Q0. Collaborators: Robert Liu for debugging help in Part 4

Sources:

https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/#cryptography.hazmat.primitives.asymmetric.rsa.RSAPublicNumbers.e, https://docs.python.org/3/library/csv.html, https://github.com/pyca/pyopenssl/blob/main/src/OpenSSL/crypto.py#L106, https://crypto.stackexchange.com/questions/3110/impacts-of-not-using-rsa-exponent-of-65537, https://linux.die.net/man/1/wget, https://medium.com/perimeterx/user-agent-based-attacks-are-a-low-key-risk-that-shouldnt-be-overlooked-1a5d487a1aa0, https://stackoverflow.com/questions/21965484/timeout-for-python-requests-get-entire-response, https://www.browserstack.com/guide/get-current-url-in-selenium-and-python, https://deviceatlas.com/blog/list-of-user-agent-strings, https://stackoverflow.com/questions/7108080/get-the-first-character-of-the-first-string-in-a-list, https://selenium-python.readthedocs.io/waits.html#implicit-waits, https://medium.com/pipedrive-engineering/socket-timeout-an-important-but-not-simple-issue-with-python-4bb3c58386b4, https://developer.mozilla.org/en-US/docs/Web/HTTP/Status#successful_responses, https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/#cryptography.hazmat.primitives.asymmetric.rsa.RSAPublicKey

Q1. 1) index.html 2) 18279 bytes

Q2. 1) 4 files 2) This gets everything needed to display the webpage locally including all images, etc., but only those that are contained within the page (does not follow externally linked docs)

Source: https://linux.die.net/man/1/wget

Q3. In the first wget call that produced index.html, we can see this text at the beginning of the file, after the content tag

Q4. A window opens displaying the image of the Google browser home page, but just for a few seconds before disappearing

Q5. Prints out the rendered contents of the Google webpage page onto the terminal, and does not launch a browser window with image display

Q6. Request URI is served over HTTP (http://chicagoreader.com/) and final destination is HTTPS (https://chicagoreader.com/)

Request URL does not have an ending slash (http://washington.edu) whereas final destination does (http://www.washington.edu/), but similar otherwise (in terms of HTTP vs HTTPS

Q7. The first method of accessing http://uchicago.edu results in an HTTP status code of 403, meaning that we are forbidden from accessing the non-secure version of UChicago's website and we are redirected to https://uchicago.edu (secure).

The second method of accessing http://uchicago.edu results in an HTTP status code of 200, meaning that we were successful in accessing the non-secure version of UChicago's website through passing a full set of headers along with a user-agent string to more accurately simulate a request from Chrome, and as a result, bypassing the user-agent based blocking scripts utilized by the UChicago website. The resulting final URL is http://uchicago.edu, which means we circumvented the automatic redirect to https://uchicago.edu.

Q8. When trying to rewrite the code accessing itturnsoutgrantdoesnotlikehotdogs.fyi to be of a form more similar to the earlier examples (i.e. without the try-except block) an long Connection Error message is printed on the Terminal, and the subsequent code block, which is supposed to attempt to access squid-cache.org, is not run. This is consistent with the linked website's conclusion that "Without proper exception handling, the program might abruptly terminate, leaving the user without feedback or results." In this case we received feedback, though it is congested and clunky, but the program did terminate. Instead, when using the try-except block, we are able to print out a concise and easily understandable error message, and keep the program still running so that the subsequent code can still run regardless of the failures that occurred earlier.

Q9. Disclaimer: I did not use status codes when comparing HTTP access vs HTTPS access, as I deemed an accessible website as something that can be reached through an HTTP -> HTTPS redirect, as the user can still access the website in its full capacity, without any compromises in experience or content. An 'available' website for me was a site that I could load, either through HTTP or HTTPS, before the 10 second timeout was activated. This is a good definition of available in my opinion as a website that takes longer to load with good quality internet connection, like the school WiFi, would take even longer to load in areas with poorer internet, thus impacting universal availability. Thus, this threshold was determined to be valid.

