UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

Privacy Aware Sharing of IOCs in MISP

Dissertation presented by
Charles JACQUET

for obtaining the Master's degree in
Computer Science and Engineering

Supervisor(s)
Ramin SADRE

Reader(s)
Antoine CAILLIAU, Alexandre DULAUNOY, William ROBINET , François-Xavier STANDAERT

Academic year 2016-2017

Abstract

Malware are plaguing this computer age. An actual solution to protect computer systems against them is threat information sharing: Organisations share lists of Indicators Of Compromise with each other.
For that, an interesting open source platform called MISP allows companies to share these IOCs but also malware analyses and attack correlations in an online fashion.
On the other hand, offline lookup would also be an important feature but is stopped by the need of data privacy and confidentiality. This master thesis looks for sharing the dataset of IOCs while still protecting privacy and confidentiality by not directly disclosing the information.
After a state of the art, a possible found solution had been implemented with some improvements before finally being analysed.

Contents

Change that title

Chapter 1

Introduction

1.1 Interaction with this Latex Document (To be removed)

In order to facilitate comments, I've added some new commands explained by[1]:

- unsure => What I need to check
- change => What have to be modified
- info => Add simple comment
- improvement => Indicate a possible improvement

1.2 Introduction

In the last report of the Ponemon institute, the estimated cost of annual data breaches for 384 companies in 12 countries[2] is about $4 million. 48% of theses breaches are due to malicious and criminal attacks and gives a damage cost per capita around $170 only for malicious and criminal attacks.
An other example is Cybersecurity Ventures that predicts the global annual cybercrime costs will grow from $3 trillion in 2015 to $6 trillion by 2021.

Our computer defenses need to be improved and being aware of new threats faster than malware detectors or antiviruses. This results in a new kind of security level called threat sharing. As soon as an organization discovers a threat, they can prevent others to be compromised just by informing them about it.
Unfortunately, this is not enough as, theses information traces, here and after called IOCs, are often considered to be confidential. Revealing them, can reveal damaging information for the companies.
Thus a technique is needed for **sharing** information without **giving** it directly to the other parties. This master thesis works on finding a "more secure" way of sharing information. More precisely, if we attempt to create an encrypted database of these IOCs, the idea would be to slow down a brute force attack for someone who has complete access to the database while still making this database useful for the targeted user.

The first chapter is about getting a rough idea on what this work is about by defining the used terms as well as a state of the art.

[1] http://tex.stackexchange.com/questions/9796/how-to-add-todo-notes

[2] United States, United Kingdom, Germany, Australia, France, Brazil, Japan, Italy, India, the Arabian region (United Arab Emirates and Saudi Arabia), Canada and, South Africa

Then the second chapter is on different ideas I've got, their strengths but also their weaknesses. This will softly bring the next chapter about the implementation. The last important chapter is about the results. An analyze of the performance mixed with discussions on the work.
Finally, I will conclude on my work with some possible improvement and future works.

1.3 Organization

Before going down in the subject, I also think that it is interesting to point out some aspects of the way I've worked on this project.
Firstly, last year, I was searching for a security subject but unfortunately, I wasn't really excited about the proposed ones. But in the other side, I had no idea either on what I could work. I had just the idea to specialize myself into cyber security. Thus, I've looked for ideas where there are plenty of them, directly in security companies.
I've found a company called Conostix which provides security and system services in Luxembourg. They weren't short of ideas, and even if I have to admit that I didn't understood everything, it seemed really interesting. Later on, William Robinet (from conostix) contacted me again to say that they have a better idea and it was to work on MISP. I've thus met Alexandre Dulaunoy from CIRCL and they gave me the starting point of the project which was malware sharing in MISP.

As I had zero knowledge in this domain, I've chosen to make an internship of four weeks in the company (no credited) where I've learned the basics of MISP and I've also worked on network log parsing. But what I mostly learned was that four weeks for trying to specialize myself in a completely new domain was only enough to give me a rough idea of what it is.
During these four weeks, I mostly worked on logs. I've created nothing new, actually it was quite the opposite but it was still interesting to understand how things were working and why they could be useful.
In short, during this internship, I've started to understand how squid3 was working. Then I've installed it on my computer in order to get logs to work on.
Once I've my logs, I would like to get information from them, thus, I've tried to manually parse the logs (which was a mistake). For that purpose, I've started to implement a parser that was using regex to transform each log lines into a set of objects. But even if it was funny to implement, I spent a lot of time on it and each time I finished, I found it not good enough, not modular enough, losing locality and ordering informations, and so on. I have thus realized that I was loosing my time as, of course they are plenty of tools for this kind of job like Logstash.
Then I've started to work with MISP to see how it was working. I've reimplemented a similar hash store like the misp-workbench project.
I've then used it with my log system by parsing the data from MISP the same way I was parsing the logs before adding them to the hash database on Redis.
(Redis was also a great tool I've learned about there.)
I've then created a system to feed the parsed log in my system and briefly analyze the logs.
Beside that, I also had the opportunity to benefit from a one day MISP training organized by CIRCL. I have then continued to work on my master's thesis the whole year long while keeping in touch with Conostix and CIRCL.

1.4 Indicator of Compromise (IOC)

While there is a crime, detectives come and look for clues. They do that because they are wondering questions like what make it happened, why did it happened and how did it happened. The what and why questions can be interesting in order to avoid this crime to happen again while the last one but not the least one, the how can be used to compare this crime with other

similar ones.
This idea of being able to compare them is really interesting as seeing the clues of only one place can be not enough while there could be enough of them if we succeed in gathering the whole set of clues in all related crime.
This is exactly the same idea here, even if the attacker try to minimize its traces, there are always some. It can be email addresses, Internet Protocol(IP) addresses, malware, URL and so on.
The idea, is then to use these traces to trigger alarms on other computer system as soon as they are seen. This is the reason why we call them Indicator Of Compromise and we can quickly understand why theses intrusion clues are so important as attackers use the same techniques to compromise similar companies.

1.5 MISP and Threat Sharing

I've introduced the idea of malware sharing to defend our computer systems against digital threats and attacks. But MISP is not the only platform to do so, current ones are DShield, Critical Stack's Intel, Microsoft's Interflow, AlienVault's Open Threat Exchange, ThreatExchange (Facebook), ...
But here I focused on the MISP project which is, as said on their website, an open source software solution for collecting, storing, distributing and sharing cyber security indicators and threat about cyber security incidents analysis and malware analysis. MISP is designed by and for incident analysts, security and IT professionals or malware reverser to support their day-to-day operations to share structured informations efficiently.
MISP also rely on an hub and spoke sharing model (TAXII [1]). There is a central point called the hub which is the server where all data are kept and the spokes are the organizations that interact with the hub.
A recent paper on MISP [32] had been published this year and could be interesting to get a better understanding of the underlying system. Additionally, there is also a recent ietf draft on MISP core format[4].
I want also to point out that MISP is spreading, and I've the feeling that communities will continue to grow. I've discussed in Belgium with companies like banks, insurances, consultancy companies that are using it. I've even met people knowing it when I was in Asia for the black hat conference (2017). All this is, at least to my point of view, a good sign on the utilization of the system.

1.5.1 History

Christophe Vandeplas started the project as he was tired of the way IOCs were shared (email, pdf, ...). Thus, he created a cakePHP website as a proof of concept and managed to convince the Belgian Defense (where he was working) that the project was worth to work on. They even allowed him to work on it during his work-hours. The project continued to move forward and now, Andras Iklody is the lead developer of the MISP project and works for CIRCL.

1.5.2 Basics of MISP

The basic idea of MISP was thus to create an IOC database. If there is an attack and we have two IOCs, "IOC@malware.mail" and "192.168.16.2". We are interested to keep the information that there has been an attack that can be recognized thanks to these two IOCs. That is why they have created events (which is much more general than the term attack that I've been using) and they use these IOCs as the attributes of this specific event.
Then thanks to this idea, we can analyze the event and even make correlations with other events if they possess similar IOCs.

