Transport Layer Security

Vulnerabilities and Attacks Analysis

Neethu Manoharan

Jonathan Metzger

Maneesh Reddy Tamma

Panuwat Tragulroong

Northeastern University

CY6740 Network Security

Professor William Robertson

May 6th, 2022

Introduction

Transport Layer Security or TLS is a protocol that protects data by encryption to leverage the security from senders to receivers throughout the internet. TLS helps avoid data leaks and secure sensitive information that must not be sent unencrypted. This way hackers cannot see the real message behind the encrypted ones. Moreover, TLS can be applied to different tools and protocols such as FTP, SMTP, and DNS. Because of the internet, data can be transmitted anywhere, and TLS is making sure they are delivered with confidentiality and integrity. Another validity of TLS is that it is originally developed from Secured Socket Layer or SSL which is the technology to secure computer transmissions in the 1900s. The evolution of TLS made internet communication safer. However, TLS can only protect the data that is being transmitted between computers. It cannot protect or take responsibility for the computers themselves.

The importance of Transport Layer Security (TLS) is to implement the missing part of computer communications which is security. Back in the day, technologies were developed to assist and ease humans in doing work and make it more convenient and effective. However, the development usually did not come with the safety protocols. Therefore, TLS is developed to be a part of many protocols to gain the protection of the data transmitting process. Nowadays, there are different versions of TLS which were improved to close the loophole from the previous versions. We all know that the updated versions are necessary to keep fighting the ongoing and upcoming threats. Nevertheless, there is plenty of data on the internet that shows the huge number of devices that are still using these previous versions. According to Shodan, there are over 95.6 million devices that are using TLS, and 75.8 million are not supporting the latest version TLS 1.3. This shows that the announcement of the flaw in TLS's previous version did not encourage users to keep their TLS version up to date.

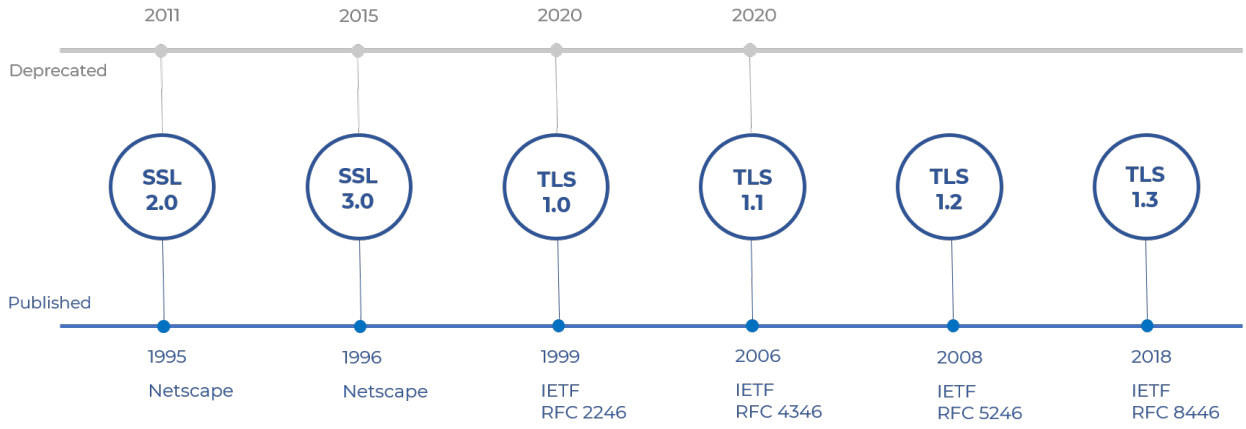
The significance of TLS has led the interesting questions about the vulnerabilities that come with it. This paper is focusing on the Common Vulnerability and Exposure (CVE), well-known vulnerabilities on different TLS versions. For example, Heartbleed and Poodle are one of the famous attacks that happened on prior TLS versions. We explored the attack vector on TLS which explained the methods of attacks and how they are related to different TLS versions. The research provided data from Shodan and Common Vulnerability Scoring System (CVSS) to help analyze the attacks. We used the CVSS to determine the difficulty of the attacks and briefly explain the possible way to mitigate the risks.

History - Transport Layer Security

Netscape Communications developed the earlier and deprecated versions of the Secure Sockets Layer (SSL 1.0, 2.0, 3.0 -1994, 1995, 1996). Transport Layer Security (TLS) was built upon the foundations of SSL specifications and was proposed by the open standard organization for the internet group - Internet Engineering Task Force (IETF). Both SSL and TLS are cryptographic protocol that facilitates secured communication channels over the internet. The protocol is used in several applications such as email, instant messaging, voice over IP, etc. However, the most widely used and known is its use in HTTP over TLS client-server communications.

