Visualization of certificate chains associated with selected domains

Matteus Henriksson
Linköping University
Linköping, Sweden
Supervisor: Niklas Carlsson
mathe228@student.liu.se

Rami Latif
Linköping University
Linköping, Sweden
Supervisor: Niklas Carlsson
ramab817@student.liu.se

1 ABSTRACT

Certificate chains are important to provide a secure connection with a server. It is in fact essential to use certificates in order to avoid some attacks and they are mandatory when establishing connections through TLS. To gain further intuition a tool that visualizes certificate chains has been developed.

The tool is intended to illustrate how the certificate chains have changed through the years with the help of graphs constructed in graphviz.

2 INTRODUCTION

The chain of trust is a model made up of a list of certificates starting from the server certificate and ending at the root certificate, between these two there are intermediate certificates. All of them are signed by their respective CA:s. This relationship allows the communications to be encrypted.

Using intermediate certificate authorities instead of Certificate Authorities is a somewhat rising trend among domains, we have seen domains such as Google that have started to use this as a substitute and as a result of this moved closer to the roots. By using intermediate certificates the CA:s do not need to issue certificates to their client from their root certificate and can thereby shorten the distance.

To further research this trend we have developed a tool in python using the graphviz tool to create complete graphs of certificate chains throughout the years. The tool uses a domain name as input and extracts all data from 2000 to 2021. The tool uses data from crt.sh database. With this tool we hope to shed some light on the area.

Roadmap: The paper starts off with introducing concepts and tools that are necessary to understand the tool. After that the Methodology is explained, the section starts off with an overview describing the four main modules of the tool. Later on the results and conclusions where performance and outputs of the tool is discussed.

3 BACKGROUND

Concepts and resources for understanding the report are explained in this section.

3.1 Certificates

Digital Certificates are used to help browsers to determine whether a website is safe to use or not. The browser does this by inspecting the certificates that contain data (signature, key, issuer and more) to ensure their online identity. Digital certificates are issued by Certificate Authorities (CAs) and play a key role since they are asserting the identity of the website.

So to be sure that the website is secure we need to not only trust our browser but also CAs that are managing certificates. All certificates that are issued are stored in certificate logs that can be accessed publicly so that the browsers can check the identity claim from the website.

The root certificate also known as trusted root is central for the trust model. Most devices have a root store, where a list of root certificates are placed, these certificates generally live on the device itself and are fully trusted to sign any certificate. There are many different root stores but, generally the devices use the native root store [8] for example if you own an iPhone the device uses Apple's root store. The root certificates can in turn be used to sign certificates that will automatically be trusted by the browser.

Going directly to the trusted root is not the only option to sign certificates, this can be done with the help of intermediate certificates as well. An intermediate certificate is a certificate signed by a trusted root and can in turn sign other intermediate certificates to create a chain of trust.

3.2 Crt.sh

One of the ways (since 2011) that the web browser decides to trust certain certificates is by looking at public transparency logs. Public transparency logs contain issued certificates by publicly trusted certificate authorities and are used to avoid mis-issued certificates. [9].

Crt.sh is a certificate search engine that is used to look up certificates that have been logged in Certificate Transparancy logs which has been more or less mandatory for certificate authorities [3, 4].

As of 2020 Crt.sh offers a *graph* [1] feature that displays color-coded certificate chains for certificates. This feature has been a keystone for our project.

3.3 Graphviz

Graphviz is an open source visualization software. The software is used to structure data so that it can be represented in diagrams, graphs and networks. The software is widely used in areas such as networking, bioinformatics, software engineering, database design, web design, machine learning [1].

Graphviz allows the user to choose between different layouts that have different purposes, the most common way of using Graphviz is by downloading the package from their website and start writing code with their syntax in some editor. Since our tool is intended to not only display graphs we have used the python package graphviz [2] (version 0.16) so that we could combine our python code and utilize the graphviz functionality to generate graphs directly without any stopovers.

List of layouts:

- "dot" The dot layout is used to make hierarchical drawings of directed graphs constructed with an algorithm that aims to avoid edge crossing and to minimize length of the edge.
- "neato" The neato layout is used to attempt to minimize a global energy function. For example this layout can be used when constructing Entity Relation Diagrams.
- "fdp" The fdp is similar to neato but implements the Fructher-Reingold heuristic.