For clarity concerns, attributes are divided into 13 categories where they are again divided into types. As MISP is no more only targeting malwares, it is interesting to notice that we can also see categories like Financial Fraud. More standard ones are Network Activities, Antivirus detection, Artifacts dropped, and so on.
All the category and types can be found on this web pages `https://www.circl.lu/doc/misp/categories-and-types/index.html`

One other really interesting feature is sightings. When an IOC is referenced, we know that it has been seen by someone, but it does not confirm that the IOC is really related to that particular event and is not an error. Moreover, what says that if an IP address is an IOC today that it is going to be the case in 3 months ?
Sightings is thus the solution to this problem as we can monitor an IOC, knowing if it has been seen at other places, and if they are still relevant.

There are also two important things on the sharing strategy that I want to point out: MISP instances and communities. The first one is easy to understand, MISP is an open source project which means that everybody can decide to run a completely isolated MISP instance for its own needs like a company for storing its own confidential data. But that does not mean that the instance **must be** isolated, they have implemented a way to share information between these instances. While the second one is a good property of an instance. When we are connected to MISP as a user, we are connected thanks to the organization that we belongs to. Then, the organization can choose to share or not to share their events/attributes with other communities or even with other instances.

Now, as a good open source project, a whole ecosystem has been created around and more and more companies are using it.
MISP becomes a really good IOC database with automated correlation system. We can even create correlation graphs to see how different events are connected together what helps a lot for the threat analyses.
Beside all that, a lot of different tools like a python client library are available on their repository: `https://github.com/MISP/`.
They have also added a lot of plugins to the web api like the ability of exporting the attributes in a lot of different formats, standards one and some for specific IDS.

1.5.3 My Settings

MISP is really easy to use, but if we want to go deeper in it, there are a lot of things to understand. That's why they are giving trainings and make available a MISP training virtual machine in order to train ourselves and test new developments.
At the beginning, I've started to work directly with the private instance of MISP hosted by CIRCL but, every time that I was working, I needed to have a really good web connection to connect myself and to download all the data. That's why, I have finished by downloading the virtual machine. But, as I was going to work on my computer and I didn't want to be stuck programming inside the virtual machine, I've done some modifications to make available everything to my local network.
The mysql database wasn't accessible from the outside of the virtual machine (which is completely understandable as they don't want anyone to have a direct access to the complete database) and I've also got some network problems.
The first step was that my virtual machine running on virtualbox was using the virtual vboxnet0 interface and was using, as its IPv4 address, 192.168.56.50. The problem was that, normally it works without problems but my computer does not know about the subnetwork. I thus had to add an ip in the right range by using the command : **ip add add 192.169.56.2/24 dev vboxnet0** .

I'm also sometimes using a second virtual machine to work with and on this virtual machine, I've configured two network interfaces, the first one was to have an internet connection while the second one was the vboxnet0 interface.
For simplifying the use of the vbox interface, I've configured my network to automatically add an ip in the right range (/etc/network/interfaces). This is the addition on the basic configuration:

```
auto eth1
iface eth1 inet static
address 192.168.56.1
netmask 255.255.255.0
```

After that, I was able to contact the virtual machine via the network but it wasn't enough to have a mysql access. For that, as the MISP virtual machine was only accessible from the vboxnet0 interface, it was a problem to create external access for all ip addresses with only a small password protection.
The first step for that was to modify the configuration file (/etc/mysql/my.conf) where I've juste commented the "bind-address 127.0.0.1" line. The next step was to create a user with the rights from the outside. For that, I've connected myself to the database as the misp user and I've added the user:

- mysql -uroot -pPassword1234
- CREATE USER 'username'@'%';
- GRANT ALL ON *.* TO 'username'@'%';

Once done, I could continue to program more easily and create some tests as I could control the whole available set of data. For testing when the code was modified, I had used a really small set of data.

1.5.4 MISP in a few Pictures

There is nothing better to explain all that than real screen shot of the web application. But, it raised one really important question: what can and cannot be shared from MISP and with who? For example, Am I allowed to show screen shot of data from MISP without any risk ?
To respond to that question, I need to introduce the Traffic Light Protocol (TLP) that was created in order to facilitate information sharing by defining the authorized level of disclosure. And thus, it can be used to know what can be shared with a specific audience.
This protocol is defined by FIRST in [13] but also explained in this NIST cyber threat information sharing guide [17].

Knowing that, I know that I can share the data from the virtual machine that I have explained in the previous section as the default OSINT feed is TLP:WHITE. Which means, according to FIRST, that the disclosure is not limited:
Sources may use TLP:WHITE when information carries minimal or no foreseeable risk of misuse, in accordance with applicable rules and procedures for public release. Subject to standard copyright rules, TLP:WHITE information may be distributed without restriction.

The first pages (after the connection) of the web application is a list of all latest events on figure 1.1 (notice the "tlp:while" appearing in the Tags):

Then, by clicking on an event, we can get information on it 1.2:

As well as the attribute list 1.3:

And the last thing that I want to show is one of the way of showing the correlations with other events and is called the correlation graph 1.4:

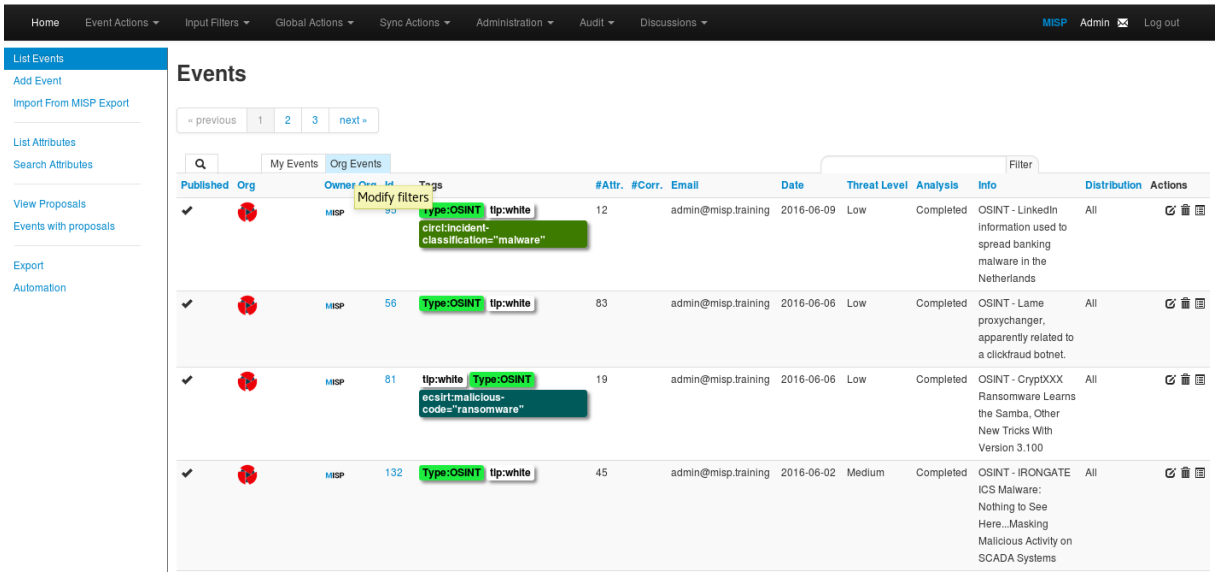


Figure 1.1: MISP : List of events

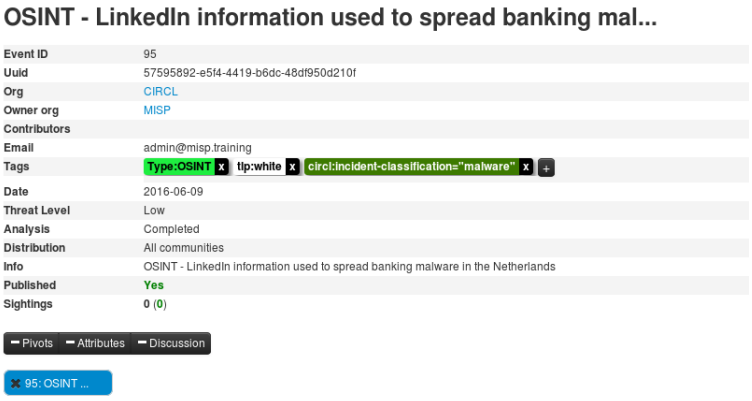


Figure 1.2: MISP : Specific information on the event

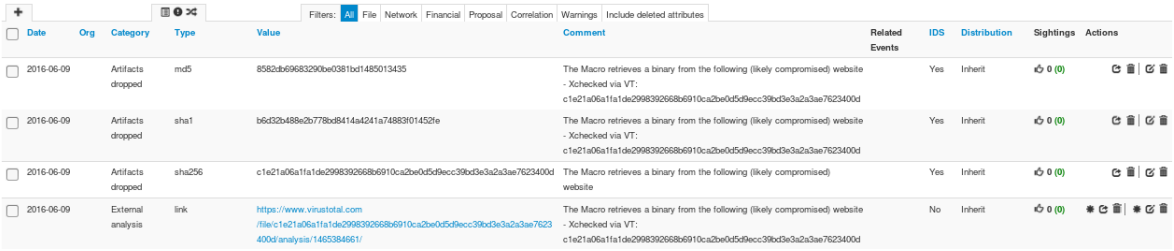


Figure 1.3: MISP : Attributes of the event

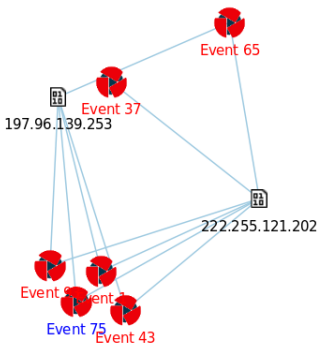


Figure 1.4: MISP : Correlation graph for an event

Chapter 2

Information Sharing State of the Art

This chapter is aimed to give an overview of existing works in mainly two specific domains, the first one will be about information sharing itself as well as techniques used for privacy concerns. I will also take advantage of this chapter to explain standards before concluding on this state of the art.

2.1 Information Sharing

As usual, a good starting point is the already made state of the art on the subject, one really interesting is done by Gregory White and Keith Harison in [34] where they explain the evolution of the information sharing by comparing it with the evolutions of the USA laws and their impacts. Even if the analysis is about the USA, it is interesting to have an understanding of how it starts. For example, the first try of sharing standardization was done just after the worm released by Robert Morris in 1988.
With a step back, they rapidly realized that information sharing could have been a better and faster way to protect critical infrastructures. They also explain, in correlation with the two principal laws called the Presidential Decision Directive-63 in 1998 and the Executive Order 13636 in February 2013; how organization like Information Sharing and Analysis Organizations (ISAOs) where created with their four foundational objectives. I've sum up them:

- Each organization should be able to participate.
- Development should be kept public.
- Voluntary is not a requirements
- Take into account the need for confidentiality and privacy.

They also introduce the first sharing program developed by the Department of Homeland Security (DHS) which is the Cyber Information Sharing and Collaboration Program (CISCP).
With that, they introduce some standards like TAXII, STIX and CybOX on which I will spend more time later in their respective sections. After that, they go through challenges that are facing the ISAOs. The major one is a privacy and confidentially concern, any information that an organization agrees to share with others, no matter who those others are need to be kept private and confidential and only released to individuals or organizations that have a right to have access based on the agreement that are signed by members of an ISAO. This concern is really interesting as this master thesis is really on trying to ensure that property without the need of trust. What is also interesting is that they explained that privacy means that personal information about individuals within a member organization should remain private (in the context of information sharing). While, confidentiality refers to information about organization that could lead to give others a competitive advantage. In this article they were focus a lot on trust which is one of the

most important property to ensure when dealing with a sharing group.
We cannot share confidential information if we do not trust the other party to keep it confidential!

After that, a recent survey trying to compare the different vendors for threat information sharing was published in 2017 [26]. The authors have conducted a systematic study of 22 threat intelligence sharing platforms and this article is a short sum up of their key findings I believe not always true and with a lack of example to support their say. Thus it was a general still really interesting survey even if a detailed analyze of the 22 different tools would be really welcome.
On the other hand, their analyze of the different types, focuses, standards and licenses used by the different platforms bring a lot of information on what should be used in todays sharing platform.

I've introduced the name of some standards. They are really important as we cannot share if we cannot speak the same language. Thus these standards make possible threat sharing but also allow to make it automated. Beside that, when we decide to start in this subject. It is really too important to avoid making mistakes that perhaps have already been made. That is why some organizations like the National Institute of Standards and Technology (NIST) and the European Union Agency for Network and Information Security (ENISA) took care of publishing guidelines to help organization to involve themselves in threat sharing.
In Europe, the Commission recognized that they have a role to play in making the information more secure. In order to do that, they wanted to create the first pan European Information Sharing and Alerting System (EISAS on which we can find a report on the implementation [12]). For that, they have asked to ENISA to define the requirements. After that, member states that weren't yet using information sharing were really interested by and asked ENISA to develop a good practise guide based on the observation of existing exchange. This gave an intersting guide [11] aimed to assist member states but also relevant stakeholders (organizations that are using communication network and information systems) in setting up and running network security information exchange.
In this guide, they go through setting up information sharing but they first look at the advantages and also identify laws that could get in the way. This is really important as organizations need to ensure not being recognized as a cartel or sharing as a commercial advantage. They also focused on the trust that needs, for them, to be created thanks to regular face-to-face meetings. They also need to use TLP in these meetings.
Then, in 2016, NIST also published its guide [16] that seems to me more complete and more mature. This publication is also more focused on organization than on States (unlike ENISA) and provides guidelines to improve cybersecurity operations and risk management activities through safe and effective information sharing practices.
For them, cyber threat information consist of IOCs, TTP (tactic, techniques and procedures), suggested actions to dectect, contain or prevent attacks and finally the findings from the analyses of incidents.
They go through different topics that are benefits and challenges encountered. How to establish and participate to sharing relations. Onces again, they draw attention on the importance of trust and to be compliant with legal and organizational requirements among other challenges.

On an other side, the paper written in February 2017 by Aziz Mohaisen et al. [24] focus themselves on rethinking threat intelligence and to assess the risks. They argue by [23] that intelligence sharing is the only way to combat our growing skills gap in term of security. But they mostly claim that there are a lot of issues that needs to be explored in order to realize efficient and effective information sharing paradigms for actionable intelligence. They even go further by saying that understanding the risk of sharing or not sharing is absolutely needed for every organization.

Briefly, not sharing is a risk as we don't receive information that could avoid us to be compromise as we can see that more and more security breaches use the same attack vectors. But in the other way, sharing information without proper restrictions may leak a significant amount of information about participant and their operation context and that can be used by an attacker to learn their vulnerabilities.
One solution to that is to have a very limited sharing community only with highly trusted participants. But is it really what we need to do ? If the participant are not enough, we could not get the needed information ...
But, if we have less trusted participant, we also need to understand to risk of leakage that could lead to both monetary and reputation loss. Taking all that into account, they propose new way of thinking for threat intelligence by defining models, communities and adversaries with their threat model. They propose then architectural solutions as a way of assessing the quality of sharing.

2.2 Existing Techniques

Then, It is interesting to see existing techniques used in information sharing to ensure privacy and confidentiality in case where we do not want to rely on trust. This following paragraphs are then used to discover what is use and what was used before in order to get ideas on what could be implemented.

The first step is data sanitization, this is quite interesting even if it could not be applied in our case (as explained in a later section).
Then there are articles on confidential database and S2P computations.