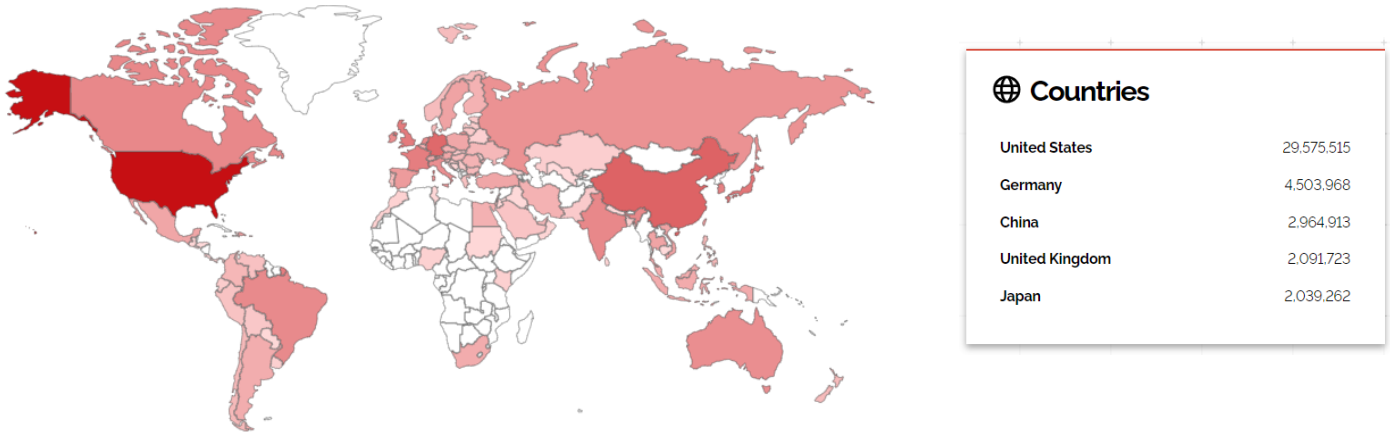
Due to serious security and usability flaws, SSL 1.0 was never released. SSL 2.0 was released first to the public in 1995 but was soon retracted. It was using the same cryptographic keys for message authentication and encryption. SSL 3.0 addressed most of the issues seen with its earlier version and was considered the base for TLS. IETF redefined the protocol and released its first version in 2006. TLS 1.1 was released a few years later in 2006 to protect against the cipher block chaining attacks, initialization vector vulnerability and handle padding errors seen with TLS 1.0. As of 2020, Apple, Microsoft, Google, and Mozilla deprecated TLS 1.1 and earlier versions.

In 2008, IETF released TLS version 1.2, and it has been the HTTPS standard for several years (even today). The version included significant changes to cryptographic features. Such as the use of SHA 256 in pseudorandom functions, hash algorithms, the ability for client and server to negotiate cryptographic algorithms during the handshake, etc. TLS 1.3 was released in 2018 with an even more exhaustive change list that provided enhanced security on par with the evolving cyberspace.

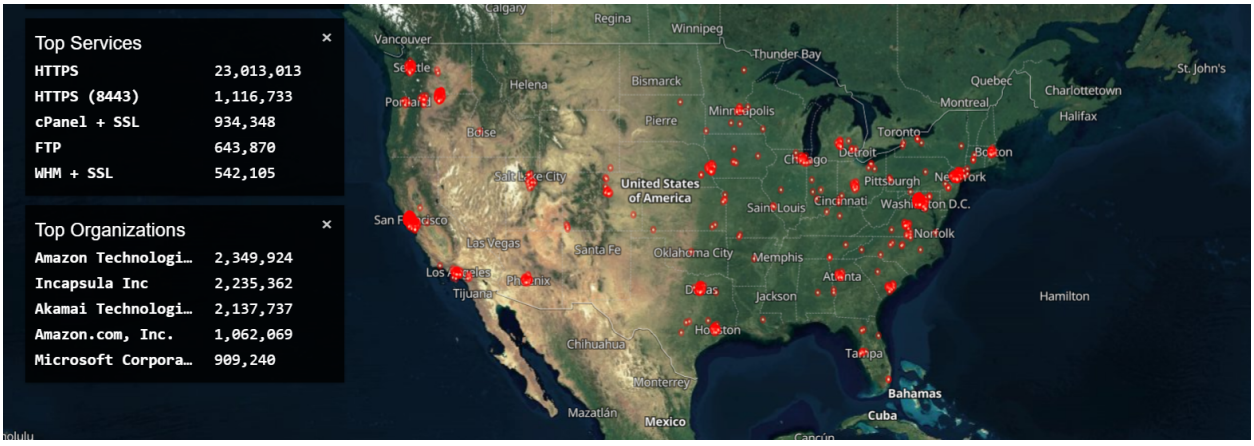


History of SSL/TLS

Though the current version of TLS is 1.3, many applications out in cyberspace have just started deprecating TLS 1.0 and TLS 1.1 and have started standardizing TLS 1.2. Due to the ubiquitous nature of TLS, such a slow upgrade can give a broad attack surface for malicious actors. In this paper we have analyzed different SSL/TLS versions to understand the protocol, its vulnerabilities, and attack vectors in depth. We also have utilized Shodan to evaluate the current state of the internet assets. For example, how many applications still use an older TLS version, say 1.0 etc. We have also analyzed various attacks on these TLS versions to understand the complexity, attack vectors, resource utilization, etc. The vulnerabilities have been ranked using the Common Vulnerability Scoring System. We have also created script for comparative analysis. [1]



Shodan report – Websites still hosting deprecated TLS versions 1.0, 1.1, 1.2



Shodan report – Top services, and organizations still supporting deprecated TLS versions in US

Vulnerabilities of Transport Layer Security – An in-depth analysis

Heartbleed: CVE-2014-0160

Summary

SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3

Heartbleed is a vulnerability in OpenSSL that could allow a remote attacker to expose sensitive data present in the server. This sensitive data could be authentication credentials, secret keys, database information, etc. This attack is based on the TLS heartbeat extension of incorrect memory handling. The attacker leverages a flaw in OpenSSL versions 1.0.1 through 1.0.1f, where an attacker can retrieve the private memory of a service hosted with this vulnerability in chunks of 64K at a time [2]. This heartbeat extension is primarily used as a keep-alive method between server and client, to make sure that the connection was not closed and both parties are still there. The process of heartbeat is initiated by the client sending a heartbeat message (data + payload length size) to the server, then the server responds with the same heartbeat request data from the client. The problem arises when a client sends a false data length, then the server will respond with the data + some other data from memory which then total size matches the number sent by the client. This makes the server leak unencrypted data from its memory, this data can be of any type like secret keys, credentials, important configuration, etc.

