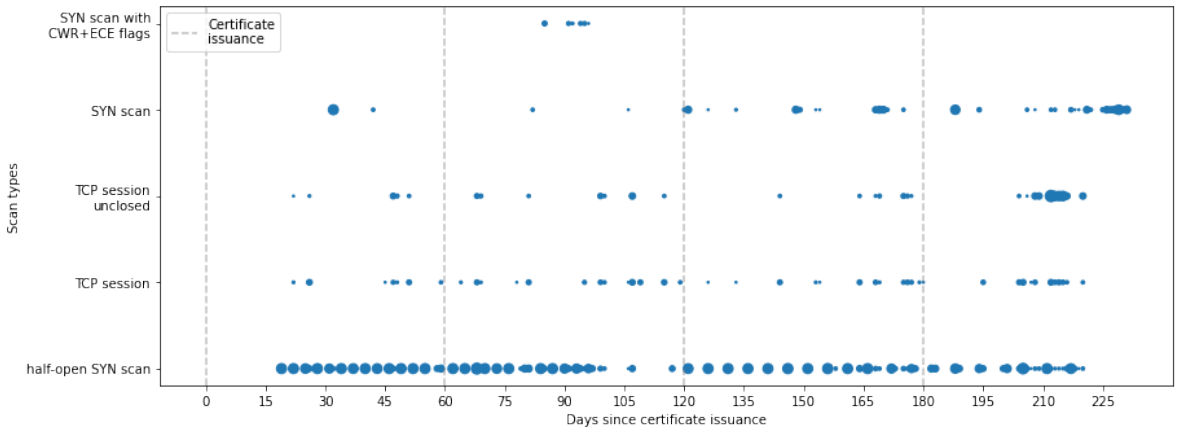

Figure 4.64: Number of daily packets per scan type in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. The size of dots gives a qualitative representation of the number of packets. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

to the TCP sessions. Sessions do not consist of 1 or 2 packets, like SYN scans, hence the difference in deviation.

Figure 4.63 shows the number of daily TCP connections in the 3 runs of experiment 610a. Note that the 3 runs are aggregated. The size of dots gives a qualitative representation of the number of packets. SYN and RST packets have a relatively uniform distribution over the whole experiment duration. The FIN-ACK, RST-ACK and PSH-ACK that occur on the same day as seen in Figure 4.63 are part of full TCP sessions, shows by 'Connection' and 'Connection no FIN'. Both types of sessions have a tendency to occur on the same days. Half-open SYN scans are detected at a regular interval throughout the duration of the experiment. Simple SYN scans occur at less regular intervals. The peculiar SYN scan with TCP congestion flags is concentrated around day 90.

Overall, we find consistent results between the 3 runs of the experiment. Packets are only detected 20 days after the leak of the domain name of the host via CT. The most hit ports are HTTP(S) ports on which we serve the website under the CT-leaked domain name of the experiment. We do not detect any specific phenomena at the reissuances, where the certificate is renewed and added to a CT log, therefore leaking the domain name. It shows that third parties are monitoring CT logs and keep track of all the encountered domain names. Subsequent addition of certificates from reissuances is ignored.

In Section 4.4.2, we will compare the results with the control experiment 610b to isolate the effect of CT on the traffic.

4.2.5. Experiment 610b

In experiment 610b, we are hosting a domain on one address in IPv6 and generate a self-signed certificate for the domain locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv6, affects DNS traffic pertaining to the domain of the experiment. This experiment will be used as a baseline for comparison with experiment 610a. Experiment 610b was run 5 times and ran from March 12, 2021 to September 5, 2021. During the time period of the experiment, we do not detect any traffic at the hosts of the experiment.

Experiment	HTTP requests	HTTPS requests
410a	31269	39798
410b	39585	41703
4p0c	12417	2398
6p0c	0	0
610a	356	399

Table 4.2: Overview of HTTP(s) requests logged at the web servers of the experiments

4.3. Web server traffic analysis

In this section, we will analyse data collected at the web servers running on the experiment hosts. We start by giving an overview of the collected HTTP(S) traffic, then we proceed with the analysis of the data for all the experiments run.

4.3.1. Data description

We collect a total of 638 MB of HTTP(s) requests at the web servers from the experiments. The collected data consists of 2.2M HTTP requests and 28K HTTPS requests. It was collected from January 16, 2021 at 4:59:18 until October 30, 2021 11:25:25. Table 4.1 summarizes the data collected from the web servers, specifying the number of HTTP and HTTPS requests per experiment.

4.3.2. Experiment 410a

In experiment 410a, we are hosting a domain on one address in IPv4 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv4, via CT affects scanning traffic at the host of the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. We collect 18.1K HTTP(s) requests for a total of 39 MB at the web server of the experiment.

We distinguish between requests for the experiment domain and requests for other domains. Figure 4.65 shows requests for the experiment domains, with HTTP and HTTPS separated; we record 509 HTTP requests and 731 HTTPS requests. Before the CT leak, we do not detect any requests for the domain. The first HTTP request is received after 60 seconds and the first HTTPS request is after 84 seconds. After the first request, HTTPS traffic is growing linearly and seems not to be affected by the reissuances of the certificate. HTTP traffic does not experience the same growth and consistency as HTTPs. We detect 4 phases: first low growth until day 100, followed by stagnation until day 140 before it surges, tripling the request count and finally stagnates again. This surge is not observed in the HTTPS traffic.

In Figure 4.66, we show requests for other domains – not the experiment domain – detected at the web server of the experiment host. These requested domains are either domains not within our control or the IP address of the experiment host. We find similar results for HTTP and HTTPS traffic: before the leak of the domain, traffic to the web server is sparse and after the leak, traffic rate increases to about 180 requests per day for HTTP and HTTPS. Traffic for both protocols seems to be unaffected by the reissuances. The increase in traffic rate occurs after the domain leak via CT, but this might not be the cause. Shortly before the leak, the web server on the host is set up. Before, any requests to the HTTP(s) ports would not return a valid response, as no web server would be running on the interface. Hence, we cannot conclude that the increasing traffic rate was due to the setup of the web server or the leak of the domain running on the web server. In Section 4.4.1, we will compare these results with the baseline experiment 410b to determine the root cause of the increase.

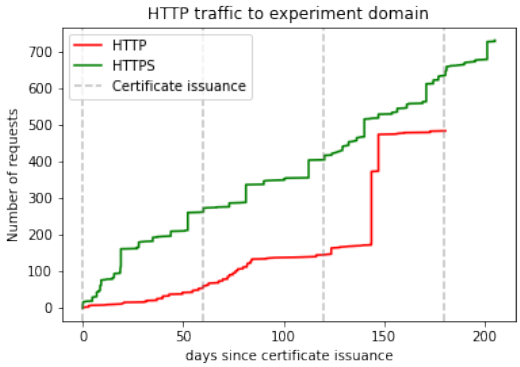

Figure 4.65: Cumulative number of HTTP(s) requests for the domain in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

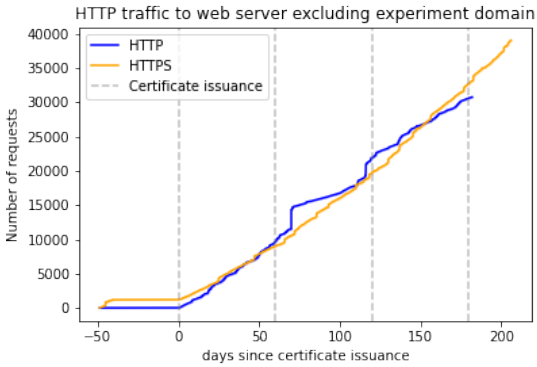

Figure 4.66: Cumulative number of HTTP(s) requests not for the domain, but detected at the webserver in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

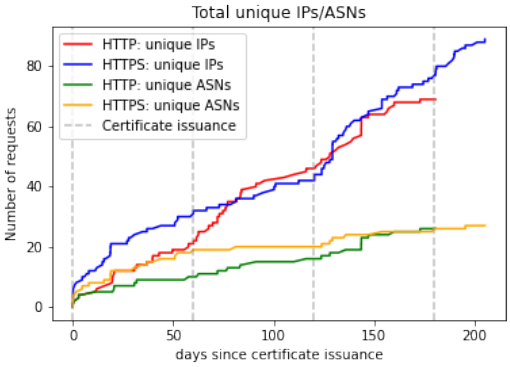

Figure 4.67: Cumulative number of unique IPs and ASes detected in HTTP requests for the domain in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

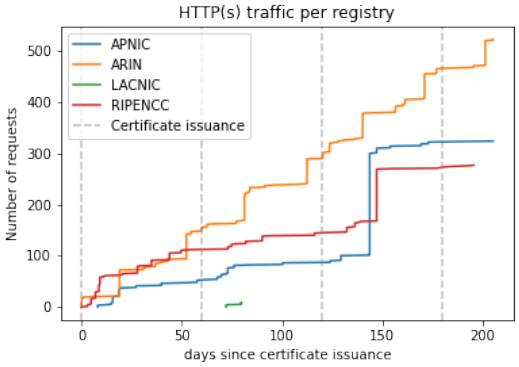

Figure 4.68: Cumulative number of HTTP requests per registry for the domain in experiment 410a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.67 shows the total number of unique IPs and ASes, sending HTTP requests for the domain of the experiment. The total count is separated into HTTP and HTTPs, IPs and ASes. On average, 3.4 addresses are detected per AS, with a standard deviation of 4.5. Over 50 % of the detected ASes only use one address to send HTTP(s) requests for the experiment domain. Most addresses come from a small portion of the detected ASes; the AS with the most addresses has 21. The detection rates of new IPs are similar for both protocols as is the detection rate of new ASes. Even though the count is similar in both protocols, we do not observe the same cumulative count of HTTP request in both protocols, see Figure 4.65 and Figure 4.66. HTTPS traffic follows a steady increase as is the number of detected IPs. On the other hand, HTTP traffic experiences slow growth alternated with a surge, whereas the number of unique IPs do not reflect the same behaviour. The surge in HTTP traffic stems from a single IP address. Both protocols have in common that reissuances of the domain's certificate does not affect the number of detected IPs and ASes.

Figure 4.68 shows the cumulative number of HTTP requests for the experiment domain per registry. We only detect 3 registries: APNIC, ARIN and RIPENCC. Traffic from ARIN is growing linearly similar to HTTPS traffic. Traffic from the remaining two registries shows the same behaviour as HTTP traffic, where we observe a surge on day 150 and outside the surge slow growth. Looking at the registry distribution for HTTP and HTTPS, we find that the registry ARIN favors HTTPS traffic where it is responsible for 58.5 % of the requests, while for HTTP traffic it only covers 21.8 %, being the registry with the least requests. Correspondingly, the other two registries, APNIC and RIPENCC, are responsible for 76 % of

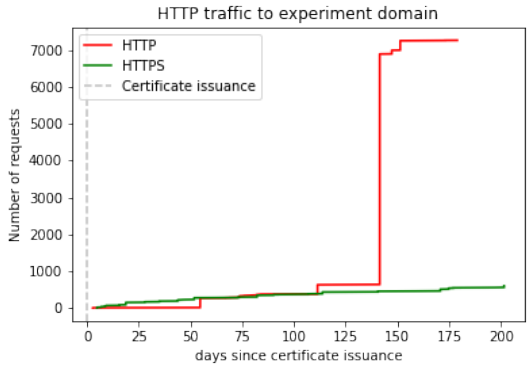

Figure 4.69: Cumulative number of HTTP(s) requests for the domain in experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

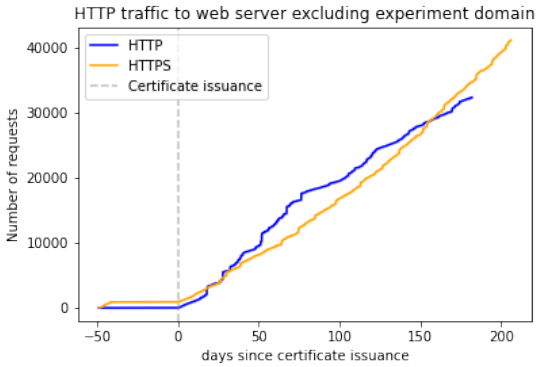

Figure 4.70: Cumulative number of HTTP(s) requests not for the domain, but detected at the webserver in experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

the HTTP requests, but only 30.7 % of the HTTPS requests. Hence, we have the similarities between Figure 4.68 and Figure 4.65.

4.3.3. Experiment 410b

In experiment 410b, we are hosting a domain on one address in IPv4 and generate a self-signed certificate locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv4, affects scanning traffic at the host of the domain. The experiment was run once and ran from March 3, 2021 to September 5, 2021. We collect 86.2K packets for a total of 18.2 MB at the host of the experiment.

We distinguish between requests for the experiment domain and requests for other domains. Figure 4.69 shows requests for the experiment domains, with HTTP and HTTPS separated; we record 7343 HTTP requests and 603 HTTPS requests. Before the leak, we do not detect any requests for the domain. The first HTTP request is received after 2.7 days and the first HTTPS request is after 4.8 days. After the first request, the traffic of both protocols is growing linearly at a similar rate. On day 140, HTTP traffic diverts from the linear growth and surges with more than 6000 requests within one minute before it returns to the previous trajectory.

In Figure 4.70, we show requests for other domains – not the experiment domain – detected at the web server of the experiment host. These requested domains are either domains not within our control or the IP address of the experiment host. We find similar results for HTTP and HTTPS traffic: before the certificate issuance of the domain, traffic to the web server is sparse and thereafter, traffic rate increases to about 200 requests per day for HTTP and HTTPS. The increase in traffic rate occurs immediately after generating the certificate. Because the certificate generation is carried out in private, it cannot be the cause of the increase in traffic. The leak of the domain via PTR records is likely not the sole cause. The remaining factor is the start of the web server: since HTTP(s) requests sent to the experiment host now yield proper HTTP responses from a running web server, there are more incentives to send HTTP(s) requests.

Figure 4.71 shows the total number of unique IPs and ASes, sending HTTP requests for the domain of the experiment. The total count is separated into HTTP and HTTPs, IPs and ASes. On average, 2.1 addresses are detected per AS, with a standard deviation of 2.0. Over 50 % of the detected ASes only use one address to send HTTP(s) requests for the experiment domain. Most addresses come from a small portion of the detected ASes; the AS with the most addresses has 12. The detection rates of new IPs are similar for both protocols as is the detection rate of new ASes. Even though the count is similar in both protocols, we do not observe the same cumulative count of HTTP request in both protocols, see Figure 4.69 and Figure 4.70. HTTPS traffic follows a steady increase as is the number of detected IPs.

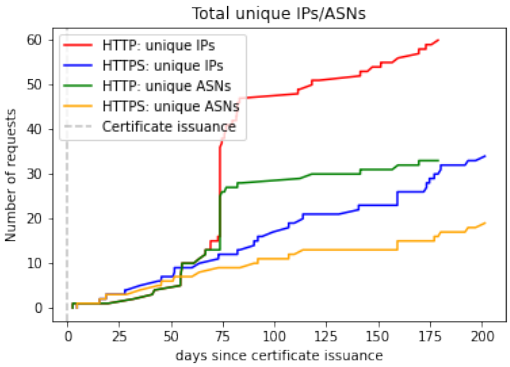

Figure 4.71: Cumulative number of unique IPs and ASes detected in HTTP requests for the domain in experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

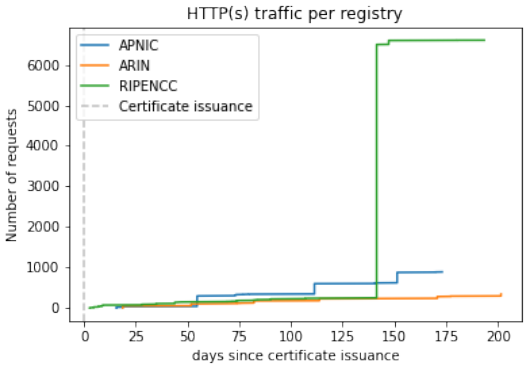

Figure 4.72: Cumulative number of HTTP requests per registry for the domain in experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

On the other hand, HTTP traffic experiences slow growth alternated with a surge, whereas the number of unique IPs does not reflect the same behaviour. Both protocols have in common that reissuances of the domain's certificate do not affect the number of detected IPs and ASes.

Figure 4.72 shows the cumulative number of HTTP requests for the experiment domain per registry. We only detect 3 registries: APNIC, ARIN and RIPENCC. Traffic from ARIN is growing linearly similar to HTTPS traffic. Traffic from the remaining two registries shows the same behaviour as HTTP traffic, where we observe a surge on day 150 and outside the surge slow growth. Looking at the registry distribution for HTTP and HTTPS, we find that the registry ARIN favors HTTPS traffic where it is responsible for 51.2 % of the requests, while for HTTP traffic it only covers 1.2 %, being the registry with the least requests. Correspondingly, the registry RIPENCC is responsible for 87.7 % of the HTTP requests, but only 30.5 % of the HTTPS requests. Hence, we have the similarities between Figure 4.72 and Figure 4.69. The surges in traffic from the registries APNIC and RIPENCC contribute together to the surge seen in HTTP traffic.

4.3.4. Experiment 610a

In experiment 610a, we are hosting a domain on one address in IPv6 and leak its name via CT by requesting a certificate from LE. In this context, the goal of the experiment is to determine whether leaking a domain, hosted in IPv6, via CT affects scanning traffic at the host of the domain. Experiment 610a was run 3[3] times and ran from March 12, 2021 to September 5, 2021. We collect 707 HTTP requests for a total of 177 KB at the host of the experiment.

We distinguish between requests for the experiment domain and requests for other domains. Figure 4.73 shows requests for the experiment domains, with HTTP and HTTPS separated; we record 308 HTTP requests and 399 HTTPS requests. Before the leak, we do not detect any requests for the domain. The first HTTP requests of the three experiment domains are received after (77, 54, 47352) seconds and the first HTTPS requests after (110, 110, 65) days. Within the first detected HTTP requests, we find one outlier differing from the other detection times by a factor of nearly 1000. Even with the outlier, we find lower detection times for HTTP than its secured version HTTPS: the first HTTP requests are detected within a day, whereas the first HTTPS requests are only detected after at least 65 days. After the first request, HTTPS traffic is growing linearly and seems not to be affected by the reissuances of the certificate. HTTP traffic is detected at a lower rate than HTTPS. In one of the experiment runs, we detect a high influx of HTTP requests on day 96, 257 requests from a single address. The surge was omitted in Figure 4.73 for better clarity as it only occurred in one of the runs.

[3]The Experiment 610a was run 5 times, but 2 runs failed because of an issue in the certificate issuance process

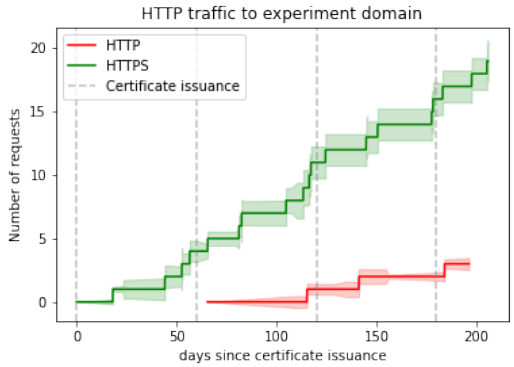

Figure 4.73: Average cumulative number of HTTP(s) requests for the domain in the 3 runs of experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

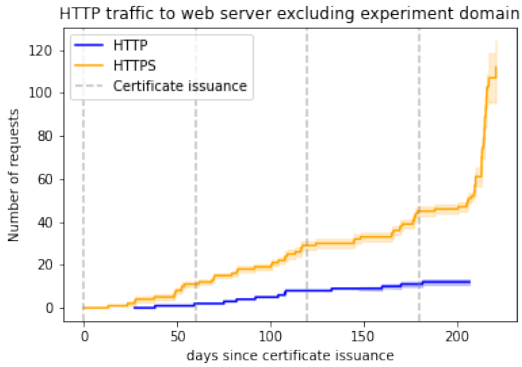

Figure 4.74: Average cumulative number of HTTP(s) requests not for the domain, but detected at the webserver in the 3 runs of experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued.The shaded area shows the standard deviation.

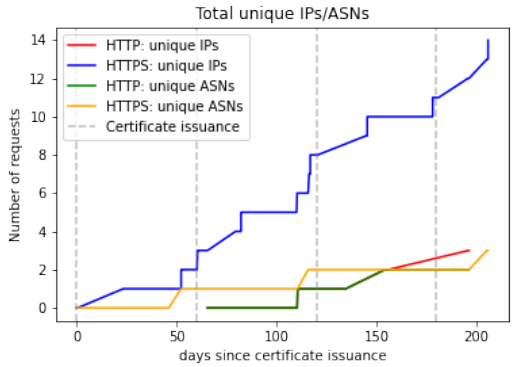

Figure 4.75: Average cumulative number of unique IPs and ASes detected in HTTP requests for the domain in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

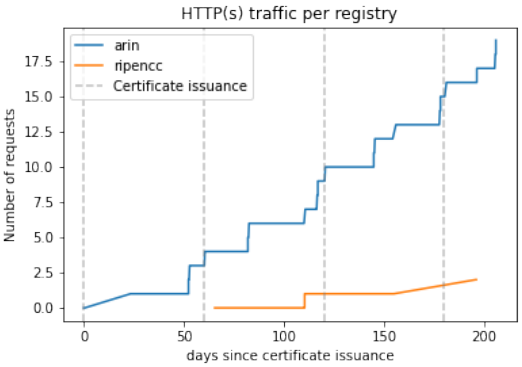

Figure 4.76: Average cumulative number of HTTP requests per registry for the domain in experiment 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

In Figure 4.74, we show requests for other domains – not the experiment domain – detected at the web server of the experiment host. These requested domains are either domains not within our control or the IP address of the experiment host. Before the certificate issuances, we detect no HTTP(s) traffic. HTTPS traffic is detected shortly after the initial issuance, whereas HTTP traffic is only detected after 25 days. For both protocols, we see a linear growth in traffic, but HTTPS traffic experiences stronger growth and a surge of 200 packets after day 200. The surge occurs for all 3 experiment runs at the same relative moment

Figure 4.75 shows the total number of unique IPs and ASes, sending HTTP requests for the domain of the experiment. The total count is separated into HTTP and HTTPs, IPs and ASes. The number of unique IPs and ASes nearly coincide for HTTP traffic, ie. we mostly detect one IP address per AS. Regarding HTTPS traffic, we detect on average 4.0 addresses per AS. The count is increasing linearly for both IPs and ASes. The difference in detection rate between HTTP and HTTPS reflects the difference in the protocol distribution we find in Figure 4.73. Both protocols have in common that reissuances of the domain's certificate do not affect the number of detected IPs and ASes.

