Exploring Common Website Vulnerabilities

CS 6262/ECE 6612 Network Security Final Project Report
Georgia Tech Spring 2021

Collin Avidano
Computer Science
cavidano3@gatech.edu

Joshua Dierberger
Computer Science
jdierberger3@gatech.edu

Abigail Drun
Electrical and Computer Engineering
adrun3@gatech.edu

Eric Hsieh
Computer Science
hsieh.eric@gatech.edu

Tara Poteat
Electrical and Computer Engineering
tpoteat3@gatech.edu

Abstract—**In this paper, we present the findings of our Network Security final project for Spring 2021. We want to gain insight on a wide range of vulnerabilities over a large number of sites to see if there are any common patterns or trends across popular websites. For the project, we conducted a study on website vulnerabilities across 10,000 websites randomly selected from a list of the 1 million sites from 2014.**

To determine vulnerabilities, we scanned for the following: DNS records, open ports, cipher suites, certificates, traceroutes, and forms and templates. After gathering this data, we analyzed it to extract tangible security issues from these websites. This included determining how many websites have IPv6 addresses, which ports are open and susceptible to attacks, which websites accept insecure cipher suites, which websites have expired or invalid certificates, which IP addresses serve as bottle necks in the traceroute, and if any forms and templates that have known bugs are used on a site. From this analysis and the results we obtained, we were able to abstract our findings and draw conclusions about the Internet as a whole.

I. Introduction

Website security is an important aspect to keep in mind when it comes to network and Internet security. It has many impacts on users, as vulnerabilities can cause sensitive information to be leaked, malicious software to be downloaded, service to be denied, and many other issues. In many cases, users assume that websites they visit are secure; however, this is not always the case. There are many websites that are not up-to-date and have security flaws, and so in this project we decided to explore how well different websites are designed, maintained, and managed. We looked at the most popular websites to determine the common vulnerabilities that these websites might be susceptible to.

In order to accomplish this, we scanned 10,000 randomly selected websites from a top 1 million list to search for web vulnerabilities. Our scanner searched for DNS records, open ports, ciphersuites, certificates, traceroutes, and forms and templates. From this information, we were able to determine websites that had out-of-date DNS records, open ports that can be used for malicious communication, cipher suites that are insecure, as well as outdated certificates and vulnerable forms and templates.

The GitHub respository linked to this project can be found at https://github.com/CollinAvidano/cs6262-research-project.

II. Related Work & Background

Although website security is such an important concept, there have not been many studies conducted on a wide range of vulnerabilities over a large number of sites. The most common studies focus on one aspect of security and go in-depth regarding that vulnerability. Collecting information on many different vulnerabilities is important because it can provide valuable insight of the overall Internet and how websites are being configured as far as security is concerned.

In past research, there have been studies conducted on common web-service vulnerabilities. There is one research experiment in particular that we took a closer look at before beginning our project, since there were some similarities between our goals [1]. The scale of this project was limited to 2,000 web services. The project primarily focused on classifying vulnerability types into static and dynamic, and changeable and unchangeable. Their study examined attack-points centered on WSDL (web service definition language) files. This examination is conducted through parsing WSDL files associated with each web service. The researchers assessed confidentiality and integrity issues — which they classified to be static vulnerabilities — by seeing if encryption and signature policies were defined. Other static vulnerabilities looked into include logging, invalid XML, invalid parser, and password in clear. Dynamic vulnerabilities examined were error on interface, SQL and XPATH injection, and session replay.

For this project, we will explore a group of different vulnerabilities to determine how secure websites are and whether or not there are any common security vulnerabilities between them.

III. Approach

In order to find website vulnerabilities, we used Python to write code to check for resources that could result in security flaws. We used Python libraries and methods for the core of this in addition to multithreading to speed up execution. In order to decide the list of websites that we analyzed, we used a script to randomly select 10,000 of the top websites from a 2014 list of top 1 million websites.