Exploitation technique

In a normal Heartbeat request, the client sends data, for example, “bird”, a 4-character length word, then the server responds with the same 4-letter word “bird”. But in an attack the attacker will modify this Heartbeat request by sending the word “bird” but specifying the length of the data as 500, now the server will respond with 500 lengths of data starting from the location in the memory where “bird” is stored. Here the client will get data starting with bird and 496 characters of data from the memory. This data can be important credentials, secret keys, or garbage random data. This can be prevented by updating to the latest version of OpenSSL, if not possible then recompile the installed version with -DOPENSSL_NO_HEARTBEATS.

Figure: CVSS base score, Impact sub score, Exploitability sub score

Shodan Analysis

The Heartbleed vulnerability's prevalence is relatively low compared to the total count of websites hosting TLS versions 1.2 and older. The primary reason is that Heartbleed was not a vulnerability within the TLS version but was caused by TLS implementation in OpenSSL cryptographic software library. If the users are still using the vulnerable version of OpenSSL, the vulnerability will still exist. This vulnerability gained fame due to the ease with which the attack could be formulated and the catchy name. Shodan statistics show that around 50k devices in the United States are still vulnerable to this attack. Apache and nginx are the top products used by the devices vulnerable to Heartbleed.

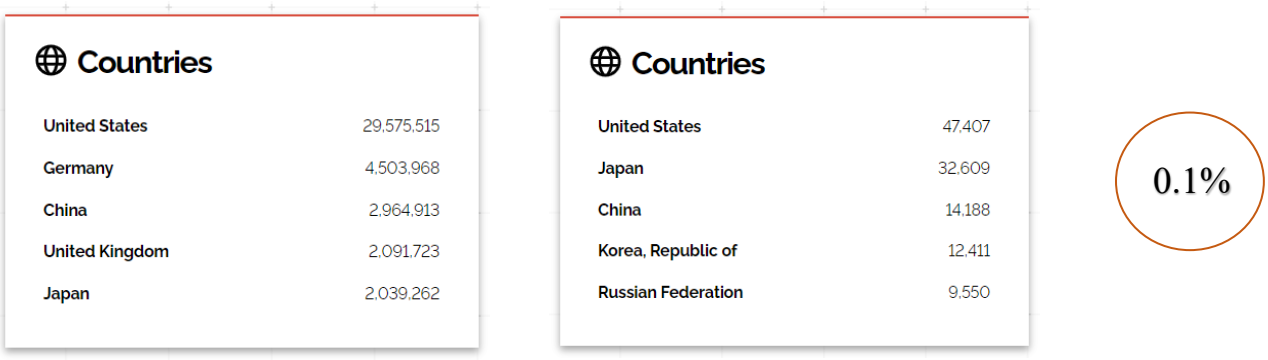


Figure: Total number of devices with TLS vulnerability Vs ones vulnerable to Heartbleed

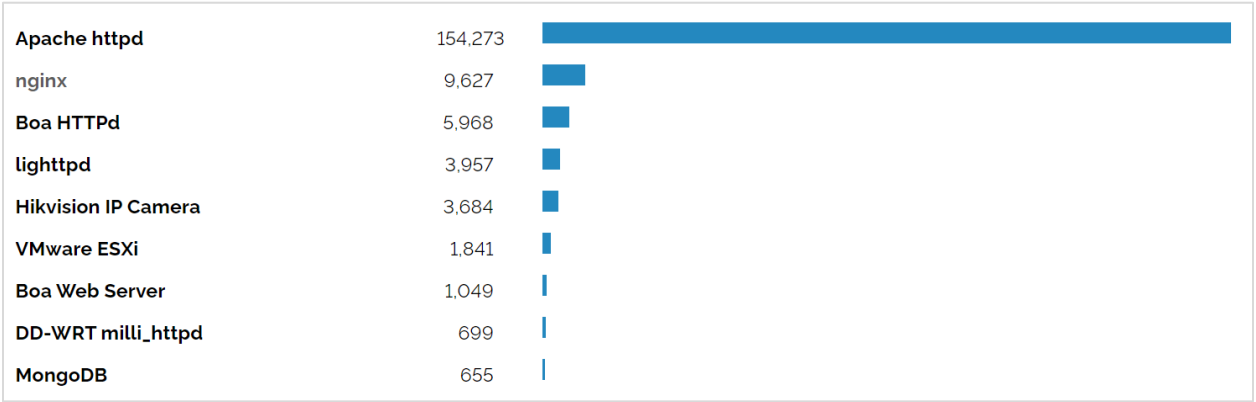


Figure: Top Products used in devices with CVE-2013-0169

LUCKY 13: CVE-2013-0169

Summary

SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3

Lucky 13 is a cryptographic timing attack that is used against TLS using CBC (Cipher Block Chaining) mode. The attack depends on the timing of server response over the network, it might not work in the real world properly, because requests travel very long in different paths that might cause disturbances in calculating the time taken for each request. The origin of the name Lucky 13 comes from the part of this calculation of TLS MAC of 13 bytes in this vulnerability [3].

Exploitation technique

The attacker must be in a Man-in-the-middle position, here the attacker can observe all the encrypted packet data flown from servers to the client. However, the data cannot be read but can be edited by shortening them, altering them, changing version numbers, etc. Each time the values were altered this breaks the TLS session that was being attacked, what if the modification is done in such a manner that might leak information that was encrypted. The attacker modifies and truncates the ciphertext generated by AES and HMAC-SHA1 algorithms. A normal LUCKY 13 attack requires about 223 TLS sessions in collecting a TLS-encrypted whole block of plain text. This can be decreased several times, at best the attacker requires 2^13 TLS sessions to recover one plain text byte [4]. There are some ways to protect from this type of attack – adding random time delays between each request sent by the user, moving to another cipher suites that are stronger than TLS-CBC, and changing Apache and Nginx configurations in the server.