Requests	Top Sites	Other Sites
HTTPS only	57.1%	57.0%
Both	11.7%	17.0%
HTTP only	4.3%	11.6%
Neither	27%	14.3%

The state of encryption on the web is relatively secure as the majority of sites across both lists are either accessible through both HTTP and HTTPS or just HTTPS. Only a small percentage of the Top Sites use only HTTP, and the substantial amount of sites that use neither could be due to expired certificates or due to timing out, as some of the sites call servers in foreign countries.

Q10.

User-agent	Top Sites	Other Sites
HTTPS only	54.9%	57.9%
Both	13.3%	16.2%
HTTP only	5.1%	11.9%
Neither	26.7%	13.9%

The results are not significantly different in my opinion which could indicate that popular websites, who are more prone to cybersecurity attacks, have accounted for users trying to bypass access through using user-agent strings and have patched that security issue. We were able to access a small percentage of sites now through both HTTP and HTTPS, especially from the Top Sites list, which proves that there are still some websites where this bypass is possible.

Q11.

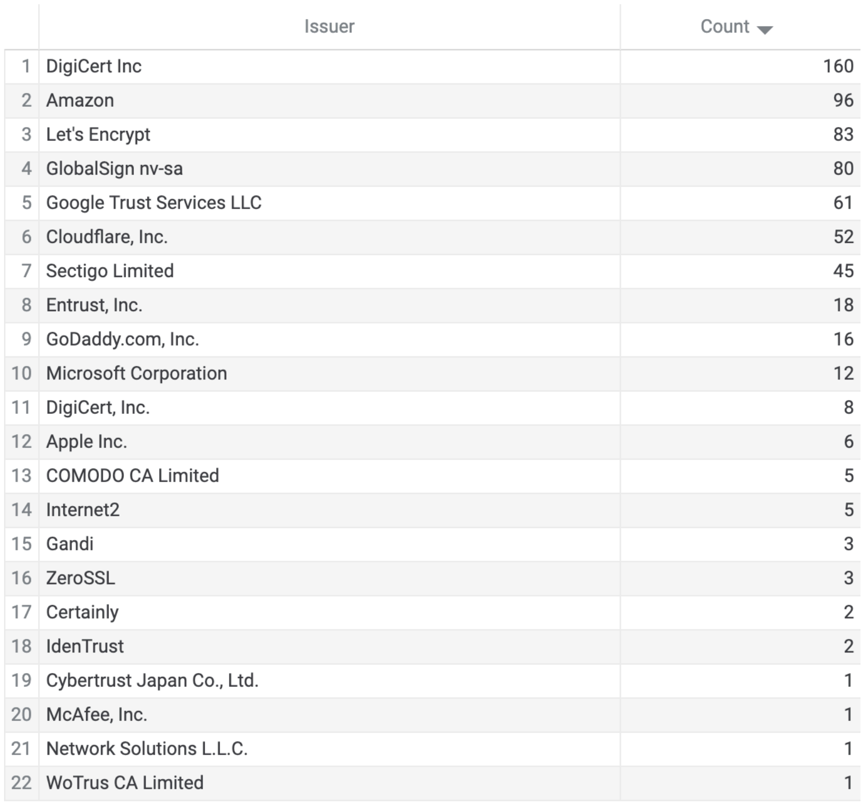
Selenium	Top Sites	Other Sites
HTTPS only	64.0%	72.6%
Both	5.5%	7.7%
HTTP only	2.6%	4.8%
Neither	27.8%	14.5%

These were significantly different from both previous measurement results as a lot more sites were labeled as ‘HTTPS only’ across both Top Sites and Other Sites. The amount of sites that could be accessed through both HTTP and HTTPS diminished significantly as well. This indicates to me that it is more difficult, as it takes more time and space, for the driver to create a display each time it accesses the site, so my code might have timed out when trying to access the HTTP site by itself. My conclusion is that Selenium is not the efficient method to use when measuring across a large sample of sites.