Figure 4.76 shows the average cumulative number of HTTP requests for the experiment domain per registry. We only detect 2 registries: ARIN and RIPENCC. Traffic from both registries is growing linearly similar to HTTP(S) traffic in Figure 4.73. Traffic from ARIN shows similar behaviour to HTTPs

traffic: in fact, 90 % of the traffic generated from ARIN is HTTPs traffic. This behaviour is observed in all 3 runs of the experiment.

4.3.5. Experiment 610b

In experiment 610b, we are hosting a domain on one address in IPv6 and generate a self-signed certificate for the domain locally on the host. The domain is not leaked via CT. In this context, the goal of the experiment is to determine whether self-issuing a certificate for the experiment domain, hosted in IPv6, affects DNS traffic pertaining to the domain of the experiment. This experiment will be used as a baseline for comparison with experiment 610a. Experiment 610b was run 5 times and ran from March 12, 2021 to September 5, 2021. During the time period of the experiment, we do not detect any HTTP requests at the web servers of the experiment.

4.4. Effect of CT on scanning traffic at a host

In this section, we aggregate the results from the experiments in Section 4.1, Section 4.2 and Section 4.3 to find an answer to the research question 1.1: How does Certificate Transparency affect scanning traffic seen by a website host? From the captured network traffic, we isolate potential scanning traffic stemming from CT. We evaluate the effect of CT in both IPv4 and IPv6 address spaces.

We determine the effect of CT by comparing the main experiment 410a/610a and the associated control experiment 410b/610b. In each of the experiments, we run one host with a web server hosting a domain. In experiment 410a/610a, we leak the domain by requesting a certificate which is added by the issuing CA to a CT log. To have a baseline for comparison, we create experiment 410b/610b, in which CT is not involved – the certificate is self-signed locally on the experiment host and therefore it is not introduced into any CT log. In the main experiment, an external CA issues the certificate and also appends automatically the issued certificate to a CT log. The control experiment helps in isolating the resulting effect of CT in the main experiment.

First, we look at the differences between the experiments at the DNS server. CT logs make certificates of domains publicly accessible. The certificate reveal among other things, the domain name to which it belongs. However, no information is revealed about the physical address of the domain. The address can easily be retrieved when knowing the domain name with DNS. Since we only make the domain name public in CT logs and indirectly through PTR records, any DNS queries for the domain name show that domain names are being collected either from CT logs or PTR records. Hence, the authoritative DNS server of the domain is the frontline where we detect the first signs of scanning traffic stemming from CT and PTR records. In Section 4.1.6 and Section 4.1.6, we showed the effect of PTR records on scanning traffic. With the results from the PTR experiment 4p0c/6p0c, and the control experiment, we filter out scanning traffic not pertaining to CT.

4.4.1. Effect in IPv4

DNS traffic

We first start assessing the IPv4 experiments 410a and 410b. Figure 4.77 shows the cumulative number of detected DNS requests for both experiments. Before the certificate issuance, the experiments share the same rate of 3 queries per day. After day 0, the two experiments have different setups: 410a involving CT and 410b without. We detect an immediate effect on the rate of requests in experiment 410a: after only 20 seconds, we detect a burst of 245 requests. The same phenomena is observed at the reissuance dates, days 60 and 120, with smaller bursts of 169 and 183 requests. Control experiment 410b does not experience the same surges. We show a different visualization of the DNS requests as the daily number of DNS queries with Figure 4.78, which makes surges more distinct.

(0) On the days before the certificate issuance, we detect a similar low number of queries in both experiments. (1-3) The surges at the issuances become more apparent in contrast to the traffic on the other days, in the main experiment. We attribute these surges to CT as we do not detect similar surges in the control experiment. PTR cannot be the reason for such surges because otherwise such surges would appear in both experiments since both have them created for the domains. From the observations made on the issuance dates in experiment 410a, we find that introducing a certificate to a CT log causes a significant spike in DNS queries for the domain. The initial issuance causes the largest spike, while subsequent spikes are of smaller but similar size. The difference in size suggests that some scanners keep track of the encountered domains in CT logs, and only query ones that are new. Consequently, these scanners ignore domains leaked via reissuances, therefore resulting in less DNS queries at the reissuances.(4)(6) Surges aside, both experiments experience an increase in rate of query after the initial issuance, resulting in a rate of 14/11 queries per day in experiment 410a/410b. However, the increase only occurs after 6 days in the control experiment 410b, whereas in the main experiment it occurs already after 20 seconds. (5)(7) A further rise in rate to 22 queries per days is detected in both experiments. We detect the second at the first reissuance on day 60 in experiment

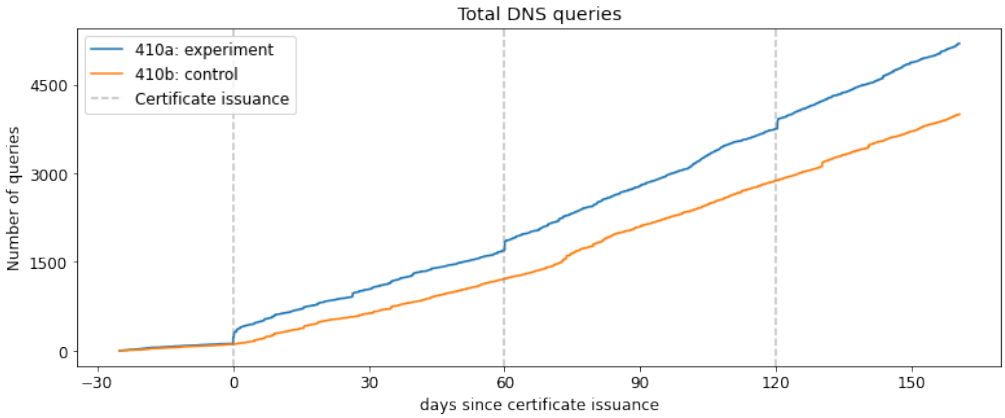

Figure 4.77: Comparison of cumulative number of DNS queries in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

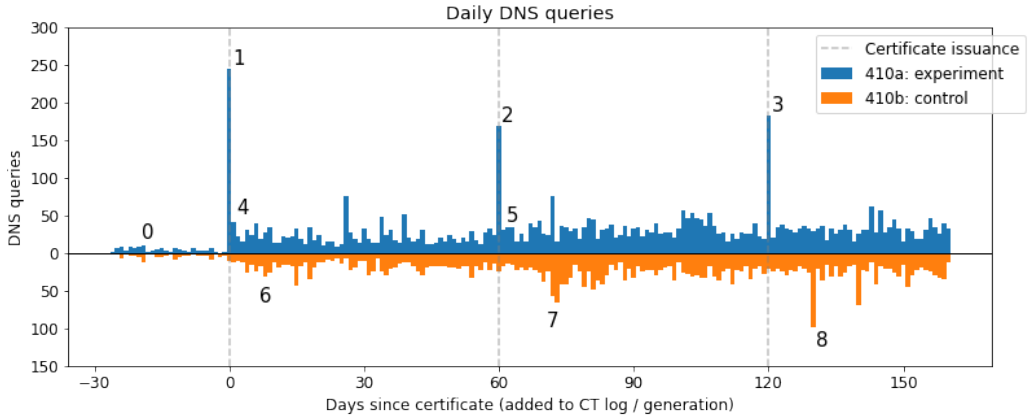

Figure 4.78: Comparison of daily number of DNS queries in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued. Special events are denoted by a number.

410a and 10 days later in experiment 410b. Like (4) and (6), there is a delay between the rise in experiment 410a and 410b.

Surges aside, both experiments experience an increase in rate of query after the initial issuance, resulting in a rate of 14/11 queries per day in experiment 410a/410b. However the increase only occurs after 6 days in the control experiment 410b, whereas in the main experiment it occurs after already 20 seconds. A further rise in rate to 22 queries per days is detected in both experiments. Similarly to the first rise, the second one is detected at the first reissuance on day 60 in experiment 410a and is delayed by 10 days in experiment 410b. Exclusively in experiment 410b, we detect a small jump in DNS queries on day 130. The increases in rates are detected at an interval of approximately 60 days. As both experiments experience those increases on 60 day intervals, we conclude that they are not caused by CT. However, the increases occur earlier in the main experiments, thus we suspect that CT expedites it. Figure 4.79 shows the cumulative number of queries with reverse lookups and Figure 4.81 shows the cumulative number of queries with the domain name of the experiment with 0x20 bit encoding – the queried domain name only matches the one from the experiment when converted to lowercase. We see that in both figures the previously discussed increases rate occurring. Reverse DNS lookups and DNS queries for the domain with 0x20 encoding attribute to the detected increases. The findings suggest that scanners change their strategy after 60 days of scanning. After discovering a new domain, some scanners could be waiting 60 days before ramping up their scanning endeavors. The 60 day wait could be to make sure to the target domain is not a short-lived one. Further experimentation is required to verify this claim. CT causes this scanning behaviour to occur earlier. Without CT, the domain can

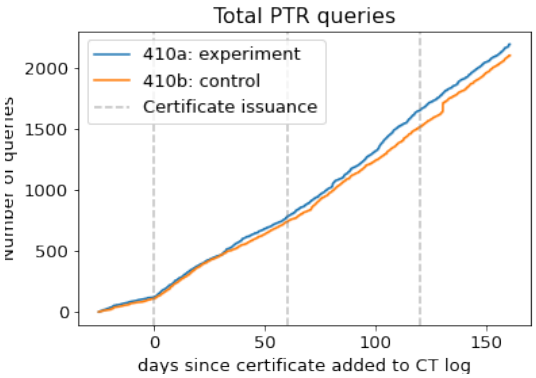

Figure 4.79: Comparison of cumulative number of reverse lookups in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

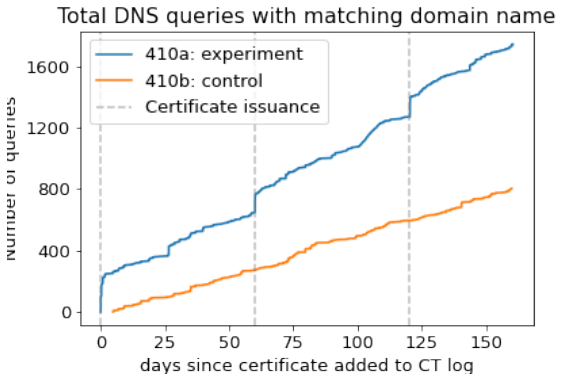

Figure 4.80: Comparison of cumulative number of DNS queries for the domain names in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

only be leaked via its PTR record. Therefore, it is likely that scanners with this strategy are actively monitoring CT logs to discover domains, besides making reverse DNS lookups.

The interval between the first and second increase in rates is approximately 60 days. Since both experiments experience those increases on 60 day intervals, we conclude CT is not the cause but PTR records. However, the increases occur about 6 days earlier in the main experiment than in the control. Therefore, we suspect CT expedites the process. Some actors with this scanning strategy could be actively monitoring CT logs to discover domains, besides making reverse DNS lookups.

Without CT, the domain can only be leaked via its PTR record. Comparing the detection times of the first query for the domain, we find that the main experiment has a much shorter detection time of 23 seconds compared to the control experiment with a detection time of 4.7 days. The difference in detection times is similar to the delay we observed between increases (4) and (6), (5) and (7). CT causes DNS scans of a domain to occur 6 days earlier. We will further study the nature of these increases by segmenting DNS queries by query types.

In Section 4.1.6, we have studied the effect of PTR records on DNS queries. The findings show that PTR records are being queried to discover new domains. Hence, the PTR record is also a source of leak for domains. As a result, we detect queries for the domain name 5 days after the creation of the PTR record. Using this finding, we can attribute the increases in rate detected in both experiments to the PTR record created during the setup of the experiment.

To better understand the findings in Figure 4.77, we segment DNS queries into different types: reverse DNS lookups, DNS queries with/without 0x20 encoding, DNS queries for non-existing subdomains, DNS queries with/without DNSSEC support and DNS queries made via TCP.

Figure 4.79 shows the cumulative number of reverse lookups. Reverse lookups make up the whole traffic before the initial certificate issuance. Both experiments share a similar curve; we only detect 5 % more in experiment 410a than in the control. CT causes a slight increase in reverse lookups, but no surges at issuances as detected in Figure 4.77.

In Figure 4.79 and Figure 4.81, we see the previously discussed increases in rate, (4), (6), (5) and (7). Reverse DNS lookups and DNS queries with 0x20 encoding are the sole sources for the detected increases. The findings suggest scanners change their strategy after 60 days of scanning. After discovering a new domain, some scanners wait 60 days before ramping up their scanning endeavors. The 60 day wait could be to make sure that the targeted domains are not short-lived in order to save computer resources.

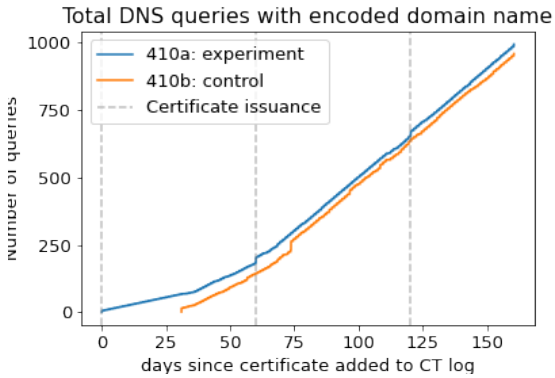

Figure 4.81: Comparison of cumulative number of DNS queries for the domain names with 0x20 bit encoding in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

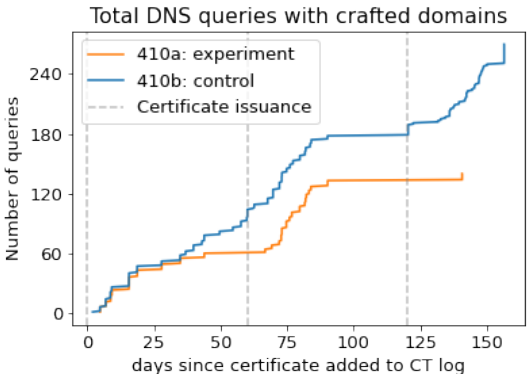

Figure 4.82: Comparison of cumulative number of DNS queries for non-existing subdomains in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

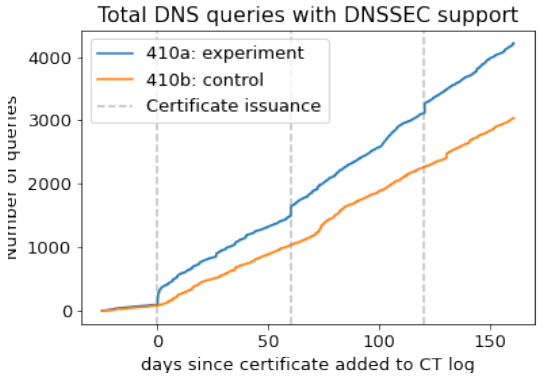

Figure 4.83: Comparison of cumulative number of DNS queries with DNSSEC support in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

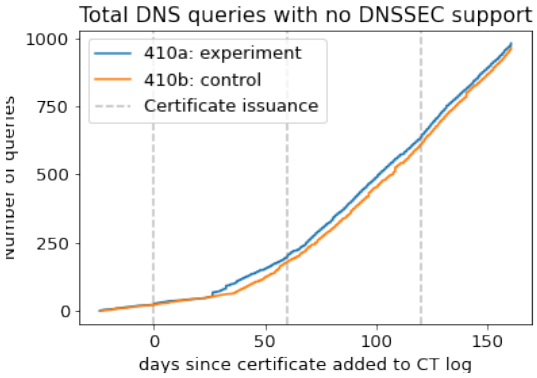

Figure 4.84: Comparison of cumulative number of DNS queries without DNSSEC support in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.81 shows the cumulative number of queries for the experiment domain with 0x20 bit encoding – the queried domain name only matches the one from the experiment when converted to lowercase. In the experiment 410a, we detect these queries shortly after the initial issuance, about 20 seconds, whereas in the control experiment, we detect these queries starting on day 30. Similarly to Figure 4.80, the main experiment shows some jumps in queries at the issuances, albeit on a lower scale. Both experiments have different phases with distinct detection rates. In experiment 410a, we detect rising rates on day 0, 35 and 60; in experiment 410b, we detect rising rates on day 30 and 70. Despite receiving these queries 30 days earlier and experiencing bursts at the issuances in the main experiment 410a, the total count in both experiments is only differing by less than 3 %.

Figure 4.80 shows the cumulative number of queries for the domain name of the experiment. The difference between the experiments is prominent. In experiment 410a, we detect surges at the issuances and a steady rate of 5 queries per day, whereas in the control experiment 410b, we only detect a steady rate of queries 4 days after the initial issuance. From Figure 4.80, we notice that the initial issuance caused the highest burst of queries; the remaining are of smaller but similar size. The detected surges coincide with the surges in Figure 4.77. Queries for the domain name make up 90 % of the surges detected. Disregarding the issuance days, the rate in experiment 410a is slightly higher than in the control experiment. The findings show that appending a certificate into a CT log causes a spike in DNS queries for the associated domain name on issuance dates and a slight increase in detection rate. Those two factors result in doubling the number of received DNS queries for the domain.

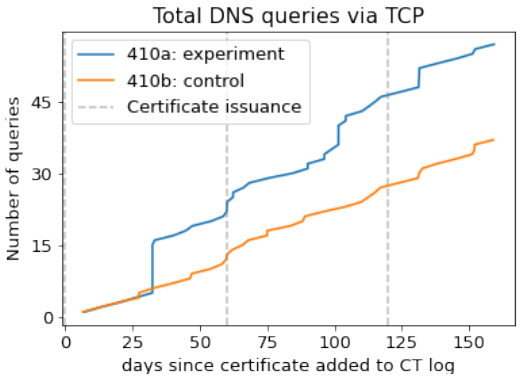

Figure 4.85: Comparison of cumulative number of DNS queries made via TCP in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

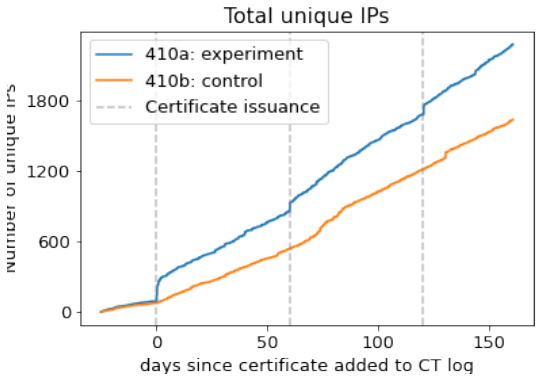

Figure 4.86: Comparison of cumulative number of unique IPs detected at the DNS server in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.82 shows the cumulative number of queries for crafted domains. The crafted domains are non-existing subdomains of the experiments. Up to day 35, we detect a similar number of queries for crafted domains in both experiments. For the next 35 days, the number of detected queries diverges: the rate remains constant in the main experiment, while the number of queries stagnates in the control. The next divergence occurs approximately on day 120, where the rate increases in the main experiment, but in the control, we only detect one additional query. The traffic jumps at issuances in experiment 410a as detected in Figure 4.77 . There is not a clear pattern how queries for crafted domains increases due to CT, but we can only show that it increases by 93 % with minor jumps in queries at the issuances.

Next, we are classifying the queries by their properties. Figure 4.83 and Figure 4.84 show the cumulative number of queries with and without DNSSEC support. For queries with DNSSEC support, we observe an increase in rate after the initial issuance in both experiments. In the experiment 410a, we notice issuances provoke a spike in requests, with the initial issuance provoking the largest spike. Furthermore, the detection rate is slightly higher than the control experiment. We come to the same conclusion as in Figure 4.80, that appending a certificate into a CT log causes a spike in DNS queries for the associated domain name for the first 1-3 days and a slight increase in detection rate. These factors result in a 38 % increase in DNS requests with DNSSEC support. As for queries without DNSSEC support, the total count in the experiments diverges on day 25, where the detection rate slightly increases in the main experiment 410a. But this increase only results in an overall count increase of 0.6 %.

