Analyzing Spatial Differences in the TLS Security of Delegated Web Services

Joonhee Lee*
Seoul National University
Seoul, Republic of Korea
hello82@snu.ac.kr

Hyunwoo Lee
Purdue University
West Lafayette, Indiana, USA
lee3816@purdue.edu

Jongheon Jeong
Seoul National University
Seoul, Republic of Korea
jjong0025@snu.ac.kr

Doowon Kim
University of Tennessee, Knoxville
Knoxville, Tennessee, USA
doowon@utk.edu

Ted "Taekyoung" Kwon
Seoul National University
Seoul, Republic of Korea
tkkwon@snu.ac.kr

ABSTRACT

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 (or origin web servers) need to deploy and 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 of distributed servers may vary spatially depending on the managing entities or locations.

Based on these *spatial differences* in TLS security, we find that the security level of TLS connections (and their web services) can be lowered. After collecting the information of (web) domains that exhibit different TLS versions and cryptographic options depending on clients' locations, we show that it is possible to redirect TLS handshake messages to weak TLS servers, which both the origin server and the client may not be aware of. We investigate 7M domains with these spatial differences of security levels in the wild and conduct the analyses to better understand the root causes of this phenomenon. We also measure redirection delays at various locations in the world to see whether there are noticeable delays in redirections.

CCS CONCEPTS

• **Security and privacy → Network security**.

KEYWORDS

domain name delegation; downgrade attack; TLS deployment

*Also with Korea Financial Telecommunications & Clearings Institute.

ASIA CCS '21, June 7–11, 2021, Virtual Event, Hong Kong

ACM ISBN 978-1-4503-8287-8/21/06…$15.00
https://doi.org/10.1145/3433210.3453107

ACM Reference Format:
Joonhee Lee, Hyunwoo Lee, Jongheon Jeong, Doowon Kim, and Ted "Taekyoung" Kwon. 2021. Analyzing Spatial Differences in the TLS Security of Delegated Web Services. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security (ASIA CCS '21), June 7–11, 2021, Virtual Event, Hong Kong*. ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3433210.3453107

1 INTRODUCTION

Transport Layer Security (TLS) [31, 32] 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 [18], 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 [15] was published in 1995, TLS/SSL have exposed various vulnerabilities such as weak cryptographic algorithms (*e.g.*, [5, 16, 24, 28, 38, 39]) and have been constantly attacked (*e.g.*, BEAST [12], CRIME [34], BREACH [17], Lucky Thirteen [4], Heartbleed [13], POODLE [26], FREAK [7], Logjam [3], DROWN [6], and SLOTH [8]).

TLS has been evolving and becomes more secure by solving or mitigating such security problems through continuous version upgrades [20]. The latest TLS version is 1.3 [31] that was redesigned and approved in August 2018. For example, it mandates *perfect forward secrecy* by allowing only 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 [19]. In other words, the other web servers (approximately 69%) still rely on lower 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.*, 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 server management responsibilities may lead to other security threats.

In this paper, we present a new threat model where an adversary 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, a Man-in-the-Middle attacker can redirect the clients' connection requests to vulnerable TLS servers even if the clients are supposed to establish connections with TLS servers with secure TLS parameters. We examine more than 7M domains from multiple vantage points across different continents and find that 28,956 domains in the wild exhibit different TLS configurations depending on the locations of the clients. To the best of our knowledge, this paper is the first to investigate the spatial differences in the TLS security settings of distributed delegated Web servers.

To better understand the root causes of such spatial differences, we also propose a new method to identify how the web servers of insecurely-configured domains are managed. In addition, we measure to what extent the time to establish a TLS session is lengthened if a TLS session is redirected to insecurely-configured servers. The result demonstrates the overhead latency of the redirection is not so substantial; that is, clients may not be aware of being redirected.

