공학석사 학위논문

A TLS Downgrade Attack using Spatial Differences in Security Configurations

보안 설정의 공간적 차이를 이용한 TLS 다운그레이드 공격

2021 년 2 월

서울대학교 대학원

컴퓨터공학부

이 준 희

A TLS Downgrade Attack using Spatial Differences in Security Configurations

보안 설정의 공간적 차이를 이용한 TLS 다운그레이드 공격

지도교수 권 태 경

이 논문을 공학석사 학위논문으로 제출함

2020 년 12 월

서울대학교 대학원

컴퓨터공학부

이 준 희

이준희의 석사 학위논문을 인준함

2021 년 1 월

위 원 장	김 종 권	(인)
부위원장	권 태 경	(인)
위 원	전 화 숙	(인)

Abstract

A TLS Downgrade Attack using Spatial Differences in Security Configurations

Joonhee Lee
Dept. of Computer Science and Engineering
The Graduate School
Seoul National University

To provide secure content delivery, Transport Layer Security (TLS) has become a *de facto* standard over a couple of decades. However, TLS has a long history of security weaknesses and drawbacks. Thus the security of TLS has been enhanced by addressing security problems through continuous version upgrades. Meanwhile, to provide fast content delivery globally, websites need to administer many machines in globally distributed environments. They often delegate the management of machines to web hosting services or Content Delivery Networks (CDNs), where the security configurations on distributed servers may vary depending on the managing entities or locations.

By leveraging these *spatial differences* in TLS security, we present a new TLS downgrade attack, called a Teleport attack. In our attack model, an adversary collects the information of (web) domains that exhibit different TLS versions and cryptographic options depending on clients' locations. Then

the adversary redirects TLS handshake messages to weak TLS servers, and downgrades TLS sessions, which both the server and the client may not be aware of. We measure how many domains in the wild are vulnerable to the TELEPORT attack, and seek to better understand the root causes of the spatial differences in TLS security configurations. We also measure the redirection delay in various locations over the world to demonstrate the feasibility of the TELEPORT attack.

Keywords: TLS, Downgrade attack, CDN, Domain name delegation, Distributed environment

Student Number: 2019-23689

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Transport Layer Security (TLS) [1, 2] is designed for secure communications on the Internet. It is being used for most web-based services. The ratio of TLS encrypted traffic has been increasing rapidly; according to the Google's Transparency Report [3], 96 percent of web pages loaded by Chrome are encrypted with HTTPS as of July 2020. However, TLS is not a perfectly secure protocol. Since SSL 2.0 [4] was published in 1995, TLS/SSL have exposed various vulnerabilities such as weak cryptographic algorithms (*e.g.*, [5, 6, 7, 8, 9, 10]) and have been constantly attacked (*e.g.*, BEAST [11], CRIME [12], BREACH [13], Lucky Thirteen [14], Heartbleed [15], POODLE [16], FREAK [17], Logjam [18], DROWN [19], and SLOTH [20]).

TLS has been evolving and becomes more secure by solving or mitigating these security problems through continuous version upgrades [21]. The latest TLS version is 1.3 [2] that was redesigned and approved in August 2018. For example, it mandates *perfect forward secrecy* by only allowing ephemeral

keys during the ECDH key agreement. Also it extends the length of the handshake transcript against the hash collision attacks. It was reported that only approximately 31 percent of the Alexa Top 1M domains support TLS 1.3 as of November 2019 [22]. In other words, the other web servers (approximately 69%) still rely on older TLS versions, which may be subject to old-version TLS attacks. This indicates that keeping the TLS protocol up-to-date is the key challenge for providing web services securely. Considering the history of TLS security issues, threats to TLS may continue to emerge, and even cryptographic algorithms currently known to be secure may not be safe in the future (*e.g.*, when quantum computing is deployed). That is why web server operators are required to vigilantly monitor emerging vulnerabilities and continue to upgrade TLS versions and security settings.

Apart from this security aspect, the scale of web services is growing exponentially. While there are still small-scale services for local users, global services targeting numerous users around the world continue to emerge and proliferate—*e.g.*, online social networks such as Twitter, Facebook, and Instagram, or video streaming websites such as Youtube and Netflix. These services need to deliver content with low delay for geographically dispersed users, which may not be achieved simply by operating large data centers. Thus, domain owners typically rely on third parties for optimized delivery, such as Content Delivery Networks (CDNs). CDNs deploy geographically scattered edge servers all over the world to deliver web content to worldwide users on behalf of the domain owners. In this case, the domain owner must delegate to a third party the commitment to keep the end-to-end security for content delivery up-to-date. The delegation of these administrative

responsibilities may provide a source of threat to Web security.

In this paper, we propose a new attack, named the TELEPORT attack that can trick web clients into handshaking with vulnerable servers with lower TLS versions (say, weakened ciphers) if the domains that the clients attempt to access are served from multiple TLS servers with different TLS security settings. Specifically, an adversary can successfully conduct this attack by *teleporting* (or redirecting) the clients' connection requests to vulnerable TLS servers even if the clients are supposed to establish connections with TLS servers with secure TLS parameters.

The TELEPORT attack is possible by exploiting *spatial differences* in TLS server settings. We examine more than 7M domains from multiple vantage points across different continents, and find that 28,956 domains in the wild are vulnerable to the attack. Clients who access the vulnerable domains can be exposed to such attacks since the clients can be diverted to vulnerable TLS servers under different cryptographic conditions. In particular, it is possible to make the clients to establish downgraded TLS connections. Finally, the adversary is now able to conduct a man-in-the-middle (MitM) attack against the vulnerable TLS sessions between the client and the vulnerable TLS server (to which the clients are diverted) using the known TLS attacks such as POODLE [16], FREAK [17], Logjam [18], DROWN [19].

Many studies have been conducted on the TLS downgrade attacks. For example, POODLE [16] relies on the fallback mechanism of a web browser. FREAK [17], Logjam [18] and DROWN [19] exploit weak points in the TLS protocol, and SLOTH [20] is an attack leveraging hash collisions. However, none of the prior studies have focused on inconsistency in configurations of

distributed web servers. To the best of our knowledge, the TELEPORT attack is the first one that leverages the spatial differences in the security settings on the server-side.

We also propose a method to identify how the web servers of the vulnerable domains are managed to better understand the root causes of such spatial differences. In addition, we measure how much the TLS handshake is lengthened if TELEPORT attacks are carried out across several continents, demonstrating that the overhead latency of the TELEPORT attack is hardly substantial. This indicates that victims may not recognize being attacked.

In summary, our contributions are as follows:

(1) Introducing a new type of attack, the TELEPORT attack (§3): We design a TLS downgrade attack called the TELEPORT attack that leverages the *spatial differences* in TLS security settings, due to the faulty or outdated management of server-side TLS configurations. We demonstrate (i) how to gather information (*e.g.*, IP addresses, TLS versions, ciphersuites, etc.) of web servers for spatial downgrade attacks, and (ii) how an adversary can downgrade a TLS session and obtain sensitive information of users.

(2) Analysis on the effect of the TELEPORT attack (§4): We analyze the effect of the TELEPORT attack on the real-world domains. To understand the reasons of the vulnerable domains, we provide an approach to identifying how web servers are managed. From the analysis, we find that about 25% of the vulnerable domains rely on CDNs.

(3) Evaluation of the TELEPORT attack (§5): To show the feasibility of the TELEPORT attack, we build the TELEPORT infrastructure on AWS cloud servers and measure the overhead (*i.e.*, network latency) incurred by

the TELEPORT attack. Interestingly, the session establishment time with the vulnerable domains is within 1.3 seconds at most; thus, the attack can be performed to these domains without being noticed by users.

In the rest of the paper, we first present the background in §2, followed by description of the TELEPORT attack in §3. Next, we analyze the effect of the TELEPORT attack in the real-world in §4 and evaluate the latency of the TELEPORT attack from a user's perspective in §5. Then, we discuss further issues in §6 and related work in §7, respectively. We finalize this paper with concluding remarks in §8.

Chapter 2

Background

2.1 TLS Handshakes and Downgrade Attacks

Agreement on the version and ciphersuite. Transport Layer Security (TLS) [1, 2] is a protocol designed to provide secure communications between two entities (e.g., a web server and its client) over the untrusted Internet. In a TLS handshake, a server and a client negotiate parameters (TLS version, ciphersuites, etc.) for cryptographic communications, and the server is authenticated (the client authentication is optional). The TLS handshake makes an agreement between the two entities on a TLS protocol version and a ciphersuite. The TLS protocol version should be selected between 1.0 and 1.3 under mutual agreement. The TLS ciphersuite is a combination of asymmetric cipher and key types, key exchange algorithms, symmetric ciphers and key, hash algorithms, etc. The client sends the highest TLS version she can support and a list of possible ciphers in the `ClientHello` message. The server

then selects the highest TLS version and the ciphersuite that both can support. Such selections are specified in the `ServerHello` handshake message. Subsequently, the other handshake messages are exchanged according to the agreed TLS version, and key materials are also exchanged by the determined ciphersuite, and finally the TLS session is initiated.

