POLITECNICO DI TORINO

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea

Censorship Detector: using X.509 certificates to detect censorship implemented via DNS manipulation

Relatore
Prof.ssa Diana Gratiela Berbecaru

Riccardo Fiorilla

Novembre–Dicembre 2024

Summary

DNS manipulation attacks can be implemented by different kinds of malicious actors, with the purpose of luring the victim into a different webpage compared to the one that was originally meant to be visited. However, the content of the malicious page can vary depending on the attacker's reasons, which span from credentials theft to censorship. During the years, different kinds of methodologies have been proposed to identify when such attacks happen. One of the latest heuristics consist in exploiting information obtained by X.509 certificates, which can indicate if a manipulation attack has been put into place.

This work focuses on DNS manipulation attacks implemented with the purpose of hiding domains and censoring them, typically implemented by malicious DNS resolvers. The aforementioned methodology, based on certificates analysis, is proven to be a valid one to effectively identify such attacks.

This methodology is used by *Censorship Detector*, a tool which has been developed with the purpose of identifying if a resolver provides manipulated DNS responses with the purpose of blocking domains. This tool is mainly based on certificates fields' check, but also implements other auxiliary analysis techniques, taking into account other data retrieved when connecting to a website, including the response status code and the page's HTML.

When using this tool to inspect DNS resolvers, the results obtained are consistent with the ones presented by previous studies, confirming that X.509 certificates provide essential information for detecting DNS manipulation attacks.

Contents

Chapter 1

Introduction

The DNS protocol is a useful and irreplaceable technology in internet connections, which is constantly used by countless users without even noticing it. Thanks to this protocol it is possible for a user to obtain the actual IP address of the specific server hosting a domain, only by knowing the domain name.

This protocol presents some vulnerabilities, mainly due to some of the characteristics that it natively had when it was firstly designed. During the years, multiple vulnerabilities were discovered and patched, but nowadays the protocol can still be insecure, making it possible for user to be victim of various kind of attacks, which most of the time consist in the manipulation of a DNS response. This type of attack is possible because the DNS protocol was not originally designed taking response authentication into account, along with the fact that resolvers itself could also return a manipulated record if their owner is not trusted. Victims of these attacks are typically redirected to different webpages, which do not correspond to the legit domain and are usually created and owned by the malicious actors themselves.

This work focuses on how DNS manipulation attacks can be identified, proposing a methodology based on Public Key Certificates' analysis, which turn out to be a strong indicator of when such attacks happen.

The thesis begins by providing a technical background on the DNS protocol and Public Key Infrastructures, focusing on the X.509 certificate standard. The topics covered include the description of a DNS message's format, X.509 certificates' fields and extensions, certificate revocation methods, the process of issuing a certificate for a domain, Certificate Transparency, and general descriptions of attacks related to these technologies.

Five different articles are listed and used as a support for this work. Such articles revolve around DNS manipulation attacks and domain impersonation.

Schwittmann et al. [1] present a complete description of various techniques used to perform domain impersonation, in particular on how to make it possible for a CA to issue a certificate for a domain not actually owned by the attacker, showing how the domain validation phase can be very vulnerable if performed by using certain protocols.

Akiwate et al. [2] propose a methodology based on Certificate Transparency with the purpose of retroactively identifying when a domain has been victim of DNS hijacking attack, usually performed by the attackers with the purpose of obtaining a certificate issued to the victim's domain.

Tsai et al. [3] discuss the topic of DNS manipulation attacks performed by DNS resolvers located in countries governed by authoritarian regimes. Such manipulations are performed with purpose of enforcing censorship over online contents and redirecting users to blockpages, which replace entirely the original domain's webpage. The study proposes to identify when such manipulation happens by using Public Key Certificates, which give plenty of information that can be used to determine whether a domain has been hidden or not. The authors of this article developed a tool called *CERTainty* and used it to analyse a big dataset of certificates provided by webpages obtained when resolving domains via resolvers suspected of implementing manipulation-based censorship. The work I made on this thesis is strongly based on *CERTainty* and on the way it works.

Roberts et al. [4] base their study on domain impersonation attacks implemented by exploiting target embedding in domain names. The attack in question consists in obtaining a certificate from a CA for a malicious domain which embeds the domain name of a legit website in its name.

Shulman et al. [5] show a demonstration of how an off-path attacker can control the domain validation process performed by a CA with the purpose of obtaining a certificate for a domain not owned by the attacker and impersonate it. Such attack is based on DNS response fragmentation, after which malicious data is injected in the message before it reaches the CA.

My personal contribution consists in a tool I developed for this thesis, called *Censorship Detector*. Its purpose is to use information provided by X.509 certificates to identify when a malicious resolver performs DNS manipulation. The tool analyses the certificates fields, looking for some specific suspicious patterns, along with trying to establish an HTTPS connection with the websites and retrieving both the response status codes and the pages' HTMLs.

This study is concluded by showing the results obtained by an execution of *Censorship Detector* when testing a resolver suspected of manipulating DNS responses and censoring contents. The results obtained by my tool are consistent with the ones obtained by previous studies, including the ones retrieved by *CERTainty*.

The results obtained in this work demonstrate that X.509 certificates can be very useful for detecting the occurrence of DNS manipulation attacks.

Chapter 2

Background

2.1 DNS protocol

In order to perform a connection with a webserver, it is necessary for a client to use the server's IP address, which univocally identifies it among the internet architecture. However, it is impossible for a web client to know exactly every corresponding IP address for all the websites on the internet. This created the need of a protocol that allows the translation of IP address into common names and vice versa.

The Domain Name System (DNS) protocol, standardized in RFC-1034 [7] and RFC-1035 [8], allows users to visit websites simply by using their domain names, hence making it unnecessary to remember their IP addresses. It consists of a hierarchical and distributed architecture, where multiple servers are in charge of storing the correspondence between IP addresses and domain names. The process of translation from domain name to IP address performed by a DNS server is referred to as *resolution*.

The DNS infrastructure is based on a tree of servers, each one with a different role. When a client asks for a DNS resolution, it exploits a recursive DNS resolver, which is in charge of passing the request to every other DNS server needed to perform the translation between name and IP address. The first server reached by the resolver is the root DNS server, which is in charge of routing the request to the right top-level domain (TLD) DNS server, depending on the domain's extension (e.g., "`.com`", "`.net`", "`.org`"). Once the query reaches the TLD server, it gets routed again to an authoritative DNS server, that stores the correspondence between IP addresses and complete domain names (figure 2.1).

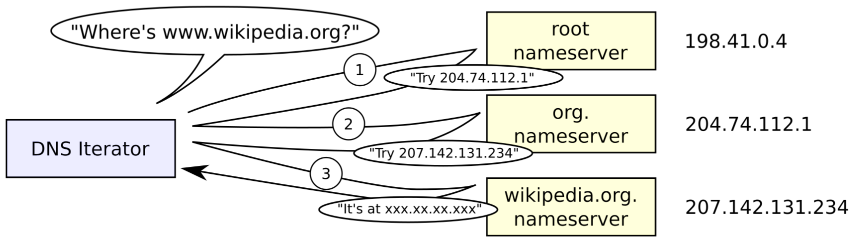


Figure 2.1. Example of DNS resolution (source: [9]).

Following the previous description, it is easy to understand how domain name are exactly composed: each domain name is composed by different levels, divided by each other using a full stop (".") symbol, allowing each domain to be mapped among DNS servers in a clear way. Going top-down in a DNS server tree corresponds to going from right to left on a domain name. The TLD corresponds to the rightmost part of the domain name, then at its left we find the second-level domain (2LD), then the third-level domain (3LD) and so on. Going from right to left, each next level domain is considered a subdomain of the previous one.

For example, in `en.wikipedia.org`, "`org`" is the TLD, "`wikipedia`" is the 2LD and "`en`" is the 3LD; "`wikipedia`" is a subdomain (figure 2.2) of the "`org`" TLD and "`en`" is a subdomain of "`wikipedia.org`". The complete domain name "`en.wikipedia.org`" is referred to as a fully qualified domain name (FQDN).

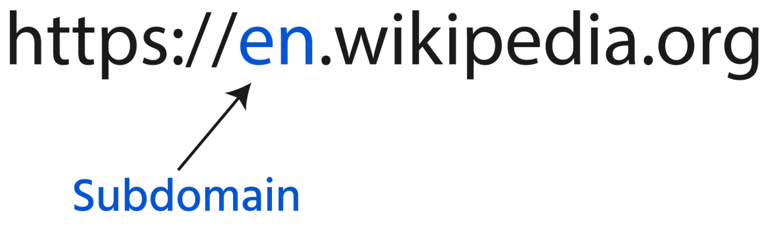


Figure 2.2. One of the most common type of subdomain (source: [10]).

DNS caching

All the operations needed to perform a DNS resolution can result in a burden on the servers, slowing down the process. In order to improve efficiency, some internet service providers introduced caching servers, with the purpose of enhancing the protocol's performance. These servers' duty consists in storing the results of some DNS resolutions, so that a client can obtain a domain's IP address without going through the whole servers' tree.

These records are stored along with a time to live (TTL) indicating for how long the resolution is considered valid. The TTL can vary from a few seconds to several weeks, and it is determined by the authoritative DNS server's administrator.

2.1.1 DNS record types

DNS Resource Records (RRs) stored in nameservers have the goal of pairing a domain name with various kinds of information. For this reason, different types of DNS records exist. Some of them are listed below, along with their definition according to IBM [11].

A records Also referred to as "address records", they are the most common DNS records used, creating an association between a domain name and an IPv4 address.

AAAA records They are similar to A address, but they connect domain names to IPv6 addresses, which are nowadays becoming more common.

CNAME records They are canonical name records and are used to link subdomains to domain's address records. They are very useful since they allow the use of one single address record for multiple subdomains, so that if the main domain's IP address is changed, it is only necessary to update the corresponding A record, while the CNAME records connected to it will update accordingly.

DNAME records They are delegation name records and are used similarly to CNAME records. The only difference is that they are used for subdomains to refer to a DNS tree portion which has a different name than them, hence pointing to a completely different domain.

MX records They are mail exchange records and are used to direct emails to a specific mail server. These records allow the creation of personal email accounts linked with a specific domains.

NS records They are nameserver records and show which server acts as an authoritative nameserver for a specific domain. An NS record points to all the different records held by a domain.

PTR records They are pointer records and are used for the opposite purpose of an A record: they are used to perform an inverse resolution, starting from an IP address with the goal of finding the corresponding domain name.

TXT records They are text records and are used to store some textual information related to their respective domain.

2.1.2 DNS message format

As described in RFC-1035 [8], DNS messages present a standard format, which consists of a header followed by four other sections: question, answer, authority, and additional space. The header section is always present, while the other ones can also be empty in certain cases. The header includes fields that specify which of the other sections are present, along with stating whether the message is a query or a response. The general structure of a DNS message is shown in figure 2.3.

```
+---------------------+
|        Header       |
+---------------------+
|       Question      | the question for the name server
+---------------------+
|        Answer       | RRs answering the question
+---------------------+
|      Authority      | RRs pointing toward an authority
+---------------------+
|      Additional     | RRs holding additional information
+---------------------+
```

Figure 2.3. General structure of a DNS message (source: [8]).

The header section format is shown in figure 2.4 and its fields are described below.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      ID                       |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|QR|   Opcode  |AA|TC|RD|RA|   Z    |   RCODE   |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    QDCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ANCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    NSCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                    ARCOUNT                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.4. Header section format (source: [8]).

`ID` A 16-bit long identifier produced by the program that generated the query, used to match a query to its corresponding response.

`Flags` A 16-bit long section split in multiple fields, each one with a different purpose:

- `QR`: a 1-bit field that specifies whether the message is a query (`QR = 0`) or a response (`QR = 1`);
- `OPCODE`: a 4-bit field that specifies the type of query in the message (e.g., `OPCODE = 0` indicates a standard query, while `OPCODE = 1` indicates an inverse query); this value is set by the originator of a query and copied into the corresponding response;
- `AA`: a 1-bit field active only in responses with the goal of specifying if the responding server is an authority for the domain name present in the question;
- `TC`: a 1-bit field used to specify if the message has been truncated due to length bigger than the one permitted by the transmission channel;
- `RD`: a 1-bit field used to specify to the nameserver to pursue the query recursively;
- `RA`: a 1-bit field used to denote if recursive query support is available in a nameserver;
- `Z`: a 3-bit field filled with zeros, reserved for future use;
- `RCODE`: a 4-bit field which has the duty of specifying the status of the response; `RCODE = 0` means no error, `RCODE = 1` indicates a format error, `RCODE = 2` is returned in case of a server failure, `RCODE = 3` means that the domain name referenced in the query doesn't exist, `RCODE = 4` is returned in case the nameserver doesn't support that specific kind of query, and `RCODE = 5` indicates that the nameserver refused to perform the specified operation for policy reasons.

`QDCOUNT` An unsigned 16-bit integer used to specify the number of entries in the question section.

`ANCOUNT` An unsigned 16-bit integer used to specify the number of RRs in the answer section.

`NSCOUNT` An unsigned 16-bit integer used to specify the number of nameserver RRs in the authority records section.

`ARCOUNT` An unsigned 16-bit integer used to specify the number of RRs in the additional records section.

The question section format is shown in figure 2.5 and is composed by three fields.

```
                                1  1  1  1  1  1
  0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                     QNAME                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QTYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     QCLASS                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.5. Question section format (source: [8]).

`QNAME` A domain name written as a sequence of labels, each one consisting of a length octet (8 bits) followed by the specified number of octets, hence making the total length of this section variable.

`QTYPE` A 2-octet long code specifying the type of the query.

`QCLASS` A 2-octet long code specifying the class of the query.

The answer, authority, and additional sections all share the same format, which is shown in figure 2.6 and is referred to as resource record format. This section is composed by six fields.

```
                               1  1  1  1  1  1
 0  1  2  3  4  5  6  7  8  9  0  1  2  3  4  5
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                                               |
/                                               /
/                      NAME                     /
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TYPE                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                     CLASS                     |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                      TTL                      |
|                                               |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
|                   RDLENGTH                    |
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--|
/                     RDATA                     /
/                                               /
+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+--+
```

Figure 2.6. Resource record section format (source: [8]).

`NAME` A domain name associated with the resource record.

`TYPE` A 2-octet long field specifying the RR type code, so the meaning of the RDATA field.

`CLASS` A 2-octet long field specifying the class of the data in the RDATA field.

`TTL` A 32-bit long field specifying (in seconds) the RR's time to live, so how long the resource can be cached before being discarded. If this value is set to zero, the resource can only be used for the current transaction and cannot be cached.

`RDLENGTH` An unsigned 16-bit long integer specifying the length in octets of the RDATA field.

`RDATA` A variable length string of octets describing the resource itself, whose format varies according to the TYPE and CLASS fields.

2.1.3 DNS security

The original version of the DNS protocol didn't include any security features, which allowed attackers to exploit vulnerabilities in the system, mainly by posing by an authoritative server. This was possible since a DNS response could be accepted without need of authentication by the responder. The ID field is indeed the only (weak) form of authentication for DNS messages: a message is authenticated if the response ID is equal to the query ID. Over the years, some security protocols were proposed to enhance DNS security. Two of them are the Domain Name System Security Extensions (DNSSEC) and the DNS-based Authentication of Named Entities (DANE).

DNSSEC consists in a suite of extensions, standardized in RFC-4033 [12], with the purpose of providing cryptographic authentication, integrity and authenticated denial of existence regarding data exchanged using the DNS protocol. Since its proposal, the main goal of DNSSEC was to add security while also maintaining backward compatibility, which is still not a simple thing to do.

DNSSEC was designed in order to protect applications from receiving forged or manipulated DNS data. Every answer coming from DNSSES protected zones is digitally signed. A DNS resolver can check this digital signature and check whether said data is equal to the one published by the zone owner. This operation can protect any kind of DNS records.

It is important to notice that DNSSEC doesn't provide data confidentiality (DNSSEC responses are authenticated, but not encrypted) and doesn't grant data availability (e.g., cannot protect against DoS attacks). Moreover, the adoption of this protocol is still far from being widespread, and there are some disagreements between different entities regarding multiple topics yet, such as who should be the owner of the TLD root keys.

DANE, proposed in RFC-6698 [13], has the goal of providing authenticity by enabling the administrator of a domain name to store its domain's key into the DNSSEC infrastructure, hence creating a bond between a key and a domain name. This protocol has a very limited support in applications (e.g., Google Chrome and Mozilla Firefox don't support it), and it has been developed as an alternative solution to another method used to provide authenticity for web domains, a way more popular one: Public Key Certificates issued by Certification Authorities.

2.2 Public Key Certificate

A Public Key Certificate (PKC) is a data structure used to securely bind a public key to some attributes. It consists in an electronic document that is typically used to ensure the ownership of a particular Public Key (PK) by some entity, with the final goal of achieving non-repudiation of digital signatures. This whole architecture is based on some trusted authorities (figure 2.7), each one with a different role:

- Certification Authority (CA): it generates, signs and publishes PKCs, along with information about their status;
- Registration Authority (RA): it verifies claimed identity and attributes, and authorises issuing or revocation of a PKC;
- Validation Authority (VA): it provides services used to verify the validity status of a PKC (e.g., CRL download or OCSP responder);
- Revocation Authority: it is delegated to timely revoke a PKC, which is a very urgent operation (it is an unofficial term, its role can be assigned to RA or CA).

If a user wants to obtain a certificate, it generates an asymmetric key pair, stores the Private Key (SK) in a secure location accessible only by him, and sends the PK to a CA, along with some identification attributes. These attributes are also sent to a RA, which checks if they are valid and, if the answer is positive, authorises the CA to issue the certificate. The CA creates the certificate and uses its own SK to digitally sign both the user's PK and attributes. Finally, the CA sends the certificate to the user and stores it in a public repository.

This is not the only possible architecture to generate a PKC. The Public Key Infrastructure (PKI) is the set of roles and policies used in the process of creation, management, distribution, storage and revocation of a PKC.

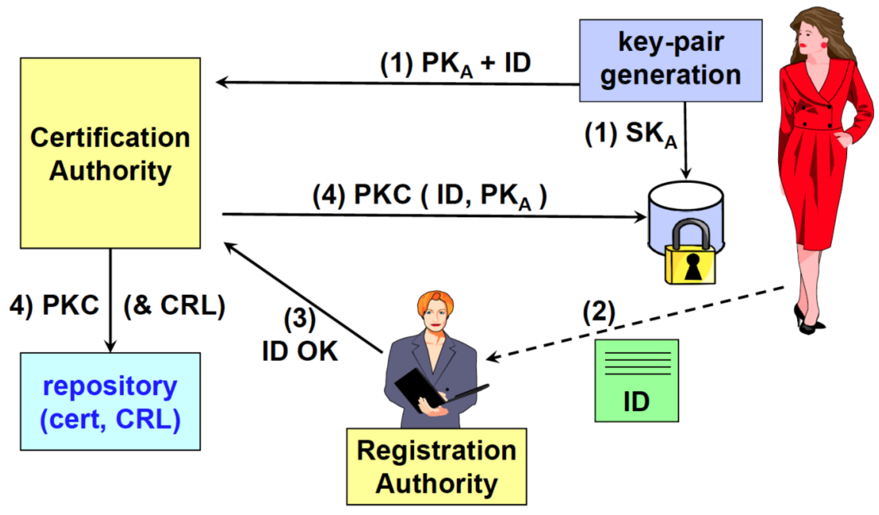


Figure 2.7. Certificate issuing process (source: Prof. Lioy's slides on PKI).

2.3 X.509 certificates

X.509 is the main standard used for PKCs. It was defined by the International Telecommunications Union's Standardization Sector (ITU-T) and its first version (X.509v1) was published in 1988 as part of the X.500 standards for directory services. X.509v2 was published in 1993, but the differences from the previous version were minor. These early versions still didn't ensure the needed requirements: the quality of the CAs was not guaranteed, there was no X.500 infrastructure, and it was difficult to establish the certification path between two arbitrary users.

To solve these problems, in 1996, X.509v3 was created. It was the result of a joint work of ISO/ITU and IETF, which developed a more flexible and expressive format for the certificate. This version extends the previous one by adding some extension fields, which are classified in two categories:

- Public extensions: they are defined by the standard and consequently made public to anybody;
- Private extensions: they are unique for a certain user community.

These two types of extensions will be discussed in detail further in this work.

Certificate Policy and Certification Practice Statement

A Certificate Policy (CP) consists of a set of rules used to indicate the applicability of a PKC to a particular community or class of application with common security requirements. A CP can be followed by many CAs (e.g., a government CP to be implemented by all certification providers), which must fit the minimum requirements specified by it.

A Certification Practice Statement (CPS) defines all the practices employed by a particular CA in the process of issuing a PKC. A CPS contains implementation details and is specific to a single CA.

The framework for writing both CP and CPS is standardised by RFC-3647 [14].

2.3.1 Certification path

To make sure its signature is trustworthy, a CA must be verified by another CA, which signs the certificate of the first one. The order and priority used to verify different CAs depends on the PKI. A hierarchical PKI consists in a tree of CAs, each one signing the certificates of the CAs of the level below, while the whole tree is rooted at a self-signed root CA.

RFC-1422 [15] defines a standard for a hierarchical infrastructure used for Privacy-Enhanced Electronic Mail (PEM). This certification hierarchy also embodies semantics which are not explicitly addressed in X.509 certificates, but which are consistent with X.509 precepts. In this way, the semantic is forced in the context external to the certificate (e.g., in the application).

In this infrastructure there are three types of authorities:

- Internet Policy Registration Authority (IPRA): it is the root of the hierarchy, and it issues certificates for the level below; every certification path has IPRA as root;
- Policy Certification Authorities (PCAs): they are on the second level of the hierarchy, and each one of them is certified by the IPRA; each CPA has a list of different policies used to issue a certificate, so that a user can choose is CPA based on its needs: a high-assurance PCA has a stricter set of rules than a mid-level one;
- Certification Authority (CA): they are on the third level of the hierarchy, and each one of them is certified by a CPA.

This infrastructure worked with X.509v1, presenting a series of problem such as limited flexibility of the hierarchical structure, undesired limits to the assignment of X.500 names caused by the name subordination, and the impractical use of PCAs in commercial applications caused by the need of an operator to take a decision.

With the introduction of X.509v3 certificates, the aforementioned problems were solved:

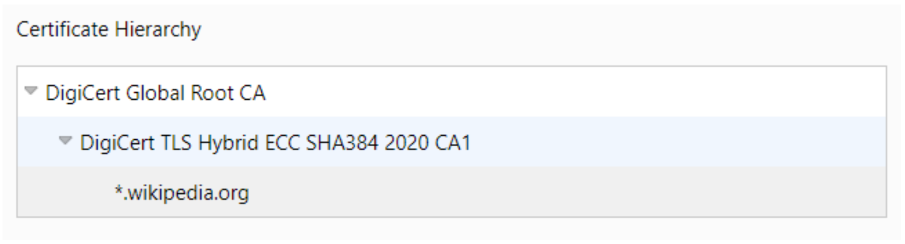
- The certification path root is no longer the IPRA: it can be a CA present in a user's trusted list;
- The name subordination rule can be imposed using the name constraints extension;
- The certification protocol has been automated, so that the use of PCAs is no longer needed.

In an X.509v3 certificate, it is possible to tell if the subject of the certificate is a CA or an End Entity (EE). X.509v3 PKI is described in RFC-5280 [16], and the certificates are classified in three different categories:

- Cross-certificates: the issuer and the subject are different, and they describe a trust relationship between two CAs;
- Self-issued certificates: the issuer and the subject are the same entity, and they are generated to support changes in policies or operations;
- Self-signed certificates: they are self-issued certificates where the digital signature may be verified by the public key bound into the certificate, and they are used to begin certification paths.

Using a browser, we can check the validity of a website's certificate and visualise its certification path. Figure 2.8 shows the certification path of `en.wikipedia.org`:

- `*.wikipedia.org` is the domain name of the site and it is the EE of the certification path; the asterisk at the beginning of the name means that the certified domain includes various versions of the website (e.g., `en`, `it`, `es` for English, Italian and Spanish versions, respectively);
- `DigiCert TLS Hybrid ECC SHA384 2020 CA1` is the intermediate CA that verifies the Wikipedia domain;
- `DigiCert Global Root CA` is the self-signed root CA that certifies the intermediate CA.


Certificate Hierarchy

DigiCert Global Root CA

DigiCert TLS Hybrid ECC SHA384 2020 CA1

*.wikipedia.org

Figure 2.8. Certification path for `en.wikipedia.org`.

2.3.2 X.509 certificate base syntax

This is the structure of an X.509 certificate, as it is described in [16]:

```
Certificate ::= SEQUENCE {
  signatureAlgorithm     AlgorithmIdentifier,
  tbsCertificate         TBSCertificate,
  signatureValue         BIT STRING
}
```

`signatureAlgorithm` This field identifies the cryptographic algorithm used by the CA to sign the certificate.

`signatureValue` This field contains the digital signature of the certificate computed by the CA, encoded as a bit string.

`tbsCertificate` This field contains multiple values regarding the certificate, the subject and the issuer. It consists in a `TBSCertificate` sequence, and this is its structure:

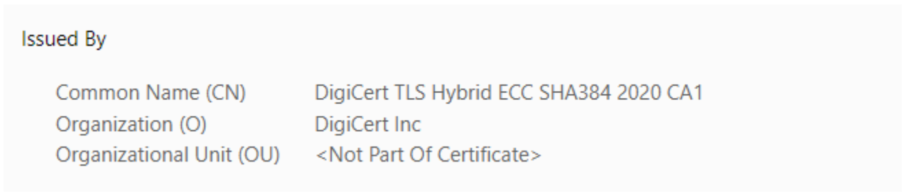
```
TBSCertificate ::= SEQUENCE {
  version                [0] Version DEFAULT v1,
  serialNumber           CertificateSerialNumber,
  signature              AlgorithmIdentifier
  issuer                 Name,
  validity               Validity,
  subject                Name
  subjectPublicKeyInfo   SubjectPublicKeyInfo,
  issuerUniqueID         [1] IMPLICIT UniqueIdentifier OPTIONAL,
                          -- if present, version must be v2 or v3
  subjectUniqueID        [2] IMPLICIT UniqueIdentifier OPTIONAL
                          -- if present, version must be v2 or v3
  extensions             [3] Extensions OPTIONAL
                          -- if present, version must be v3
}
```

`version` This field identifies the version of the certificate. The default value of this field represents v1. If extensions are present in the certificate, version must be v3. If no extensions are present, but a unique identifier is present, version should be v2.

`serialNumber` This field represents a positive integer value assigned by the CA to each certificate. This number must be unique, and it is used to identify a particular certificate.

`signature` This field contains the identifier for the algorithm used by the CA to sign the certificate. The value contained in this field must be equal to the value of the `signatureAlgorithm` field in the `Certificate` sequence.

`issuer` This field identifies the entity that has signed and issued the certificate (figure 2.9). It must contain a non-empty distinguished name.

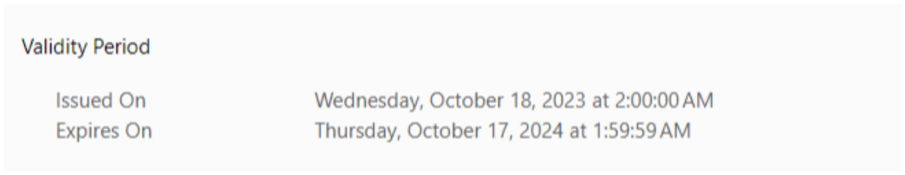


Issued By

Common Name (CN)	DigiCert TLS Hybrid ECC SHA384 2020 CA1
Organization (O)	DigiCert Inc
Organizational Unit (OU)	<Not Part Of Certificate>

Figure 2.9. Issuer field in `en.wikipedia.org` certificate.

`validity` This field defines the interval of time during which the CA guarantees that it will maintain information about the status of the certificate, which will be considered valid only between two specific dates (figure 2.10). These two dates are defined by the **`notBefore`** and **`notAfter`** values: the first one defines the date and time from which the validity of the certificate starts, while the second one represents the date and time of the expiration of the certificate.



Validity Period

Issued On	Wednesday, October 18, 2023 at 2:00:00 AM
Expires On	Thursday, October 17, 2024 at 1:59:59 AM

Figure 2.10. Validity field in `en.wikipedia.org` certificate.

`subject` This field identifies the entity associated with the public key stored in the certificate (figure 2.11). If non-empty, the subject field must contain an X.500 distinguished name, which must be unique for each entity certified by the CA defined in the issuer field. A CA may issue more than one certificate with the same distinguished name to the same entity.

`subjectPublicKeyInfo` This field (figure 2.12) carries the public key and identify the algorithm with which the key is used (e.g., RSA, DSA, Diffie-Hellman).

`issuerUniqueID/subjectUniqueID` This fields are present only in v2 and v3 certificates and they are used to handle the possibility of reuse of issuer or subject names over time. Anyway, reusing the same name for different entities is not recommended, and CAs conforming to this rule must not generate certificates with unique identifiers.

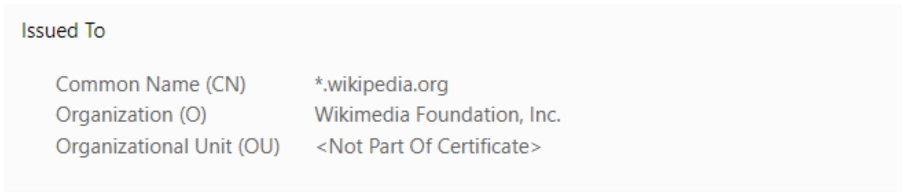
`extensions` This field must only appear in v3 certificates, and it consists in a sequence of one or more certificate extensions.

2.3.3 X.509 extensions

Certificate extensions were added in X.509v3, in order to include more information in a PKC. As mentioned before, these extensions can be classified as public or private: the first ones are defined in RFC-5280 [16], while the second ones are unique for a certain community of users.

An extension can also be defined as critical or non-critical: in the verification process, a certificate containing an unrecognized critical extension must be rejected by the system using the certificate, while an unrecognized non-critical extension may be ignored, but must be processed if recognized.

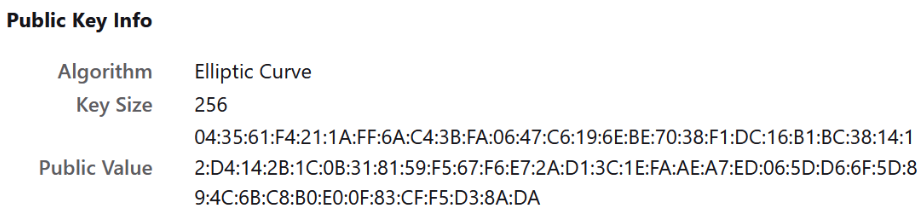
The decision of accepting or rejecting the certificate is full responsibility of the entity performing the verification, which is referred to as the Relying Party (RP).



Issued To

Common Name (CN)	*.wikipedia.org
Organization (O)	Wikimedia Foundation, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Figure 2.11. Subject field in `en.wikipedia.org` certificate.



Public Key Info

Algorithm	Elliptic Curve
Key Size	256
Public Value	04:35:61:F4:21:1A:FF:6A:C4:3B:FA:06:47:C6:19:6E:BE:70:38:F1:DC:16:B1:BC:38:14:12:D4:14:2B:1C:0B:31:81:59:F5:67:F6:E7:2A:D1:3C:1E:FA:AE:A7:ED:06:5D:D6:6F:5D:89:4C:6B:C8:B0:E0:0F:83:CF:F5:D3:8A:DA

Figure 2.12. Public Key Info field in `en.wikipedia.org` certificate.

Public extensions

X.509v3 define four public extension classes:

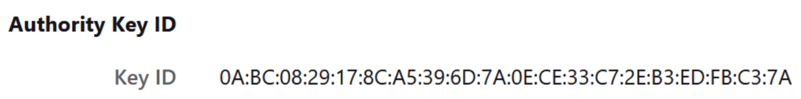
- Key and policy information;
- Certificate subject and certificate issuer attributes;
- Certificate path constraints;
- CRL distribution points.

Key and policy information

Authority Key Identifier This extension (figure 2.13) is used to identify the public key corresponding to the private key used to sign a certificate, and it is used in the case of a CA owning multiple key couples. The key is identified by means of a key identifier, which is typically the digest of the PK, or using the issuer name and the serial number as a pair. The AKI is classified as a non-critical extension but is very useful in some applications to build the certification chain: for this reason, it must be included in all certificates generated by conforming CAs. The only exception for this is when a CA distributes its own public key in the form of a self-signed certificate: in this case, the AKI may be omitted.

Subject Key Identifier This extension is used to identify a specific public key used in an application and is very useful in case the public key is updated. It is considered a non-critical extension. In certificates issued by conforming CAs, the value of AKI must match the value of SKI of the CA that issued the certificate. There are two main ways to compute a SKI: using a 160-bit SHA-1 hash of the value of the subject's PK, or using the four-bit value 0100 followed by the least significant 60 bits of the SHA-1 hash of the subject's PK. Other methods are also accepted, as long as they generate unique numbers.

Key Usage This extension (figure 2.14) defines the purpose of the key contained in the certificate, identifying the application domain for which the PK can be used. It can be a critical or non-critical extension. If it is critical, the certificate can be used only for the purposes specified by the KU extension, which is defined as a bit string, each one representing a specific usage for the key. If the bit corresponding to a particular usage is asserted, then the key can be used for that operation. These are the name of the fields of the KU extension, which define the allowed cryptographic operations:

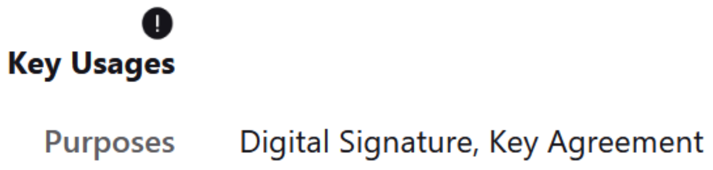


Authority Key ID

Key ID 0A:BC:08:29:17:8C:A5:39:6D:7A:0E:CE:33:C7:2E:B3:ED:FB:C3:7A

Figure 2.13. Authority Key Identifier extension in `en.wikipedia.org` certificate.

- `digitalSignature`: subject PK can be used to verify a digital signature;
- `nonRepudiation` (`contentCommitment` in recent X.509 versions): subject PK can be used to provide a non-repudiation service;
- `keyEncipherment`: subject PK can be used for enciphering a private or secret key (e.g., for key transport); `dataEncipherment`: subject PK can be used for enciphering raw user data, which is extremely uncommon, since almost all applications use key transport or key agreement to establish a symmetric key;
- `keyAgreement`: subject PK can be used for key agreement operations (e.g., Diffie-Hellman);
- `keyCertSign`: subject PK can be used to verify a signature on a PKC (valid only for CAs); `cRLSign`: subject PK can be used to verify a signature on Certificate Revocation List (valid only for CAs);
- `encipherOnly`: if the `keyAgreement` bit is set, the subject PK can be used only for enciphering data while performing a key agreement operation;
- `decipherOnly`: if the `keyAgreement` bit is set, the subject PK can be used only for deciphering data while performing a key agreement operation.



Key Usages

Purposes Digital Signature, Key Agreement

Figure 2.14. Key Usage extension in `en.wikipedia.org` certificate, marked as critical.

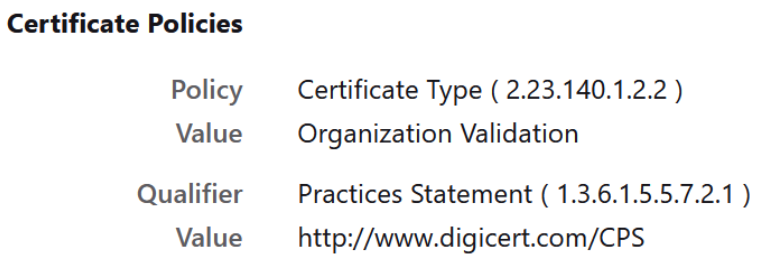
Private Key Usage Period This extension defines the usage period of the private key. It is always non-critical, and its usage is discouraged.

Certificate Policies This extension (figure 2.15) contains a sequence of one or more policy information terms, each one consisting of an object identifier (OID) and optional qualifiers. Its purpose is to describe the list of policies followed when the certificate was issued and the operations for which the certificate can be used. It can be critical or non-critical. In a CA certificate, this extension limits the set of policies for its certification path. If a CA doesn't want to limit the certification policies for its path, it can set this extension with the `anyPolicy` value of { `2 5 29 32 0` }.

Policy Mappings This extension is used only in CA certificates and indicates the correspondence of policies among different certification domains. It is a non-critical extension.

Certificate subject and certificate issuer attributes

Subject Alternative Name This extension (figure 2.16) is used to allow the identification of the owner of the certificate using different formalisms (e.g., e-mail address, IP address, URL). The SAN extension is always considered critical if the subject field contains an empty sequence, while it can be considered non-critical if the subject field is not empty.



Certificate Policies

Policy	Certificate Type (2.23.140.1.2.2)
Value	Organization Validation
Qualifier	Practices Statement (1.3.6.1.5.5.7.2.1)
Value	http://www.digicert.com/CPS

Figure 2.15. Certificate Policies extension in `en.wikipedia.org` certificate.

Issuer Alternative Name This extension allows to identify the issuer of a certificate using different formalisms (e.g., e-mail address, IP address, URL). The IAN is always considered a critical extension if the issuer field contains an empty sequence, while it can be marked as non-critical if the issuer field is not empty.

This is the list of possible alternative names, used both is SANs and IANs:

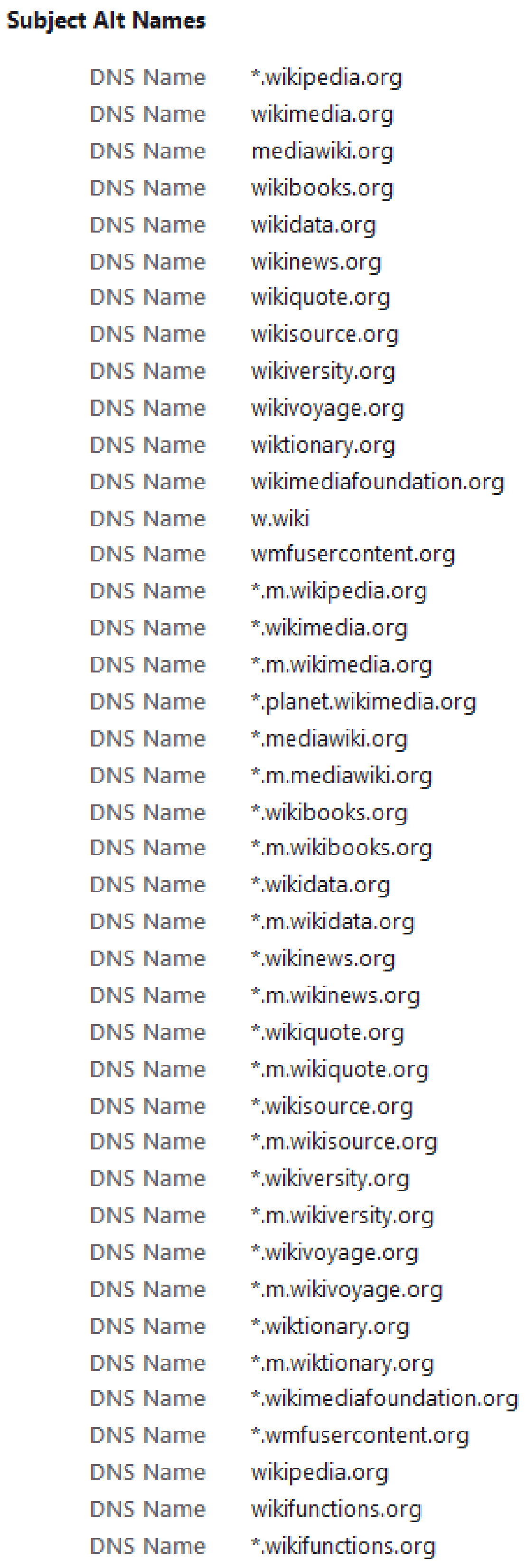
- `rfc822Name`
- `dNSName`
- `iPAddress`
- `uniformResourceIdentifier`
- `directoryName`
- `X400Address`
- `ediPartyName`
- `registeredID`
- `otherName`

Subject Directory Attributes This extension allows to store directory attributes associated with the owner of the certificate (e.g., nationality). The usage of these attributes strongly depends on the application, as there is no standard definition for this extension, which is considered non-critical.

Certificate path constraints

Basic Constraints This extension (2.17) is used to identify whether the subject of a certificate is a CA (`BC = true`) or an EE (`BC = false`). In case the subject is considered a CA, it is possible to define the maximum depth of a valid certification path that contains the certificate. If the BC extension doesn't consider the subject a CA, the certificate's public key cannot be used to verify certificate signatures, so the `keyCertSign` bit in the KU extension must not be asserted. Thanks to this extension, it is possible to specify the maximum number of non-self-issued intermediate certificates that may follow this one in a certification path. The BC extension can be marked as critical or non-critical, but the first choice is always strongly suggested.

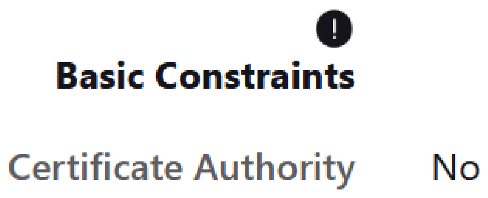
Name Constraints This extension is only used in CA certificates and its purpose is to indicate a name space within which all subject names in subsequent certificates in a certification path must be located. This restriction is applied both to the subject distinguished name and to the subject alternative names. The name space specification can be applied in a whitelist or in a blacklist fashion. The NC extension can be marked as critical or non-critical.



Subject Alt Names

DNS Name	*.wikipedia.org
DNS Name	wikimedia.org
DNS Name	mediawiki.org
DNS Name	wikibooks.org
DNS Name	wikidata.org
DNS Name	wikinews.org
DNS Name	wikiquote.org
DNS Name	wikisource.org
DNS Name	wikiversity.org
DNS Name	wikivoyage.org
DNS Name	wiktionary.org
DNS Name	wikimediafoundation.org
DNS Name	w.wiki
DNS Name	wmfusercontent.org
DNS Name	*.m.wikipedia.org
DNS Name	*.wikimedia.org
DNS Name	*.m.wikimedia.org
DNS Name	*.planet.wikimedia.org
DNS Name	*.mediawiki.org
DNS Name	*.m.mediawiki.org
DNS Name	*.wikibooks.org
DNS Name	*.m.wikibooks.org
DNS Name	*.wikidata.org
DNS Name	*.m.wikidata.org
DNS Name	*.wikinews.org
DNS Name	*.m.wikinews.org
DNS Name	*.wikiquote.org
DNS Name	*.m.wikiquote.org
DNS Name	*.wikisource.org
DNS Name	*.m.wikisource.org
DNS Name	*.wikiversity.org
DNS Name	*.m.wikiversity.org
DNS Name	*.wikivoyage.org
DNS Name	*.m.wikivoyage.org
DNS Name	*.wiktionary.org
DNS Name	*.m.wiktionary.org
DNS Name	*.wikimediafoundation.org
DNS Name	*.wmfusercontent.org
DNS Name	wikipedia.org
DNS Name	wikifunctions.org
DNS Name	*.wikifunctions.org

Figure 2.16. Subject Alternative Name extension in `en.wikipedia.org` certificate.



Basic Constraints

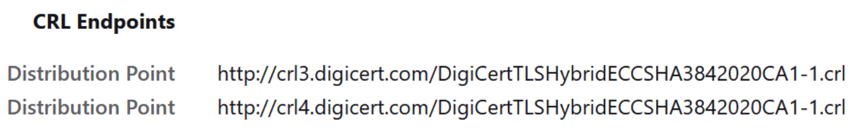
Certificate Authority	No

Figure 2.17. Basic Constraints extension in en.wikipedia.org certificate, marked as critical.

Policy Constraints This extension is only used in certificates issued to CAs (not to EEs), and it applies two different kinds of constraints: it can be used to prohibit policy mapping or to require that each certificate in a path contain an acceptable policy identifier. It can be marked as critical or non-critical.

CRL Distribution Point

This extension (also known as CRLDP or CDP) is used to identify how to obtain information about Certificate Revocation Lists (detailed description about certificate revocation further in this work). In particular, this extension is used to identify the distribution point of the CRL to be used to validate a certificate (figure 2.18). The distribution point can be indicated with directory entry, e-mail or URL. It can be considered a critical or a non-critical extension.



CRL Endpoints

Distribution Point	http://crl3.digicert.com/DigiCertTLSHybridECCSHA3842020CA1-1.crl
Distribution Point	http://crl4.digicert.com/DigiCertTLSHybridECCSHA3842020CA1-1.crl

Figure 2.18. CRL Distribution Points extension in en.wikipedia.org certificate.

Other extensions

Extended Key Usage This extension is used to indicate one or more purposes for which the certificate's public key can be used, in addition or in place of the basic purposes indicated in the KU extension. In general, this extension is used only in EE certificates, and can be marked as critical or non-critical.

Inhibit `anyPolicy` This extension is used only in certificates issued to CAs, and its purpose is to indicate that the special `anyPolicy` OID, with value { `2 5 29 32 0` }, is not considered an explicit match for other certificate policies except when it appears in an intermediate self-issued CA certificate. The value of this extension indicates the number of additional non-self-signed certificates that may appear in the path before `anyPolicy` is no longer permitted. It is marked as a critical extension.

Freshest CRL (a.k.a. Delta CRL Distribution Point) This extension has the same syntax as the CRL Distribution Point one and is used to provide an additional source to download information about any certificate revoked since the last update of the full CRL. Thanks to this extension, an application is not forced to download the full version of the CRL. It is considered a non-critical extension.

Private extensions

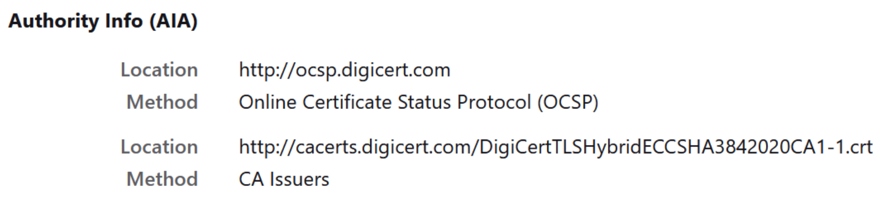
Private extensions are defined only for a specific user community. IETF-PKIX defined three private extensions for the Internet user community.

Subject Information Access This extension is used to specify a method (e.g., HTTP, LDAP) to obtain information about the owner of a certificate and a name that indicates the location of said information. It is marked as non-critical.

Authority Information Access This extension (figure 2.19) is used to indicate how to access information and services of the CA that issued the certificate. Some of the services provided are:

- `certStatus`: it retrieves information about how to get the certificate's issuer;
- `certRetrieval`: it retrieves the certificate of the CA itself;
- `cAPolicy`: it retrieves the certificate of the CA itself; it retrieves policies followed by the CA;
- `cACerts`: it gets the list of all the certificates issued by the CA.

It can be marked as critical or non-critical.


Authority Info (AIA)

Location	http://ocsp.digicert.com
Method	Online Certificate Status Protocol (OCSP)
Location	http://cacerts.digicert.com/DigiCertTLSHybridECCSHA3842020CA1-1.crt
Method	CA Issuers

Figure 2.19. Authority Information Access extension in `en.wikipedia.org` certificate.

CA Information Access This extension is used to indicate how to access information and services of the CA that owns the certificate. It provides the same services as the AIA extension but is valid only in a CA certificate. It can be marked as critical or non-critical.

2.4 Certificate Revocation

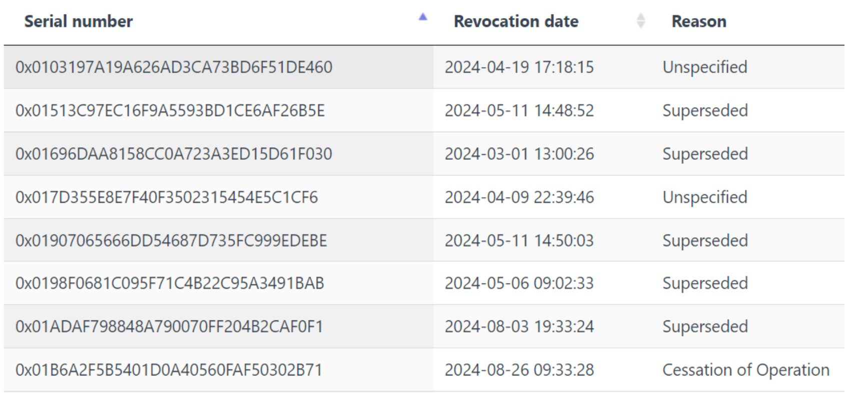
As seen before, PKCs are considered valid only during a specific period of time, which spans between the two dates defined in the validity field. A certificate can be renewed when it's close to its expiration, but this is not always the case. There are some circumstances in which it might be necessary to invalidate a certificate before it expires, for example if the private key is compromised. This is done because keeping the certificate usable would threaten the security of the whole system, as someone could easily impersonate the legit server with malicious intents.

In the case of a compromised key, the certificate revocation is typically requested by the certificate owner itself, but it is not always like this. The revocation can be requested also by the certificate sponsor, for example in case an employee abandons the organization, and it can also be requested by the issuer, in case of issuance error or fraud.

The certificate status must be checked by the entity that accepts it to protect a transaction: the Relying Party (RP). If not expired, a PKC is considered valid unless otherwise stated. There are two main protocols used to check if a certificate has been revoked: the Certificate Revocation List (CRL) and the Online Certificate Status Protocol (OCSP).

2.4.1 Certificate Revocation List

The RFC-5280 [16] also describes the standard for Certificate Revocation Lists, which is one of the main ways used to detect whether or not a certificate has been revoked. A CRL simply shows a list of revoked certificates, along with their revocation date and the reason why they were revoked (figure 2.20).



Serial number	Revocation date	Reason
0x0103197A19A626AD3CA73BD6F51DE460	2024-04-19 17:18:15	Unspecified
0x01513C97EC16F9A5593BD1CE6AF26B5E	2024-05-11 14:48:52	Superseded
0x01696DAA8158CC0A723A3ED15D61F030	2024-03-01 13:00:26	Superseded
0x017D355E8E7F40F3502315454E5C1CF6	2024-04-09 22:39:46	Unspecified
0x01907065666DD54687D735FC999EDEBE	2024-05-11 14:50:03	Superseded
0x0198F0681C095F71C4B22C95A3491BAB	2024-05-06 09:02:33	Superseded
0x01ADAF798848A790070FF204B2CAF0F1	2024-08-03 19:33:24	Superseded
0x01B6A2F5B5401D0A40560FAF50302B71	2024-08-26 09:33:28	Cessation of Operation

Figure 2.20. Certificate Revocation List for en.wikipedia.org.

This is the main structure of a CRL, as it is described in [16]:

```
CertificateList ::= SEQUENCE {
  tbsCertList            TBSCertList,
  signatureAlgorithm     AlgorithmIdentifier,
  signatureValue         BIT STRING
}