Figure: CVSS base score, Impact sub score, Exploitability sub score

Shodan Analysis

Lucky 13 is not much prevalent compared to other vulnerabilities analyzed in this paper. But almost 18k devices were still tracked by Shodan to be vulnerable to the attack. As noted previously, the attack in TLS versions using cipher suite with CBC mode was affected in the implementation of OpenSSL and GnuTLS. Though Lucky 13 is not a very practical attack, it is interesting to note that the result set of the devices vulnerable to Lucky 13 also uses the weak cryptographic signature algorithm. SHA1 and MD5 were deprecated over a decade ago, and there are systems out in the world still utilizing the unsecured ones. The comparison of the number of devices in the United States with this vulnerability was 0.04% compared to the total number of devices with TLS vulnerabilities.

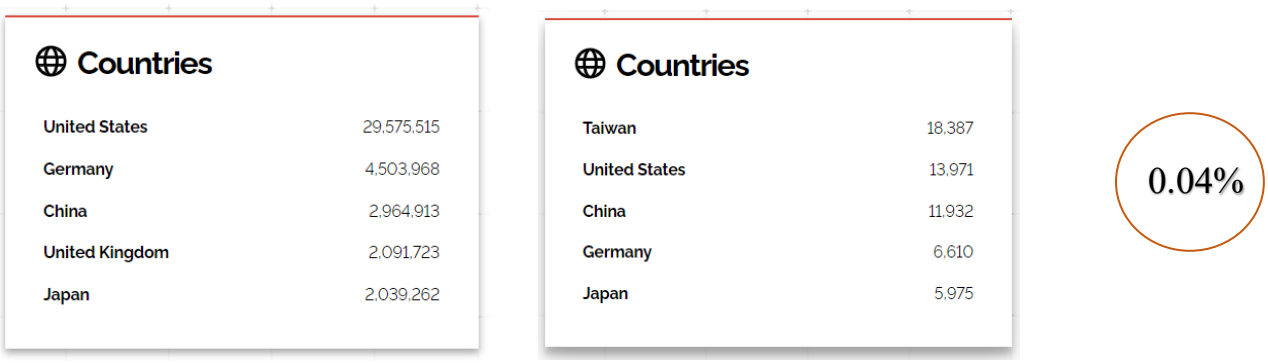


Figure: Total number of devices with TLS vulnerability Vs ones vulnerable to Lucky 13

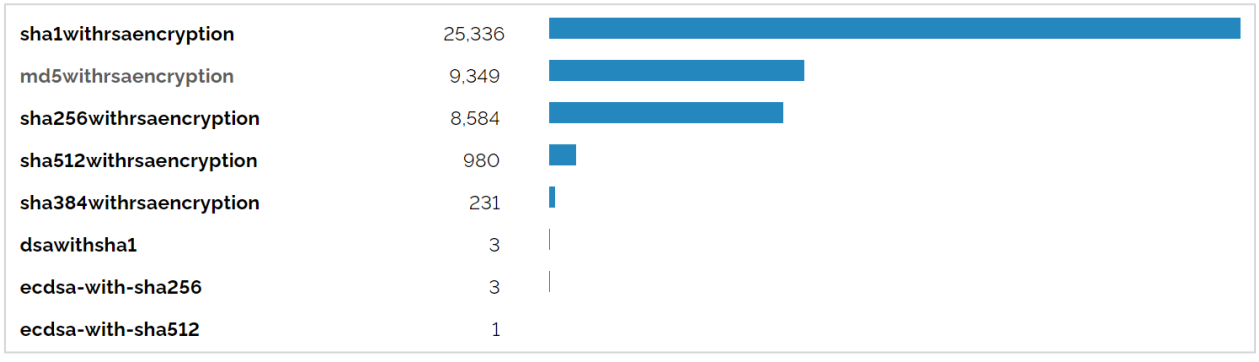


Figure: Top SSL certificate algorithm used in devices with CVE-2013-0169

POODLE: CVE-2014-3566

Summary

SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3

POODLE attack or Padding Oracle on Downgraded Legacy Encryption is an attack where an old unsecured SSL protocol is used to steal confidential information from secured connections. An attacker can eavesdrop on encrypted HTTPS traffic with the use of the SSL 3.0 protocol [5]. The attacker intercepts the connection between the user browser and the server. Then the attacker will force the browser to downgrade the server's security protocol from any TLS 1.2 version to the SSL 3.0 version. In this protocol the encryption is weak, which helps the attacker to steal any credentials sent to the server. Works on any TLS version except TLS 1.3.

Exploitation technique

Initially, the attacker performs a successful Man-in-the-Middle (MITM) attack, intercepting traffic between client and server. Nothing can be read or viewed as the user is using the secure TLS version, all the data is encrypted. Now the attacker must convince the server to communicate using an old SSL 3.0 protocol so that attacker can view the data easily. The attacker can drop the connections, so that server attempts to connect over an older version than the current one. This process is repeated until the servers use the SSL 3.0 protocol. This is also called a protocol downgrade attack [6]. After the server and client communicate over SSL 3.0 protocol, the attacker can decrypt the selected parts of the communication and steal confidential information. To prevent POODLE attacks, users/website admins can disable SSL 3.0 protocol, upgrade to major browsers like Google Chrome, Firefox, Safari, etc., and upgrade to TLS 1.3 version.

Figure: CVSS base score, Impact sub score, Exploitability sub score

Shodan Analysis

The POODLE was one of the significant weaknesses found in SSL 3.0 that led to the recreation of the protocol by IETF to TLS 1.0. The number of devices in the United States with this vulnerability was 0.4% compared to the total number of devices with TLS vulnerabilities. Almost 60K devices in cyberspace can still be exploited with this attack. Further analysis of these devices showed us the existence of several other vulnerabilities such as expired certificates, self-signed certificates, and unsecured ports. These snapshots of information give an insight into the security standards of an organization in general. Information obtained from such poorly secured organizations could then be exploited to be used as bots for large-scale attacks.