Downgrade attack. The biggest drawback in this bilateral protocol agreement is that the higher security level on one side has to be degraded to meet the lower one on the other side. For example, if a client provides only vulnerable hash algorithms, such as MD5 [23, 9], in the `ClientHello` message, the server has no choice but to choose the vulnerable one. This results in lower security level between the client and the server, and the attacker can exploit vulnerabilities in weak cryptographic algorithms or the older TLS version (say, TLS 1.0, 1.1, or 1.2), which is called a *downgrade attack*. Such attacks, which force two endpoints to use an insecure channel, are still one of the most threatening methods of attacks against TLS. There are many well-known TLS downgrade attacks including POODLE [16], FREAK [17], Logjam [18], DROWN [19] and SLOTH [20].

On the contrary, even though the client uses the the latest web browser that fully supports the latest TLS protocol (say TLS 1.3) and strong ciphersuite, the client can still be vulnerable against the downgrade attacks if the server supports only older TLS version and weaker ciphersuite. The TLS connection between the client and the server is established with insecure parameters and hence the connection becomes vulnerable. Accordingly, the adversary may be able to perform a man-in-the-middle (MitM) attack by exploiting the vulnerability.

Although there have been many reports of vulnerabilities of protocol designs or client implementations that enable such downgrade attacks, we present a new downgrade attack that takes advantage of vulnerabilities of *server-side configurations in distributed environments* that has not been reported.

2.2 CDN Redirection

A Content Delivery Network (CDN) is a geographically distributed network of edge servers, which delivers content to a client on behalf of an origin server. CDNs help reduce content delivery delays since the client retrieves the content from a physically close edge server. Specifically, a CDN replicates the content of the origin server to a network of edge servers geographically scattered at different locations. When the client attempts to access the origin server, the CDN redirects the request to an edge server located physically closer to the client in two ways: (1) DNS-based mapping and (2) anycast routing.

(1) DNS-based mapping is a method of assigning a group of edge servers to the clients based on the location of the client's local DNS resolver. The domain owner (i.e., the owner of the origin server) can delegate its name resolution to a CDN service provider via CNAME or NS records. Then the CDN's DNS server responds to a client with an appropriate A record considering the client's location. Note that the CDN's DNS server is informed of the client's location using the EDNS Client Subnet [24] field to enhance the location proximity of the client's local DNS resolver.

(2) Anycasting is a network routing mechanism in which a single IP address is mapped to multiple endpoints (i.e., multiple edge servers in the CDN

context). For example, `Cloudflare`, a CDN service provider, leverages this anycast routing to redirect HTTP requests from users around the world. When a user sends a packet whose destination IP address corresponds to a domain name managed by Cloudflare, the IP routing delivers the packet to a topologically-nearest edge server of the anycast group by the Border Gateway Protocol (BGP).

Many global web services are using CDNs, and we argue that it is necessary to understand the redirection mechanism of CDNs in order to better understand the root causes of the spatial differences of TLS security settings we have discovered. For this, we trace how HTTPS requests for vulnerable domains are redirected and why our attack is effective.

Chapter 3

Teleport Attack

In this chapter, we introduce the Teleport attack that downgrades the TLS version or the ciphersuite by exploiting the spatial differences in the TLS configuration settings across distributed CDN edge servers.

3.1 Threat Model

The adversary's goal is to obtain the sensitive information of users (*e.g.*, passwords or web cookies) heading to the target web server. The capability that the adversary needs to achieve the goal is having distributed proxies under his control running around the world.

More specifically, the adversary can intercept the packets between clients and web servers, and break confidentiality using the known TLS downgrade attacks described in §7. To this end, the adversary first compromises or deploys a local gateway such as an WiFi Access Point (AP) to eavesdrop and

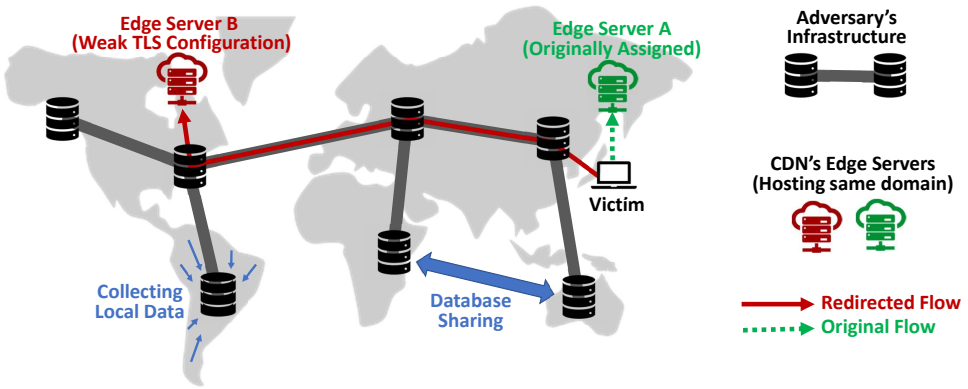

Figure 3.1: **The TELEPORT infrastructure.** An adversary aims to obtain sensitive information toward the target web servers located in several regions. We assume that the adversary runs an infrastructure that consists of proxies throughout the world. In the TELEPORT attack, each proxy in the infrastructure has the two main roles: (i) collecting information about the target web servers (*e.g.*, TLS security parameters) and (ii) redirecting TLS messages to the particular region.

to tamper with packets between the clients and servers. For example, the adversary can install a fake WiFi access point with an attracted SSID (*e.g.*, Free WiFi) in public (*e.g.*, at a coffee shop) to eavesdropping packets from arbitrary users. In sessions with end-to-end encryption with TLS, the adversary can only tamper with unencrypted fields such as IP headers. We also assume that the adversary runs proxies throughout the world. For instance, the adversary can purchase AWS instances from several regions or manage his machines to build his own infrastructure as illustrated in Figure 3.1. Over the TELEPORT infrastructure, the adversary can arbitrarily forward packets to proxies under his control. We assume that the adversary *cannot* compromise endpoints such as web browsers and web servers; thus, the TLS protocol is correctly performed.

With the above capabilities, the adversary can perform the TELEPORT attack, which is illustrated in Figure 3.2, 3.3 and 3.4. The TELEPORT attack is conducted in three steps. As illustrated in Figure 3.2, the adversary first gathers the list of potential target domains—*e.g.*, from Alexa 1M web domains—whose content is delivered from multiple edge servers if they rely on CDN services. Note that some of edge servers may have potential TLS vulnerabilities (see §3.2). Then, as shown in Figure 3.3, the adversary intercepts the TLS handshake messages, which are redirected to vulnerable edge servers through the TELEPORT infrastructure (see §3.3). Finally, the adversary uses any known attack mechanisms against a downgraded session to steal sensitive information as illustrated in Figure 3.4 (see §3.4).

3.2 Populating Target Database

As a preliminary step of the attack, the adversary should find out domains and their corresponding edge servers with weak TLS settings that allow him (i) to redirect victims to the vulnerable edge servers located in some regions and (ii) to conduct MitM attacks against the victims.

Target domains & IP addresses. An adversary can target any clients and perform the TELEPORT attack. For an unspecified target, the attacker periodically scans some of the domain namespace (*e.g.*, Alexa 1M domains) to build a list of vulnerable domains. If a particular client is targeted, the database can be populated by continuously scanning only the websites that she frequently visits.

Once the target domains are determined, the adversary collects the IP addresses belonging to the domains. A domain may have multiple corresponding

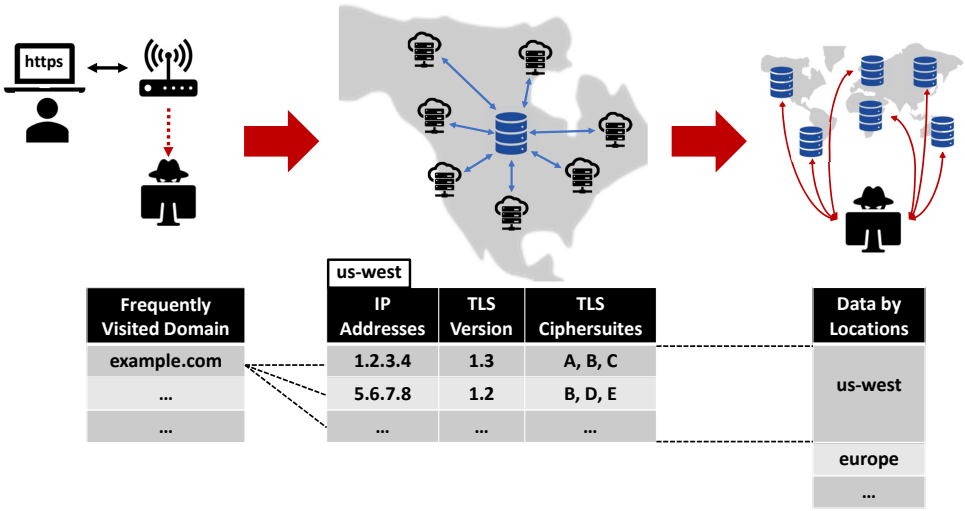

Figure 3.2: **Collecting the information of vulnerable TLS servers (*i.e.*, populating a Target Database)**: First, the adversary determines a target domain list. Second, the adversary queries DNS servers and logs the TLS versions and ciphersuites in each region. Finally, the adversary builds a merged database by combining the log data from different locations.