Once the website domain names were selected, we first used a DNS resolver to gather a list of IP addresses associated with each domain. This step must come first because some vulnerabilities we wanted to search, including traceroutes and open ports, are associated with IP addresses rather than domains themselves. We ran the IP-dependent scripts for these vulnerabilities and used the target domain to check cipher suites, certificates, and forms and templates. The data that was collected through these scripts was then written to a database so that it can be analyzed in future steps.

Due to the large-scale size of our project, we chose to run our program on a Dell R710 server. The program took a while to fully execute, which presented a major challenge for testing our code in a timely manner. We decided to use multithreading so that multiple domains could be scanned simultaneously, therefore speeding up the process of scanning 10,000 websites. With this design, race conditions were present because each thread had to commit to the mySQL database individually. To overcome this, we designed a custom multithreaded engine to deal with the asynchronous commits to the database.

We then ran this program on eight threads – one per each CPU core on the server – to speed up collection. Once the thread ran, the data was saved to the database. Our database contained ten different tables and was organized and separated in a way that we could easily extract information for analysis.

Each one of our threads ran the entirety of our pipeline, which can be seen in the flowchart of our work in Figure 1.

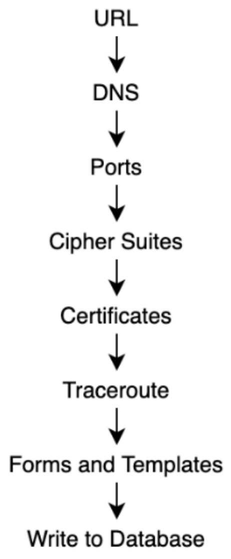

Fig. 1: Individual thread pipeline.

A. Libraries and Methods

For our program, we used many network-related Python libraries to assist in scanning.

1) Openssl: Openssl is the de facto standard for performing operations related to the secure sockets layer (SSL) and its successor, transport layer security (TLS). Openssl was used to both retrieve certificates and validate them against their owners for all of the tested websites.

2) Dnspython: Dnspython is a Python library which simplifies the process of retrieving DNS results, allowing this to be done entirely in Python. Dnspython was used to retrieve the DNS results providing the mapping from our domains to IPv4 addresses.

3) Nmap: Nmap was used to scan the open ports of every target IP that was returned as part of the dnspython result.

4) Scapy: Scapy was used to run the traceroute to a specific IP from Python.

5) Scrapy: Scrapy was used to search the contents of pages from the list of domains and the linked pages for forms such as search fields and logins.

6) Mozilla Cipher Suite: The Mozilla cipher suite is a set of bash scripts for repeatedly querying a server to get all available ciphers through OpenSSL. This was used to get the list of all available ciphers for a server before checking the set against a list of known secure ciphers.

B. Dataset

The dataset was taken from a list of top 1 million sites of 2014. More recent top 100,000 site lists were locked behind a paywall, so we opted for the free 2014 dataset, compiled by Mozilla. Of these 1 million sites, we took a random subset of 10,000 for our project. The dataset can be found at this GitHub link [2].

C. Multithreading

The time taken to fully analyze a website varies significantly. Some websites can be analyzed in less than five seconds, while other websites may take closer to one minute to fully analyze. We decided to multithread our program, launching one full analysis per thread as this was more straightforward than dividing up each full analysis into individual component tests.

Default Python multithreading queues and asyncio did not work with this program, as the database we used was not threadsafe. We could not use asyncio as most of our functions were written synchronously. We custom coded a program which deployed at most eight futures at a time and waited for them to complete. They would then be synchronously committed to the database, in the order which they finished, from the main thread.

D. SQL

As the multithreaded code is being run on the server, the results are written to a database that is hosted on the same server. The reason we decided to use a database is so that the raw data can exist in a single location and then manipulated when completing the analysis. With a database, counts and values can easily be retrieved and manipulated using SQL statements. This allows for us to find patterns and trends in the dataset. Additionally, writing out to a CSV or another local storage file would take a significantly longer time to organize, sort, and gather when analyzing the data.

IV. Findings & Analysis

We obtained the following results from searching for the aforementioned vulnerabilities. The data presented in this section allowed us to see patterns and trends, as well as pinpoint areas where security issues exist and determine approximately how prominent these are across the Internet. Our findings are summarized in Table I.