One of the first article that presents some concerns about privacy in sharing security alert is [21]. More precisely, they were concerned about protecting site-private topology, proprietary content, client relationship and site defensive capabilities or vulnerabilities.
This was done in two steps, the first one, data sanitization, consist in removing confidential data and remove useless information: Don't take the risk revealing information to an attacker if this information is not needed by the user.
The second one is the correlation/aggregation work were alerts are linked together for analysis purpose.
They have used three different categories of data coming from DShield ansd Symantec's DeepSight:

- Firewalls : They consider all "deny" as a possible attack
- IDS : They remember logs of attacks that the IDS has found
- Anti-viruses softwares : gives also some interesting logs

For removing all useless informations, we can hash all confidential data.
This work is directly done by the company. This has for advantage to avoid the need of trust on the repository.
This technique is quite well working if, the data has a certain size. But, on the other hand, it is not useful for IP addresses, if an attacker is targeting a company, it has to precompute only 256 or perhaps 65536 IP address hashes. Thus this is not brute force resistant.
For each alert, we have two different IPs, the source IP (ip_src) and the destination IP (ip_dest). We can classify all these IPs in two categories:

- Internal IPs : IPs that belong to the company
- External IPs : IPs external to the company

The first category is, of course, the one that we want to protect and in order to do so, these IPs are hashed with a keyed hash function like HMAC. While the second type is hashed by a simple hash algorithm like SHA-1.
The result is that we can compare all SHA-1 hashed IPs together while only companies can decrypt their own internal IP addresses.
An attacker can not make the difference between an IP hashed by HMAC or SHA-1. Thus, for attacking the dataset, he would need to try to crack all IPs as if they were SHA-1 fingerprints (For example TMTO). This, without forgetting that they receive millions of IPs, make it unfeasible for an attacker to bruteforce the whole dataset.
They are also using another set of protections like the randomized threshold for publication of an alert but it goes out of the scope of this work.
In sanitization, the also round all timestamp to the nearest minute in order to add some uncertainty.
The second step is the correlation, they spoke about historical trend analyses, source/target-based analyses and event-driven analyses. These analyses worth spending time on but independently of this specific focus.

Then, [35] was also working on confidential data sharing starting from the first article, but they came up with a new interesting idea, instead of hashing confidential data, why not generalize it and do probabilistic correlations.
(They also used a technique to create probabilistic attack scenario which is a set of alerts that are put together to create a bigger attack).
Guided alert sanitization with concept hierarchies:
For example, if we have an IP 192.168.1.123/32, we can generalize it to 192.168.0.0/16.
The depth of the generalization is chosen thanks to the entropy or by the differential entropy technique explained in [9].
Alert Correlation:
They focused on defining similarity functions between sanitized attributes and building attack scenarios from sanitized attributes

This article was interesting for seeing a technique of data obfuscation. And then to create correlation analyze but, it's difficult to apply that technique in order to create a database of still usable data for prevention or detection !

I've explained some solutions that can be applied to IP addresses or file (just hashing them). But, what if we could do the same with all network packets and still getting some privacy!
That's the goal of [25] ! Today, it's not enough to analyze IPs, URLs and so on. We need to go deeper in it, that's why they propose a technique based on the byte distribution of the packets. They used PAYL and Anagram [33], systems that they have created and which are really useful in these analyses.

To remove ??? => Check articles

Sanitization is used to protect information by keeping privacy, but, as [24] is referencing, there are other ways for sanitizing data. Some of them are k-anonymity [28], l-diversity [22] and the key privacy guarantee that has emerged is the differential privacy [10].
First, k-anonymity is an attempt to solve the problem of anonymizing a person-specific field structured data with formal guaranteed while still producing useful data. A dataset is said to have the k-anonymity property if the information for each person in the dataset cannot be distinguished from at least k-1 individuals. This is done by removing attributes or by generalization as seen earlier. But this technique was unfortunatly performing poorly for certain applications. Thus, l-diversity was an extension to it in order to handle some of the weakness of k-anonymity. It does that by increasing group diversity for sensitive attributes in the anonymization mechanism.

Read cited articles instead of simple the first article to be sure of my understanding

As confidentiality and privacy are such a big deal, could we only share data when there is mutual benefits ? Yes and it is what will be done in the master thesis but there are also existing techniques where each organization are able to do some secure two-party computations with others.
The interest is to get some metrics, for example, if there are only IPs in the databases, if we can get the intersection or even only the cardinality of this intersection. We can decide, if it is non-negligible, that it could be interesting to share with them!
The paper written by Freudiger et al.[15] focused on this problem by working on a DShield dataset. They've experimented some strategies to know if it could be useful to share or not with another organization. And then, they also experimented to share the whole dataset of the company, only the data set linked to the intersection just found or only, the intersection (just to get a rough idea of what they have in common ...).
Their conclusion were intuitively expected but still interesting :

- More information we get on an attacker, the better the prediction are.
- The chosen collaboration strategy has a really big impact (some of the strategies are really useless).
- Collaboration with companies improves not only the predictions but also removes a lot of false positive.
- Sharing only about common attackers is almost as useful as sharing everything.

Mutiparty private matching peut aussi être interessant dans mon domaine non ?

Now that we see more concretely what is information sharing, a good question is to understand how it is used in todays system. A usual way of using the information is via Intrusion Detection System (IDS). Contrarily at what we could think, IDS are as important as Intrusion Prevention System (IPS) as all analyses cannot be done in real time and moreover, new information on new threats could appear latter on thanks to sharing or other ways. This could then allow to discover the attacker, even if he is already in, we need to discover him as soon as possible! Analyses had shown that the mean time an attacker stay in the system before beeing detected is about 200 days which is unbelievable.

Find article about it

Parler un peu des IDS ..

2.3 Standards

As we want organizations to share threat informations, we need them to "speak the same language" or more formally, to use the same standards.
For that, we can categorize the standards into four categories (lists of other existing standards can be found in [2, 24]):

1. Enumerations
2. Scoring Systems
3. Languages (CybOX, STIX, MISP-core format)
4. Transport (TAXII)

I will only focus on the four that I've cited before as we can see in [14, 26] that the most promising standards for a threat sharing intelligence sharing infrastructure are CybOX, STIX and TAXII. They have been developed under coordination of the MITRE Corporation and have very strong momentum in adoption by industry leaders and threat intelligence communities such as the Financial Services - Information Sharing and Analysis Center (FS-ISAC).

On the other hand, MISP core format is defined in a draft written by A. Dulaunoy and A. Iklody in [4] and is an alternative to STIX that supports more attribute types.
The Structured Threat Information eXpression (STIX) [6] provides a language to represent cyber threat information in a structured manner and is really interesting as it provides a structure to express a wide set of contextual information regarding threats in addition of the IOCs. The complete list is :

- Cyber Observables
- Indicators
- Incidents
- Adversary Tactics, Techniques, and Procedures (including attack patterns, malware, exploits, kill chains, tools, infrastructure, victim targeting, etc.)
- Exploit Targets (e.g., vulnerabilities, weaknesses or configurations)
- Courses of Action (e.g., incident response or vulnerability/weakness remedies or mitigations)
- Cyber Attack Campaigns
- Cyber Threat Actors

Moreover it tries to stay as "human-readable as possible". And the chapter 9 of [6] may be really interesting to spend time on it. It explain in detail the structure of STIX.
For expressing the observable, STIX is using Cyber Observable eXpression (CybOX)[7] that allows to state the specification of events or stateful properties. Then, we need a transport standard which is the Trusted Automated eXchange of Indicator of Information (TAXII) [8] .

Taxii explanation = ?