In summary, our contributions are as follows:
(1) Presenting a new threat model to weaken the security level of web services (Section 3): We design an infrastructure under an adversary's control that leverages the *spatial differences* in TLS security settings, due to the faulty or outdated management of server-side TLS configurations. We elaborate on (i) how to gather information (*e.g.*, IP addresses, TLS versions, ciphersuites, etc.) of web servers, and (ii) how an adversary can obtain the sensitive information of clients.
(2) Experimentation and analysis on real-world web services (Section 4): We conduct experiments to see if web services in the real-world show inconsistencies in TLS security settings depending on the clients' geographical locations. To better understand the reasons of the vulnerable domains, we provide a method of identifying how these web servers are managed. From the analysis, we find that about 25% of the insecurely-configured domains rely on CDNs.
(3) Measurement of redirection delays (Section 5): To show the feasibility of our threat model, we measure the overhead (*i.e.*, network latency) incurred by redirection to servers located in different continents. Interestingly, an increased time to establish a TLS session with a vulnerable domain is less than 1 second at most; thus, the redirection may go unnoticed by users.

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

2 BACKGROUND

2.1 TLS Handshakes and Downgrade Attacks

Agreement on the version and ciphersuite. Transport Layer Security (TLS) [31, 32] 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 clients send the highest TLS version they 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 [33, 39], 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 [26], FREAK [7], Logjam [3], DROWN [6] and SLOTH [8].

On the contrary, even though the client uses 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 novel approach that takes advantage of vulnerabilities of *server-side configurations in distributed environments*, which 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 [10] 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 rely on 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 method is effective.

3 NEW ATTACK MODEL

In this section, we present a new threat model where an adversary downgrades a TLS version or its ciphersuite by exploiting the spatial differences in the TLS security configuration settings across distributed CDN edge servers.

3.1 Overview

An 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 have for this goal is having distributed proxies under their control running around the world.

Specifically, the adversary intercepts packets between clients and web servers, and break confidentiality using the known TLS attacks as described in Section 7. To this end, the adversary first compromises or deploys a network node (close to a client) such as a WiFi Access Point (AP) to eavesdrop and to tamper with the packets exchanged between the clients and servers. For example, the adversary can install a WiFi access point with an attractive SSID (*e.g.*, Free WiFi) in public (*e.g.*, at a shopping mall). We also assume that the adversary runs proxies throughout the world; for instance, the adversary can purchase AWS instances from several regions, or manage machines to build their own infrastructure as illustrated in Figure 1. Using their global scale infrastructure, the adversary (e.g. a WiFi AP) can forward packets to any arbitrary proxies under their control. On the other hand, we assume the adversary *cannot* break cryptographic primitives such as AES. Furthermore, the adversary *cannot* compromise endpoints such as web browsers and web servers; thus, the TLS protocol is correctly performed.

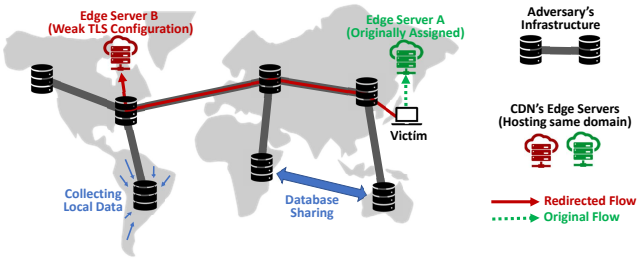


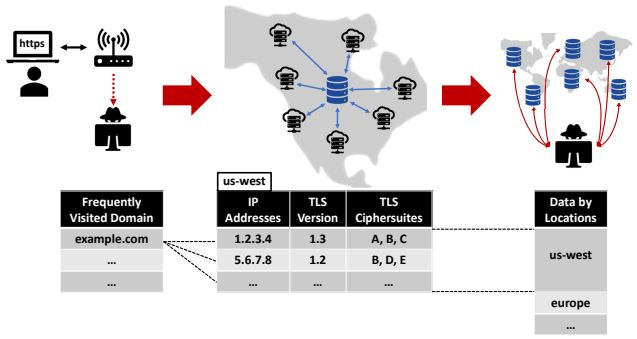
Figure 1: An infrastructure under the adversary's control—The adversary aims to obtain sensitive information of a client destined to target web servers. We assume that the adversary runs an infrastructure that consists of WiFi APs and proxies throughout the world. Each proxy in the infrastructure plays the two main roles: (i) collecting information about a target web server (*e.g.*, TLS security parameters) and (ii) redirecting TLS messages to an insecure server in a particular region.