IP addresses. For example, a global website may have multiple IP addresses since it relies on multiple servers for load balancing or service stability, or delegates fast content delivery to CDNs. Even when multiple clients access the same website, they may connect to (machines with) different IP addresses depending on their locations. Therefore, the adversary should query multiple DNS servers at multiple vantage points—*e.g.*, across different continents, and log the target domains and their IP addresses into the Target Database as illustrated in Figure 3.2.

TLS versions and ciphersuites. After obtaining the target domains and their IP addresses, the adversary now finds vulnerable edge servers with older TLS versions and weak cryptographic algorithms that can be exploited to

conduct a MitM attack against target victims. To investigate the TLS configurations, the adversary first sends `ClientHello` messages to the IP addresses of the target domains, and receives `ServerHello` messages. This process allows the attacker to figure out the TLS version and the ciphersuite supported by each edge server. Note that the adversary's goal is to find vulnerable TLS servers around the world for the same domain name. If the adversary can find older TLS versions and weak cryptographic algorithms from `ServerHello` messages from target domains, he records the versions and algorithms of the TLS servers to the log.

3.3 TLS Handshake Redirection

Recall that the adversary cannot decrypt the ciphertext part of the TLS messages. Therefore, he can see only the unprotected TCP/IP headers. The adversary can obtain the IP addresses and domains from TLS handshake messages that the victims send as follows. When a client accesses an HTTPS website, a TCP connection is first established and then TLS handshake messages are exchanged between the client and the web server. The attacker can obtain the destination IP address from the message headers, and also learn the domain name of the website by looking at the Server Name Indication (SNI) [25] TLS extension in the `ClientHello` message. The SNI extension is a technique that specifies the domain name to which the TLS client wants to connect, which is necessary if the server machine hosts multiple web servers with their TLS certificates in its IP address.

The attacker looks for the IP addresses and/or domains in the Target Database to see if a website with which the victim communicates has any

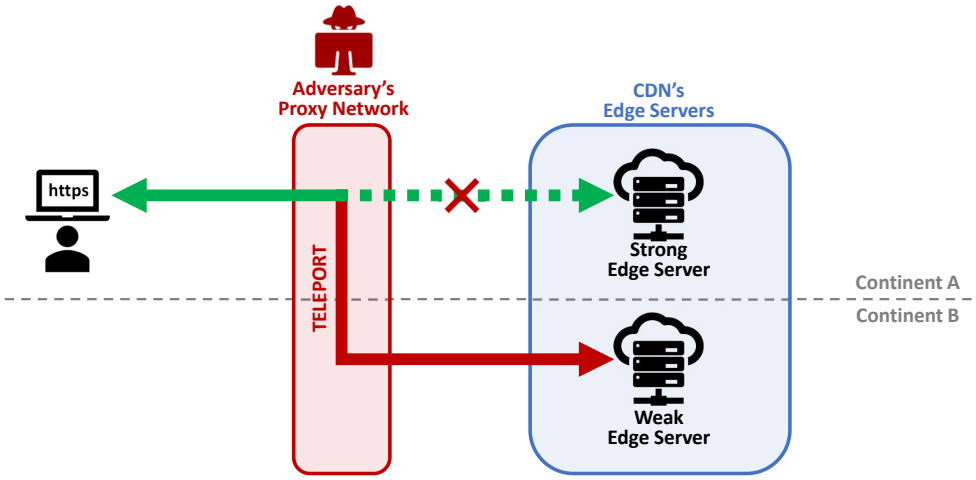

Figure 3.3: **Handshake redirection:** The adversary can *teleport* a client's request to a vulnerable TLS server possibly located in another continent (*e.g.*, downgraded TLS version or weakened ciphersuite) through the TELEPORT infrastructure.

vulnerable edge servers to the TELEPORT attack. If found, it means that the victim can be redirected to the vulnerable TLS server with the same domain name instead of the one the victim is supposed to set up a TLS connection. The victim may be unaware of being redirected to another IP address (*i.e.*, for the same web server) since a legitimate website is connected, its certificate is valid, and its webpages are displayed.

For example, suppose that a client in France accesses `example.com`, which relies on the CDN service. Hence its content is replicated to edge servers of the CDN service provider across the world. The client is supposed to access to one of the edge servers located physically closer to her location in France. The adversary learns that one of the web servers located in the US runs with an older TLS version and weak cryptographic algorithms, which is vulnerable to known TLS attacks (*e.g.*, POODLE [16] and Lucky Thirteen [14]). Then,

the adversary can manipulate the IP header or perform DNS spoofing to redirect the client's access request to the vulnerable server. DNS spoofing or DNS cache poisoning [26] makes clients to connect to a malicious machine (for a given domain) by taking advantage of the fact that DNS records are cached by a local DNS resolver. Since the adversary knows the vulnerable domains and its IP addresses in advance, he can redirect users by spoofing the cached DNS records.

The client cannot recognize that he/she is redirected to another server since the normal web pages of `example.com` is shown, which is expected. The TLS connection between the client and the insecure TLS server thus becomes vulnerable to the known TLS attacks. In this way, the adversary successfully performs the Teleport attack and obtains the sensitive information of the client (*i.e.*, victim).

These simple redirection attacks may go unnoticed if victims cannot readily notice additional network delays. We evaluate in §5 how much network delays occur due to the Teleport attack, and whether the attack is noticeable by the user. Furthermore, it is not necessary to redirect all the TLS handshake messages. For example, suppose the purpose of the adversary is to leak a login password from the user who accesses `example.com`, which operates a login-purpose subdomain named `login.example.com`. Then, the attacker needs to redirect handshake messages only for the domain name being `login.example.com`, and the other handshake messages need not to be redirected, which can significantly reduce the overall network latency.

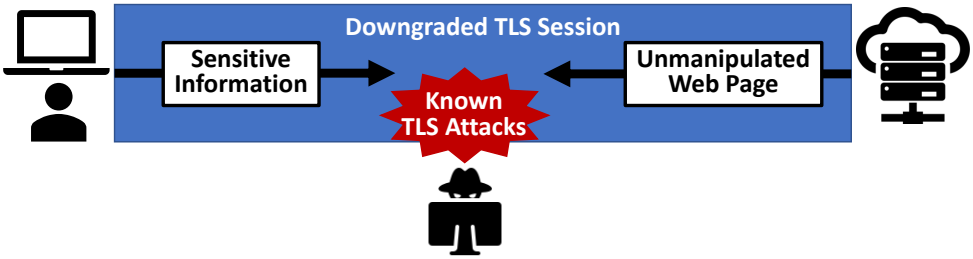

Figure 3.4: **Downgraded session exploitation.** The adversary can successfully conduct the TELEPORT attack against the downgraded or weakened TLS sessions and steal sensitive information.

3.4 Downgraded Session Exploitation

After making the victim perform handshake in a downgraded vulnerable TLS version or ciphersuite, the known attacks mentioned in §7 can be applied to the encrypted session. For example, if a TLS session is downgraded to TLS 1.0, the adversary can conduct the POODLE or BEAST attack. If the adversary deceives the server and the client to agree with the ciphersuite including vulnerable hash algorithms such as SHA-1 or MD5, he can use the SLOTH attack. TLS 1.2 has been recognized as relatively safe until recently, but several problems exist including the forward secrecy issue. In addition, new attack techniques targeting TLS 1.2 (e.g., Raccoon Attack [27]) can emerge anytime.

Different edge servers may have different ciphersuites even if they support the same TLS version. Known attacks can be executed if a deprecated or vulnerable crypto algorithm remains. For example, if the TLS session is downgraded to 1.2 and the RSA key exchange algorithm can be used, forward secrecy is no longer guaranteed.

3.5 Summary

To summarize, an adversary can (i) populate the IP addresses of vulnerable TLS servers (of targeted domains) and their TLS security settings, and (ii) redirect clients' TLS handshake messages to the vulnerable servers, then (iii) obtain the users' sensitive information by conducting known TLS attacks on downgraded sessions. In the next chapter, we will investigate the effect in the real world, especially regarding the steps (i) and(iii). Then, we will evaluate how this redirection affects user experiences.