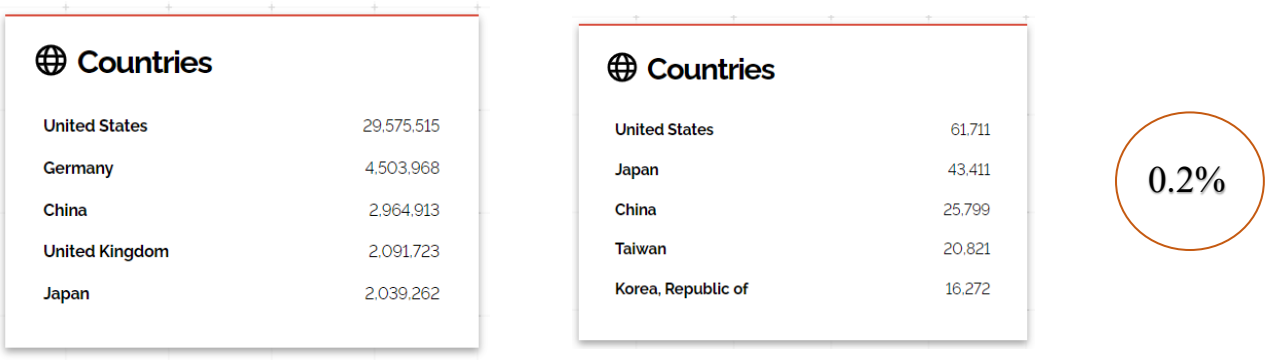


Figure: Total number of devices with TLS vulnerability Vs ones vulnerable to POODLE

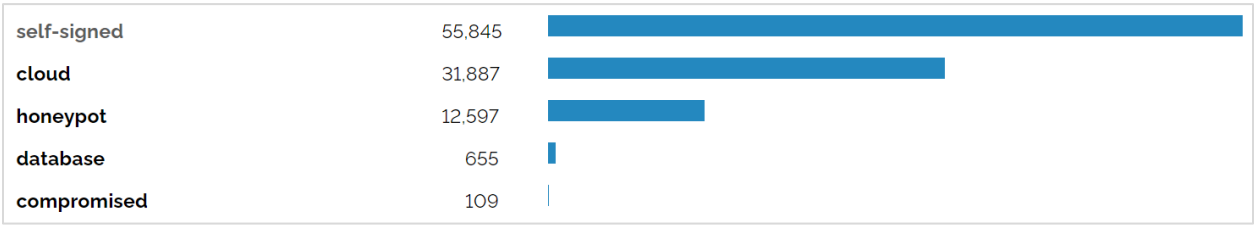


Figure: Top tags used in devices with CVE-2014-3566

Figure: Number of devices with expired certificate

BREACH: CVE-2013-3587

Summary

SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3

BREACH attack, also known as Browser Reconnaissance and Exfiltration via Adaptive Compression of Hypertext, is a similar type of attack compared to the CRIME attack. It is a security vulnerability against HTTPS when using HTTP compression. The attacker exploits the HTTP compression technique by simply guessing characters and symbols. This is done without any downgrading of SSL and not tampering with SSL/TLS, so it is vulnerable to any SSL/TLS version. The attack primarily works by taking advantage of the compressed size of the text [7]. There are some prerequisites for this BREACH attack – the application must support HTTP compression, User input should be reflected in the response, and a MITM (Man-in-the-middle) attack must be possible on the victim by the attacker, and the response from the HTTP must contain some secret data like CSRF token.

Exploitation technique

Consider a web application with a get request containing some query data sent in the URL, like *GET /api/data.php?id=456789*, then this request might generate a response with some token value like *token=dfvefcrgad556g*. This functionality can be leveraged by the attacker, by injecting a token value in the id parameter. If the attacker managed to inject a correct token value that matches the actual token, then the length of the response decreases by the length of the duplicate strings due to HTTP compression [8]. The attack begins by sending a single character as the token value, for example, the token is sent as '*?id=token=a*', of character '*a*', the response length is noted, and this continues with each character. If there is matched character, the actual token begins with the letter 'n', then the response for '*?id=token=n*' will be less length, this determines the correct letter in the actual token. The same process is done to guess each character in the token value string. The BREACH attack can be executed in very less time unless there are some rate limitations in the service the attack is being conducted. Typically, it should not take more than a few minutes, depending on the computing power the attacker possesses, internet connectivity, number of requests sent per second, and the actual length of the token value to guess. There are some mitigations for this BREACH attack – It can be avoided by disabling HTTP compression, randomizing secrets per request, masking secrets by XORing with random values, and hiding the

length by randomizing in responses, limiting the number of requests for a user to send (rate limitation).

Figure: CVSS base score, Impact sub score, Exploitability sub score

Shodan Analysis

Shodan search did not fetch any results for BREACH: CVE-2013-3587.

FREAK (CVE-2015-0204)

Summary

SSL 3.0	TLS 1.0	TLS 1.1	TLS 1.2	TLS 1.3

Factoring RSA Export Keys is an SSL/TLS vulnerability attack, where the RSA key exchange is leveraged to be used to secretly negotiate pre-master secrets. This attack allows an attacker to intercept HTTPS connections between vulnerable clients and servers, then force them to use weak encryption. This helps the attacker to break the insecure encryption and steal any confidential information exchanged between client and server. This attack is only possible when a vulnerable client browser connects to a server that accepts export-grade encryption [9]. Clients and servers use export RSA which are 512-bit-long weak export RSA encryption keys. As of March 6th, 2015, around 9.5% of Alexa top 1 million websites were vulnerable.

