TLS Is The Best

An analysis of TLS configurations for the web's most popular commercial sites

Ben Greenberg, Spencer Howell, and Daniel Troutman

***Abstract*— Our team analyzed the TLS configuration of the 10,000 most popular commercial websites in order to better understand the modern state of web security. We scraped this data using the SSL Labs API and analyzed the results to look for security issues and best practices. Our results showed that the majority of sites are well-secured, with the most common vulnerabilities being a result of supporting older TLS versions. For instance, we found that the BEAST vulnerability still affects over 50% of modern web servers due to a theoretical exploit in TLS v1.0 or any SSL version. Overall, we found that most endpoints in the web are on the more secure side with very little of the web actually being vulnerable to well-known TLS exploits.**

I. OBJECTIVE

The Transport Layer Security (TLS) is a widely-used protocol for ensuring secure communications over the Internet. Specifically, it is the method that the HTTPS protocol uses to ensure confidentiality, integrity, privacy, and authenticity. The TLS protocol allows a client device and web server to agree upon a common method to securely generate a variety of cryptographic keys to use in their communication. This common method is referred to as a "ciphersuite," and there are a variety of cipher suites to choose from.

The TLS protocol has a long history of revisions, and it began life as the proprietary Secure Sockets Layer (SSL) protocol developed by Netscape. TLS 1.0 was released in 1999 as an open standard developed by the Internet Engineering Task Force (IETF). TLS has had multiple revisions over the years, with TLS 1.1 releasing in 2006, TLS 1.2 following in 2008, and the most modern TLS 1.3 launching in 2018 [1].

TLS has been continually updated to improve the security provided to applications utilizing it, as well as to fix bugs and exploits that might be able to circumvent the security measures implemented by TLS. Some high-profile attacks against TLS include Heartbleed, FREAK, Logjam, POODLE, and BEAST. These vulnerabilities make it vital that modern versions of TLS be utilized, to avoid data breaches or stolen information. It is important to note that both the user's web browser and the web server they are communicating with need to support updated versions of TLS to remain secure. In addition, some ciphersuites are more secure than others, and known vulnerabilities make some ciphersuites unsuitable for secure use.

Our team set out to analyze the current TLS configurations utilized by the most popular websites, including what versions and ciphersuites they support, to better understand the modern state of web security.

II. DATA COLLECTION

To investigate the current usage of the TLS protocol on the Internet, we needed a tool that gave us information about a website's certificates and protocols. After investigating various methods, we settled on the API provided by SSL Labs [2]. This API for their "SSL Server Test" allows us to query a variety of information about any public website, including the TLS version used for each endpoint, whether or not each endpoint is vulnerable to a variety of exploits based off of the version and ciphersuite used, and information about each certificate provided by the website.

We decided to target our analysis to the top 10,000 websites with a ".com" top-level domain. This targeted approach ensured we were comparing similar websites with each other and avoided any potential issues that might arise from testing government or nonprofit websites for vulnerabilities. We obtained the list of .com websites from magestic.com's "Majestic Million," a list of the websites ranked by their number of referring subnets [3]. We were able to filter the data on the site and download a CSV with domains of the top ranked commercial sites.

Our team then wrote Python scripts that tested each endpoint in each domain from the CSV file using the SSL Server Test API and saved the results to a local JSON file. However, some tests failed due to the servers no longer being active or are otherwise unreachable. At the end of all of our queries, we successfully tested 8,981 domains with a total of 24,177 endpoints.

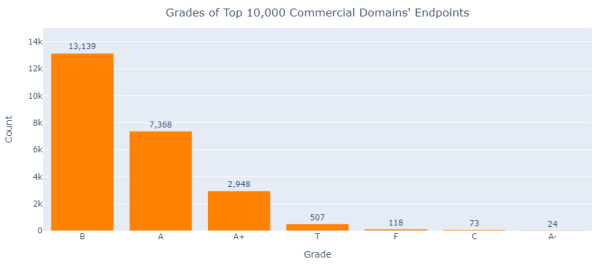
The file containing our test results was then parsed and inserted into a MongoDB database, with the relevant information for our analysis selected and reorganized. Using this database, we were able to run Python queries using PyMongo to aggregate data and generate graphs answering our research questions.

III. RESULTS

Using our scraped data, we were able to answer a variety of questions about the tested websites. These answers can give us insights into the current state of web security for highly trafficked websites, including what they are doing well and what they can improve at.