TBSCertList ::= SEQUENCE {
  version                Version OPTIONAL,
                          -- if present, version must be v2
  signature              AlgorithmIdentifier
  issuer                 Name,
  thisUpdate             Time,
  nextUpdate             Time OPTIONAL
  revokedCertificates    SEQUENCE {
    userCertificate      CertificateSerialNumber,
    revocationDate       Time,
    crlEntryExtensions   Extensions OPTIONAL,
  }
  crlExtensions          [0] Extensions OPTIONAL
}
```

The main components of a CRL are the algorithm used to sign it, the value of the signature and the list of revoked certificates. CRLs are updated periodically, even if no other revoked certificate is added to the list: this is done to ensure that a user can always access a fresh CRL and to avoid replay attacks. CRLs can be signed by the CA issuing the certificates or by a revocation authority delegated by the CA, thus becoming and indirect CRL (iCRL).

In a certificate, the CRL distribution point extension provides some URLs which can be used to download the CRL associated with the certificate. In this way, it is possible to consult the history of all the certificates revoked to a particular entity, and to check how often the CRL is updated.

These files can become very large (in the order of some MB) and costly to download. One of the main solutions to this problem consists in publishing only one base CRL and then the updated differences with respect to it. This is referred to as the delta CRL technique, and it is very useful to speed up the process of obtaining a CRL, since it is not necessary to download the whole list but only the differences from the base CRL.

2.4.2 Online Certificate Status Protocol

In RFC-6960 [17] we can find the standardization of another system used to check the validity of a certificate, the Online Certificate Status Protocol. This method allows the user to check if a certificate is in a valid state by querying an OCSP server, which provides the revocation status of a specific certificate.

The user sends a request to the OCSP server, asking about the status of a certificate during that exact moment. The request contains the protocol version, the target certificate identifier (can also be more than one in the same request), and some optional extensions. The server responds with the syntax of the version, the name of the responder, information about all the certificates in the request, some optional extensions, the OID of the signature algorithm, and the signature computed over the hash of the response.

OCSP can be implemented both as a binary protocol, or encapsulated in HTTP, HTTPS, LDAP or SMTP. Depending on the provider, some OCSP servers may accept only signed requests, with the purpose of restricting the number of clients. The digital signature of the response is computed to avoid responses from fake OCSP servers.

The most important information communicated by the OCSP server to the client can be found in the SingleResponse sequence. The server produces one SingleResponse for each queried certificate, and the values contained in this data structure are:

- `certID`: the ID of the certificate, which is composed by four other values (`hashAlgorithm`, `issuerNameHash`, `issuerKeyHash`, `serialNumber`);
- `certStatus`: the status of the certificate, which can be good, revoked or unknown (this last case usually happens when the request indicates an unrecognized issuer that is not served by this responder);
- `thisUpdate`: the most recent time at which the status being indicated is known by the responder to have been correct;
- `nextUpdate`: the time at or before which newer information will be available about the status of the certificate (optional field).

If the certificate is revoked, the responder may also provide some optional information about when and why the certificate has been revoked (`revocationTime`, `revocationReason`).

The main reason why OCSP is used is to avoid downloading an entire CRL when a check about the status of a certificate is needed. Anyway, this protocol also has its own drawbacks: the reliability of this technique strongly depends on the availability of the OCSP server, which, in order to provide a good quality service, must give low latency responses, as well as being down as rarely as possible. Moreover, OCSP server are vulnerable to replay attacks and DoS attacks.

2.5 Certification Path Validation

When a web browser tries to connect to a domain, it must verify the validity of the domain's certificate, in order to make the connection secure. However, since said certificate is only the EE of a certification path, it is necessary to check if every certificate in the chain is valid at that exact moment in time.

This has to be done since, even if the domain's certificate appears to be valid, it could have been issued by a CA with an invalid certificate. If a single intermediate certificate in the chain is found to be invalid, every certificate up to it is considered invalid too; in this case, the browser must discard the whole path.

RFC-5280 [16] proposes a standard for a certification path validation algorithm, which should be implemented by any system handling and using certificates, in order to verify their validity. A series of constraints checks is performed iteratively on every certificate of the chain; if at the end of the process no certificate is marked as invalid, the whole path is considered good, and the EE's certificate is accepted.

Going top-down in the chain, the algorithm starts by verifying that the first certificate is issued by a trust anchor, which is an authoritative entity for which trust is assumed and not derived. In the case of the X.509 architecture, a self-issued root certificate would be considered a trust anchor.

In a path composed by `n` certificates, certificate `n` is the one to be validated (the EE's certificate). For every certificate `x` in {`1, ..., n-1`}, the algorithm verifies if the subject of certificate `x` is equal to the issuer of certificate `x+1`, hence checking if every certificate in the chain did in fact issue the next one (figure 2.21).

Finally, for every certificate `x` in {`1, ..., n`}, the algorithm verifies the validity of `x` at the time in question, checking whether the certificate has expired or not, or if it has been revoked by some authority before its natural expiration date; this last thing is done by consulting a CRL or an OCSP server.

Moreover, the algorithm performs further checks on name constraints (domain name limitation to a specific domain tree), policy constraints, basic constraints (maximum path length), key usage (e.g., CA certificates should have the certificate signing indicator set) and other critical extensions.

2.6 Domain Validation, Organization Validation and Extended Validation

To request a PKC for a web domain, the owner provides to a CA the name of the domain and the PK from an asymmetric key pair that he owns. However, these two elements are not enough for the CA to certify the ownership of the domain by the requester.

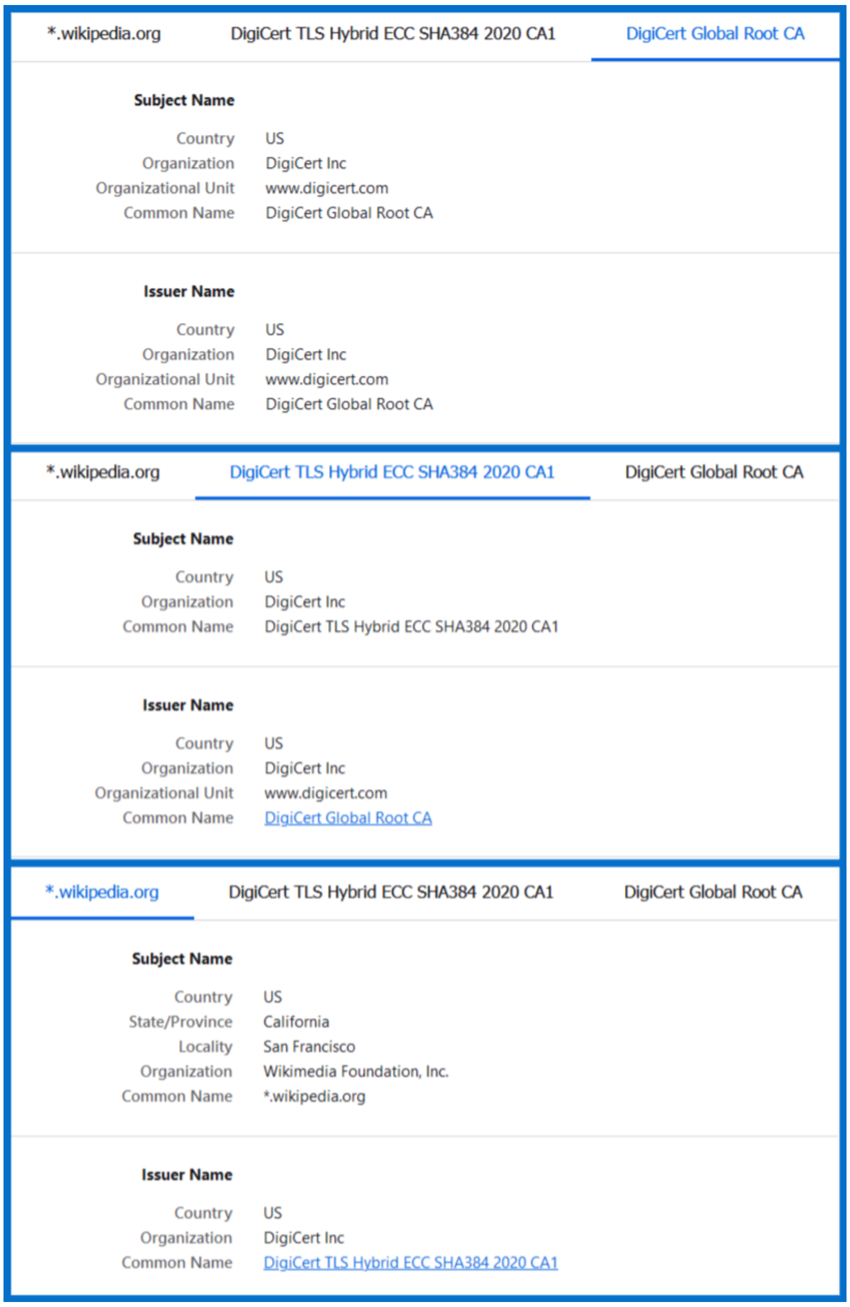
When a CA issues a certificate for a specific domain, it performs a series of security checks in order to verify if the requester is actually the domain owner. The layers of security involved in this phase can be different, depending on the requester's needs; faster methods generally grant a lower validation level.

On DigiCert [19] website, it is possible to find a classification of three different types of validation (figure 2.22) regarding certificate requests for internet domains: Domain Validation (DV), Organization Validation (OV) and Extended Validation (EV).

DV is the simplest validation technique: after receiving the request, the CA looks for the domain owner email address in the WHOIS database and contacts him using this method. The WHOIS protocol is standardized in RFC-3912 [18]: it consists of a query/response TCP-based protocol originally used to provide "white pages" services and information about registered domain names. This is the lowest standard of validation on the Internet; it is useful in case a domain owner needs a certificate as soon as possible. DV is typically performed on blogs, personal websites and any website that doesn't conduct transactions or gather personal information.

OV is a validation technique that provides mid-level business certificates. In particular, it performs nine validation checks on the company that owns the website, mainly over the business name, type, status and physical address. When visiting a website with a OV certificate, all these details about the company are visible by the user. OV is typically performed on log-in screens and business sites.

EV guarantees the highest level of brand identity security, performing a total of 16 validation checks, including vetting of the business' public phone number, length of time in business, registration number and jurisdiction, as well as domain fraud check, contact blacklist check and a telephone call to authenticate the employment status of the requestor. When visiting a website with a EV certificate, all these details about the company are visible by the user. EV is typically performed on websites owned by global banks and e-commerce services.



*.wikipedia.org | DigiCert TLS Hybrid ECC SHA384 2020 CA1 | DigiCert Global Root CA

Subject Name

Country	US
Organization	DigiCert Inc
Organizational Unit	www.digicert.com
Common Name	DigiCert Global Root CA

Issuer Name

Country	US
Organization	DigiCert Inc
Organizational Unit	www.digicert.com
Common Name	DigiCert Global Root CA

*.wikipedia.org | DigiCert TLS Hybrid ECC SHA384 2020 CA1 | DigiCert Global Root CA

Subject Name

Country	US
Organization	DigiCert Inc
Common Name	DigiCert TLS Hybrid ECC SHA384 2020 CA1

Issuer Name

Country	US
Organization	DigiCert Inc
Organizational Unit	www.digicert.com
Common Name	DigiCert Global Root CA

*.wikipedia.org | DigiCert TLS Hybrid ECC SHA384 2020 CA1 | DigiCert Global Root CA

Subject Name

Country	US
State/Province	California
Locality	San Francisco
Organization	Wikimedia Foundation, Inc.
Common Name	*.wikipedia.org

Issuer Name

Country	US
Organization	DigiCert Inc
Common Name	DigiCert TLS Hybrid ECC SHA384 2020 CA1

Figure 2.21. Valid certification path for `en.wikipedia.org`.

DigiCert states that its validation team rejects approximately 3750 EV certificate applications every year, mainly because of fraudulent requests.

2.7 Let's Encrypt

Let's Encrypt [20] is the world biggest CA in terms of emitted certificates, with a market share of 58.8% according to W3Techs [21]. Its popularity is due to the easiness and quickness in requesting and receiving a certificate for a domain, along with the fact that this service is provided completely for free.

The issuing process of a Let's Encrypt certificate is completely automated. Every certificate issued or revoked is publicly recorded and available for anyone to inspect. The protocol used

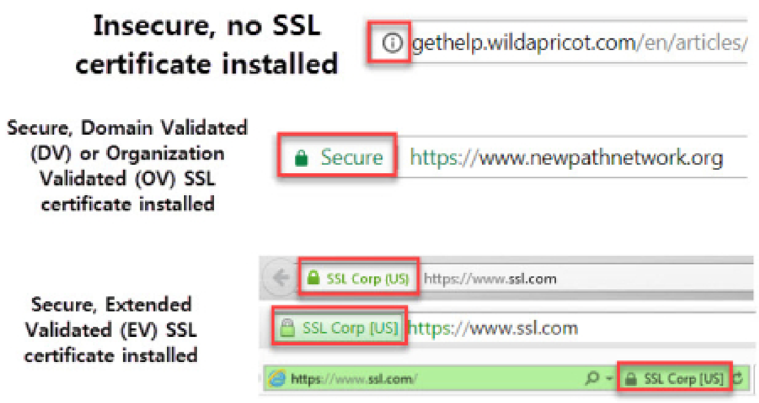

Insecure, no SSL certificate installed
gethelp.wildapricot.com/en/articles/
Secure, Domain Validated (DV) or Organization Validated (OV) SSL certificate installed
Secure
https://www.newpathnetwork.org
Secure, Extended Validated (EV) SSL certificate installed
SSL Corp (US)
https://www.ssl.com
SSL Corp [US]
https://www.ssl.com
https://www.ssl.com/
SSL Corp [US]

Figure 2.22. Example of DV, OV and EV (source: [22]).

.

to issue and revoke certificates is published as an open standard, following the free software philosophy.

Due to its automated nature, Let's Encrypt only issues DV certificates.

2.7.1 ACME Protocol

The issuance of Let's Encrypt certificates is performed according to the Automatic Certificate Management Environment (ACME) protocol, which is standardized in RFC-8555 [23]. Thanks to the ACME protocol, it is possible to set up an HTTPS server and make it automatically obtain a browser-trusted certificate, without any human intervention. This is accomplished by running a certificate management agent on the web server.

This process consists of two steps: first, the agent proves to the CA that the requester (a web server administrator) controls the specific domain for which the certificate is being requested; then, the agent will be able to request, renew, and revoke certificates for that domain.

In the Domain Validation phase, the agent asks the Let's Encrypt CA what it needs to do to prove the ownership of the domain. The CA issues one or more challenges (figure 2.23), that usually consist in provisioning an HTTP resource under a well-known URI on the domain; in this way, it is possible to verify if the requester is able to edit the domain's paths. Moreover, the CA also provides the agent with a nonce to be signed with its private key (figure 2.24).

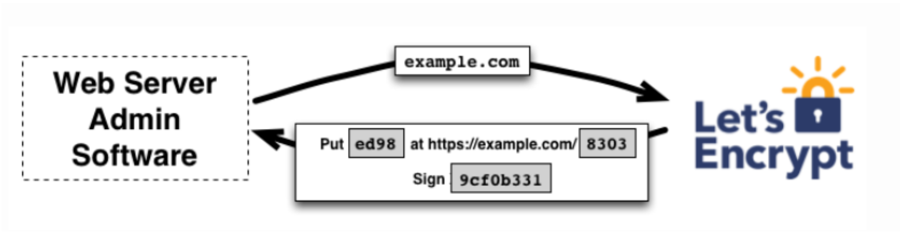

Web Server Admin Software
example.com
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt

Figure 2.23. Example of Let's Encrypt issuing a challenge (source: [20]).

.

If the challenges are completed successfully and the signature over the nonce is valid, then the agent identified by the public key is authorized to do certificate management for that domain. The key pair used by the agent are referred to as an "authorized key pair" for the domain.

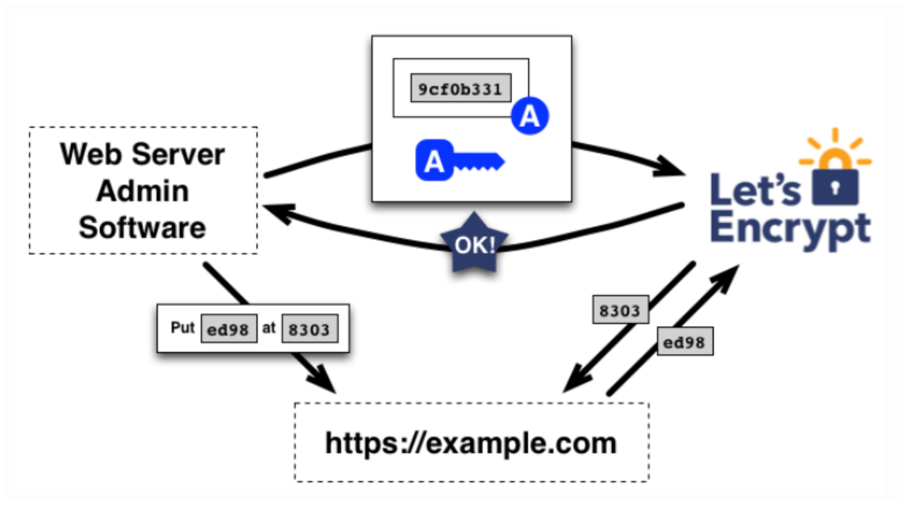


Figure 2.24. Let's Encrypt successful challenge solving process (source: [20]).

.

For the Certificate Issuance process, the agent constructs a PKCS#10 Certificate Signing Request (standardized in RFC-2986 [24]) that contains the domain name and a specified public key; this CSR also includes a signature by the private key corresponding to the public key in the request. The whole CSR is signed by the agent with the authorized key.

When the Let's Encrypt CA receives the request, it verifies both signatures. If everything looks good, it issues a certificate for that domain with the public key from the CSR and returns it to the agent (figure 2.25).

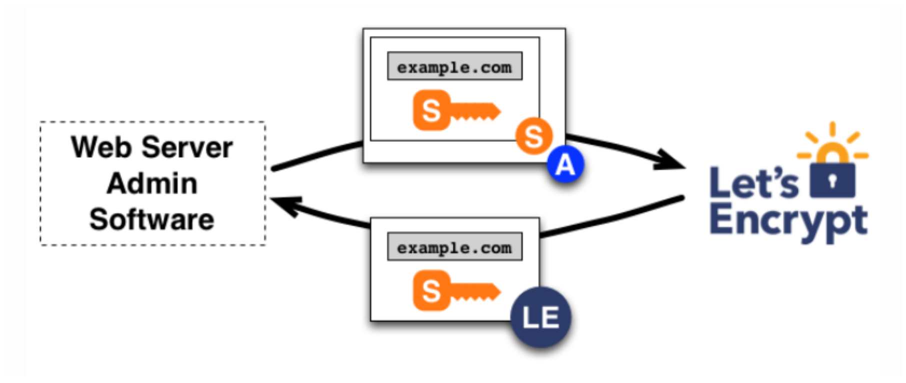


Figure 2.25. Let's Encrypt certificate issuing process (source: [20]).

.

The Certificate Revocation process works in a similar manner. The agent sends to the CA a revocation request, signed with the authorized key. After verifying that the request is authorized, the CA publishes revocation information into the dedicated CRLs or OCSP servers (figure 2.26).

2.8 Certificate Transparency

According to RFC-6962 [25], Certificate Transparency is defined as an open global auditing and monitoring system, based on public logs containing all the certificates issued by CAs. This protocol allows domain owners to verify that no fraudulent certificates have been issued for their domains, while the duty of web clients is to only accept publicly logged certificates. Using CT, it is impossible for a CA to issue a certificate for a domain without it being publicly visible.

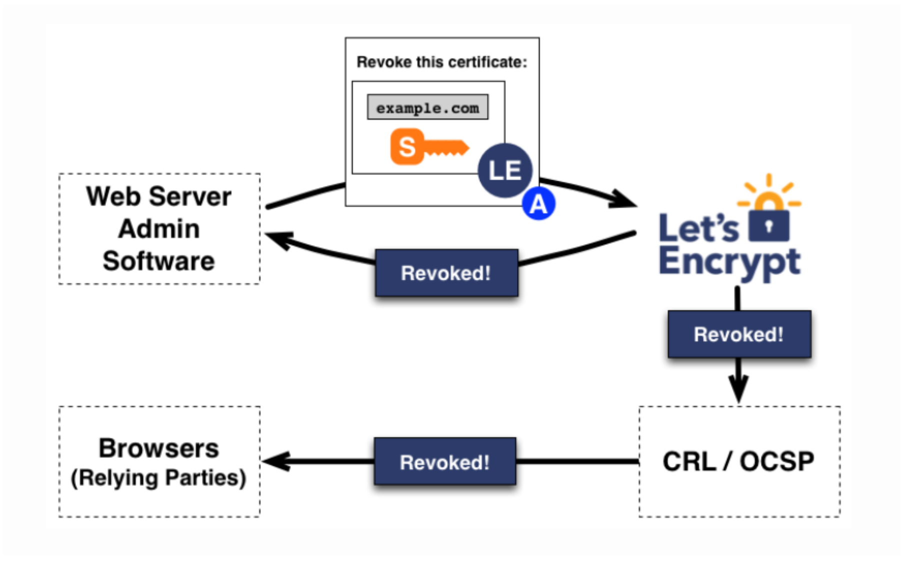


Figure 2.26. Let's Encrypt certificate revocation process (source: [20]).

.

CT has been initially standardized as an experimental protocol, so it doesn't have a precise configuration and can be structured in a number of different ways. However, there are four main actors that are always present in every configuration. What can be changed is the way these actors interact with each other.

2.8.1 CT actors

The CT framework is composed of several actors, each one with a different role:

- Loggers
- Submitters
- Monitors
- Auditors

Loggers Also known a log server, they are the core of the CT system, since they contain a secure log of all issued certificates. The process of adding a certificate to a log server works in an append-only way: a certificate can only be added to the end of a log server, and it is impossible to delete or modify a certificate already present in the log, or retroactively inserting one. Log servers are cryptographically assured in order to prevent tampering and misbehaviour, while also being publicly auditable, in order to verify if they are behaving in the right way or simply to check if a specific certificate is correctly present in them. In order to provide their authenticity, logs are digitally signed. In particular, logs are structured as a tree of certificates and every time a new entry is placed into them, they compute a signature over the hash of the tree itself, which is referred to as the Signed Tree Head (STH).

Submitters They are the actors that submit certificates to the CT log servers. Anyone can submit a certificate to a log server, even if most of the times this process is dome by CAs and server operators. After this operation, the submitter is provided by the log server with a Signed Certificate Timestamp (SCT), which consists in a promise to log the certificate within a certain amount of time.

Monitors They consist of private or public services that are in charge of detecting if a log is misbehaving or if it contains suspicious certificates. Monitors periodically download information from log servers, inspecting every new entry, while also using older log versions to verify consistency between them and the new ones

Auditors They mainly perform two operations:

- They check the overall integrity of the logs by periodically fetching and verifying log proofs, which are signed cryptographic hashes used by logs to prove that they are in good standing.
- They verify the presence in the log of specific certificates. Since every certificate is required to be registered into a log, its absence flags it as a suspicious one, meaning that every client should refuse a connection to its corresponding website.

2.8.2 SCT delivery methods

There are three different methods (figure 2.27) for an SCT to be delivered along with a certificate:

- X.509v3 extension
- TLS extension
- OCSP stapling

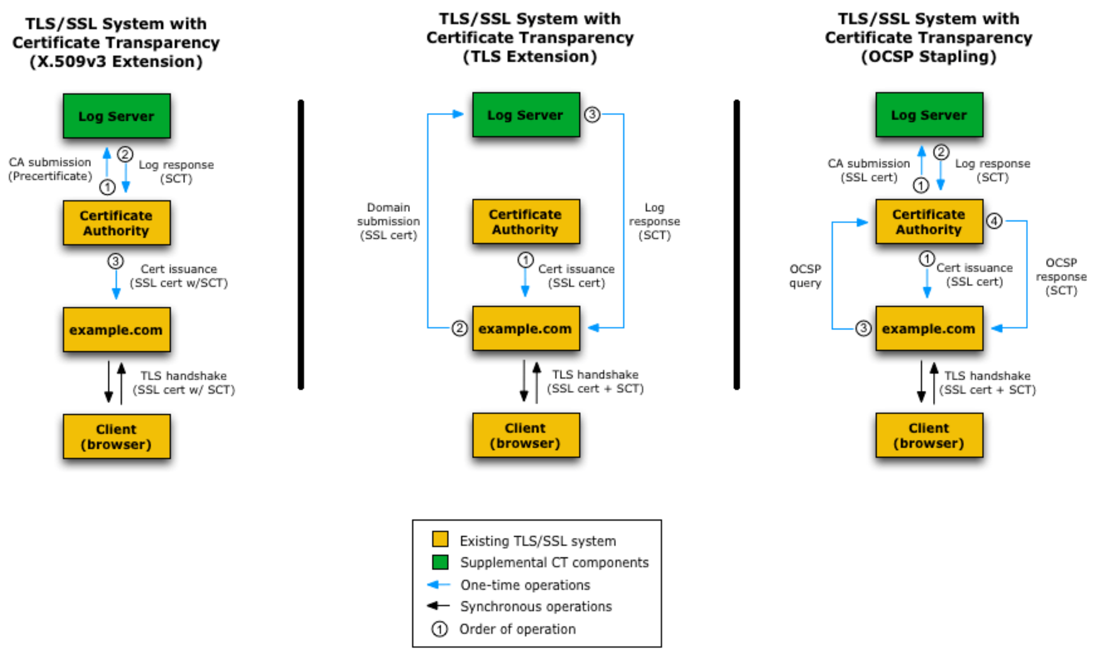


Figure 2.27. SCT delivery methods (source: [26]).

.

SCT via X.509v3 extension This method begins with the submission by a CA of a precertificate to the log server, which returns an SCT to the CA. The CA then attaches the SCT to the precertificate as an X.509v3 extension, signs the certificate and delivers it to the server operator.

SCT via TLS extension In this method, a CA issues a certificate to the server operator, which then submits it to the log server. The log server sends the SCT to the server operator, which then will use the TLS `signed_certificate_timestamp` extension to deliver the SCT to the client during the TLS handshake process.

SCT via OCSP stapling This method consists in making a CA provide the certificate both to the log server and the server operator. The log server sends the SCT to the CA and the server operator performs an OCSP query to the CA. The OCSP response provided by the CA to the server operator will contain the SCT, which can be provided to the client in the OCSP extension during the TLS handshake process.

2.8.3 Example of a CT configuration

CT actors can interact with each other in a variety of different ways, exchanging information useful to each other's jobs. There is no such thing as standard configuration when talking about CT. However, some configurations are more popular than others, like the one depicted in figure 2.28, proposed by Dr. Douglas Stebila [27].

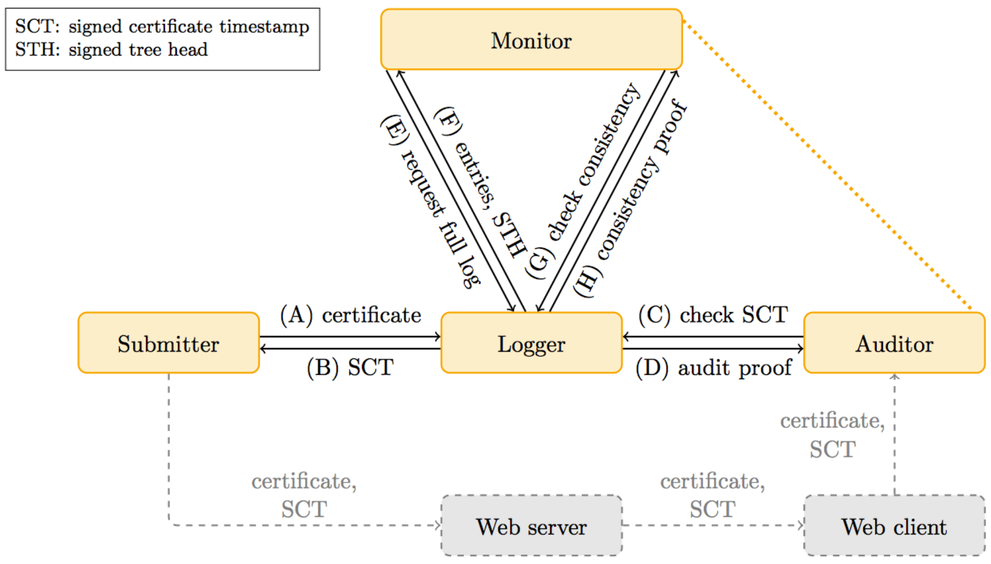


Figure 2.28. Stebila's configuration (source: [27]).

.

These are the interactions between CT actors in Stebila's configuration:

- A: the submitter provides the logger with a certificate;
- B: the logger responds to the submitter by providing it with an SCT, which will then be provided by the submitter to the server operator, along with the certificate; the server operator provides the certificate and the SCT to every client which tries connecting to it;
- C: the auditor, working on behalf of a web client (e.g., a browser), queries the logger in order to check the presence of the certificate corresponding to the SCT;
- D: the logger provides the auditor with audit proof;
- E: the monitor performs a full log request to the logger;
- F: the logger provides the monitor with the log entries and the STH;
- G: the monitor asks the logger to provide consistency proof;
- H: the logger provides the monitor with the requested consistency proof.

2.9 Related attacks

2.9.1 Domain Impersonation

After a domain is validated and a certificate is issued to it by a CA, it is possible for a user to establish a secure connection with it. When connecting to a domain with a valid certificate (along with its whole certification path), the browser usually shows a lock icon next to the website URL, which visually informs the user that the connection is secure. Domains that don't own a valid certificate are marked by browsers as not secure, and connection with them is strongly discouraged.

While validating a domain, a CA checks if the requester of the certificate is actually in control of said domain. The reason why this is done is because issuing a certificate to someone who is not the real owner of the domain would be an enormous security problem. Unfortunately, this is exactly what some malicious actors aim to do.

By altering the correct flow of the domain validation process, it is possible for an attacker to provide a CA with a PK belonging to a key pair in his possession, while requesting the issuing of a certificate for a website's domain he doesn't own. If the CA falls for it, the attacker will be provided with a certificate proving that he is the owner of the domain. After this, the attacker can easily create a fake copy of the legit website and use the certificate to acquire credibility, thus performing *domain impersonation*.

After receiving a false – but valid – certificate, a browser shows the user the lock icon, deceiving him into believing that the connection has been established with a legit domain. From that moment on, every single information that the user inserts into the website is at risk.

2.9.2 DNS cache poisoning

This attack, also referred to as DNS spoofing, has the goal of corrupting the records present into a DNS resolver's cache, causing it to return an incorrect result to the victim, for example a false association between a domain name and an IP address. In this way, a malicious actor is able to lure the victim into a server operated by him, which usually consists in a replica of the website that the victim was originally trying to reach.

In order to enhance the performance of the DNS protocol, some servers store in their cache the resolutions of some DNS queries for a certain period of time. The goal of this attack is making the server store a false resolution, so that when it is queried by the victim, it provides him with a *poisoned* record (figure 2.29).

As we know, DNS data is not protected by any security feature by standard, and the only form of authentication that a query has is its 16-bit `ID` value: each response must have the same `ID` of the corresponding query, otherwise they are discarded by the receiver. This attack [28, 29] usually exploits the vulnerability of the most used DNS server on the internet, which is BIND (Berkeley Internet Name Domain). In the 90s, performing this attack was very easy, since BIND had a very severe vulnerability: its generation of `ID` numbers was sequential and not randomized, making guessing the `ID` for the next query trivial. This attack was carried out by simply providing the victim server with a spoofed query followed by a spoofed answer, in order to poison the cache by inserting a fake record.

During the years, BIND was updated in order to fix most of the vulnerabilities that allowed this kind of attack, but this didn't make it completely resistant. The transaction `ID`s used by BIND were not sequential anymore, but randomized. However, a form of attack was still possible: it was discovered that BIND would send multiple simultaneous recursive queries for the same IP address, hence making this system vulnerable to the birthday paradox phenomenon. Because of this, it is possible to perform a cache poisoning by only sending to the server a few hundreds of spoofed requests and spoofed replies.

In detail, when a certain number of DNS clients send simultaneous requests to the same server, all asking to resolve the same domain name, said server will forward the requests to other servers,

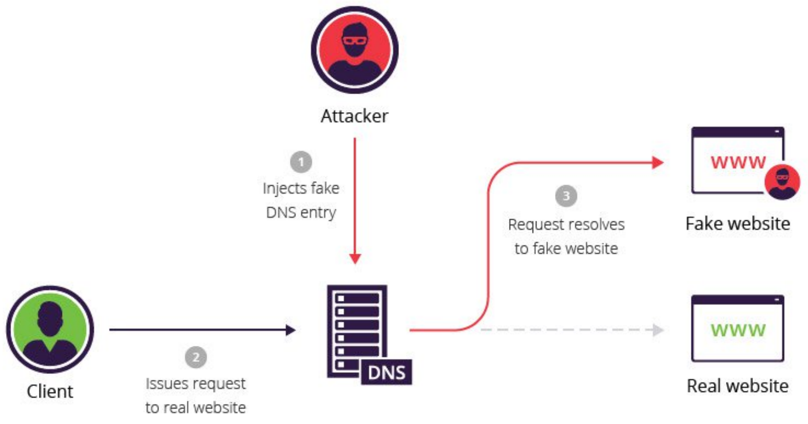


Figure 2.29. DNS spoofing schema (source: [30]).

.

waiting for them to reply with information useful for the resolution. If an attacker sends a number of requests, each one asking to resolve the same domain, but all with a different IP source address, the victim DNS server will forward all the requests to other servers, assigning to each one of them a different ID, and waiting for responses. At this point, the attacker sends several spoofed replies to the victim server, each one with a different ID, attempting to guess at least one of the expected replies' IDs. If this happens, the spoofed reply with the ID matching the one of one of the queries will be accepted as good, the record will be stored in the server and the cache will be poisoned (figure 2.30).

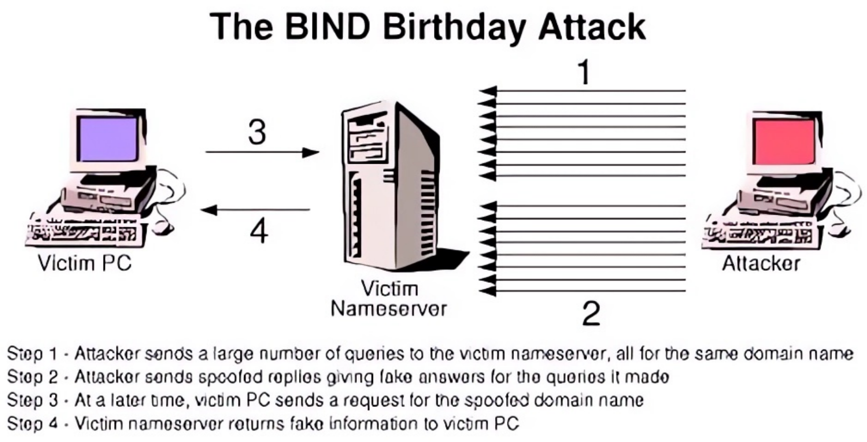

The BIND Birthday Attack
1
3
4
Victim PC
Victim Nameserver
2
Attacker
Step 1 - Attacker sends a large number of queries to the victim nameserver, all for the same domain name
Step 2 - Attacker sends spoofed replies giving fake answers for the queries it made
Step 3 - At a later time, victim PC sends a request for the spoofed domain name
Step 4 - Victim nameserver returns fake information to victim PC

Figure 2.30. BIND poisoning attack schema (source: [28]).

.

Conventional spoofing would mean sending n spoofed replies for one query, and the probability of guessing the correct ID is $n/65535$. By exploiting the birthday attack, so sending n spoofed replies for n different queries, the probability of success is determined by the formula