Chapter 4

Effect of the Teleport attack

In this chapter, we analyze the effect of the Teleport attack in reality. To this end, we perform the first step of the Teleport attack (§3.2) and find vulnerable domains in the wild. Then, we study how web servers are managed, to understand why the spatial differences exist.

4.1 Data Collection

We collect three types of datasets as summarized in Table 4.1: target domains, ClientHello/ServerHello messages, and additional DNS records.

Target domains. We need a list of domains in the wild to measure how many domains and their vulnerable TLS servers exist. We first make the list of the target domains by extracting CommonNames from their TLS certificates collected by `Rapid7` [28] on June 16th, 2020. For wildcard names, we remove asterisks from names, which results in apex domains. Then we

Dataset	Description	Source	# of Records
Target domains	Domain names	Rapid7 [28]	7,032,829
Hello messages	TLS versions Security parameters IP addresses	Our TLS 1.3 client	42,196,974
DNS records	IP-domain mappings	ActiveDNS [29]	782,446,164

Table 4.1: **Summary of our datasets:** We used Rapid7 to analyze our target domains, which results in 7M target domains. To obtain the TLS version and security parameters of the 7M target domains, we implement our own TLS client, which sends/receives ClientHello/ServerHello messages to/from the domains. We also use ActiveDNS (782M DNS records) to find more vulnerable TLS servers in the wild.

perform nslookup on the apex domains. If nslookup returns NXDOMAIN, we prepend www to the domain name, which is added to the list. We do not add CommonName to the list if they are IP addresses or irrelevant FQDNs. After performing these steps, we obtain 7,032,829 domains.

ClientHello/ServerHello messages. Recall that the TELEPORT attack requires the information of the TLS versions and ciphersuites of TLS servers. To obtain such information, we implement a TLS 1.3 client application based on OpenSSL-1.1.1g, which sends ClientHello, receives ServerHello, and terminates a connection. These messages are recorded with the server's IP address. We access the 7M domains with our client applications located at six vantage points: six cities in five continents—Asia (India, South Korea), Europe (France), North America (US West), Oceania (Australia), and South America (Brazil).

Additional DNS records. The collection of 7M domains does not cover the entire web, and thus we may need to extend our coverage of domains. Considering that a general edge server may host multiple domains in a single machine or IP address, it is worth revisiting the IP addresses of vulnerable domains that we already discovered. To find more vulnerable domains, we use `ActiveDNS` [29], an active DNS probing project, that actively queries DNS servers for domains in the wild and collects DNS records such as IP addresses and domain names. We obtain total 782,446,164 DNS records on July 17th, 2020 from `ActiveDNS`. After finding out the IP addresses of the TLS servers vulnerable to the Teleport attack from the first dataset (7M domains), we find 272,406 additional domains, which are potentially vulnerable, mapped to those vulnerable IP addresses on the DNS records.

4.2 Vulnerable Domains

As specified in §4.1, we connect the 7M domains from the six vantage points to find vulnerable domains to the Teleport attack. We claim that a domain is vulnerable if either of the two following conditions is satisfied:

- **Downgraded TLS Version:** A TLS session from at least one vantage point is established with a *lower* TLS version, compared with other vantage points. For example, some points support the latest version TLS 1.3, while at least one of the other points supports only the vulnerable TLS 1.0.

- **Weakened Ciphersuite:** A TLS session from at least one vantage point is established with a *non-forward secret*, a *deprecated*, or a *shorter*

Downgraded TLS Version	# of Domains
TLS1.3 → TLS1.2	17,740 (80.98%)
TLS1.3 → TLS1.1	7 (0.03%)
TLS1.3 → TLS1.0	452 (2.06%)
TLS1.2 → TLS1.1	73 (0.33%)
TLS1.2 → TLS1.0	3,648 (16.65%)
TLS1.1 → TLS1.0	15 (0.07%)
Total	**21,907 (100%)**

Table 4.2: **Downgraded TLS Version**. For example, TLS1.3 → TLS1.0 indicates that among TLS servers for a domain, one server supports the highest TLS version 1.3 while another supports the lowest version 1.0. That means that the domain is vulnerable to the TELEPORT attack since clients can be redirected to the server with vulnerable TLS 1.0.

key algorithm, compared with other vantage points that support strong ciphersuites.

In this regard, we find that 28,956 domains are vulnerable to the TELEPORT attack from ClientHello/ServerHello messages. The breakdown of the results is shown in Table 4.2 and 4.3, which leads to the following observations:

First, 21,907 domains have some servers with downgraded TLS versions. Given the size of the entire dataset, it may not be a threatening number. However, as Figure 4.1 shows, the more popular domains are the more likely vulnerable to the TELEPORT attack. This means that global scale web services are forced to operate more servers in more scattered areas for quality of service, resulting in more spatial differences. On the other hand, surprisingly, TLS 1.0 servers (4,115, 18.78%) are still being in use. Most of domains have

Weakened Ciphersuite	# of Domains
Forward secret → Non-forward secret (*e.g.*, ECDHE/DHE → RSA)	1,618 (9.95%)
Non-deprecated → Deprecated (*e.g.*, SHA256/SHA384 → SHA1)	4,735 (29.11%)
Larger key → Shorter key (*e.g.*, AES 256bit → AES 128bit)	12,107 (74.43%)
Total	**16,267 (100%)**

Table 4.3: **Weakened Ciphersuite.** Ciphersuites can also be weakened. For example, a domain has multiple web servers located around the world. One server supports forward secrecy, while another does not. In this case, clients who access the domain are subject to the Teleport attack.

their TLS servers (17,740, 80.98%) ranging from TLS 1.3 to TLS 1.2. The wide usage of old TLS versions highlights the vulnerability to the attack. That is, by conducting the Teleport attack, an adversary can perform known attacks to those "downgradable" domains.

Second, an adversary can weaken the security of the TLS sessions. We find that an adversary can make clients have TLS sessions without forward secrecy for 1,618 domains by changing the (EC)DHE-related ciphersuite to the RSA-related ciphersuite. Furthermore, 4,735 domains have some TLS servers that can be weakened from the SHA256/SHA384-related ciphersuite to the SHA1-related ciphersuite. An adversary can also decrease the key size from 256 bits to 128 bits used for the AES algorithm for 12,107 domains.

Third, we observe that some of vulnerable domains may allow attackers to obtain sensitive user data. For example, 91 domain names include login and 282 domain names contain auth or account or sso. Furthermore, 6,429 domain names start with mail or webmail or email, and 398 domain names

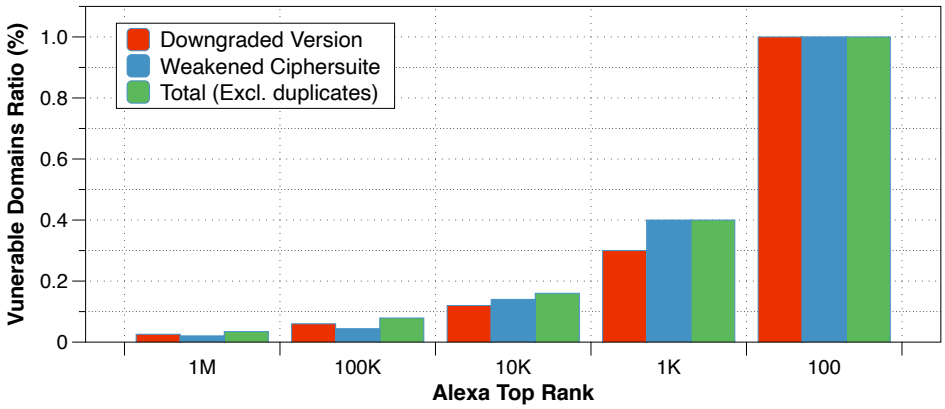

Figure 4.1: **Vulnerable domains ratio.** The ratio is shown higher as the ranks of Alexa top sites go higher.

start with admin. It implies that many of the vulnerable domains we have found are dealing with sensitive information.

Lastly, our collected 7M domains cannot cover the entire Web. To find more vulnerable domains that we may miss from the 7M domain dataset, we use `ActiveDNS` that provides 782M domains and its associated IP addresses in the wild. We look for other vulnerable domains whose servers mapped to the same IP addresses of the already-found-to-be-vulnerable TLS servers, which results in additional 272,406 domains (out of the total 419,559 pairs of IP addresses and domains) that are not found from the 7M domain dataset. The breakdown of the domains are shown in Table 4.4. Interestingly, 114,003 domains (41.8% out of 272,406) are pointed to more than one IP addresses, which indicates that these domains have TLS servers wotj vulnerable settings. Thus, the attackers can have a wider option of choosing vulnerable TLS servers when redirecting clients. This may help reduce the network latency to perform the Teleport attack.