2.4 Sum up???

When I started the master thesis, my knowledge about threat information sharing was clearly none. I had to look up for articles and information where I was able to find some. I've finally chosen some articles and some points that I found interesting to share in order to understand the different step I followed and the different ideas on which I thought about.
Also, I've spoken about MISP as it is the platform I'm interesting on. There is an available really good awesome list [2] available where all different existing standards, tools and techniques available in this domain.
But in order to come back to MISP. The main advantages compared to the proprietary solutions are that :

- Transparency in term of code and of the contributor.
- Quantity and diversity of connected entities
- The price and the non-existence of entrance barriers

On the other hands, there exist also other kind of sharing types like DShield but they are sharing the whole bench of indicators and deals with a lot of data while MISP is more interested in events and in the analyses of what is going on.

The difference between privacy and confidentiality had been well explained but I want to add that here, when I speak about anonymity in sharing, I mean that what would be interesting would be to have a way of sharing what is going on without the need of sharing who had seen the indicators.

Then, I could find a lot of techniques in order to generate blacklists, detect event correlations and protect companies with it.
In the state of the art, I've also spoken about attack scenarios and cardinality of intersection to evaluate the benefit of sharing before doing it. These kind of techniques are not yet implemented in MISP but could appear one day.

As discussed with the risk, there are a lot of privacy concerns about sharing data. And thus, a lot of separate techniques. But there is an additional important problem that we need to take care of. For example, one of the solution for preserving privacy was simply hash the data. For big sized data, It's not feasible to brute force in order to get back the information but, for IP addresses, since in general, there are only 256 or 65536 IP addresses for a specific company, we could create a table with all possible hashes and test them all!

Then, it was really useful to see these existing techniques but it unfortunately does not mean that it could solve all of our problems. They had other purposes when designing their solutions and it can lead to some disadvantage in what we are attempting to do.
For example, Bloom Filters are a really good idea but they are not enough by their own has they do not give data at the end but just a probability of a specific IOC to belong to a specific set.
On the other hand, sanitization is not good for us either as we would lose all the interesting data with the explained techniques. Noise is thus not a valid choice either.

2.5 Where did it lead me ?

We have a data set of malicious data, IOCs, events and we want to share them. This is what is already done by the MISP project. But now, what if we want to distribute these information by giving the data set to an organization without having to trust her?
It would be really nice, computer specialists would be able to check on computers to discover infections, problems and moreover learn how to fix it thanks to the previous analyses contained in MISP. The problem is that some information are confidential and we need to have some privacy concerns while sharing.
Actually, if a company discover information on its system, why should they share it ? They already have been compromised and sharing could give useful information to an attacker, as the company topology ...
But if they do not share, why other should accept to share with them ? That is why, actually all sharing groups rely on trust. Here, the trust must be replaced by something else.
Sanitization is a good idea but, it would modify data up to make them unusable for every one.
What then if we could find a way to share only if the user has really knowledge of the event and can share information on it as well, or is really infected by and need help that could be given by the information.
I will consider this problem but with two different kind of solution. The first part is when the database could be shared because an attacker could not get information from it (Easier to get data, so perhaps we cannot put all data on it).
And the second approach is one that still need a server to respond but allowing more privacy.

But still, the idea that a common user have access to data while an attacker, which is a specialist, cannot seems infeasible in a lot of cases. The attacker always ends with the data but, if he takes 1 second, 1 day, 1 week, 3 months or years is different because, we can then think about how long a data is valid?

2.6 Misp-Worbench - hashstore

Misp workbench[1] is a set of tools to export data out of the MISP MySQL database and use and abuse them outside of this platform.

They have already implemented a privacy aware tool called hashstore. This was implemented in redis[2]. This is working by creating a dataset of all hashed IOCs. And then by the use of the redis server, we can use a redis client to request data.
Like that, if you want an information on a particular IOC, you need to already know it to be able to take its value's fingerprint and make the request.
On the other hand, coming back on the small data problem, if we consider an attacker that want to try every possible IPs, for IPv4 it represents 4.228.250.625 different IPs that needs to be tested. Even if it is a lot it is still feasible as redis is really fast (The attacker even pipeline its requests)! Moreover, not all IPv4 need to be tested, for an example we can avoid private subnet like 10.0.0.0/8, 172.16.0.0/12, 192.168.0.0/16 which already represent 17.891.328 addresses.

Check that, I think that it was not feasible in the system but I need to check why

2.7 The goal

Change that title

- Explain that we want the user to have a data base directly on his computer
- Explain that this database should help the user to get information on the events
- Explain that even if the information we still need ot have access to MISP => But as we don't have the whole information we could have in this database data from other instances that we have not access yet and allow to later share
- Explain the analyses that will be done

2.8 Limitation: What could be expected

What we want ? A way to share information only with people that have already some information or that are facing the related event. It would be amazing to do so but it would mean to make available a dataset where we could allow them to query.
The hashstore was a great idea as we need to now what we are looking for to request but, for a small data that could have only 10000 possibilities. What if the opposite player try ever 10000 possibilities to get what we know?
We could make a limitation on the allowed number of request but, a system that would like to check everything that it sees on its network should still be allowed. Could we differentiate both ?
So if we want to make the task more complicated for the attacker, we need to make it more complicated as well for the authorized used. This is quite difficult because if we make it infeasible for the attacker, we are also making it infeasible for the user as, with the example, there is only a 10000 factor between legal utilization and an attack.

We thus know that it is intrinsically impossible to fully hide small IOCs while still allowing a subscriber to evaluate the rule. But we want to find a solution that could be an approach this problem.

[1] https://github.com/MISP/misp-workbench
[2] http://redis.io

Chapter 3

Standards Ideas

Now that the subject seems clearer, there are a lot of different possibilities that could be explored. This chapter will be the opportunity to dwell on some of them in order to look for their strong points as well as their weakness.

3.1 Bloom Filters

Bloom filter is a space efficient probabilistic data structure used to efficiently test the membership of specific values.
If a dataset is only used for testing the membership, instead of storing the whole dataset that could be really large, we can use a bloom filter structure that use less than 10 bits of memory per element with a 1% rate of false positive which makes it space efficient.
A simple example to show the interest could be a data set of all the students' name and surname at university. Taking the hypothesis that each name and surname has approximately 6 characters and that there are about 20 000 students (Harvard). The dataset should use 20 000 * 6 * 2 * 8 (=1920000) bits while a 1% false positive bloom filter could be only about 20 000 * 10 bits which is already 9 times smaller for such a small data example.

3.1.1 Data Structure

A bloom filter is a m bits array (fig. 3.1) all set to 0 at the starting point. Beside that, there are also k independent hash functions.

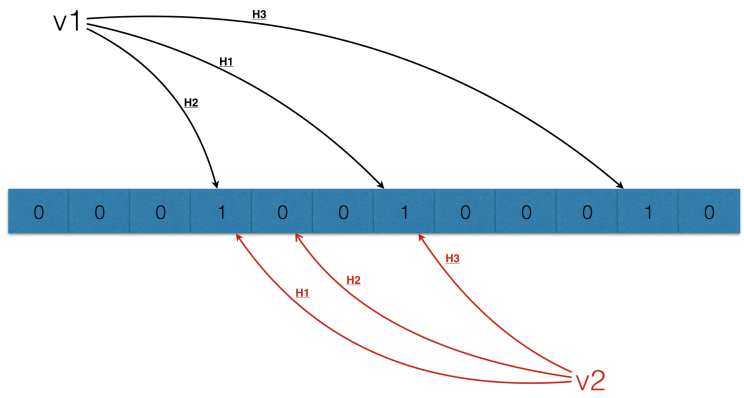


Figure 3.1: Structure of a bloom filter with m=12 and k=3

To add the value v1, each hash function gives an index where we set the indexed bit to 1. In the opposite, if we want to test a value for membership, we have to check all indexes given by the hash functions. As we can see on figure 3.1, v2 is not in the set.

3.1.2 False Positive Rate

Bloom filter is a probabilistic data structure as when we are checking for a value, either the value is **not** in the set or is **perhaps** in the set.
In the latest case, this is due to false positive where all the k hash functions gives indexes already set to 1 (fig. 3.2) while the element should not be in the set.
Even if this could be annoying that is what could make that data structure interesting while trying not to reveal the whole data set.