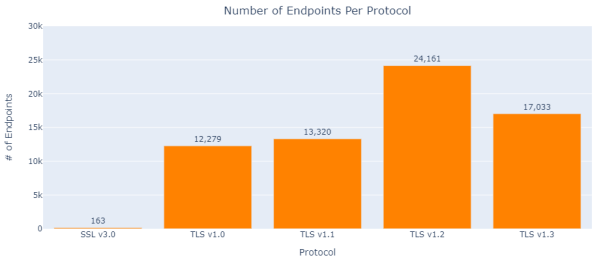
First, we wanted to answer the question "how secure are commercial websites?" The SSL Server Test tool reports a grade for each tested endpoint. This grade ranges from "A+" for the most secure to "F" for the least secure, with a value of "T" meaning the certificate is invalid and a value of "M" indicating a certificate name mismatch. The grade is based on many factors such as certificates, protocol support, key

exchange algorithms, and cipher strength. For example, if an endpoint still supports TLS 1.0 and TLS 1.1, then the grade is capped at a B. This might be why we see such a large number of 'B's where they could be higher if they dropped support for these deprecated protocols. Below is a graph showing the count of each grade for all endpoints queried on the top 10,000 websites.



In addition, the tests provided a grade ignoring the validity of the certificate. This can be useful for if a certificate is simply temporarily invalid and needs to be renewed. In our sampling, we did not see much change compared to the grades with valid certificates.

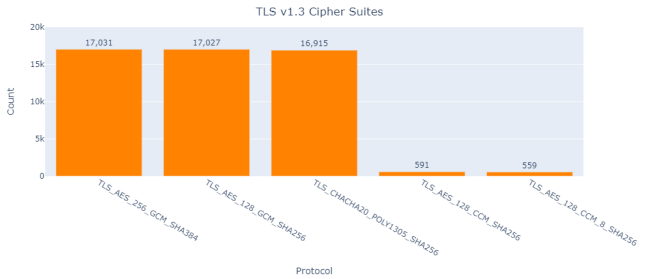
Part of the security rating of a site is which versions of TLS (or SSL) it currently supports. At the time of testing, TLS 1.2 and TLS 1.3 are the currently supported versions, with TLS 1.0 and TLS 1.1 deprecated in 2020. SSL 3.0 is a very outdated version, being deprecated in 2015.



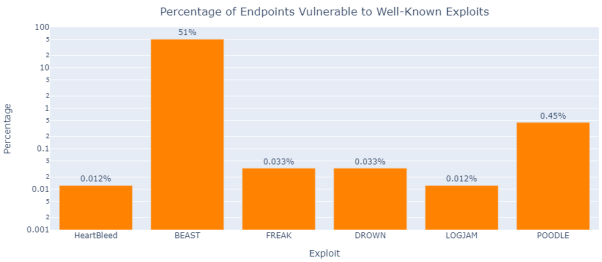
The results show a large number of websites still supporting TLS 1.0, TLS 1.1, and a few even support SSL 3.0. This might be in order to support users with older or unupdated devices, but this still results in security vulnerabilities as an attacker can request the server to use the older versions of TLS.

We also parsed information regarding the most commonly supported ciphersuites for TLS 1.3, one of the two currently supported TLS versions. We totalled up the number of servers that supported each ciphersuite, and present the results in the chart below.

We also looked at cipher suites for TLS 1.2, however there were 109 different configurations! While the two most supported suites were secure, the next 4 were all weak because they used an insecure encryption mode. In TLS 1.3, all of the insecure and outdated ciphersuites were removed which is why we only have data on five ciphersuites.



A big question we had going into this project was "What vulnerabilities are large commercial websites still vulnerable to?" Each endpoint was tested for these vulnerabilities based on the parameters of its TLS implementation, and the number of endpoints vulnerable to each exploit is listed below.



Thankfully, many of the older exploits are not a concern for most of the tested websites, with Heartbleed, FREAK, DROWN, and Logjam vulnerabilities appearing very infrequently. However, we were surprised at the large number of websites vulnerable to the BEAST attack. This vulnerability was patched in TLS 1.1, but all sites supporting TLS 1.0 are still vulnerable. This is a theoretical attack, meaning no practical threat is known, which might also be responsible for the slow response by server administrators.

Finally, our results allowed us to analyze the root certificates used by each tested endpoint. These Root CAs are at the end of the certificate chains presented by each server. Each operating system and browser chooses which Root CAs to trust, and which to ignore. We found a total of 151 unique root certificates with Baltimore CyberTrust Root (DigiCert) being the most common one with 2,142 copies. The second most present root certificate was ISRG Root X1 (Let's Encrypt) with 1,888 copies. Another Let's Encrypt (DST Root CA X3) root certificate was the third most popular with 1,748 entries. This certificate actually expired in September 2021 so it is interesting to see it still present a few months later.