With the above capabilities, the adversary can obtain the client's sensitive information in the three steps, which is illustrated in Figure 2. As illustrated in Figure 2a, the adversary's infra 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 Section 3.2). Then, as shown in Figure 2b, the adversary's infra (e.g., a WiFI AP) intercepts the TLS handshake messages, which will be redirected to vulnerable edge servers through their infrastructure (see Section 3.3). Finally, the adversary infra uses any known attack mechanisms against a downgraded session to steal sensitive information as illustrated in Figure 2c (see Section 3.4).

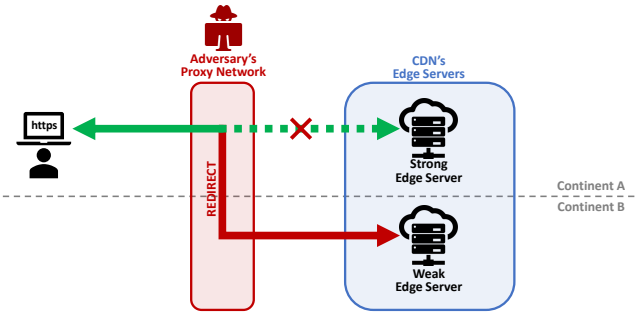
3.2 Populating Target Database

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

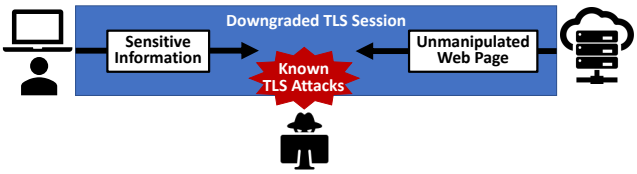
Target domains & IP addresses. An adversary can target any clients and perform the redirection. For an unspecified target, they periodically can scan some of the domain names (*e.g.*, Alexa 1M



(a) 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.



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



(c) Downgraded session exploitation: The adversary can successfully perform one of known attacks against the downgraded or weakened TLS sessions and steal sensitive information.

Figure 2: The methodology of downgrading TLS security is illustrated in three steps.

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 the client frequently visits.

Once the target domains are determined, the adversary collects the IP addresses that correspond to the domains. A domain may have multiple corresponding IP addresses. For example, a global website may have multiple IP addresses since it relies on multiple servers for load balancing or service stability/availability, or delegates fast content delivery to CDNs. Thus 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 2a.

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 adversary 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 lower TLS versions and weak cryptographic algorithms from `ServerHello` messages from target domains, he records the vulnerable versions and cipher 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, they 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) [1] 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 vulnerable edge servers. If found, it means that the victim can be redirected to a vulnerable TLS server (with the same domain name) instead of the one with which 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 authentic 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 its 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 [26] and Lucky Thirteen [4]). 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 [36]

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, they can redirect the client by spoofing the cached DNS records.

A client cannot recognize that they are redirected to another server since the authentic web pages of `example.com` are 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 may successfully obtain the sensitive information of the client (*i.e.*, victim).

If the network delay caused by such redirection is not significant, victims would not recognize whether they have been attacked. To understand to what extent the delay occurs, we quantify the increased latency due to the attack in Section 5. Note that 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.

3.4 Downgraded Session Exploitation

After making the victim perform a TLS handshake in a vulnerable version or ciphersuite, the known attacks mentioned in Section 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 [25]) 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 section, we will investigate the effect in the real world, especially regarding steps (i) and (iii). Then, we will evaluate how this redirection affects user experiences.

4 REAL-WORLD EXPERIMENTS

In this section, we examine real-world web services and analyze the results. To this end, we perform the first step of our methodology (Section 3.2) and find vulnerable domains in the wild. Then, we study how web servers are managed to better understand why the spatial differences exist.

Table 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 additional vulnerable TLS servers in the wild.

Dataset	Description	Source	# of Records
Target domains	Domain names	Rapid7 [30]	7,032,829
`Hello` messages	TLS versions & security parameters & IP addresses	Our TLS 1.3 client (from six vantage points)	42,196,974
DNS records	Mappings between IP addresses and domain names	ActiveDNS [21]	782,446,164

4.1 Data Collection

We collect three types of datasets as summarized in Table 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` [30] on June 16th, 2020. For wildcard names, we remove asterisks from names, which results in apex domains. Then we 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 our method 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* [21], 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,

Table 2: Types of vulnerable domains

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 (Excl. duplicates)	**21,907 (100%)**

(a) 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. This means that the domain is vulnerable since clients can be redirected to the server with vulnerable TLS 1.0.