Downgradable TLS Version			Weak Ciphersuite			Total
TLS 1.0	**TLS 1.1**	**TLS 1.2**	**TLS_RSA**	**SHA1**	**AES_128**	
2,948	7	340,620	549	837	74,598	419,559

Table 4.4: **More domains vulnerable to the Teleport attack.** We present the breakdown of vulnerable domains in the wild from the 782M domain dataset.

4.3 Cases of Spatial Differences

To understand why such a spatial disparity exists, we further investigate the vulnerable domains. There are cases where the different TLS versions or ciphersuites for the same domain are found from different vantage points. Specifically, the different IP addresses of the weak TLS servers are found across the vantage points and also across different CDN networks. For example, the IP addresses of `www.aliceblue*****.com` are 104.18.15.** from Paris, France and 148.72.249.** from the rest.[1] Note that the former is served by `Cloudflare` using TLS 1.3, but the latter is served with TLS 1.2 from `GoDaddy`. That is, only the clients located around Paris can securely communicate with the server (if there is no Teleport attacks) while others may be vulnerable to other TLS attacks.

As another example, `sl***.com` is the only domain listed in Alexa top 100 sites, which can be downgraded from TLS 1.3 to 1.2 and from AES256 to AES128. The domain is using `Amazon Web Service` (`AWS`) `EC2` and is mapped to different IP addresses each time by `Amazon` name servers. Some of them supports TLS 1.3, while the others did not.

In some of the Alexa top 100K domains, TLS 1.3 can be downgraded

[1]The domain and its IP addresses are masked for privacy.

to TLS 1.0. The `toc***.net` domain is mapped to several anycast IP addresses owned by `Cloudflare`; however sessions in some areas are established with TLS 1.0. The `tuy***.com` domain, hosted by a non-anycast single IP address in all regions, exhibits different TLS versions; we can infer that differently configured multiple servers are using a single public IP address.

From our observations, we can conclude that the spatial difference is mainly due to the different TLS security settings in the multiple servers located around the world that are primarily relying on multiple CDNs [30], or web hosting service providers, or multiple servers in the cloud services. Therefore, we need to identify how the web server of each domain is managed, which helps better understand the root causes of the spatial differences and ultimately present mitigations against the attack.

4.4 How Web Servers are Managed

We first need to know how web servers are managed and operated (or which content delivery platform is used) for each vulnerable domain to better understand why the TLS versions and ciphersuites are differently configured depending on the clients' locations. A simple approach of web server provisioning is that the domain owner operates and manages his web servers directly. In this case, keeping the web servers secure and up to date is the responsibility of the domain owner. Another option is to delegate content delivery to a third-party like CDNs or web hosting service providers. In this case, the administration of web servers such as configuring TLS parameters or updating TLS versions is performed by the third party. Meanwhile the domain owner only needs to manage web content.

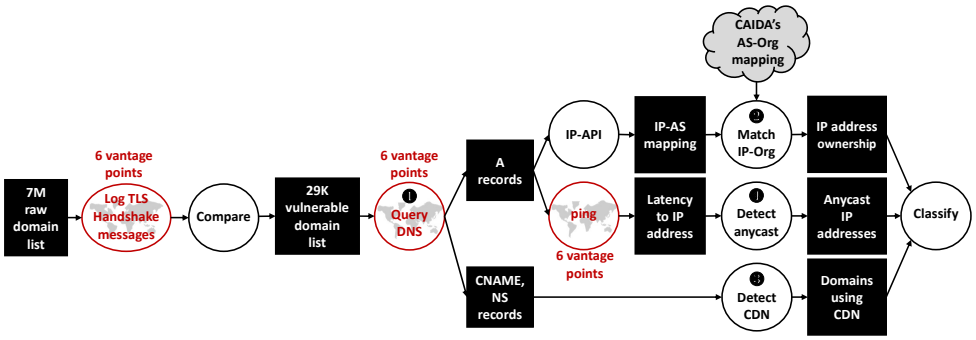

Figure 4.2: **Identification process of the web server management.** To better understand the root causes of spatial differences in TLS versions or ciphersuites, we devise a method to categorize the web hosting service providers (including CDNs) that exhibit the spatial differences. First, we find the vulnerable domains by establishing TLS handshakes with the 7M target domains from the six vantage points. Then, we query DNS servers for the vulnerable domains and obtain A, CNAME, and NS records. With the three types of the DNS records, we identify the ownership of IP addresses and check whether anycasting or CDN is used.

We present a method to identify how web servers are managed. First, ❶ we obtain the IP addresses of domains by querying DNS servers from the six vantage points across the five continents; we also collect CNAME, NS, and A records as well. A records are used for ❷ IP address owner identification and ❹ anycast IP address identification. Moreover, CNAME and NS records can used for ❸ CDN identification. Figure 4.2 illustrates our entire process to identify how the web servers are managed.

❶ **DNS query by region.** When a domain name is given, the basic approach to obtain its associated IP addresses is to use DNS servers. The DNS servers return CNAME, NS, and A records as a response to the query. A CNAME (Canonical Name or Alias) record is an alias of another canonical

domain name, which helps redirect to another domain. A NS record specifies an authoritative name server for a domain. Typically, CDN service providers use the two types of records (CNAME and NS) to redirect clients' requests to their edge servers located nearby the clients. With an A record, we can infer who owns and manages an IP address. Moreover, we can also identify the anycast CDN services such as `CloudFlare`[2].

We query DNS servers for the 29K vulnerable domains from the six vantage points, and obtain CNAME, NS, and A records of the domains. The collected DNS records are used to identify ❷ the ownership of IP address and CDN services (❸ DNS-based mapping CDN services and ❹ anycast CDN services).

❷ IP address owner identification. A simple approach to identify who owns and manages the IP addresses of the 29K vulnerable domains is to use the Autonomous System (AS) information. An AS is a collection of IP routing prefixes under the control of a single administrative entity. Typically, all IP addresses belonging to a single AS are managed by a single organization; however, organizations that operate large or complicated internet infrastructures may have multiple ASes in different IP address ranges.

This information of IP address ownership can be used to infer whether different IP addresses are under the control of the same organization. We use the IP geolocation API service, called IP-API[3] that provides not only the geolocation information, but also the AS information. Then, we use CAIDA's Inferred AS-to-Organization Mapping Dataset [31] that maps between organizations and ASes; in total, it has 95,806 ASes mapped to 78,309 organiza-

[2]https://www.cloudflare.com/
[3]https://ip-api.com/

	US West	Oceania
Domain	www.hermes.com	www.hermes.com
CNAME chain	CNAME → 2-01-272f-0080.cdx.cedexis.net. → CNAME → cs1001.wpc.phicdn.net. → A	CNAME → 2-01-272f-0080.cdx.cedexis.net. → CNAME → www.hermes.com.wtxcdn.com. → A
A record	192.229.211.218	163.171.197.13, 163.171.208.212

Figure 4.3: **Examples of multi-CDN detection.** There are two or more different CDN service providers for a CNAME chain of the domain `www.hermes.com`. Specifically, this domain is served from two CDNs: Verizon CDN from the US west and Alibaba Cloud Computing CDN from Oceania.

tions. With the two datasets, we can infer the ownership of the IP addresses. For 25,777 IP addresses from the DNS A records of 28,956 vulnerable domain names, we observe that they belong to 5,427 organizations.

❸ **CDN detection.** As explained in §2.2, recall that websites rely on CDNs for various reasons and two methods are mainly used for CDNs: (i) DNS-based mapping and (ii) anycasting. In the case of DNS-based mapping CDNs, they can be readily identified if the vulnerable domains redirect clients' requests to CDNs using the CNAME or NS records from the DNS queries, which helps us check whether CDN edge servers can be the one of the root causes of the vulnerability. Detecting anycasting will be explained in ❹.

Typically, a website uses only a single CDN service provider. However, a single website may rely on multiple CDN service providers. Such cases are

called multi-CDNs if more than one CDN service provider are found from the chains of NS records, as illustrated in Figure 4.3. Obviously, in multi-CDN cases, A records point to different IP addresses.

We first remove the subdomains from CNAME and NS records of the 29K vulnerable domains, and check whether they match with the domain names of known CDN service providers. Among the vulnerable domains, 1,731 domains rely on CDN services, and 358 (out of 1,731) domains are using multi-CDNs.

❹ **Anycasting detection.** Another method to redirect clients to CDN edge servers is anycasting. It is challenging to figure out whether anycasting is used for a website when we look at only its DNS records. A records may offer a hint; that is, a CDN employing anycasting may be used if all the A records have the same IP address. However, it does not guarantee that a single IP address from A records means anycasting since it is possible that a single machine hosts the website with its single IP address.

