2014-2015

UFR MIM

Master 2 - Security of information systems

INTERNSHIP REPORT

Monitoring of the SSL certificates revocation lists

Pierre CAILLET

Supervisor: Alexandre DULAUNOY

Abstract

The main part of communications to protect on the nets are operated with the TLS (Transport Layer Security) protocol, or its predecessor SSL (Secure Socket Layer). This secure protocol uses the asymetric cryptographic principles to establish the connections. Public keys, necessary to everyone who wants to connect to a service to identify it, are encapsulated in SSL certificates, which could be seen as the real keys of all these secured exchanges.

These certificates and especially the way to revoke them are the subject of this internship. Certificates are created, live during a defined time and then finally die. Sometimes a certificate can live again, or be considered as alive again. They can be delivered by secure authority but also by entities which seem to be less secure apparently. A certificate can identify an authority, which is going too to deliver certificates to end entities, or also to intermediate authorities. All this system creates paths of certification, which can become more and more complex. This simple view has the objective to show that this complicated system, which is the fundation of all SSL communications, is also vulnerable.

The integrity of this system is a major part of the security of all the numeric communications.

Acknowledgments

For his availability and his openmindedness, I must warmy thank Alexandre DULAUNOY, CIRCL manager and computer security analyst. Furthermore, in addition of my supervisor, I would like to thank Raphaël VINOT and Richard WAGENER, computer security analysts, since they often helped me too to solve my coding problems. For their kindness and this extraordinary cheerfulness in working, thanks also to:

- Sascha ROMMELFANGEN, computer security analyst
- Steve CLEMENT, computer security analyst
- Michael HAMM, computer security analyst

Finally, thanks to all the people of SMILE, especially the managing director, Pascal STEICHEN, and the human ressources manager, Rita BRESSANUTTI.

Contents

Introduction

Officer in the french army, I passed an exam in 2013 in order to specialize on information systems security. Since human ressources needs are high as far as cybersecurity is concerned, the army send me to Metz, for one year in master 1, general computer science, and one year in master 2, with an information systems security speciality.

To conclude this period of two years of studies at the university of Lorraine, I spent six months in Luxembourg at **CIRCL, computer incident response center luxembourg**, as an intern.

The subject of this internship was to **monitor the SSL certificates revocation lists**. This document presents the different aspects of this work, from the definition of the objectives to the final results, with a description of the main tools used.

Chapter 1

Presentation of CIRCL

CIRCL(Computer Incident Response Center Luxembourg) is an entity of the ministry of economy of Luxembourg. Its aim is to provide responses to computer security incidents and threats. For the private sector, CIRCL is the CERT (computer emergency response team) and works also for communes and non-governmental entities in Luxembourg.

The 6 researchers of CIRCL acts like a fire-brigade, which is able to react efficiently and quickly when an incident occurs or when a threat is identified.

1.1 Security made in Lëtzebuerg

CIRCL is part of a GIE (in french: "groupe d'interet economique"[1]) named SMILE (security made in Lëtzebuerg). SMILE is also operating two other entities, CASES and AWARE. The figure 1.1 shows the organization of SMILE.

On the CIRCL website, these missions are pointed out:

- *provide a systematic response facility to ICT-incidents*
- *coordinate communication among national and international incident response teams during security emergencies and to help prevent future incidents*
- *support ICT users in Luxembourg to recover quickly and efficiently from security incidents*
- *minimize ICT incident-based losses, theft of information and disruption of services at a national level*

[1]There is no english translation for such an entity

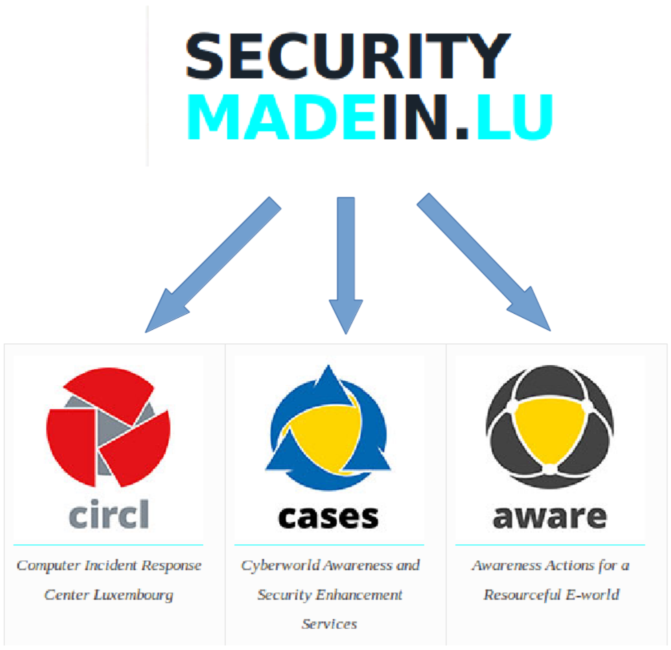

SECURITY
MADEIN.LU
circl
Computer Incident Response Center Luxembourg
cases
Cyberworld Awareness and Security Enhancement Services
aware
Awareness Actions for a Resourceful E-world

Figure 1.1: Organization of SMILE GIE

- *gather information related to incident handling and security threats to better prepare future incidents management and provide optimized protection for systems and data*

- *provide a security related alert and warning system for ICT users in Luxembourg*

- *foster knowledge and awareness exchange in ICT security*

SMILE was born from the need to preserve and enhance, within a permanent structure, the results obtained through many years of work in the field of informations security activity of CASES and CIRCL and the results of all Luxembourg activities supported by the European program *"Safer Internet"* (known under the name BEE SECURE). Its activities and goals are build on the *"Information and Communication Systems Security Strategy"* defined by the Ministry of the Economy. The interdisciplinary team of SMILE has 18 members. There skills are very large. Thus, people are working on information security, but also on marketting, numeric-oriented laws or audit.

All the financial purposes and aims of SMILE are defined by its operating ministry in a five-years agreement. The entity has been created on the 5th of May, 2010.

The activities of CASES (cyberworld awareness and security enhancement services) are quite different from these of CIRCL. CASES is in charge of audit and guidance with its clients. The team is able to evaluate the level of security awareness and organization in a company, and then to product advice to improve behaviours and limit the risks of attack.

Finally, AWARE is a global project on information security awareness. The aim is to provide the good information to encourage people to use new ways of communication with the maximum of security.

The stakeholders for the project are the following:

First, the state of Luxembourg, with three of its ministries:

1. Ministry of Family, Integration and the Greater Region
2. Ministry of Education, Childhood and Youth
3. Ministry of the Economy

The local government federations SIGI (*"Syndicat Intercommunal de Gestion Informatique"*) and SYVICOL (*"Syndicat des Villes et Communes du Luxembourg"*) are also managing SMILE g.i.e. and all its activities in a partnership. As a consequence, the management board is composed of representatives of all this stakeholding members.

1.2 The team of security specialists of CIRCL

The 6 members of CIRCL are high-skilled people, analysts and researchers who passed a Phd or the same level of qualification. It's a real art to combine their works and goals with the other parts of SMILE, because their technical and specific work is really different from the others. Finally, the whole seems to work perfectly, with the good level of compartmentalization.

1.3 Some projets of CIRCL

1.3.1 MISP

Malware Information Sharing Platform (MISP) is a platform for storing and sharing some indicators about attacks and malwares. Then, the members benefit from all the knowledge of the others users. As a consequence, it facilitates the improvment of preventive actions. Companies or organizations could ask for an access to the platform to CIRCL, in order to collaborate.

1.3.2 CIRCLean

CIRCLean is freely available on Github, it's a specific operating system for raspberry pi. Its goal is unique. The system copies on his memory the content of a suspect key. Then, it converts the content into a secure format and copies it on another key, trusted by the user. Thus, the user can safely recover its documents, in a readable format.

1.3.3 HACK.lu

HACK.LU is a conference operated by CIRCL which takes place in the city of Luxembourg, generally in October. People can discuss about privacy, computer security and all the subjects close to it. An important number of ICT companies are involved in the conference with partnerships. Everybody has the possibility to suggest a talk, but the choice is finally done by SMILE. Places for this convention are limited since the level and the quality of talks are very known as high.

Obviously, CIRCL is maintaining a lot of other projects, which are for a numerous part available on the github account[2] of the entity or on the website[3].

[2]https://github.com/CIRCL
[3]www.circl.lu

Chapter 2

Results of the internship

2.1 Some points to clarify

First, it's important to clarify some global and major points.

2.1.1 Principles of PKI

Public key infrastructures are based on the principle of *"trusted third party"*. If two systems are to communicate together, they have to ask a third trusted party if the other entity is the entity that it pretends to be, and if the communication could be done on a secure channel. These third trusted parties are named *"certification authorities (CA)"* and edits SSL certificates, which should prove the identity of a subject. The CA is called issuer of the CRL. The figure 2.1 shows an overview of this process.

2.1.2 SSL connection

The establishment of an SSL connection is shown on the figure 2.3. It's during the *"key exchange"* step that the client has to verify the integrity of the server certificate, with a checking of the available CRLS.

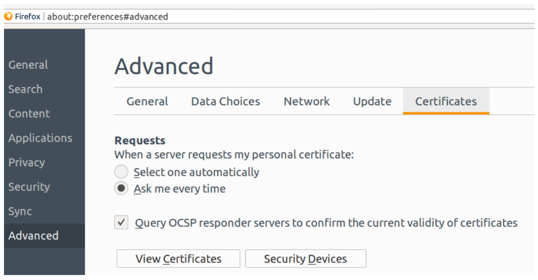
This is with this verification that the *"trusted third party"* has its role to play. Nevertheless, this role can be played offline, it's the main interest of the CRL processing. Actually, depending on the system, the different CRL are stored in cache, and regularly updated when the system is connected on the net. On the contrary, online systems of checking like OCSP[1] are not efficient anymore if the system is disconnected from the responder. Thus, the last version of the firefow browser just operate

[1]Online Certificate Status Protocol

```
+---+
| C |                       +------------+
| e | <-------------------->| End entity |
| r |       Operational     +------------+
| t |       transactions          ^
| i |      and management         |   Management
| f |       transactions          |   transactions        PKI
| i |                             |                      users
| c |                             v
| a | =======================  +--+------------+  ==============
| t |                          ^               ^
| e |                          |               |         PKI
|   |                          v               |      management
| & |                       +------+           |       entities
|   | <---------------------|  RA  |<----+     |
| C |  Publish certificate  +------+     |     |
| R |                                    |     |
| L |                                    |     |
|   |                                    v     v
| R |                                +------------+
| e | <------------------------------|     CA     |
| p |   Publish certificate          +------------+
| o |   Publish CRL                     ^      ^
| s |                                   |      |  Management
| i |                +------------+     |      |  transactions
| t | <--------------| CRL Issuer |<----+      |
| o |   Publish CRL  +------------+             v
| r |                                      +------+
| y |                                      |  CA  |
+---+                                      +------+
```

Figure 2.1: Overview of PKI

an OCSP checking. If the third party is not responding, the system will authorize by default the connection. The 2.2 figure shows that it's impossible to force the verification of CRL with firefox 40.0.3.