	Currently Vulnerable	Change Since Mar. 3
HTTPS servers at Alexa Top 1 Million domain names	8.5%	down from 9.6%
HTTPS servers with browser-trusted certificates	6.5%	down from 36.7%
All HTTPS servers	11.8%	down from 26.3%

Figure: Websites vulnerable to FREAK attack

Exploitation technique

To make this attack possible there are some criteria – the server must support RSA export cipher, the client is using a vulnerable OpenSSL version or offer an RSA export cipher or using Apple Secure Transport. Initially, the attacker must inspect traffic between client and server, by establishing a Man-in-the-Middle attack. The client and server normally communicate using a secure 2048-bit RSA with AES 128-bit ciphers, but now the attacker will send a client hello with only export-grade cipher like – TLS_RSA_EXPORT_WITH_DES40_CBC_SHA. The server simply proceeds to generate 512-bit RAS key pair and communicate using these insecure keys with the client. The TLS extension at the server sends the 512-bit public key to the client. As this key is only 512 bits in length, the attacker after sniffing can break this 512-bit private key, then this key can be used to decrypt all the traffic data exchanged between the client and server. There are some mitigations for this attack, like – patching the OpenSSL version and disabling export-grade ciphers, using stronger cipher suits like Diffie-Hellman (DH) key exchange or Elliptic DH in ephemeral mode.

Figure: CVSS base score, Impact sub score, Exploitability sub score

Shodan Analysis

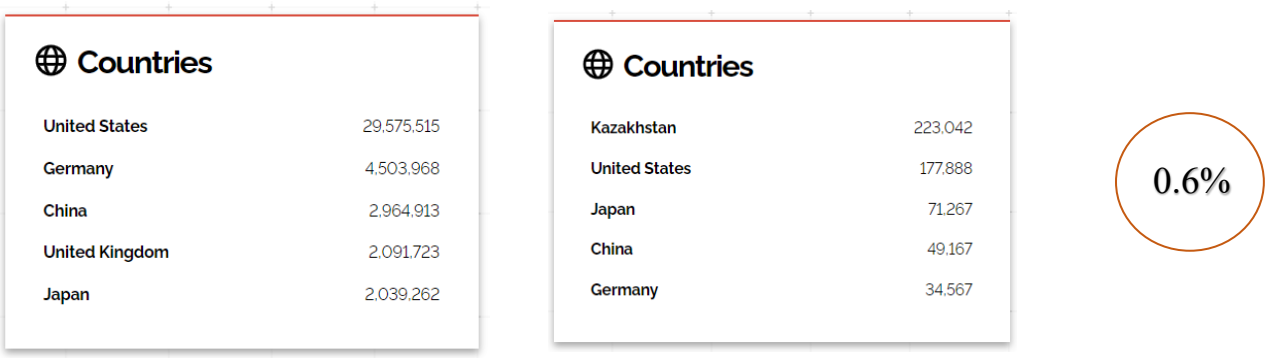


Figure: Total number of devices with TLS vulnerability Vs ones vulnerable to FREAK

MD5 with RSA encryption and SHA1 with RSA encryption made the top of the SSL certificate signature algorithms used in devices already vulnerable to Freak attacks. MD5 and SHA1 were deprecated by the National Institute of Standards and Technology (NIST) a decade back, and still, there is a prevalence of these hashes.

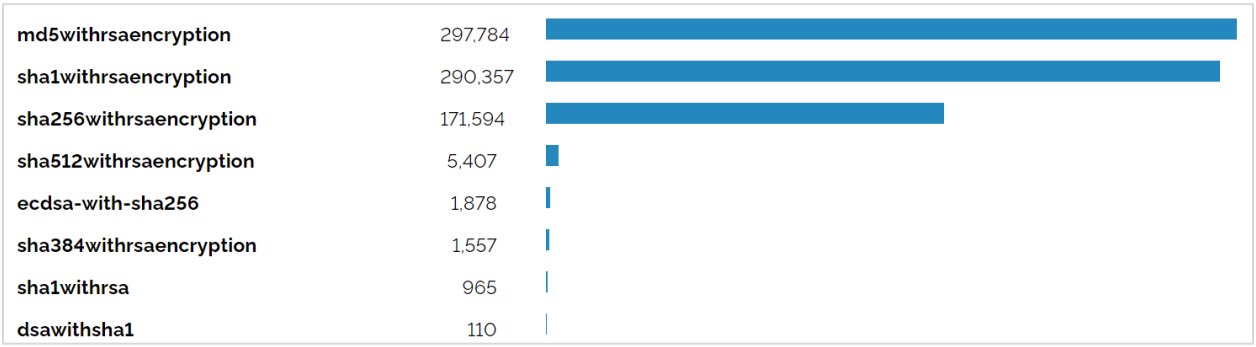


Figure: SSL Certificate algorithms used in devices vulnerable to FREAK

Shodan analysis of the devices vulnerable to the FREAK attack showed quite interesting data. Though a limited number we observed the existence of the vulnerabilities in print server, printer, webcam, VoIP. Firewall, WAP, routers showed the major share of vulnerability. These data validate the importance of security measures and upgrading the security patches.