TABLE I: Summary of Data Findings.

Website Categories	Count
Total Websites Crawled	10000
DNS Records – IPv4	8223
DNS Records – IPv6	1763
Ciphers	7542
Certificates	5003
Traceroutes	8222
Forms	2007

A. DNS Records

In crawling approximately 10,000 websites, a total of 8,223 websites returned valid DNS records. Out of these websites, all of them had valid IPv4 records, whereas only 1,763 (21.4%) of them had valid IPv6 records as well.

Another intention of examining DNS records was to examine the failover options for each website. More failover options is generally better as it removes a central point of command that attacks, such as denial-of-service, could affect. We found that the sites analyzed had an average number of 1.768 failover options, with the smallest number of failover options being 1 and the highest being 16. We also found that most websites had only one failover; however, the second most common number of failovers was four. A summary of the spread of failover options is shown in Table II.

However, as mentioned earlier, the dataset utilized was from 2014, which explains why 1,777 websites did not have DNS records, since some of these websites may not exist anymore. While these websites are common, due to the method through which they were sampled (randomly selecting 10,000 sites from the original dataset), many of these sites may not actually be popular enough to justify keeping them up. This lack of maintenance could also contribute to the low average number of failover options. This is an important realization that should be kept in mind when examining our project's results.

B. Open Ports

The main reason for any port to be open is to allow for communication between applications and services. Otherwise, it is generally good practice to close ports to reduce attack surfaces. Even with secured open ports, there are always opportunities for attack. In general, closing all possible ports would be beneficial for lowering the likelihood of any attacks. We used Nmap to search for any open ports associated with the IP addresses of the various websites. There were 1,000 different port

TABLE II: Spread of failover options.

Failover	# Websites
1	5849
2	701
3	103
4	1281
5	10
6	246
7	2
8	11
9	1
10	9
11	1
12	7
16	2

numbers across all IP addresses found to be open. There were too many open ports to check individually, so we opted to look into ports that had more than 1,000 instances of being open on unique IP addresses. These ports and their respective instances and port name are shown in Table III.

TABLE III: List of open ports with over 1,000 instances.

Port	Instances	Name
80	10822	HTTP
443	10657	HTTPS
8443	4224	SSL
8080	3924	HTTP
21	3537	FTP Control
25	3262	SMTP
22	2945 (20%)	SSH
110	2927	POP3
995	2919	POP3S
587	2913	SMTP
143	2900	IMAP
993	2856	IMAPS
465	2791	SMTPS
53	2595 (17%)	DNS
3306	2158 (14%)	MySQL
20	1101	FTP Data Transfer

As mentioned earlier, there are some ports that should be left open since applications need to be able to communicate through them. These ports include http, https, and SSL. The top four most commonly left-open ports (80, 443, 8443, and 8080) correspond to http, https, and SSL. This is a good sign as far as vulnerabilities are concerned, because these ports are less of a vulnerability. These ports account for 47.4% of ports left open.

The other 52.6%, however, correspond to ports that, for security purposes, should be kept closed. A breakdown of these ports is shown in Figure 2. While most of the open ports corresponded to mail servers, there were a significant number of File Transfer Protocol (FTP), SQL, DNS, and SSH.

A challenge we ran into with the open ports is that port scanning only worked for IPv4 addresses. With the IPv6 addresses, there were issues with the arguments field for the module used. Although the python-nmap package includes a flag for IPv6, when running the code errors still occurred. This is likely due to the code being outdated, as it has not been updated since 2018 [3].

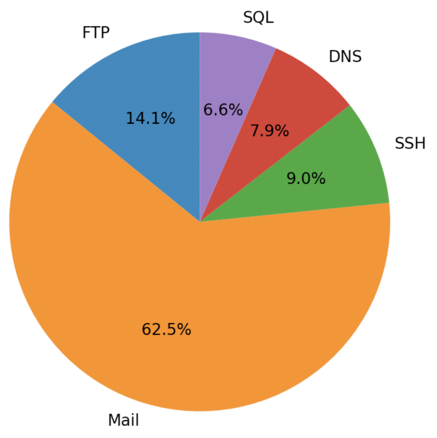

Fig. 2: Breakdown of ports normally closed that were found to be open.