Advanced

Requests
When a server requests my personal certificate:
○ Select one automatically
● Ask me every time

✓ Query OCSP responder servers to confirm the current validity of certificates

Figure 2.2: On firefox, no way to use CRL

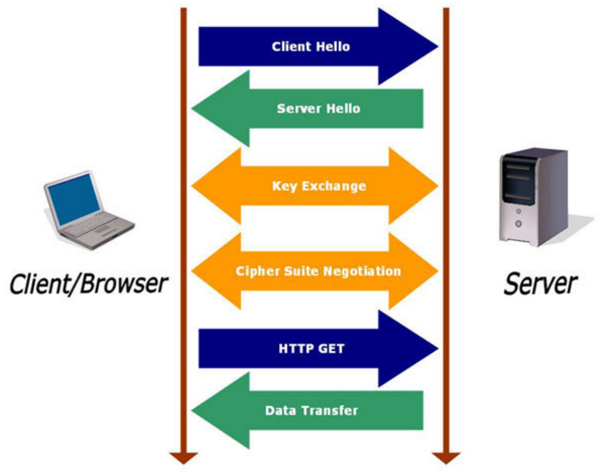


Figure 2.3: SSL connection establishment

2.1.3 DER and PEM encoding

All the CRL processed during this internship are PEM or DER encoded ones. It could be useful to present what are these two formats, which are used actually to encode the same thing.

DER (Distinguished Encoding Rules) is one of ASN.1 encoding rules defined in ITU-T X.690[2] specification. ASN.1 encoding rules can be used to encode any data object into a binary file. The basic encoding rule of DER is that a data value of all data types shall be encoded as four components in the following order:

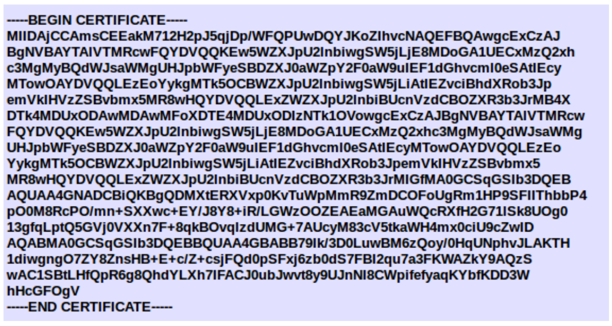
- Identifier octets
- Length octets
- Contents octets
- End-of-contents octets

DER is used as the most popular encoding format to store X.509[3] certificates in files. Those certificate DER files are binary files, which can not be viewed with text editors. But they can be processed by application without any problems.

On the other hand, PEM is an abbreviation for Privacy Enhanced Mail (RFC 1421 - RFC 1424), an early standard for securing electronic

[2]International Telegraph Union, Telecommunication Standardization Sector

[3]In this report, SSL certificates and X509 certificates have the same meaning



```
-----BEGIN CERTIFICATE-----
MIIDAjCCAmsCEEakM712H2pJ5qjDp/WFQPUwDQYJKoZIhvcNAQEFBQAwgcExCzAJ
BgNVBAYTAlVTMRcwFQYDVQQKEw5WZXJpU2lnbiwgSW5jLjE8MDoGA1UECxMzQ2xh
c3MgMyBQdWJsaWMgUHJpbWFyeSBDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eSAtIEcy
MTowOAYDVQQLEzEoYykgMTk5OCBWZXJpU2lnbiwgSW5jLiAtIEZvciBhdXRob3Jp
emVkIHVzZSBvbmx5MR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMB4X
DTk4MDUxODAwMDAwMFoXDTE4MDUxODIzNTk1OVowgcExCzAJBgNVBAYTAlVTMRcw
FQYDVQQKEw5WZXJpU2lnbiwgSW5jLjE8MDoGA1UECxMzQ2xhc3MgMyBQdWJsaWMg
UHJpbWFyeSBDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eSAtIEcyMTowOAYDVQQLEzEo
YykgMTk5OCBWZXJpU2lnbiwgSW5jLiAtIEZvciBhdXRob3JpemVkIHVzZSBvbmx5
MR8wHQYDVQQLExZWZXJpU2lnbiBUcnVzdCBOZXR3b3JrMIGfMA0GCSqGSIb3DQEB
AQUAA4GNADCBiQKBgQDMXtERXVxp0KvTuWpMmR9ZmDCOFoUgRm1HP9SFIIThbbP4
pO0M8RcPO/mn+SXXwc+EY/J8Y8+iR/LGWzOOZEAEaMGAuWQcRXfH2G71lSk8UOg0
13gfqLptQ5GVj0VXXn7F+8qkBOvqlzdUMG+7AUcyM83cV5tkaWH4mx0ciU9cZwID
AQABMA0GCSqGSIb3DQEBBQUAA4GBABB79Ik/3D0LuwBM6zQoy/0HqUNphvJLAKTH
1diwgngO7ZY8ZnsHB+E+c/Z+csjFQd0pSFxj6zb0dS7FBI2qu7a3FKWAZkY9AQzS
wAC1SBtLHfQpR6g8QhdYLXh7IFACJ0ubJwvt8y9UJnNI8CWpifefyaqKYbfKDD3W
hHcGFOgV
-----END CERTIFICATE-----
```

Figure 2.4: A PEM encoded certificate

mail. PEM never has been widely adopted as Internet Mail Standard. The PEM format is often used now for representing a certificate, a certificate request or another things in US-ASCII[4] by base64[7] encoding it and putting the encoding between the typical PEM delimiters.

Actually, a .pem file is just a base64 encoded .der file. In addition, for our certificates or CRL, the file begins with —–BEGIN CERTIFICATE(or CRL)—- and ends with —–END CERTIFICATE(or CRL)—–.

An example of a PEM encoded certificate is shown on figure 2.4.

2.1.4 RFC5280

This document, “Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile” [3], defines precisely how a SSL certificate and a CRL are composed. In order to give a right idea of the composition of this document, I put its table of contents on the appendice C. In this appendice, I pointed out some importants parts of this document. Obviously, it was impossible to put all the document in this report, since it counts 150 pages.

2.1.5 What is a SSL certificate?

A SSL certificate establishes with asymetric cryptographic tools the link between a subject, an entity, and a public key. This certificate is guaranteed by the certification authority which has edited it. Its time to live is defined from the beginning. Finally, it could be revoked before the end of its life, for several reasons.

A SSL certificate could be signed by itself. Actually, the issuer and the subject of these certificates are the same entity. They could be the

[4]American Standard Code for Information Interchange

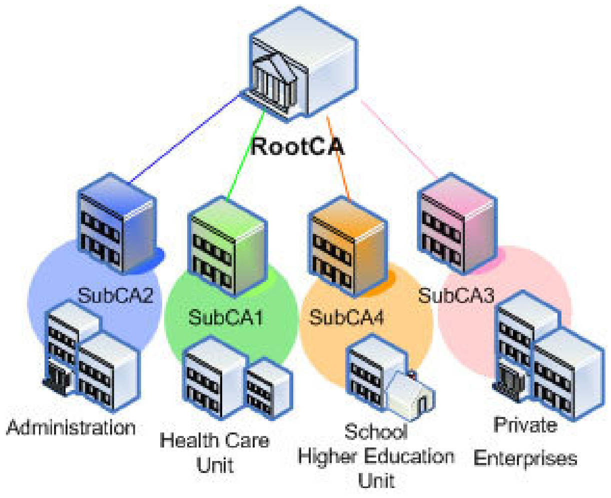
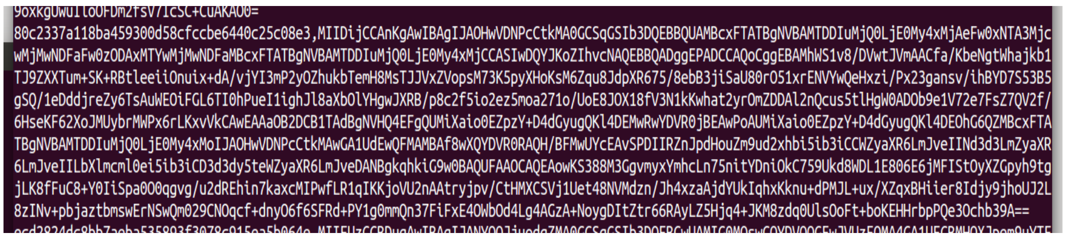


Figure 2.5: A self-signed root certificate



```
9oxkgUwuIloOFDm2fsV71cSC+CuAKAO0=
80c2337a118ba459300d58cfccbe6440c25c08e3,MIIDijCCAnKgAwIBAgIJAOHwVDNPcCtkMA0GCSqGSIb3DQEBBQUAMBcxFTATBgNVBAMTDDIuMjQ0LjE0My4xMjAeFw0xNTA3Mjc
wMjMwNDFaFw0zODAxMTYwMjMwNDFaMBcxFTATBgNVBAMTDDIuMjQ0LjE0My4xMjCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAMhWS1v8/DVwtJVmAACfa/KbeNgtWhajkb1
TJ9ZXXTum+SK+RBtleeiiOnuix+dA/vjYI3mP2yOZhukbTemH8MsTJJVxZVopsM73K5pyXHoKsM6Zqu8JdpXR675/8ebB3jiSaU80rO51xrENVYwQeHxzi/Px23gansv/ihBYD7S53B5
gSQ/1eDddjreZy6TsAuWEOiFGL6TI0hPueI1ighJl8aXbOlYHgwJXRB/p8c2f5io2ez5moa271o/UoE8JOX18fV3N1kKwhat2yrOmZDDAl2nQcus5tlHgW0ADOb9e1V72e7FsZ7QV2f/
6HseKF62XoJMUybrMWPx6rLKxvVkCAwEAAaOB2DCB1TAdBgNVHQ4EFgQUMiXaio0EZpzY+D4dGyugQKl4DEMwRwYDVR0jBEAwPoAUMiXaio0EZpzY+D4dGyugQKl4DEOhG6QZMBcxFTA
TBgNVBAMTDDIuMjQ0LjE0My4xMoIJAOHwVDNPcCtkMAwGA1UdEwQFMAMBAf8wXQYDVR0RAQH/BFMwUYcEAvSPDIIRZnJpdHouZm9ud2xhbi5ib3iCCWZyaXR6LmJveIINd3d3LmZyaXR
6LmJveIILbXlmcml0ei5ib3iCD3d3dy5teWZyaXR6LmJveDANBgkqhkiG9w0BAQUFAAOCAQEAowKS388M3GgvmyxYmhcLn75nitYDniOkC759Ukd8WDL1E806E6jMFIStOyXZGpyh9tg
jLK8fFuC8+Y0IiSpa0O0qgvg/u2dREhin7kaxcMIPwfLR1qIKKjoVU2nAAtryjpv/CtHMXCSVj1Uet48NVMdzn/Jh4xzaAjdYUkIqhxKknu+dPMJL+ux/XZqxBHiier8Idjy9jhoUJ2L
8zINv+pbjaztbmswErNSwQm029CNOqcf+dny06f6SFRd+PY1g0mmQn37FiFxE40WbOd4Lg4AGzA+NoygDItZtr66RAyLZ5Hjq4+JKM8zdq0UlsOoFt+boKEHHrbpPQe3Ochb39A==
ecd2824dc8bb7aeba535893f3078c915ea5b064e,MIIEUzCCBDugAwIBAgIJANYQ0Jiuedq7MA0GCSqGSIb3DQEBCwUAMIG0MQswCQYDVQQGEwJVUzEQMA4GA1UECBMHQXJpem9uYTE
```

Figure 2.6: SSL certificate in an archive

root of a path of certification, as shown on the figure 2.5. Obviously, a so signed certificate can not be revoked.