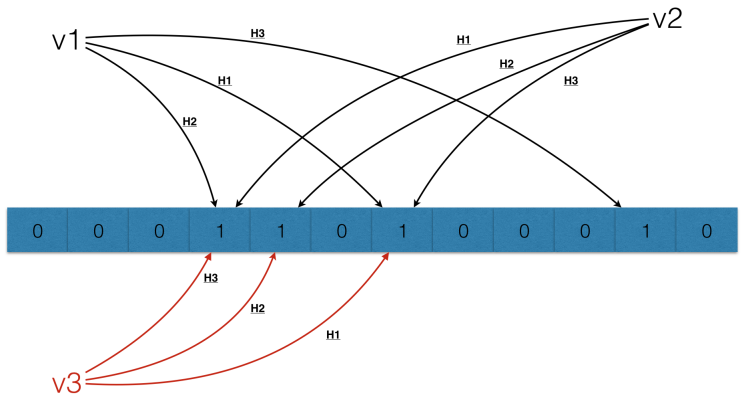


Figure 3.2: False positive v3 in a bloom filter with m=12 and k=3

3.1.3 Control the False Positive Rate

If we check for a value in a bloom filter, at the end, we only know if it is not in the set or a probability to be in the set. Playing with this probability could be interesting for hiding some informations.
If the probability is about 50% in IPs, it looses all interest for an attacker to brute force attack as at the end it would give him no information. While, for a real user, requesting for one IP could still bring some information for example knowing that an element had never been signaled before.
This is why it could be useful to get more information on the false positive rate and how we could use it.

If n is the number of element inside the data structure we can approximate[1] the false positive rate by $(1 - e^{\frac{-kn}{m}})^k$ but a more useful way is to define k in function of the willing rate for the false positive. In this case, we can notice that k is independent of the number of element inside the set while on the opposite, it is m who need to be changed in function of the number of values n:

$$m = -\frac{m \ \ln(p)}{\ln(2)^2}$$

$$k = -\frac{\ln(p)}{\ln(2)}$$

[1] https://en.wikipedia.org/wiki/Bloom_filter

(These are approximations)

3.1.4 Information Leaked by Bloom Filters

It could seems useless but nevertheless it could have huge impacts. If we create bloom filter and we ensure a false positive rate, with the size of the array (m) and the number of hash functions (k). An attacker could approximate the number of values in a bloom filter like shown by Swamidass et al. [27]:

$$n* = -\frac{m}{k} \ln \left[1 - \frac{X}{m} \right]$$

with X the number of bit set to 1.

Is it important ? => Important à expliquer vu que j'ai le mm prob :O

3.1.5 Is it enough?

It is not enough alone as we not only need to test element for membership but we also need to get back information if it is. Moreover, when developing a system with bloom filter we need to take into account the small amount of information that could leak.
But it is still an interesting solution and twinned with the later explained implementation could have great benefits.

3.2 Machine Learning

Machine learning is **the** actual technology for security. A system that could evolve and learn by itself. No need for updates!
Early in April, "Beyond the blacklists: Detecting malicious URL through machine learning" showed at the Black Hat Asia conferences that it can work.
They focused on URL lexical feature like the URL length, vocabulary, path, hostname, parameters, HTTP headers (in average, there are less fields with malicious connections) and they manage to get a 84% detection rate on about 750K malicious URLs which is really promising.

Machine learning is thus a really promising tool in this domain but, the idea here would be to use it completely differently. We don't want it to continuously learn but we want an algorithm to learn a specific data set.
Then the control of false negative and positive would be done via overfitting.
In this particular case, overfitting is what is needed as we are not interested in inference but just in recognizing the elements' membership.

This technique could be interesting to keep more control on the privacy and confidentiality, an attacker would learn even less things than with a bloom filter.
For the size, a model can also fit with less memory than remembering the whole set but will differ in each algorithm.
So the base concept representation (BCR) of this technique on different example could bring a lot of information on the usability of this technique. But it could be difficult to tune and would have the same disadvantage as the bloom filter which is that it only decides membership and cannot give useful data back to the user. One last important thing to be aware is that some kind of models could leak informations inside them for someone really understanding the system. (Ex clear decisions for a decision tree)

3.3 Secure Multi-Party Computation

Each organization possesses its IOC database and is not ready to share all their informations. But, once again, if it could help an other organization to solve its problem, they agree on sharing only these informations.
That is where Secure Multi-Party Computation could play a role as there are techniques to find the intersections of 2 datasets. This is also called private matching [5, 20].

This technique has only a big disadvantage which is heavy computations on both side. But still, it could be very useful to share information between MISP instances.

3.4 Proof of Work Database

This idea is to slow down brute force attack thanks to proof of work. Even if this idea is one step closer to the final implementation, it is still different by the need of a server to communicate with for each requests.

Here, we want to keep a database, we don't want any false positive nor false negative. But by doing that, we make our database available to brute force attacks.
We just need to add a new wall between the user and the dataset that add time consuming computation for the requests.

A possible implementation could be with a random key chosen by the database server. This key could be regenerated with a small period and also every 1000 thousands requests.
On the other hand, the client should brute force this small key each time it has changed. This slow down brute force attack while still be useful for a user that is doing less than 1000 requests to test its system.
For example, a possible request message for an ip could be : hash(IP)||hash(IP||key)

The idea is interesting even if it is still not bullet proof against brute force attack. It also need a server to respond at each request.

3.5 Conclusion

I've explained some basic ideas that could lead to different implementations, lot of work are done especially on private multi-party matching that could lead to the best option if requests are rare or, for example if it is used to periodically compare and synchronize data related to the same events between MISP instances.
Nearly all solutions still needs a server that we would like to get rid of (After the data had been generated) and for that, a machine learning model or bloom filters are the right track.

Chapter 4

Implementation

I've explained ideas in the previous chapter, some of these could be implemented. But a nice article [31] was published in the few first months of my work and they had found a technique that could also work here as their goal was similar one: allow parties to share information without the need to immediately reveal private information.
Their implementation was for bro, here I will make an implementation working with MISP and improve its matching properties and the modularity of the code.
By modularity, I mean that I to be able to add "cryptographic modules" in order to test different techniques with the same code.
In this chapter, I will first explain their article, then the modifications I've made with some improvements.
A discussion on the libraries and tools will follow before benchmarking tests and discussions on the implementation will follow in the next chapter on results.

4.1 Private Sharing of IOCs and Sightings [31]

This paper consider a cryptographic approach to hide the details of an indicator of compromise. They consider two different phases, sharing these IOCs and privately reporting the sightings of IOCs.

Reporting the sightings is important for monitoring the events and attributes in MISP.
Moreover MISP is now able to remember a sightings number per attribute but their technique seems difficult to implement in MISP for a standard subscriber as we should go through each attributes for each instances and each reporters while beeing online periodically (not practical at all). I've got some other ideas but I finally only focused myself on the sharing part as I would have not get enough time.
But still, It could be the most important and really useful further work to do.

They also started by warning the reader that using these cryptographic function is better than not using them at all but is not a miracle technique as it is theoretically impossible to hide the IOC's content in the used context (Subscriber receive the data to check on its system).
Additionally, it has a performance cost but in my beliefs, it worths it.
Their implementation was also in the source-subscriber information sharing model which fits our model where MISP is at the center.
First, they define an IOC as rules that are propositional formula where the propositional variables are defined over features or observables like Internet Protocol (IP) or fingerprints of malicious program (but it could be any categories of MISP). They also claim that every rules (IOCs) can be expressed in the Disjunction Normal Form (DNF) without any negation (e.g. destIP= 198.51.100.43 $\wedge$ destPort = 80).