[1]Example of graph: `https://crt.sh/?graph=1&opt=nometadata`

- "sfdp" The sfdp layout is used for larger graphs and works on the same basis as the fdp layout.
- "twopi" The twopi is a radial layout where the nodes are placed in concentric circles.
- "circo" The circo layout is a circular layout, could for example be used for diagrams that have cyclic structures such as .

In this paper the dot layout was chosen to generate directed graphs. Image of a directed graph can be seen in Figure 1.

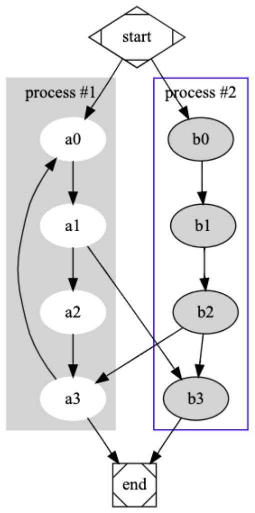


Figure 1: Simple directed graph taken from graphviz website

4 METHODOLOGY

This section provides an overview of the construction, infrastructure and design decisions that were made.

4.1 Overview

The implemented tool was written in python and consists of four parts, firstly to perform a lookup of given URLs on crt.sh and extract information about all accosted certificates. Secondly to filter out irrelevant certificates by matching keywords with the associated DNS-addresses for the certificates, and download certificate chains for the remaining certificates. Thirdly to sort the data in the downloaded certificate chains based on publisher (certificate authority) and year of issue. Lastly to construct graphs that visualise the data.

4.2 Data collection from crt.sh

All data was collected directly from the website crt.sh and downloaded as JSON-files.

The website crt.sh only allows small throughputs, which makes it inefficient to perform a data collection that requires hundreds or thousands of requests to the website. To streamline the data collection process, A Python library was used to make the requests run asynchronous and concurrent. Crt.sh only allows 5 concurrent connections from the same host and if it is exceeded, the server returns status code 429 - too many requests. For that reason semaphores were implemented to set a limit on 5 concurrent requests.

Another risk in addition to the risk of status code 429 when to many requests are is that the server is very unstable and often goes down for short periods of time, usually 10sec - 60sec. During a collection that lasts about an hour, it is common for the website to go down several times. To handle this, all status codes are checked, and if a request fails, a new attempt is made after 20 seconds to ensure that no certificates are missed.

To make it more feasible, the focus was on certificates used for main domain, in other words domains that would have the high rank and be most used. This was performed by matching URL-addresses associated with certificates with a set of keywords, and filter out certificates that do not match. So when a domain is looked up on crt.sh and a list of associated certificates is returned, the list goes through a process where every certificate is analysed and those that are not directly associated with the top domain are filtered out. For example, if the domain "google.com" is looked up certificates associated with the lower level domain "*.mail.google.com" are filtered out.

4.3 Construction of graphs

To construct the graphs we use the *dot* layout described in Section 3.3.

4.3.1 Color-coding.

Since the purpose of the tool is to observe new patterns among the trust relationship and not looking into specific certificates and leaf nodes there has been some filtering and categorization. The categorization is based on CAs and issue years for the certificates associated with the leaf nodes. Leaf nodes that have the same color, year and come from the same CA are grouped into one node.

The categorization of the leaf nodes looks as follows:

- 0-1 years old: blue
- 1-4 years old: green
- 4-8 years old: yellow
- >8 years old: red

Image that illustrates this can be seen in Figure 2.

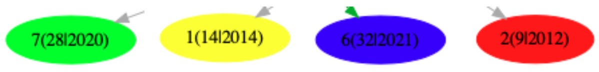


Figure 2: Number of certificates(unique ID of CA|Issue year)

4.3.2 Rendering.

To create a directed graph the library graphviz was used. The library contains a function called *Digraph* that lets the user create a canvas which in turn can be filled with nodes, edges and sub graphs.

To begin with, a sub-graph was created with highest rank so that the graph is aligned at the top of the canvas. The sub-graph was filled with the root store nodes since they are supposed to be at the top of the hierarchy. During this process the unique CA:s where added to the canvas and after that edges and lastly the leaf nodes and their respective edges.

5 RESULT

This section presents the results and is divided into two subsections. Section 5.1 presents statistics related to the performance of the tool and section 5.2 presents generated graphs.