$$P = 1 - (1 - \frac{1}{t})^{\frac{n(n-1)}{2}},$$

where t is the total number of possible combinations. By using only 700 spoofed requests and replies, the probability of success reaches 97.7% (figure 2.31).

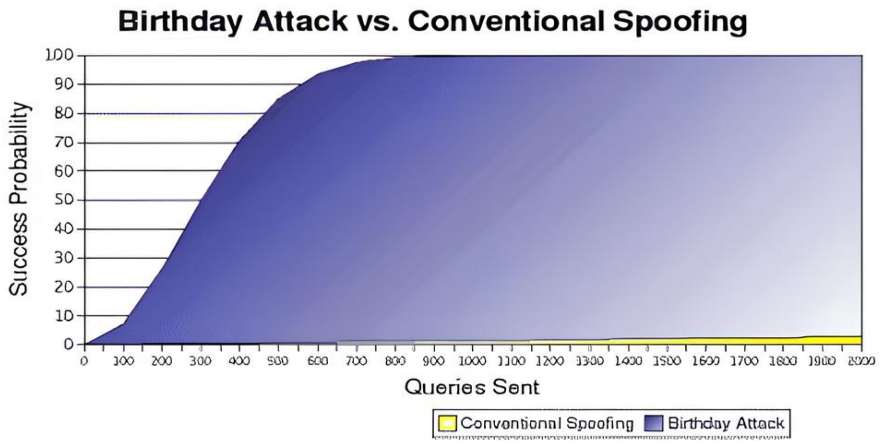


Figure 2.31. Comparison between conventional spoofing and birthday attack (source: [28]).

.

Since this vulnerability is bound to the limited size of the ID field in a DNS message, there is no real solution to be implemented to fix it. Nowadays, the only two ways that could mitigate attacks like this are port randomization, which will increase the number of spoofed messages to be sent, and the use of DNSSEC to provide response authentication.

Chapter 3

Related Works

3.1 Domain Impersonation is Feasible

Schwittmann et al. [1] demonstrate how multiple CAs present one or more vulnerabilities that allow a malicious actor to request and obtain a certificate for a website he doesn't own. The wrongful issuing of these certificate is due to the fact that most CAs still rely on a combination of insecure protocols, like DNS and HTTP instead of DNSSEC and TLS.

This article presents a complete definition of different kinds of attackers, along with the multiple methodologies used by CAs to perform domain validation.

The two types of attackers are (figure 3.1):

- *Off-path attackers*, which have the capability of spoofing IP packets with a source address claiming to originate from the rightful domain owner, while not being able to see the network traffic between the CA and the domain owner's server;
- *On-path attackers*, which are capable of passive eavesdropping or performing an active man-in-the-middle attack.

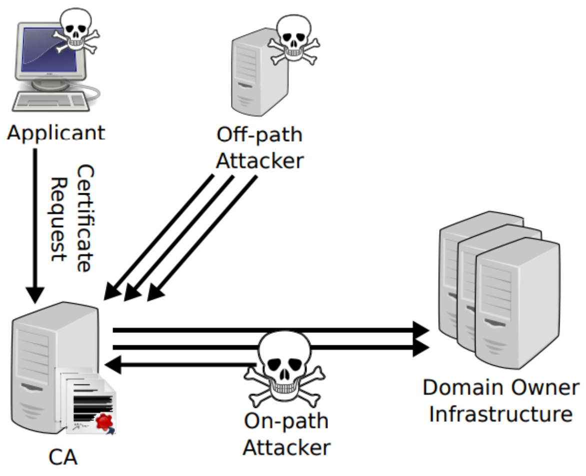


Figure 3.1. Difference between off-path and on-path attackers (source: [1]).

.

Validation methods should consist of out-of-bands techniques such as fax, SMS, phone, or postal mail. These methods are considered safe, but they are rarely used in practice. This study shows how none of the considered CAs offered such methods to perform domain validation. Instead, the techniques used by the CAs were:

- *DNS Change*: the CA generates a random token and instruct the applicant to publish it in the DNS zone file as a TXT, CNAME or CAA resource record (figure 3.2);

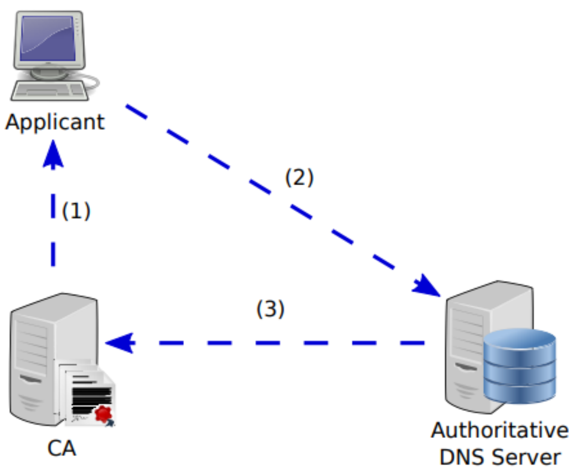


Figure 3.2. DNS-based validation (source: [1]).

.

- *Agreed-Upon Change to Website*: the CA generates a random token and instructs the applicant to publish it under a specific URL within the domain (figure 3.3);

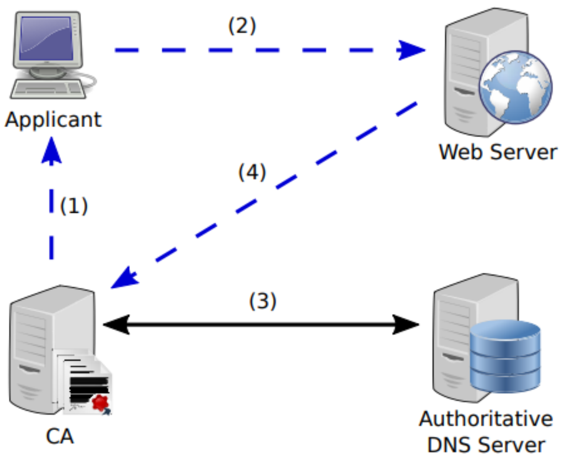


Figure 3.3. HTTP-based validation (source: [1]).

.

- *TLS Using a Random Number*: the CA generates a random token and instructs the applicant to generate a TLS certificate containing that value and serve it on that domain (figure 3.4);

- *Email to Domain Contact*: the CA sends a random token via email to the email address stored in the domain's WHOIS record; the applicant has to submit this token, usually via a website (figure 3.5);

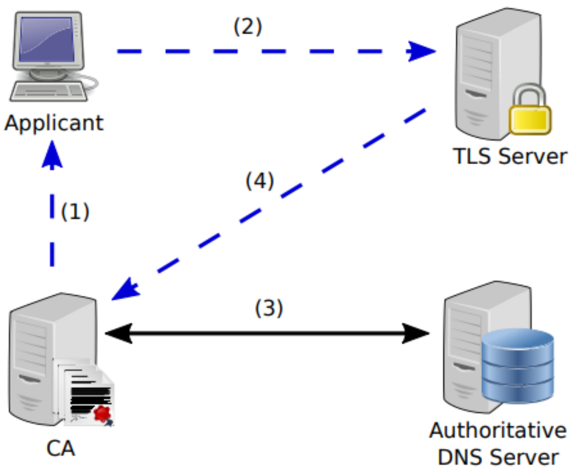


Figure 3.4. TLS-based validation (source: [1]).

.

- *Constructed Email to Domain Contact*: the CA sends a random token via email to an email address constructed by using “admin”, “administrator”, “webmaster”, “hostmaster” or “postmaster” `@ domain` (figure 3.5).

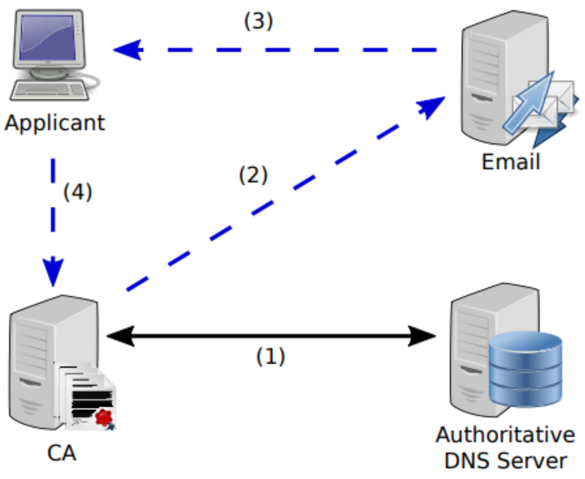


Figure 3.5. Email-based validation (source: [1]).

.

If the CA uses a DNS-based validation technique, it is possible to perform an on-path attack by sending back to the CA a spoofed reply, tricking the CA into thinking that it was sent by an authoritative DNS server (figure 3.6). To mitigate this attack, it can be useful to perform a *DNS multipath* technique, using redundant queries sent from different vantage points.

HTTP-based validation is vulnerable to man-in-the-middle attacks. Moreover, some CAs allows the applicant to choose between HTTP or HTTPS for this process, which is considered a seriously dangerous practice to implement (figure 3.7). In this case, performing multiple HTTP requests from multiple vantage points is considered a good countermeasure (*HTTP multipath*).

For email-based validation methods, an attacker can provide the CA with a spoofed response, impersonating an authoritative DNS server. In this way, the CA is tricked into providing the

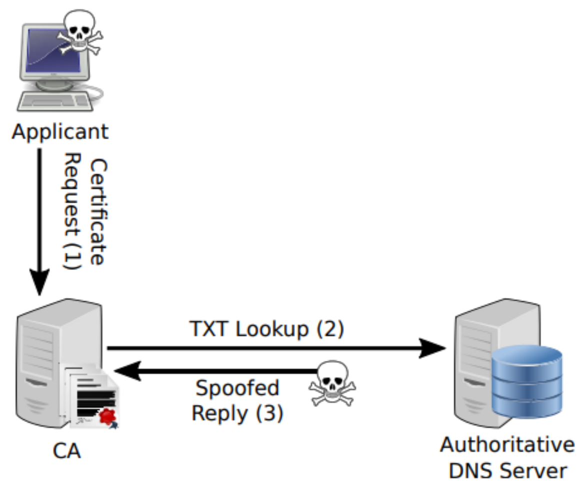


Figure 3.6. DNS-based attack (source: [1]).

.

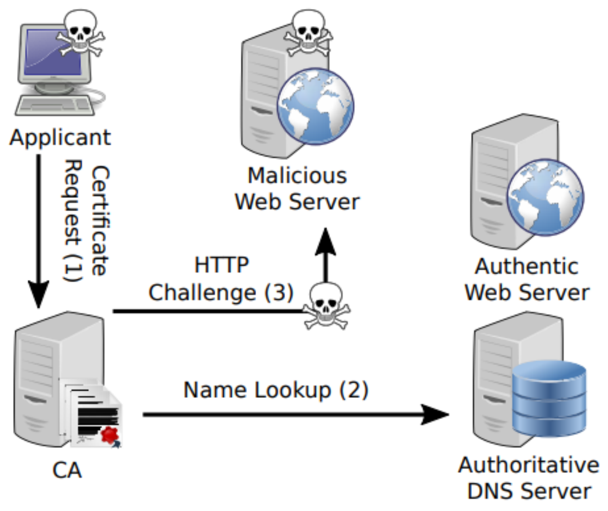


Figure 3.7. HTTP-based attack (source: [1]).

.

email challenge to a malicious email server, instead of the authentic one (figure 3.8).

The results of this study show that all 16 of the tested CAs show at least one vulnerability that could make domain impersonation doable. 5 out of 16 of these CAs issue certificates for free, while the other 11 were paid services.

3.2 Retroactive Identification of Targeted DNS Infrastructure Hijacking

Akiwate et al. [2] perform a study on a class of attack in which an adversary has obtained the capability to manipulate a target domain's DNS configuration. This capability is usually obtained by compromising the domain holder's account with its registrar or compromising the registrar itself. Using this capability, an attacker can temporarily divert a domain's traffic in order to pass the domain validation check performed by CAs and obtaining a certificate.

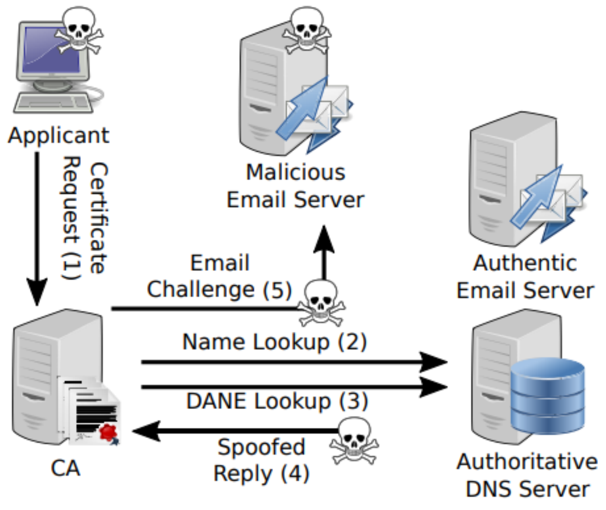


Figure 3.8. Email-based attack (source: [1]).

.

Such attacks are particularly challenging to detect, since they can be very short lived, while also being able to bypass the protection of TLS and DNSSEC, making them imperceptible to users. This paper focus on identifying them retroactively, with the help of various records, such as Certificate Transparency logs. In this way it is possible to check the presence of multiple suspiciously deployed infrastructure for the same domain during its history. Such deployments strongly correlate with hijacks attempts and can be visualized by building a *deployment map*.

This is a very sophisticated category of attacks, for which the malicious actor needs a lot of resources. The main victims are government agencies.

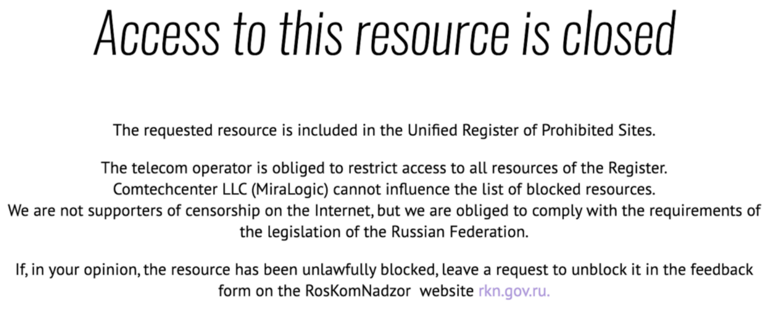
3.3 CERTainty: Detecting DNS Manipulation at Scale using TLS Certificates

Tsai et al. [3] discuss the topic of DNS manipulation performed by dictatorship's governments when enforcing censorship on websites' contents. In particular, the authors propose a new method called *CERTainty*, used to precisely identify when access to a domain is prevented by means of a blockpage. This technique was developed since the older ones had a 70% false positive rate.

CERTainty is a blockpage (figure 3.9) identification methodology based on certificate analysis and it works by arranging the domain's certificates into different categories. Depending on the category it belongs to, a website has a lower or higher probability of being blocked by a blockpage. The categories are:

- Domains with valid certificates, which are considered as not manipulated;
- Domains with untrusted certificates with matched hostnames, which are considered as potential DNS manipulation;
- Domains with trusted certificates with mismatched hostnames, which are considered as potential DNS manipulation;
- Domains with untrusted certificates with mismatched hostnames, which are considered as a very strong sign of domain manipulation.

Censorship via DNS manipulation can be performed at different levels (e.g., ISP level, filtering products level), and this technique is successful regardless of the blocking methodology used.



Access to this resource is closed

The requested resource is included in the Unified Register of Prohibited Sites.

The telecom operator is obliged to restrict access to all resources of the Register.
Comtechcenter LLC (MiraLogic) cannot influence the list of blocked resources.
We are not supporters of censorship on the Internet, but we are obliged to comply with the requirements of the legislation of the Russian Federation.

If, in your opinion, the resource has been unlawfully blocked, leave a request to unblock it in the feedback form on the RosKomNadzor website rkn.gov.ru.

Figure 3.9. Example of a blockpage (source: [3]).

.

For example, the Russian Federation enforces websites' censorship mainly via ISP level DNS manipulation, whose decentralized nature is revealed by the 31 unique blockpage fingerprints found by the authors. By checking the blockpages' certificates, it is possible to notice how they mainly consist of self-signed certificates issued by the ISPs themselves. Said certificates are usually badly configured, since they contain no information in all the fields except for the expiration date and country of the issuer ("`RU`").

3.4 You Are Who You Appear to Be

It is not necessary to perform sophisticated DNS manipulation attacks to make domain impersonation feasible; sometimes, an attacker doesn't even need to obtain another domain's certificate to do it. Since a specific website is identified only by means of a URL, it is possible for a malicious actor to obtain a certificate for a domain he owns that looks identical to the impersonated one and has a slightly different URL.

Attackers can trick the victims into believing that they are browsing a legit website by performing some URL manipulation technique, such as:

- Typosquatting (e.g., `youtueb.com`);
- Homographs (e.g., `y0utube.com`);
- Combosquatting (e.g., `youtube-videos.com`).

An attacker can easily be able to obtain a valid certificate for such websites, for example using Let's Encrypt. Since the certificate is valid, when visiting the website, the browser shows the victim the lock icon, which tricks him into believing that what he is seeing is the legit website and not a malicious replica. At this point, the victim probably won't even notice the counterfeit URL.

Roberts et al. [4] focus on an impersonation attack similar to the ones previously described called target embedding, which is part of the broader "subdomain spoofing" category. The main difference between this attack and the ones mentioned before consists in the domain level at which they are performed: typosquatting, homographs and combosquatting are located in an effective second-level domain (e2LD), while target embedding happens at subdomain level.

In this attack, the target website's domain name is not modified, but it is embedded as a subdomain of the actual domain. For example, `apple.com-signin.id` embeds the target domain `apple.com`, but the actual domain is `com-signin.id`. This is a real domain and Let's Encrypt released a valid certificate for it in 2018.

When forging these URLs, attackers usually use unpopular TLDs (`.ga`, `.ml`, `.cf`, `.tk`, and `.gq`), since they allow anyone to register domains under them for free. Moreover, these attacks have become more frequent with the rise of CAs like Let's Encrypt, mainly because of the fast and free certificate issuing process offered.

3.5 Off-Path Attacks Against PKI

Shulman et al. [5, 6] show how an off-path attacker can alter the correct domain validation process to obtain a certificate by a CA for a domain he doesn't own, in order to subsequently impersonate it.

The attack works by making sure that, during the domain validation phase, the CA partly uses the attacker's DNS server, instead of the one owned by the legit domain owner. The attacker crafts a malicious DNS response and sends it to the CA before the authentic response from the real nameserver arrives.

In detail, the attack depends on getting a DNS response broken into fragments, and then injecting malicious data to fool the CA into handing over the certificate to the attacker. The first of these fragments contain valid DNS challenge-response fields, while the other parts contain what is needed for the attacker to receive the certificate. However, this attack requires the malicious actor to do some prior offline work and to examine the victim's nameserver responses in order to calculate the exact offset to perform the fragmentation.

As a countermeasure, the authors propose a new validation protocol called DV++, which exploits a distributed model in which domain owners must prove their ownership to multiple agents before receiving a certificate. All of this is done in a fully automated manner.

Chapter 4

Censorship Detector

Having control over a DNS response, makes it possible for a malicious actor to provide a victim with a resolution containing an incorrect IP address, with the goal of luring him to a different webpage. However, this type of DNS attacks is not performed only by hackers with the purpose of stealing credentials or private information. Instead, the malicious actor in question can also be a bigger entity.

This work will focus on a particular kind of DNS response manipulation, which has the goal of preventing the victim from accessing a legit domain for which a resolution has been asked. This is done by providing a user with a fabricated resolution with the purpose of redirecting him to a different page instead of the one he was looking for, blocking any access to the original domain.

In this specific case, the malicious actor implementing the attack consists in a country's government.

4.1 Censorship via DNS Manipulation

Among other reasons, DNS manipulation is used to enforce web pages censorship on the internet. By exploiting the original nature of DNS protocol, which doesn't include authentication for DNS responses, a resolver owned by a malicious actor could return a counterfeit record to a client. In this kind of scenario, such record would contain a false resolution, which will redirect the user to a different domain, instead of the one he was originally looking for.

This technique is proven to be adopted in some countries with authoritarian governments, mainly with the goal of hiding truthful sources of information. The hiding is typically performed by means of a blockpage, which will completely replace the webpage that the victim is trying to reach. Since the DNS architecture is hierarchical and distributed, there is no specific way of implementing this attack. However, considering the scenario just mentioned, the manipulation is usually carried out at ISP level: as a matter of fact, dictatorships usually have full control over all of the country's ISPs, hence controlling also their DNS resolvers.

If not applied at ISP level, this type of censorship can also be implemented by means of a commercial filtering product. Such products are able to detect when a user tries to connect to a specific website, performing an automatic redirection to a customizable blockpage. These technologies are not only developed in authoritarian countries, but also exported by them as a commercial product for them to be used in other countries.

This type of manipulation is not only performed for malicious censorship reasons: for example, this technique can be used in schools' private networks, in order to hide websites that shouldn't be visited by underage students. However, even if implemented with good intentions, this operation still consists in providing a user with a poisoned DNS response by exploiting one of the biggest vulnerability of the DNS protocol (responses not being authenticated).

It is good to point out that in countries where freedom of information is stifled, this kind of censorship is implemented with the excuse of providing some sort of protection from websites defined as dangerous. This false protection narrative is carried out by multiple countries that implement censorship via DNS manipulations, the main ones being the Russian Federation and the People's Republic of China.

4.1.1 China's Great Firewall

China implements the most extended and long lasting internet control policies in the whole world, which already 20+ years ago was reported to enforce censorship for tens of thousands of websites [31]. The categories of blocked websites span from social medias to news sources, and most of them consist in the most visited websites in the world. The list of legislative actions and technologies used by the Chinese government to implement censorship is referred to as the Great Firewall (GFW).

The Chinese government openly admits that the GFW's role is to censor contents, stating that the websites that are being blocked are the ones that endanger the nation's security. This results not only in the censorship of simple website's domain, but also in the blocking of all of the most common search engines in the world, including Google and DuckDuckGo, while Bing is replaced with its Chinese version (`cn.bing.com`). The complete list of all the censored domains in China is not static and it constantly changes over time.

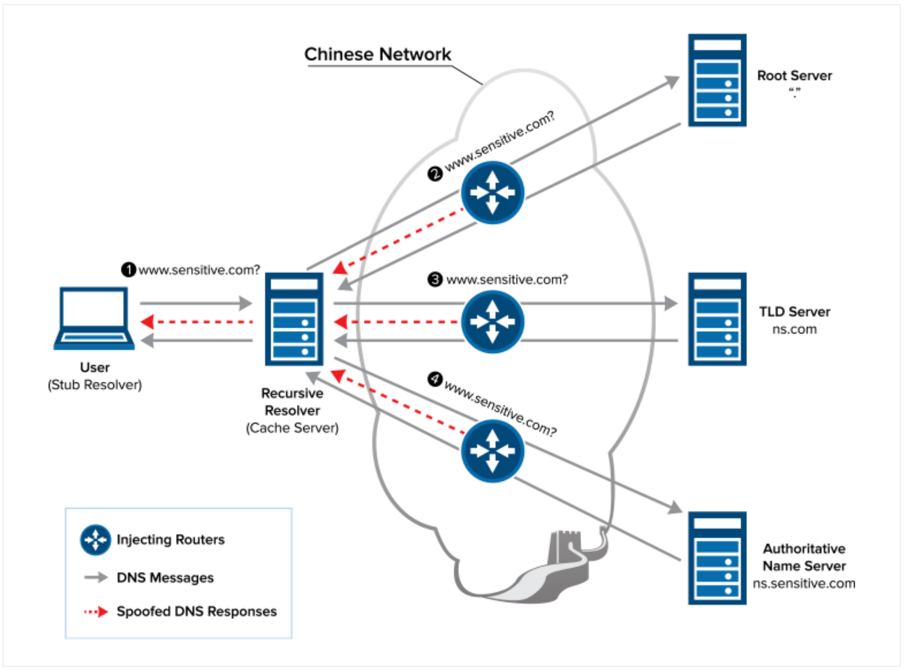


Figure 4.1. How China's GFW manipulated DNS responses (source: [32]).

.

By the technical point of view, websites are hidden with various techniques, with DNS response manipulation being the main one (figure 4.1). Other methodologies implemented to censor domains are:

- Black holing some IP ranges, hence discarding all the traffic going in their direction;
- Filtering specific URLs by means of transparent proxies, whose presence is not known to the user;
- Implementing a packet loss mechanism, hence causing connections with specific websites to time out on the client side;
- Performing a TCP reset attack, by sending a forged TCP RST packet to the user which terminates the TCP connection;

- Implementing a Man-in-the-middle TLS attack, which is possible thanks to the fact that the Chinese government is allowed to request and use the root certificate of every Chinese authority.

It is important to point out that foreign DNS resolvers (such as Google ones) are reported to be perfectly functional in the country. However, these resolvers are also subject to hijacking, since their connections are not encrypted: DNS queries do reach the servers, but if the requests contain banned keywords, the firewall infrastructure injects fake DNS replies, which anticipate the legit ones.

Even if governments of other non-democratic countries also do implement domains blockage in various ways, China's GFW is by far the most complex and organized internet censoring infrastructure in the world.

4.2 CERTainty

As seen in the previous chapter, the topic of DNS manipulation with the goal of censoring websites has been extensively discussed by Tsai et al. [3], along with the proposal of a method to identify when and how often censorship is performed on the internet. This led to the development of *CERTainty* (figure 4.2), which consists of a tool whose workflow is described both in the aforementioned article and also on the Censored Planet website [33].

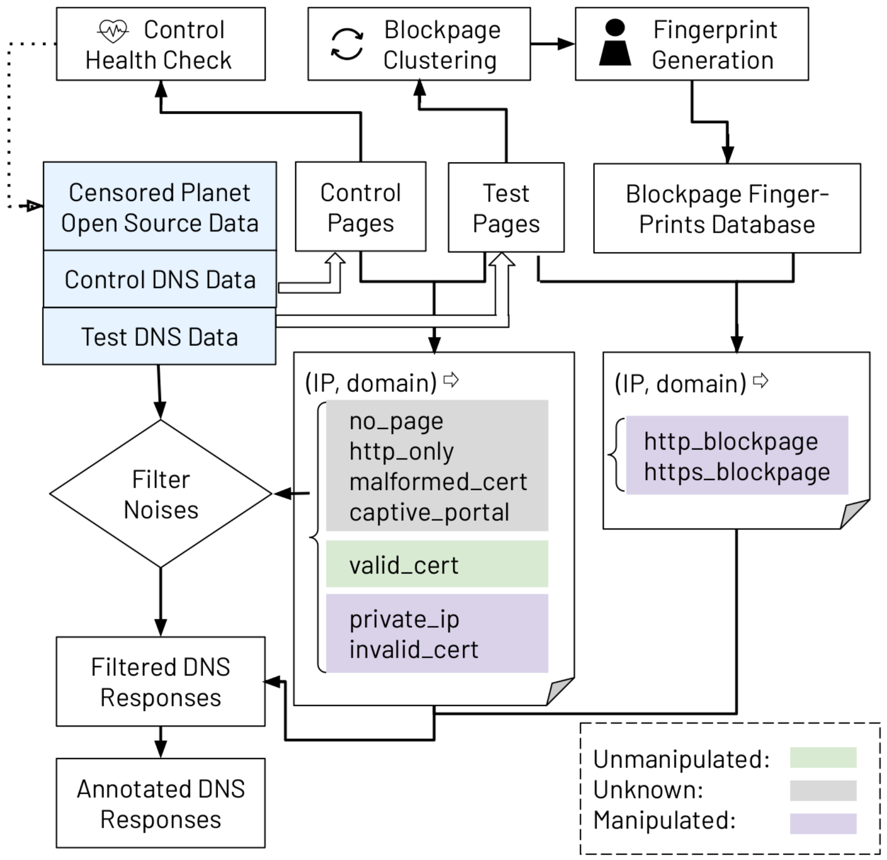


Figure 4.2. *CERTainty*'s workflow (source: [3]).

.

The peculiarity of this technique is the fact that it analyses domain's certificates in order to extract information that could indicate the presence of manipulation and censorship. This method was developed since previous techniques detected 72.45% of DNS manipulation cases to be false positive, while also failing to detect 9.70% of the true cases detected by *CERTainty*. This tool allowed the developers to 17 DNS filtering products used in 52 countries, along with 55 autonomous systems that performed ISP level DNS manipulation in 26 countries. Moreover, 226 new blockpage clusters were retrieved and their fingerprints were published in open-source databases.

CERTainty is currently one of the measurement techniques used by Censored Planet, which is a censorship measurement platform whose goal is gathering information about internet censorship in more than 200 countries.

The analysis performed using *CERTainty* starts by comparing the DNS resolutions performed by a set of trusted control resolvers (such as Google or Cloudflare) to a test list of resolutions generated by 25943 resolvers for 2098 domains. The domains list is a combination of the 500 top domains present in the Tranco top 1 Million list [34, 35] combined with the Citizen Lab Global Test list [36].

As mentioned before, what makes *CERTainty* different from other censorship detection tools is the analysis it performs over websites' certificates. A certificate is considered valid by the tool if:

- The root of its certification chain is present in the Mozilla NSS Root Store by OpenSSL [37];
- The common name or one of the subject alternative names in the certificate matches the name of the domain that the tool Is trying to reach.

This creates four distinct cases for the tool to analyse, each one indicating a higher or lower probability for the domain to be censored:

- A domain with a valid certificate is considered as a strong sign of non-manipulation, flagging the domain as not censored.
- A domain whose certificate has an untrusted root in its certification chain but shows no mismatch between its name and the name of the certificate is considered as a sign of potential manipulation. This study has observed that 86.25% of the domains in this category were hidden with a blockpage.
- A domain whose certificate is trusted but presents a mismatched hostname in its certificate compared to the real name of the domain is considered as a potential sign of manipulation. The study has observed that 10.48% of the domains in this category matched a blockpage fingerprint. It is important to notice that for the requests that returned a 4XX status code, 98.66% of them are returned by Chinese open resolvers. This technique is the preferred one for implementing censorship in China: the name on the certificate of websites blocked in China typically corresponds to large US-based companies, such as Facebook or Cloudflare.
- A domain with an untrusted root in its certificate's certification chain and with a mismatched hostname compared to the real domain name is considered as a strong sign of manipulation. The study has observed that 92.31% of the domains in this category are censored by means of a blockpage. For requests that returned a 4XX status code, most of their certificates were issued by ISPs based in countries like Russia or Singapore.

Moreover, the tool accounts also for certificates that would be invalid even in a control setting, mainly for misissuance reasons. To confirm that the origin of such certificates comes from the IP address received during the DNS resolution, the tool performs a TTL-limited traceroute test for such domains. This test consists of performing two TLS Hello requests for the control domain and the target domain, then comparing both traceroutes in order to determine where in the network the TLS response is originating from. If both of them terminate in the same /24 subnet, then it is safe to say that the certificates are returned by the IP obtained during the DNS resolution.

The developing of this tool provided a big aid to organizations performing worldwide internet censorship auditing, confirming the presence of ISP level DNS manipulation not only in non-democratic countries, but also in some countries that are considered “free” by multiple non-governmental organizations, such as Germany, Greece and Denmark.

This study highlights the importance of integrating both information inferred from blockpages and certificates when treating the topic of internet censorship implemented via DNS manipulation. These heuristics allow to have a complete vision not only about which domains are blocked but also about how the blockage is applied in detail, making it possible to distinguish the unique ways different countries use to censor online contents.

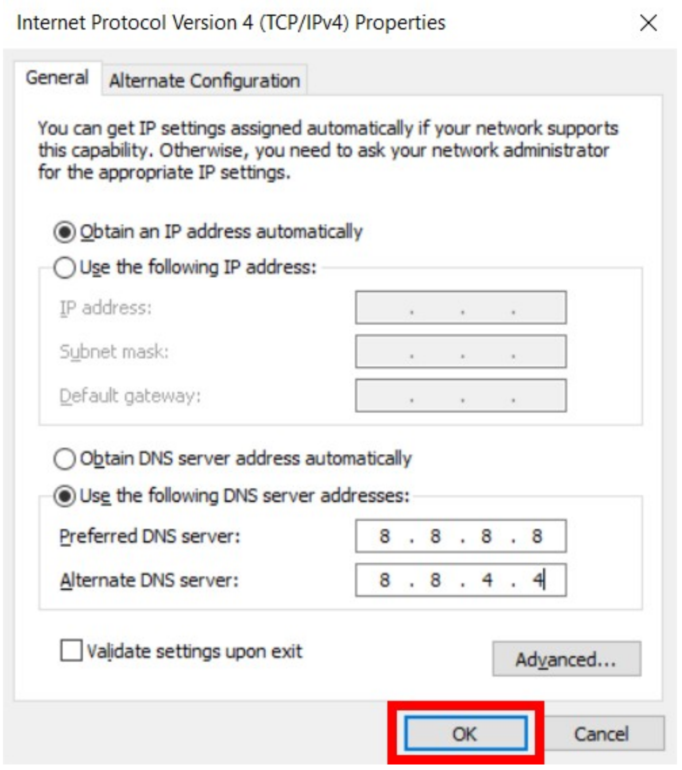
4.3 Experiencing censorship

Before going into details regarding what this work is about, it is important to provide some context about how to recreate an environment which allows to experience the previously described type of censorship.

After connecting to a network, a machine will usually choose the DNS resolvers provided by the ISP that deployed the network and will use such resolvers for every resolution performed from that moment one. Using those specific resolvers is not mandatory and it is possible to change them in order to use the desired ones. Each resolver is identified by a specific IP address (typically IPv4).

4.3.1 Changing the machine’s DNS resolvers

These are the steps to be implemented in order to change the address of the resolvers in use on a Windows 10 machine:


Internet Protocol Version 4 (TCP/IPv4) Properties

General | Alternate Configuration

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

[✓ Obtain an IP address automatically]
Use the following IP address:

IP address:	. . .
Subnet mask:	. . .
Default gateway:	. . .

Obtain DNS server address automatically
[✓ Use the following DNS server addresses:]

Preferred DNS server:	8 . 8 . 8 . 8
Alternate DNS server:	8 . 8 . 4 . 4

Validate settings upon exit

Advanced...

OK Cancel

Figure 4.3. Changing DNS resolvers on a Windows machine.

.

- Open the *Control Panel*;
- Click on *Network and Internet*;
- Click on *Network and Sharing Center*;
- Click on the *Change adapter settings* on the top-left;
- Right-click on the network in use and select the *Properties* option;
- Select and check the *Internet Protocol Version 4 (TCP/IPv4)* option;
- Click on the *Properties* button;
- Select the *Use the following DNS server addresses* option;
- Insert the desired resolvers' IP addresses (figure 4.3).

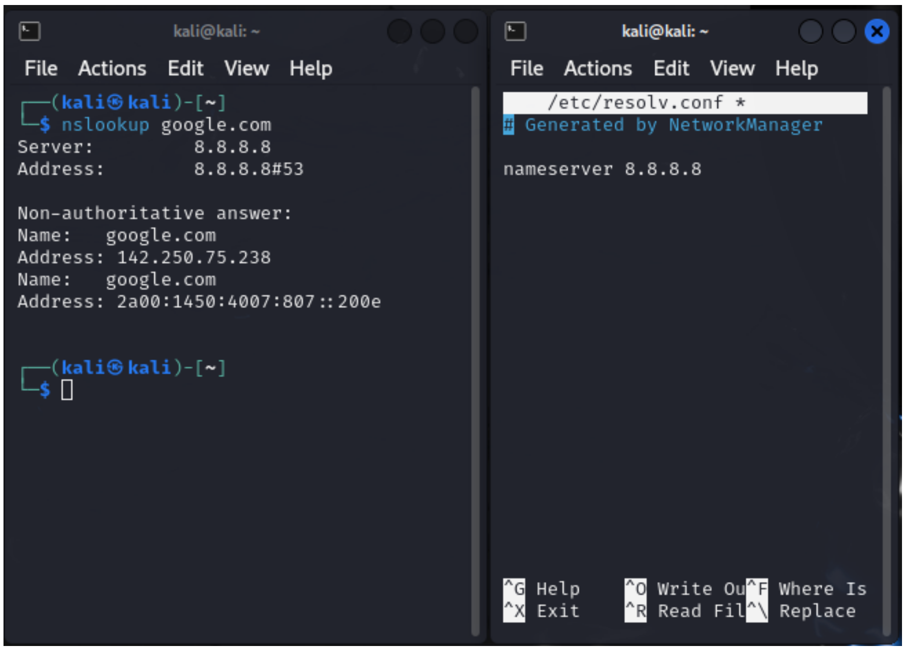
To change the DNS resolvers on a Linux machine, it is necessary to modify the `/etc/resolv.conf` file, which contains the list of the nameservers in use to perform resolutions. The file can be modified using the terminal command:

```
sudo nano /etc/resolv.conf
```

This file contains the list of nameservers in use. In order to specify which resolvers must be used, it is necessary to delete or comment the lines containing the old nameservers and add new lines containing the desired ones. A resolver is specified by simply adding a line in the format:

```
nameserver <resolver_IP_address>
```

After the DNS resolvers are changed, every DNS resolution performed by the machine will be managed by said resolvers. Thanks to this, by choosing a DNS resolver located in one of this work's countries of interest (e.g., China), it is possible to experience censorship in the same way as people actually living in those countries (figures 4.4 and 4.5).



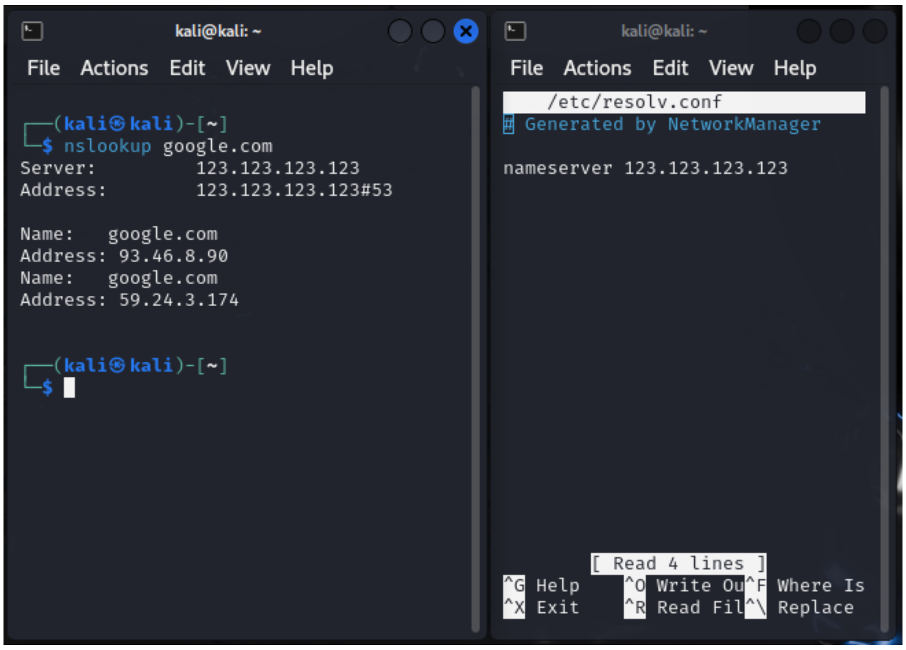
```
┌──(kali㉿kali)-[~]
└─$ nslookup google.com
Server:         8.8.8.8
Address:        8.8.8.8#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.75.238
Name:   google.com
Address: 2a00:1450:4007:807::200e


┌──(kali㉿kali)-[~]
└─$ 
```

```
   /etc/resolv.conf *
# Generated by NetworkManager

nameserver 8.8.8.8








^G Help      ^O Write Ou^F Where Is
^X Exit      ^R Read Fil^\ Replace
```

Figure 4.4. Resolution for `google.com` via 8.8.8.8.

.

It is important to remind that, in order to have an experience faithful to the real one, no VPNs or proxy servers must be used: the first ones protect from DNS manipulations by encrypting all of the machine's sent and received traffic, while the second ones don't give the possibility of choosing a specific DNS resolver, making their use completely pointless.



```
┌──(kali㉿kali)-[~]
└─$ nslookup google.com
Server:         123.123.123.123
Address:        123.123.123.123#53

Name:   google.com
Address: 93.46.8.90
Name:   google.com
Address: 59.24.3.174


┌──(kali㉿kali)-[~]
└─$ 
```

```
    /etc/resolv.conf
# Generated by NetworkManager

nameserver 123.123.123.123

            [ Read 4 lines ]
^G Help      ^O Write Ou^F Where Is
^X Exit      ^R Read Fil^\ Replace
```

Figure 4.5. Resolution for `google.com` via 123.123.123.123.

.

4.3.2 The browsers' problem

At this point, even after perfectly recreating the environment, it is basically impossible to experience censorship by using a common browser. Both in Windows and in Linux, browsers like Google Chrome, Mozilla Firefox and Opera are able to detect when a manipulated response is received when trying to reach a website. When browsers notice this kind of situation, they block the connection entirely (figure 4.6).

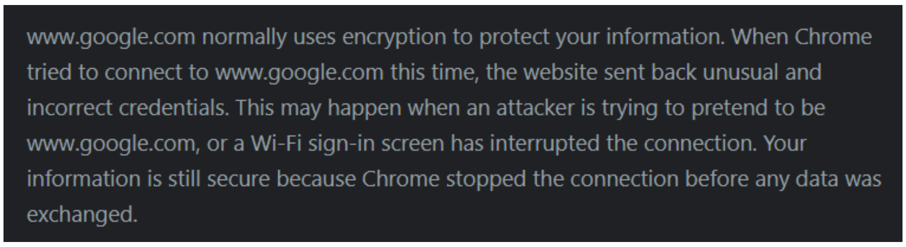

www.google.com normally uses encryption to protect your information. When Chrome tried to connect to www.google.com this time, the website sent back unusual and incorrect credentials. This may happen when an attacker is trying to pretend to be www.google.com, or a Wi-Fi sign-in screen has interrupted the connection. Your information is still secure because Chrome stopped the connection before any data was exchanged.

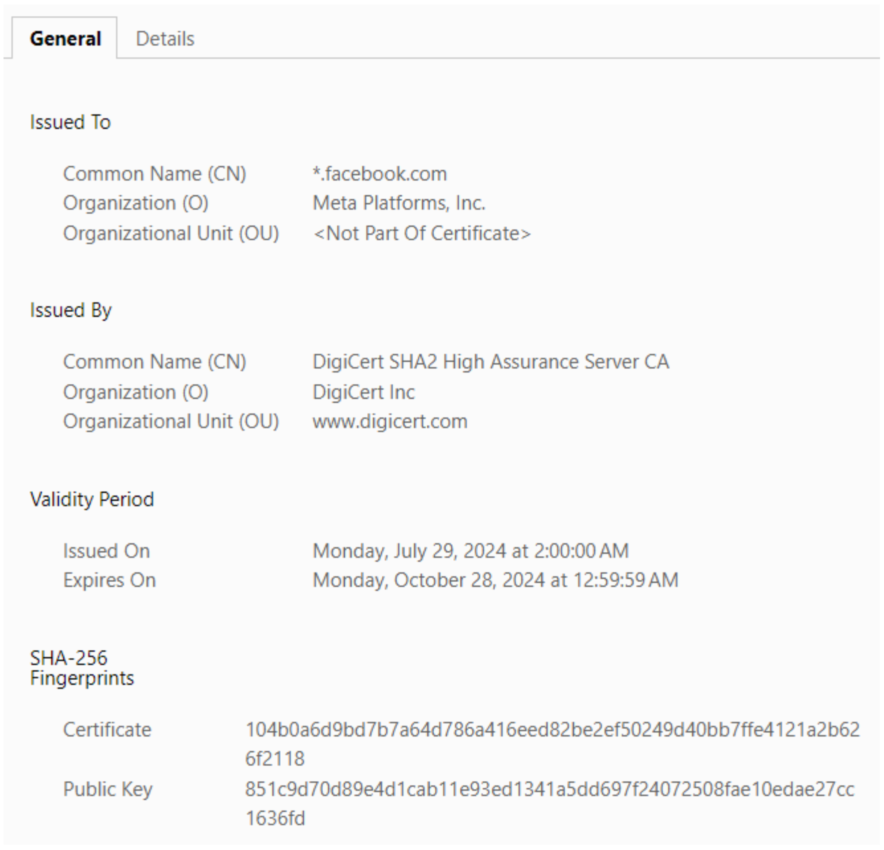
Figure 4.6. Warning shown by Google Chrome when visiting `youtube.com` via 123.123.123.123.

.

Moreover, DNS resolvers located in the countries under consideration are typically slow and have a low availability rate. Connections performed using these resolvers will usually timeout, returning no webpage at all.

However, experience shows that, even if browsers don't show this type of risky webpages, in some of the cases where the resolution is completed successfully, they sometimes provide the domains' certificate, which is useful to understand if and how censorship is performed (figure 4.7).

In conclusion, even in the case of not receiving a timeout, browsers will show no page for security reasons. So, in order to visualize a blockpage, a different technique must be used.


General | Details

Issued To

Common Name (CN)	*.facebook.com
Organization (O)	Meta Platforms, Inc.
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	DigiCert SHA2 High Assurance Server CA
Organization (O)	DigiCert Inc
Organizational Unit (OU)	www.digicert.com

Validity Period

Issued On	Monday, July 29, 2024 at 2:00:00 AM
Expires On	Monday, October 28, 2024 at 12:59:59 AM

SHA-256 Fingerprints

Certificate	104b0a6d9bd7b7a64d786a416eed82be2ef50249d40bb7ffe4121a2b62 6f2118
Public Key	851c9d70d89e4d1cab11e93ed1341a5dd697f24072508fae10edae27cc 1636fd

Figure 4.7. Certificate returned when visiting `youtube.com` via 123.123.123.123.

.

4.3.3 How to visualize a blockpage

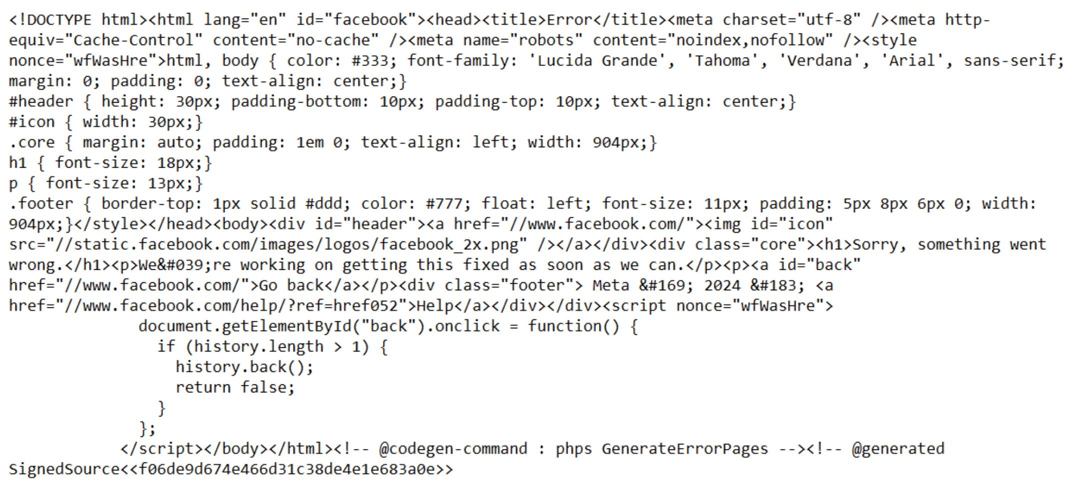
In order to visualize a blockpage, the command line tool `curl` [38, 39] can be used. Such tool allows to transfer data by means of various types of protocols, including HTTP and HTTPS. In this specific case, this command is used to perform an HTTPS connection with a censored website, with the purpose of retrieving the blockpage hiding it.

Among other options, `curl` gives the possibility of running an HTTPS connection in insecure mode, making it possible to visualize webpages that are typically blocked by standard browsers. In particular, the command is run in the following way:

```
curl https://<domain_name> -i -k -L
```

The options `-i`, `-k` and `-L` allow to respectively run the command in informative mode, insecure mode, and following every redirection step until reaching the final webpage. As said before, insecure mode is necessary in order to visualize webpages that wouldn't be normally visible using a browser. Informative mode is used in order to output every single information obtained during the connection, including HTTP status response codes, all redirection steps, and finally the HTML of the webpage (figure 4.8). Redirection must be active since most of the websites are not located at their general domain name (e.g., Google is not located at `google.com`, but at `www.google.com`), so not performing it would mean stopping at the first HTTP response, which in these cases will contain a 3XX response code and no HTML at all.

Even if it allows to perform connections that common browsers don't support, the `curl` tool doesn't solve the timeout problem, which however is impossible to be solved, since in this case it depends only on the quality of the resolvers in use.



```
<!DOCTYPE html><html lang="en" id="facebook"><head><title>Error</title><meta charset="utf-8" /><meta http-
equiv="Cache-Control" content="no-cache" /><meta name="robots" content="noindex,nofollow" /><style
nonce="wfWasHre">html, body { color: #333; font-family: 'Lucida Grande', 'Tahoma', 'Verdana', 'Arial', sans-serif;
margin: 0; padding: 0; text-align: center;}
#header { height: 30px; padding-bottom: 10px; padding-top: 10px; text-align: center;}
#icon { width: 30px;}
.core { margin: auto; padding: 1em 0; text-align: left; width: 904px;}
h1 { font-size: 18px;}
p { font-size: 13px;}
.footer { border-top: 1px solid #ddd; color: #777; float: left; font-size: 11px; padding: 5px 8px 6px 0; width:
904px;}</style></head><body><div id="header"><a href="//www.facebook.com/"><img id="icon"
src="//static.facebook.com/images/logos/facebook_2x.png" /></a></div><div class="core"><h1>Sorry, something went
wrong.</h1><p>We&#039;re working on getting this fixed as soon as we can.</p><p><a id="back"
href="//www.facebook.com/">Go back</a></p><div class="footer"> Meta &#169; 2024 &#183; <a
href="//www.facebook.com/help/?ref=href052">Help</a></div></div><script nonce="wfWasHre">
              document.getElementById("back").onclick = function() {
                if (history.length > 1) {
                  history.back();
                  return false;
                }
              };
            </script></body></html><!-- @codegen-command : phps GenerateErrorPages --><!-- @generated
SignedSource<<f06de9d674e466d31c38de4e1e683a0e>>
```

Figure 4.8. HTML of a blockpage obtained via `curl`.

.

4.3.4 Chinese resolvers' timeout mechanism

As will be discussed later, this work is mainly focused on experiencing and studying how China's censorship looks like when implemented via DNS manipulation. To make it possible, the resolvers that were taken into consideration were some of the most popular Chinese ones [40], mainly owned by China's telecommunication services (such as 210.2.4.8, owned by the China Internet Network Information Center, 180.76.76.76, owned by Baidu, and 123.123.123.123, owned by the China Unicom Beijing Province Network).

The information gathering phase of this work was slowed down by some problems caused by the previously listed resolvers. One of the issues is due to the not perfect availability of such servers, which sometimes cause the connections to timeout, making it hard to obtain the requested information. However, this didn't happen very frequently, since the resolvers used were among the country's best ones.

The biggest issue found in this part of the work consists in the temporary block that such resolvers apply to machines that try to connect to censored websites. In particular, this happens when using `curl` to retrieve the webpages and `openssl` (which will be described in detail later in this work) to retrieve the domains' certificates. After using such commands for a specific domain, it is impossible to perform them again for the same website without bumping into a timeout. This blockage lasts for a pretty long period of time, which spans from some minutes to a few hours.

It is possible that Chinese resolvers implement this timeout-based blocking mechanism depending on the IP address of the user performing the request. Such identifier is probably kept in memory by the resolvers for a certain period of time, with the purpose of killing every future connection requested by a specific user for a specific domain even before they start.

4.4 What is Censorship Detector?

My personal contribution for this work consists in a tool called *Censorship Detector*, whose purpose is to identify if a specific resolver in use is performing DNS response manipulation in order to censor some domains. The realization of this tool was heavily inspired by *CERTainty*, so it is important to clarify that the general methodology behind it was not originally conceived by me.

Even if sharing the same final purpose – detecting censorship based on DNS manipulation – these tools are not identical.

4.4.1 Comparison with CERTainty

Before describing how *Censorship Detector* works in detail, it is necessary to understand the thought process that brought to its realization, along with the contribution that *CERTainty* had in this project.

As seen before, *CERTainty* is currently used by Censored Planet with the purpose of helping in detecting how widespread internet censorship is and in which form it is implemented. So, *CERTainty* is used as an audit tool, and its job consists of analysing tens of thousands of DNS resolvers, located in more than 200 countries. In short, *CERTainty*'s goal is to answer to the question: "How common is internet censorship in the world?"

Censorship Detector, on the other hand, doesn't have the goal of performing such a large-scale analysis. Instead, this tool gives to the user the possibility of inspecting a specific DNS resolver, in order to verify how trusted it is in terms of possible censorship implementation via DNS manipulation. The goal of my tool is answering to the question: "Does this specific DNS resolver in use implement censorship?"

When analysing a certificate, *Censorship Detector* doesn't simply check for the presence of the certification chain's root inside a trusted database, but it validates the whole chain itself, also by looking for eventual expirations or the presence of self-issued certificates. The detail way in which this is done can be found in the next section.

In addition to the chain validation, *Censorship Detector* uses a name mismatch analysis similar to *CERTainty*'s one: it looks for the presence of the domain's name both in the common name and in the subject alternative names fields. It is not known how *CERTainty* performs the name mismatch analysis in detail, but it probably works in a similar way, since looking for eventual mismatched hostnames in the SAN section sounds like the only way to actively finding them.

It is important to remind that *CERTainty* is not an open-source tool, so it was impossible for me to view its code. The realization of my tool was inspired exclusively by the public information regarding *CERTainty*, meaning the general workflow it implements and the analysis methodology it uses.

Censorship Detector is a tool whose goal is to provide a user with information on how trustworthy the resolver in use actually is. It is built to be used by command line, but its execution is very simple and straightforward.

4.4.2 Technical description

Censorship Detector is based on a "test vs. control" philosophy, meaning that the first operation it performs is comparing the resolutions generated by a DNS resolver under examination to a trusted set of resolutions generated by means of a legit resolver. For this reason, the execution of this tool is split into two phases:

- Creating a trusted resolutions list, which will be used as a control set;
- Performing the analysis of the suspicious resolver, using the previously generated control set as a reference.

The first phase (figure 4.9, `control_resolutions.py`) is performed by setting the machine's DNS resolvers to some trusted ones, like the ones provided by Google (8.8.8.8, 8.8.4.4) or Cloudflare (1.1.1.1, 1.0.0.1), and then running `control_resolutions.py`. Resolvers like these are the best choices for generating a list of trusted resolutions, which will be then output in the `output.csv` file.

Such resolutions are performed for a list of domains contained in the `input.csv` file, which should contain a list of the most visited websites in the world, possibly merged with a list of some of the most usually censored websites online (e.g., the Citizen Lab Test List [36]). This should guarantee a good coverage and a better accuracy for the subsequent part of the analysis.

After the control set is generated, it is time to change the DNS resolvers used by the machine into the ones to be analysed. To obtain the best accuracy, it is important to use one suspicious DNS resolver at the time. At this point, the analysis is performed by running `censorship_detector.py`, which will take the `input.csv` list as input and perform various operations.

The first part of the tool (figure 4.9, `test_resolutions()`) generates a list of untrusted resolutions and saves it into the `output_test.csv` file. If the DNS resolver in use performs response manipulation, this list will be different from the one generated using the legit resolvers. In order to identify which responses could correspond to censored domains, a comparison between `output_control.csv` and `output_test.csv` is performed by the tool (figure 4.9, `comparison()`), which will then output the list of resolutions that returned different IP addresses in the `mismatched_resolutions.csv` file.

In both cases, the resolutions are performed by the tool by using the `nslookup` [41, 42] command line tool, which allows to perform a resolution using the machine's DNS resolvers. The command is repeated for each domain in the `input.csv` list and it is executed in this form:

```
nslookup <domain_name>
```

The second part of the tool's workflow (figure 4.9, `certificate_check()`) consist in checking the certificates provided by the domains that presented mismatched resolutions, which are saved in the `mismatched_resolutions.csv` file. This phase is heavily inspired by the check performed by *CERTainty*, since a misconfigured certificate could indicate that the domain owning it could be blocked.

In particular, what the tool does in this phase consists in performing an inspection of the certificates returned by each domain by means of the `openssl` [43] suite used along with the `s_client` [44] command, which allows to implement a generic SSL/TLS client which connects to a remote host. In particular, this operation is performed by executing the following command for each domain presenting a mismatched resolution for the two DNS resolvers used:

```
openssl s_client -showcerts -connect <domain_name>:443 \
 -CAfile ./mozilla_trusted_ca_list.crt
```

The `-connect` option allows to specify the host and optional port to connect to, the latter being number 443, which is used for the HTTPS protocol. The `-showcerts` option is the one that allows us to perform all the necessary checks on the certificate returned by the contacted domain. This option allows to obtain the certificates list as sent by the server, including the EE certificate, along with all the certificates belonging to the certification chain. The `-CAfile` option allows to specify a file which contains the list of trusted roots that are considered as valid during the verification phase.

At this point, the tool performs an analysis on the information retrieved by the received certificates. The first one is checking if the certification chain is valid or not, by automatically verifying certificates expiration, the presence of any self-signed certificate, the correctness of the issuing order, and more importantly the presence of the chain's root in the file containing the trusted CAs, which in this case is the `mozilla_trusted_ca_list.crt` file. Such file contains the list of root CAs that should be considered as trusted, and the version used by this tool is provided by Mozilla [45].

The second check performed on each certificate consists in comparing the original domain name to the certificate's common name and subject alternative names. In particular, this check consists in verifying whether the domain name is included in the certificate's CN and SAN sections. This verification is crucial, since (as will be seen further in this work) malicious resolvers usually implement censorship by providing a blockpage containing a valid certificate with a mismatched name compared to the real domain name.

Such comparison is performed by using another `openssl` function, allowing to visualize a certificate obtained with the previous command in a readable and complete format, which is then used to check also the SAN list and verify the presence of the domain name in it. The command in question is:

```
echo "<certificate>" | openssl x509 -noout -text
```

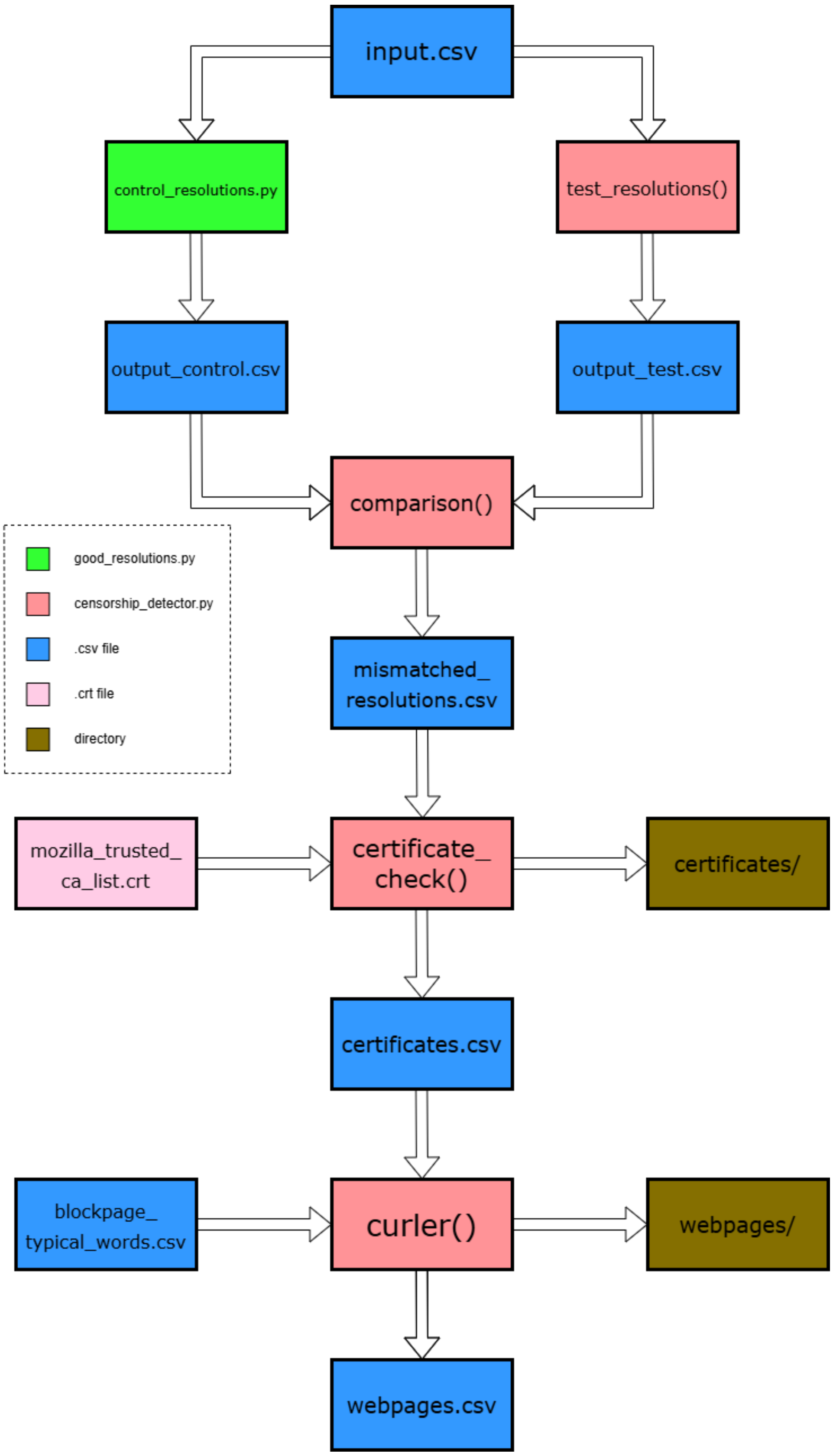


Figure 4.9. *Censorship Detector*'s workflow diagram.

.

In particular, the `-noout` and `-text` commands allow to visualize every needed information on the certificate except the base64 encoded parts.

All the sensitive information gathered in this phase is output in the `certificates.csv` file, which contains the list of all the domains that returned a certificate during the `openssl` connection, along with information about their certification chain (valid or not) and the subject name on their certificate (mismatched or not). Moreover, this file also includes the list of all the subject names in the retrieved certificates, with the purpose of visually verifying which entity's certificates are used in blockpages.

The `/certificates` certificates stores all the certificates obtained during this operation, which can be used to visually verify the obtained data. In detail, each file in this folder contains two versions of the same certificate: the original one, obtained by the `openssl s_client` operation, and the readable one, rendered by using the `openssl x509` command.

The last part of the tool (figure 4.9, `curler()`) performs an HTTPS connection with each of the domains listed in the `certificates.csv` file. This is performed using the `curl` command, in the same exact configuration described in section 4.3.3. As said before, this configuration allows to obtain an informative output, execute the connection in insecure mode and follow every redirection offered by the server.

In this phase, the information considered important are the HTTP status code returned during the connection and the HTML code of the page itself, which are fundamental in order to understand if the page in question is a blockpage. As will be seen later, a lot of blockpages usually return a 4XX status code, which corresponds to a hypothetical bad request by the user, making it a valid censorship indicator for the analysis in question.

Finally, the tool performs an analysis on the webpage itself, after which it generates a score indicating how likely it is for the page in question to be a blockpage: in particular, higher scores indicate a higher chance for a page of representing a blockpage. In order to calculate the score, the tool uses as input a file called `blockpage_typical_words.csv`, which consists in a list of words commonly used in blockpages, paired with a score which is higher for words used more frequently in such pages. This list was generated by performing a word count on a public list of blockpage fingerprints [46] which contains data from OONI, Censored Planet and Citizen Lab.

By using the HTML of the page and the typical words list, the tool calculates the blockpage score for each retrieved page, considering every n word in the list. Such score is the output of this formula

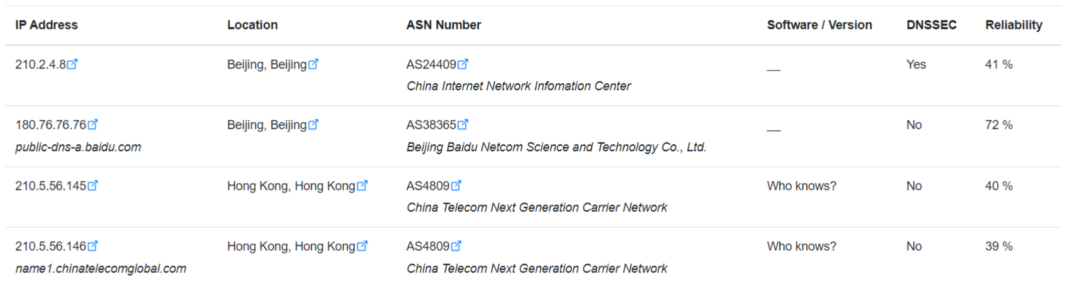
$$S = \frac{\sum_{i=1}^{n} ws_i^2 * wp_i}{length^2},$$

where ws_i is the score of a word in the list, wp_i is a boolean value indicating the presence of the word in the page (1 if present, 0 if absent), and length is the length of the page itself. This formula is based on the empirical experience of blockpages having a short HTML code, along with of course containing some recurring words.

All the information retrieved in this phase is returned in the `webpages.csv` file, which contains the list of domains that returned a valid response to the HTTPS connection, along with their HTTP status code and the generated blockpage score. Moreover, the HTML code of all the retrieved pages is saved in the `/webpages` directory, in order to make it possible to visualize every page and verify what type of blockpages are typically returned using the resolver under analysis.

4.4.3 Limitations of Censorship Detector

Censorship Detector is a particularly slow-running tool, and the long time it takes while performing its various analyses is not due to the tool itself, but to the time some resolvers take to perform each of the `nslookup`, `openssl` and `curl` calls (figure 4.10). There is really no easy solution to this, since any possible improvement in the tool's code cannot speed up the resolvers' responses. One of the possible ideas could be implementing code parallelization, but this methodology is not particularly useful since each section of the tool uses as input the output of the previous one.



IP Address	Location	ASN Number	Software / Version	DNSSEC	Reliability
210.2.4.8	Beijing, Beijing	AS24409 *China Internet Network Infomation Center*	—	Yes	41 %
180.76.76.76 *public-dns-a.baidu.com*	Beijing, Beijing	AS38365 *Beijing Baidu Netcom Science and Technology Co., Ltd.*	—	No	72 %
210.5.56.145	Hong Kong, Hong Kong	AS4809 *China Telecom Next Generation Carrier Network*	Who knows?	No	40 %
210.5.56.146 *name1.chinatelecomglobal.com*	Hong Kong, Hong Kong	AS4809 *China Telecom Next Generation Carrier Network*	Who knows?	No	39 %

Figure 4.10. List of Chinese DNS resolvers, along with their reliability indexes (source: [40]).

.

Another issue in the tool is the presence of some badly formatted output which ends up being added in the various output CSV files. This problem is due to the fact that most of these values are scraped by the responses provided by the previously named command line tools, which most of the time provide standard formatted responses, but not always. The tool handles some of the cases in which the output could present some of these problems, mainly by implementing try-except heuristics. However, it is particularly difficult to handle all the cases of weirdly formatted output, especially for badly configured certificates and for particular HTTP responses, hence some of these wrong values are present in the files' final versions. This happens also when collecting the final blockpages' HTMLs, which sometimes are not saved in their entirety.

Finally, it should be expected that for each step executed by the tool a good chunk of the domains under analysis is lost due to timeout reasons, which is once again caused by the badly configured nature of the resolvers in question. The tool's code also incorporates some timeouts, which have the purpose of speeding the overall execution.

More information about the tools code can be found in the Developer's Manual in the Appendix B section.

Chapter 5

Data Analysis performed with Censorship Detector

This chapter presents and discusses the results obtained from two executions of *Censorship Detector* while testing two DNS resolvers located in China. These runs of the tool were performed on a virtual machine running Kali Linux 2024.2 as operating system. The list of trusted root CAs present in the `mozilla_trusted_ca_list.crt` file was downloaded on November 1st 2024 and was not manually modified by adding any new custom trusted root, nor deleting any of the certificates in the list.

The first resolver under analysis is owned by the China Unicom Beijing Province Network and its IP address is 123.123.123.123; it has a reliability score of 68% and doesn't support DNSSEC. The second resolver under analysis is owned by Beijing Baidu Netcom Science and Technology Co., Ltd. and its IP address is 180.76.76.76; it has a reliability score of 72% and also doesn't support DNSSEC.

Since they belong to the national Chinese network, these resolvers are suspected of implementing censorship via DNS manipulation. Moreover, choosing such resolvers makes it possible to compare this study's results with the ones obtained by *CERTainty* when analysing resolvers from the same country.

The `input.csv` domain list in use was composed by entries taken from two public lists:

- The top 500 most visited domains from the Tranco top 1 Million list [34, 35];
- The full Citizen Lab Global Test list [36], which at the time of execution counted 1655 entries.

Such list contained a total of 2155 domains, and it was built in the same way as the one used for the execution of *CERTainty*. It is important to remind that these lists are constantly updated, so the one used in my execution, even if built in the same way as the one used for *CERTainty*, presents some differences regarding its size and its elements.

The two lists in use for creating the final one provide a good coverage for censorship inspections, since the first one contains a list of very common and highly visited websites, while the second one contains domains that are typically blocked by various resolvers according to previous studies and experiences. This allows to perform an accurate analysis for a suspicious resolver, hence – even if not mandatory – using a list generated with these criteria is strongly suggested.

The DNS resolvers used to generate the list of trusted resolutions were Google's primary and secondary resolvers (figure 5.1), whose IP address are 8.8.8.8 and 8.8.4.4, respectively; they both have a reliability score of 100% and support DNSSEC.

As in every study performed using a "test vs. control" methodology, it is necessary to consider the control data set as trusted, even without performing any verification on it: choosing Google's

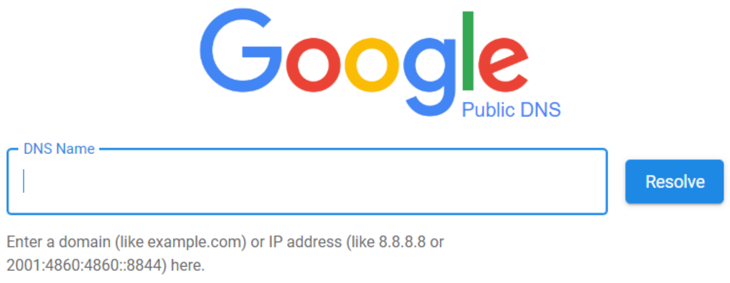

Google
Public DNS
DNS Name
Resolve
Enter a domain (like example.com) or IP address (like 8.8.8.8 or 2001:4860:4860::8844) here.

Figure 5.1. Google's DNS resolver service, located at `dns.google` or at `8.8.8.8`

.

resolvers as a trusted one does not mean that it has been tested by me, and there is no claim in this work that such resolver is censor-free. Google's resolvers are some of the most used in the whole world and they were considered trustworthy in *CERTainty*'s study too, hence I decided that they were a valid choice for representing the control section.

5.1 123.123.123.123

5.1.1 DNS resolutions

After generating the trusted resolutions list, the output obtained (`output_control.csv`) consisted of 2155 entries, 1288 of them containing correctly formatted IP addresses (59.8%). In this operation, 325 resolutions failed (marked with "N/A", 15.1%), while 542 resolutions generated badly formatted output (25.2%), not corresponding to an IP address. Such results are depicted in figure 5.2.

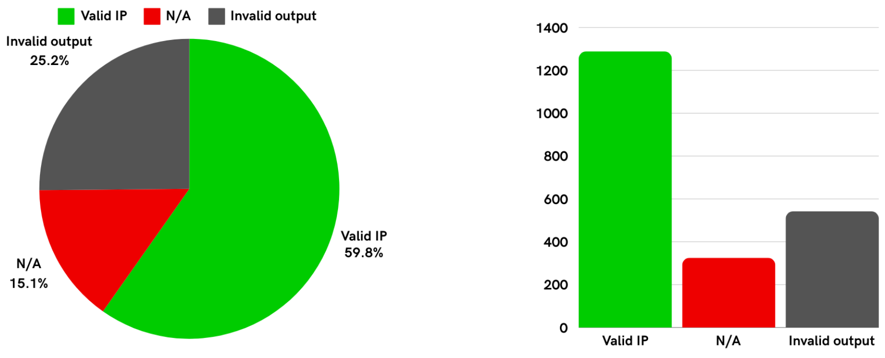


Figure 5.2. Results of the trusted resolutions generation phase.

.

It is important to generate such list before performing any analysis for any suspicious resolver: most websites frequently change their IP addresses, so using an old version of such trusted resolutions could cause inaccuracy while comparing them with the untrusted ones. In particular, using a not updated list could cause more mismatched resolutions to be found, but most of them would consist of false positives.

When executing the main part of the tool, the output of the untrusted resolutions generation phase (`output_test.csv`) contained 2155 elements, 1375 of them consisting of correctly formatted IP addresses (63.8%). While performing such resolutions, 332 of them failed (marked with "N/A", 15.4%), while 448 of them consisted of a badly formatted output (20.8%), not corresponding to an IP address. Such results are depicted in figure 5.3.

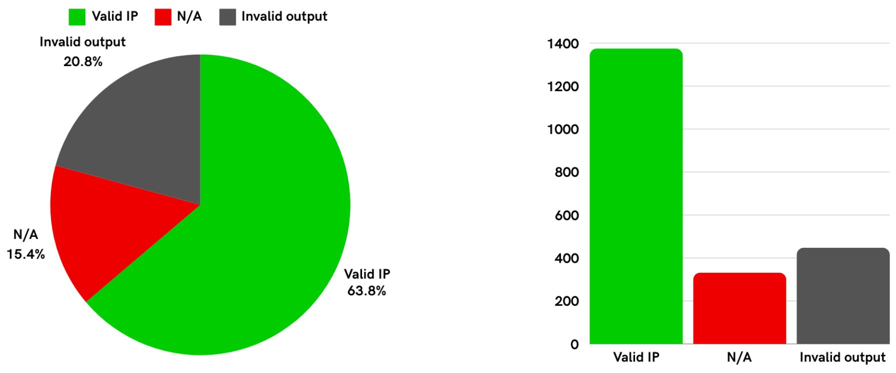


Figure 5.3. Results of the untrusted resolutions generation phase.

.

The two generated resolutions sets were compared, in order to find how often the two resolvers provided different IP addresses for the same domain. The total set of mismatched resolutions (`mismatched_resolutions.csv`) contained 812 entries.

The presence of so many domains in this list is a strong signal that the untrusted resolver under analysis is not returning the same IP addresses as the ones returned by the trusted one. In particular, the size of this set indicates that 38.9% of the domains are not resolved with the same IP by the two resolvers. Such set is used as input to perform the following certificate analysis.

5.1.2 Certificates check

For every domain in the previously generated list, a certificate-retrieving operation was performed, in order to obtain various information about the certificates of the domains that presented mismatched resolutions. The result of this operation was output in the `certificates.csv` file.

The domains that provided a suspicious certificate were 91: 2 of them had an invalid certification chain and a mismatched hostname (2.2%), 73 had a valid certification chain and a mismatched hostname (80.2%), while 16 returned no certificate at all (17.6%). None of the certificates had an invalid certification chain with a matched hostname. Such results are depicted in figure 5.4.

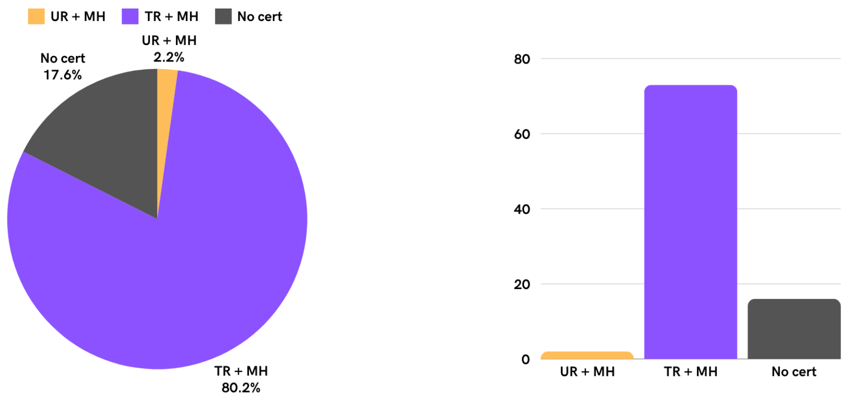


Figure 5.4. Results of the certificates check phase.

.

CERTainty's study focus particularly on Chinese resolvers, showing how they typically censor domains by providing blockpages along with certificates issued to large US-based entities. This was also verified in this execution of my tool, since all suspicious certificates showed a mismatched hostname, meaning that for all the cases where a suspicious certificate was retrieved, the name of the domain was not found in the certificate's SAN list or in its common name field.

This execution confirmed the results already obtained with *CERTainty* (and also by other previous studies on China's DNS architecture): in this list it is possible to see that 42 domains in the whole set returned a certificate whose subject was `*.facebook.com` or some of its subdomains. It is needless to say that none of the domains that owned such certificates was actually Facebook. This seems to be the most common way to implement censorship in China: when trying to reach a hidden website, the webpage where the user is redirected by the resolver is shown as owning a valid certificate, which however is not issued to the domain in question, but typically belongs to large companies that have nothing to do with the requested domain. The methodology that *Censorship Detector* identified as the most used by Chinese resolvers is the same one discovered by *CERTainty* for resolvers from the same country.

This is the list of the 40 domains that provided a certificate issued to `*.facebook.com`:

- `wikipedia.org`
- `vimeo.com`
- `reddit.com`
- `washingtonpost.com`
- `asiatimes.com`
- `secure.avaaz.org`
- `btguard.com`
- `duckduckgo.com`
- `riseup.net`
- `rsf.org`
- `www.change.org`
- `www.csmonitor.com`
- `www.heritage.org`
- `www.ifex.org`
- `www.lemonde.fr`
- `www.megaproxy.com`
- `www.pokerstars.com`
- `www.reddit.com`
- `www.sendspace.com`
- `www.twitch.tv`
- `unwatch.org`
- `www.bbc.com`
- `ru.wikipedia.org`
- `fr.wikipedia.org`

- `line.me`
- `pt.m.wikipedia.org`
- `www.omct.org`
- `etherscan.io`
- `joinmastodon.org`
- `wire.com`
- `telegram.org`
- `securevpn.com`
- `git.io`
- `mega.io`
- `www.gofundme.com`
- `www.bitchute.com`
- `openverse.org`
- `rutube.ru`
- `linktr.ee`
- `cdns.grindr.com`

Other domains were hidden in a similar way, always using certificates belonging to Facebook's subdomains. For example, `pinterest.com` provided a certificate issued to `*.secure.latest.facebook.com`, while `bouncer.ooni.io` provided a certificate issued to `*.extern.facebook.com`.

This is the list of the 3 domains that returned a certificate issued to `*.ar.meta.com`:

- `tumblr.com`
- `briarproject.org`
- `libgen.st`

These are the 2 domains that returned a certificate issued to `*.internet.org`:

- `youtube.com`
- `www.hootsuite.com`

These are the 2 domains that returned a certificate issued to `api.twitter.com`:

- `dailymotion.com`
- `www.4shared.com`

The 2 domains that provided an invalid certificate with mismatched hostname were:

- `laborrights.org`
- `discordapp.com`

Among these last two domains, the first one provided an expired certificate issued to a subdomain of `cloudflaressl.com`, while the second one was issued to `sni.dreamhost.com`, and contained a self-signed certificate in its certification chain.

These lists span from social networks to news services, including also websites of organizations advocating human rights and free speech. It is easy to imagine the interest that China's government would have in hiding these contents.

This experience confirms the fact that X.509 certificates can be useful in detecting when and how DNS manipulation happens. This methodology is at the moment one of the most precise ones, even one year after the developing of *CERTainty* in 2023. Moreover, almost all browsers are able to detect when this kind of manipulation happens, making it basically impossible to access a website if the certificate it owns presents a name mismatch.

This operation also outputs as suspicious 16 domains that returned no certificate at all when performing an `openssl` connection. The domains are classified as possible victim of manipulation, since the lack of certificates is a strong indicator that the webpage could not be the original one. However, these cases must be interpreted carefully, since they could also consist of bad connection events. For this type of domains, the webpage analysis phase is very useful to have a deeper understanding about their actual behaviour.

The domains in question are:

- `tiktok.com`
- `nytimes.com`
- `criteo.net`
- `google.de`
- `freedomhouse.org`
- `www.aljazeera.net`
- `www.amnesty.org`
- `www.cbc.ca`
- `www.democracynow.org`
- `www.ebaumsworld.com`
- `www.grindr.com`
- `www.torproject.org`
- `www.goodreads.com`
- `nordvpn.com`
- `matcha.xyz`
- `zenvpn.net`

The list of the certificates retrieved in this phase was output in the `/webpages` directory, while information about them were put in the `webpages.csv` file, which was used as input for the webpages' retrieval phase.

5.1.3 Blockpages retrieval

Finally, the previous list was used as input for retrieving the webpages of censored domains, which are expected to be blockpages. The total number of webpages retrieved in this phase is 18, while all the other connection attempts provided whether a timeout or no response at all.

Among all the successful connections, these were the results in terms of returned status codes:

- 10 connections returned a 400 Bad Request status code (55.6%);
- 4 connections returned a 200 OK status code (22.2%);
- 3 connections returned a 407 Proxy Authentication Required status code (16.7%);
- 1 connection returned a 404 Not Found status code (5.6%);

Such results are shown in figure 5.5

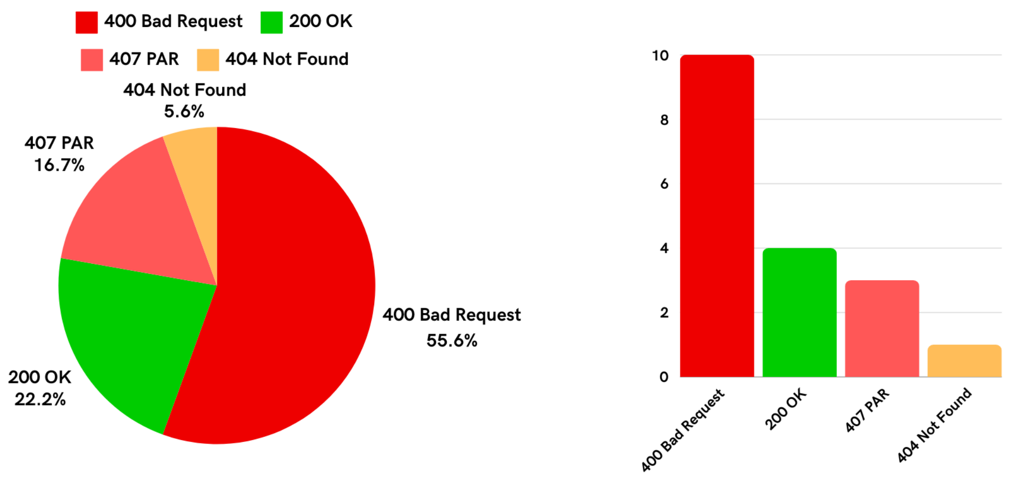


Figure 5.5. Results of the blockpages retrieval phase.

.

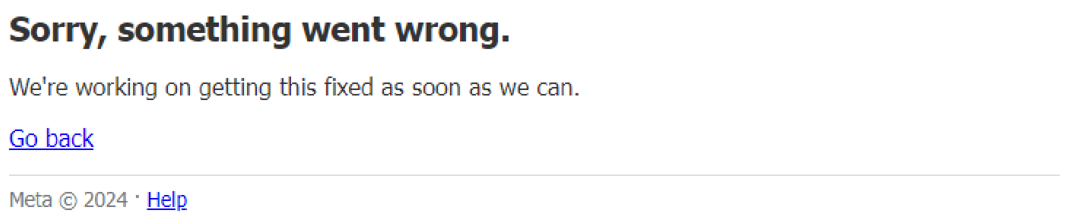
At this point, it is expected to find blockpage in the HTMLs of responses that returned a 4XX status code. While the 404 responses might consist of legit informative pages about resources not found, the 400 and 407 responses are the ones that are most common in being blockpages. Moreover, in the case that this phase would have output some 5XX responses, they would have been considered to be legit server errors.

This is the list of the 10 domains that returned a 400 status code:

- `pinterest.com`
- `freedomhouse.org`
- `www.ifex.org`
- `fr.wikipedia.org`
- `pt.m.wikipedia.org`
- `securevpn.com`
- `wire.com`
- `telegra.ph`

- `advox.globalvoices.org`
- `ru.wikipedia.org`

6 out of 10 of these domains were hidden by the same blockpage, which consisted in a fake Meta/Facebook error page, depicted in figure 5.6, while `wire.com` returned the same blockpage with less details, depicted in figure 5.7.


Sorry, something went wrong.

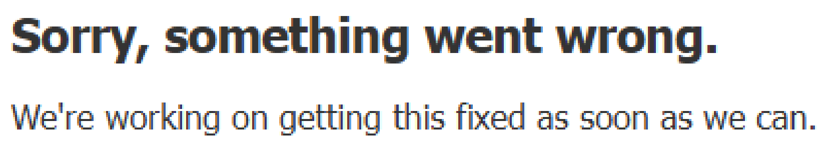
We're working on getting this fixed as soon as we can.

Go back

Meta © 2024 · Help

Figure 5.6. Fake Meta/Facebook blockpage.

.


Sorry, something went wrong.

We're working on getting this fixed as soon as we can.

Figure 5.7. Fake Meta/Facebook blockpage, version 2.

.

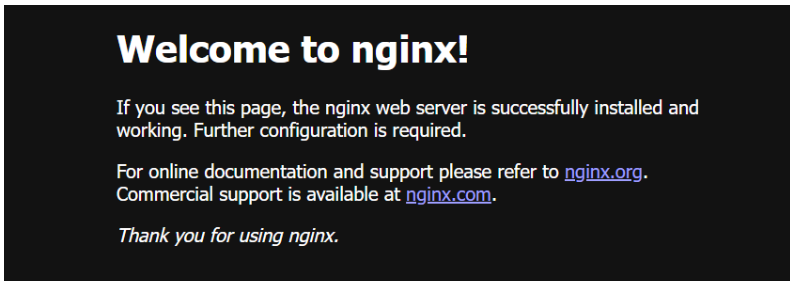
This is the list of the 4 domains that returned a 200 status code:

- `riseup.net`
- `briarproject.org`
- `www.scruff.com`
- `washingtonpost.com`

Such domains were hidden using different blockpages. For example, `riseup.net` was hidden with fake nginx configuration page (figure 5.8), `briarproject.org` was hidden with a fake DreamHost page (figure 5.9), and `www.scruff.com` was replaced by an empty page whose HTML source code says "You need to enable JavaScript to run this app".

This is the list of the 3 domains that returned a 407 status code:

- `vimeo.com`
- `google.de`
- `btguard.com`



Welcome to nginx!

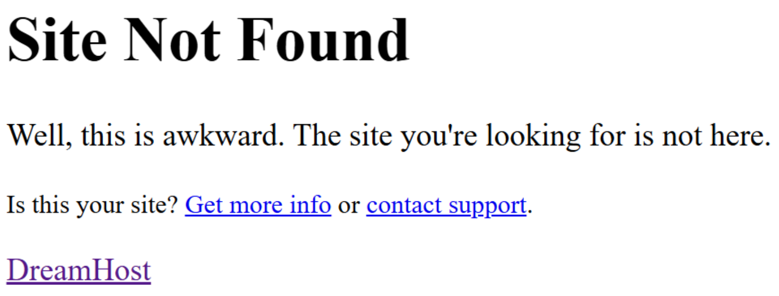
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Figure 5.8. Fake nginx configuration page.

.



Site Not Found

Well, this is awkward. The site you're looking for is not here.

Is this your site? Get more info or contact support.

DreamHost

Figure 5.9. Fake DreamHost "not found" page.

.

All three of these domains were hidden by another version of the fake Meta/Facebook blockpage, depicted in figure 5.10

Finally, `wikimapia.org` was the only domain returning a 404 status code, but however its HTML was empty.

The list of the domains that provided such pages was output in the `webpages.csv` file, along with their HTTP status codes and their computed blockpage score, while the HTMLs of the pages were saved in the `/webpages` directory.

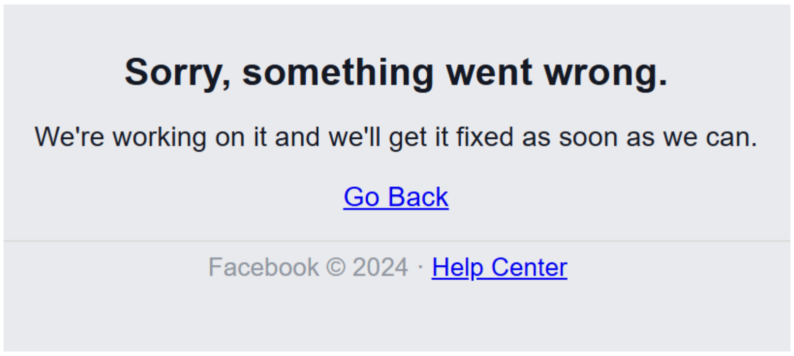
5.1.4 Results analysis

By analysing the data obtained by this run of the tool, it is possible to confirm most of the results reported by previous similar studies, including the way certificates, HTTP status codes, and blockpages are fabricated and modified when a DNS response is manipulated at resolver level.

The main indicator that a victim has been redirected to a different webpage by means of a DNS manipulation by a resolver implementing censorship lies in the certificate provided by the final webpage and in how it is configured. Providing a valid certificate with a mismatched hostname (meaning both common name and SAN) seems to be the most used technique. In such cases:

- the certificates' subjects typically consist in large US-based companies;
- the certificates are typically not expired;
- the certificates might have an invalid certification chain, but this happens very rarely.

In addition to this, it is possible to use other information as indicators of how likely it is for a domain to be hidden, but this is a secondary check which needs to perform an HTTPS connection with the domains in question, hence being very prone to fail and give no results at all.


Sorry, something went wrong.

We're working on it and we'll get it fixed as soon as we can.

Go Back

Facebook © 2024 · Help Center

Figure 5.10. Fake Meta/Facebook blockpage, version 3.

.

When a result is obtained, it is very common for a blockpage to return a 4XX status code, with 400 Bad Request and 407 Proxy Authentication Required being the most common ones, even if also some 200 OK responses are present. The blockpages themselves can be less or more informative regarding what to show to the victim: sometimes they are explicit about the visited content being unavailable (usually justifying it with false explanations), while other times they can simply consists of a blank page.

5.2 180.76.76.76

5.2.1 DNS resolutions

After generating the trusted resolutions list, the output obtained (`output_control.csv`) consisted of 2155 entries, 1308 of them containing correctly formatted IP addresses (60.7%). In this operation, 323 resolutions failed (marked with “N/A”, 15.0%), while 524 resolutions generated badly formatted output (24.3%), not corresponding to an IP address. Such results are depicted in figure 5.11.

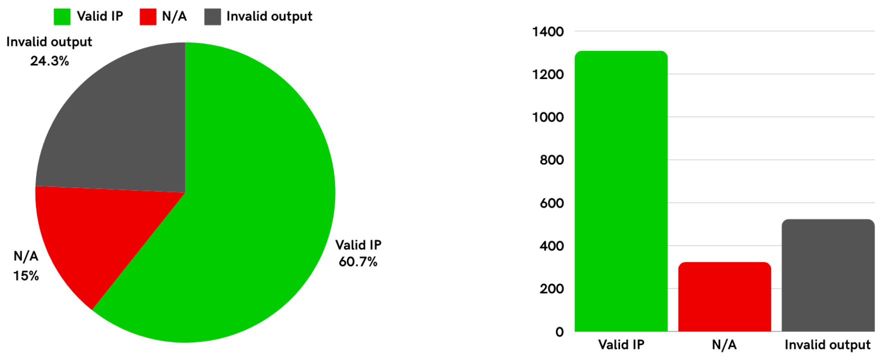


Figure 5.11. Results of the trusted resolutions generation phase.

.

As mentioned before, it is important to generate such list before performing any execution of the tool. This is the reason why the number of the different types of results present in the `output_control.csv` file for this run is not the same compared to the last execution, even if the control DNS resolvers are always the ones provided by Google.

When executing the main part of the tool, the output of the untrusted resolutions generation phase (`output_test.csv`) contained 2155 elements, 1797 of them consisting of correctly formatted

IP addresses (83.4%). While performing such resolutions, 340 of them failed (marked with “N/A”, 15.8%), while only 18 of them consisted of a badly formatted output (0.8%), not corresponding to an IP address. Such results are depicted in figure 5.12.

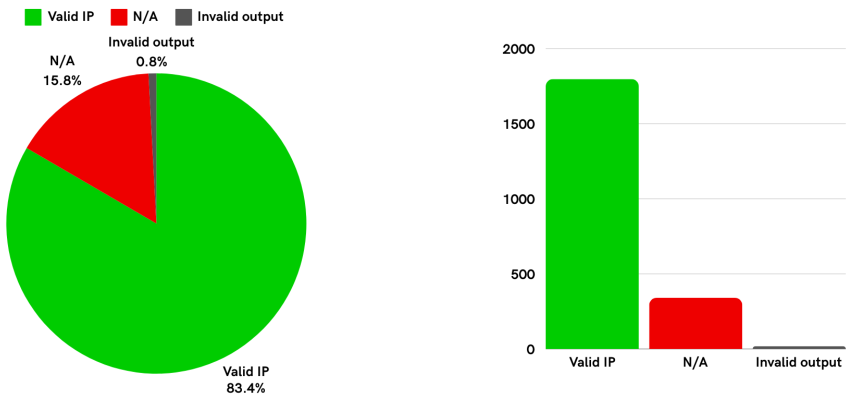


Figure 5.12. Results of the untrusted resolutions generation phase.

.

At this point, the comparison operation was performed, with the purpose of understanding how often the two resolvers provided non-matching IP addresses for the same domain. The total set of mismatched resolutions (`mismatched_resolutions.csv`) contained 1203 entries.

The size of this set is even bigger that the one obtained in the previous execution, indicating that it is very likely that such test resolver is returning manipulated DNS records. In particular, the size of this set indicates that 55.8% of the domains are not resolved with the same IP by the two resolvers. Such set is used as input to perform the following certificate analysis.

5.2.2 Certificates check

For every domain in the previously generated list, a certificate-retrieving operation was performed, in order to obtain various information about the certificates of the domains that presented mismatched resolutions. The result of this operation was output in the `certificates.csv` file.

The domains that provided a suspicious certificate were 92: 7 of them had an invalid certification chain and a mismatched hostname (7.4%), 68 had a valid certification chain and a mismatched hostname (71.6%), 4 of them had an invalid certification chain but a matching hostname (4.2%), and 13 of these domains returned no certificate at all (16.8%). For this run of the tool, some of the certificates showed a matching hostname with an invalid certificate, differently from the last execution. Such results are depicted in figure 5.13.

Once again, it is possible to notice how certificates with mismatched names are the most common ones. In detail, in this list it is possible to see that 37 domains in the whole set returned a valid certificate whose subject was `*.facebook.com` or some of its subdomains. Once again, Facebook has nothing to do with the domains returning such certificates. This run of the *Censorship Detector* also confirms the data obtained by *CERTainty*.

This is the list of the 34 domains that provided a certificate issued to `*.facebook.com`:

- `youtube.com`
- `pinterest.com`
- `bbc.co.uk`
- `godaddy.com`

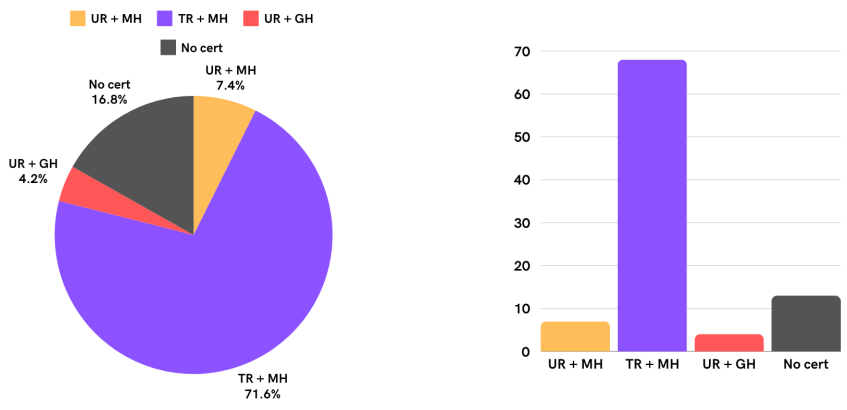


Figure 5.13. Results of the certificates check phase.

.

- `google.com.hk`
- `bouncer.ooni.io`
- `explorer.ooni.org`
- `www.lushstories.com`
- `www.wsj.com`
- `www.ask.com`
- `www.cbc.ca`
- `www.csmonitor.com`
- `www.fidh.org`
- `www.ifj.org`
- `www.indiatimes.com`
- `www.reddit.com`
- `www.rfa.org`
- `www.sendspace.com`
- `unwatch.org`
- `www.xnxx.com`
- `en.wikipedia.org`
- `de.wikipedia.org`
- `ru.wikipedia.org`
- `fr.wikipedia.org`
- `ca.wikipedia.org`
- `zh.wikipedia.org`
- `www.betternet.co`
- `www.omct.org`

- `www.dailymail.co.uk`
- `telegra.ph`
- `mega.io`
- `yts.mx`

Other domains were hidden in a similar way, always using certificates belonging to Facebook's subdomains. For example, `disqus.com`, `www.gofundme.com`, and `katcr.to` provided certificates issued to `*.extern.facebook.com`.

This is the list of the 3 domains that returned a certificate issued to `*.instagram.com`:

- `vimeo.com`
- `www.tumblr.com`
- `www.bitchute.com`

These are the 4 domains that returned a certificate issued to `*.internet.org`:

- `freenetproject.org`
- `simple.wikipedia.org`
- `protonvpn.com`
- `briarproject.org`

These are the 2 domains that returned a certificate issued to `*.twitter.com`:

- `guardster.com`
- `www.securityinabox.org`

Moreover, `www.kickassclassical.com` returned a certificate issued to `*.storage.googleapis.com`.

The 7 domains that provided an invalid certificate with mismatched hostname were:

- `telegram.org`
- `www.nifty.org`
- `www.article19.org`
- `www.blogsome.com`
- `www.facebook.com`
- `www.rferl.org`
- `www.trendsmap.com`

Among these domains, `telegram.org`, `www.facebook.com`, `www.rferl.org`, and `www.trendsmap.com` returned expired certificates issued to subdomains of `cloudflaressl.com`, while `www.nifty.org` and `www.article19.org` returned certificates issued to `sni.dreamhost.com` which showed a self-signed certificate in their chain. Finally, `www.blogsome.com` returned a certificate issued to `sh.gritmonkey.com`, which was also expired.

This is the list of the 4 domains that returned invalid certificates with a matching hostname:

- `www.4online-gambling.com`,
- `www.oicc.org`
- `www.originalelf.com`
- `www.tobacco.org`

All of the certificates provided by these 4 domains were expired.

This operation also outputs as suspicious 13 domains that returned no certificate at all when performing an `openssl` connection. The domains are classified as possible victim of manipulation, since the lack of certificates is a strong indicator that the webpage could not be the original one. However, as it was for the previous execution, these cases must be interpreted carefully, since they could also consist of bad connection events. For this type of domains, the webpage analysis phase is very useful to have a deeper understanding about their actual behaviour.

The domains in question are:

- `criteo.net`
- `wsj.com`
- `www.hootsuite.com`
- `imgur.com`
- `www.acquisitionx.com`
- `www.coinbase.com`
- `www.formercatholic.com`
- `hustler.com`
- `www.kcna`
- `mastodon.social`
- `mask.icloud.com`
- `mask-h2.icloud.com`
- `kat.am`

The list of the certificates retrieved in this phase was output in the `/webpages` directory, while information about them were put in the `webpages.csv` file, which was used as input for the webpages' retrieval phase.

5.2.3 Blockpages retrieval

Finally, the previous list was used as input for retrieving the webpages of censored domains, which are expected to be blockpages. The total number of webpages retrieved in this phase is 23, while all the other connection attempts provided whether a timeout or no response at all.

Among all the successful connections, these were the results in terms of returned status codes:

- 11 connections returned a 400 Bad Request status code (47.8%);
- 8 connections returned a 200 OK status code (34.8%);
- 2 connections returned a 404 Not Found status code (8.7%);
- 1 connection returned a 503 Service Unavailable status code (4.3%);

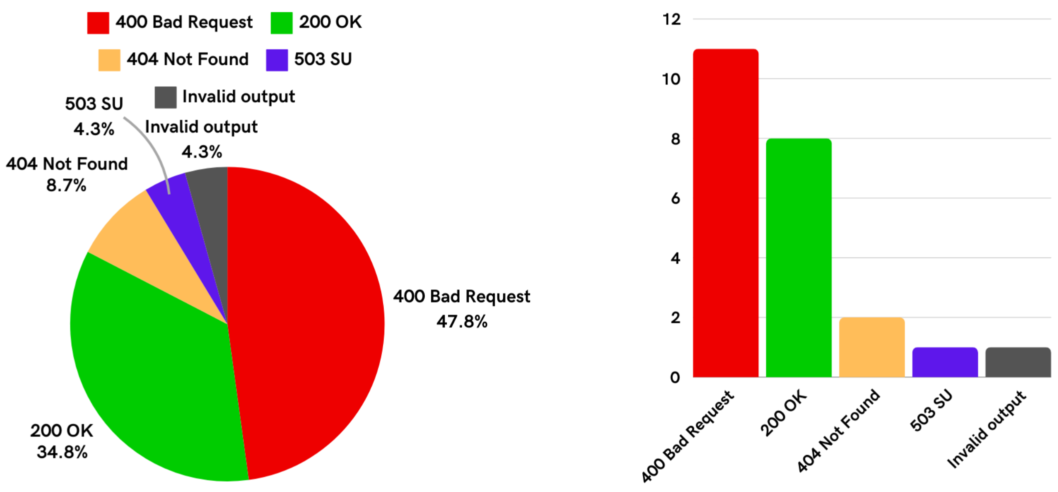


Figure 5.14. Results of the blockpages retrieval phase.

.

- 1 connection returned an invalid output, not corresponding to any status code (4.3%);

Such results are shown in figure 5.14

At this point, it is expected to find blockpage in the HTMLs of responses that returned a 4XX status code. While the 404 responses might consist of legit informative pages about resources not found, the 400 responses are most likely to be blockpages. It is importanto to remind that also connections that returned a 200 status code could consist of blockpages. 5XX responses are interpreted to be legit server errors.

This is the list of the 11 domains that returned a 400 status code:

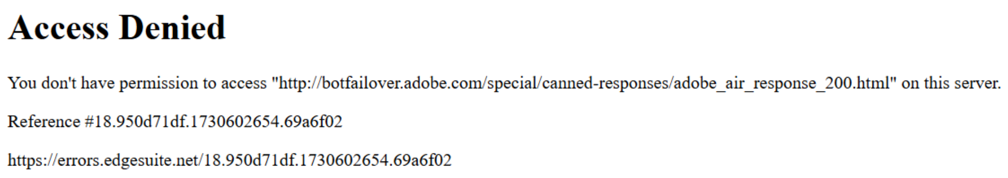
- `simple.wikipedia.org`
- `youtube.com`
- `vimeo.com`
- `google.com.hk`
- `www.coinbase.com`
- `www.indiatimes.com`
- `ru.wikipedia.org`
- `fr.wikipedia.org`
- `ca.wikipedia.org`
- `briarproject.org`
- `explorer.ooni.org`

9 out of 11 of these domains were hidden by the same fake Meta/Facebook error page depicted in figure 5.6, while `explorer.ooni.org` was hidden with the webpage depicted in figure 5.10. A very interesting case is `simple.wikipedia.org`, which retuned a very simple no-CSS blockpage citing only "5xx Server Error", even if the actual returned status code was not a 5xx code but a 400 code, showing how such blockpage is completely fabricated and no internal server error is actually occurring.

This is the list of the 8 domains that returned a 200 status code:

- `adobe.net`
- `www.mail.lycos.com`
- `coronavirus.app`
- `www.originalelf.com`
- `www.tobacco.org`
- `www.4online-gambling.com`
- `www.om.org`
- `yts.mx`

Among these, a new blockpage discovered was the one used to censor `adobe.net`, consisting of a generic No-CSS error page, depicted in figure 5.15.



Access Denied

You don't have permission to access "http://botfailover.adobe.com/special/canned-responses/adobe_air_response_200.html" on this server.

Reference #18.950d71df.1730602654.69a6f02

https://errors.edgesuite.net/18.950d71df.1730602654.69a6f02

Figure 5.15. No-CSS error page used to hide `adobe.net`.

.

This is the list of the 2 domains that returned a 404 status code:

- `pinterest.com`
- `www.bitchute.com`

These domains returned no HTML at all.

The list of the domains that provided such pages was output in the `webpages.csv` file, along with their HTTP status codes and their computed blockpage score, while the HTMLs of the pages were saved in the `/webpages` directory.

5.2.4 Results analysis

The results obtained by this run of the tool confirm the ones obtained by the previous run, along with the ones obtained by similar studies on the topic, especially from the certificates' manipulation point of view.

The main indicator that a victim has been redirected to a different webpage by means of a DNS manipulation by a resolver implementing censorship keeps being the certificate provided by the final webpage and how it is configured. Providing a valid certificate with a mismatched hostname (meaning both common name and SAN) seems to be the most used technique. However, also certificates with invalid chains are present, both with matching or mismatching hostnames. Such certificates are less rare in this execution compared to the previous one

In addition to this, it is possible to use other information as indicators of how likely it is for a domain to be hidden, but this is a secondary check which needs to perform an HTTPS connection with the domains in question, hence being very prone to fail and give no results at all.

When a result is obtained, it is very common for a blockpage to return a 400 Bad Request status code, but also 200 OK appears in the final results. The blockpages themselves can be less or more informative regarding what to show to the victim: sometimes they are explicit about the visited content being unavailable (usually justifying it with false explanations), while other times they can simply provide a blank page.

Chapter 6

Conclusions

Because of how it was natively conceived in 1983, the DNS protocol comes with multiple vulnerabilities, most of them caused by the lack of response authentication, which makes many attacks possible.

Attacks like DNS cache poisoning were first discovered decades ago. During the years, various solutions have been developed for these kind of attacks, which made them become less common. However, record poisoning attacks are still possible with the right amount of resources, as it is possible to see from some of the articles cited in this work.

The purpose of these attacks is typically to lure a victim into visiting a website that doesn't correspond to the legit one. If the fake website is constructed with the purpose of looking similar to the original one, it is considered a case of domain impersonation, which is typically performed to steal credentials or personal information from the victim.

The manipulation of DNS responses can be performed by malicious actors placed at various layers. Since DNS responses are not authenticated, this allows also malicious DNS resolvers to provide fake resolutions. This kind of manipulation is very typical for resolvers located in countries which employ censorship of online content, and its purpose is to redirect the victim to a blockpage.

Manipulations like the ones just described can be identified more easily by exploiting Public Key Infrastructures, in particular data retrieved from X.509 certificates. When hiding a content, the blockpage that replaces the legit domain has to come with a valid certificate, in order to look as legit as possible. Blockpages' certificates show some recurring patterns, whose presence can be utilized to recognize when this kind of manipulation happens.

This study comes to the conclusion that manipulation-based censorship can be easily noticed by checking if the certificate provided by the webpage reached when resolving a domain contains a mismatched name compared to the original domain's name, or if the root of its certification chain is not trusted. This claim perfectly mirrors what has already been discovered in previous studies about this specific topic, such as the 2023 study performed by Tsai et al. [3], where they propose *CERTainty* as a new detecting method for when this type of manipulation occurs.

The results I obtained in this study were retrieved thanks to a tool I realized, strongly inspired by *CERTainty*, called *Censorship Detector*, whose goal is to identify when DNS manipulation occurs by analysing the retrieved certificates and looking into them for the suspicious patterns previously described.

Along with the certificates analysis, my tool also retrieves data from HTTPS connections performed with the domains in question, with the purpose of obtaining the returned status codes and the webpages' HTMLs. Other recurring patterns in domains subject to manipulation can also be observed in the data just mentioned, since blockpages typically return a 4XX status code.

My tool also includes a feature that calculates a score associated to each retrieved webpage, which is higher for pages that are most likely to be blockpages. This blockpage score calculation method is far from being perfect, since it generates the final score only by analysing the pages'

HTML content, which sometimes may not provide enough information to verify whether the page in question is actually a blockpage or not.

The final objective of this study is to prove that heuristics based on X.509 certificates analysis are perfectly valid for identifying DNS manipulation attacks.

Appendix A

User's Manual

A.1 Virtual environment

Censorship Detector was coded and tested entirely on a virtual machine running Kali Linux 2024.2 as the operating system, hence its correct functioning is guaranteed only for machines using the same OS. However, it is expected to work correctly for every Linux distribution.

Before running the tool, it is recommended to perform a general update of the system, by executing the commands:

```
sudo apt update
```

```
sudo apt upgrade -y
```

This tool is completely coded in Python, which has to be installed in order to execute the tool. If no version of Python is installed, the latest one can be obtained by running the command:

```
sudo apt install python3
```

These operations generate an appropriate environment for running the tool.

A.2 Tool installation

Censorship Detector can be retrieved by its public GitHub repository [47] by running the command:

```
git clone https://github.com/rfiorilla/thesis.git
```

A.3 Tool execution

The tool execution is based in two phases: the construction of the trusted resolutions list, and the execution of the analysis of the data obtained by the untrusted resolver.

In order to execute the tool correctly:

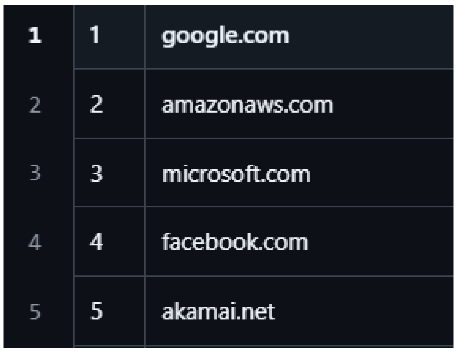
- Set the machine's DNS resolver to a trusted one, by running the command `sudo nano etc/resolv.conf` and editing the file by removing or commenting the resolvers in use and adding the trusted one, e.g., `nameserver 8.8.8.8`;
- Run the command `python control_resolutions.py` in order to generate the trusted resolutions list;

- Set the machine's DNS resolver to the untrusted one to be analysed, by running the command `sudo nano etc/resolv.conf` and editing the file by removing or commenting the resolver previously set and adding the untrusted one, e.g., `nameserver 123.123.123.123`;
- Run the command `python censorship_detector.py` in order to analyse the resolver under consideration.

A.4 Files schema

This is the description of the files used by the tool and their content:

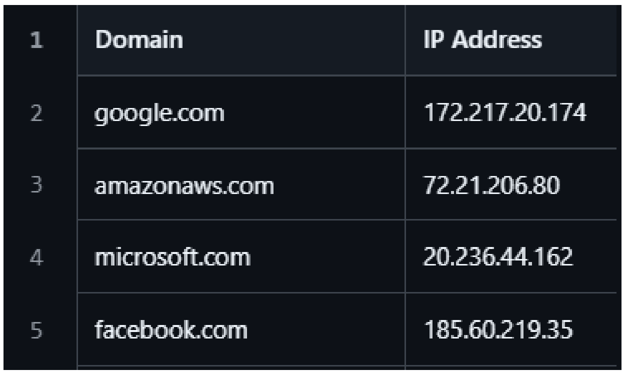
- `input.csv` (figure A.1) is the file containing the list of domains to be analysed by the tool; when the repository is originally cloned, this file contains a list of the most visited domains on the internet, composed by the top 500 domains from the Tranco top 1 Million list [34, 35] and by the full Citizen Lab Global Test list [36]; the columns of this CSV file represent the position of a domain in the list and the domain's name; no header is present.
- `output_control.csv` (figure A.2) contains the list of trusted resolutions generated using the trusted DNS resolver; the columns of this CSV file represent the domain name and the IP address obtained by the resolution; a header is present.
- `output_test.csv` (figure A.3) contains the list of untrusted resolutions generated using the DNS resolver under analysis; the columns of this CSV file represent the domain name and the IP address obtained by the resolution; a header is present.
- `mismatched_resolutions.csv` (figure A.4) contains the list of mismatched resolutions obtained by comparing the trusted and the untrusted ones; the columns of this CSV file represent the domain name, the IP address obtained by the trusted resolution, and the IP address obtained by the untrusted resolution; a header is present.
- `mozilla_trusted_ca_list.crt` (figure A.5) contains a list of trusted root CAs provided my Mozilla, which are used during the validation phase; each certificate is delimited from the other using the standard "`BEGIN/END CERTIFICATE`" technique.
- `certificates.csv` (figure A.6) is generated after analyzing the certificates of the domains with suspicious (mismatched) resolutions, and it contains some information about their certificates; the columns of this CSV file represent the domain name, a boolean value indicating if its certificate has an untrusted root, a boolean value indicating if its certificate has a mismatched name, and the value of the common name in the certificate; a header is present.
- `/certificates` (figure A.7) is a directory filled with the certificates retrieved during the validation phase; for each domain, the certificate is provided in two formats: the original one (which also contains the full chain) and the readable one (which contains all the certificate's fields).
- `blockpage_typical_words.csv` (figure A.8) contains a list of words commonly found in blockpages, paired with a score; this list was generated by performing a word count on a list of blocking fingerprints [46] that includes data from OONI, Censored Planet and Citizen Lab; the columns of this CSV file represent the score of the word and the word itself; a header is present.
- `webpages.csv` (figure A.9) is generated after trying to establish an HTTP connection with the websites that displayed suspicious certificates (invalid or with mismatched names), and it contains some information retrieved after this operation; the columns of this CSV file represent the domain name, the returned HTTP status code, and the calculated blockpage score; a header is present.
- `/webpages` (figure A.10) is a directory filled with the HTMLs of the retrieved webpages, with the purpose of visually verifying whether a specific domain was actually hidden by means of a blockpage or not.



1	google.com
2	amazonaws.com
3	microsoft.com
4	facebook.com
5	akamai.net

Figure A.1. `input.csv`'s format.

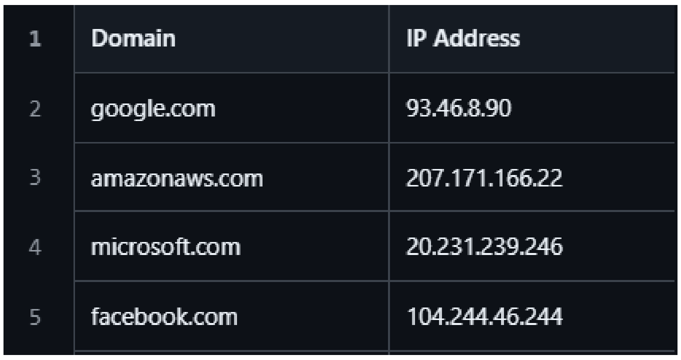
.



Domain	IP Address
google.com	172.217.20.174
amazonaws.com	72.21.206.80
microsoft.com	20.236.44.162
facebook.com	185.60.219.35

Figure A.2. `output_control.csv`'s format.

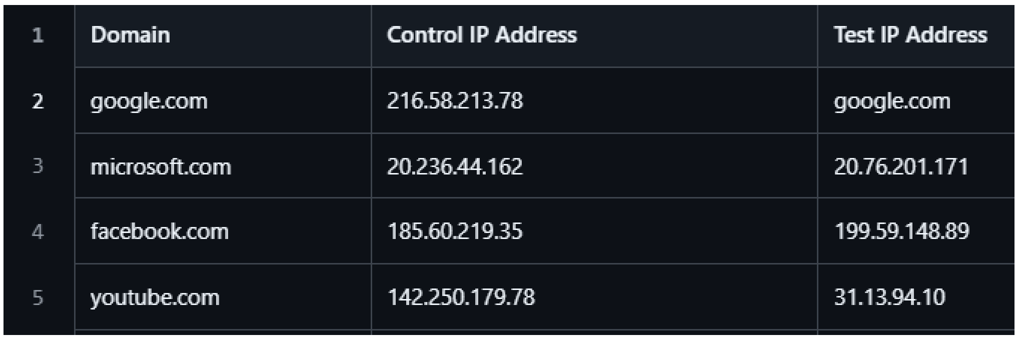
.



Domain	IP Address
google.com	93.46.8.90
amazonaws.com	207.171.166.22
microsoft.com	20.231.239.246
facebook.com	104.244.46.244

Figure A.3. `output_test.csv`'s format.

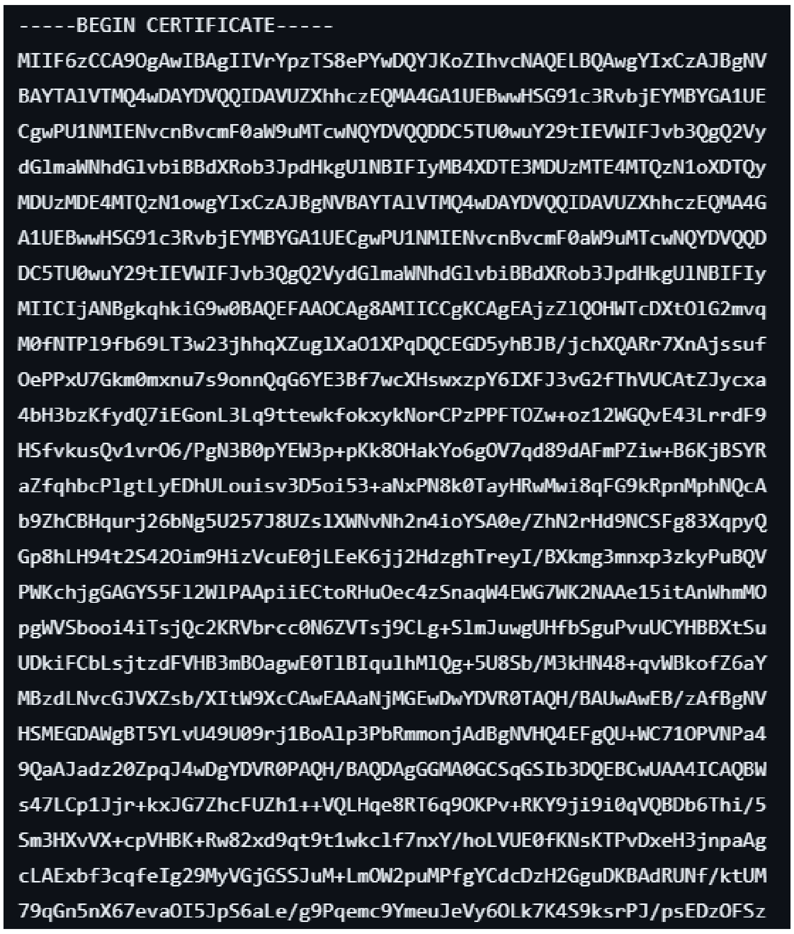
.



Domain	Control IP Address	Test IP Address
google.com	216.58.213.78	google.com
microsoft.com	20.236.44.162	20.76.201.171
facebook.com	185.60.219.35	199.59.148.89
youtube.com	142.250.179.78	31.13.94.10

Figure A.4. `mismatched_resolutions.csv`'s format.

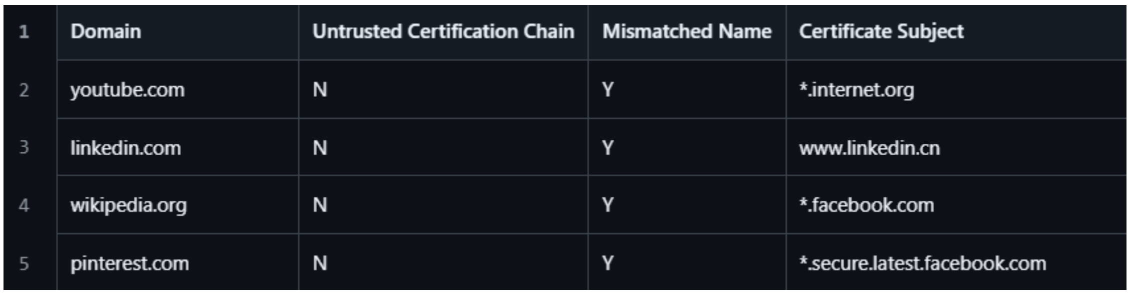
.



```
-----BEGIN CERTIFICATE-----
MIIF6zCCA9OgAwIBAgIIVrYpzTS8ePYwDQYJKoZIhvcNAQELBQAwgYIxCzAJBgNV
BAYTAlVTMQ4wDAYDVQQIDAVUZXhhczEQMA4GA1UEBwwHSG91c3RvbjEYMBYGA1UE
CgwPU1NMIENvcnBvcmF0aW9uMTcwNQYDVQQDDC5TU0wuY29tIEVWIFJvb3QgQ2Vy
dGlmaWNhdGlvbiBBdXRob3JpdHkgUlNBIFIyMB4XDTE3MDUzMTE4MTQzN1oXDTQy
MDUzMDE4MTQzN1owgYIxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVUZXhhczEQMA4G
A1UEBwwHSG91c3RvbjEYMBYGA1UECgwPU1NMIENvcnBvcmF0aW9uMTcwNQYDVQQD
DC5TU0wuY29tIEVWIFJvb3QgQ2VydGlmaWNhdGlvbiBBdXRob3JpdHkgUlNBIFIy
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAjzZlQOHWTcDXtOlG2mvq
M0fNTPl9fb69LT3w23jhhqXZuglXaO1XPqDQCEGD5yhBJB/jchXQARr7XnAjssuf
OePPxU7Gkm0mxnu7s9onnQqG6YE3Bf7wcXHswxzpY6IXFJ3vG2fThVUCAtZJycxa
4bH3bzKfydQ7iEGonL3Lq9ttewkfokxykNorCPzPPFTOZw+oz12WGQvE43LrrdF9
HSfvkusQv1vrO6/PgN3B0pYEW3p+pKk8OHakYo6gOV7qd89dAFmPZiw+B6KjBSYR
aZfqhbcPlgtLyEDhULouisv3D5oi53+aNxPN8k0TayHRwMwi8qFG9kRpnMphNQcA
b9ZhCBHqurj26bNg5U257J8UZslXWNvNh2n4ioYSA0e/ZhN2rHd9NCSFg83XqpyQ
Gp8hLH94t2S42Oim9HizVcuE0jLEeK6jj2HdzghTreyI/BXkmg3mnxp3zkyPuBQV
PWKchjgGAGYS5Fl2WlPAApiiECtoRHuOec4zSnaqW4EWG7WK2NAAe15itAnWhmMO
pgWVSbooi4iTsjQc2KRVbrcc0N6ZVTsj9CLg+SlmJuwgUHfbSguPvuUCYHBBXtSu
UDkiFCbLsjtzdFVHB3mBOagwE0TlBIqulhMlQg+5U8Sb/M3kHN48+qvWBkofZ6aY
MBzdLNvcGJVXZsb/XItW9XcCAwEAAaNjMGEwDwYDVR0TAQH/BAUwAwEB/zAfBgNV
HSMEGDAWgBT5YLvU49U09rj1BoAlp3PbRmmonjAdBgNVHQ4EFgQU+WC71OPVNPa4
9QaAJadz20ZpqJ4wDgYDVR0PAQH/BAQDAgGGMA0GCSqGSIb3DQEBCwUAA4ICAQBW
s47LCp1Jjr+kxJG7ZhcFUZh1++VQLHqe8RT6q9OKPv+RKY9ji9i0qVQBDb6Thi/5
Sm3HXvVX+cpVHBK+Rw82xd9qt9t1wkclf7nxY/hoLVUE0fKNsKTPvDxeH3jnpaAg
cLAExbf3cqfeIg29MyVGjGSSJuM+LmOW2puMPfgYCdcDzH2GguDKBAdRUNf/ktUM
79qGn5nX67evaOI5JpS6aLe/g9Pqemc9YmeuJeVy6OLk7K4S9ksrPJ/psEDzOFSz
```

Figure A.5. `mozilla_trusted_ca_list.crt`'s format.

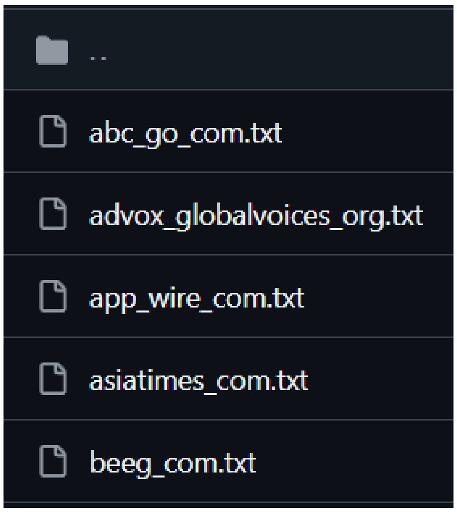
.



1	Domain	Untrusted Certification Chain	Mismatched Name	Certificate Subject
2	youtube.com	N	Y	*.internet.org
3	linkedin.com	N	Y	www.linkedin.cn
4	wikipedia.org	N	Y	*.facebook.com
5	pinterest.com	N	Y	*.secure.latest.facebook.com

Figure A.6. `certificates.csv`'s format.

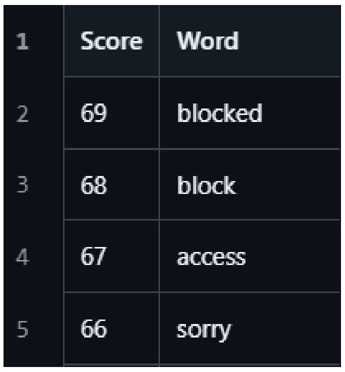
.



```
..
abc_go_com.txt
advox_globalvoices_org.txt
app_wire_com.txt
asiatimes_com.txt
beeg_com.txt
```

Figure A.7. `/certificates`' directory structure.

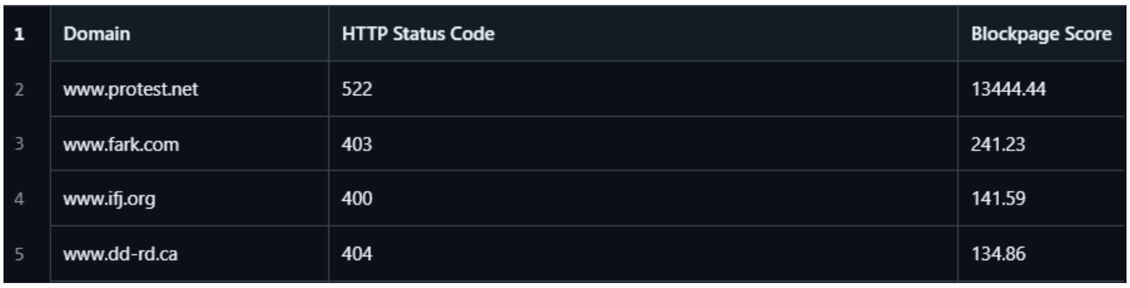
.



1	Score	Word
2	69	blocked
3	68	block
4	67	access
5	66	sorry

Figure A.8. `blockpages_typical_words.csv`'s format.

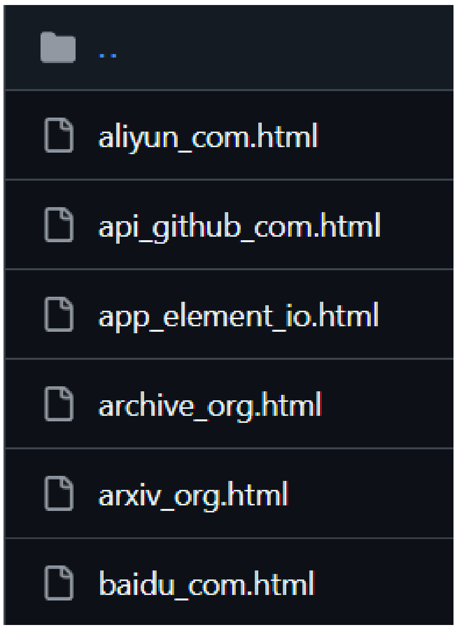
.



1	Domain	HTTP Status Code	Blockpage Score
2	www.protest.net	522	13444.44
3	www.fark.com	403	241.23
4	www.ifj.org	400	141.59
5	www.dd-rd.ca	404	134.86

Figure A.9. `webpages.csv`'s format.

.



```
..
aliyun_com.html
api_github_com.html
app_element_io.html
archive_org.html
arxiv_org.html
baidu_com.html
```

Figure A.10. `/webpages`' directory structure.

.

Appendix B

Developer's Manual

B.1 control_resolutions.py

This Python script generates the trusted resolutions list. The list of functions is:

- `percentage()`, which prints the partial completion percentage.
- `input()`, which takes as input the list of domains located in the `input.csv` file and saves it in a local variable.
- `output()`, which generates the trusted resolutions list and outputs it into the `output_control.csv` file.

The imported modules are `subprocess`, `csv` and `sys`.

B.2 censorship_detector.py

This Python script performs all the required analysis on the untrusted resolver. The list of functions is:

- `percentage()`, which prints the partial completion percentage.
- `input()`, which takes as input the list of domains located in the `input.csv` file and saves it in a local variable.
- `test_resolutions()`, which generates the untrusted resolutions list and outputs it into the `output_test.csv` file;
- `comparison()`, which compares the trusted and untrusted resolutions lists located in the `output_control.csv` and `output_test.csv` files, and outputs the mismatched resolutions list in the `mismatched_resolutions.csv` file.
- `certificate_check()`, which retrieves and analyses the certificates owned by the domains present in the `mismatched_resolutions.csv` file, and outputs the obtained information in the `certificates.csv` file; this function checks if a certificate's chain is valid and if it presents a mismatched name; only certificates that present at least one of these two features are saved in the `certificates.csv` file, and they are stored in their complete format in the `/certificates` directory.
- `blockpage_score_calculator()`, which calculates the blockpage score for a domain; this score calculation is based on the HTML of the retrieved webpage and on the list of words located in the `blockpage_typical_words.csv` file.

- `csv_sorter()`, which sorts the elements of a CSV file in descending order based on the value of column number 2; this function is used to sort the elements of `webpages.csv` according to their blockpage score value, putting the ones with a higher score on top of the list.
- `curler()`, which performs an HTTP connection for every domain present in the `certificates.csv` file, retrieving the HTTP status codes and the pages' HTMLs; it outputs the retrieved information in the `webpages.csv` file, while saving the full HTMLs of each domain in the `/webpages` directory.

The imported modules are `time`, `signal`, `subprocess`, `csv` and `sys`.

B.3 Timeouts

Most of the functions described run some terminal commands. Such commands are executed by means of the `subprocess.run()` and `subprocess.Popen()` functions.

Due to the nature of the commands executed, combined to the fact that such operations are performed by using badly configured DNS resolvers, some connections can take a long time to output a result. For optimization reasons, some of these functions include a manually inserted timeout, which has the purpose of dropping the connection attempt if the operation takes too long.

If there is no interest in running a fast version of this tool, it is possible to rise the limits set in such timeouts, with the purpose of obtaining a larger number of responses, of course along with an increase in time of execution.

Bibliography

[1] L.Schwittmann, M.Wander, T.Weis, “Domain Impersonation is Feasible: A Study of CA Domain Validation Vulnerabilities”, 2019 IEEE European Symposium on Security and Privacy (EuroS&P), Stockholm (Sweden), 17-19 June 2019, pp. 544-559, DOI 10.1109/EuroSP.2019.00046

[2] G.Akiwate, R.Sommese, M.Jonker, Z.Durumeric, K.C.Claffy, G.M.Voelker, S.Savage, “Retroactive Identification of Targeted DNS Infrastructure Hijacking”, IMC ’22: Proceedings of the 22nd ACM Internet Measurement Conference, Nice (France), 25-27 October 2022, pp. 14-32, DOI 10.1145/3517745.3561425

[3] E.Tsai, D.Kumar, R.S.Raman, G.Li, Y.Eiger, R.Ensafi, “CERTainty: Detecting DNS Manipulation at Scale using TLS Certificates”, Proceedings on Privacy Enhancing Technologies Symposium 2023, Lusanne (Switzerland), 10-15 July 2023, pp. 122-137, DOI 10.56553/popets-2023-0073

[4] R.Roberts, Y.Goldschlag, R.Walter, T.Chung, A.Mislove, D.Levin, “You Are Who You Appear to Be: A Longitudinal Study of Domain Impersonation in TLS Certificates”, CCS ’19: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, London (United Kingdom), 11-15 November 2019, pp. 2489-2504, DOI 10.1145/3319535.3363188

[5] T.Dai, H.Shulman, M.Waidner, “Off-Path Attacks Against PKI”, CCS ’18: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto (Canada), 15-19 October 2018, pp. 2213-2215, DOI 10.1145/3243734.3278516

[6] “How to nab a HTTPS cert for a stranger’s website: Step one, shatter those DNS queries...”, `https://www.theregister.com/2018/09/06/certificate_authority_dns_validation/`, Accessed: 04-09-2024

[7] P. Mockapetris, “DOMAIN NAMES - CONCEPTS AND FACILITIES”, RFC-1034, November 1987, DOI 10.17487/RFC1034

[8] P. Mockapetris, “DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION”, RFC-1035, November 1987, DOI 10.17487/RFC1035

[9] “Domain Name System”, `https://en.wikipedia.org/wiki/Domain_Name_System`, Accessed: 01-11-2024

[10] “Subdomain”, `https://en.wikipedia.org/wiki/Subdomain`, Accessed: 01-11-2024

[11] “What are DNS records?”, `https://www.ibm.com/topics/dns-records`, Accessed: 24-09-2024

[12] R.Arends, R.Austein, M.Larson, D.Massey, S.Rose, “DNS Security Introduction and Requirements”, RFC-4033, March 2005, DOI 10.17487/RFC4033

[13] P.Hoffman, J.Schlyter, “The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA”, RFC-6698, August 2012, DOI 10.17487/RFC6698

[14] S.Chokhani, W.Ford, R.Sabett, C.Merrill, S.Wu, “Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework”, RFC-3647, November 2003, DOI 10.17487/RFC3647

[15] S.Kent, “Privacy Enhancement for Internet Electronic Mail: Part II: Certificate-Based Key Management”, RFC-1422, February 1993, DOI 10.17487/RFC1422

[16] D.Cooper, S.Santesson, S.Farrell, S.Boeyen, R.Housley, W.Polk, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile”, RFC-5280, May 2008, DOI 10.17487/RFC5280

[17] S.Santesson, M.Myers, R.Ankney, A.Malpani, S.Galperin, C.Adams, “X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP”, RFC-6960, June 2013, DOI 10.17487/RFC6960

[18] L.Daigle, “WHOIS Protocol Specification”, RFC-3912, September 2004, DOI 10.17487/RFC3912

[19] “What’s the difference between DV, OV & EV SSL certificates?”, `https://www.digicert.com/difference-between-dv-ov-and-ev-ssl-certificates`, Accessed: 04-09-2024

[20] Let’s Encrypt, `https://letsencrypt.org/`

[21] “Usage statistics and market shares of SSL certificate authorities for websites”, `https://w3techs.com/technologies/overview/ssl_certificate`, Accessed: 04-09-2024

[22] “Kenapa Harga SSL Certificate Berbeda?”, `https://gudangssl.id/blog/kenapa-harga-ssl-certificate-berbeda/`, Accessed: 01-11-2024

[23] R.Barnes, J.Hoffman-Andrews, D.McCarney, J.Kasten, “Automatic Certificate Management Environment (ACME)”, RFC-8555, March 2019, DOI 10.17487/RFC8555

[24] M.Nystrom, B.Kaliski, “PKCS #10: Certification Request Syntax Specification Version 1.7”, RFC-2986, November 2000, DOI 10.17487/RFC2986

[25] B.Laurie, A.Langley, E.Kasper, “Certificate Transparency”, RFC-6962, June 2013, DOI 10.17487/RFC6962

[26] “What is Certificate Transparency (CT)? How does it work?”, `https://cheapsslsecurity.com/blog/what-is-certificate-transparency-ct-how-does-it-work/`, Accessed: 01-11-2024

[27] “Secure logging schemes and Certificate Transparency”, `https://www.douglas.stebila.ca/research/papers/ESORICS-DGHS16/`, Accessed: 24-09-2024

[28] “DNS Cache Poisoning - The Next Generation”, `https://www.secureworks.com/blog/dns-cache-poisoning`, Accessed: 24-09-2024

[29] “CAISRNP.bind”, `https://packetstormsecurity.com/files/30522/CAISRNP.bind.html`, Accessed: 24-09-2024

[30] “What is DNS spoofing?”, `https://threatcop.medium.com/what-is-dns-spoofing-9de305d34e62`, Accessed: 01-11-2024

[31] “Empirical Analysis of Internet Filtering in China”, `https://cyber.harvard.edu/filtering/china/`, Accessed: 20-10-2024

[32] “Deconstructing the Great Firewall of China”, `https://www.thousandeyes.com/blog/deconstructing-great-firewall-china`, Accessed: 20-10-2024

[33] “CERTainty: Detecting DNS Manipulation at Scale using TLS Certificates”, `https://censoredplanet.org/certainty`, Accessed: 20-10-2024

[34] V.Le Pochat, T.Van Goethem, S.Tajalizadehkhoob, M.Korczynski, W.Joosen, “TRANCO: A Research-Oriented Top Sites Ranking Hardened Against Manipulation”, Proceedings of the 26th Annual Network and Distributed System Security Symposium, San Diego, CA (USA), 24-27 February 2019, DOI 10.14722/ndss.2019.23386

[35] “TRANCO”, `https://tranco-list.eu/`, Accessed: 20-10-2024

[36] “Citizen Lab and Others. 2014. URL Testing Lists Intended for Discovering Website Censorship.”, `https://github.com/citizenlab/test-lists`, Accessed: 20-10-2024

[37] “Mozilla Root Store Policy”, `https://www.mozilla.org/en-US/about/governance/policies/security-group/certs/policy/`, Accessed: 20-10-2024

[38] “curl”, `https://curl.se/`, Accessed: 20-10-2024

[39] “curl man page”, `https://curl.se/docs/manpage.html`, Accessed: 20-10-2024

[40] “Public DNS Servers in China”, `https://dnschecker.org/public-dns/cn`, Accessed: 20-10-2024

[41] “nslookup”, `https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/nslookup`, Accessed: 20-10-2024

[42] “nslookup(1) - Linux man page”, `https://linux.die.net/man/1/nslookup`, Accessed: 20-10-2024

[43] “OpenSSL”, `https://openssl.org/`, Accessed: 20-10-2024

[44] “OpenSSL Documentation - s_client”, `https://docs.openssl.org/1.0.2/man1/s_client/`, Accessed: 20-10-2024

[45] “Mozilla Included CA Certificate List”, `https://wiki.mozilla.org/CA/Included_Certificates`, Accessed: 01-11-2024

[46] "Blocking Fingerprints", https://github.com/ooni/blocking-fingerprints?tab=readme-ov-file#blocking-fingerprints, Accessed: 20-10-2024

[47] "Censorship Detector", https://github.com/rfiorilla/thesis/