Ideally, a non self-signed certificate should provide an effective and available way to publish an eventual revocation (CRL or OCSP). This rule is not always respected, statistics produced during this internship are proving it.

There are 3 versions of SSL certificate. The 2nd has completed the first, and was completed by the third. The two first versions are not efficient any more as far as security is concerned, because of a lack of provided data to identify the entities.

In the processed archives, a SSL certificate looks like in the figure 2.6

The archives provides first the hash of the content of the certificate, and in a second part a base64 encoded certificate, as explained above. The way to encode with the base64 principles is exposed in the appendice A. Each certificate is on a line. Then an archive is a very simple structure of data. The biggest archive of the website presented in the third chapter has 38 millions of lines. The weekly archives, which update the data, has on average 500000 lines.

2.1.6 What is a CRL

The CRL structure is precisely defined in the RFC5280 mentionned above, and can be precisely seen on the appendix C.

It is important to understand the data of the field *"nextUpdate"*. It is definitively not as the data *"notAfter"* in a certificate. Actually a CRL don't lose its accuracy because this date is outdated, and no new CRL has been published yet. The revoked certificates in the older CRL are always revoked, and will never recover their validity. Moreover, if a system misunderstands this concept, it could forbid a connection whereas the certificate of the client is still valid, just because the CRL seems to be outdated. When the CA publishes a new CRL, it should become the valid one for everybody, even if the *"nextUpdate"* date is not outdated.

The complexity of the monitoring of the CRL is based on the fact that CA behaviour are not always secure-oriented. As a matter of fact, processing the certification path is not always possible in a secure way.

2.2 Methodology

2.2.1 Holding the existing scripts

This step was absolutely necessary to understand the subject and to precise the aim. It has been unavoidable to hold before all the tools presented in the following chapter.

2.2.2 Definition of the objective

Initially I thought that my aim should be to provide a service to a client. For instance, to develop a tool which could give an opinion about the trust to grant to a particular SSL certificate. Quickly, I realized that the aim had to be redefine, in order to, first, collect, process and report the data. Collecting the data, a huge number of certificates, has to be done with the processing of archives from a specialized website, which permanently scan the internet with academic motivations.

2.2.3 Operational and data infrastructure

These choices are the most important in order to store the data in the better way. Actually, as the number of certificates is huge, a little difference in the data infrastructure could have big consequences as far as avaibality of the results is concerned. In addition, after these choices are made, it is very difficult to change it, since I have then to process again all the archives, that takes a very long time.

For example, the way to name a certificate is highly tricky. In the initial existing scripts, each certificate is stored with the hash of its content as a key. To find back this certificate, this hash is thus necessary and, obviously, to have the hash, the content is needed. As explained above, in a CRL, the revoked certificates are not entirely included. There are just the issuer of the CRL and the serial number, to identify the certificate. As a matter of fact, it was impossible to find back a stored certificate with a CRL, since it is impossible to build the hash of its content without this content.

As a consequence, I decided to name each certificate with the hash of the concatenation of the name of the issuer and the serial number. Then, this hash has nothing in common with the initial hash given by the archive, but it will permit to find back the certificate more easily, just from the CRL.

On the contrary, the way to name a CRL is the hash of its content. Indeed, it's the only secure manner to ensure that a CRL has changed or not. The issuer and the *"thisUpdate"* field is not enough to ensure that, or they are rather enough just for CA which have the perfect behaviour.

Gradually, the data structure has become more and more complex, the final scheme is shown on the figure 2.7.

2.3 Final production

One of the most recurrent question during the last three months of the analysis was the following: a CRL issuer is or not the issuer of the certificates it revokes? Obviously it should be. Thus I read in the RFC5280[3]:

"...CRL issuers issue CRLs. The CRL issuer is either the CA or an entity that has been authorized by the CA to issue CRLs. CAs publish CRLs to provide status information about the certificates they issued. However, a CA may delegate this responsibility to another trusted authority."

The delegation of authority is a very fuzzy concept. Actually, a delegated CRL issuer could revoke a certificate, since its CA private key is compromised for instance, without publishing the name of this compromised CA. Then all the systems, whatever there are, must trust this issuer for revoking all the certificates of this CA. If the issuer don't revoke its individualy, these systems have no ways to forbid a connection with an other certificate issued by this CA.

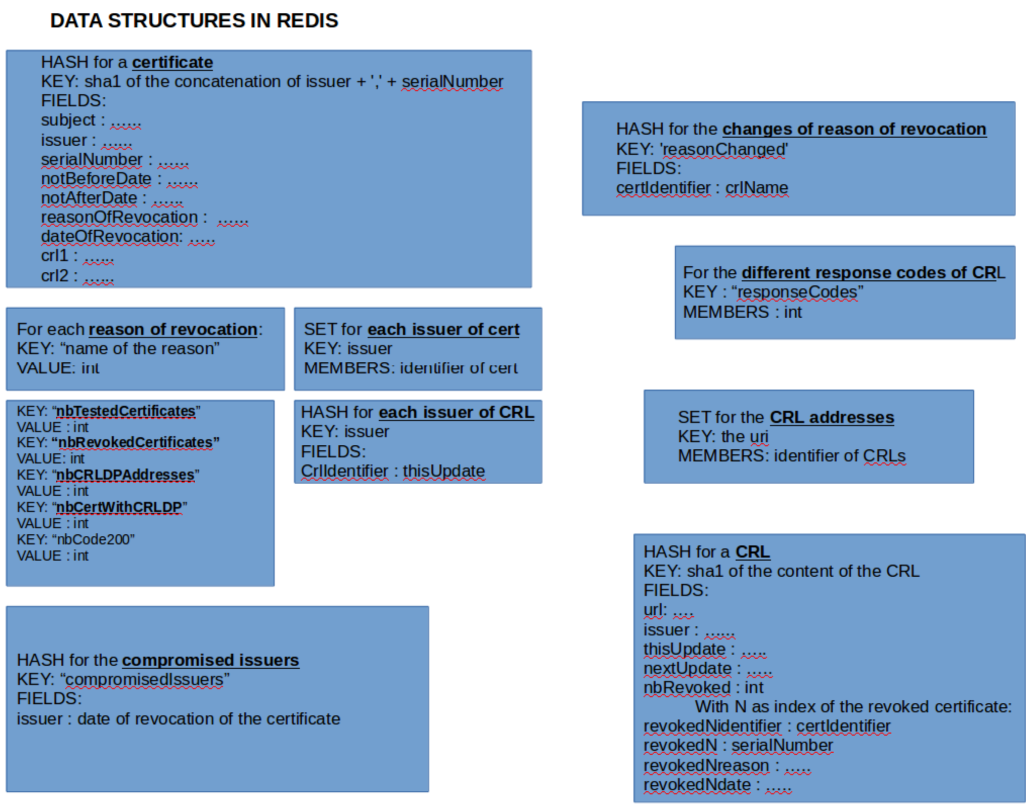

DATA STRUCTURES IN REDIS
HASH for a certificate
KEY: sha1 of the concatenation of issuer + ',' + serialNumber
FIELDS:
subject : ......
issuer : ......
serialNumber : ......
notBeforeDate : ......
notAfterDate : ......
reasonOfRevocation : ......
dateOfRevocation: .....
crl1 : ......
crl2 : ......
HASH for the changes of reason of revocation
KEY: 'reasonChanged'
FIELDS:
certIdentifier : crlName
For the different response codes of CRL
KEY : "responseCodes"
MEMBERS : int
For each reason of revocation:
KEY: "name of the reason"
VALUE: int
SET for each issuer of cert
KEY: issuer
MEMBERS: identifier of cert
KEY: "nbTestedCertificates"
VALUE : int
KEY: "nbRevokedCertificates"
VALUE: int
KEY: "nbCRLDPAddresses"
VALUE : int
KEY: "nbCertWithCRLDP"
VALUE : int
KEY: "nbCode200"
VALUE : int
HASH for each issuer of CRL
KEY: issuer
FIELDS:
CrlIdentifier : thisUpdate
SET for the CRL addresses
KEY: the uri
MEMBERS: identifier of CRLs
HASH for a CRL
KEY: sha1 of the content of the CRL
FIELDS:
url: ....
issuer : ......
thisUpdate : .....
nextUpdate : .....
nbRevoked : int
With N as index of the revoked certificate:
revokedNidentifier : certIdentifier
revokedN : serialNumber
revokedNreason : .....
revokedNdate : .....
HASH for the compromised issuers
KEY: "compromisedIssuers"
FIELDS:
issuer : date of revocation of the certificate

Figure 2.7: Data structure in Redis

As a conclusion, this delegation of revocation should not be used.

With statistics produced during this internship, it appears that a very few CA are using it, and it is a more secure behaviour.

To download just one time each CRL, it has been necessary to refactor the code. The major difficulty of this process is that all the CRL have to be downloaded. In the first version, each certificate was processed one by one, like a flow, and with the downloading of the eventual CRL. Thus, the issue was that if two certificates pointed towards the same URI of CRL, the corresponding CRL was downloaded two times. For the second time, the script didn't save it, but it was necessary to download it to conclude that it was already stored. After the refactoring, the certificates are processed for the beginning just to extract the different CRL addresses and store it in a redis set. Then, each address is tested and thus, one CRL is downloaded just one time.

Studying a huge number of SSL certificates allows to detect tendancies and strange behaviours of certifications authorities. The scripts developped permit, for each new archive published, to process it automatically, to update the CRL repository and the redis storage, and then to publish a new web page with the updated statistics.

After the processing of each new archive, a web page is automatically updated to provide available data to everyone. The figure 2.9,2.8 and 2.12 shows examples of content on this page.

The page displays for example on figure 2.8 the proportion of SSL certificates which are issued by a CA, thus their issuer has the same name as their subject.

Moreover, the proportion of non self signed certificates which provide a CRL distribution point is monitored, as shown on figure 2.9.

The number of CRL distribution addresses are about 300000. This number has to be understand, because a lot, even the majority, are no responding. Around 12 percents of these addresses are responding normally. Some CA, like godaddy, are using a lot of addresses, and some of them are not activated any more. Some CA have stopped their activities too.

Then, the programs monitor the number of non self signed certificates which provide OCSP information, as shown in figure 2.10 and the non self signed certificates which provide no way of revocation (CRL or OCSP), as shown in figure 2.11. This last number could be interesting, because it points out a kind of certificate which has not to be trusted.

As a last example, the different reasons of revocation of the certificates are counted, as on figure 2.12. The very little numbers as *"CACompromise"* don't appear on the graphic but do exist.