5.1 Performance

The constructed framework was successful in collecting data from crt.sh. however, it proved to be somewhat ineffective. The approach to perform asynchronous requests with a semaphore that allows a maximum of five concurrent connections unfortunately did not improve efficiency. This is probably because crt.sh has set a limit of a certain number of requests per IP address in some time frame, which cannot be circumvented by performing several concurrent requests from the same IP address. This resulted in slower data collection, but despite this the framework was successful in collecting data, even for very large domains such as Google. Table

1 shows that it took 92 minutes to collect data and generate graphs for the domain "google.com", but then one should be aware that this domain is very large and that a total of 1762 certificates were downloaded, while in other hand the graph for "apple.com" was constructed in only 7 minutes with 115 certificates.

Table 1 shows that similar amounts of certificates were found for both "google.com" and "apple.com", but that the amount of matched certificates differ greatly for the two domains. This is probably because only certificates associated with the top level domains were matched. A reasonable assumption is that top level domains for "google.com" are used in greater extend then for the domain "apple.com", which is probably due to the fact that people to a greater extent use different types of internet services related to Apples rather than to visit their website.

The amount of certificates matched and certificates that were downloaded are the same, for all websites. This shows that the framework is capable of handling bad requests and making sure that no certificates are missed, even when the website crashes during data collections.

	Google	Apple	Facebook
Certificates found	2735	2741	3381
Certificates matched	1762	115	1484
Certificates downloaded	1762	115	1484
Time (min)	92	7	87

Table 1: The table show some key statistics related to the data collection process for youtube.com, google.com and facebook.com

5.2 Graphs

The constructed graphs for "apple.com" and "google.com" became very large, which is due to the fact that both domains have used many different certificate authorities for publishing certificates in recent years. But the graphs are clear and intuitive. Figure 3 and 4 show two parts of the graph for "google.com", which represents the two latest certificate authorities used by google which were the only CAs to issue certificates for "google.com" in 2021.

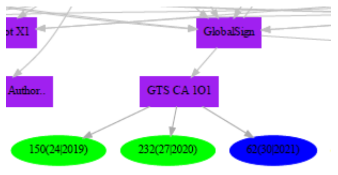


Figure 3: Displaying the amount of certificates issued for "google.com" per year by the certificate authority GTS CA 101

Figure 3 shows GTS CA 101 and it is visible that this CA has issued 150 certificates in 2019, 232 in 2020 and 62 in 2021. Note that the blue ellipse represents the certificates issued this year (2021).

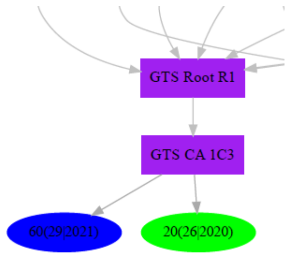


Figure 4: Displaying the amount of certificates issued for "google.com" per year by the certificate authority GTS CA 1C3

The certificate authority GTS CA 1C3 is displayed in Figure 4 and shows that this ca issued 20 certificates in 2020 and 60 certificates in 2021. Note that GTS CA 101 displayed in Figure 3 along with GTS CA 1C3 displayed in Figure 4 are the only certificate authorities used by "google.com" connected to blue ellipses, which is visible in the graph attached in appendix.

6 RELATED WORK

All though the importance of certificates and the chain of trust is well known [7] there aren't that many good available tools that clearly show all important components of the chain such as CA:s, Root stores and certificates. The similar tools that were found mainly focus on displaying adjacent certificates and more simplified relationships that give an overview of the domains [5, 6].

7 CONCLUSION

The constructed framework is very capable of collection certificate data from crt.sh and constructing graphs which provides clear compilations and visualizations of different types of certificate chains associated with a given domain, and in which years the different chains have been relevant. Based on the graphs, it is possible to study changes in certificate chains and patterns. The framework is also robust and capable of dealing with bad requests and website crashes, which ensures that all certificates are downloaded and that no one is missed. This means that the user does not have to control anything during the collection process and can let the process run in the background and trust that the framework has the situation under control.

With that said, the framework also has some shortcomings. First of all, the data collection process is not very efficient. This is not necessarily a problem because the framework is capable of constructing graphs consisting of about a thousand certificates in reasonable time, which is enough for most domains. However, problems might arise if users want to collect data related to a number of different domains and compile it in one and the same graph. Data collection might be improved by, for example, collecting data directly from the database related to the website crt.sh instead of collecting data from the website as JSON files. However, it is unclear how much more efficient it is to download directly from the database. So it is hard to say whether such a change would have made data collection more efficient.