Q12. (Tables attached on next page)

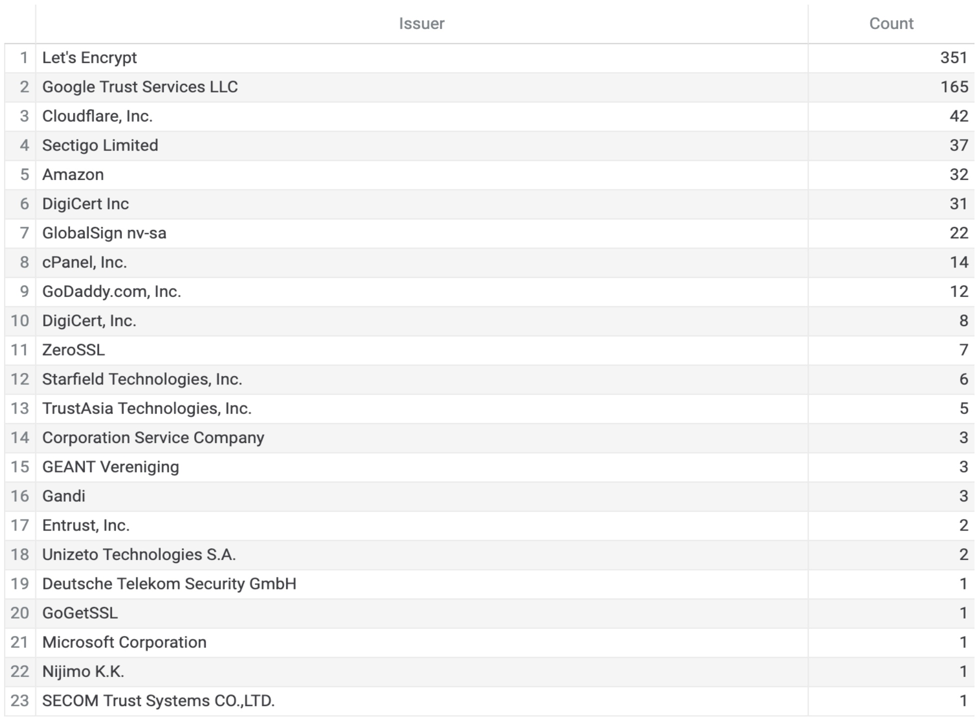
There are many similarities across the Top Sites and Other Sites in terms of who is issuing certificates, though the most common issuers are slightly different between the two lists. There seems to be more non-American CA issuers for Other Sites than Top Sites (just going off their names).

Top Sites:



	Issuer	Count
1	DigiCert Inc	160
2	Amazon	96
3	Let's Encrypt	83
4	GlobalSign nv-sa	80
5	Google Trust Services LLC	61
6	Cloudflare, Inc.	52
7	Sectigo Limited	45
8	Entrust, Inc.	18
9	GoDaddy.com, Inc.	16
10	Microsoft Corporation	12
11	DigiCert, Inc.	8
12	Apple Inc.	6
13	COMODO CA Limited	5
14	Internet2	5
15	Gandi	3
16	ZeroSSL	3
17	Certainly	2
18	IdenTrust	2
19	Cybertrust Japan Co., Ltd.	1
20	McAfee, Inc.	1
21	Network Solutions L.L.C.	1
22	WoTrus CA Limited	1

Other Sites:



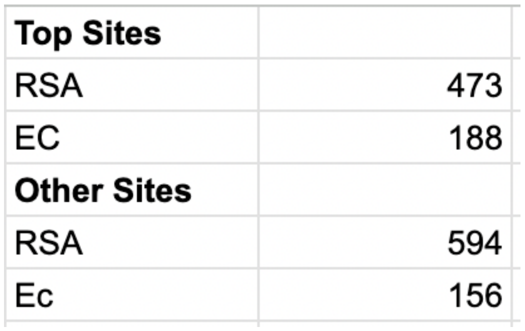
	Issuer	Count
1	Let's Encrypt	351
2	Google Trust Services LLC	165
3	Cloudflare, Inc.	42
4	Sectigo Limited	37
5	Amazon	32
6	DigiCert Inc	31
7	GlobalSign nv-sa	22
8	cPanel, Inc.	14
9	GoDaddy.com, Inc.	12
10	DigiCert, Inc.	8
11	ZeroSSL	7
12	Starfield Technologies, Inc.	6
13	TrustAsia Technologies, Inc.	5
14	Corporation Service Company	3
15	GEANT Vereniging	3
16	Gandi	3
17	Entrust, Inc.	2
18	Unizeto Technologies S.A.	2
19	Deutsche Telekom Security GmbH	1
20	GoGetSSL	1
21	Microsoft Corporation	1
22	Nijimo K.K.	1
23	SECOM Trust Systems CO.,LTD.	1