Overall, we found that the majority of the surveyed websites were deploying adequate protections against the web's most common threats. The most common choices that weaken security are supporting older versions of TLS and SSL, which resulted in BEAST being the most pressing vulnerability to the modern web. The servers tended to prefer and support larger key sizes in the ciphersuites, and the selected ciphersuites were mostly secure, especially in TLS 1.3.

IV. ISSUES

Data collection posed a significant challenge to our team. Since each SSL Server Test run takes around a minute before a result is received, we had to collect this data over the period of several weeks. We were challenged by the limitations set by the SSL Labs API, which would only allow a certain number of requests at any given time. Our team split up the task of collecting data, allowing each member to request independently. These separate data files were then combined to produce the final result.

Another problem our team ran into was the storage of the data. The compiled JSON file had a final size of over 1.2 GB. Our team needed a way to share and run queries on this data at the same time, while continuing to update it with new data as we received it. Cloud providers did not provide enough free storage for this amount of data, so we ended up hosting the MongoDB server on a Raspberry Pi microcomputer connected to the internet. This allowed all members of our team to access and update the shared data in real time.

V. FUTURE WORK

This data allowed us to learn a lot about the current state of web security and where it can be improved. This type of data scraping could be useful for cybersecurity experts to focus their efforts on fixing vulnerabilities that are most commonly present in the web today. It is also useful for web developers and server administrators to ensure that their websites are matching the standards set by the rest of the web.

For further research in this area, our team would recommend utilizing a data scraping tool that does not throttle like SSL Labs API does. While the analysis performed by the test made our investigation easier, it added significant effort to the data collection process. If repeated, we would gather information using a simpler tool that can be executed locally instead, such as OpenSSL.

In addition, we would like to expand our data collection beyond the most popular commercial sites. Testing less-visited sites and expanding to other top-level domains would give us a more accurate picture of the web security as a whole. While we limited the scope of our project for accuracy and simplicity, further analysis of the Internet might provide more conclusive insights.

VI. TIMELINE

Our team decided to change topics to TLS pretty late into the semester, and as a result our schedule did not have much room to spare. We divided our work into individual tasks, and worked on these together during each period. The rough timeline of our team is presented below:

A. October 17th - October 24nd

It was this week that our team decided to change to a more interesting and relevant topic than our original one. We investigated the topic of TLS, crafted a new proposal concept, and presented it to Dr. Mockus.

B. October 24th - November 6th

After being approved for the topic change, our team began investigating data scraping tools and APIs that we could use to gather the data we required. We considered multiple options and tested them out before deciding on the SSL Labs API.

C. November 7th - November 20th

During this period, our team ran scripts to scrape the data from the web and store the JSON files. Since each test took a minute or longer, we allocated plenty of time to ensure we were able to collect all the data we planned on.

D. November 21st - December 4th

After collecting the raw data, our team had to transform it into a useful format for analysis and report generation. We combined all of our results by adding them to a MongoDB instance, and began writing aggregations that provided answers to our research questions.

E. December 5th - December 8th

During the last week of the project, our team focused on writing the final report, creating a slideshow, and practicing our presentation. We also finalized our data and aggregations.

F. December 9th

Present final presentation and submit final report.

VII. MEMBER RESPONSIBILITIES

Our team worked as a group to plan out tasks and delegate responsibility, and then worked together to accomplish each task.Specifically, our whole team worked on researching the TLS topic and finding potential sources of data. After we decided to use the SSL Labs API, our whole team worked together to gather the data.

However, each team member focused more on some specialized tasks. Daniel and Ben took the lead on formatting the data, parsing it into a usable form, and entering it into a MongoDB database. Ben did most of the setup on the database itself. Daniel worked on writing aggregations using PyMongo on the dataset, and Ben worked on creating graphs in Plotly using the results. Spencer focused on creating research questions to answer, writing the final report, and preparing the slides. Our team then worked together to polish and practice the presentation.

REFERENCES

[1] "The Evolution of SSL and TLS | DigiCert.com." https://www.digicert.com/blog/evolution-of-ssl (accessed Dec. 08, 2021).

[2] "SSL Server Test (Powered by Qualys SSL Labs)." https://www.ssllabs.com/ssltest/index.html (accessed Dec. 08, 2021).

[3] "Majestic Million" https://majestic.com/reports/majestic-million (accessed Dec. 08, 2021).

[4] "Server Technologies - HTTPS BEAST Attack," Context Information Security. https://www.contextis.com/en/blog/server-technologies-https-beast-attack (accessed Dec. 08, 2021).