Another disadvantage is that the graphs become very large, which is mainly due to the fact that domains have used a wide variety of certificate authorities, which scales up the graphs.

7.1 Limitations

To make the visualizations more feasible, only certificates used for the main domain have been considered which may give a misrepresentation of what it would have looked like if all certificates had been taken into account.

To get clear visualizations, it was necessary to minimize the amount of data displayed in the graphs.

A lot of time was invested in implementing an asynchronous framework with several concurrent connections to crt.sh. However, this ultimately proved not to increase efficiency. An alternative solution that might have been better is to perform requests directly to the database for crt.sh. But due to lack of time, only one of the two proposed solution has been implemented.

The search function on crt.sh was perceived as unclear. All certificates related to a specific domain were not received by simply searching for a single URL. For example, a search on only “google.com” gave certificates issued until 2014, but a search for "www.google.com" gave certificates issued until 2021, but missed many of the previously issued certificates. This was handled by searching for three URLs for each page, namely "example.com", "www.example.com" and "*.example.com" and extracting all unique certificates from the combined search. However, it was not ensured that all certificates related to a domain actually were retrieved using these there URL combinations. But the designed tool allows the user to add an unlimited number of URLs for the combined search, so if one wants to add more addresses in the future, it is possible.

8 FUTURE WORK

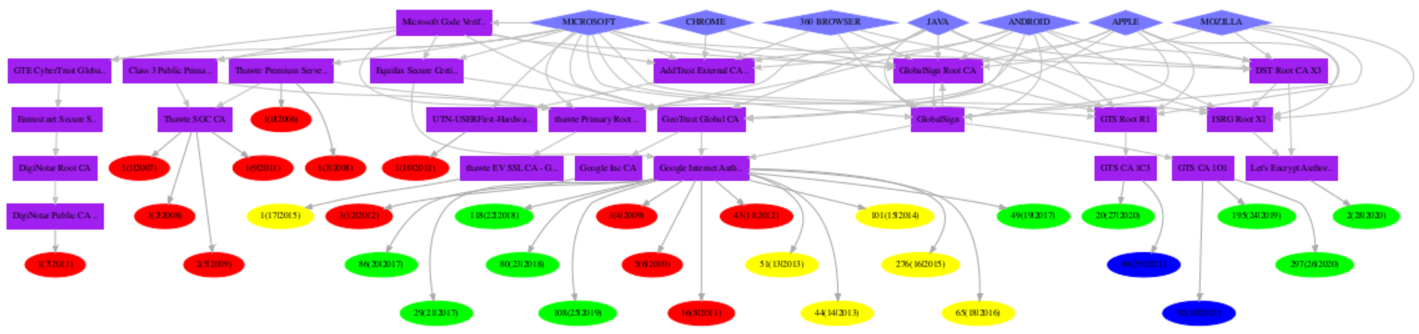
Some minor changes could potentially improve the tools efficiency, such as restructuring the code so that it collects data from the database instead of the website. It could also be possible to continue working on and refine the visualizations, to make the graphs smaller in width.