Figure 4.85 shows the cumulative number of queries made via TCP. We detect a small burst of queries on day 30 in the experiment 410a. Disregarding the burst, the main experiment shows a slightly higher rate of detection than its control, resulting in 11 more queries in 150 days. The involvement of CT resulted in a slight increase in TCP DNS queries and a burst of queries on day 35. Note that DNS over TCP represents less than 1 % of the queries we detect.

We will now shift the focus to the sources of the queries and analyse the source addresses and their respective ASes. Figure 4.86 shows the cumulative number of unique source addresses. The unique count resembles the shape in Figure 4.77: there is an increase in detection rate after the initial issuance, with the main experiment having a slightly higher rate, and having surges that occur at the issuances. The surges in unique IPs show that the burst of queries detected after the issuances does not come from sources, which already have queried before, but from new undetected sources. However, detecting new sources does not necessarily indicate that we are detecting new actors that are scanning the domain.

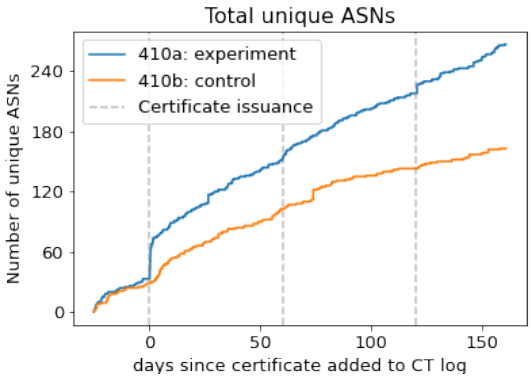

Figure 4.87: Comparison of cumulative number of unique ASes detected at the DNS server in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

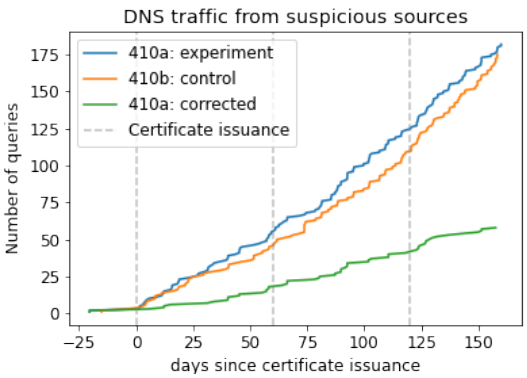

Figure 4.88: Comparison of cumulative number of DNS queries from suspicious sources in experiment 410a and 410b. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The green line

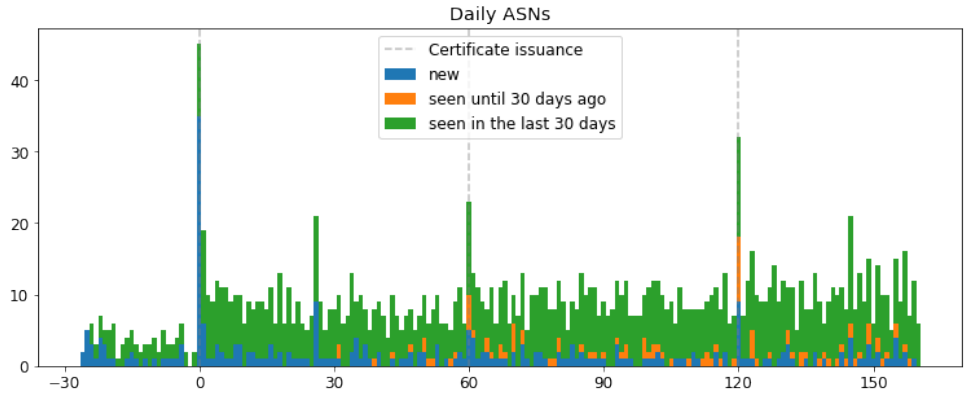

Figure 4.89: Daily detected ASes at DNS server for experiment 410a. New ASes are ASes that were not encountered prior. ASes seen until 30 days ago are ASes that have not been detected in the last 30 days. And ASes seen in the last 30 days represents frequent ASes. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.87 shows the cumulative number of unique ASes. Before the initial issuance, the traces are similar for both experiments. In 410b, we only detect a light increase in new ASes on day 5 for a short period of time. The number of unique ASes does not increase linearly, but shows a slow decline in daily ASes. Even though we see a steady increase in unique IPs in Figure 4.86, we do not see the same in the number unique ASes. Therefore, this suggests that the steady increase in unique IPs originates from already detected ASes the further away from the initial issuance.

In 410a, we detect a major influx of new ASes querying the domain on day 0. Furthermore, the overall rate increases. We do not see surges in new ASes at reissuances similar to Figure 4.86. The lack of surges suggests that the surge in new IPs originate from ASes that have already queried for the domain.

To verify our claim, we break down the ASes detected per day in Figure 4.89 . New daily ASes, shown in blue bars, gives a different visualization of Figure 4.87 of newly detected ASes. Figure 4.89 adds ASes that were detected in the last 30 days in green and ASes that were not detected in the last 30 days in orange. We refer to such ASes as recurring ASes and old ASes. The sum of the 3 types of ASes constitutes the total number of ASes detected in a day. As found in Figure 4.86, we detect a higher number of new ASes on issuance dates than on average. We notice interesting ASes compositions on issuance dates. While old ASes remain fairly steady on non-reissuance dates, we detect a significant

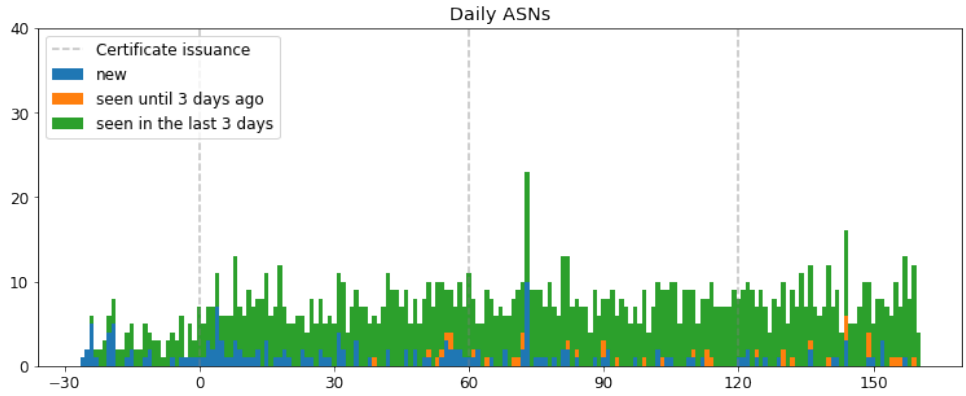

Figure 4.90: Daily detected ASes at DNS server for experiment 410b. New ASes are ASes that were not encountered prior. ASes seen until 30 days ago are ASes that have not been detected in the last 30 days. And ASes seen in the last 30 days represents frequent ASes. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

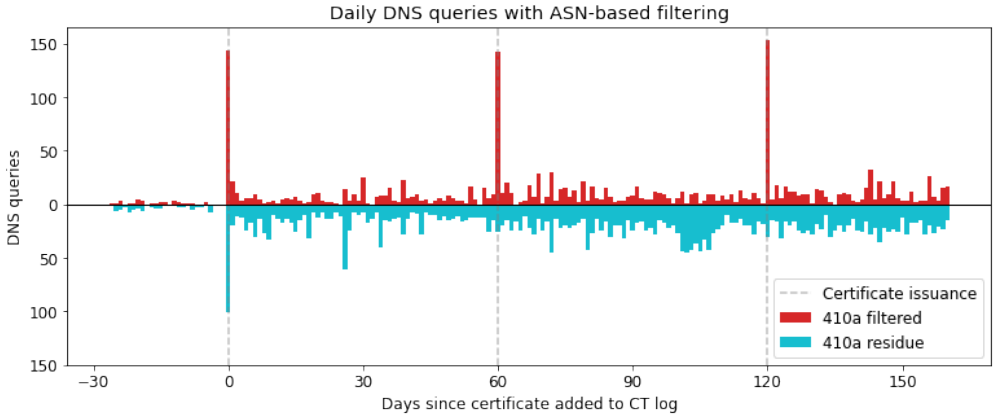

Figure 4.91: Daily number of DNS queries in experiment 410a after applying ASN-based filter. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The red bars show the DNS queries after the correction. Blue bars show the DNS queries that are removed.

increase on reissuance dates, days 60 and 120. The number of recurring ASes does not increase. So reissuances are characterized by an increase in new and old ASes. After the initial issuance, we detect a high amount of new ASes querying for the domain, because of the domain leak via CT. On reissuances, we do not see the same amount of new ASes but a higher amount of old ASes. The initial issuance attracts the most new ASes; reissuances attracts old ASes. Most of these old ASes were detected at the initial issuance. They query for the domain at every issuance regardless of whether they queried the domain. They tend to query shortly after issuances, therefore are not classified as recurring ASes. We also notice that we do detect a smaller amount of ASes on reissuances: some ASes are likely keeping track of which domains were queried and therefore ignore any domain leaks due to reissuances. On non-issuance dates, the number of ASes as well as the composition remains fairly constant and with similar values as 410b, see Figure 4.90. Since we detected the most divergence between the main and the control experiment at issuances, most ASes detected in between issuances do not participate in the DNS traffic related to CT.

Figure 4.88 shows the cumulative number of queries originating from suspicious sources. We classify addresses as suspicious based on the collected threat intel. We find an increase in queries from suspicious sources after the initial issuance, with the main experiment 410a having a slightly higher rate, resulting in 10 more queries than the control. Since DNS traffic from suspicious sources is almost the same in both experiments, it is not tied to CT, but rather to PTR. Hence, scanners who monitor CT

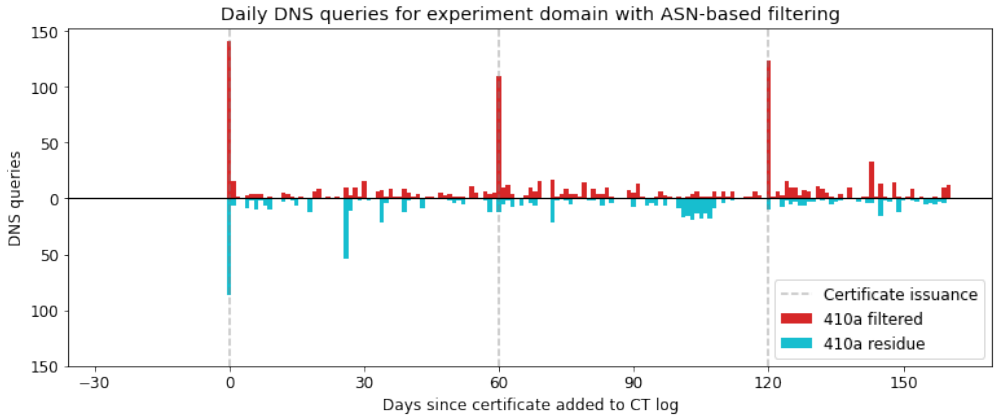

Figure 4.92: Daily number of DNS queries for the domain in experiment 410a after applying ASN-based filter. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The red bars show the DNS queries after the correction. Blue bars show the DNS queries that are removed.

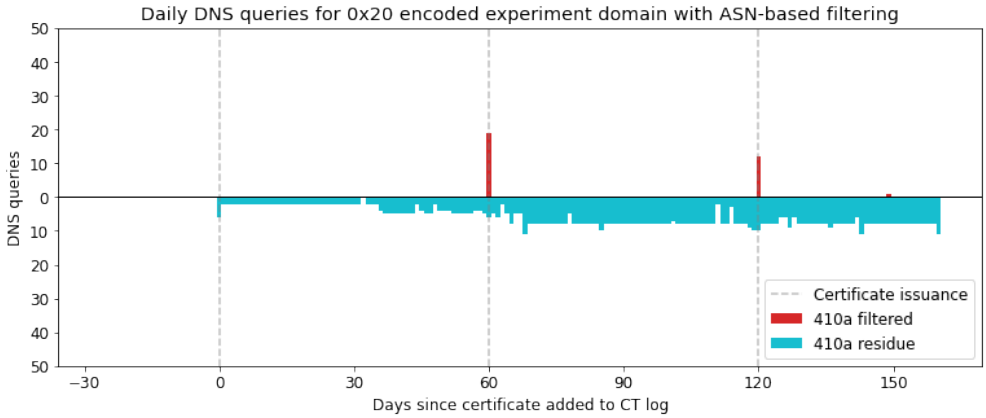

Figure 4.93: Daily number of DNS queries with 0x20 bit encoding in experiment 410a after applying ASN-based filter. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The red bars show the DNS queries after the correction. Blue bars show the DNS queries that are removed.

logs for new domains to scan use source addresses that are not flagged as suspicious or malicious by VirusTotal. It seems that these addresses are only used for scanning and not for causing harm to the target. While scanning is harmless by itself, it can be a precursor to a malicious attack as scanning is used is used as recon for potential targets. Since our threat intel does not flag these scanning addresses as suspicious, it is possible that other addresses not used for scanning are used for attacking and exploiting the target. The reason for using different sets of addresses for different purposes is that scanning with “clean” addresses might not cause any alarms at the target.

To isolate the effects of CT, we correct the DNS traffic in experiment 410a by removing queries from source addresses that appear in the control experiment 410b. The green line in Figure 4.88 shows that correcting the DNS traffic removes 68 % of queries from suspicious sources. The correction results in a similar curve than before, only with a lower rate; it does not uncover any patterns in traffic hidden by the DNS traffic due to PTR. Because most of the traffic was removed and no hidden pattern was found, the remaining traffic is likely due to the imperfect correction method.

We have also computed the corrected traffic for the different types of queries, but the resulting plots nearly coincided with the uncorrected plots, thus for readability, they were omitted. Only in Figure 4.88 is the correction making a visible change.

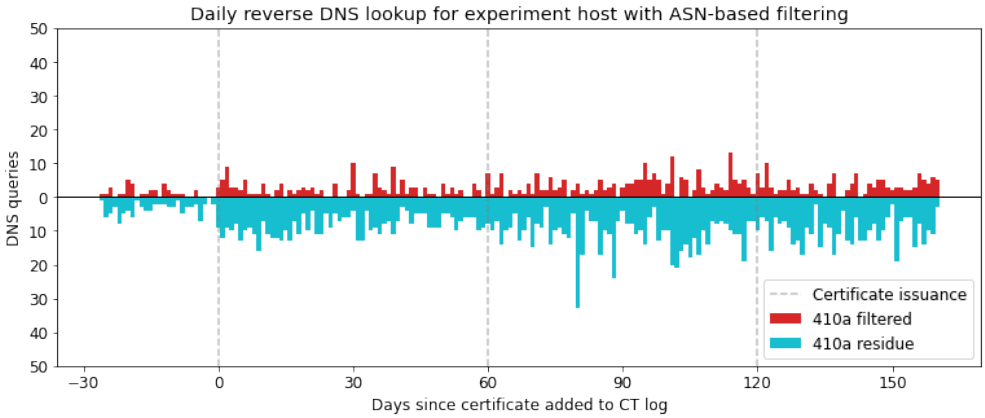

Figure 4.94: Daily number of reverse lookups in experiment 410a after applying ASN-based filter. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The red bars show the DNS queries after the correction. Blue bars show the DNS queries that are removed.

Since the previous correction attempt deemed to be ineffective in isolating traffic caused by CT, we create an AS-based filter. We examine the daily queries of each AS and check whether we detect surges on at least 2 issuances. The remaining ASes are filtered out. We impose the requirement of two surges for an AS, because an AS could query by chance on an issuance date, but it would be less likely that it would happen on two issuance dates. The second requirement is that there needs to be a surge on the day of the issuance to remove ASes that query at a constant rate through issuances. We classify a day as having a surge if we detect more queries on that day than on other days where we detect queries. After filtering, queries from 24 ASes are remaining. We are filtering by ASes instead of IP addresses, because it allows us to detect addresses that might only cause surges at one issuance, thus not satisfying the requirements of the filter, but does when combined with another address in the same AS. The AS filter detects traffic from scanners who are using a network of addresses within one AS for their scanning endeavors. There are limitations to our method. The filtering is not perfect and cannot remove all the DNS queries not caused by CT. If an AS qualifies the requirements does not mean that CT causes all the outgoing queries, but only some. There can be multiple scanners in one AS, but not every scanner necessarily monitors CT logs. Furthermore, not all queries caused by CT are captured using the filter: ASes can still be the source of queries caused by CT without abiding to the set of requirements. We will visualize the daily queries for both the filtered queries and the residue to show the effectiveness and weaknesses of the filtering.

Figure 4.91 shows the filter on the daily DNS queries. The filtering removes more than half of the queries between issuances but also removes a significant portion of the initial surge. The surges at issuances remain and have nearly the same size, as opposed to Figure 4.77 where the initial issuance causes a greater surge than on reissuances. Due to our filtering criteria, queries, from the scanners that keep track of encountered domains in CT logs, and therefore ignore reissuances, will be removed even though this kind of traffic is related to CT. The removed queries at the initial issuance are likely from that type of scanners. The filtered surges, due to being similar in size are likely from the other type of scanners, that is not actively tracking the encountered domains and therefore blindly queries domains regardless whether the leaked certificate is from an initial issuance or a reissuance. The filtering has managed to remove background noise and to recognize if queries come from scanners that keep track of encountered domains.

Figure 4.92 the filter on the daily DNS queries for the domain of the experiment. It displays similar results to Figure 4.91, except with less background noise. The filtering is particularly effective in removing background traffic in the daily DNS queries with 0x20 encoding, see Figure 4.93. It has removed nearly all the traffic except for two surges at the reissuances. The filter removes the bulk of reverse lookups in Figure 4.94. Ideally, the filtering should have removed nearly everything, since we did not find major differences between the main and the control experiment. The filtering, albeit not working

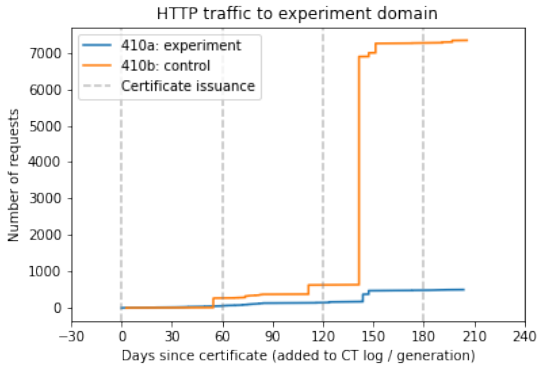

Figure 4.95: Cumulative number of HTTP requests for the domains in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

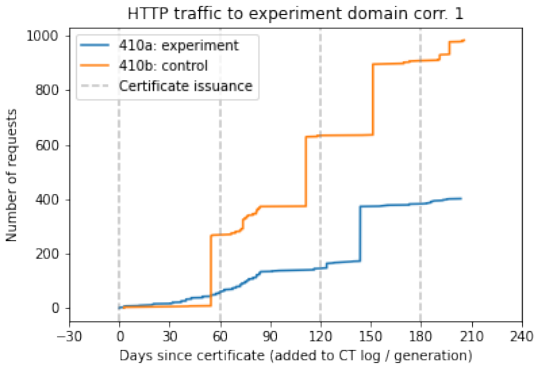

Figure 4.96: Cumulative number of HTTP requests for the domains in experiment 410a and 410b after the removal of traffic from the AS causing the surge of over 6000 requests. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

perfectly, has confirmed our findings without filtering. Our proposed filter is a useful tool for isolating DNS traffic for a domain whose certificate is logged in a CT log.

Web traffic

Next, we investigate the HTTP(S) traffic of the experiment's web server to determine whether CT affects HTTP(S) traffic. We compare the results in Section 4.3 of experiment 410a and the control experiment 410b. Additionally, we show the corrected experiment 410a, where traffic from source addresses appearing in the control experiment are removed. The logging facility records all HTTP(S) requests, whether or not they are destined to the domain. Requests destined for the domain will be referred to as domain requests, the remaining are referred to as non-domain requests.

Figure 4.95 shows the cumulative number of incoming HTTP domain requests. In the control experiment 410b, we detect a surge on day 147 of 6259 HTTP requests from a single source, all within 21 seconds. The contents of the requests show that an exhaustive Nikto vulnerability scan was performed by the source on the domain's website. The fields that revealed the nature of the scan were the User-Agent field, which explicitly included "Nikto/2.1.5", and the requested paths. In the scan, we find requests for 6237 unique paths ranging from probes for sensitive files to exploits. We detect another scan originating from this address: on day 147, we detect in both experiments a scan of 99 requests, all within 1 second. Similarly to the Nikto scan, we were able to detect the type of scan by investigating the contents of the request. The path and User-Agent fields reveal that the software used was Nmap.

These two scans did not occur at the same time, but were separated by 1 hour with the earlier scan occurring in the main experiment. Since the two experiments started simultaneously, it is possible that the leak by CT logs might have caused the scan to occur an hour earlier than in the control. However, we deem it improbable because from the DNS traffic analysis; we found that the difference in detection time of the first queries between the two experiments is a matter of days and not hours. We suspect that the cause of the 1 hour difference is because of our setup: the main experiment uses a lower IP address than the control experiment. Supposing that the scanner scans the whole Internet going from lowest to highest address, the main domain would be scanned first because of the lower value.