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 (Excl. duplicates)	**16,267 (100%)**

(b) Weakened Ciphersuite—Some TLS servers may still rely on weak ciphersuites. For example, a domain has multiple web servers located around the world. One server supports forward secrecy, while another does not.

2020 from ActiveDNS. After finding out the IP addresses of the TLS servers vulnerable 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 Section 4.1, we connect the 7M domains from the six vantage points to find vulnerable domains. 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 key* algorithm, compared with other vantage points that support strong ciphersuites.

In this regard, we find total 28,956 vulnerable domains from ClientHello/ServerHello messages. The breakdown of the results is shown in Table 2, which leads to the following observations:

First, 21,907 domains have some edge servers with downgraded TLS versions. Given the size of the entire dataset, it may not be a

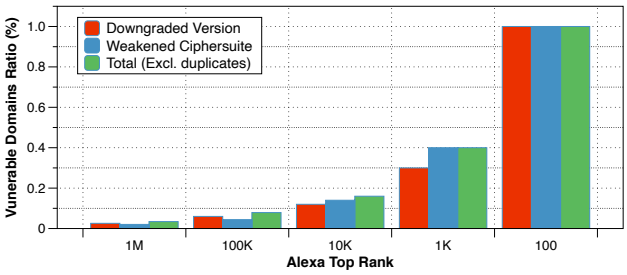


Figure 3: The ratio of vulnerable domains to the given set of domains increases as the ranks of Alexa top sites (i.e., the given set of domains) go higher.

threatening number. However, as Figure 3 shows, more popular domains are more likely to be vulnerable to the redirection attacks. This means that global scale web services are required to operate more servers in more distributed areas for quality of service, resulting in possibly more spatial differences. Also, surprisingly, TLS 1.0 servers (4,115, 18.78%) are still being in use. Most of domains have their TLS servers (17,740, 80.98%) use TLS 1.3 or TLS 1.2. The wide usage of old TLS versions highlights the vulnerability to the redirection attack. That is, an adversary may be able to 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 start with admin. It implies that many of the vulnerable domains that 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 additional vulnerable domains are shown in Table 3. Interestingly, 114,003 domains (41.8% out of 272,406) correspond to more than one IP addresses, which indicates that these domains have multiple TLS servers with 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 redirection.

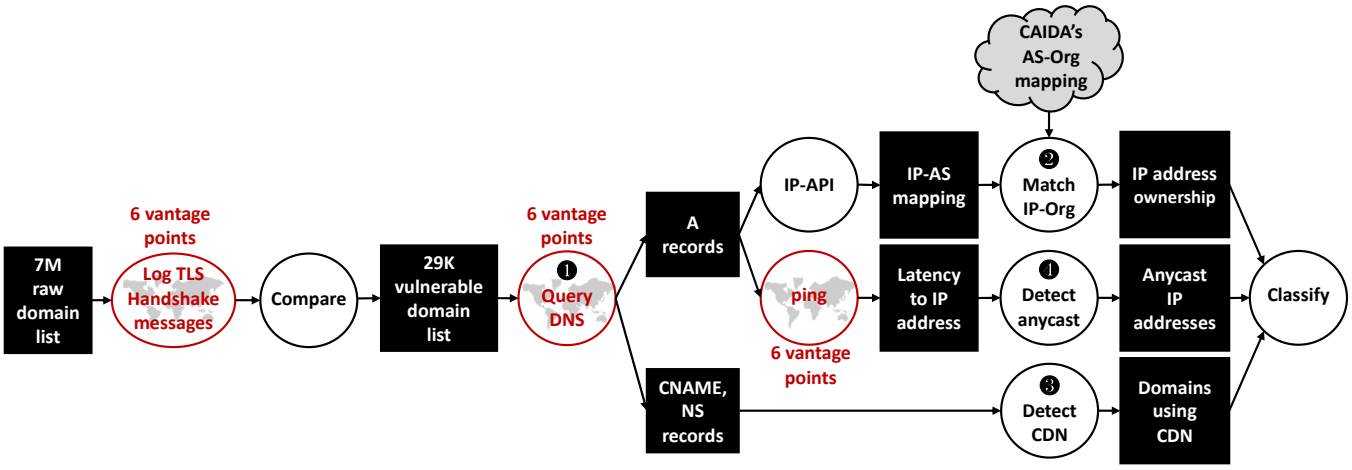


Figure 4: Identification process of the web server management—To better understand the root causes of spatial differences in TLS versions or ciphersuites, we devise a method to categorize the web hosting service providers (including CDNs) that exhibit the spatial differences. First, we find the vulnerable 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.