To be more conservative, we devise a simple algorithm to detect anycasting by observing the network latency. For example, if the network delays from the six vantage points exhibit noticeable variations, the IP address in the A record would be a single machine or a cluster of machines with the same virtual IP address. This is because the physical distances from the vantage points to the machine or cluster will vary substantially. However, in the case of anycasting, at any two vantage points ($p1$, $p2$), the sum of two network delays between the server machine (s) and each of the vantage points ($l_{p1\to s}$ and $l_{p2\to s}$) should be smaller than the network delay between the two vantage points ($l_{p1\to p2}$), whose expression is given as follows.

$$l_{p1\to s} + l_{p2\to s} < l_{p1\to p2}$$

We randomly choose any two vantage points ($p1$, $p2$) physically far from each other. We measure the network delays between the two vantage points ($l_{p1 \to p2}$) by sending an ICMP [32] Echo message. Then, we also measure the network delays between each vantage point and the server machine (s) of the vulnerable domain ($l_{p1 \to s}$ and $l_{p2 \to s}$). If the sum of the network delays from the two vantage points to the web server's IP address ($l_{p1 \to s} + l_{p2 \to s}$) is smaller than the latency between the two vantage points ($l_{p1 \to p2}$), this IP address is deemed an anycasting IP address. To detect anycasting rigorously, we check the above delay comparison for every pair among the six vantage points. For each comparison, we average the three measurements. We understand that this method might have some false positives or false negatives due to the routing instability and queuing variations. However, we believe that it would be sufficient for our purpose of detecting anycasting IP addresses.

4.5 Classification Results

The results of classifying 28,956 vulnerable domains are shown in Table 4.5. First of all, more than 94% of vulnerable domains are hosted by non-CDN operators, among which more than half are mapped to non-anycasting single IP addresses. That is, in the case of a non-anycasting single IP address, there could be multiple physical machines behind a NAT or a cluster of machines wit the same virtual address. And we find that many of the first dataset of 7M domains are uncommon domains possibly created for temporary or special purposes (*e.g.*, fcc27000000c14f100cf******.kee****.io[4]) rather than general web services, and the biases in the classification results may result

[4]Its domain name is masked for privacy.

Web Hosting Provider	Type	Subtype	# of Domains	
			7M dataset	1M dataset
CDN	DNS-based		220 (0.76%)	56 (5.18%)
	Anycast	Single IP	416 (1.44%)	40 (3.70%)
		Multiple IPs	655 (2.26%)	66 (6.10%)
	Multi-CDN		315 (1.09%)	116 (10.72%)
	Subtotal		**1,606 (5.55%)**	**278 (25.69%)**
Non-CDN	Single IP	Non-anycast	17,926 (61.91%)	409 (37.80%)
		Anycast	4,410 (15.23%)	37 (3.42%)
	Multiple IPs Same org.	Same IP by region	2,218 (7.66%)	157 (14.51%)
		Different IPs by region	991 (3.42%)	70 (6.47%)
	Multiple IPs Different orgs.	Same IP by region	752 (2.60%)	28 (2.59%)
		Different IPs by region	942 (3.25%)	102 (9.43%)
	Subtotal		**27,239 (94.07%)**	**803 (74.21%)**
Unidentifiable			111 (0.38%)	1 (0.09%)
Total			**28,956**	**1,082**

Table 4.5: **Classification of domains vulnerable to the Teleport attack.** This table summarizes the classification results for Rapid7 7M and Alexa 1M datasets, respectively. Based on this table, we analyze the root causes of the vulnerability and find that a considerable number of vulnerable domains (25.69% in Alexa 1M dataset) employ CDNs.

from such cases. Each of those domains is hosted on a single IP address since they are not web services for many clients. Since the attack of our interest assumes distributed servers across various regions, we now focus on popular domains in this section.

We further narrow down the target dataset to the Alexa top 1M domains[5].

[5]https://www.alexa.com/topsites

We find 1,082 vulnerable domains. 25.69% of the vulnerable domains are using CDNs, and particularly 10.72% are using multi-CDNs, indicating that using CDNs is one of the root causes of inconsistency across server-side TLS settings. For CDNs, there are numerous edge servers around the world and it would be challenging for CDN service providers to maintain consistent security settings for all of their machines. Especially when using multi-CDN, the spatial differences are likely to arise since edge servers are under the control of different CDN companies. Considering all the cases including non-CDNs, 22% of the vulnerable domains hosted on IP addresses owned by different entities. Therefore, it reveals that delegating end-to-end security to multiple CDN providers or web hosting providers is risky for websites.

Also, there are cases where security needs to be strengthened with the help of a third party. Among 4,195 vulnerable domains that could be downgraded to TLS 1.1 or older, only 0.6% (26 domains) of those domains are using CDN services. That means third party platforms such as CDNs mostly support at least TLS 1.2, while it is difficult for small websites running their own servers to upgrade vulnerable TLS versions or algorithms in a timely manner. Therefore, it can be recommended for small web service operators to delegate server management to third parties.

Chapter 5

Feasibility of the Teleport attack

To show the feasibility of the Teleport attack, we build the adversarial proxy network called the Teleport infrastructure across five regions and demonstrate the Teleport attack while measuring the delay latency due to redirection. Specifically, we measure the elapsed time required to establish the TLS session between a client and a vulnerable server (*i.e.*, with redirection), and compare it with the session setup time between a client and an original secure server (*i.e.*, without redirection).

Motivation. The Teleport attack redirects the client (*i.e.*, victim) to one of weak TLS servers, which might result in longer network latency. If the network delay becomes significantly higher and the victim experiences the unusual delay, she might notice that an attack is being under way. Therefore, we need to measure how much delay incurs when a Teleport attack is conducted.

The TELEPORT infrastructure. We build a TELEPORT infrastructure on the AWS service by running instances on five different regions: North America, South America, East Asia, South Asia, and Europe. For each instance, we execute `ProxyChains` to run an HTTP proxy, which forwards TLS messages to its nearest web server.

Experiments. The experiments are performed on a regional basis by establishing TLS sessions with and without the TELEPORT attack. We use `openssls_time` as a client that supports TLS 1.3. For each domain, we populate a three-tuple entry: (a domain name, the region where the client is located, the list of regions with lower TLS versions compared to the TLS version of the server in the same region as the client), based on our observations in §4. That is, we already know which regions have web servers with lower TLS versions for individual domains. For example, if `example.com` supports TLS 1.3 in regions A and C, but supports TLS 1.2 in region B and TLS 1.1 in regions D and E. If the client is in region A, we make an entry: (`example.com`, A, (B, D, E)). For the client in region B, there is an entry: ((`example.com`, B, (D, E)). Note that there is no entry for client in region D since there are no TLS servers with lower versions. Then, we conduct experiments at a client in each region. For instance, a client located in region A sends `ClientHello` directly to the web server of `example.com` in region A, while it also sends `ClientHello` redirected to the web servers in regions B, D and E.

Metric. To quantify the effect of the TELEPORT attack from a user's perspective, we introduce a ratio metric $\mathbf{r_e}$ to indicate how much delay incurs

due to the TELEPORT attack, defined as follows:

$$r_e = \frac{t_{c \to v} - t_{c \to o}}{t_{c \to o}}$$

where r_e is an expansion ratio indicating how much additional delay incurs due to the TELEPORT attack, $t_{c \to v}$ is a TLS handshake time between a client and a vulnerable server (with redirection), and $t_{c \to o}$ is a TLS handshake time between a client and its original server, which would be an edge server close to the client.

Results. The experiment results are shown in Table 5.1 and we find the two main observations.

First, the expansion ratio (r_e) varies across the regions up to 1.86. The time required to establish a session with an original server ($t_{c \to o}$) is relatively low in North America and Europe, due to the fact that many of the web servers are located in those regions. On the other hand, the average of elapsed times to establish a session with a vulnerable server is similar regardless of where the client is. As a result, the expansion ratio in North America and Europe is relatively high, which means that users in those regions might be able to recognize that they are being redirected/attacked.

Second, we find that there are some cases that exhibit similar time to establish a TLS session with and without the TELEPORT attack. This is due to the lack of an edge server in a nearby location; thus, even without a redirection, the client is connected to a server located in a different continent. In such cases, there is no significant difference in network latency due to the TELEPORT attack.