REFERENCES

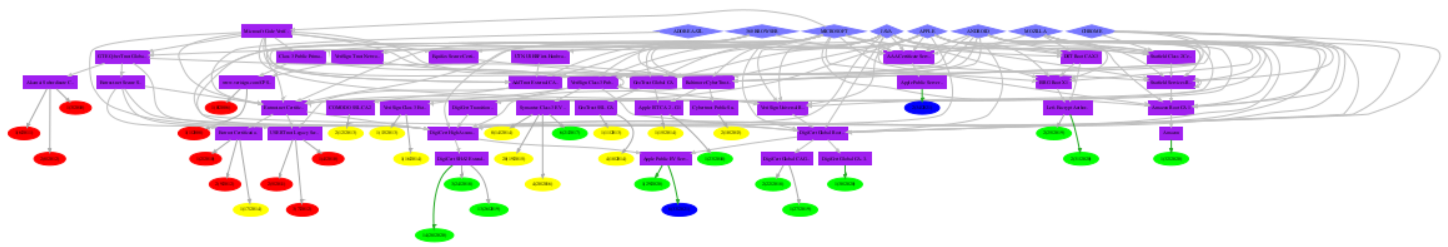
[1] 1991. *Graphviz*. Retrieved May 4, 2021 from https://graphviz.org/
[2] 2014. *Simple Python interface for Graphviz*. Retrieved May 4, 2021 from https://pypi.org/project/graphviz/
[3] 2017. *Announcement: Requiring Certificate Transparency in 2017*. Retrieved May 7, 2021 from https://archive.cabforum.org/pipermail/public/2016-October/008638.html
[4] 2018. *Apple's Certificate Transparency policy*. Retrieved May 7, 2021 from https://support.apple.com/en-us/HT205280
[5] 2019. *certgraph simple D3 "graph" (a hierarchy tree, really) of the most popular websites*. Retrieved May 11, 2021 from https://github.com/ndrix/certgraph
[6] 2020. *A tool to crawl the graph of certificate Alternate Names*. Retrieved May 11, 2021 from https://gitlab.com/kalilinux/packages/certgraph
[7] Stefan Brands. 2000. *Rethinking public key infrastructures and digital certificates: building in privacy*. Mit Press.
[8] Zakir Durumeric, James Kasten, Michael Bailey, and J. Alex Halderman. 2013. Analysis of the HTTPS Certificate Ecosystem. In *Proceedings of the 2013 Conference on Internet Measurement Conference* (Barcelona, Spain) *(IMC '13)*. Association for Computing Machinery, New York, NY, USA, 291–304. https://doi.org/10.1145/2504730.2504755
[9] Ben Laurie. 2014. Certificate transparency. *Commun. ACM* 57, 10 (2014), 40–46.

Appendices

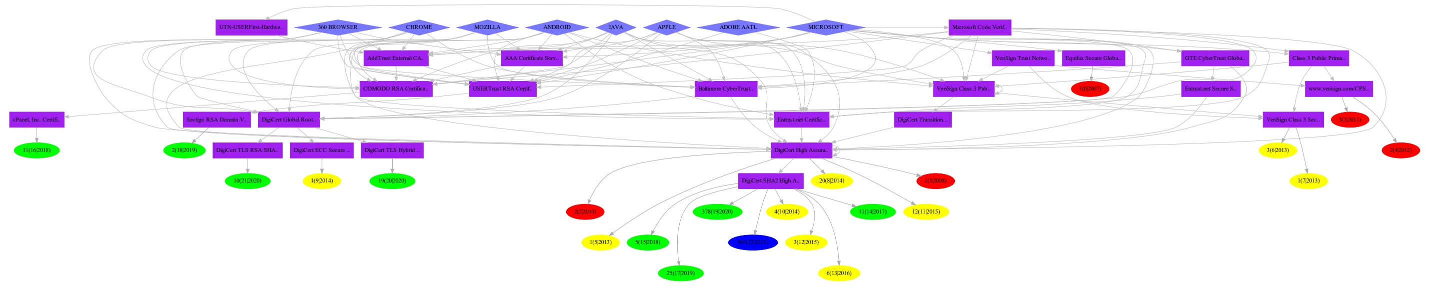
A GRAPH FOR GOOGLE



B GRAPH FOR APPLE



C GRAPH FOR FACEBOOK


UTN-USERFirst-Hardwa..
360 BROWSER
CHROME
MOZILLA
ANDROID
JAVA
APPLE
ADOBE AATL
MICROSOFT
Microsoft Code Verif..
AddTrust External CA..
AAA Certificate Serv..
VeriSign Trust Netwo..
Equifax Secure Globa..
GTE CyberTrust Globa..
Class 3 Public Prima..
COMODO RSA Certifica..
USERTrust RSA Certif..
Baltimore CyberTrust..
VeriSign Class 3 Pub..
1(0|2007)
Entrust.net Secure S..
www.verisign.com/CPS..
cPanel, Inc. Certifi..
Sectigo RSA Domain V..
DigiCert Global Root..
Entrust.net Certific..
DigiCert Transition ..
VeriSign Class 3 Sec..
3(3|2011)
11(16|2018)
2(18|2019)
DigiCert TLS RSA SHA..
DigiCert ECC Secure ..
DigiCert TLS Hybrid ..
DigiCert High Assura..
3(6|2013)
2(4|2012)
10(21|2020)
1(9|2014)
19(20|2020)
DigiCert SHA2 High A..
20(8|2014)
1(1|2008)
1(7|2013)
2(2|2010)
378(19|2020)
4(10|2014)
11(14|2017)
12(11|2015)
1(5|2013)
5(15|2018)
963(22|2021)
3(12|2015)
25(17|2019)
6(13|2016)