These results could be commented. It's a good way to evaluate the

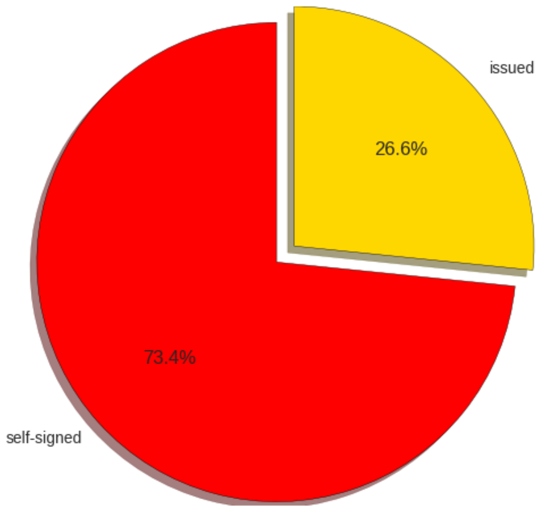


Figure 2.8: Self-signed certificates

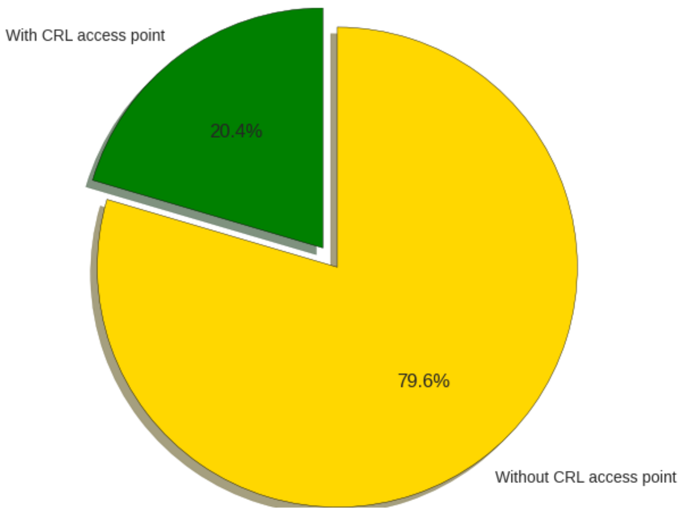


Figure 2.9: Rate of certificates with a CRL access point

quality of a certification authority. Actually, the figure 2.12 shows that a lot of certificates are revoked without any reason. Nothing can force

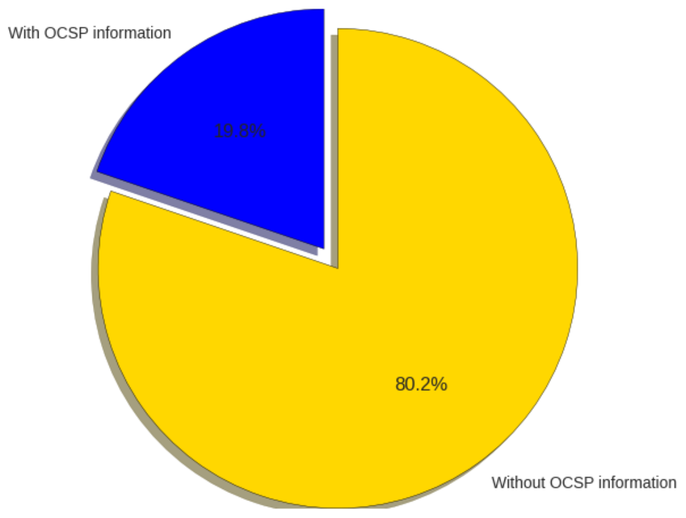


Figure 2.10: Rate of non self-signed certificates with OCSP information

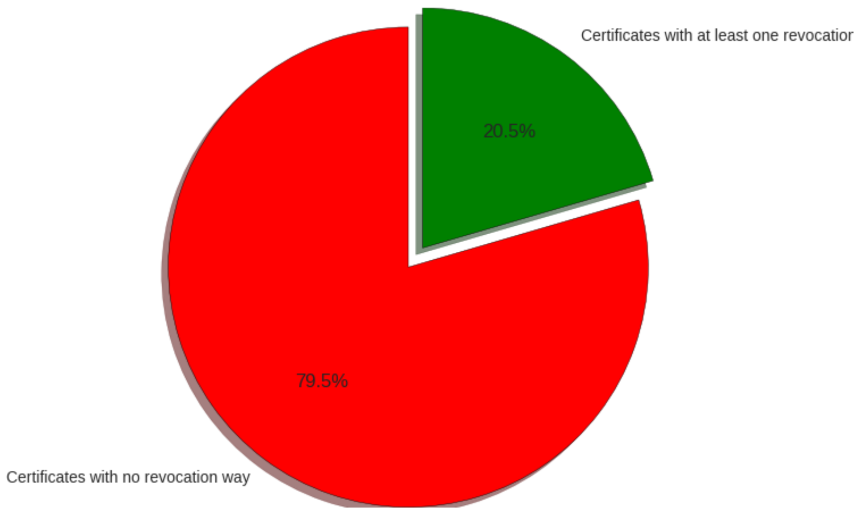


Figure 2.11: Rate of non self-signed certificates without revocation way

CAs to provide the reason of revocation of a certificate. A CRL without any reasons is thus a valid one. But as far as trust is concerned, the right behaviour for a CA is to provide these reasons to its clients. As a consequence, the system stores in the database, for each issuer of CRL, the rate of each reason of revocation. Thus, a CRL issuer which has hundred percents of its reasons to *"None"* could perhaps not be trusted for critical systems.

The figure 2.13 shows these elements for the CRL of the domain cert.fnmt.es, which is an entity of the spanish government. It shows that this CA seems to have a good revocation behaviour, because several reasons are provided, and no certificate has been revoked with a *"None"*.

On the other hand, the figure 2.14 shows a russian CA which has revoked its certificates just with a *"None"* reason.

Finally, the process points out all the CRL issuers which published a CRL with a *"CACompromise"* certificate. It appears that about a hundred of issuers have this habit. During the internship, we have contacted one of these (Lockeed Martin Corporation), in order to ask them why they edited their CRL like that. The company sent an answer, which said *"we are going to study the issue"* but that's all. The concerned CRL is always online at the end of August.

To automate the whole processing, I used finally the cron service. Initially I chose a permanent python script with a *"while True"* statement, but the use of cron is a more secure way to avoid crashes of the processing.

After the presentation of the results of this six-months work, the tools that I used will be presented. The understanding and the training for these tools has been a major part of my work, above all during the three first months of the internship.

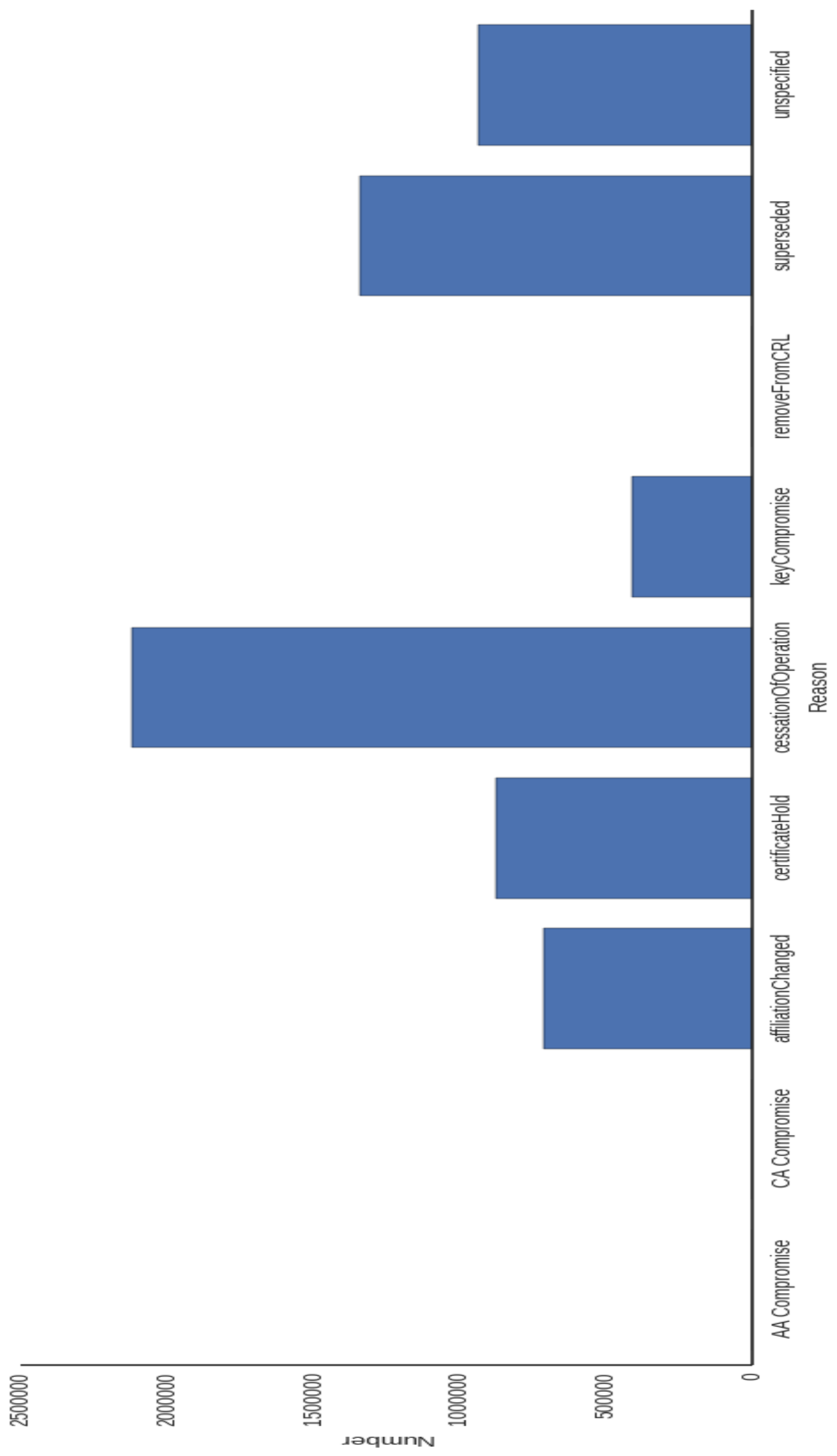
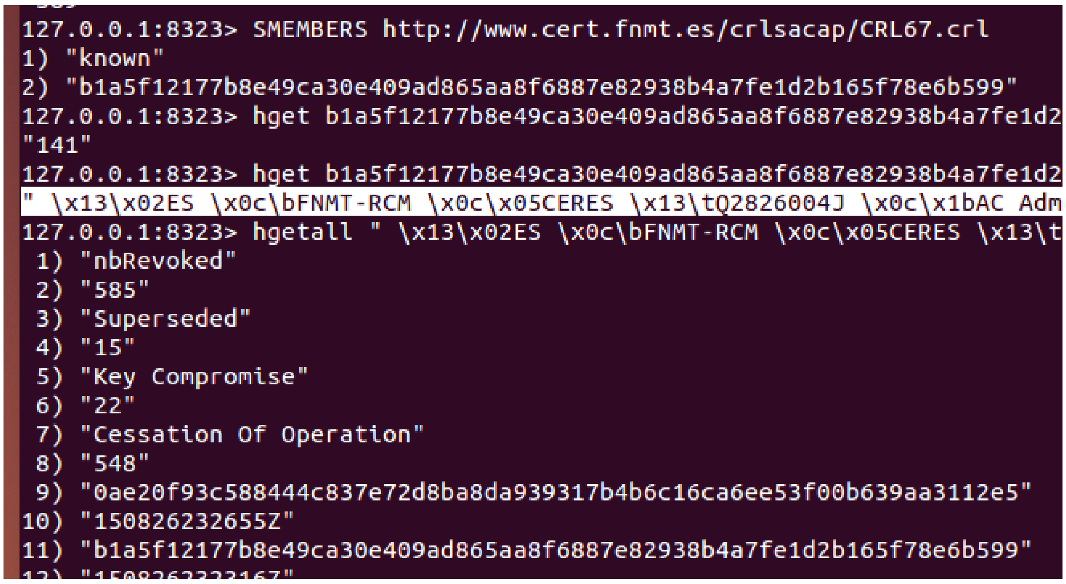
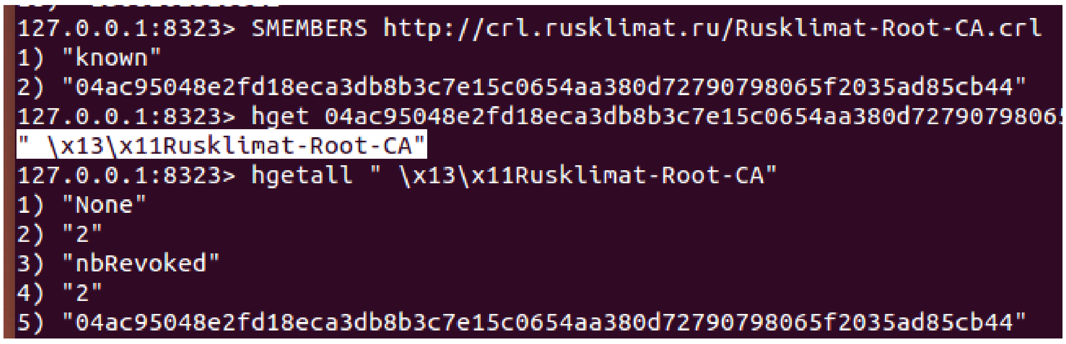