We remove traffic from this address in both experiments to reveal smaller phenomena in HTTP traffic that are hidden by the large scale of the surge. In Figure 4.96, we depict the same data as in Figure 4.95, but removed any requests from the source discussed above. Without the large surge of over 6000 requests and the two smaller ones of 99 requests, we discover other surges in both experiments, although of much smaller size. In the control experiment, we detect 3 surges of 260 requests on days 54, 111 and 151, with each surge coming from a separate source address. Even though the surges are coming from 3 different sources, we detect the same scanning profile: they

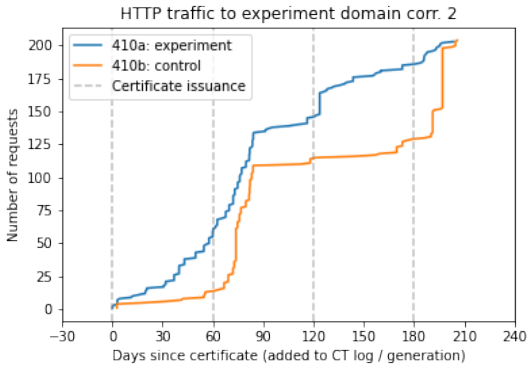

Figure 4.97: Cumulative number of HTTP requests for the domains in experiment 410a and 410b after the removal of traffic from the ASes causing the surges of at least 150 requests. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

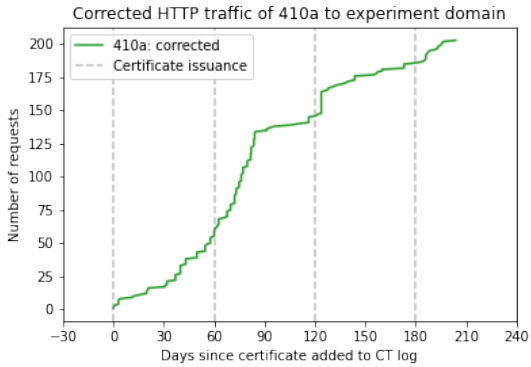

Figure 4.98: Cumulative number of HTTP requests for the domains in corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

share the same generic user-agent that indicates they use a machine running MacOS, they are in the same AS associated with a tech company in China, and the same paths are requested. The scans are probing for sensitive files, but unlike the scan on day 147, the user-agent does not reveal the software that was used. Since the scans only occur in the control experiment, they are not linked to CT.

In experiment 410a, we detect a surge of 195 requests from a single source from an AS based in Hong Kong. The user-agent reveals the host is running MacOS, and the scan is probing for sensitive files. This surge is unlikely related to the three surges in 410b even though there are similarities. They have different scanning profiles: they come from different ASes; they do not share the same user-agent and they request different paths. We suspect this scan is not related to CT because the scan has a similar size than the surges in the control experiment and is not detected close to issuance dates.

After removing the previous 4 surges, we obtain Figure 4.97. In experiment 410a, we detect HTTP requests 60 seconds after the certificate issuance, opposed to 2.7 days in the control experiment. These findings follow the same trend as in the DNS analysis, where the first DNS query for the domain was detected in 20 seconds in the main experiment, compared to 5 days in the control. In both experiments, over 80 % of the requests in the first 90 days originate from the same AS. Even though the traffic is coming from multiple addresses, it is likely that it is the same actor, as all the requested paths are related to WordPress files. The difference between the two experiments is that the scan occurred earlier in the main experiment 410a. The rate of detection is similar, but since the scan starts earlier and therefore runs for a longer period in the main experiment, we detect 20 % more requests. We detect another AS that causes surges in requests in both experiments on day 121 in the main experiment and 196 in the control experiment. Though the surges are of different size, both stem from the same AS. The scan in the main experiment occurs earlier than in the control.

We notice a trend that ASes that scan both experiments often do it earlier in the main experiment than in the control. By introducing an additional source of domain exposure with CT, domains can be discovered more easily and faster by monitoring CT logs than performing DNS lookups for the complete address space. Hence, CT could be the cause for scans to be detected at an earlier rate relative to the certificate generation. Another reason for the delay could be because the lower address that the main experiment uses. As discussed previously, these scanners could cover the whole Internet by going from lowest to highest address, therefore the main domain would be scanned first because of the lower value. Further research is needed to distinguish between the effect of the IP address order and CT.

Figure 4.98 shows the HTTP domain traffic of experiment 410a, but where we removed all the requests coming from sources detected in the control experiment. The correction removed all the surges previously described, leaving an approximately linear trace. The absence of surges shows that the scans detected in the main experiment were not related to CT. We detect no increases in requests at

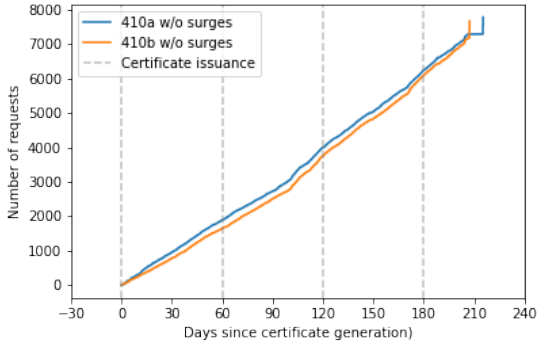

Figure 4.99: Cumulative number of benign HTTP requests detected at the webserver in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

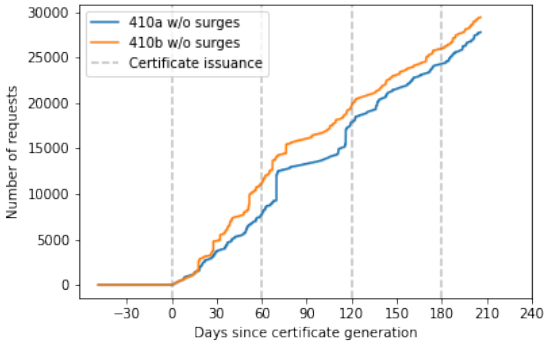

Figure 4.100: Cumulative number of malicious HTTP requests detected at the webserver in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

issuances. From the linear detection rate after the correction and the similar amount of HTTP requests in both experiments, we find CT has little impact on HTTP requests for the experiment domain. The major difference between the experiments is the detection of the first HTTP domain request, which occurs after seconds in the main experiment, but takes days in the control.

In the two experiments, we only create the root page denoted by "/", which is the only accessible page. HTTP requests requesting the root page are referred to as harmless requests. These requests do not cause any harm to the web server and only reveal whether the domain is active. Even though only the root page exists, we detect requests for non-existing paths. Some of the requested paths are pointing to sensitive files, such as configuration or password files. This sensitive information could be used as a foundation for an attack. We detect other requests where an exploit is embedded in the path. Compared to requests for the root page, these requests are a more serious threat to the web server, as the intent of these requests is more malicious in nature. We refer to these requests as malicious requests. Requests for pages other than the root page are not always malicious. Besides the root page, we are classifying requests with the following paths as harmless: '/favicon.ico', '/sitemap.xml', '/robots.txt' and '/.well-known/security.txt'. (The first page is commonly used by browser to display the logo of a website; the next two pages are commonly requested by web crawlers such a Googlebot; the last page is used to display contact information for security researchers to report vulnerabilities.) These pages are among the most requested benign paths with at least 50 requests.

We examine the types of paths requested for the HTTP traffic after the second correction, as in Figure 4.97. Figure 4.99 and Figure 4.100 split up the HTTP traffic to the experiment domain into malicious and benign requests, showing both experiments side by side. We find a similar trace in malicious requests in both experiments, but find significant differences in benign requests. The main experiment detects 40 % more benign HTTP requests for its domain than the control. The differences show CT increases HTTP traffic to the domain, but the increase is mainly in benign requests. Benign HTTP request can be part of an overall malicious scan. We look at whether source addresses that make benign requests are also making malicious ones. They will be referred to as malicious sources, in contrast to benign sources, which only do benign requests. In the main experiment, we find 45 benign sources and 21 malicious, as opposed to 13 and 16 sources in the control experiment, respectively. Not only does the overall number of sources vary between the experiments but also the ratio of malicious and benign sources. We find a general increase in both sources in the main experiment compared to the control, but most of the increase lies in the number of benign sources skewing to a benign-to-malicious ratio of 2.1:1 compared to 1:0.8 in the control. The increase in HTTP traffic from CT comes mainly from sources that only request benign paths. We suspect these sources are only probing whether domains leaked via CT are reachable but do not engage in malicious activity.

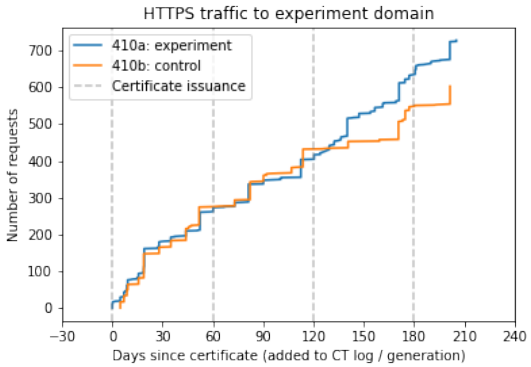

Figure 4.101: Cumulative number of HTTPs requests for the domains in experiment 410a and 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

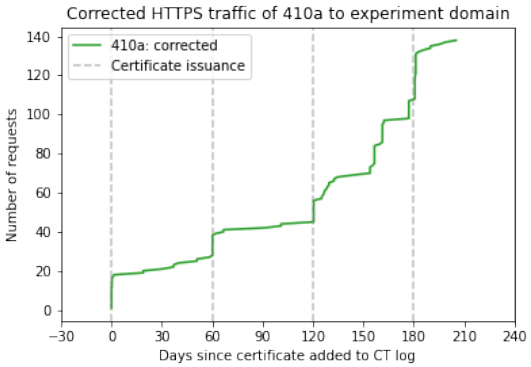

Figure 4.102: Cumulative number of HTTPs requests for the domains in corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.95 shows the cumulative number of incoming HTTPS domain requests. Similarly to HTTP requests, we detect the first request in the main experiment shortly after the leak, 84 seconds, whereas in the control, the first requests are only detected after 4.8 days. Despite the difference, the number of requests is growing at nearly the same rate in both experiments. We only start to see both diverge after day 140, where the rate decreases in the control experiment. In total, we detect 17 % more requests in the main experiment, due to the decrease in rate. Figure 4.102 shows the HTTPS traffic for the experiment domain for experiment 410a, but where we removed all the requests coming from sources detected in the control experiment. The correction with the control group uncovers a hidden pattern: we detect small spikes in requests at the issuances. The originators of these spikes are 5 sources coming from the same AS. As these source addresses generate the same exact number of requests, we suspect it to be one actor controlling the 5 addresses.

Compared non-secured HTTP domain requests, we find clear evidence of traffic tied to CT as we detect spikes in traffic at issuances. The CT related traffic amounts to a 17 % increase in HTTPS domain traffic. From analysing the sources of the requests, we find that only one actor is behind the spikes at the issuances. As the requests always occur within the same day after the issuance, it is likely that this actor is monitoring CT logs for the domains to scan. There are possibly more actors monitoring CT logs and making requests for the experiment domain, but have escaped the correction or because of the low volume of requests, their requests could not be distinguished from background traffic. Overall, there are not yet many actors who use CT logs to gathering websites to scan for via HTTPS.

Figure 4.103 shows the cumulative number of incoming HTTP non-domain requests detected at the web server of the experiment. The total number of requests is nearly identical in both experiments, with the control experiment detecting 4 % more requests. Between the two experiments, we find two major differences: during the first 60 days, the detection rate is higher in the control experiment; we detect a surge of requests in the main experiment on day 69. Further inspecting the requests, we see the requests attempt to exploit the implementation of HTTP parsing of the web server. The correction with the control shows a similar trace as the uncorrected trace of 410a, but with a lower overall detection rate. After applying the correction, the surge on day 69 is not removed, showing that this originator only attacked the main experiment and not the control. The originator of the surge could have extracted the domain from CT logs. Examining the ratio of malicious and harmless requests of HTTP traffic for the experiment domain, we find the same ratio in both experiments with 75 % of malicious requests. These findings show that CT has little to no effect on HTTP non-domain requests. Figure 4.104 shows the cumulative number of incoming HTTPS non-domain requests detected at the web server of the experiment. The total number of requests is nearly identical in both experiments, with only a difference of 5.3 % more requests in favor of the control experiment. Similarly to the non-secured HTTP non-domain requests, the difference in traffic volume between the main and control experiment are of similar

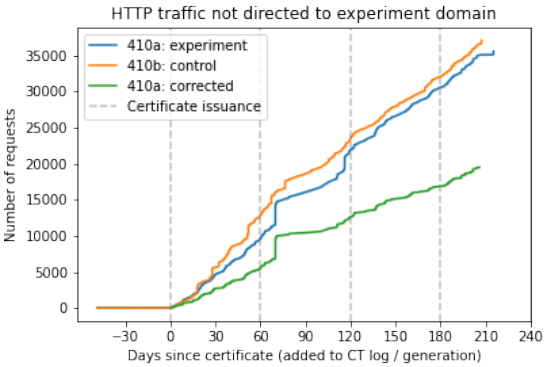

Figure 4.103: Cumulative number of HTTP non-domain requests detected at the web server of the experiments 410a, 410b and the corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

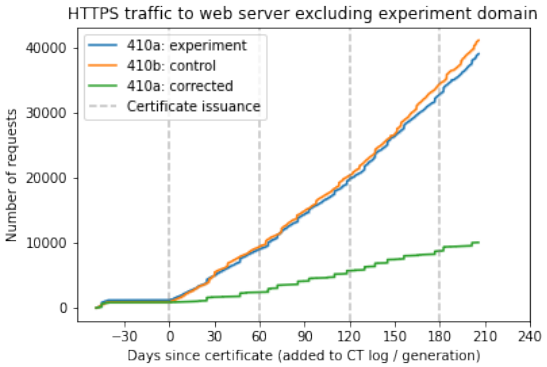

Figure 4.104: Cumulative number of HTTPs non-domain requests detected at the web server of the experiments 410a, 410b and the corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

amount. HTTPS non-domain traffic is growing linearly with the same rate in both experiment. The correction removes 74 % of the requests leaving the same trace as in the uncorrected experiment 410a, albeit with a lower detection rate. The remnants are likely because of the correction method failing to isolate traffic caused by CT, as the corrected traffic retains the shape of the uncorrected traffic. From the overall view of HTTPS non-domain requests, we find no effects of CT.

As was seen in Figure 4.95 where a single surge could hide differences between the two experiments due to scale, the same could apply to the experiment unrelated HTTP(S) traffic, where, for example, smaller surges could be hidden in the mass. We manually examine traffic for the 20 most detected ASes and check if the ASes show a specific behaviour close to issuances. We find one AS that sends requests at every issuance: it is the same source that is responsible for the surges in HTTPS domain requests. This CT-induced traffic only amounts to 0.25 % of the total traffic. Introducing CT in the experiment marginally affects HTTPS non-domain traffic, as we only detect one actor.

In the logs of the web server, we find evidence that HTTP(S) traffic is affected by introducing CT in the experiment. We find the most sporadic differences between the main and control experiment in the HTTP traffic, where large-scale scans are detected at unpredictable intervals. These scans are likely not related to CT but rather part of random scans. Analysing malicious and benign requests allowed for showcasing the effect of CT on HTTP domain requests: while malicious requests remains constant from the control to the main experiment, we detect an increase of 40 % in benign requests. An interpretation for this result is that the scanners responsible for that increase are monitoring CT logs for domains to scan, then probing the website of the domain to check whether it is online, but do not proceed to malicious actions. As for HTTP non-domain requests, we do not find conclusive evidence of CT effects. In HTTPS traffic, for both domain and non-domain requests, we find little differences between the main and control experiments. We manage to find only one actor consistently probing the website within one day of the (re)issuances in the main experiment. At every (re)issuance, the certificate is issued by a CA and appended to a CT log, therefore exposing the domain in public. Because this actor is only making HTTPS requests on the same days as the issuances, it is unlikely that he does not monitor CT logs for domains to scan. The effect is an increase of 17 % more domain requests and 0.25 % more non-domain requests. Even though we found evidence of substantial effects of CT on DNS traffic, the same cannot be repeated for HTTP(S) traffic. We only detect few actors that use CT for scanning websites. Furthermore, the resulting traffic from CT is predominantly benign as mostly benign paths are requested. The traffic increase and changes are only marginal compared to the overall volume of HTTP(S) traffic.

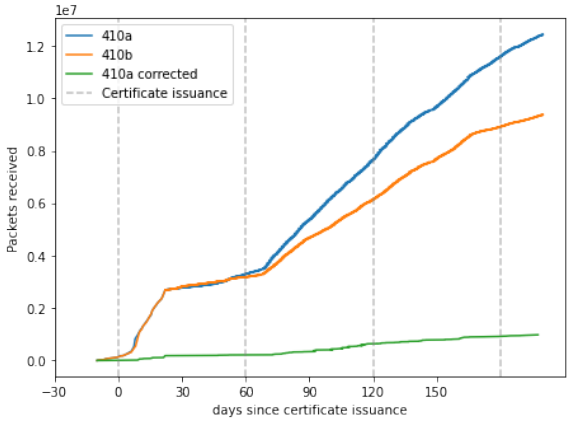

Figure 4.105: Cumulative number of packets in experiment 410a, 410b and corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

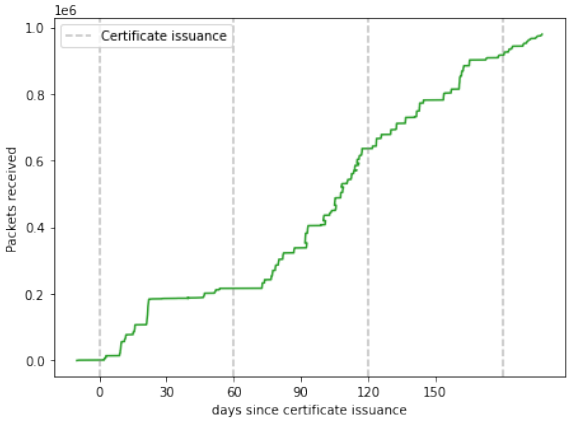

Figure 4.106: Cumulative number of packets in only corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Network traffic

Next, we examine the effect of CT on the network traffic observed at the host of the experiment. Figure 4.105 shows the cumulative number of packets detected at the hosts. We notice two different phases in the traffic. In the first phase, from the start to day 69, we notice that both experiments have the same trace. In the second phase starting on day 69, we detect an increase in detection rate for both experiments, with the main experiment showing a greater increase. Even though the rate increase is different in both experiments, the growth rate in the main experiment shows the same sublinear characteristics. The difference in rate results in 14 % more packets in the main than in the control experiment.

Figure 4.106 shows the cumulative number of packets detected at the host of experiment 410a, but where packets from sources that appear in the control experiment 410b are removed. With Figure 4.106, we try to isolate the effect of CT by removing traffic from source addresses that appeared in the control experiment. Of the total traffic in 410a, 94 % is removed. The resulting shape of the trace is similar than in Figure 4.105. Ideally, the correction would isolate traffic related to CT, while removing all the irrelevant traffic, ie. the correction would remove all the traffic that would appear in both experiments and retain the CT-related traffic in the main experiment. In the second phase, the phase where both experiments diverge, we detect 0.8M packets. But when comparing the second phase of the two experiments, the difference in traffic volume amounts to 3M packets. Since both numbers are not the same, the correction has overcorrected. The difference shows that a part of the increase in traffic – 2.2M packets – come from sources that appear in both experiments, but only choose to generate more traffic in the main experiment. The remaining 0.8M packets come from sources that are exclusively detected in the main experiment. As the correction does not provide insight into how CT affects scanning traffic at the host, we further break down the logged network traffic to see whether any effects can be uncovered.

We further break down network traffic into the different protocols. Figure 4.107, Figure 4.109, Figure 4.111 and Figure 4.113 show the cumulative number of packets detected at the hosts for protocols ICMP, TCP, UDP and GRE, respectively. Figure 4.108, Figure 4.110, Figure 4.112 and Figure 4.114 show their corrected counterpart, where we remove traffic from sources that appear in the control experiment. We ignored protocols for which less than 50 packets were detected, ie. protocols that contribute less than 0.05 % to the total traffic.

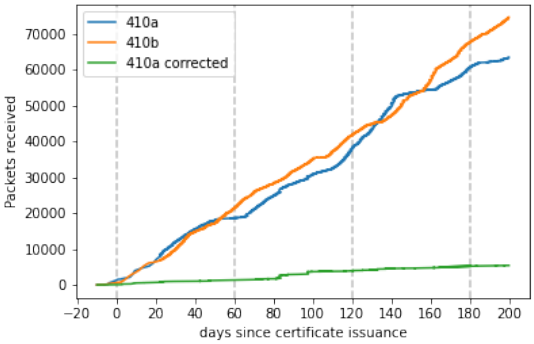

Figure 4.107: Cumulative number of ICMP packets in experiment 410a, 410b and corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

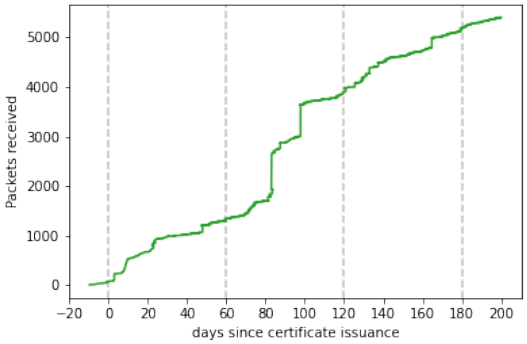

Figure 4.108: Cumulative number of ICMP packets in only corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

In Figure 4.107, we find that ICMP traffic is similar in both experiments. The detection rate is similar, but the traces of both experiments are not identical as in Figure 4.113. We detect 15 % more ICMP traffic in the control experiment than in 410a. Looking at the trace, we cannot find apparent patterns occurring because of issuances. It is not clear how CT affects ICMP traffic, especially that the control experiment detects more packets. Figure 4.108 shows ICMP traffic after the correction, which removed 99.2 % of the traffic. The corrected traffic grows at a linear rate except for two surges on days 82-83 and 97, of 886 and 639 packets, respectively. These surges come each from a single AS, one based in the UK and one in the US. These two ASes are, besides causing the two surges, generating the most ICMP traffic. As these surges do not occur close to reissuances, nor do they occur at predictable intervals, for example always occurring X days after an issuance, we cannot attribute them with high probability to CT. The traffic on the remaining days follows a linear trend similar to the uncorrected traffic. With the control detecting more ICMP packets, we suspect the corrected traffic on the remaining days to be background traffic. The findings show we detect no apparent changes in ICMP traffic due to CT. When examining ICMP traffic per AS one by one, we find one AS that sends ICMP packets only at issuances, 20 to 60 packets on the days of issuance. Traffic from this single AS is the only finding that shows a clear effect of CT on ICMP traffic. The traffic from the AS in question is likely to originate from a single actor, as the source addresses generate the same number of ICMP packets. ICMP traffic remains largely unaffected by CT, only one actor was detected to have a scanning behaviour related to CT.