Based on the above two observations, we believe that it is difficult for

Origin	Metric	Redirected Location				
		North America	**South America**	**Europe**	**South Asia**	**East Asia**
North America (7,614 cases)	RTT (ms)	-	176.13	142.84	227.42	135.40
	$t_{c\to o}$ (ms)	-	468.75	526.46	463.86	526.83
	$t_{c\to v}$ (ms)	-	1259.35	751.99	1324.74	736.99
	$\mathbf{r_e}$	-	**1.69**	**0.43**	**1.86**	**0.40**
South America (6,311 cases)	RTT (ms)	176.17	-	197.64	302.80	293.96
	$t_{c\to o}$ (ms)	857.54	-	825.41	876.81	831.33
	$t_{c\to v}$ (ms)	999.69	-	978.83	1633.27	1135.54
	$\mathbf{r_e}$	**0.17**	-	**0.19**	**0.86**	**0.37**
Europe (5,667 cases)	RTT (ms)	142.77	197.36	-	106.84	252.11
	$t_{c\to o}$ (ms)	475.43	554.67	-	434.66	557.55
	$t_{c\to v}$ (ms)	906.10	1348.73	-	916.64	1113.48
	$\mathbf{r_e}$	**0.91**	**1.43**	-	**1.11**	**1.00**
South Asia (5,872 cases)	RTT (ms)	227.28	303.42	106.81	-	141.61
	$t_{c\to o}$ (ms)	703.92	694.63	673.20	-	644.43
	$t_{c\to v}$ (ms)	1226.90	1694.12	659.85	-	822.86
	$\mathbf{r_e}$	**0.74**	**1.44**	**-0.02**	-	**0.28**
East Asia (7,153 cases)	RTT (ms)	135.57	292.21	250.59	140.63	-
	$t_{c\to o}$ (ms)	782.49	702.17	835.62	725.25	-
	$t_{c\to v}$ (ms)	863.98	1671.61	1172.52	1108.15	-
	$\mathbf{r_e}$	**0.10**	**1.38**	**0.40**	**0.53**	-

Table 5.1: **Measurements of the latency incurred by the Teleport Attacks.** Experiments with at least 5K domains in each region show that the delay caused by the Teleport attack has an expansion ratio of up to 1.86. Even in the worst case, only less than a second is additionally required to establish a session. Interestingly, we find that there are some cases where a session is established faster.

users to recognize the Teleport attack. Since only less than one second incurs by the Teleport attack in the worst case and sometimes redirected sessions are established faster, it may be hard for users to recognize that the browser is redirected to a server located in a different continent [33].

Chapter 6

Discussions

6.1 Mitigation

Distributed monitoring system. The TELEPORT attack suggests that a TLS downgrade attack is practically feasible and can be successfully conducted without anyone (web server operators[1] and clients) noticing. Web server operators are burdened with keeping the security level of numerous servers. To mitigate this, a continuous monitoring system can be established. Rather than isolated monitoring, continuously checking the security status of edge servers at various points across the world, similar to the first step of the TELEPORT attacks, is necessary to cover the cases like multi-CDNs.

Moreover, the monitoring system periodically should update, aggregate, and disseminate the mapping dataset. If the size of the dataset is too large to handle, we may devise a space-efficient data structure similar to CRLite [34].

[1] If they analyze the IP addresses of the incoming clients, it might be possible to detect anomaly. However, we believe it is not easily configurable.

Also, browser vendors can add this functionality to their web browsers for clients. This functionality is to periodically download and use this dataset to check if users access domains that have multiple IP addresses. The web browser keeps track of the network latency of the domains and calculates the average of the network delays. If the network delays are notably longer than the average, the web browser may show a warning message to users or report anonymized packet logs to its security center for post-mortem analysis.

Automated management. It is also necessary to establish a system to keep the TLS configurations of multiple servers up to date. Of course, many companies already have similar systems, but small businesses will find it difficult to deploy such systems. Just as `Let's Encrypt` [35] has contributed to increasing the proliferation of TLS by automating the issuance and management of TLS certificates, it will help to maintain robust security of web servers including small websites if there is a configuration management system that automatically manages security settings for web servers in real time.

6.2 Limitations

Variability of targets. Our attack model is based on spatial differences in server-side configurations. Note that such kind of vulnerability cannot be fixed since the protocols, software programs, and their settings are continuously upgraded and changed. Hence the machines that correspond to a particular website all over the world cannot be synchronized in terms of security aspects. Also, a CDN service provider's mapping clients to servers may be dynamically determined. That is, the same user can be redirected to different servers at different times. Thus, the attack may not be successful if

the gap between the time of collecting the vulnerability data and the time of launching redirection attacks is long. In fact, there have been cases where the list of vulnerable domains we have collected is not valid after a few weeks; the vulnerable servers are upgraded in the meantime.

Browser warnings. Vulnerabilities in TLS 1.1 and 1.0 have long been known, and the recent decision by major browsers to no longer support those versions has laid the foundation for improved overall web security levels. Therefore, attempts to downgrade below TLS 1.1 using our attacks may be thwarted by browser warnings. However, our focus is not just attacks on specific TLS versions, but attacks on the inconsistency of supported versions and settings of distributed servers, often under the control of different entities.

Chapter 7

Related Work

We classify the related work into three areas: downgraded by manipulating protocol messages, by browsers, or by middleboxes.

Downgraded by manipulating protocol messages. In the Factoring RSA Export Keys (FREAK) [17] attack, MitM attackers manipulate the `ClientHello` message to make a server and a client handshake using a weak export-grade RSA algorithm that is not requested by the client. If a session is encrypted by an export-grade RSA algorithm with weak level encryption, the attacker can recover the RSA decryption key, and then decrypt all the messages following. The Logjam [18] attack is similar but takes advantage of the protocol defects of TLS 1.2 or earlier, forcing the server to select the export-grade Diffie-Hellman Ephemeral (DHE) key exchange algorithm. Both FREAK and Logjam attacks are a kind of TLS downgrade attacks that exploit 1990's export-grade algorithms.

The Decrypting RSA with Obsolete and Weakened eNcryption (DROWN)

[19] attack also exploits a vulnerability that shares a public key credential with a server that supports SSLv2, which can be mitigated only if SSLv2 is disabled on the server side. In the Security Losses from Obsolete and Truncated Transcript Hashes (SLOTH) [20] attack, an adversary downgrades a TLS session by forcibly adding vulnerable algorithms (RSA and MD5) to the SignatureAndHashAlgorithm field of the ServerKeyExchange message in TLS 1.2. Then, the client uses a hash algorithm which is not selected by the server, which enables attacks based on hash collisions.

Downgraded by browsers. The Padding Oracle On Downgraded Legacy Encryption (POODLE) [16] attack leverages a block padding vulnerability, taking advantage of the fact that some servers or clients still support SSL 3.0 for the compatibility with legacy systems. An MitM attacker deliberately drops a client's handshake messages and forces her to have a session downgraded to SSL 3.0. In a downgraded session, the attacker can exploit CBC padding vulnerabilities to decrypt sensitive data such as passwords among encrypted messages.

Downgraded by middleboxes. Several studies [36, 37, 38, 39] have reported that middleboxes can downgrade the TLS version due to incorrect implementations in the middlebox software. For example, a middlebox splits the TLS session between a client and a server, and establishes an old version TLS session with the server, while having an up-to-date TLS session with the client. Thus, the client cannot be aware that the TLS version is downgraded between the middlebox and the server.

There have been many attempts to perform new attacks against TLS. However, these studies typically focus on leveraging a hash collision or a

fallback mechanism of browsers. Note that our TELEPORT attack relies on the spatial differences of TLS versions and security settings of web servers across the globe, in contrast with the aforementioned TLS attacks.

Chapter 8

Conclusion

TLS is the de facto standard for secure communications on the Internet, and has addressed its vulnerabilities by upgrading versions. Web server operators may have tried to keep their software and security configurations up-to-date and secure. However, for world-wide popular web services, it is difficult to consistently manage globally distributed servers, resulting in spatial differences in terms of the TLS security level. We have presented a new Teleport attack exploiting this server-side disparity, investigated total 7M domains, and found 29K domains are subject to Teleport attacks. In addition, we devised a new algorithm to identify how web servers are managed and to analyze the causes of spatial differences in the vulnerable domains. Finally, we measured the overhead of the Teleport attack and demonstrated its feasibility.

Bibliography

[1] E. Rescorla and T. Dierks, "The Transport Layer Security (TLS) Protocol Version 1.2." RFC 5246, Aug. 2008.

[2] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3." RFC 8446, Aug. 2018.

[3] Google, "Google Transparency Report HTTPS encryption on the web." `https://transparencyreport.google.com/https/overview`. Accessed: 2020-07-29.

[4] D. T. Elgamal and K. E. Hickman, "The SSL Protocol," Internet-Draft draft-hickman-netscape-ssl-00, Internet Engineering Task Force, Apr. 1995. Work in Progress.

[5] S. Fluhrer, I. Mantin, and A. Shamir, "Weaknesses in the Key Scheduling Algorithm of RC4," in *Selected Areas in Cryptography* (S. Vaudenay and A. M. Youssef, eds.), (Berlin, Heidelberg), pp. 1–24, Springer Berlin Heidelberg, 2001.

[6] I. Mantin and A. Shamir, "A Practical Attack on Broadcast RC4," in *Fast Software Encryption* (M. Matsui, ed.), (Berlin, Heidelberg), pp. 152–164, Springer Berlin Heidelberg, 2002.

[7] G. Paul and S. Maitra, "Permutation after RC4 key scheduling reveals the secret key," in *Selected Areas in Cryptography, 14th International Workshop, SAC 2007, Ottawa, Canada, August 16-17, 2007, Revised Selected Papers* (C. M. Adams, A. Miri, and M. J. Wiener, eds.), vol. 4876 of *Lecture Notes in Computer Science*, pp. 360–377, Springer, 2007.