Figure 2.12: Reasons of revocation of the revoked certificates



```
127.0.0.1:8323> SMEMBERS http://www.cert.fnmt.es/crlsacap/CRL67.crl
1) "known"
2) "b1a5f12177b8e49ca30e409ad865aa8f6887e82938b4a7fe1d2b165f78e6b599"
127.0.0.1:8323> hget b1a5f12177b8e49ca30e409ad865aa8f6887e82938b4a7fe1d2
"141"
127.0.0.1:8323> hget b1a5f12177b8e49ca30e409ad865aa8f6887e82938b4a7fe1d2
" \x13\x02ES \x0c\bFNMT-RCM \x0c\x05CERES \x13\tQ2826004J \x0c\x1bAC Adm
127.0.0.1:8323> hgetall " \x13\x02ES \x0c\bFNMT-RCM \x0c\x05CERES \x13\t
 1) "nbRevoked"
 2) "585"
 3) "Superseded"
 4) "15"
 5) "Key Compromise"
 6) "22"
 7) "Cessation Of Operation"
 8) "548"
 9) "0ae20f93c588444c837e72d8ba8da939317b4b6c16ca6ee53f00b639aa3112e5"
10) "150826232655Z"
11) "b1a5f12177b8e49ca30e409ad865aa8f6887e82938b4a7fe1d2b165f78e6b599"
```

Figure 2.13: A good revocation behaviour



```
127.0.0.1:8323> SMEMBERS http://crl.rusklimat.ru/Rusklimat-Root-CA.crl
1) "known"
2) "04ac95048e2fd18eca3db8b3c7e15c0654aa380d72790798065f2035ad85cb44"
127.0.0.1:8323> hget 04ac95048e2fd18eca3db8b3c7e15c0654aa380d7279079806
" \x13\x11Rusklimat-Root-CA"
127.0.0.1:8323> hgetall " \x13\x11Rusklimat-Root-CA"
1) "None"
2) "2"
3) "nbRevoked"
4) "2"
5) "04ac95048e2fd18eca3db8b3c7e15c0654aa380d72790798065f2035ad85cb44"
```

Figure 2.14: Bad revocation behaviour

Chapter 3

Used tools

3.1 Asn1

Abstraction is the best way to manage software development. To insure good interconnection between computers from the physical layer to to the application one, Abstract Syntax Notation One[5] provides a way for specifying systems without concern for how it is implemented or represented. Actually, it is a formal and flexible notation to describe abstract types and values.

With ASN1, a type is a set of values. It could have a finite or infinite number of values. Types can be simple (atomic), structured (which have components), tagged (derived from other types, or other classified (like the CHOICE and ANY types).

Every ASN1 type (other than CHOICE or ANY) has a tag, which is a class and a tag number. It is the real identifier of the type, it is not its name.

Tags classes can be universal (same in all applications), application (specific to an application), private (specific to an enterprise) or context-specific.

Here are some universal types:

- INTEGER
- BOOLEAN
- PrintableString
- BIT STRING

Furthermore, some other types are defined, for instance:

- SEQUENCE

```
Validity ::= SEQUENCE {
     notBefore      Time,
     notAfter       Time }

Time ::= CHOICE {
     utcTime        UTCTime,
     generalTime    GeneralizedTime }
```

Figure 3.1: Validity field of a SSL certificate

- SEQUENCE OF
- choice between several options (CHOICE)

To give a concrete example, the field *"validity"* of a SSL certificate has to give two dates, the notBefore and notAfter dates. The figure 3.1 shows how thes elements are described with the ASN1 syntax.

3.2 Python

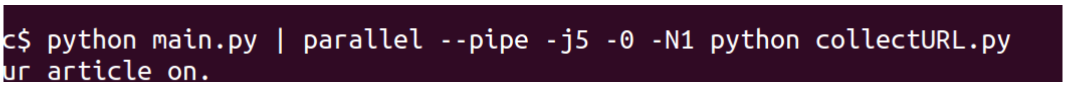
The question of the use or not of the Python language has not been asked on the beginning of this internship. Actually, the existing scripts was written in Python and the whole team of CIRCL seemed to use this tool in the major part of their works, and above all for all matters that concern certificates. Naturally I began with Python and my six months work was Python oriented.

However, with a more global approach of my subject, I should have asked the question: why must I use the Python language to study and process the SSL certificates and monitor the associated CRL?

First, it is naturel with this subject to choose an object-oriented language. Certificates and CRL are very numerous and have the same attributes and common characteristics.

Moreover, the Python language allows an easy manipulation of strings. Then it adapts very well with a redis storage system, that I use for my work. The Python redis package is very well written and allows to access quickly to data, after some tests and training.

Python is known furthermore for its portability, and my scripts could be use on several platforms, since the CIRCL team is going to refactor



```
c$ python main.py | parallel --pipe -j5 -0 -N1 python collectURL.py
ur article on.
```

Figure 3.2: Command line with GNU parallel

and reuse them. Finally, the implementation of openSSL for python seems to be very complete.

With hindsight, it should have been very uncomfortable to code with another language than Python, above all since the CIRCL analysts are all highly-skilled with it, and less with other languages, that they don't use day to day.

3.3 GNU parallel

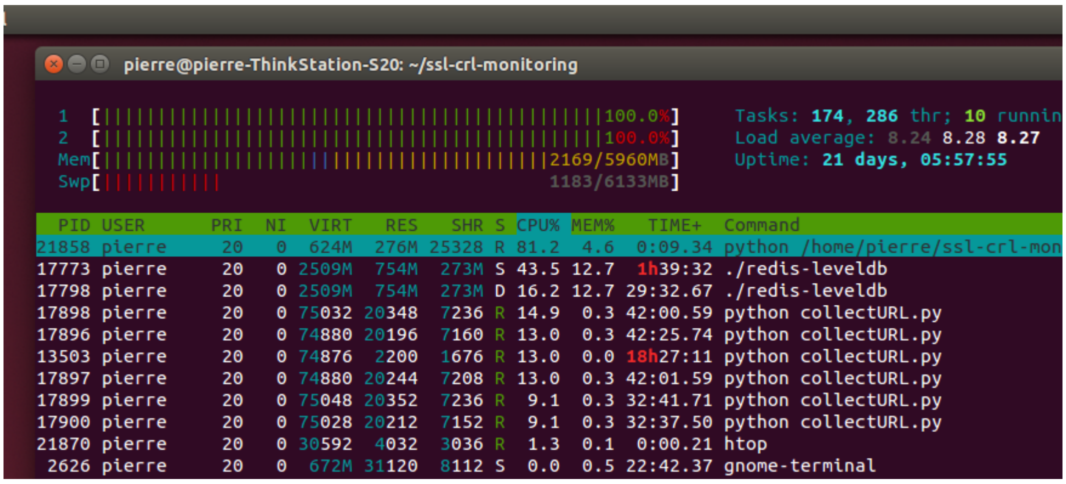
The huge amount of data to process implies that the memory of the computers has to be used at their maximum capacity. Without that, the processing time could largely be more important than necessary. GNU parallel, which is a *"shell tool for executing jobs in parallel using one or more computers. A job can be a single command or a small script that has to be run for each of the lines in the input. The typical input is a list of files, a list of hosts, a list of users, a list of URLs, or a list of tables. A job can also be a command that reads from a pipe. GNU parallel can then split the input and pipe it into commands in parallel."*[1] For instance, the command line for processing the archive is shown on figure 3.2.

Then it allows the computer to be used on its maximum capacity, it could be verified with the htop tool, as shown in figure 3.3.

3.4 Github and git

The collaborative github platform and the tool git are two major elements of work at CIRCL. Indeed, all the developments of the analysts are hosted by this way, and all the communications and advances are based on it. For example, the main library used within this work, M2crypto, is available on github and maintained by Martin Paljak.

[1]https://www.gnu.org/software/parallel/



```
pierre@pierre-ThinkStation-S20: ~/ssl-crl-monitoring

  1  [|||||||||||||||||||||||||||||||||||||||||||||||100.0%]     Tasks: 174, 286 thr; 10 runnin
  2  [|||||||||||||||||||||||||||||||||||||||||||||||100.0%]     Load average: 8.24 8.28 8.27
  Mem[||||||||||||||||||||||||||||||||||||||||||2169/5960MB]     Uptime: 21 days, 05:57:55
  Swp[||||||||||                                1183/6133MB]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
21858 pierre     20   0  624M  276M 25328 R 81.2  4.6  0:09.34 python /home/pierre/ssl-crl-mon
17773 pierre     20   0 2509M  754M  273M S 43.5 12.7  1h39:32 ./redis-leveldb
17798 pierre     20   0 2509M  754M  273M D 16.2 12.7 29:32.67 ./redis-leveldb
17898 pierre     20   0 75032 20348  7236 R 14.9  0.3 42:00.59 python collectURL.py
17896 pierre     20   0 74880 20196  7160 R 13.0  0.3 42:25.74 python collectURL.py
13503 pierre     20   0 74876  2200  1676 R 13.0  0.0 18h27:11 python collectURL.py
17897 pierre     20   0 74880 20244  7208 R 13.0  0.3 42:01.59 python collectURL.py
17899 pierre     20   0 75048 20352  7236 R  9.1  0.3 32:41.71 python collectURL.py
17900 pierre     20   0 75028 20212  7152 R  9.1  0.3 32:37.50 python collectURL.py
21870 pierre     20   0 30592  4032  3036 R  1.3  0.1  0:00.21 htop
 2626 pierre     20   0  672M 31120  8112 S  0.0  0.5 22:42.37 gnome-terminal
```

Figure 3.3: Using the capacity of CPUs at their maximum

The understanding of git is not quite natural, so it has been necessary to apprehend it in the first weeks of the internship. Using git and github allows several developpers to work on a project in a collaborative manner. Each modification is *"committed"* and everybody can involve it or not in his working directory.