C. Cipher Suites

Cipher suites are sets of algorithms used for securing TLS network connections. In order to ensure the security and integrity of both client and server, each website supports several different cipher suites that include algorithms such as key exchange, bulk encryption, and message authentication codes.

Of the 8,823 websites that returned DNS records, 7,542 websites had cipher suites; from these cipher suites, we identified 82 unique sets. 680 valid websites did not return cipher suites, indicating that they do not have encryption policies and thus lacked confidentiality. There were over 90,000 total cipher suites used. The most common cipher suites were also very secure, using Elliptic-Curve Diffie-Hellman, with RSA cryptography, AES encryption, Galois/Counter Mode, and variations of SHA.

Most websites supported multiple ciphers, with one website supporting 51 distinct ones. About 75% of the websites scanned had at least one secure cipher, meaning they used Diffie-Hellman, RSA, AES-128 bit or higher, and SHA-256 bit or higher. The most popular cipher suites used were combinations of ECDHE, RSA, AES, GCM, and SHA. Here, AES was either 128 or 256 bit, while SHA was either 256 or 384 bit:

- ECDHE-RSA-AES128-GCM-SHA256 (6050 instances)
- ECDHE-RSA-AES256-GCM-SHA384 (6010 instances)

3% of websites had very unsecured ciphers, which contained algorithms like RC, ADH, MD5, DES, and SHA. One important note here is that the most updated RFC we checked stated that SHA meant SHA-1, which is unsecure, as opposed to SHA256, which is secure.

Additionally, 1,582 websites supported post-quantum secure cipher suites, which was about 18% of all websites. In order for a cipher suite to be post-quantum secure, its encrypting algorithms had to have greater than 128-bit encryption. The most popular one was a combination of Elliptic-Curve Diffie-Hellman, Elliptic Curve Digital Signature Algorithm, AES256, GCM, and SHA384 (ECDHE-RSA-AES256-GCM-SHA384).

Because our list of websites is from 2014, some of them may not have kept their cipher suites updated. This would explain why some websites still support MD5 and similar unsecured algorithms.

D. *Certificates*

Security certificates establish the identity and authenticity of a website. Out of the valid websites scanned, 3,220 (40%) did not have valid certificates. Without these certificates, users do not have a means of verifying the authenticity of public keys and the validity of the site. Of the 5,003 sites (60%) with certificates, many did not contain additional information such as organization, country, and location.

Earlier in the development of this certificate parsing code, there were errors with some websites tested since the code wanted to explicitly extract fields like the country of the certificate instead of making that an option. Most of the websites that caused errors in these cases were websites with country codes outside of the United States. Thus, it is likely that websites from other countries contain different information from those in the United States, or omit this information altogether. This would explain why the errors occurred.

The tool used to conduct this certificate parsing – Python SSL – did not specify why certificates were unable to be retrieved, but it is highly likely that it was due to the dataset being from 2014: a lot of the certificates have since expired for websites that fallen out of maintenance, which would explain why there were more expired or nonexistent certificates than expected. Most certificates last 2-3 years, so the 7-year gap would give plenty of time for these to expire.

E. *Traceroute*

We ran a traceroute from our host server to each of the IPs in our dataset in a one-to-many mapping. The results are shown in the following figure.