Table 3: More vulnerable domains—We present the breakdown of vulnerable domains in the wild from the 782M domain dataset.

Downgradable TLS Version			Weak Ciphersuite			
TLS 1.0	TLS 1.1	TLS 1.2	TLS-RSA	SHA1	AES-128	Total
2,948	7	340,620	549	837	74,598	419,559

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 redirections), 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 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 across the multiple servers located around the world that are primarily relying on multiple CDNs [35], 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 her/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.

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 be used for

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

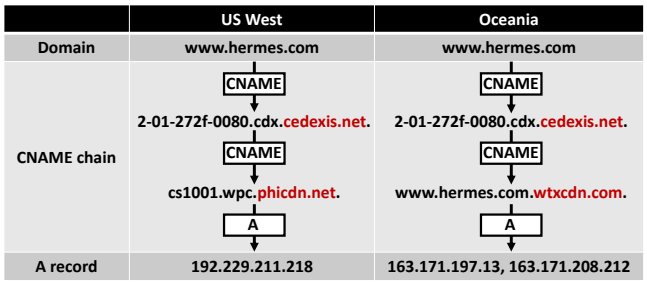


Figure 5: Examples of multi-CDN detection—There are two or more different CDN service providers for a CNAME chain of the domain `www.hermes.com`. Specifically, this domain is served from two CDNs: Verizon CDN from the US west and Alibaba Cloud Computing CDN from Oceania.

❸ CDN identification. Figure 4 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. An 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 a particular 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 information informs us of 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 [9] that maps between organizations and ASes; in total, it has 95,806 ASes mapped to 78,309 organizations. 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.

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

❸ **CDN detection.** As explained in Section 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 5. 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 check more rigorously, 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 [29] 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.

Table 4: Classification of vulnerable domains—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.

Web Hosting Provider	Type	Subtype	# of Domains	
			Rapid7 7M	Alexa 1M
CDN	DNS-based		220 (0.76%)	56 (5.18%)
	Anycast	Single IP address	416 (1.44%)	40 (3.70%)
		Multiple IP addresses	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 address	Non-anycast	17,926 (61.91%)	409 (37.80%)
		Anycast	4,410 (15.23%)	37 (3.42%)
	Multiple IP addresses owned by the same organization	Same IP address by region	2,218 (7.66%)	157 (14.51%)
		Different IP addresses by region	991 (3.42%)	70 (6.47%)
	Multiple IP addresses owned by different organizations	Same IP address by region	752 (2.60%)	28 (2.59%)
		Different IP addresses 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 (100.00%)**	**1,082 (100.00%)**

4.5 Classification Results

The results of classifying 28,956 vulnerable domains are shown in Table 4. 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 with 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 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], which gives us 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 website owners.

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 lower, 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.

5 OVERHEAD MEASUREMENT

To show the feasibility of our method, we build the adversarial proxy network across five regions and measure the delay latency due to the redirection. Specifically, we measure the session setup time to establish the TLS session between a client and a vulnerable server (*i.e.*, with redirection), which is compared with the one between a client and an original secure server (*i.e.*, without redirection).

Motivation. The adversary 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, the victim might notice that an attack is being under way. Therefore, we need to measure how much delay incurs when the TLS handshake is redirected.

The adversarial infrastructure. We build an adversarial 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

[4]Its domain name is masked for privacy.
[5]https://www.alexa.com/topsites

Table 5: Measurements of the latency incurred by the redirection—Experiments with at least 5K domains in each region show that the delay caused by the redirection 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.

Origin		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**	-

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 redirection. We use `OpenSSL s_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 Section 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 attack from a user's perspective, we introduce a ratio metric $\mathbf{r_e}$ to indicate how much delay incurs due to the redirection, 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 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 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 session setup delays to establish a TLS session with and without the redirection. 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 little difference in network latency.

Based on the above two observations, we believe that it is difficult for users to recognize the redirection. Since less than one second additionally incurs by the attack, and sometimes redirected sessions are established faster, it may be difficult for clients to figure out

that the browser is redirected to a server located in a different continent [27].