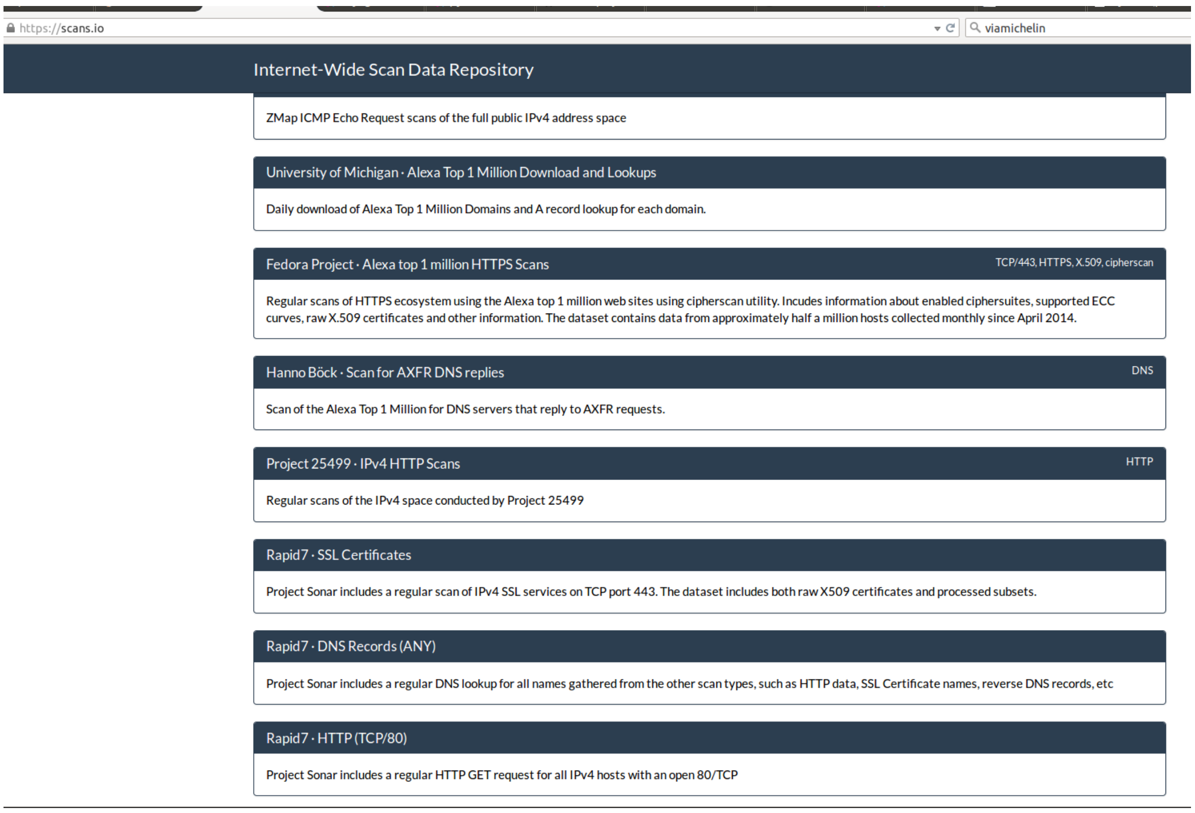
Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

"Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM[2] *tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows."*[2]

3.5 Scans.io

"The Internet-Wide Scan Data Repository is a public archive of research data collected through active scans of the public Internet. The repository is hosted by the ZMap Team at the University of Michigan."[6] All the data processed in this work comes from the directory "sonar.ssl"

[2]Source code management


https://scans.io

viamichelin

Internet-Wide Scan Data Repository

ZMap ICMP Echo Request scans of the full public IPv4 address space

University of Michigan · Alexa Top 1 Million Download and Lookups

Daily download of Alexa Top 1 Million Domains and A record lookup for each domain.

Fedora Project · Alexa top 1 million HTTPS Scans — TCP/443, HTTPS, X.509, cipherscan

Regular scans of HTTPS ecosystem using the Alexa top 1 million web sites using cipherscan utility. Incudes information about enabled ciphersuites, supported ECC curves, raw X.509 certificates and other information. The dataset contains data from approximately half a million hosts collected monthly since April 2014.

Hanno Böck · Scan for AXFR DNS replies — DNS

Scan of the Alexa Top 1 Million for DNS servers that reply to AXFR requests.

Project 25499 · IPv4 HTTP Scans — HTTP

Regular scans of the IPv4 space conducted by Project 25499

Rapid7 · SSL Certificates

Project Sonar includes a regular scan of IPv4 SSL services on TCP port 443. The dataset includes both raw X509 certificates and processed subsets.

Rapid7 · DNS Records (ANY)

Project Sonar includes a regular DNS lookup for all names gathered from the other scan types, such as HTTP data, SSL Certificate names, reverse DNS records, etc

Rapid7 · HTTP (TCP/80)

Project Sonar includes a regular HTTP GET request for all IPv4 hosts with an open 80/TCP

Figure 3.4: Scans.io website

of this website[3]. *"A JSON interface to the repository is also available"*, which allowed me to monitor the data permanently with a JSON parser.

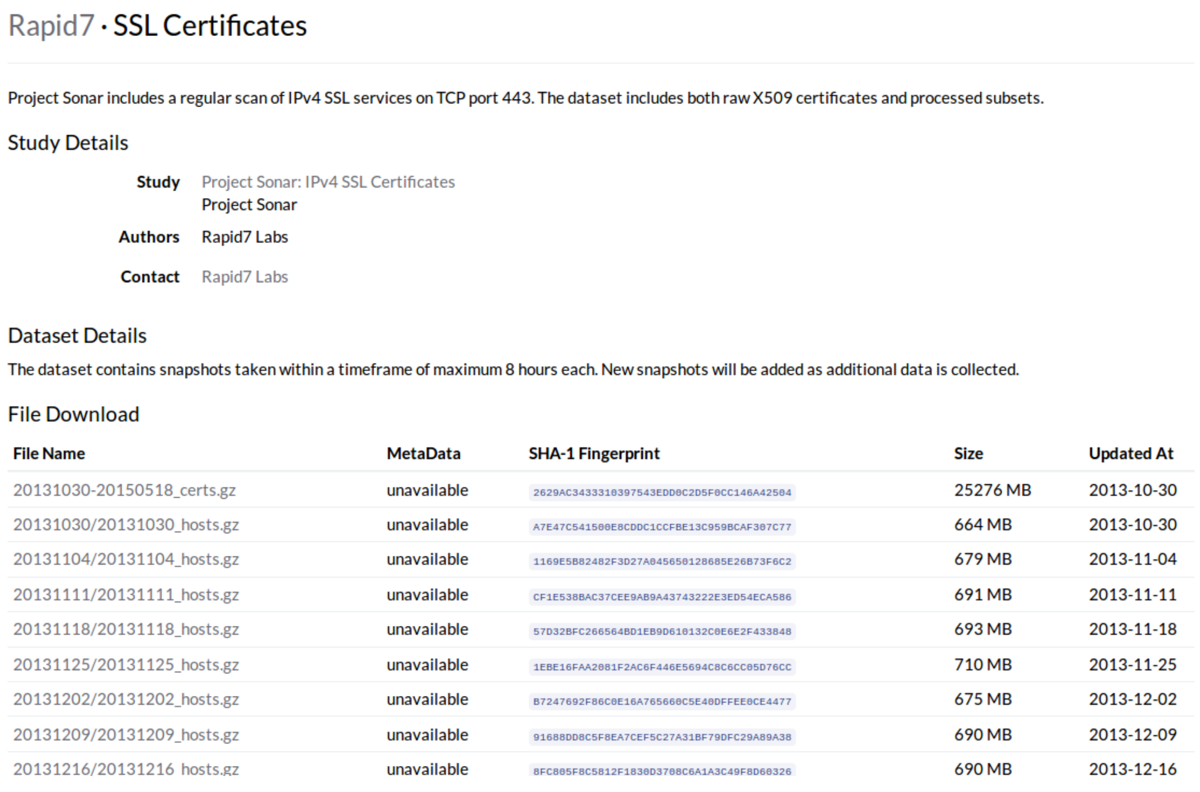
The 3.4 figure presents this website interface. It shows that there are a lot of available data on this website, for instance DNS records, UDP scans, ICMP echo requests, etc...

During this internship, I used archives containing a huge number of SSL certificates. Every two weeks, the maintainer of the website posts a new archive, which provides the last certificates collected on the internet. An initial archive with 25 GB of certificate gathers all the data collected before January 2015. Then, the weekly archive have a smaller size, around 300 MB. The files contain base64 encoded certificates, on the PEM format[4]

The figure 3.5 shows the list of available archives.

Basically, all the archives represents a volume of data of **32 GB of SSL certificates** to process. Obviously, this amount is continuously

[3]https://scans.io/study/sonar.ssl
[4]Privacy Enhancement mail



Rapid7 · SSL Certificates

Project Sonar includes a regular scan of IPv4 SSL services on TCP port 443. The dataset includes both raw X509 certificates and processed subsets.

Study Details

Study	Project Sonar: IPv4 SSL Certificates Project Sonar
Authors	Rapid7 Labs
Contact	Rapid7 Labs

Dataset Details

The dataset contains snapshots taken within a timeframe of maximum 8 hours each. New snapshots will be added as additional data is collected.

File Download

File Name	MetaData	SHA-1 Fingerprint	Size	Updated At
20131030-20150518_certs.gz	unavailable	2629AC3433310397543EDD0C2D5F0CC146A42504	25276 MB	2013-10-30
20131030/20131030_hosts.gz	unavailable	A7E47C541500E8CDDC1CCFBE13C959BCAF307C77	664 MB	2013-10-30
20131104/20131104_hosts.gz	unavailable	1169E5B82482F3D27A045650128685E26B73F6C2	679 MB	2013-11-04
20131111/20131111_hosts.gz	unavailable	CF1E538BAC37CEE9AB9A43743222E3ED54ECA586	691 MB	2013-11-11
20131118/20131118_hosts.gz	unavailable	57D32BFC266564BD1EB9D610132C0E6E2F433848	693 MB	2013-11-18
20131125/20131125_hosts.gz	unavailable	1EBE16FAA2081F2AC6F446E5694C8C6CC05D76CC	710 MB	2013-11-25
20131202/20131202_hosts.gz	unavailable	B7247692F86C0E16A765660C5E40DFFEE0CE4477	675 MB	2013-12-02
20131209/20131209_hosts.gz	unavailable	91688DD8C5F8EA7CEF5C27A31BF79DFC29A89A38	690 MB	2013-12-09
20131216/20131216_hosts.gz	unavailable	8FC805F8C5812F1830D3708C6A1A3C49F8D60326	690 MB	2013-12-16

Figure 3.5: List of SSL certificates archive

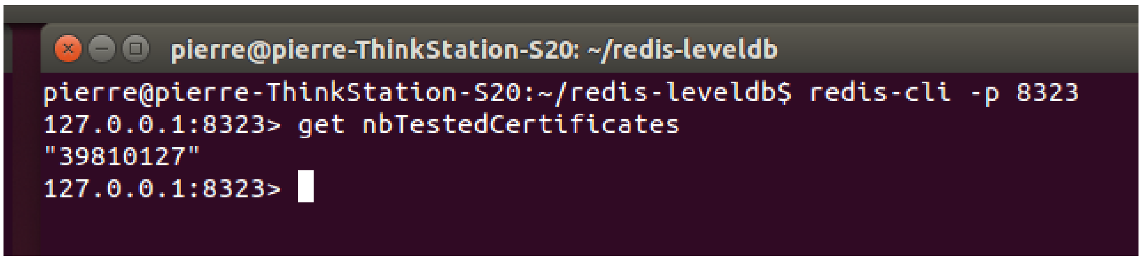
growing.