Figure 4.107 shows TCP traffic of both experiments, 410a and 410b. As TCP traffic constitutes 98 % of the traffic, we obtain a near identical figure as Figure 4.105. Similarly, the correction results in the near identical Figure 4.108 compared to Figure 4.106. Isolating TCP traffic does not give further insight into the effects of CT on network traffic.

Figure 4.111 shows UDP traffic of both experiments, 410a and 410b. Overall, UDP traffic is growing at the same constant rate in both experiments. But, we detect two differences between the experiments. We detect surges of similar size – around 22K packets – on days 135 and 137 in experiment 410a and 410b, respectively. The surge occurred 2 days earlier in the main experiment with CT than in the control group. The earlier occurrence in the main experiment could be caused by CT, but could also be because the network was scanned sequentially, and thus scanned the main experiment first, since the used IP address of the main experiment, when converted into an integer, is smaller than the one used in the control experiment. Another difference we observed is that right after the start of the experiment, we have an increase in traffic for 15 days in the main experiment. In the control, a smaller, shorter and later increase is detected compared to the main experiment. To sum up, we detected two differences between the two experiments: the main experiment 410a experiences an earlier and greater increase of traffic at the start of the experiment, and a surge around day 130, 2 days earlier than the control. The difference between the increases leads to a difference of 6K UDP packets in favor of the main experiment.

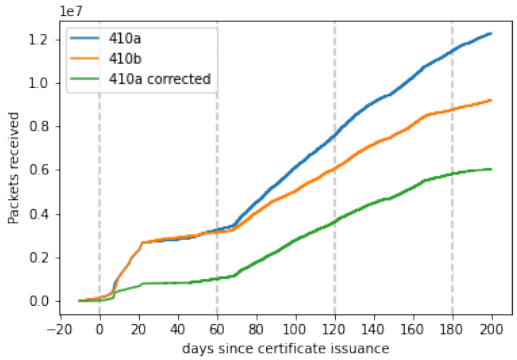

Figure 4.109: Cumulative number of TCP packets in experiment 410a, 410b and corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

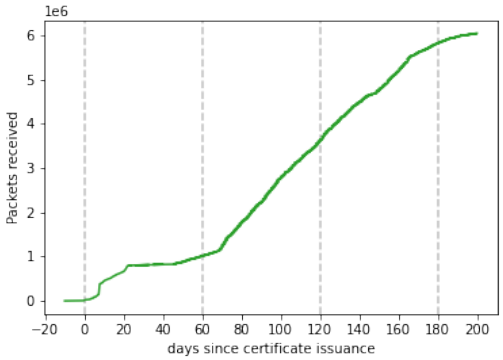

Figure 4.110: Cumulative number of TCP packets in only corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

Figure 4.112 shows the corrected UDP traffic. The correction removed the surges around day 130 in both experiments, showing that they originated from the same source. We suspect that the source scanned the whole network in ascending order and therefore scanned the main experiment first because of the numerical order of the used IP addresses. The correction stressed the differences we observed at the start of the experiments. It seems that the initial issuance, hence the first leak, exposed the IP address of the domain host and caused an influx of UDP traffic in the following 20 days. The influx results in 6K UDP packets, coinciding with the difference observed between the main and control experiment in Figure 4.111. It is an indication that the correction accurately isolated traffic related to CT. Examining the first 20 days of the corrected traffic, we find that the increase in traffic predominantly originates from ASes based in Korea, 67 %. 85.0 % of the traffic during that period is NTP traffic using the same source and destination port 123. After day 20, the presence of Korean sources decreases and is taken over by three ASes, based in China, generating 24.2 % of the traffic for the remaining days. Even without the Korean ASes, NTP remains the most hit protocol, but only accounts for 10 % compared 85.0 % during the first 20 days. The next 2 most hit protocols are NCADG and DNS, accounting for 7.3 % and 8.3 % of the total traffic. The remaining protocols are under 5 %. When examining UDP traffic per AS, we could not find any ASes, which showed a predictable behaviour in relation to issuances. The effect of CT on UDP traffic is limited to the first 20 days, where we observed a greater and earlier increase in traffic for the first 20 days of the main experiment, originating from Korea and mainly composed of NTP traffic. We deem this increase to be likely stemming from the presence of CT in the experiment. Subsequent issuances did not affect UDP traffic.

Figure 4.113 shows GRE traffic in both experiments, 410a and 410b. The traffic trace looks identical in both experiments, indicating that the traffic in both experiments occurred independently of CT. Figure 4.114 corrects the GRE traffic of the main with the control experiment resulting in the same shape as in Figure 4.113, thus not revealing any potentially hidden phenomena. The detected GRE traffic is unsolicited. Botnets such as Mirai have been seen to use the GRE protocol in flooding attacks [47]. But due to the low traffic volume, the detected GRE traffic is unlikely to be a DDoS attack. Because we get the near same trace in both experiments, we deem GRE traffic to be unaffected by CT.

Figure 4.117 shows the composition of the ports targeted in the network traffic for both experiments 410a and 410b, as well as the corrected experiment 410a. We only show ports that account for at least 1 % of the total traffic. Because of the cutoff threshold, ports show in Figure 4.117 are exclusively TCP ports. The predominant port is port 22 or SSH protocol accounting for 57 and 59 % of the total traffic in 410a and 410b, respectively. For the remaining 4 protocols, we detect a slightly higher ratio in the main experiment than in the control experiment. More traffic on ports 80 and 443, for protocols HTTP and HTTPs, is expected as CT leaks a certificate tied to a domain, and therefore traffic to the web server hosting the domain should experience an increase. The two other ports, 8291 and 8728, ports used by MikroTik routers, have not such a connection to CT. MikroTik routers were found to

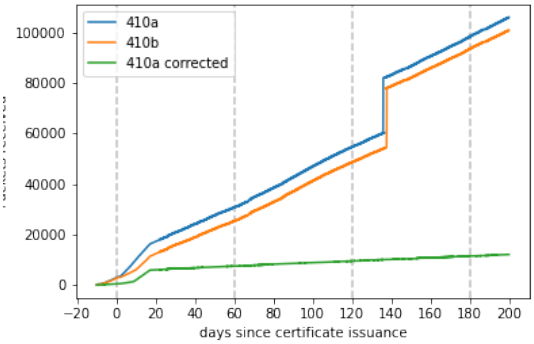

Figure 4.111: Cumulative number of UDP packets in experiment 410a, 410b and corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

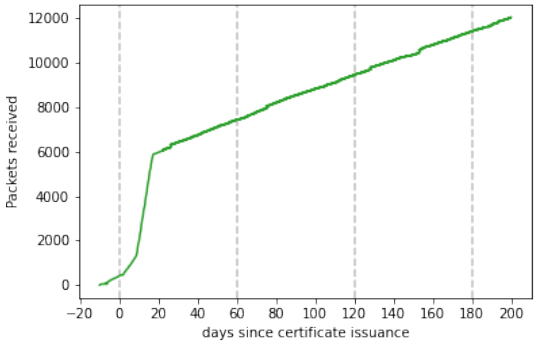

Figure 4.112: Cumulative number of UDP packets in only corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

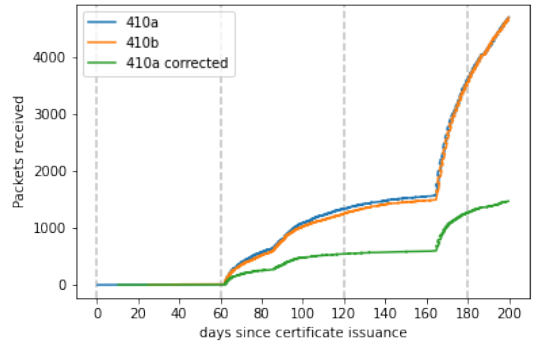

Figure 4.113: Cumulative number of GRE packets in experiment 410a, 410b and corrected 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

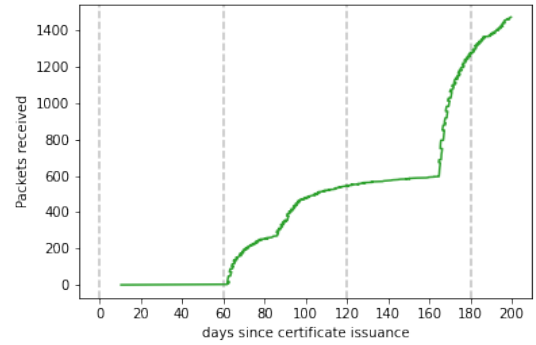

Figure 4.114: Cumulative number of GRE packets in only corrected experiment 410a. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b. The grey dotted lines indicate when the certificate of the domain was issued or reissued.

have serious vulnerabilities in 2018, allowing attackers remote code executions [6][10]. As no MikroTik router is running on the addresses used in the experiments, any traffic detected at the MikroTik ports is considered vulnerability scans. Introducing CT in the main experiment leads to a higher ratio of HTTP(S) traffic as well as MikroTik scans at the addresses used to host the domain of the experiment. To compensate for the increase in the ratio of the aforementioned ports, we see a slight decrease of 2 % in SSH traffic and a decrease of 4 % in traffic from the remaining ports. The correction of 410a, denoted by 410a* in Figure 4.117, skews towards the MikroTik and HTTP(S) ports while reducing the ratio of SSH and the remaining ports. It confirms our findings that the increase in traffic towards the HTTP(S) and MikroTik ports seen in the main experiment is not due to background traffic but to CT, as the correction does not reduce but increases their ratio.

Figure 4.118, shows the absolute number of packets detected at the top 5 destination ports as opposed to the relative composition of the traffic. We always detect more packets in the main experiment than in the control for the top destination ports. Even though the ratio of SSH traffic is lower in the main than in the control experiment, the volume detected is larger. The correction removes traffic in ports 22, 80 and 443, while MikroTik ports remain largely unaffected. Removal of HTTP(S) and SSH traffic by the correction shows that, with or without CT, the presence of a web server and an open SSH port is sufficient to attract traffic towards to the corresponding ports. Since not all the traffic is removed, the remnants confirm that CT affects all the top 5 ports, as was already discovered when comparing the traffic volume between the two experiments.

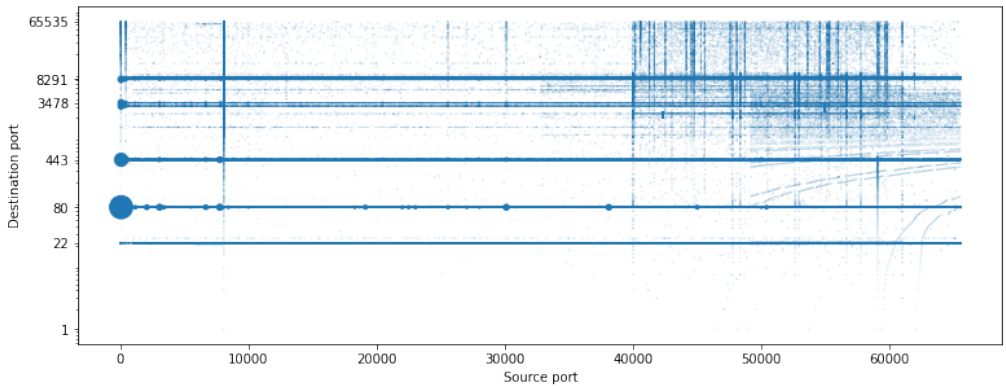

Figure 4.115: Relation between TCP source and destination ports in the corrected experiment 410a. Traffic from sources appearing in the control experiment 410b were removed. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

Figure 4.119 shows the absolute number of packets from the top 10 source ports. The difference between the experiment 410a and 410b is more prominent for source ports than destination ports. In experiment 410a, the most used source port is port 80, a port from the privileged port range. The port is used over 4 times more than in the control. On the other hand, the control experiment detects port 8080 as the most used port, about 15 % more than in 410a. Some notable difference between the two experiments are ports 443, 22 and 53, which are more prevalent in experiment 410a than the control. The remaining ports are detected a similar number of times. The main experiment with CT shows overall an increase in traffic from ports in the privileged range, but also a decrease in traffic from port 8080 compared to the control without CT.

With Figure 4.115, we show the relation between TCP source and destination ports for the corrected experiment 410a. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports. The correction has removed a lot of traffic that was not within vertical or horizontal scans, compare with Figure 4.115. Most of the removed traffic were using source ports from 30000 onwards. We see that there are a significant amount of vertical scans detected at the host, most of them using source ports greater than 40000. There are 4 scans in the lower destination port ranges, where the source port was incremented, resulting in a curve. The heavy traffic from the lower ports to the HTTP(S) ports still remain after the correction. Scanning traffic due to CT uses more predictable scanning patterns, where the scanner uses the same port number to perform vertical scans. There are only a handful of scanners that are incrementing the source port after each packet. Since the correction removed mostly traffic that was not forming clear vertical or horizontal lines, it shows that scanners that target the CT-leaked domain tend to not randomize their source ports to obfuscate their scan of the domain host.

We perform the same analysis on the UDP traffic in Figure 4.116. The correction removes nearly all the traffic, especially traffic with source port greater than 30000. The result of the correction leaves faint traces of 5 vertical scans around port number 50000. We notice a patch of traffic in the upper right corner, targeting the highest destination ports. The correction confirms our suspicion that UDP traffic is not affected much by CT in the system as found in Figure 4.111.

We section the IPv4 address space into 256 /8 bins and examine how much traffic each bin generates. Figure 4.120 shows the difference in traffic generated per bin of experiment 410a and 410b. A positive value indicates more traffic in experiment 410a. 177 bins generate more traffic in the main experiment compared 29 in the control. However, the largest difference is detected in bin 134, which is generating 380K packets in the control experiment, but is generating 30K packets in the main. The difference between the experiments is of a factor of over 10. When investigating the sources within the bin, we find one address generating 311K packets, but appears not to generate any traffic in the main experiment. Since we detect this address only in the control experiment, traffic from this source is likely

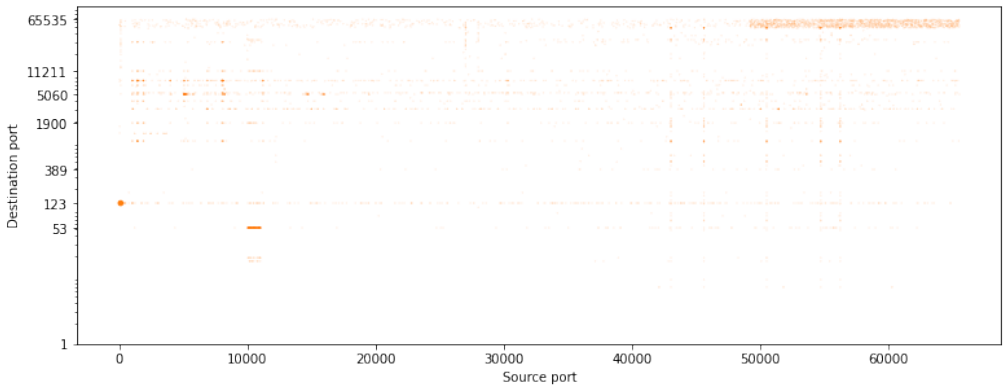

Figure 4.116: Relation between UDP source and destination ports in the corrected experiment corrected. Traffic from sources appearing in the control experiment 410b were removed. The size of dots gives a qualitative representation of the number of packets. The Y-axis is logarithmic to put focus on the well-known port range for destination ports.

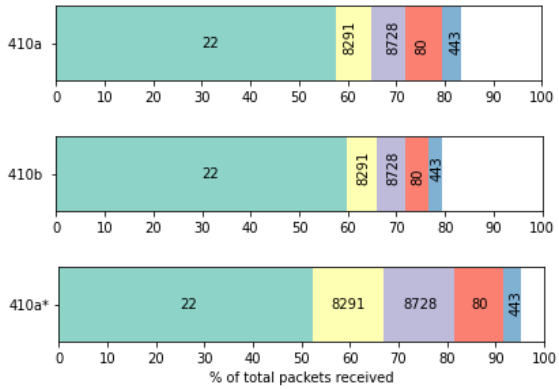

Figure 4.117: Composition of network traffic by destination port for experiments 410a, 410b and corrected 410a. Only destination ports that constitute at least 1 % of the total traffic are shown. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b.

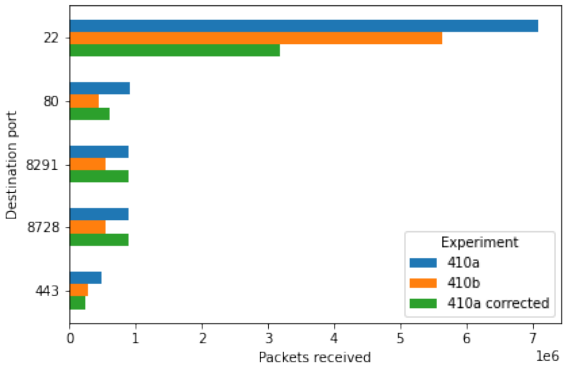

Figure 4.118: Number of packets by destination port for experiments 410a, 410b and corrected 410a. Only destination ports that constitute at least 1 % of the total traffic are shown. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b.

from an arbitrary scan on the host, unrelated to the experiment run. Overall, we detect an increase in traffic generated per /8 bin in the main than control experiment. We perform the same analysis on an AS-basis. The findings show that 50.1 % of the ASes detected in both experiments generate more traffic in the main experiment than in the control with a mean of 10853 packets. 30.6 % ASes generate more traffic in the control experiment, but with only a mean of 62 packets. The remaining 19.3 % are generating the same amount in both experiments. Breaking down traffic per AS shows that the introduction of CT in the experiments leads to an increase in traffic, with an average of 10K more packets, in over 50 % of the ASes. ASes that generate more traffic in the control experiment, do at a much lower average volume of 62 packets, insignificant compared to the 10K additional packets generated.

4.4.2. Effect in IPv6

In this section, we determine the effect of CT on scanning traffic at a host in the IPv6 address space. We evaluate experiment 610a and use the control experiment 610b as a baseline. In experiment 610a, we are hosting a domain on one address in IPv6 and leak its name via CT by requesting a certificate from LE. The control experiment 610b follows the same setup except that the certificate is self signed to prevent it from being automatically added by a CA to a CT log. In the control experiment, we capture no traffic at the host of experiment, nor do we detect any DNS queries for the experiment domain or

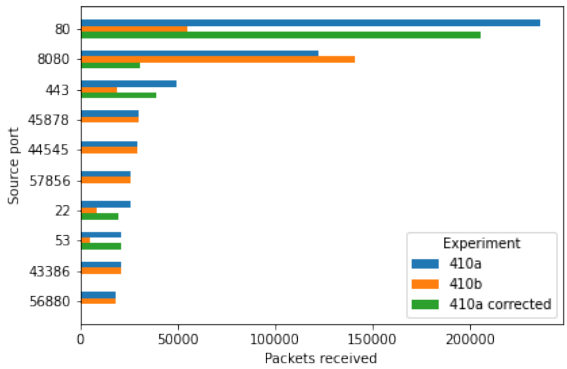

Figure 4.119: Number of packets by source port for experiments 410a, 410b and corrected 410a. The top 10 source ports are shown. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b.

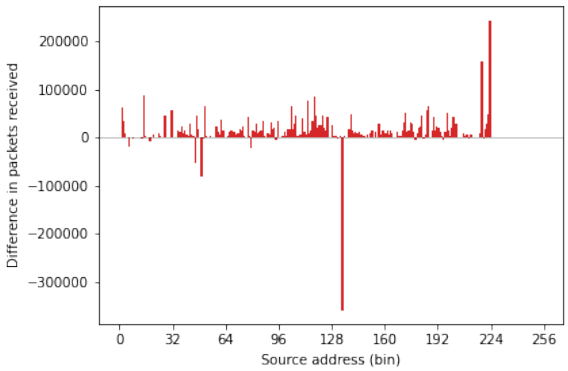

Figure 4.120: Difference in packets received by address bin from experiment 410a and 410b. The IPv4 address space is sectioned into 256 /8 bins. A positive value indicates more traffic in the main experiment 410a than in the control 410b. A negative value indicates more traffic in the control experiment 410b than in the main 410a.

the host address. Since nothing is detected in the baseline, any traffic at the experiment host or DNS queries related to the domain are the consequence of CT in the system.