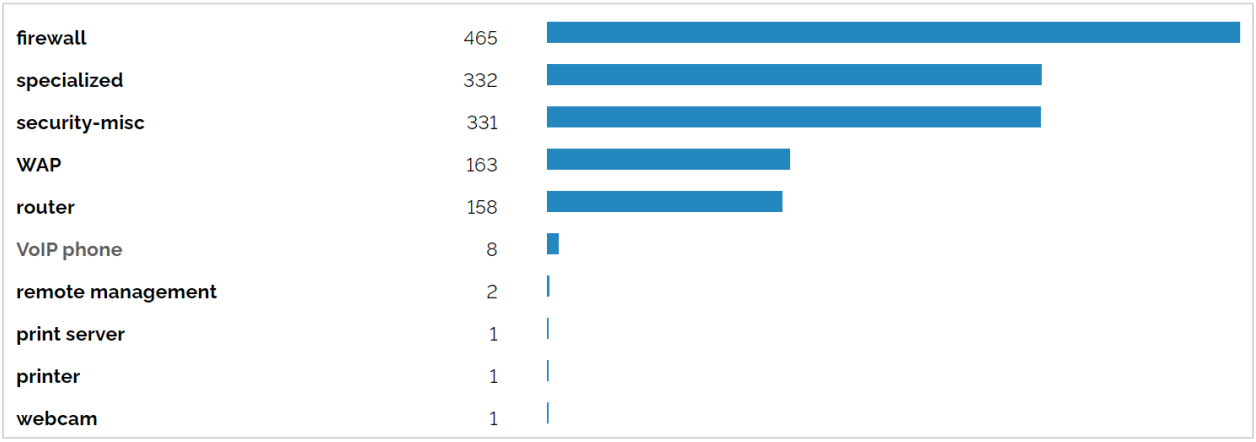


Figure: Products vulnerable to FREAK

Comparative Study

Applications using Protocols

We used the following shodan command to list organizations and countries around the world that use SSL/TLS in their applications, with {version} being sslv2, sslv3, tls1, tls1.1, tls1.2, and tls1.3. The result can be identified in the table below

```
$ shodan stats --facets ssl.version:{version} HTTP
```

Organizations and Countries	sslv2	sslv3	tlsv1	tlsv1.1	tlsv1.2	tlsv1.3	Total
Italy	12,034						**12,034**
Taiwan	25,296						**25,296**
Chunghwa Telecom Co.,Ltd.	8,156	18,640					**26,796**
Jumpline Inc	13,215	25,694					**38,909**
Charter Communications Inc	4,746	61,957					**66,703**
Comcast Cable Communications, LLC	5,803	35,787		189,368			**230,958**
Deutsche Telekom AG	2,663	15,766			312,269	142,396	**473,094**
Kazakhstan	237,422	239,757					**477,179**
DigitalOcean, LLC			173,529	195,736		261,417	**630,682**
Unified Layer					536,789	172,278	**709,067**
Aliyun Computing Co., LTD	2,745	14,847	206,529	314,780	349,122		**888,023**
Amazon Data Services NoVa			384,204	405,489	719,442	143,856	**1,652,991**
Korea	17,043	125,204	914,766	900,884			**1,957,897**
Amazon.com, Inc.		36,833	725,350	749,173	1,088,039	451,452	**3,050,847**
India			604,997	680,482	1,234,475	547,139	**3,067,093**
Netherlands			621,356	669,820	1,736,127	513,151	**3,540,454**
France		95,697	940,915	995,695	1,956,842	645,350	**4,634,499**
England	12,310	99,027	1,018,759	1,081,053	2,123,708	674,111	**5,008,968**
Japan	13,902	151,238	1,246,384	1,301,301	2,080,970	558,629	**5,352,424**
Amazon Technologies Inc.		24,367	1,364,521	1,431,069	2,392,487	410,588	**5,623,032**
China	29,905	181,517	2,091,891	2,418,308	3,045,744	515,354	**8,282,719**
Germany	19,060	150,610	1,903,827	2,000,082	4,625,123	2,030,442	**10,729,144**
United States	93,264	871,367	17,293,747	18,328,633	30,516,171	5,878,715	**72,981,897**
Total	**497,564**	**2,148,308**	**29,490,775**	**31,661,873**	**52,717,308**	**12,944,878**	**129,460,706**

Table 1 List of Applications that use SSL/TLS for Countries and Organizations

Attacks from Application Vulnerabilities

We then used the following shodan command to list organizations and countries around the world that were vulnerable to multiple attacks, with {cve} being the CVE number of lucky13 (2013-0169), heartbleed (2014-0160), poodle (2014-3566), freak (2015-0204), logjam (2015-4000) and sweet32 (2016-2183). The result can be identified in the table below

```
$ shodan stats --facets vuln:CVE-{cve}
```

Organizations and Countries	lucky13	heartbleed	poodle	freak	logjam	sweet32	Total
Deutsche Telekom AG					1,008		**1,008**
Amazon Data Services NoVa						1,074	**1,074**
Amazon.com, Inc.						1,451	**1,451**
DigitalOcean, LLC		1,468	1,319			1,503	**4,290**
Comcast Cable Communications, LLC	474			3,144	1,530		**5,148**
Charter Communications Inc	302			3,317	1,569		**5,188**
India		6,781					**6,781**
Unified Layer	265	1,376	2,055	2,304		1,397	**7,397**
Netherlands		5,258			10,157		**15,415**
Amazon Technologies Inc.	419	2,935	3,363	3,565		5,633	**15,915**
Aliyun Computing Co., LTD	1,260	1,731	4,008	3,644	951	4,537	**16,131**
Chunghwa Telecom Co.,Ltd.	6,558		6,952	6,029	3,939	2,440	**25,918**
England				17,922	11,612		**29,534**
Korea	5,047	12,644	16,677			11,116	**45,484**
Jumpline Inc				43,400	20,277		**63,677**
France	5,283	7,411	10,914	24,233	14,333	8,840	**71,014**
Italy	6,039	5,450	10,116	25,554	19,250	7,441	**73,850**
Taiwan	19,578		22,072	20,847	12,956	8,818	**84,271**
Germany	6,948	9,537	13,829	36,292	21,620	10,409	**98,635**
China	12,583	14,642	27,056	51,692	26,973	24,667	**157,613**
Japan	6,187	33,093	44,570	74,429	17,186	37,363	**212,828**
Kazakhstan				238,081			**238,081**
United States	14,412	48,208	63,116	182,674	90,861	52,629	**451,900**
Total	**85,355**	**150,534**	**226,047**	**737,127**	**254,222**	**179,318**	**1,632,603**

Table 2 List of Vulnerabilities for Countries and Organizations

Analyze Number of Applications and Number of Attacks

From the information taken of the number of applications per organization and country and the number of attacks on these applications, we rank and compare the two in the table below. The table ranks the total applications and total attacks from most to least. We compare the two to find the ratio of protocols implemented on each organization and county's applications and the number of attacks on said applications. Anything above "Total" is below average, and below is above average. We can see that Amazon.com Inc. has the least vulnerability in its applications, and the country of Italy has the most.