[8] N. AlFardan, D. J. Bernstein, K. G. Paterson, B. Poettering, and J. C. N. Schuldt, "On the security of rc4 in TLS," in *22nd USENIX Security Symposium (USENIX Security 13)*, (Washington, D.C.), pp. 305–320, USENIX Association, Aug. 2013.

[9] X. Wang and H. Yu, "How to Break MD5 and Other Hash Functions," in *Advances in Cryptology – EUROCRYPT 2005* (R. Cramer, ed.), (Berlin, Heidelberg), pp. 19–35, Springer Berlin Heidelberg, 2005.

[10] X. Wang, Y. L. Yin, and H. Yu, "Finding Collisions in the Full SHA-1," in *Advances in Cryptology – CRYPTO 2005* (V. Shoup, ed.), (Berlin, Heidelberg), pp. 17–36, Springer Berlin Heidelberg, 2005.

[11] T. Duong and J. Rizzo, "Here Come The $\bigoplus$ Ninjas," 2011.

[12] J. Rizzo and T. Duong, "The CRIME attack," in *ekoparty security conference*, vol. 2012, 2012.

[13] Y. Gluck, N. Harris, and A. Prado, "BREACH: reviving the CRIME attack," *Unpublished manuscript*, 2013.

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

[15] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman, "The Matter of Heartbleed," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14, (New York, NY, USA), p. 475–488, Association for Computing Machinery, 2014.

[16] B. Möller, T. Duong, and K. Kotowicz, "This POODLE bites: exploiting the SSL 3.0 fallback," *Security Advisory*, 2014.

[17] B. Beurdouche, K. Bhargavan, A. Delignat-Lavaud, C. Fournet, M. Kohlweiss, A. Pironti, P. Strub, and J. K. Zinzindohoue, "A Messy State of the Union: Taming the Composite State Machines of TLS," in *2015 IEEE Symposium on Security and Privacy*, pp. 535–552, 2015.

[18] D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. A. Halderman, N. Heninger, D. Springall, E. Thomé, L. Valenta, *et al.*, "Imperfect forward secrecy: How Diffie-Hellman fails in practice," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pp. 5–17, 2015.

[19] N. Aviram, S. Schinzel, J. Somorovsky, N. Heninger, M. Dankel, J. Steube, L. Valenta, D. Adrian, J. A. Halderman, V. Dukhovni, E. Käsper, S. Cohney, S. Engels, C. Paar, and Y. Shavitt, "DROWN: Breaking TLS using sslv2," in *25th USENIX Security Symposium*

(USENIX Security 16), (Austin, TX), pp. 689–706, USENIX Association, Aug. 2016.

[20] K. Bhargavan and G. Leurent, "Transcript collision attacks: Breaking authentication in TLS, IKE, and SSH," in *Network and Distributed System Security Symposium*, 2016.

[21] P. Kotzias, A. Razaghpanah, J. Amann, K. G. Paterson, N. Vallina-Rodriguez, and J. Caballero, "Coming of age: A longitudinal study of tls deployment," in *Proceedings of the Internet Measurement Conference 2018*, pp. 415–428, 2018.

[22] R. Holz, J. Hiller, J. Amann, A. Razaghpanah, T. Jost, N. Vallina-Rodriguez, and O. Hohlfeld, "Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization," *SIGCOMM Comput. Commun. Rev.*, vol. 50, p. 3–15, July 2020.

[23] R. L. Rivest, "The MD5 Message-Digest Algorithm." RFC 1321, Apr. 1992.

[24] C. Contavalli, W. van der Gaast, D. C. Lawrence, and W. A. Kumari, "Client Subnet in DNS Queries." RFC 7871, May 2016.

[25] D. Eastlake *et al.*, "Transport layer security (TLS) extensions: Extension definitions," 2011.

[26] S. Son and V. Shmatikov, "The hitchhiker's guide to DNS cache poisoning," in *International Conference on Security and Privacy in Communication Systems*, pp. 466–483, Springer, 2010.

[27] R. Merget, M. Brinkmann, N. Aviram, J. Somorovsky, J. Mittmann, and J. Schwenk, "Raccoon attack." `https://raccoon-attack.com/`, 2020.

[28] Rapid7, "Accelerate security, vuln management, compliance." `https://www.rapid7.com/`. Accessed: 2020-07-29.

[29] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe, "Enabling network security through active DNS datasets," in *Research in Attacks, Intrusions, and Defenses - 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings* (F. Monrose, M. Dacier, G. Blanc, and J. García-Alfaro, eds.), vol. 9854 of *Lecture Notes in Computer Science*, pp. 188–208, Springer, 2016.

[30] R. Singh, A. Dunna, and P. Gill, "Characterizing the deployment and performance of multi-cdns," in *Proceedings of the Internet Measurement Conference 2018*, pp. 168–174, 2018.

[31] "The CAIDA UCSD AS to Organization Mapping Dataset." `https://www.caida.org/data/as-organizations/`. Accessed: 2020-04-10.

[32] "Internet Control Message Protocol." RFC 792, Sept. 1981.

[33] F. F.-H. Nah, "A study on tolerable waiting time: how long are web users willing to wait?," *Behaviour & Information Technology*, vol. 23, no. 3, pp. 153–163, 2004.

[34] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "Crlite: A scalable system for pushing all tls revocations to all

browsers," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 539–556, IEEE, 2017.

[35] J. Aas, R. Barnes, B. Case, Z. Durumeric, P. Eckersley, A. Flores-López, J. A. Halderman, J. Hoffman-Andrews, J. Kasten, E. Rescorla, S. Schoen, and B. Warren, "Let's encrypt: An automated certificate authority to encrypt the entire web," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, CCS '19, (New York, NY, USA), p. 2473–2487, Association for Computing Machinery, 2019.

[36] Z. Durumeric, Z. Ma, D. Springall, R. Barnes, N. Sullivan, E. Bursztein, M. Bailey, J. A. Halderman, and V. Paxson, "The security impact of HTTPS interception," in *Network and Distributed Systems Symposium*, 2017.

[37] L. Waked, M. Mannan, and A. Youssef, "To Intercept or Not to Intercept: Analyzing TLS Interception in Network Appliances," in *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*, pp. 399–412, ACM, 2018.

[38] X. de Carné de Carnavalet and M. Mannan, "Killed by proxy: Analyzing client-end tls interception software," in *Network and Distributed System Security Symposium*, 2016.

[39] H. Lee, Z. Smith, J. Lim, G. Choi, S. Chun, T. Chung, and T. T. Kwon, "maTLS: How to Make TLS middlebox-aware?," in *NDSS*, 2019.

초 록

지난 수십 년간 TLS(Transport Layer Security)는 안전한 웹 콘텐츠의 전달을 위한 사실상의 표준으로 자리매김했다. 그러나 TLS는 오랜 기간동안 지속적으로 취약점을 노출해 왔으며, 그로 인해 TLS의 안전성은 지속적인 버전 업그레이드를 통해 보안 문제들을 해결함으로써 유지되어 왔다. 한편, 세계각지의 사용자들에게 웹 콘텐츠를 빠르게 전달하기 위하여 웹서비스 제공자들이 지리적으로 분산된 환경에서 많은 서버들을 유지할 필요성이 대두되었다. 그 결과 웹 호스팅 또는 CDN(Content Delivery Networks) 서비스 제공자에게 자신들의 웹 콘텐츠를 위임하는 경우가 많아졌으며, 이때 분산된 서버들의 보안 설정 또한 위임되어 관리 주체나 서비스 지역에 따라 달라질 수 있게 되었다.

이러한 TLS 보안 설정의 공간적 차이를 활용하여 우리는 새로운 TLS 다운그레이드 공격으로 이른바 텔레포트(Teleport) 공격을 제시한다. 이 공격 모델에서 공격자는 클라이언트의 지리적 위치에 따라 다른 TLS 버전과 암호 옵션을 제공하는 도메인들의 정보를 수집한다. 그런 다음, 클라이언트의 TLS 연결 메시지를 다른 지역의 취약한 서버로 우회시켜 서버와 클라이언트 양자가 알아차리지 못하게 TLS 세션을 다운그레이드한다. 우리는 실제 환경에서 얼마나 많은 도메인들이 텔레포트 공격에 취약한지를 측정하였으며, TLS 보안 설정의 공간적 차이가 발생하는 근본적인 원인을 추적하기 위한 분석을 수행하였다. 또한 여러 지역에서 세션 우회로 인한 지연 시간을 측정하여 텔레포트 공격의 실효성을 입증하였다.

주요어: 전송 계층 보안(TLS), 다운그레이드 공격, 콘텐츠 전송 네트워크(CDN), 도메인명 위임, 분산 환경

학번: 2019-23689