To monitor the website and its ssl sonar permanently, it has been necessary to process a JSON document, which provides the whole list of available document. The appendix B exposes what is the JSON format, and the way used to parse it.

3.6 Redis

All the data storage in my work has been done with redis, which is *"an open source, BSD*[5] *licensed, advanced key-value cache and store. It is often referred to as a data structure server since keys can contain strings, hashes, lists, sets, sorted sets, bitmaps and hyperloglogs."*[6] Redis is a part of *"NoSQL"* world. It is particularly adapted to my work during this internship regarding its features. Nevertheless, it has limits like all the systems.

[5]Berkeley Software Distribution
[6]http://redis.io/topics/introduction



```
pierre@pierre-ThinkStation-S20: ~/redis-leveldb
pierre@pierre-ThinkStation-S20:~/redis-leveldb$ redis-cli -p 8323
127.0.0.1:8323> get nbTestedCertificates
"39810127"
127.0.0.1:8323>
```

Figure 3.6: Extract a value with GET

NoSQL NoSQL means *"not only SQL"* and is an alternative solution different from classical data base system. It is very important not to translate NoSQL with *"no SQL"* since the two solutions are quite complementary. The choice of one or the other depends on the project you are processing, each system has its pros and its cons.

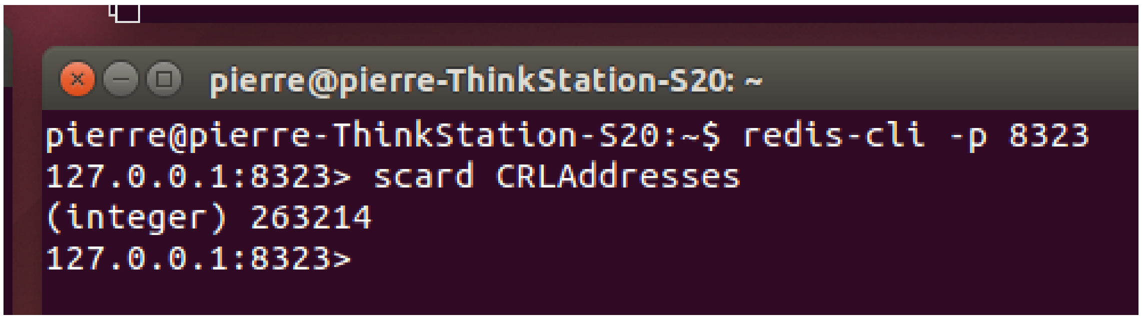
Redis features Data is simply stored with key/value pairs. The main advantage of that with me, is that the request system is supported by the application language. For my work, the Python scripts are the real tools for requesting, that implies a real easy way to implement complex requests.

1. Keys

   A key is just a way to store a value. For instance, I stored the number of tested SSL certificates with the key *"nbTestedCertificates"*. Scripts can modify it with a *"SET"* command, and it can be found with a *"GET"* command. The figure 3.6 shows the way to retrieve the value.

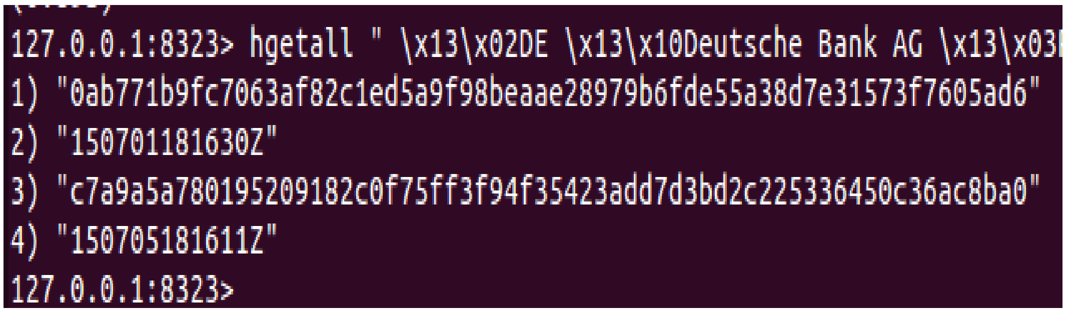
2. Sets

   Sets are ordered collection of strings. To one key is associated a collection of objects. An object could be add to a set even it is already inside the set, it will has no effect. Then the user can test if a value is in a set, know the number of values or extract one ore all values. With my work I used a set to store the different CRL addresses I collected in the SSL certificates. The SMEMBERS command allows to extract all the objects. Then users can easily process loops with Python on the collection. The SCARD command number the elements of the set. The 3.7 figure shows how many CRL addresses have been stored during the processing.



```
pierre@pierre-ThinkStation-S20: ~
pierre@pierre-ThinkStation-S20:~$ redis-cli -p 8323
127.0.0.1:8323> scard CRLAddresses
(integer) 263214
127.0.0.1:8323>
```

Figure 3.7: Number of CRL collected addresses



```
127.0.0.1:8323> hgetall " \x13\x02DE \x13\x10Deutsche Bank AG \x13\x03
1) "0ab771b9fc7063af82c1ed5a9f98beaae28979b6fde55a38d7e31573f7605ad6"
2) "150701181630Z"
3) "c7a9a5a780195209182c0f75ff3f94f35423add7d3bd2c225336450c36ac8ba0"
4) "150705181611Z"
127.0.0.1:8323>
```

Figure 3.8: Display of a hash

3. Hashes

   Hashes are more elaborated structures. They associate a key to one or several fields, where each field has a value. I use this structure for instance to store the CRL. The figure 3.8 shows that a hash could include a lot of information. Here are displayed for a german bank the sha1 codes of the CRL published and their publication dates.

Redis is offering more other structures as lists or sorted sets. I did not use them during my work, so they will not be presented here.

Limits, partial migration to Redis-level-db The huge number of certificates and revocation lists has quickly saturated the memory of my computer. Thus, to go on with my scripts, a migration towards the redis-level-db has been necessary. This system, which uses the same syntax than redis, has a capacity of storage more important since data is stored on the harddisk. Nevertheless, the rapidity is lower as far as data access is concerned. Actually, there is a compromise to find. Data which should be accessed quickly could be stored on a redis system, in memory, while more voluminous data could be stored on a redis-level-db system. **The two systems can cohabitate on the same computer**, and it allows to use common functions since the employed syntax is the same.

3.7 LaTeX

Documents Writing With LATEX

For the edition of my internship report, I was suggested to use LaTeX, which is a *"high-quality typesetting system; it includes features designed for the production of technical and scientific documentation. LaTeXis the de facto standard for the communication and publication of scientific documents. LaTeX is available as free software. "*[1].

Even if this tool is not really a part of the research work of my internship, I have to mention it. Actually, the discovery of this system was a great pleasure, above all at the end of the six-months period. Moreover, I was very sad not to have learn it before. Maybe it should be teached somewhere in a academic period. The only problem with Latex is to do the first step: to accept to give up with WYSIWYG[7].

[7]What you see is what you get

Conclusion

Obviously, this work has to be pursued. On the first hand, the CIRCL team is going to review my scripts and build an adapted tool. On the second hand, I am going to try to find time, in order to maintain it and to make it evoluate, on my personnal github account[8]. I don't know if my future posting will be compatible with this will. Unfortunatly, nothing is less sure.

Actually the subject of this internship was quite interesting, but I have been more involved in the discovery of new tools. The main part of the tools exposed in the third chapter are highly useful, I didn't know its before these months with CIRCL. I'm sure that I'm going to work differently with that now, and I will try to convert the maximum of people in my future posting. With hindsight, I'm sure now that my supervisor and its analysts was quite more involved in converting me into using their free favorite tools than in orienting me for the subject of the study. It often seems to be like a crusade for them. Maybe it was the right aim to pursue, since I will reuse these tools for many subjects in the future.

[8]https://github.com/cailletpierre

Appendix A

Base 64 encoding

All the information about base 64 encoding is available in the RFC[1] 3548[7]

Principles

The principle of Base64 encoding is to involve US-ASCII characters (no accented characters) to encode any type of data coded on 8 bits. The email protocols were indeed originally planned to carry text messages only. However, given the variety of email systems, the exchange of binary data provokes changes of the content, making the original document unreadable. Base64 format, massively used in email exchanges, can transmit any binary document (application, video, audio, etc.) in the attachment of an email by encrypting using classic character. This encoding causes an increase of one third in the volume of data to be encoded.

Base64 encoding uses 4 printable characters (in US-ASCII format) for encoding a group of any 3 bytes (3 * 8 bits = 24 bits). It uses an alphabet of 64 printable characters to represent a 6-bit word. The 64 symbols of this alphabet are chosen in order to be universally readable, and for not having a meaning in the main email protocols (in particular SMTP). Walking through the binary data from left to right, 24-bit groups are created by concatenating blocks of three 8-bit data. Each 24-bit group is then divided into 4 groups of 6 bits, corresponding to 4 characters of the Base64 alphabet.

The Base64 encoding data is provided for forming a multiple of 24 bits. Thus, if the volume of data to be encoded are not a multiple of 24 bits, the result of the Base64 encoding has then to be completed by 0-3

[1]request for comments

character "=" to obtain a multiple of 24 bits. This 65th character may be present at the end of encoded data.

Moreover, to ensure compatibility with all email systems, Base64 data is formatted with line breaks so that each line does not exceed 76 characters.

To give an example

The binary ascii translation of "UL" is:

01010101 01001100

The base64 alphabet is:

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz 0123456789+/

I divide the message into 6-bits groups, with some 0 added if necessary:

010101 010100 1100(+00)

It is three base64 characters, which rank is given with a base-10 translation:

21 20 48

The base64 encoding of UL is then:

VUw=

The increase of one third of data volume is visible.

Appendix B

The JSON format[4]

JSON (JavaScript Object Notation - Object Notation outcome of JavaScript) is a lightweight data interchange format. It is easy to read and write for humans. It is easily analyzable or generable by machines. It is based on a subset of the JavaScript programming language (JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999). JSON is a completely independent text of any language format, but the conventions it uses will be familiar to any regular programmer to descendant languages C, for example: C itself, C ++, C #, Java, JavaScript, Perl, Python and many others. These properties make JSON an ideal data exchange language.

JSON is based on two structures:

1. A collection of name / value pairs. Depending of the language, it becomes an object, a recording, a structure, a dictionary, a hash table, a typed list or associative array.

2. A list of ordered values. With a language, it becomes an array, vector, list, or a suite.

These data structures are universal. Actually all the modern programming languages provide them in one form or another. It is reasonable that an interchangeable data format with programming languages is also based on these structures.

In JSON, they take the following forms:

- One object, which is a set of unordered pairs name / value. An object begins with a left brace and ends with a right brace. Each name is followed by : (a colon) and the name / value pairs are separated by , (a comma).