Organizations and Countries	Rank Apps	Total Apps	Rank Attacks	Total Attacks	Ratio
Amazon.com, Inc.	11	3,050,847	22	1,451	0.05%
Amazon Data Services NoVa	13	1,652,991	23	1,074	0.06%
Deutsche Telekom AG	18	473,094	24	1,008	0.21%
India	10	3,067,093	18	6,781	0.22%
Amazon Technologies Inc.	5	5,623,032	15	15,915	0.28%
Netherlands	9	3,540,454	16	15,415	0.44%
England	7	5,008,968	12	29,534	0.59%
United States	2	72,981,897	2	451,900	0.62%
DigitalOcean, LLC	16	630,682	21	4,290	0.68%
Germany	3	10,729,144	6	98,635	0.92%
Unified Layer	15	709,067	17	7,397	1.04%
Total	1	129,460,706	1	1,632,603	1.26%
France	8	4,634,499	9	71,014	1.53%
Aliyun Computing Co., LTD	14	888,023	14	16,131	1.82%
China	4	8,282,719	5	157,613	1.90%
Comcast Cable Communications, LLC	19	230,958	20	5,148	2.23%
Korea	12	1,957,897	11	45,484	2.32%
Japan	6	5,352,424	4	212,828	3.98%
Charter Communications Inc	20	66,703	19	5,188	7.78%
Kazakhstan	17	477,179	3	238,081	49.89%
Chunghwa Telecom Co.,Ltd.	22	26,796	13	25,918	96.72%
Jumpline Inc	21	38,909	10	63,677	163.66%
Taiwan	23	25,296	7	84,271	333.14%
Italy	24	12,034	8	73,850	613.68%

Table: Comparing Total Applications with SSL/TLS and Total Vulnerabilities for Countries and Organizations

We compare organizations and countries worldwide that use SSL/TLS and have been vulnerable to the attacks we mention. The data points out that Chunghwa Teleco Co., Ltd only has SSL implemented in their applications, and 96.77% have been attacked. Kazakhstan is similar, with 49.89% of its applications vulnerable to these attacks. On the contrary, Amazon utilizes more recent versions of TLS and has the least number of attacks on their applications from the data below. England also has the latest versions of TLS on its applications and only has 0.59% of its applications vulnerable to various attacks.

This analysis paints a picture that organizations and countries that utilize the latest versions of TLS in their applications are more protected from vulnerabilities and attacks than applications that do not support later versions of the internet protocol. The latest implementation of TLS1.3 will become more necessary when more attacks are created to try to invade organizations and countries around the world.

Conclusion

According to the analysis of Shodan data, we can understand that TLS is the security implementation. Some of the vulnerabilities were from the main application that used TLS to support the security approach like Heartbleed. The main vulnerability is the OpenSSL which plays the main part. Even though TLS is updated to the latest version, OpenSSL stays the susceptible one. It can't mitigate the flaws. Moreover, the CVE helps identify the well-known vulnerabilities to establish the specific flaws. Using Shodan data, we found that most of the mentioned CVEs have been dealt with to mitigate the risk. Nonetheless, they do not relate directly to TLS, since the susceptible version is still in use. Lastly, we determined that the updated TLS version most likely reduces the chance of an attack on the system. As the number of attacks in companies and countries that use TLS 1.3 has fewer attacks regarding the TLS-related protocols. However, some applications and protocols do not allow users to choose the TLS version to use. It means that the providers choose the version for the users to add to their application to implement security. This may explain why there is still a lot of the previous version of TLS that is being used.

References

1. “Final Project”, https://github.com/CY6740-Network-Security-Cormorants/Final_Project
2. Gincy Mol A G, “Importance of TLS 1.3: SSL and TLS Vulnerabilities, ”, 06 Jul 2020, https://beaglesecurity.com/blog/article/importance-of-tls-1-3-ssl-and-tls-vulnerabilities.html
3. CISA, “OpenSSL 'Heartbleed' vulnerability (CVE-2014-0160)”, April 08, 2014. Https://www.cisa.gov/uscert/ncas/alerts/TA14-098A
4. Borislav Kiprin, “What Is SSL LUCKY13 Attack and How to Prevent It from Happening”, Apr 3, 2021. Https://crashtest-security.com/prevent-ssl-lucky13/
5. Crashtest Security, “What Is LUCKY13 and How to Prevent It”, https://wiki.crashtest-security.com/prevent-ssl-lucky13
6. Chris Odogwu, “What Is the POODLE Attack and How Can You Prevent It?”, Nov 27, 2021. Https://www.makeuseof.com/what-is-the-poodle-attack/
7. Tomasz Andrzej Nidecki, “What Is the POODLE Attack?”, June 1, 2020. Https://www.acunetix.com/blog/web-security-zone/what-is-poodle-attack/
8. Rorot, “The breach attack”, Oct 7, 2013. Https://resources.infosecinstitute.com/topic/the-breach-attack/
9. Asif Balasinor, “SSL/TLS attacks: Part 3 – BREACH Attack”, Dec 16, 2013. Https://niiconsulting.com/checkmate/2013/12/ssltls-attacks-part-3-breach-attack/
10. “Tracking the FREAK Attack”, https://freakattack.com/
11. “MD5 signature algorithm support”, https://openvpn.net/faq/md5-signature-algorithm-support/
12. “TLS Basics. Internet Society”, https://www.internetsociety.org/deploy360/tls/basics/
13. “THE ULTIMATE GUIDE What is SSL, TLS and HTTPS”, https://www.websecurity.digicert.com/security-topics/what-is-ssl-tls-https
14. John Matherly, “shodan: The official Python library and CLI for Shodan”, https://github.com/achillean/shodan-python
15. Konstantin Stadler, “country converter”, https://github.com/konstantinstadler/country_converter