6 DISCUSSIONS

6.1 Mitigation

Distributed monitoring system. Our analysis demonstrates that a TLS handshake message could be successfully redirected to insecurely configured servers without anyone (web server operators[6] 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 our methodology, is necessary to cover the cases like multi-CDNs.

Moreover, the monitoring system should periodically 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 [22]. 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` [2] 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.

[6]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.

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 attacking on specific TLS versions, but attacks on the inconsistency of supported versions and settings of distributed servers, often under the control of different entities.

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) [7] 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 [3] 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) [6] 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) [8] 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) [26] 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 the client 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 [11, 14, 23, 37] 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 method relies on the spatial differences of TLS versions and security settings

of web servers across the globe, in contrast with the aforementioned TLS attacks.

8 CONCLUSION

TLS is the de facto standard for secure communications on the Internet and has addressed its vulnerabilities by upgrading its 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 method exploiting this server-side disparity and found 29K domains exhibit such spatial differences in their TLS configurations. In addition, we devised a new algorithm to identify how web servers are managed and to analyze why spatial differences exist in the vulnerable domains. Finally, we measured the overhead of the redirection and demonstrated its feasibility.

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their insightful feedback and suggestions. This work was supported by the National Research Foundation of Korea (NRF) grant (No. NRF-2016M3C4A7952587) funded by the Ministry of Science and ICT (MSIT) of Korea.

REFERENCES

[1] Donald E. Eastlake 3rd. 2011. Transport Layer Security (TLS) Extensions: Extension Definitions. RFC 6066. https://doi.org/10.17487/RFC6066

[2] Josh Aas, Richard Barnes, Benton Case, Zakir Durumeric, Peter Eckersley, Alan Flores-López, J. Alex Halderman, Jacob Hoffman-Andrews, James Kasten, Eric Rescorla, Seth Schoen, and Brad Warren. 2019. Let's Encrypt: An Automated Certificate Authority to Encrypt the Entire Web *(CCS '19)*. Association for Computing Machinery, New York, NY, USA, 2473–2487. https://doi.org/10.1145/3319535.3363192

[3] David Adrian, Karthikeyan Bhargavan, Zakir Durumeric, Pierrick Gaudry, Matthew Green, J. Alex Halderman, Nadia Heninger, Drew Springall, Emmanuel Thomé, Luke Valenta, Benjamin VanderSloot, Eric Wustrow, Santiago Zanella-Béguelin, and Paul Zimmermann. 2015. Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security* (Denver, Colorado, USA) *(CCS '15)*. Association for Computing Machinery, New York, NY, USA, 5–17. https://doi.org/10.1145/2810103.2813707

[4] Nadhem J. Al Fardan and Kenneth G. Paterson. 2013. Lucky Thirteen: Breaking the TLS and DTLS Record Protocols. In *Proceedings of the 2013 IEEE Symposium on Security and Privacy (SP '13)*. IEEE Computer Society, USA, 526–540. https://doi.org/10.1109/SP.2013.42

[5] Nadhem J. AlFardan, Daniel J. Bernstein, Kenneth G. Paterson, Bertram Poettering, and Jacob C. N. Schuldt. 2013. On the Security of RC4 in TLS. In *Proceedings of the 22nd USENIX Conference on Security* (Washington, D.C.) *(SEC '13)*. USENIX Association, USA, 305–320.

[6] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. 2016. DROWN: Breaking TLS Using SSLv2. In *Proceedings of the 25th USENIX Conference on Security Symposium* (Austin, TX, USA) *(SEC '16)*. USENIX Association, USA, 689–706.