Q13. The shortest amount of time for which a certificate will be valid amongst Top Sites is 29 days, the longest. The shortest amount of time for which a certificate will be valid amongst Other Sites is 27 days. The longest for both groups is 397 days. The most common amount of times for validity are 89 days, followed by 365 days. 365 days seems like an obvious amount of time to maintain a certificate as it is one year. 89 days, starting on Jan 1st, would end on March 31st, which is the end of Q1, which also makes sense as many of these sites are for-profit businesses that run on a Quarterly calendar year. Following 365 days in frequency is 396 days (1 year + 1 month, so a full calendar year and then January), and 90 days (or 3 months), both which seem reasonable.

Google Trust Services LLC seems to always issue certificates for 83 days, Microsoft for 360 days, Let's Encrypt 89 days, Amazon 394-395 etc. All of these have some outliers here and there but these seem to be the main patterns across issuers. I am assuming this is because they have a default amount of days that a site can get a certificate for and any changes have to be done manually.

The most common amount of time for Other Sites is 89 days while the most common amount of time for Top Sites is 396 days. It makes sense that the most visited sites have a certificate with longer validity. At a high level, there is a larger spread of the number of days of validity for Top Sites than Other Sites.

Q14.



Top Sites	
RSA	473
EC	188
Other Sites	
RSA	594
Ec	156

At a high level, the primary cryptographic algorithms do not vary between the Top Sites and Other Sites. Both lists of sites only use RSA and EC (or Elliptic Curve) algorithms, and both have more sites that use RSA than EC.

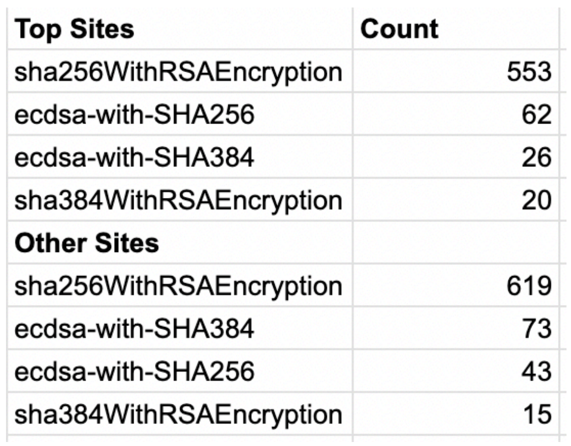
Q15. The key lengths that were used across both lists, Top Sites and Other Sites, were 256, 384, 2048, 3072, and 4096 bits long. At a high level, both Top Sites and Other Sites use a 2048 bit key most frequently, followed by a 256 bit key.

Q16. Yes, there is a "pattern". Every site that has an associated exponent uses 65537 as the exponent, in both Top Sites and Other Sites.

Sites commonly use a RSA encryption exponent of 65537 due to its balanced attributes. This choice strikes a compromise between maintaining a sufficiently high value and avoiding unnecessary computational costs associated with higher exponents. The prime nature of 65537 facilitates RSA modulus generation, while its security advantages, particularly in message encryption and signature schemes, make it a practical and widely adopted exponent for cryptographic applications.

Source:
https://crypto.stackexchange.com/questions/3110/impacts-of-not-using-rsa-exponent-of-65537

Q17.



Top Sites	**Count**
sha256WithRSAEncryption	553
ecdsa-with-SHA256	62
ecdsa-with-SHA384	26
sha384WithRSAEncryption	20
Other Sites	
sha256WithRSAEncryption	619
ecdsa-with-SHA384	73
ecdsa-with-SHA256	43
sha384WithRSAEncryption	15

At a high level, there is a little variance in the trends of the signature algorithm between Top Sites and Other Sites, as the most frequent algorithm to least frequently used is the same across both lists, as seen above.