Hereandafter the value is considered has the concatenation of all values of the IOCs contained in a rule : IPv4||port_number. It has two advantages, the first is to represent all linked values directly together and then increase the size and the number of element hashed increase the difficulty for an attacker to perform a bruteforce attack or to precompute all possible hashes.

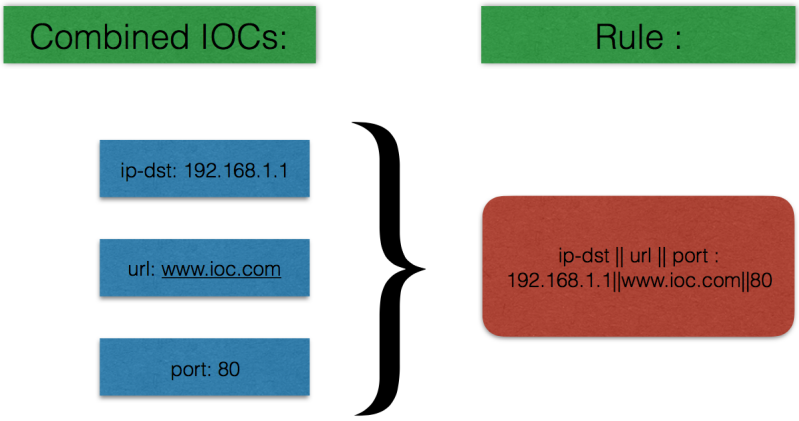


Figure 4.1: IOCs transformed into a rule

Rules as created can be directly checked on computers but they do not fit with our model yet as every values are still in clear. To solve this problem, they decided to hash the values concatenation while still keeping the structure (e.g. ip-dst||port) in clear.
The structure is what make it still usable as it says the user which are the values they need to try against the rule.
The only additional computation is that the user needs to hash his values each time he tried to match the rule.
In this way, we are still really close to the hashstore previously explained. We still lack some protections since every individual feature variable can only take a limited set and we do not get data while there is a match.
For solving that, they are using a non secret salt value chosen at random for **each** rules (IOC). Beside that, the attack time can be increased as well thanks to the cryptographic hash function used. The second part is really affecting the honest subscriber performance as well but is needed to slow down enough an attack.

They didn't stop there, considering all these ideas, they have added the possibility to include a Course Of Action (COA) message with each rule. This is the real advantage compared to other techniques. The output is precise, there are neither false positive nor false negative and additionally, a match can give the interesting data to the honest user.
To do so, instead of hashing the combined values, they use it as input for a key derivation function (KDF) that generates a key used for encrypting a message.
Thus this new type of rule, instead of just obfuscating the data, contains the type of value (as before), the salt used for the rule and the encrypted message.

This pseudo code shows how to generate a rule from the IOCs:

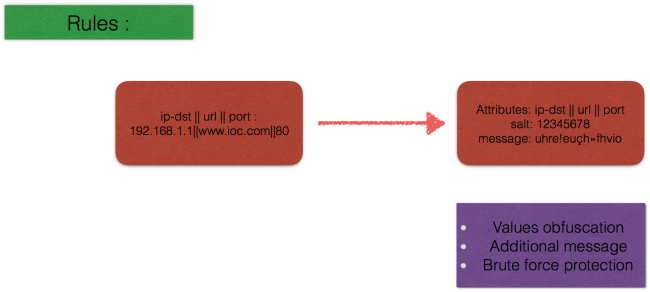


Figure 4.2: New format for the rules

```
Function createRule(AttributeTypes, values, message, Token):
        pass = '||'.join(values)
        salt = GenSalt()
        key = kdf(token + salt + pass)
        message = 16 * '\x00' + message
        encr_message = encrypt(message, key)
        return dictionary(attributes=AttributeTypes,
                          salt=salt,
                          ciphertext=encr_message)
```

Then, this pseudo code explain how to check values against one rule:

```
Function matchRule(dictianary(type -> value), rule, token):
        pass = '||'.join(values in rule.attributes)
        key = kdf(token + rule.salt + pass)
        if (decipher_16first_bytes(rule.message, key) = 16 * '\x00'):
                return decipher(rule.message, key)
        else
                return notMatched
```

Trust is one of the major problem in information sharing and is discussed in all standards, guides and articles. Here they have an attempt of solution that could really works. As a user cannot normally decrypt the complete rules data set, if for each generated rule, we add a source identifier in the KDF. This result in the ability to track the origin of a leak.
Even if it is a small add, I'm intimately convinced that it could help organization to decide sharing more confidential data as they could turn themselves again the leakage source in case of problems which could limit the cost damage as well as the reputation damage. Moreover, in MISP there is also an identifier that could be used and which is called a Token.

In their implementation, they are using both HMAC and PBKDF2 for the key derivation function. I had the opportunity to exchange mails with the authors of the paper that prefers the HMAC algorithm as it is designed to give a pseudo random key in just a single iteration and that it has been deeply analyzed [18]. While PBKDF2 is generating random keys that becomes more secure as we the number of iterations increase. Moreover, the behavior has not been described (to the best of our knowledge) in academic paper even if the is a NIST recommendation on its usage.
They also used an AES encryption scheme.

Their proof of concept code is available on their repository, `https://github.com/CRIPTIM/private-IOC-sharing`.

4.2 My Implementation

The article of Tim van de Kamp et al. gave me the starting point I needed as the code was available and under the MIT license. I've thus created two basic scripts starting from theirs where I've written it back to fit with MISP. At first, I've kept exactly the same scheme as they are.
The first step was to retrieve data from MISP, either from the web API, or by a direct connection towards the mysql database.
Then I've improved the code with different new functionalities and realized that, on the contrary of all university projects we had, the project needed to often be completely refactored to allow additional features and to keep it simple for an external user to understand how it works. I will thus go through theses different implementation parts in the next few sections.

4.2.1 Get data from MISP

To create the rules with the encrypted messages, we first need to get a subset of IOCs contained in MISP that are destined to IDS.
For doing so, there is two possibilities, the first one is working with the MISP API while the second one is directly connected to the mysql database.
The recommended mode will be with the web API, but as rules will be generated directly on the server, it could be more efficient to use directly the MySQL connection.

For making automation request on MISP, they have implemented a python library called PyMISP. The first idea was to use it but it was finally not interesting to import the whole library only to make one request. That is why, I've changed the system by making myself the request to the automation API by using the requests python library and the MISP token referenced in the configuration file.

The Virtual machine has been a great utility to understand how it is working, and especially how is structured the database. I've also added additional check like verifying that the email address specified is the one related to the token.

4.2.2 Parsing logs

The idea was to test the system with log files as I did with the hashstore. First I tried to reuse the parser I've created but I tried to go to deep in the analyze and it has bad performances as well as a lot of particular cases that were not working.
I thus changed my mind and used logstash and redis to store the parsed log.
It also has the big advantage to be modular and only a new configuration file is needed to parse any other type of log files.
Beside that, I'm storing the data inside a redis queue and polling on it in the script for trying to match a rule.

Add details, schemas, Structure explanation, ...;

4.2.3 Multiprocessing

When there are a lot of logs, doing one by one take too much time, but we can test simultaneously more than one log line at the same time by creating process.
Also it is important to point out that I've started this by trying to use thread but, as I'm using cpython, it is impossible to have two program running simultaneously due to the Global Interpreter Lock (GIL). That is why, I finally choose to create a pool of processes instead.

Data could be shared on redis instead

4.2.4 I/O and Rule Size Optimization

In the implementation of Tim van de Kamp et al., for each rule, they created a file to store the rule.
I can understand it in the point of vue of a proof of concept but it is a real bad idea as in unix systems, each files have a minimum size of 4k. Already with 6300 events each one possessing 153 attributes in average, it is easy to see the problem coming. Moreover, the I/O were really slowing down the system.
These problems were easy to solve. Create a complete csv file for all rules. But is still not a solution because it would mean that we should load everything in memory even if we don't use everything. That is why I finally chosen to have one csv file per type of IOCs.