At the authoritative DNS server, we detect an average of 1978 DNS queries per experiment run with a standard deviation of 127. The first query for the domain arrived on average 18.4 seconds after the certificate issuance and the addition to a CT log. Within the same first day, we detect on average 148 queries: the initial issuance causes a surge in traffic. The effects of CT are detected quickly. Not only does the initial issuance affect DNS traffic, but also subsequent issuances, which occur at a 60 day interval. The reissuances also trigger surges similar to the surge after the initial issuance. The size of the surges are of similar size showing that the actors that are querying our domains do not track which domains they already have queried. They indiscriminately query any domains they encounter when monitoring CT logs. CT does not only affect DNS traffic on the days where the certificate is issued, but also in between: we receive requests for the domain at a constant rate. While we detect a significant increase for DNS queries for the domain, reverse lookups, or queries for PTR records, remain rare.

The domain is collected by scanners from a CT log shortly after the certificate issuance, since we detect a DNS query after 18.4 seconds. Though, it takes over 20 days for first network packets to be detected at the host of the domain. At the host of the domain, we detect 2673 packets with a standard deviation of 261. After the first packets, we detect them at a constant rate until day 210, where we observe an increase in traffic for 20 days. This increase is occurring in all runs of the experiment. Most traffic is HTTPS traffic on port 443 accounting for over 53 % of the total traffic, followed by HTTP traffic on port 80 accounting for 39 % of the total traffic. Since the domain was leaked via CT, it means that there is a valid certificate associated with the domain. The certificate is often used for secured HTTPS traffic, therefore we expect to detect HTTPS traffic, which coincides with the findings. The remaining protocols, that are detected, are DNS, SSH and SNMP, each accounting for less than 5 %. Looking at the sources of the traffic, we find 88 addresses in use. We partition the address space into 256 /120 subnets and notice that out of the 256, only 4 subnets generate traffic, with one subnet generating over 80 %. From the packet analysis, we find that there are not many actors that are using CT logs to discover and scan IPv6 hosts. Traffic in IPv6 is sparse and only originates from a handful of actors.

HTTP(S) traffic constitutes 8.8 % of the packets received. The web servers detect on average 235 HTTP(S) queries. The detected traffic was collected in 220 days. As was seen in the network traffic, we detect the first HTTPS requests after 20 days. HTTP traffic is only detected 90 days later. We find similar results than in the overall network traffic, especially that HTTP(S) accounts for 92 % of the total network traffic: Issuances do not affect HTTP(S) traffic and we detect little deviation between experiment runs. When looking at the host requested in the HTTP(S) traffic, we find that over 80 % of the requests are not for the domain of the experiment.

Our findings show that the effect of CT is detected swiftly at the DNS server, but takes considerably longer to appear at host of the domain. It is important to note that the DNS server is within the IPv4 address space, while the host is not. Unlike DNS traffic, network traffic does not seem to be affected by reissuances, as in no surges are detected after an issuance. The different address spaces might be the cause for this difference.

For more detailed information on the effects of CT on scanning traffic of a host in IPv6, please refer to Section 4.1.4, Section 4.2.4 and Section 4.3.4,

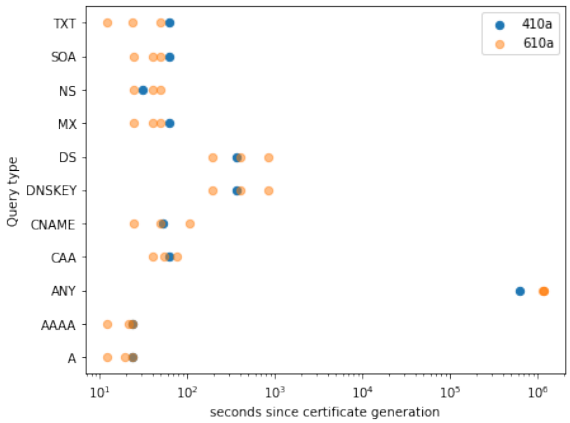

Figure 4.121: First DNS query containing honeytoken of experiment for every DNS query type in experiments 410a and 610a.

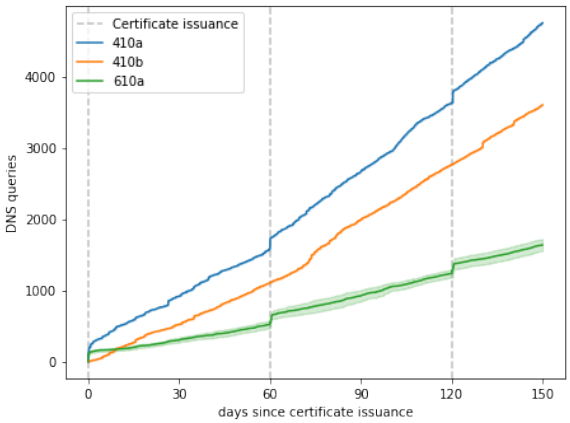

Figure 4.122: Average cumulative number of DNS queries in experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

4.5. Differences between v4 and v6

In this section, we compare the effect of CT on scanning traffic between experiments running in IPv4 and IPv6. We show how CT affects scanning traffic depending on the address space using our findings in Section 4.4.1 and Section 4.4.2 to determine how scanning traffic differs and therefore find an answer to the research question 1.3 How does Certificate Transparency affect scanning traffic in the IPv4 address space compared to the IPv6 address space? We compare the results from the three data sources – DNS, webserver and network traffic – using the IPv4 experiments 410a and 410b, and the IPv6 experiment 610a.

Experiments in IPv6 have the advantage over their IPv4 counterpart in isolating the effects of CT. In the IPv6 control experiments, we do not detect any traffic at the host or at the DNS server, therefore providing an extremely simple baseline to compare with. Any traffic detected in the main experiment can be attributed to CT without the need to correct other sources of domain leak. IPv4 experiments do not share the same privilege. Studying the effect of CT in IPv4 is more complicated as we do not have a control experiment with no traffic. IPv4 results require isolating traffic related to CT from non-related traffic. The control experiment 610b is therefore omitted, as no traffic was detected in all three data sources.

4.5.1. DNS data analysis

The DNS infrastructure in both IPv4 and IPv6 is shared: they use the same CA with the same certificate issuance process as well as the same DNS infrastructure, which is running on IPv4. Only difference is that in IPv4, the domain is stored in an A record, as opposed to a AAAA record in IPv6. Certificates do not specify in what address space the domain is hosted in. Therefore, any third-parties that monitor CT logs cannot determine the address space of the leaked domain. For the DNS analysis, the results in Section 4.4.1 have shown that, the correction of experiment 410a, where traffic from source addresses appearing in the control experiment 410b, is ineffective. The correction barely removes any traffic. Therefore, in the analysis, we compare both the main and control experiments in IPv4, with the IPv6 experiment 610a.

Figure 4.121 shows the detection time per query type of the first query containing the honeytoken, the randomly generated string in the domain name of the experiments, for experiments 410a and 610a. The detection time indicates how long it takes from the domain leak to the first query. For experiment 610a, there is a detection time for each of the three runs. The detection times of both experiments are always in proximity. The largest difference in detection times detected, albeit still small, is for ANY

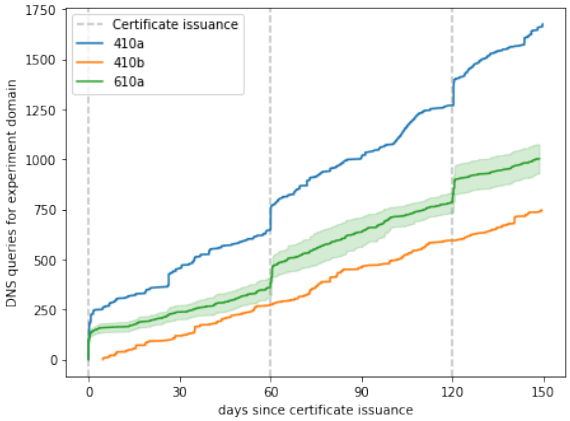

Figure 4.123: Average cumulative number of DNS queries with the domain name of experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

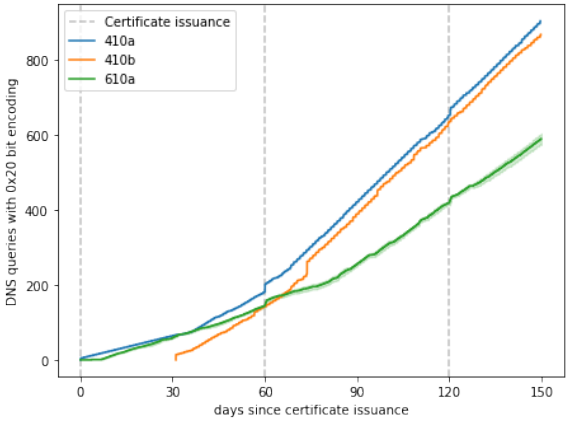

Figure 4.124: Average cumulative number of DNS queries with the domain name encoded with the 0x20 bit of experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

records, which also boast the longest detection times than any other query types. The results show that the address space does not matter in the detection time. As discussed in the paragraph above, CT logs reveal nothing about the hosting location of a logged domain. Third parties monitoring CT logs cannot differentiate between domains hosted in IPv4 or IPv6.

Even though CT reveals nothing about the hosting location, we find a difference in the volume of DNS queries in Figure 4.122, where we show the average cumulative number of DNS queries in experiments 410a, 410b and 610a. While queries detected for experiment 610a are solely attributed to CT, queries in experiment 410a are not, since there are other factors at play other than CT. As a simple correction method to isolate CT traffic in the IPv4 experiment, we take the difference of the main experiment 410a with the control 410b to obtain 1136 requests. We will refer to this correction method as volume correction. We find a similar number in the IPv6 experiment 610a with 1637 requests. Another similarity is the surges on issuances in both address spaces. Though, the initial issuance causes a greater surge in the experiment 410a. It shows that there are scanners who ignore domains leaked because of the reissuance, but only limited to domains hosted in IPv4.

Figure 4.122 shows the average cumulative number of DNS queries with the domain name of experiments 410a, 410b and 610a. Using the same correction method as before, we find that correcting experiment 410a with 410b results in 927 requests, very close to the 903 requests in experiment 610a. Same as before, we see that in both address spaces that a surge of queries is detected at the issuances and that the initial issuance causes a greater surge in IPv4 than IPv6. We do not find the same results in Figure 4.124, where we show the average cumulative number of DNS queries with the domain name encoded with the 0x20 bit in experiments 410a, 410b and 610a. There is only a difference of 36 queries between the main experiment and the control on IPv4, while the IPv6 experiments detect 590 queries. In Section 4.4.1, we came to the conclusion that CT barely affects these types of queries in IPv4. However, we see that in IPv6, a significant amount of queries were detected. Based on this, we determine that CT causes more DNS queries with 0x20 bit encoding in IPv6 than IPv4.

Figure 4.122 shows the average cumulative number of DNS queries for non-existing subdomains of experiments 410a, 410b and 610a. These requested subdomains are likely part of a subdomain enumeration attack. Using the volume correction, we get 75 requests in 410a and 21 in 610a. The results show that CT causes 3.5 times more queries from subdomain enumeration attacks in IPv4 than IPv6. Subdomain enumeration usually starts with a request of the domain before proceeding to subdomains. It is possible that some scanners limit their enumeration to A records, records with IPv4 addresses. Hence, fewer queries are detected for domains hosted in IPv6. Figure 4.122 shows the average cumulative number of reverse lookups in experiments 410a, 410b and 610a. Similarly to

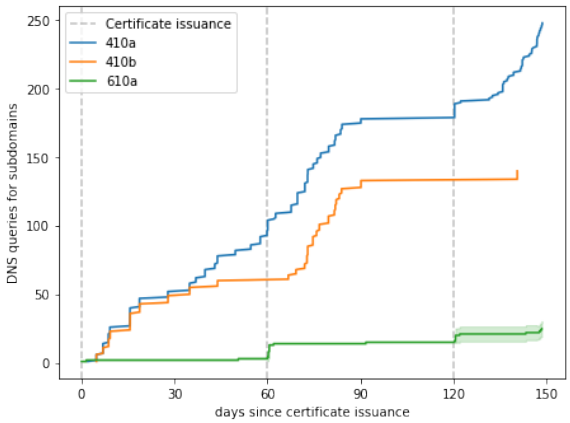

Figure 4.125: Average cumulative number of DNS queries for non-existing subdomains of experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

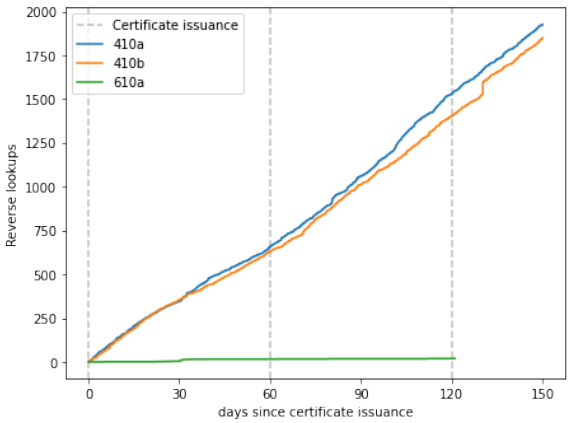

Figure 4.126: Average cumulative number of reverse lookups of experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

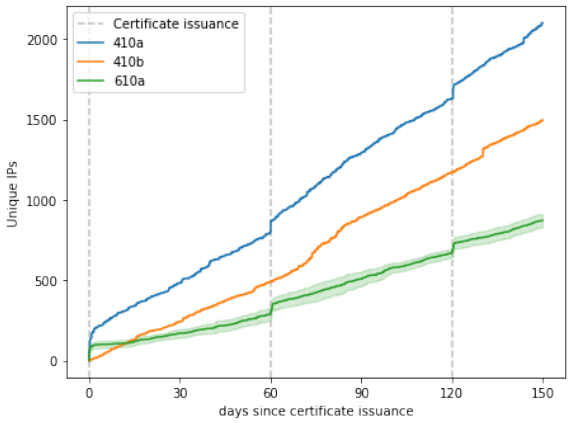

Figure 4.127: Average cumulative number of unique IPs in experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

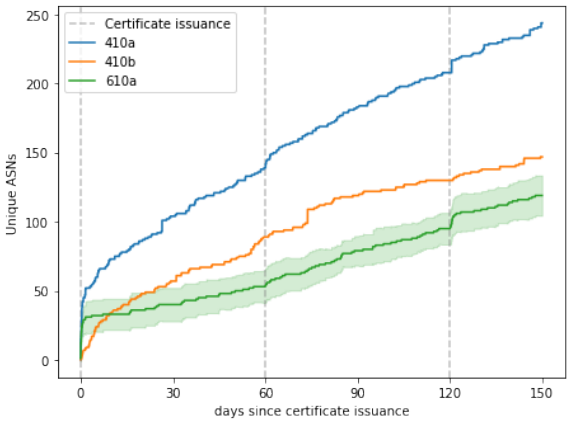

Figure 4.128: Average cumulative number of unique ASes in experiments 410a, 410b and 610a. The grey dotted lines indicate when the certificate of the domain was issued or reissued. The shaded area shows the standard deviation.

Figure 4.124, we have determined in Section 4.4.1 that CT only has a negligible influence on reverse lookups. The detected reverse lookups are third-parties scanning the whole or part of the IPv4 address space. Since IPv6 address space is considerably larger than IPv4, such scanning practices are not feasible. Confirming our suspicion, we detect only 20 reverse lookups in 610a. Relating back to CT, we find that CT only marginally affects reverse lookups in both address spaces.

We divert our focus to the originators of the DNS queries. Figure 4.122 shows the average cumulative number of unique IPs in experiments 410a, 410b and 610a. The trace follows a similar shape than Figure 4.122, where we show the total DNS traffic. Surges at the issuance show that a burst of new addresses are detected at the surges coinciding with the cumulative DNS traffic. Using the volume correction gives 1136 unique addresses compared with 1637 in IPv6. It is not a sign that CT causes more addresses to query the experiment domain, because the number of DNS queries needs to be taken into consideration. Since the volume correction also leads to fewer queries in IPv4 than IPv6, this difference in addresses is expected. Figure 4.122 shows the average cumulative number of unique IPs in experiments 410a, 410b and 610a. It explains what type of sources are detected at reissuances. Even though there is a surge of addresses at reissuances, there is none for ASes, only at the initial

Protocol	410a	610a	410a corrected
TCP	12274853 (98.6 %)	5925 (75.8 %)	6041954 (99.7 %)
UDP	106249 (0.9 %)	499 (17.8 %)	12054 (0.2 %)
ICMP	63388 (0.5 %)	1389 (6.4 %)	5411 (0.1 %)
Other	4757 (0.0 %)	0 (0.0 %)	1478 (0.0 %)

Table 4.3: Overview of Internet protocols detected in experiments 410a, 410b and 610a. Under the columns named after the experiments, you find the number and percentage of packets that were detected for each protocol indicated by column 'Protocol'.

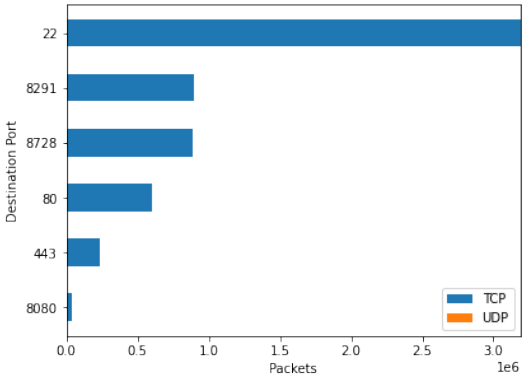

Figure 4.129: Number of packets received at the top 6 destination ports in the corrected experiment 410a. Blue bars show packets from TCP traffic and orange bars show packets from UDP traffic. However, UDP traffic is minuscule compared to TCP, hence not showing up in the plot. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b.

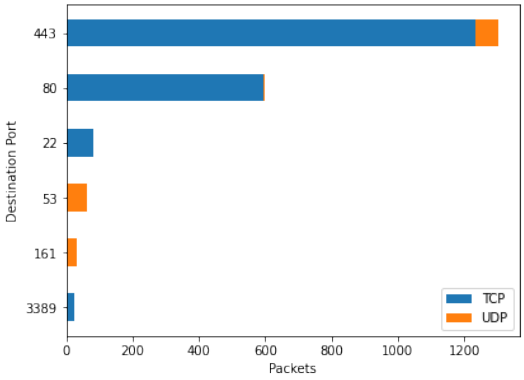

Figure 4.130: Number of packets received at the top 6 destination ports in experiment 610a. Blue bars show packets from TCP traffic and orange bars show packets from UDP traffic.

issuance. It shows that the burst of addresses are from ASes that were already detected. We suspect that these addresses come from the same scanners, who are simply rotating their address pool.

4.5.2. Network traffic

Unlike the shared DNS infrastructure, IPv4 and IPv6 experiments are not hosted in the same address space. In the following analysis, we isolate the effect of CT in experiment 410a using the correction, where traffic from sources detected in the control experiment 610b are removed. Table 4.3 shows which Internet protocols are detected in the experiments. We find a vast difference in traffic volume in the IPv4 experiment, even after the correction, than in the IPv6 experiment. We find two similarities between the two address spaces: The order of the 3 most detected protocols remains. However, we find different ratios. TCP dominates in IPv4, while in IPv6, more is allocated to the UDP and ICMP protocols, which account for less than 1 % in IPv4. The results show that CT cause vastly more scanning traffic in IPv4 than IPv6. It is possible that the difference does not lie in CT but in the imperfect correction method that does not perfectly isolate traffic related to CT. Even so, we believe that CT still causes more traffic in IPv4 than IPv6, since IPv6 adoption is slow.

We further investigate TCP and UDP traffic by analysing the ports used. Figure 4.129 and Figure 4.130 show the average cumulative number of packets received at the destination ports in the corrected experiment 410a and experiment 610a. Between the two experiments, we find that they have 3 out of the six most hit destination ports in common: ports 22, 80 and 443. While port 22 dominates in IPv4, it is only in third place in IPv6. There are fewer SSH connection attempts (port 22) because of CT in IPv6. Reversely, we find HTTP(S) ports at the front in the IPv6 experiment, accounting for 89 %

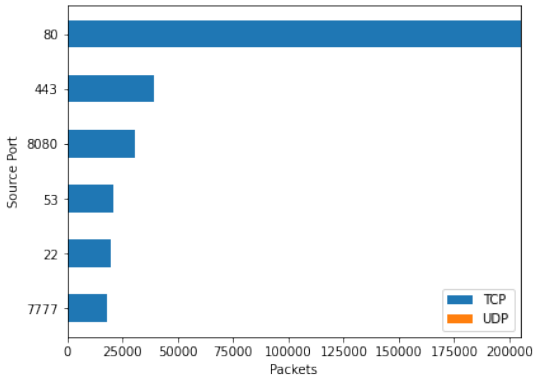

Figure 4.131: Number of packets received at the top 6 source ports in the corrected experiment 410a. Blue bars show packets from TCP traffic and orange bars show packets from UDP traffic. However, UDP traffic is minuscule compared to TCP, hence not showing up in the plot. In the corrected experiment 410a, we remove all traffic from sources detected in the control experiment 410b.