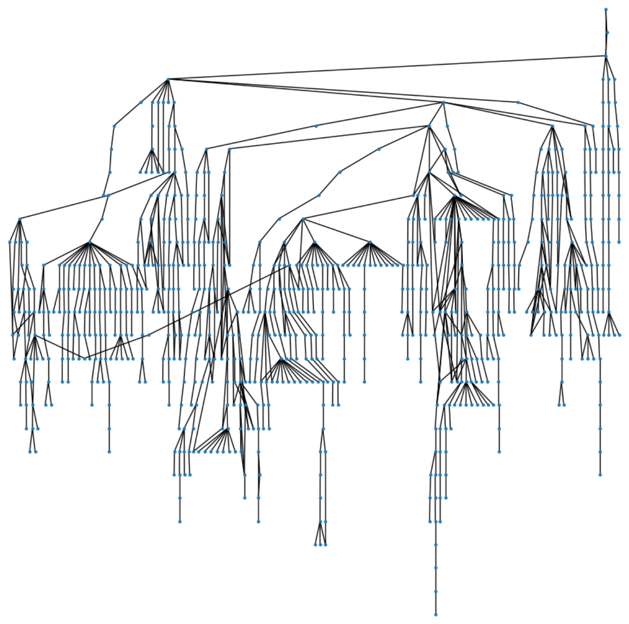

Fig. 3: A sample representation image of our traceroutes with 10,000 hops.

Figure 3 visualizes a representative sample of our traceroute results. Since this was run from a personal server (top right of Figure 3), many of the early bottleneck nodes are to be expected; home users such as ourselves must use ISPs to route to the broader internet.

Bottleneck nodes later on in the graph represent nodes that service main sites. Such bottleneck nodes represent a major threat to the availability of sites, as denial-of-service attacks on such nodes would affect a large number of sites simultaneously.

In addition to the previous findings, we also found that of the approximately 10,000 sites scanned, only one did not respond to ICMP requests, while the rest did.

This is strange as, in general, ICMP is typically blocked because it is commonly used as the protocol of choice for DDOS attacks.

F. Forms & Templates

Many websites have input forms and, additionally, use templates for website formatting. Input forms can cause security issues because, depending on the sanitation techniques used, an improper setup can allow for SQL injections or cross site scripting. In the future, a more in-depth analysis should be completed to see which of these forms are insecure.

We searched for some of the common templates used on websites which includes WordPress, Wix, GoDaddy, Weebly, Squarespace, and Shopify. A total of 2,007 templates were found across all the websites crawled. One of the most common template vulnerabilities returned included Wordpress. Wordpress provides a list of security issues and significant vulnerabilities found for each of the different versions. GoDaddy was the only other template that gave version numbers. Other templates did not use the standard meta tag to indicate the generator.

We searched the National Vulnerabilities Database provided by NIST for common website vulnerabilities using CVSS Version 3. Of these, WordPress has the most extensive list of vulnerabilities, both for common plugins and WordPress itself. To determine if these vulnerabilities were actually present requires a more in-depth analysis of the specific themes and plugins used, which is difficult if not impossible to do programmatically. However, we can identify the number of websites which are possibly vulnerable, and look for commonly used plugins and themes which are likely to be present.

We picked five vulnerabilities which we believed to be both common and severe and counted the number of WordPress sites which had lower versions than the ones we found:

- CVE-2020-29045: Version 2.2.0: The food-and-drink-menu plugin suffers from an arbitrary code execution bug. Number of vulnerable versions found: 1.
- CVE-2021-24222: Version 6.3: The WP-Curriculo Vitae Free WordPress plugin suffers from an arbitrary file upload bug. Number of vulnerable versions found: 1513 (all WordPress sites).
- CVE-2021-24175: Version 4.17: The Plus Addons for Elementor Page Builder WordPress plugin allows standard authentication procedures to be bypassed. Number of vulnerable versions found: 7.
- CVE-2015-9435: Version 3.1.5: The oauth2-provider plugin has incorrect random number generation. Number of vulnerable versions found: 1.
- CVE-2017-18634: Version 6.7.2: The newspaper theme suffers a script injection vulnerability. Number of vulnerable versions found: 1513 (all WordPress sites).

As stated above, we cannot be sure these websites necessarily used these plugins. However, we also believe the selected vulnerable plugins are commonly used, and given that this is a random sample of only 10% of the total population, it is possible that there are other out-of-date sites in this sample set which have failed to update their WordPress templates and thus could still be susceptible to these vulnerabilities.