[7] Benjamin Beurdouche, Karthikeyan Bhargavan, Antoine Delignat-Lavaud, Cédric Fournet, Markulf Kohlweiss, Alfredo Pironti, Pierre-Yves Strub, and Jean Karim Zinzindohoue. 2017. A Messy State of the Union: Taming the Composite State Machines of TLS. *Commun. ACM* 60, 2 (Jan. 2017), 99–107. https://doi.org/10.1145/3023357

[8] Karthikeyan Bhargavan and Gaëtan Leurent. 2016. Transcript Collision Attacks: Breaking Authentication in TLS, IKE, and SSH. In *23rd Annual Network and Distributed System Security Symposium* (San Diego, CA, USA) *(NDSS '16)*. Internet Society, Reston, VA, USA. https://doi.org/10.14722/ndss.2016.23418

[9] CAIDA. 2020. *The CAIDA UCSD AS to Organization Mapping Dataset.* Retrieved April 10, 2020 from https://www.caida.org/data/as-organizations/

[10] Carlo Contavalli, Wilmer van der Gaast, David C Lawrence, and Warren "Ace" Kumari. 2016. Client Subnet in DNS Queries. RFC 7871. https://doi.org/10.17487/RFC7871

[11] Xavier de Carné de Carnavalet and Mohammad Mannan. 2016. Killed by Proxy: Analyzing Client-end TLS Interception Software. In *23rd Annual Network and Distributed System Security Symposium* (San Diego, CA, USA) *(NDSS '16)*. Internet Society, Reston, VA, USA. http://dx.doi.org/10.14722/ndss.2016.23374

[12] Thai Duong and Juliano Rizzo. 2011. Here Come the ⊕ Ninjas.

[13] Zakir Durumeric, Frank Li, James Kasten, Johanna Amann, Jethro Beekman, Mathias Payer, Nicolas Weaver, David Adrian, Vern Paxson, Michael Bailey, and J. Alex Halderman. 2014. The Matter of Heartbleed. In *Proceedings of the 2014 Conference on Internet Measurement Conference* (Vancouver, BC, Canada) *(IMC '14)*. Association for Computing Machinery, New York, NY, USA, 475–488. https://doi.org/10.1145/2663716.2663755

[14] Zakir Durumeric, Zane Ma, Drew Springall, Richard Barnes, Nick Sullivan, Elie Bursztein, Michael Bailey, J. Alex Halderman, and Vern Paxson. 2017. The Security Impact of HTTPS Interception. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium* (San Diego, CA, USA). Internet Society, Reston, VA, USA. http://dx.doi.org/10.14722/ndss.2017.23456

[15] Taher Elgamal and Kipp E.B. Hickman. 1995. *The SSL Protocol.* Internet-Draft draft-hickman-netscape-ssl-00. Internet Engineering Task Force. https://datatracker.ietf.org/doc/html/draft-hickman-netscape-ssl-00 Work in Progress.

[16] Scott R. Fluhrer, Itsik Mantin, and Adi Shamir. 2001. Weaknesses in the Key Scheduling Algorithm of RC4. In *Revised Papers from the 8th Annual International Workshop on Selected Areas in Cryptography (SAC '01)*. Springer-Verlag, Berlin, Heidelberg, 1–24.

[17] Yoel Gluck, Neal Harris, and Angelo Prado. 2013. BREACH: Reviving the CRIME Attack. In *Black Hat USA 2013* (Las Vegas, NV, USA). http://breachattack.com

[18] Google. 2020. *Google Transparency Report: HTTPS encryption on the web.* Retrieved July 29, 2020 from https://transparencyreport.google.com/https

[19] Ralph Holz, Jens Hiller, Johanna Amann, Abbas Razaghpanah, Thomas Jost, Narseo Vallina-Rodriguez, and Oliver Hohlfeld. 2020. Tracking the Deployment of TLS 1.3 on the Web: A Story of Experimentation and Centralization. *SIGCOMM Comput. Commun. Rev.* 50, 3 (July 2020), 3–15. https://doi.org/10.1145/3411740.3411742

[20] Platon Kotzias, Abbas Razaghpanah, Johanna Amann, Kenneth G. Paterson, Narseo Vallina-Rodriguez, and Juan Caballero. 2018. Coming of Age: A Longitudinal Study of TLS Deployment. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 415–428. https://doi.org/10.1145/3278532.3278568

[21] Athanasios Kountouras, Panagiotis Kintis, Chaz Lever, Yizheng Chen, Yacin Nadji, David Dagon, Manos Antonakakis, and Rodney Joffe. 2016. Enabling Network Security Through Active DNS Datasets. In *Proceedings of the 19th International Symposium on Research in Attacks, Intrusions and Defenses* (Paris, France) *(RAID '16)*. Springer, 188–208. https://doi.org/10.1007/978-3-319-45719-2_9

[22] James Larisch, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. 2017. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE, 539–556.