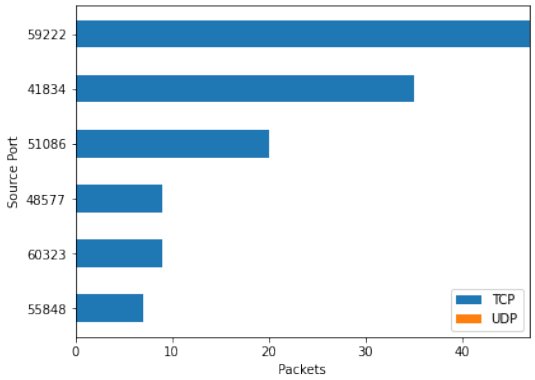

Figure 4.132: Number of packets received at the top 6 source ports in experiment 610a. Blue bars show packets from TCP traffic and orange bars show packets from UDP traffic. No UDP traffic was detected from the 6 most detected source ports.

of the total traffic. But the difference to 410a is only in proportions. Table 4.3 showed that network traffic in the IPv4 constituted of less than 1 % UDP. The disparity between UDP and TCP makes that UDP is not visible in Figure 4.129. In IPv6, we see significant UDP traffic at destination ports 443, 53 and 161. We detect scans for the vulnerable MikroTik routers in IPv4, but no traces in IPv6. From the observations, we find that CT affects scanning traffic differently based on the address space: in IPv4, there is proportionally more scanning traffic for services SSH and MikroTik, whereas in IPv4, scanning traffic is focused on the HTTP(S) ports. Because of the relation between CT and HTTPS, we expect traffic to the HTTP(S) ports. But in IPv4, the primary target are not those services. There is more traffic towards exploitable services: a weak user-password combination in SSH or outdated MikroTik software are easy targets. We do not see this trend in IPv6.

To end the port analysis, we look at the source ports. Figure 4.129 and Figure 4.130 show the average cumulative number of packets received at the source ports in the corrected experiment 410a and experiment 610a. The analysis of source ports in IPv6 shows that, in the top 6 most detected source ports, all of them are in the ephemeral range. Looking at the complete distribution, we find that the source port selection is fairly uniform in the ephemeral range. There are no single ports that are considerably favored. We find a different result in IPv4. While source ports in the ephemeral range are uniformly distributed, the 5 most used ports come from the privileged range used for well-known ports. The top 3 source ports detected are HTTP ports: 80, 443 and 8080. The comparison shows apparent differences between the two address space in terms of the selection of source ports. Most of the scanning traffic due to CT comes from privileged ports in IPv4, but from ephemeral ports in IPv6. Using privileged ports for source ports is considered as bad practice, since it requires elevated privileges. Furthermore, detecting source ports in the privileged range is more suspicious than source ports uniformly distributed in the ephemeral range. We have two hypothesis for the difference in port choice. Since IPv4 is the more popular address space, it is more likely that novice scanners use IPv4 to learn the craft. Due to being inexperienced, they do not follow the best practices in source port selection and choose ports from the privileged range. Another reason might be because IPv4 is older than IPv6; 17 years difference between the two [40][24]. It is possible that some of the scanning traffic detected in IPv4 stems from machines compromised a long time ago. The scanning methods found on those machines date back where no thoughts were put into the source port choice.

SSL	To domain	410a	610a	410a corrected
		35605 (46.9 %)	40 (5.7 %)	20091 (65.5 %)
x		39010 (51.4 %)	338 (47.8 %)	10101 (33.0 %)
	x	503 (0.7 %)	11 (1.6 %)	312 (1.0 %)
x	x	728 (1.0 %)	61 (8.6 %)	138 (0.5 %)

Table 4.4: Overview of HTTP traffic in experiments 410a, 410b and 610a. The SSL column shows whether the SSL protocol was used, i.e. HTTPS. The column 'To domain' indicates whether the domain requested contains the generated honeytoken of the experiment. Under the columns named after the experiments, you find the number and percentage of requests that were detected.

4.5.3. Web server traffic

In this section, we compare how CT affects HTTP(S) traffic in IPv4 and IPv6. The traffic recorded at the web server is only a subset of the complete network traffic. Table 4.4 summarizes what type of requests were detected in experiment 410a and 610a. And we also compute the results for the corrected experiment 410a, where we isolate requests due to CT by removing HTTP requests from sources detected in the control 410b. We differentiate between HTTP and HTTPS, and between domain requests – requests for the domain of the experiment – and non-domain requests – requests not for the domain of the experiment but for other domains that are not hosted on the experiment host. This gives 4 types of requests. We give the number of requests for each type and the percentage of all HTTPS requests in the experiment. We compare the corrected experiment 410a with experiment 610a, i.e. the difference between IPv4 and IPv6. We detect for all types of queries more in IPv4 than IPv6. Both experiments record fewer domain queries than non-domain queries, but the IPv6 experiment has a higher percentage of 10.2 % compared to 1.5 % in IPv4. It seems that CT causes more focused HTTP(S) requests to the experiment domain in IPv6. Probable reason for non-domain queries is to target the webserver itself to either find a flaw or enumerate domains that it hosts. With that in mind, in IPv4, not only is the domain the target, but also the hosting web server.

4.6. Effect of CT on network

In this section, we evaluate the effect of CT on scanning traffic in a network. The analysis is limited to IPv6 networks. We examine network traffic from the whole network instead of focusing on the website host. This section relates back to the formulated research question 1.2 How does Certificate Transparency affect scanning traffic seen by the network of a website host? The relevant experiments to the analysis are 610a and 610b.

In experiment 610a, we are only running one host listening on 1 IPv6 address. The remaining addresses in the /64 subnet are left untouched. Traffic directed to these unused addresses is recorded. The control experiment without CT does not capture any traffic, nor does the authoritative DNS server detect reverse lookups for unused addresses in the network. The baseline is no scanning traffic. In the main experiment, we detect no packets for addresses in the subnet except for the address in use. We find the same result at the authoritative DNS server, where no reverse lookups of unused addresses were detected. The findings show that in the IPv6 address space, leaking a domain via CT does not result in any scanning traffic in the /64 subnet where the web host of the domain is located.

5

Discussion

In this chapter, we discuss the all the results computed until now. Using our findings, we compare the current state of CT misuse in 2021 to 2018. We evaluate the severity of the impact of CT on scanning traffic at the technical infrastructures. In light of the new current state, we determine whether the benefits of CT still outweigh the downsides of increased scanning traffic. We examine label redaction, a countermeasure to the effects of CT, to not expose sensitive domains via CT. The following section includes recommendations and best practices to protect hosts against the CT-induced scanning traffic. We finish the chapter with a section where we list potential further research that could not be covered in this paper.

5.1. Update on CT misuse since 2018

The experimentation performed for this paper reveals the amplitude of the effect of CT on scanning traffic at 3 places: DNS server, webserver and domain host. The findings show the current state in 2021. Our experimental setup is based on the experiment model, CT honeypot defined by Amann et al. [46] in 2018. Using their results, we can compare how the misuse of CT has evolved over 3 years. Since our experimental setup is based on the model from Amann et al. [46], they have a lot in common. There are two differences. We first enumerate the differences between the setups, and argue that the differences in setup are not causing different outcomes.

The randomly generated subdomain label in our experiments has 13 characters (see Section 3.2.1), as opposed to 12 characters in the experiments of Amann et al. [46]. We believe that the length of the subdomain label does not play any role in the CT-induced scanning traffic. Scanners monitoring CT logs for domains have no reason to differentiate between subdomain labels of 12 or 13 characters. Subdomain labels come in a variety of sizes, from 1 to 63 characters. Ignoring or favoring certain label lengths does not give the scanner any benefits, but only fewer scanning targets. Moreover, the length of the domains without the subdomain label of our experiments is likely different from the one used by Amann et al. [46]. In both research, the domain was redacted to ensure privacy. Therefore, we deem this difference to be inconsequential to the outcome of the experiments.

The second difference is the inclusion of a PTR record in the DNS configuration of the subdomain. RFC 1912 imposes that, for every domain, there must be a PTR record [7]. Amann et al. [46] chose not to follow RFC 1912, because PTR record leaks the domain through rDNS walking. We correct for this source of leakage by running, for every experiment, a parallel experiment where the certificate is not added to a CT log. These experiments act as control of the main experiments, allowing to correct for leakage sources of the domain other than CT. The advantage of our experiment setup is that we can correct for all sources of leakages, whereas Amann et al. [46] only eliminate discovery through

rDNS. There are other sources of leakage that cannot be controlled by the experiment settings. The subdomains in the experiments can be stored in DNS threat intelligence, like Farsight's DNS database, but the initial leakage comes from CT logs. HTTP and HTTPS traffic hold the domain name in clear text, which could be retrieved by eavesdroppers. In HTTP, the domain name is stored in the host field of the HTTP header. In HTTPS, the domain name is sent in clear text during the TLS handshake in the SNI field. But the findings of Amann et al. [46] only limit to the first detected queries. We have found that the sources of domain leakage other than CT do not affect the detection time of the first queries at the DNS server or at the webserver.

As we have discussed, the differences between the experimentations does not affect the outcome of the first detected queries, we have a common base to compare. Starting with IPv4 experiments, we detect the first DNS queries for the domain of the experiment 23.3 seconds after appending the certificate of the domain in a CT log (see Section 4.1.2). Amann et al. [46] detect the first query after 73 to 197 seconds, or an average of 3 minutes. Taking the shortest detection time, we find it is has reduced by more than a factor of 3 in the span of 3 years. The difference in the HTTP(S) queries is even more extreme. We detect the first query for the experiment domain after 60 seconds for HTTP and 84 seconds for HTTPS (see Section 4.3.2). Amann et al. [46] have a detection time of 59 minutes to 15 days, which is a reduction factor of at least 42.

We suspect two causes for this significant reduction in detection time. First, scanners might retrieve certificates from CT logs at a shorter interval, leading to near live monitoring. During 3 years, the prolonged exposure of the CT infrastructure enabled more and more scanners to discover how CT logs can be misused and become more familiar with its inner workings. Likely, CT monitoring has improved and allowed for more frequent retrieval of the CT tree. Second, we think CT logs update their logs more frequently. Requests for certificate appending to a CT log are processed faster: it takes less time for the CT log to append the certificate to the tree and update the view presented to monitors. Both causes are not independent: the faster processing by CT log might encourage scanners to poll the view more frequently. When scanners poll the view too often, then some of the views returned by the CT log will be duplicates. This leads them to reduce their polling rate. Reversely, if the poll rate is lower than the update rate, then scanners will be presented with always different views. The poll rate can be slowly increased until duplicate views are encountered. When this point is reached, the poll and update rate are nearly equal. The scanner discovers new domains as fast as is possible.

Comparing IPv6 experiments, we find the most drastic difference. We go from no CT-induced scanning traffic at the experiment host in 2018 to 2700 inbound packets detected. We detect the first DNS queries after 19.4, 23.7 and 12.2 seconds, similar values to the IPv4 experiments (see Section 4.1.4). The first HTTP queries are detected after 77, 54 and 47352 (outlier) seconds; HTTPS queries are detected much later, after 110, 110 and 65 days (see Section 4.3.4). Disregarding the outlier in HTTP, the first HTTP(S) queries are detected earlier in IPv6 than in IPv4 in 2018. We believe that the causes listed above for IPv4 also apply for IPv6. In 2018, introduction of CT in a system only caused scanning traffic in IPv4 and not IPv6. Since IPv4 was and is the dominant address space of the Internet, it was likely targeted first by the new scanning method. In 3 years, IPv6 adoption rate has increased from 20 % to 35 % [20]. The strong and seemingly steady growth of IPv6 renders it attractive to scanners to invest resources in. Not only is the growing adoption of IPv6 a good reason, but using CT specifically for IPv6 scanning has an inherent benefit. The vast size of IPv6 makes exhaustive address space scans, such as rDNS walking, infeasible. IPv4 address space is 2*96 times smaller and is densely populated, and therefore it permits for this brute-force approach. IPv6 is the opposite: a vast and sparsely populated address space. It is not possible to use CT to discover only IPv6 hosted domains as the location of the domain is not divulged in the certificate stored in the log. But using DNS, it is simple to determine in which address space the domain is hosted in. The response to an A or AAAA query reveals the address space, IPv4 or IPv6, respectively. This method facilitates the discovery of IPv6 domain hosts. Exhaustive scans are not possible; random scans in the address space are unlikely to find success due to the sparsity.

The comparison between the result of our and Amann et al. [46] studies has shown that CT misuse has increased from 2018 to 2021, and lead to faster discovery of domains hosted in IPv4 or IPv6. The consequences of CT on scanning traffic in IPv4 poses less of a concern, as IPv4 hosts are already

subject to substantial scanning without the involvement of CT. CT-induced scanning traffic contributes to 14 % of the total traffic (see Section 4.2.2). However, IPv6 hosts detect no packets without CT to 2700 packets with CT (see Section 4.2.4). In the future, we expect more misuse of CT, especially for scanning hosts in IPv6 due to the increasing adoption of IPv6 and to the inherent advantages of host discovery with CT in the vast and sparsely populated address space. We recommend for further monitoring to be done to track the evolution of CT misuse in IPv6.

5.2. Abolish CT?

After having examined the effect of CT on scanning traffic and analysed its evolution, are the advantages of CT in making the Certificate ecosystem more transparent worth the rapid detection of certified domains and increased in scanning traffic at the NS servers, webservers and domain hosts? CT could be abolished or domain owners could simply opt out when requesting certificates. Removing CT from the cyberspace can be done easily on a personal basis. We will evaluate the negative effects of CT on scanning traffic and determine whether they outweigh the benefits.

The increase in traffic volume due to CT that was detected in our research is not substantial enough to exceed the infrastructures system and network capacity. In IPv4, the added load at the DNS server is an increase of 40 % in queries, or 1207 queries, concentrated shortly after a certificate issuance. On these days, we observe surges of at most 250 queries. At the domain host, we detect an increase of 14 % in network traffic (4M packets), but evenly distributed with no surges. At the web server in the CT experiment, we observe less traffic than in the control, because of exhaustive vulnerability scans independent of the experiment (see Section 4.4.1). In IPv6, the additional load on the infrastructure is much lower than in IPv4: IPv6 hosts capture less than 0.1 % of the traffic volume (2.7K packets) at IPv4 hosts (see Section 4.5.2). Same observation is found at the webserver (see Section 4.5.3). At the DNS server, we find similar results than in IPv4 experiments but little to no reverse lookups. The similarities are because the DNS server is shared between the IPv4 and IPv6 experiment. See Section 4.5.1. The lack of reverse lookups in IPv6 is likely because it is not feasible to bruteforce IPv6 due to the size of the space. Fiebig et al. [15] have shown that it is possible to scan the whole address space by exploiting NXDOMAIN, but no such scanning was detected.

We believe that this increase in traffic should be tolerated by any properly equipped infrastructure. If the added load overwhelms the system capacity, it is rather an indication of a poor system. Webservers, and therefore the domain host (the domain host is running the web server), and DNS servers are usually not only hosting one domain and therefore should have enough bandwidth and resources to host them with little downtime. Regarding surges at issuances, it is unlikely that all the domains are issued at the exact same time, causing a cumulative surge. Failing to guarantee good uptime for customers is a sign of a bad system that does not protect customer domains against traffic fluctuations, for example, increased legitimate traffic or malicious activities. Such an issue reduces customer satisfaction and can lead to loss of customers. It is a logistical and financial matter to have adequate load capacity. Hence, the increase in traffic due to CT should not strain the capacity of DNS and webservers.

Sensitive domains are preferable not to be advertised publically. Their sensitive nature makes them attractive targets for malicious actors. Certificates might be required for the sensitive domains in order that any communications to them are secured. However, browsers such as Chrome require certificates to be inserted into CT logs. This poses the problem that the sensitive domains are exposed as soon as the certificate is issued, since it is added directly to a CT log. Hiding domains to keep them secure can be classified as security by obscurity. Such a practice violates the second axiom of Kerckhoff's principle which states that a "system must not require secrecy and can be stolen by the enemy without causing trouble' [39] (translated from French: "Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi." [29]). Domains should be maintained secure no matter if the domain is leaked via CT or not. It should be expected that the domain will be discovered. Therefore, we deem that CT leaking domains is not a major security issue, as domains should not rely on security by obscurity.

We have analyzed and discussed the downsides of CT and determined that they do not introduce a significant weakness in the Internet ecosystem. Having a more transparent certificate ecosystem is largely more beneficial in assuring a secure environment. CT is only harmful in already weak and vulnerable systems. Here, the problem lies in the system and not in CT. By default, infrastructure should be well equipped and secure to be robust against various threats. The addition of CT in such systems should not cause any difficulties. Certificate aggregation is no new concept of CT. Services such as Censys or scans.io have provided certificate databases for some time.

There are ways to reduce the downsides which will be covered in the following sections. We explore how to reduce the exposure of domains in CT and propose best practices to make infrastructure more robust against CT-induced scanning traffic. For administrators that absolutely do not want to expose sensitive domains, we provide alternatives.

5.3. Mitigations

In this section, we evaluate mitigations to the downsides of CT. We explore how to reduce the domain exposure in CT logs, such that the logged certificates do not reveal the complete domain name, and how to minimize the security threat from CT-induced scanning traffic.

Hall [22] proposes a new feature to CT that allows to obfuscate the subdomain label in order to partially hide the sensitive domain. The CT log named Deneb and operated by Symantec implements a similar idea by truncating subdomain labels [2]. Another idea is hashing the domain name instead of leaving the domain name in cleartext. However, the popular browser Chrome shows no intention of backing this new feature, as it violates the browser's policy [50].

Using hashes instead of domain names in CT logs introduces a new vulnerability to the CT system [54]. Hashes can be exploited using dictionary attacks. For example, a variation of a subdomain enumeration can be performed, where the crafted subdomains are hashed and then compared to the values stored in CT logs. Furthermore, domain names tend to have low entropies as they follow a language, making it even more vulnerable to attacks on hashes [53].

Redacting part or the entire domain name in CT logs lowers the effectiveness of CT. It complicates the detection of misissuances, as the complete domain name cannot be read by retrieving the logs. It would require additional contact with the CT logs holding the certificate to retrieve the domain name associated with the logged certificate [54]. Compromised or misbehaving CAs can hide part of phishing domains used to for malicious attacks [19]. Hiding subdomains for security reasons is relying on security by obscurity, which is considered a bad security design principle (see Kerckhoff's principle [39]). Allowing redaction in CT will probably lead to over-redaction, because domain owners would prefer to hide their domain name to 'improve' security [54]. Redaction works against the purpose of CT in making the certificate ecosystem more transparent.

CT has had a positive security impact by making the certificate ecosystem more open for monitoring. Redaction solves the issue of domain leakage through CT logs, but at the same time contradicts in what CT stands for. No further progress and development were made since 2016. For domains where redaction is necessary, administrators should question whether CT is the right solution.

Instead of weakening CT, we suggest that cybersecurity best practices should be followed, not only when introducing CT logs, but at all times. Web servers, websites, domain hosts and networks are already subject to scanning traffic and malicious traffic without CT. Though CT hastens the discovery of domains and increases scanning traffic, they should already be robust against these threats. Failure to do so is more likely an indication that best practices were not followed, rather than CT being the culprit. For example, DNS and web servers that use insecure or outdated hosting software, that have misconfiguration exposing sensitive information or that do not posses enough resources to deal with traffic fluctuations of benign or malicious nature will be vulnerable to any attackers whether or not they used CT to find it. CT just adds a small threat vector to an already vulnerable system. Similarly, security should always be taken into consideration during web app development and when setting up hosts

connected to the Internet. Relying on secrecy of the website or host to shield them from malicious actors is not advisable. One should expect that a system will be discovered, and therefore, should secure the system appropriately, as stated by the Kerckhoff's principle.

5.4. Alternatives

While CT helps to publicly disclose issued certificates from trusted CAs and therefore allow rapid detection of missuances, some people or organizations prefer to forego the public exposure of their domains through logged certificates. We provide some alternatives to the regular CT-logging of certificate at issuance.

Wildcard certificates are certificates that match to multiple domains. The wildcard label * is placed at the most specific domain label. For example, the certificate *.example.com can be used by both domains sensitive.database.example.com and website.example.com. Using wildcard certificates hides part of the domain name and therefore limits domain name disclosure in CT logs. This solution allows to partially hide domains while profiting from the limited benefits of CT – limited due to the redaction. However, wildcard certificates need to be treated with care. As wildcard certificates can be used for multiple domains, it becomes harder to track what domains are certified. There is a risk that the wildcard certificate vouches for a rogue domain controlled by a malicious actor. The rogue domain would appear as legitimate due to the validity of the wildcard certificate in use [45]. Furthermore, National Security Agency (NSA) has warned that wildcard certificates are vulnerable to Application Layer Protocols Allowing Cross-Protocol Attacks (ALPACA) [38], a technique that exploits wildcard certificates to compromise TLS handshakes [8]. Though wildcard certificates do not expose domains in CT logs, they introduce new security vulnerabilities and reduce the benefits of CT. Opting for this solution requires careful management of certificates. We think that if keeping domains hidden is a requirement, one should question whether CT is necessary. The following alternatives will cover this more radical approach of eliminating CT altogether.