G. Fuzzing SSH Ports for Weak Passwords

As upwards of 20% of our websites had open SSH ports, we decided it would be worthwhile to test weak username and password combinations. We wrote a simple Python script using paramiko [4] and pythonping [5] which attempted an SSH connection to all websites which had port 22 open and tried as many username and password combinations as possible before the connection was closed. We spaced the attempts reasonably so as to not cause any sort of overloading of the server. We also did not perform any actions on the remote server and simply sent this data and awaited a response so as to not cause any adverse effect on the server.

We tested the username and password combinations given in Table IV. We tested two usernames and four passwords. No websites were susceptible to these combinations tried.

TABLE IV: Usernames and Passwords attempted for SSH credentials

Username	Password
root	ubuntu
-	admin
ubuntu	root
-	password

H. Guessing OSes

The nmap program has a powerful feature which allows it to guess a remote server's operating system (OS) heuristically. This can be used to probe for servers with an OS that has known vulnerabilities and could be susceptible to a remote attack. Therefore, we determined it would be worth analyzing the OSes of these remote servers to identify if they were susceptible to known vulnerabilities. A summary of the major OS groups found is given in Table V. Although these are only heuristic guesses, this gives us the possibility to analyze these servers with some more depth.

TABLE V: Major OS Groups Scanned

OS Group	Number Found
No scan	603
Misc. embedded	719
Linux 4.X	1151
Linux 3.X	2224
Linux 2.6.X	4801
Linux other versions	31
macOS/OS X 10	5
Misc. Windows	343
Misc. iOS	5
2-Series	608
Android 5 and up	109
OpenBSD other	247
FreeBSD 6 and up	146
Misc. Other	26

Out of all the IP addresses scanned, 26 IP addresses ran some form of macOS or OS X, or Windows 7, Vista, or Windows 8.1, which are primarily home-operating systems. One IP address ran Symbian OS, which is a deprecated mobile device OS. Two IP addresses also ran HiveOS, which is a cryptocurrency mining rig OS. Additionally, five IP addresses were running some version of Apple's iOS.

Lastly, although we could not perform this part of the analysis for all OSes we scanned, we were able to inspect some OS versions through the National Vulnerabilities Database provided by NIST to identify vulnerabilities in certain OS software, especially deprecated OSes. We searched only for vulnerabilities which applied to the general operating system kernel, not just software running on it, were executable over a network, and which had a critical impact. We present some of the more significant results found below:

- Multiple CVEs, e.g. CVE-2021-24074: Windows 7 suffers from a TCP/IP remote code execution vulnerability. This affects 11 servers.
- CVE-2016-3236: Windows Search for some Windows 7 versions, most Windows Vista versions, and most Windows 2003, 2008, 2012, and 2016 versions allows remote code execution. This affects approximately 343 servers.
- CVE-2020-9866: A buffer overflow in CoreAudio in macOS versions 13 and lower allows arbitrary code execution. This affects at least two servers.
- CVE-2011-3188: Linux Kernel versions before Linux 3.1 use MD4 (an insecure hash function) to generate sequence numbers, allowing attackers to hijack communications. This affects at least 4,831 servers.

I. Betweenness Centralities of Traceroutes

As noted earlier, nearly every single IP address that hosted one of the domains in question responded to ICMP. However, the results of our Betweenness Centrality analysis yielded some interesting results. The Betweenness Centrality (BC) metric measures how central a particular node is to a network through how many other nodes must rely on that node for pairwise communication. Table VI shows the top five records for betweenness centrality measurements. Unsurprisingly, most of the nodes that had the highest BC scores were those closest to the server from which we ran the analysis. Note that tcore is part of all domains for TATA communications, which is Google Fiber's upstream network provider. However, slightly down the list, the names become much more recognizable, as seen in Table VII.

Additionally, manual inspection of the paths between such nodes and their traceroute records indicated many IP "skips". That is, the nodes that connected them did not respond to ICMP requests. This is beneficial for security purposes as the nodes that bridge nodes with a high BC score represent critical chokepoints with regards to accessing these sites. Thus, for security purposes, the nodes that connected them should not respond to probing.