[23] Hyunwoo Lee, Zach Smith, Junghwan Lim, Gyeongjae Choi, Selin Chun, Taejoong Chung, and Ted "Taekyoung" Kwon. 2019. maTLS: How to Make TLS middlebox-aware?. In *26th Annual Network and Distributed System Security Symposium* (San Diego, CA, USA) *(NDSS '19)*. Internet Society, Reston, VA, USA. https://dx.doi.org/10.14722/ndss.2019.23547

[24] Itsik Mantin and Adi Shamir. 2001. A Practical Attack on Broadcast RC4. In *Revised Papers from the 8th International Workshop on Fast Software Encryption (FSE '01)*. Springer-Verlag, Berlin, Heidelberg, 152–164.

[25] Robert Merget, Marcus Brinkmann, Nimrod Aviram, Juraj Somorovsky, Johannes Mittmann, and Jörg Schwenk. 2021. *Raccoon Attack: Finding and Exploiting Most-Significant-Bit-Oracles in TLS-DH(E).* https://raccoon-attack.com

[26] Bodo Möller, Thai Duong, and Krzysztof Kotowicz. 2014. This POODLE Bites: Exploiting The SSL 3.0 Fallback. *Security Advisory* 21 (2014), 34–58.

[27] Fiona Fui-Hoon Nah. 2004. A Study on Tolerable Waiting Time: How Long are Web Users Willing to Wait? *Behaviour & Information Technology* 23, 3 (2004), 153–163. https://doi.org/10.1080/01449290410001669914

[28] Goutam Paul and Subhamoy Maitra. 2007. Permutation after RC4 Key Scheduling Reveals the Secret Key. In *Proceedings of the 14th International Conference on Selected Areas in Cryptography* (Ottawa, Canada) *(SAC '07)*. Springer-Verlag, Berlin, Heidelberg, 360–377.

[29] John Postel. 1981. Internet Control Message Protocol. RFC 792. https://doi.org/10.17487/RFC0792

[30] Rapid7. 2020. *Rapid7 Open Data: SSL Certificates.* Retrieved July 29, 2020 from https://opendata.rapid7.com/sonar.ssl/

[31] Eric Rescorla. 2018. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446. https://doi.org/10.17487/RFC8446

[32] Eric Rescorla and Tim Dierks. 2008. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246. https://doi.org/10.17487/RFC5246

[33] Ronald L. Rivest. 1992. The MD5 Message-Digest Algorithm. RFC 1321. https://doi.org/10.17487/RFC1321

[34] Juliano Rizzo and Thai Duong. 2012. The CRIME Attack. In *Ekoparty Security Conference 2012.*

[35] Rachee Singh, Arun Dunna, and Phillipa Gill. 2018. Characterizing the Deployment and Performance of Multi-CDNs. In *Proceedings of the Internet Measurement Conference 2018* (Boston, MA, USA) *(IMC '18)*. Association for Computing Machinery, New York, NY, USA, 168–174. https://doi.org/10.1145/3278532.3278548

[36] Sooel Son and Vitaly Shmatikov. 2010. The Hitchhiker's Guide to DNS Cache Poisoning. In *Security and Privacy in Communication Networks (SecureComm '10)*, Sushil Jajodia and Jianying Zhou (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 466–483.

[37] Louis Waked, Mohammad Mannan, and Amr Youssef. 2018. 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* (Incheon, Republic of Korea) *(ASIACCS '18)*. Association for Computing Machinery, New York, NY, USA, 399–412. https://doi.org/10.1145/3196494.3196528

[38] Xiaoyun Wang, Yiqun Lisa Yin, and Hongbo Yu. 2005. Finding Collisions in the Full SHA-1. In *Proceedings of the 25th Annual International Conference on Advances in Cryptology* (Santa Barbara, California) *(CRYPTO '05)*. Springer-Verlag, Berlin, Heidelberg, 17–36. https://doi.org/10.1007/11535218_2

[39] Xiaoyun Wang and Hongbo Yu. 2005. How to Break MD5 and Other Hash Functions. In *Proceedings of the 24th Annual International Conference on Theory and Applications of Cryptographic Techniques* (Aarhus, Denmark) *(EUROCRYPT '05)*. Springer-Verlag, Berlin, Heidelberg, 19–35. https://doi.org/10.1007/11426639_2