To avoid disclosing domains via CT, a more radical approach can be taken by foregoing CT. Some CAs allow to request certificates without logging them in CT logs. However, this alternative comes with the downside that some browsers and software will issue a warning when using these certificates to establish TLS connections: These connections will be classified as insecure. It is possible to add an exception for the certificate in browsers, software or operating systems. All benefits from CT would be abandoned. But if the domain is accessed by the public, it is not realistic to expect all the users to set up their software to avoid CT verification. Not removing the verification will show a warning to the user who might distrust the services hosted in the domain. Therefore, foregoing CT is only a viable option for private domains accessed by a group of people, such as organizations. [19]

Organizations can run private and local CA for their domains. The locally trusted CA is manually installed on all the devices of the organization and the CT verification is removed for certificates from this CA. It allows for users in the organization to connect to domains without any warnings. The organization's domains will not undergo CT verification, while other domains retain the added security of CT. Note that certificates from locally trusted CAs are not accepted by CT logs. Such a system does not expose domains and keeps the internal structure of the organization private. However, it comes at a cost: running a private CA adds a security risk to the organization. If the local CA is compromised, all the users of the organization are put in danger, as all their traffic can be eavesdropped. For traffic to the organization's domains, the malicious actor can decrypt it because he has access to the private keys of the domains. The remaining traffic is vulnerable to MITM attacks, since the attacker can misissue certificates for the contacted domains, as the compromised CA is trusted by the devices of the organization. The security of the CA is not the only concern: configuring and updating all the devices to trust the CA adds considerable work load.

Depending on the use case, opting for one of the alternatives might be the best course of action. Even though they do help to not publicly advertize domains, additional security risks are introduced and the benefits of CT are lost. Careful consideration needs to be taken to evaluate whether the trade-offs are justifiable.

5.5. Further work

We have studied how CT affects scanning traffic at DNS servers, webservers and domain hosts. As our work is partially based on the work of Amann et al. [46], we were able to compare the results from 3 years prior. It would be beneficial to replicate parts of the experimentation in the future, to track how CT-induced scanning traffic evolves. Special attention should be put to domains residing in IPv6. The comparison between Amann et al.'s [46] and our findings have shown that CT is slowly being leveraged to find IPv6 scanning targets. The vastness and sparseness of the IPv6 address space renders exhaustive scans of the space infeasible. CT logs are ideal sources of IPv6 to scan. The hosting location of the address space is not revealed in the CT logs, but this information can easily be retrieved by DNS. With the steady adoption of IPv6, it will be interesting to monitor how CT-induced scanning traffic will evolve.

In our experiments, trusted certificates were reissued by the CA at an interval of 60 days. At each of the reissuances, we detected a surge in DNS traffic as at the initial issuance. DNS traffic between the issuances was elevated compared to the control experiment. But we have not experimented without reissuances. Are expired certificates in CT logs exploited and, if yes, for how long? How long does the elevated traffic last before returning to the baseline? The two questions tackle how long the effect of CT lingers in a system.

Our work only looks at a subset of the negative effects of CT. We propose new research ideas that can help to better understand the full scope of CT. The experiments only covered one type of device, namely computers. A similar study could be done on different types of devices, such as IoT devices and how CT affects their scanning traffic. Similarly, we have examined only one application protocol, HTTP, running over TLS and could extend the study to other protocols running over TLS.

The analysis can be extended from scanning traffic to attacks. This can be achieved by running honeypots at the various contact points in the attacker model, in our case, the DNS server, domain host, and webserver. This would determine whether the increased scanning traffic is accompanied by an increase and more sophisticated attacks.

Because of the low availability of IPv4 networks, we were not able to analyse the effect of CT on networks, as was done for IPv6 experiments. It would involve repeating our IPv4 experiments with each dedicated IPv4 network. In Section 5.4, we have touched upon wildcard certificates to partially hide a domain name. Since the CT-logged certificate does not hold a complete domain name, the question arises how this information is used for scanning.

6

Conclusion

In this thesis, we have determined with experiments how Certificate Transparency affects scanning traffic in computer networks (main research question). The performed experiments set up CT honeypots, where we disseminated random domain names acting as honeytokens to lure scanners to our hosts. We examine the CT-induced scanning traffic from 3 data sources: the authoritative DNS server of the domain names, webservers hosting the domains, domain hosts and their subnetwork. All traffic is recorded at the 3 data sources and analysed. Experiments are run in both IPv4 and IPv6 address space. The findings from the experiments helped to answer the 3 sub-questions derived from the main research question.

RQ 1.1 How does Certificate Transparency affect scanning traffic seen by a domain host? Before looking at computer networks, we focused the analysis on the host affected by CT. Starting with the first detected queries at the DNS, we find that it only required on average 19.7 seconds to detect, from the moment when the certificate was added to a CT log, and by consequence, the embedded certificate was leaked publicly. Experiments from both address spaces show approximately the same detection times. Refer to Section 4.1.2 and Section 4.1.4. Comparing the current results to 2018 in the findings of Amann et al. [46], we find that the detection time has decreased by approximately 160 seconds. Similar results were found in IPv4, for the first HTTP(S) queries that were detected after 60 seconds – Section 4.3.2 and Section 4.3.4 – compared to 59 minutes in 2018 [46]. Within 3 years, domains are being scanned quicker after being leaked via CT. Out of the 3 data sources analysed, we find that the DNS server shows the most apparent effect of CT in Section 4.4.1 and Section 4.4.2. At the initial issuances and subsequent reissuances, we detect surges in requests for the leaked domain. The initial issuance experiences the largest surge, while reissuances experience smaller but constant ones. It shows that some scanners monitor CT logs for domains to scan, but ignore any domains that were already discovered through CT: they keep track of encountered domains. As such, they do not scan domains twice and therefore can save on scanning resources. The phenomenon is detected for experiments in both address spaces. Moving on to the web server, we notice differences between experiments from the two address spaces. In IPv4, we find marginally more HTTP(S) requests for a domain whose name was leaked in a CT log. The first HTTP(S) request for the domain is detected after 60 seconds. In contrast, the control experiment, where the domain is not leaked via CT, detects the first request only after 2.7 days. We propose a correction method, where we use a control experiment without CT to isolate CT traffic from background noise. The corrected traffic shows small jumps in HTTP(S) requests on the days after the issuances of certificates. The findings show that CT causes a small increase in HTTP traffic for domains, whose certificates are logged in CT, and earlier scanning traffic. The effect of CT is only limited to the CT-logged domains and not to other domains hosted on the webserver. Further details are found in Section 4.4.1. In IPv6, the earliest HTTP(S) query for the domain is detected 54 seconds after the domain's certificate was appended in a CT log. IPv6 experiments show the effect of CT clearly: the control experiment without CT detects no traffic at all. Hence, any traffic detected is bound to CT and therefore no correction is needed. The number of

HTTP(S) requests is growing linearly and seems to remain unaffected by reissuances, occurring at 60 day intervals. The complete results can be found in Section 4.3.4. In Section 4.4.1, we take a step back and look at the all the scanning traffic detected at the domain host in IPv4. Our findings show that the TCP protocol is the most affected by CT: we detect an increase in 25 % of TCP traffic. For the remaining IP protocols, we detect nearly no effect of CT. Examining TCP traffic further, we find that the CT-induced scanning traffic not only targets the HTTP ports where the domain is hosted but also the SSH and Mikrotik ports, used by Mikrotik routers, which had serious vulnerabilities in 2018 [6]. Traffic to these ports is likely of malicious nature to find SSH servers with weak credentials and Mikrotik routers with outdated software. In IPv6 hosts, TCP is also the predominant protocol, but different ports are targeted than in IPv4. The scanning traffic is mainly focused on the HTTP ports, 80 and 443. The remaining services hit account for less than 5 % and are services DNS, SSH and SNMP. Mikrotik ports are not targeted.

RQ 1.2 How does Certificate Transparency affect scanning traffic seen by the subnet of a domain host? We extend the analysis of how CT affects scanning traffic to a whole /64 subnetwork in IPv6. All inbound traffic was recorded on the subnetwork. The findings show that, in the IPv6 address space, leaking a domain via CT does not result in any scanning traffic in the /64 subnetwork other than the domain host. Scanners do not use the CT-leaked domains to do further reconnaissance on the subnetwork of the leaked domain.

RQ 1.3 How does Certificate Transparency affect scanning traffic in the IPv4 address space compared to the IPv6 address space? Using the results from RQ 1.1, we can compare how CT affects scanning traffic in IPv4 and IPv6 domains. At the DNS, the major difference between experiments from the two address spaces is the lack of reverse lookups in IPv6. Note that the IPv6 experiments are served by a DNS server running in IPv4. As the leakage via CT does not reveal any information on the address space used, scanners cannot differentiate between domains hosted in IPv4 or IPv6. All the DNS differences are found in Section 4.5.1. At the webserver, we notice that the volume of HTTP(S) requests that CT causes is vastly different. We detect 30k requests caused by CT in IPv4. The number of requests in IPv6 is approximately 1 % of the requests in IPv4, or 450 requests. Furthermore, IPv4 domains favor HTTP traffic opposed to HTTPS traffic by IPv6 domains. All the webserver differences are found in Section 4.5.3. Examining all the scanning traffic of the domain, we find that CT causes an additional 6M packets to be detected at the domain host in IPv4 versus 8K packets in IPv6. In addition to the large size difference, the traffic is distributed differently: in IPv4, TCP dominates with 99.7 %, while IPv6 shows a more balanced distribution with TCP (75.8 %), UDP (17.8 %) and ICMP (6.4 %). Regarding TCP ports, we discover that the most of the traffic is targeted at the HTTP ports in IPv6. In IPv4, domain hosts have also their SSH and Mikrotik ports targeted, showing that scanners look beyond the HTTP ports.

Our findings show that CT affects scanning traffic of domains hosted in both address spaces. We deem that the effects of CT on scanning traffic in IPv4 poses less of a concern, as IPv4 hosts are already subject to substantial scanning without the involvement of CT. However, IPv6 hosts detect no packets when domains are not logged in CT. CT logs are being used to detect IPv6 hosts for scanning, which was not the case 3 years prior [46]. With the increasing adoption of IPv6, we expect more misuse of CT for scanning domains hosted in IPv6. We suggest continuing monitoring of how CT affects scanning traffic in IPv6. Despite that CT increases scanning traffic, we believe that the net security gain is positive. CT makes the certificate ecosystem more transparent by helping to detect misbehaving CAs faster. This largely outweighs the increased scanning traffic. Hosts should not rely on being hidden to be secure. Security by obscurity is a bad security principle to adhere to. Therefore, we expect that the hosts are robust against any scanning traffic, whether the cause was CT or not. CT is an invaluable tool to keep users on the Internet safe.

Bibliography

[1] Maarten Aertsen, Maciej Korczyński, Giovane C. M. Moura, Samaneh Tajalizadehkhoob, and Jan van den Berg. No domain left behind: Is let's encrypt democratizing encryption? In *Proceedings of the Applied Networking Research Workshop*, ANRW '17, page 48–54, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351089. doi: 10.1145/3106328.3106338. URL https://doi-org.tudelft.idm.oclc.org/10.1145/3106328.3106338.

[2] Johanna Amann, Oliver Gasser, Quirin Scheitle, Lexi Brent, Georg Carle, and Ralph Holz. Mission accomplished? https security after diginotar. In *Proceedings of the 2017 Internet Measurement Conference*, IMC '17, page 325–340, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450351188. doi: 10.1145/3131365.3131401. URL https://doi.org/10.1145/3131365.3131401.

[3] Appsecco. the-art-of-subdomain-enumeration, 2019. URL https://github.com/appsecco/the-art-of-subdomain-enumeration.

[4] AWS. What is dns? URL https://aws.amazon.com/route53/what-is-dns/.

[5] M. Bailey, D. Dittrich, E. Kenneally, and D. Maughan. The menlo report. *IEEE Security & Privacy*, 10:71–75, 2012-03.

[6] Jacob Baines. Winbox in the wild, 2019. URL https://medium.com/tenable-techblog/winbox-in-the-wild-9a2ee4946add.

[7] D. Barr. Rfc1912: Common dns operational and configuration errors, 1996.

[8] Marcus Brinkmann, Christian Dresen, Robert Merget, Damian Poddebniak, Jens Müller, Juraj Somorovsky, Jörg Schwenk, and Sebastian Schinzel. ALPACA: Application layer protocol confusion - analyzing and mitigating cracks in TLS authentication. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 4293–4310. USENIX Association, August 2021. ISBN 978-1-939133-24-3. URL https://www.usenix.org/conference/usenixsecurity21/presentation/brinkmann.

[9] crt.sh. crt.sh. URL https://crt.sh/?q=tud-ct.fiebig.nl.

[10] CVE. Mikrotik: Security vulnerabilities, 2021. URL https://www.cvedetails.com/vulnerability-list.php?vendor_id=12508&product_id=&version_id=&page=2&hasexp=0&opdos=0&opec=0&opov=0&opcsrf=0&opgpriv=0&opsqli=0&opxss=0&opdirt=0&opmemc=0&ophttprs=0&opbyp=0&opfileinc=0&opginf=0&cvssscoremin=0&cvssscoremax=0&year=0&month=0&cweid=0&order=1&trc=62&sha=ceae065bce8f7e1c8a98de53f22abf92fae90a9f.

[11] Team Cymru. Ip to asn mapping service. URL https://team-cymru.com/community-services/ip-asn-mapping/.

[12] C. Doerr. *Network Security in Theory and Practice*. 2019.

[13] Let's Encrypt. Let's encrypt, 2021. URL https://letsencrypt.org/.

[14] Let's Encrypt. Chain of trust, 2021. URL https://letsencrypt.org/certificates/.

[15] Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. Something From Nothing (There): Collecting Global IPv6 Datasets From DNS. In Mohamed Ali Kâafar, Steve Uhlig, and Johanna Amann, editors, *Proceedings of the 18th Passive and Active Measurement (PAM)*, volume 10176 of *Lecture Notes in Computer Science (LNCS)*, pages 30–43. Springer International Publishing. ISBN 978-3-319-54328-4. doi: 10.1007/978-3-319-54328-4_3. URL http://dx.doi.org/10.1007/978-3-319-54328-4_3.

[16] Tobias Fiebig, Kevin Borgolte, Shuang Hao, Christopher Kruegel, and Giovanni Vigna. Something from nothing (there): Collecting global ipv6 datasets from dns. pages 30–43, 02 2017. ISBN 978-3-319-54327-7. doi: 10.1007/978-3-319-54328-4_3.

[17] Toby Gaff. What is the certificate chain of trust? URL https://www.keyfactor.com/blog/certificate-chain-of-trust/.

[18] Oliver Gasser, Benjamin Hof, Max Helm, Maciej Korczynski, Ralph Holz, and Georg Carle. *In Log We Trust: Revealing Poor Security Practices with Certificate Transparency Logs and Internet Measurements*, pages 173–185. 03 2018. ISBN 978-3-319-76480-1. doi: 10.1007/978-3-319-76481-8_13.

[19] Google. Certificate transparency, . URL https://chromium.googlesource.com/chromium/src/+/master/net/docs/certificate-transparency.md.

[20] Google. Statistics, . URL https://www.google.com/intl/en/ipv6/statistics.html.

[21] Josef Gustafsson, Gustaf Overier, Martin Arlitt, and Niklas Carlsson. A first look at the ct landscape: Certificate transparency logs in practice. pages 87–99, 02 2017. ISBN 978-3-319-54327-7. doi: 10.1007/978-3-319-54328-4_7.

[22] Kirk Hall. New rfc on ct domain label redaction, 2017. URL https://archive.cabforum.org/pipermail/public/2017-November/012458.html.

[23] John A. Hawkinson and Tony J. Bates. Guidelines for creation, selection, and registration of an Autonomous System (AS). RFC 1930, March 1996. URL https://www.rfc-editor.org/info/rfc1930.

[24] Bob Hinden and Dr. Steve E. Deering. Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, December 1998. URL https://www.rfc-editor.org/info/rfc2460.

[25] Paul E. Hoffman, Andrew Sullivan, and Kazunori Fujiwara. DNS Terminology. RFC 7719, December 2015. URL https://www.rfc-editor.org/info/rfc7719.

[26] Russ Housley, John Curran, Geoff Huston, and David R. Conrad. The Internet Numbers Registry System. RFC 7020, August 2013. URL https://www.rfc-editor.org/info/rfc7020.

[27] ISO 7497-1. Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model. Standard, International Organization for Standardization, Geneva, CH, 1996.

[28] ISO 9594-8. Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks. Standard, International Organization for Standardization, Geneva, CH, 2019.

[29] Auguste Kerckhoffs. La cryptographie militaire). pages 161—191, 1883.

[30] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, June 2013. URL https://www.rfc-editor.org/info/rfc6962.

[31] Cynthia Bailey Lee, Christian Roedel, and Elena Silenok. Detection and characterization of port scan attacks. 2003.

[32] Ioana Livadariu. How accurate are ip geolocation services?, 2020. URL https://blog.apnic.net/2020/09/15/how-accurate-are-ip-geolocation-services/.

[33] Matt Mackall. random.c – a strong random number generator, 2005. URL https://git.kernel.org/pub/scm/linux/kernel/git/stable/linux.git/tree/drivers/char/random.c?id=refs/tags/v3.15.6#n52.

[34] Uriel Maimon. Port Scanning without the SYN flag, TCP port Stealth Scanning., 1996.

[35] P.V. Mockapetris. Domain names - concepts and facilities. RFC 1034, November 1987. URL https://www.rfc-editor.org/info/rfc1034.

[36] Nmap. Service and version detection. URL https://nmap.org/book/man-version-detection.html.

[37] Linus Nordberg, Daniel Kahn Gillmor, and Tom Ritter. Gossiping in CT. Internet-Draft draft-ietf-trans-gossip-05, Internet Engineering Task Force, January 2018. URL https://datatracker.ietf.org/doc/html/draft-ietf-trans-gossip-05. Work in Progress.

[38] NSA. Avoid dangers of wildcard tls certificates and alpaca, 2021.

[39] Fabien Petitcolas. Kerckhoffs' principles from « la cryptographie militaire ». URL https://www.petitcolas.net/kerckhoffs/index.html.

[40] J. Postel. Internet Protocol. RFC 791, September 1981. URL https://www.rfc-editor.org/info/rfc791.

[41] J. Postel. Transmission Control Protocol. RFC 793, September 1981. URL https://www.rfc-editor.org/info/rfc793.

[42] Eric Rescorla and Tim Dierks. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, August 2008. URL https://www.rfc-editor.org/info/rfc5246.

[43] Laura M. Roberts and David Plonka. Watching the watchers: Nonce-based inverse surveillance to remotely detect monitoring, 2020.

[44] Scott Rose, Matt Larson, Dan Massey, Rob Austein, and Roy Arends. DNS Security Introduction and Requirements. RFC 4033, March 2005. URL https://www.rfc-editor.org/info/rfc4033.

[45] Peter Saint-Andre and Jeff Hodges. Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS). RFC 6125, March 2011. URL https://www.rfc-editor.org/info/rfc6125.

[46] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. The rise of certificate transparency and its implications on the internet ecosystem. In *Proceedings of the Internet Measurement Conference 2018*, IMC '18, page 343–349, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450356190. doi: 10.1145/3278532.3278562. URL https://doi.org/10.1145/3278532.3278562.

[47] Liron Segal. Mirai: The iot bot that took down krebs and launched a tbps attack on ovh, 2017. URL https://www.f5.com/labs/articles/threat-intelligence/mirai-the-iot-bot-that-took-down-krebs-and-launched-a-tbps-attack-on-\ovh-22422.

[48] Robert W. Shirey. Internet Security Glossary, Version 2. RFC 4949, August 2007. URL https://www.rfc-editor.org/info/rfc4949.

[49] H. Singh. *Implementing Cisco Networking Solutions*. Packt, 2017.

[50] Ryan Sleevi. Policy discussion: Name redaction. URL https://groups.google.com/a/chromium.org/g/ct-policy/c/vsTzv8oNcws/m/imuC3iloBwAJ.

[51] Ryan Sleevi. Certificate transparency in chrome - change to enforcement date, 2017. URL https://groups.google.com/a/chromium.org/g/ct-policy/c/sz_3W_xKBNY/m/6jq2ghJXBAAJ.

[52] Lance Spitzner. Honeytokens: The other honeypot. 2003. URL https://community.broadcom.com/symantecenterprise/communities/community-home/librarydocuments/viewdocument?DocumentKey=74450cf5-2f11-48c5-8d92-4687f5978988&CommunityKey=1ecf5f55-9545-44d6-b0f4-4e4a7f5f5e68&tab=librarydocuments.

[53] Rob Stradling and Eran Messeri. Certificate Transparency: Domain Label Redaction. Internet-Draft draft-strad-trans-redaction-01, Internet Engineering Task Force, January 2017. URL https://datatracker.ietf.org/doc/html/draft-strad-trans-redaction-01. Work in Progress.

[54] Tadahiko. Ca/ct redaction. URL https://wiki.mozilla.org/CA/CT_Redaction#Against.

[55] Certificate Transparency. Working together to detect maliciously or mistakenly issued certificates., . URL https://certificate.transparency.dev/.

[56] Certificate Transparency. How ct works: How ct fits into the wider web pki ecosystem, . URL https://certificate.transparency.dev/howctworks/.

[57] Benjamin VanderSloot, Johanna Amann, Matthew Bernhard, Zakir Durumeric, Michael Bailey, and J. Alex Halderman. Towards a complete view of the certificate ecosystem. In *Proceedings of the 2016 Internet Measurement Conference*, IMC '16, page 543–549, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450345262. doi: 10.1145/2987443.2987462. URL https://doi-org.tudelft.idm.oclc.org/10.1145/2987443.2987462.

[58] VirusTotal. Virustotal. URL https://www.virustotal.com/.

[59] Paul A. Vixie and David Dagon. Use of Bit 0x20 in DNS Labels to Improve Transaction Identity, March 2008. URL https://datatracker.ietf.org/doc/html/draft-vixie-dnsext-dns0x20-00. Work in Progress.