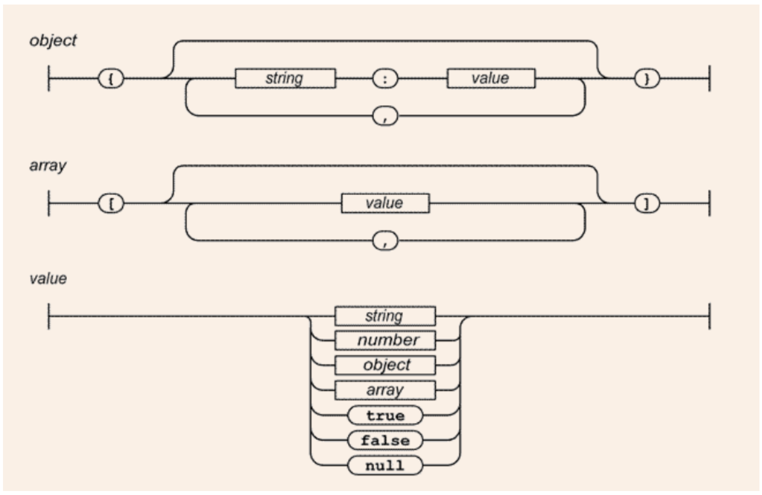


Figure B.1: Scheme of the JSON language

- An array is a collection of ordered values. An array begins with a left bracket and ends with a right bracket. Values are separated by , (a comma).
- A value can be either a string in quotes, or a number, or true or false or null, or an object or an array. These structures can be nested.
- A string is a sequence of zero or more Unicode characters, quotation marks, and using the exhaust with backslash. A caratère is represented by a string of one character.

Apart from a few encoding details, the figure B.1 describes the language in its entirety.

A string is very close to its equivalent in C or Java. A number is very close to those that can be found in C or Java, except that the octal and hexadecimal formats are not used. The figure B.2 shows these two formats precisely.

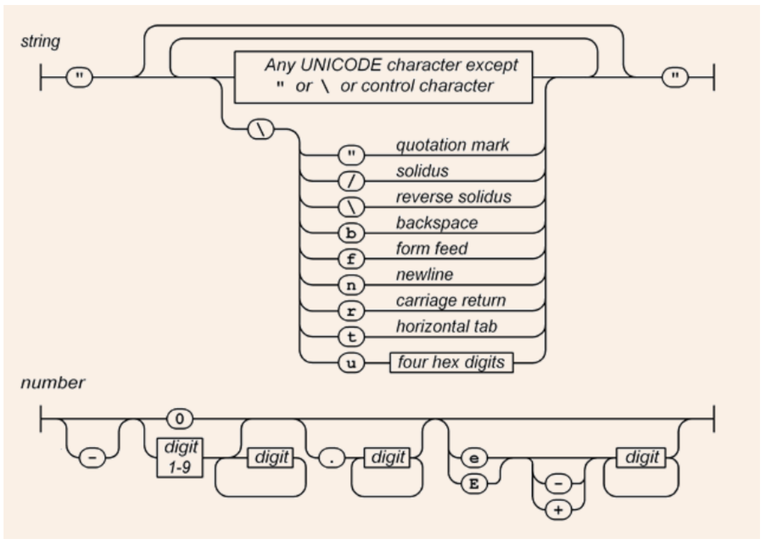


Figure B.2: JSON: String and number

Appendix C

RFC 5280: Table of contents and important paragraphs

Abstract This memo profiles the X.509 v3 certificate and X.509 v2 certificate revocation list (CRL) for use in the Internet. An overview of this approach and model is provided as an introduction. The X.509 v3 certificate format is described in detail, with additional information regarding the format and semantics of Internet name forms. Standard certificate extensions are described and two Internet-specific extensions are defined. A set of required certificate extensions is specified. The X.509 v2 CRL format is described in detail along with standard and Internet-specific extensions. An algorithm for X.509 certification path validation is described. An ASN.1 module and examples are provided in the appendices.

1. Introduction

2. Requirements and Assumptions
    - 2.1. Communication and Topology
    - 2.2. Acceptability Criteria
    - 2.3. User Expectations
    - 2.4. Administrator Expectations

3. Overview of Approach

Following is a simplified view of the architectural model assumed by the Public-Key Infrastructure using X.509 (PKIX) specifications.
The components in this model are:

- end entity: user of PKI certificates and/or end user system that is the subject of a certificate;

- CA: certification authority;
- RA: registration authority, i.e., an optional system to which a CA delegates certain management functions;
- CRL issuer: a system that generates and signs CRLs; and
- repository: a system or collection of distributed systems that stores certificates and CRLs and serves as a means of distributing these certificates and CRLs to end entities.

CAs are responsible for indicating the revocation status of the certificates that they issue. Revocation status information may be provided using the Online Certificate Status Protocol (OCSP) [RFC2560], certificate revocation lists (CRLs), or some other mechanism. In general, when revocation status information is provided using CRLs, the CA is also the CRL issuer. However, a CA may delegate the responsibility for issuing CRLs to a different entity.

Note that an Attribute Authority (AA) might also choose to delegate the publication of CRLs to a CRL issuer.

3.1. X.509 Version 3 Certificate9 Users of a public key require

confidence that the associated private key is owned by the correct remote subject (person or system) with which an encryption or digital signature mechanism will be used. This confidence is obtained through the use of public key certificates, which are data structures that bind public key values to subjects. The binding is asserted by having a trusted CA digitally sign each certificate. The CA may base this assertion upon technical means (a.k.a., proof of possession through a challenge-response protocol), presentation of the private key, or on an assertion by the subject. A certificate has a limited valid lifetime, which is indicated in its signed contents. Because a certificate's signature and timeliness can be independently checked by a certificate-using client, certificates can be distributed via untrusted communications and server systems, and can be cached in unsecured storage in certificate-using systems.

ITU-T X.509 (formerly CCITT X.509) or ISO/IEC 9594-8, which was first published in 1988 as part of the X.500 directory recommendations, defines a standard certificate format [X.509]. The certificate format in the 1988 standard is called the version 1 (v1) format. When X.500 was revised in 1993, two more fields were added, resulting in the version 2 (v2) format.

The Internet Privacy Enhanced Mail (PEM) RFCs, published in 1993, include specifications for a public key infrastructure based on X.509 v1 certificates [RFC1422]. The experience gained in attempts to deploy RFC 1422 made it clear that the v1 and v2 certificate formats were deficient in several respects. Most importantly, more fields were needed to carry information that PEM design and implementation experience had proven necessary. In response to these new requirements, the ISO/IEC, ITU-T, and ANSI X9 developed the X.509 version 3 (v3) certificate format. The v3 format extends the v2 format by adding provision for additional extension fields.

Particular extension field types may be specified in standards or may be defined and registered by any organization or community. In June 1996, standardization of the basic v3 format was completed [X.509].

ISO/IEC, ITU-T, and ANSI X9 have also developed standard extensions for use in the v3 extensions field [X.509][X9.55]. These extensions can convey such data as additional subject identification information, key attribute information, policy information, and certification path constraints.

However, the ISO/IEC, ITU-T, and ANSI X9 standard extensions are very broad in their applicability. In order to develop interoperable implementations of X.509 v3 systems for Internet use, it is necessary to specify a profile for use of the X.509 v3 extensions tailored for the Internet. It is one goal of this document to specify a profile for Internet WWW, electronic mail, and IPsec applications. Environments with additional requirements may build on this profile or may replace it.

3.2. Certification Paths and Trust

3.3. Revocation

When a certificate is issued, it is expected to be in use for its entire validity period. However, various circumstances may cause a certificate to become invalid prior to the expiration of the validity period. Such circumstances include change of name, change of association between subject and CA (e.g., an employee terminates employment with an organization), and compromise or suspected compromise of the corresponding private key. Under such circumstances, the CA needs to revoke the certificate.

X.509 defines one method of certificate revocation. This method involves each CA periodically issuing a signed data structure called a certificate revocation list (CRL). A CRL is a time-stamped list identifying revoked certificates that is signed by a CA or CRL issuer and made freely available in a public repository. Each revoked certificate is identified in a CRL by its certificate serial number. When a certificate-using system uses a certificate (e.g., for verifying a remote user's digital signature), that system not only checks the certificate signature and validity but also acquires a suitably recent CRL and checks that the certificate serial number is not on that CRL. The meaning of "suitably recent" may vary with local policy, but it usually means the most recently issued CRL. A new CRL is issued on a regular periodic basis (e.g., hourly, daily, or weekly). An entry is added to the CRL as part of the next update following notification of revocation. An entry MUST NOT be removed from the CRL until it appears on one regularly scheduled CRL issued beyond the revoked certificate's validity period.

An advantage of this revocation method is that CRLs may be distributed by exactly the same means as certificates themselves, namely, via untrusted servers and untrusted communications.

One limitation of the CRL revocation method, using untrusted communications and servers, is that the time granularity of revocation is limited to the CRL issue period. For example, if a revocation is reported now, that revocation will not be reliably notified to certificate-using systems until all currently issued CRLs are scheduled to be updated – this may be up to one hour, one day, or one week depending on the frequency that CRLs are issued.

As with the X.509 v3 certificate format, in order to facilitate interoperable implementations from multiple vendors, the X.509 v2 CRL format needs to be profiled for Internet use. It is one goal of this document to specify that profile. However, this profile does not require the issuance of CRLs. Message formats and protocols supporting on-line revocation notification are defined in other PKIX specifications. On-line methods of revocation notification may be applicable in some environments as an alternative to the X.509 CRL. On-line revocation checking may significantly reduce the latency between a revocation report and the distribution of the information to relying parties. Once the CA accepts a revocation report as authentic and valid, any query to the on-line service will correctly reflect the certificate validation impacts of the revocation. However, these methods impose new security requirements: the certificate validator needs to trust the on-line validation service while the repository does not need to be trusted.

3.4. Operational Protocols

3.5. Management Protocols

4. Certificate and Certificate Extensions Profile

4.1. Basic Certificate Fields
- 4.1.1. Certificate Fields
- 4.1.2. TBSCertificate

4.2. Certificate Extensions
- 4.2.1. Standard Extensions
- 4.2.2. Private Internet Extensions

5. CRL and CRL Extensions Profile

5.1. CRL Fields

The X.509 v2 CRL syntax is as follows. For signature calculation, the data that is to be signed is ASN.1 DER encoded. ASN.1 DER encoding is a tag, length, value encoding system for each element.

- CertificateList ::= SEQUENCE

  tbsCertList TBSCertList,

  signatureAlgorithm AlgorithmIdentifier,

  signatureValue BIT STRING

- TBSCertList ::= SEQUENCE

  version Version OPTIONAL,

  signature AlgorithmIdentifier,

  issuer Name,

  thisUpdate Time,

  nextUpdate Time OPTIONAL,

  revokedCertificates SEQUENCE OF SEQUENCE

  userCertificate CertificateSerialNumber,

  revocationDate Time,

  crlEntryExtensions Extensions OPTIONAL

  crlExtensions [0] EXPLICIT Extensions OPTIONAL

5.1.1. CertificateList Fields

The CertificateList is a SEQUENCE of three required fields. The fields are described in detail in the following subsections.

5.1.2. Certificate List "To Be Signed"

List of Figures

Bibliography

[1] *LaTeX – A document preparation system.* April 2015. http://latex-project.org/.

[2] Software Freedom Conservancy. *The git SCM.* http://www.git-scm.com/.

[3] Network working group Internet engineer task force. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile.* http://www.ietf.org/rfc/rfc5280.txt.

[4] RFC7159 Internet engineer task force. *The JavaScript Object Notation (JSON) Data Interchange Format.* https://tools.ietf.org/html/rfc7159.

[5] Burton S. Kaliski Jr. *A Layman's Guide to a Subset of ASN.1, BER, and DER.* November 1993. http://luca.ntop.org/Teaching/Appunti/asn1.html.

[6] University of Michigan. *Internet-Wide Scan Data Repository.* 2015. https://scans.io.

[7] The Internet Society. *The Base16, Base32, and Base64 Data Encodings.* July 2003. http://tools.ietf.org/html/rfc3548.