V. Discussion & Future Work

In our analysis, we found that only 8,223 websites returned a valid IPv4 DNS record with 1,763 additionally returning an IPv6 address. Given the fact that only 21.4% of websites have IPv6, our findings seem to indicate that the Internet, in its present state, is not prepared for a large-scale transition towards IPv6.

For ports, our findings show that 52.6% of open ports present security vulnerabilities. This is dangerous because leaving ports open and accessible can lead to a wide range of security implications including remote control of the server, unauthorized access to files, and infections of other computers on the network.

TABLE VI: Top Five Records for Betweenness Centrality

IPv4 Address	Reverse DNS Result	BC Score
64.86.113.186	ix-ae-60-0.tcore1.a56-atlanta.as6453.net	0.6442133566497867
192.119.18.105		0.5730795248877394
23.255.224.39	23-255-224-39.mci.googlefiber.net	0.37983277109429897
23.255.225.180	23-255-225-180.googlefiber.net	0.30071594002900826
216.6.57.5	if-ae-1-3.tcore3.njy-newark.as6453.net	0.22747309309887437

TABLE VII: Known Cloud Provider Nodes

IPV4 Address	Reverse DNS Result	BC Score
66.198.154.24	if-ae-21-2.tcore1.aeq-ashburn.as6453.net	0.0482559750439312
206.126.236.17	8057.microsoft.com	0.015395414005842784

When analyzing cipher suites, we found that approximately 3% of websites contained ciphers that were very unsecure. This can pose a problem since man-in-the-middle attacks can intercept information if a weak cipher is chosen. However, more importantly, 75% of websites provided at least one secure cipher so the client has the option of choosing a secure cipher suite.

Over 40% of sites had expired or invalid certificates. Although this is most likely due to the websites no longer being maintained or used, it is still an important aspect to note. These invalid certificates mean that users cannot establish a secure connection with the server or verify the authenticity of the site. Therefore, attackers can easily impersonate these websites.

With forms and templates, we gathered some data on this, but were not able to explore related vulnerabilities.

As mentioned before, the dominant factor contributing to our results is the chosen dataset. The dataset used consisted of the top websites from 2014, while the querying of these sites was completed more recently in 2021. With this difference, there is a 7-year gap that needs to be accounted for when analyzing the results. Some of the websites we analyzed may no longer exist or are no longer maintained. This can explain the relatively high proportion of missing DNS records, invalid certificates, weak cipher suites, and other unusual findings in our results. Given more time and resources, it would be beneficial to run this experiment on a more current list of popular websites as well as a larger sample of websites.

Some additional vulnerabilities that should be inspected more in our future work would include looking at SQL injection and cross site scripting for the forms. This would involve using the forms found in this experiment and then attempting to insert commands into the input. Another improvement on our current data collection would be to analyze the IPv6 addresses more in-depth. Currently, we only scanned the IPv4 address for ports, certs, ciphers and other items; ideally, IPv6 addresses should also be included in our scanning.

VI. Conclusion

Our project examines common website vulnerabilities across 10,000 websites. Throughout the project, we were able to find results on DNS records, open ports, cipher suites, certificates, traceroutes, and forms and templates. Although the dataset utilized in our study may have contributed to elements such as DNS records and missing certificates, these records that were found are reflective of the current 2021 status. The code produced by this study and the data collected in this project have set the premises for future work to be conducted on both further exploring web vulnerabilities as well as for these websites to take steps towards addressing these vulnerabilities.

VII. References

[1] Sushama Karumanchi and A. Squicciarini. “A Large Scale Study of Web Service Vulnerabilities”. In:J. Internet Serv.Inf. Secur.5 (2015), pp. 53–69.

[2] Kario, H (2014) top-1m.csv (Version 2.0) [Source Code]. https://github.com/mozilla/cipherscan/blob/master/top1m/top-1m.csv.

[3] Liu, D (2018) python-nmap (Version 135.0) [Source Code]. https://github.com/liudonghua123/python-nmap.

[4] Forcier, J (2020) paramiko (Version 2.7.2) [Source Code]. https://www.paramiko.org/.

[5] Maggio, A (2021) pythonping (Version 1.0.16) [Source Code]. https://pypi.org/project/pythonping/.