Plus de détails

4.2.5 CSV or TSV?

This is a really small modification but makes files more readable and avoid existing coma in the data to make the parsing not working.

4.2.6 On URL Normalization

The goal of the system is looking for matches in MISP data. But off course, if an event has for Uniform Resource Locator (URL) www.ioc.com I would like http://www.ioc.com, http://www.ioc.com/, http://www.ioc.com:80/, ... To match as well.
I first dug a little into URL normalization like the standards and techniques already used in practice. I've realized that normalization is difficult as, in the standards, they want URL to always work after, and really be the same. For example http://www.ioc.com is not https://www.ioc.com while in matching, I absolutely don't care and I would like to both trigger an alert.
Besides that, there are standards on URL normalization [RFC 3986] that contains either normalizations that preserve or usually preserve the semantic of the URL but non that does not preserve it.
Standards are thus focus on keeping the number of false positive near to 0 but it has for impact to miss a lot of real positive and we cannot afford that.
I've thus found this paper [19] which was focus on url normalization and that goes beyond standards normalization steps like case sensitivity in the path, the last slash symbol at the end of the path and the designation of the default page (index.html, ...).
Beside that, a great summary of all these standardizations steps is available on wikipedia [3].
In the implementation, I've used the python library url_normalize:

- Take care of IDN domains.
- Always provide the URI scheme in lowercase characters.
- Always provide the host, if any, in lowercase characters.
- Only perform percent-encoding where it is essential.
- Always use uppercase A-through-F characters when percent-encoding.
- Prevent dot-segments appearing in non-relative URI paths.
- For schemes that define a default authority, use an empty authority if the default is desired.
- For schemes that define an empty path to be equivalent to a path of "/", use "/".
- For schemes that define a port, use an empty port if the default is desired
- All portions of the URI must be utf-8 encoded NFC from Unicode strings

I thus have to add some additions like:

- Removing directory index : default.asp, index.html, index.php, index.shtml, index.jsp, default.asp.
- Remove the fragment (#...) which is never seen by the server.
- Removing the protocol http://, https://.
- Removing “www” as the first domain label.
- Removing the "?" when the query is empty.

4.2.7 Only PBKDF2

While I was adding stuffs, it becomes more and more difficult to allow all these possible choices, I’ve thus started by limiting the key derivation function (KDF) choice to PBKDF2 which is not the author advised one but allow to parametrize the difficulty for an attacker to brute force some small data.
But in the opposite we cannot have a small number iteration if we want the key to look random.

4.2.8 library: Pycrypto towards Cryptography

While discussing with Tim van de Kamp by email, he said that if he had to start over, he would choose the cryptography library [1] instead of pycrypto because it seems to be a more professional implementation.
I thus have made some research on it. There are some discussion about it but `libhunt.com` gives this table 4.1 where we can see that even if pycrypto seems better written, the cryptography library is more up to date and still with a lot of activities.
Beside that, I found the library well defined and really easy to use that is why, I’ve chosen to refactor the code with this particular library instead.

Table 4.1: Cryptography Libraries Comparison

	PyCrypto	**Cryptography**
Popularity	7.2	**7.3**
Activity	0.0	**9.0**
Stars	1 345	**1 431**
Watchers	**103**	86
Last Commit	about 3 years ago	**7 hours before checking on the 13 April 17**
Code Quality (L1 to L5)	**L4**	L2

4.2.9 Structure Improvement

The code was becoming less and less clear to understand and as I wanted it to be modular for the cryptographic system used. I’ve decided to completely re factor the code.
The used structure is as following:

- src: Source code.
- conf: Configuration files.

[1] https://cryptography.io/en/latest/

- res: All downloaded useful resources like the csv downloaded from MISP.
- rules: Rules generated.

Inside src, there is also a crypto subdirectory in which are located all cryptographic function in different possible schemes implemented.

4.2.10 Top Configuration File

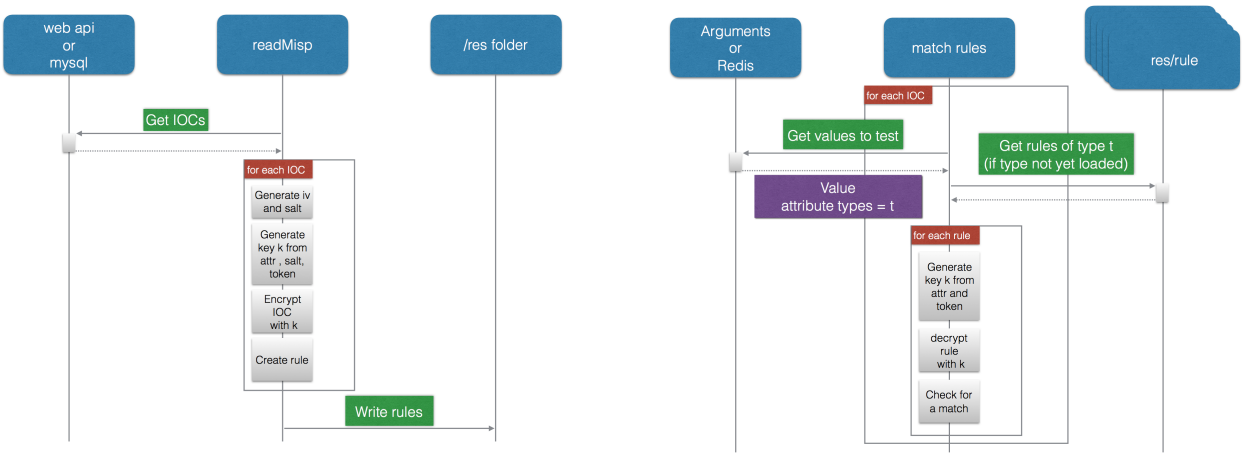
Instead of having a lot of configuration files as when I started, I've gathered them back in one general function where we only have to fill what will be used (Specified in function).
The configuration file is divided in categories and is read thanks to the configparser python library.

4.3 Chosen Cryptographic System

The idea of the generalization was to add module for handling the way data are stored / encrypted, we can use completely different cryptographic systems later called schemes. In this section, I will discuss my implementing choice for each schemes implemented.

4.3.1 PBKDF2

This is the basic system for which all the structure had been optimized and had been thought for. In this section, I will thus briefly introduce these two sequence diagrams to help the understanding the system:



For the cryptographic system, there is also an small difference in the rules that allow a really faster matching.
Instead of directly encrypt the whole message, I've decided to divide it in two parts (easy to do with the cryptography library). The first is the ciphertext-check and the other one is the ciphertext. The ciphertext-check is simply the first bock of the encrypted message. It is used only for matching, decrypt it must result in only zeros if there is a match.

For this algorithm, the user must choose an important meta parameter that will determine the "security level". This meta parameter is the number of iterations for the key derivation functions.
In 2000, the RFC advised 1000 iterations for password storing. In 2015, OWASP password storage cheat sheet [30] advised about 10 000 iterations while NIST recommends on its appendix

A.2.2 of its guide [29] to use as many iterations as it is possible with the needed performances.

Some additional analyze can be make on the ability to sustain an attack, the article make by the CryptoSense[2] website is really well documented.
They have analyzed previous work and have done some estimation for 2015. Their approximation is more a guess than anything else and I wouldn't dare trying their approximation for 2017 but I will nevertheless show their results to get a rough ideas of what could be the time to crack a password from a key generated with PBKDF2 (AES + SHA-1):

Table 4.2: My caption

Password Complexity	Estimated Entropy (bits)	1 000 Iterations	10 000 iterations
Comprehensive8	33	4h46	47h
8 random lowercase letters	37	12h	5 days
8 random letters	45	123 days	3 years 5 months
8 letters + numbers + punctuation OR 4 random Diceware words	52	325 years	3250 years

4.3.2 Bcrypt

The implementation with Bcrypt is similar as my implementation with PBKDF2. It is a password hash function based on the Blowfish cipher. It also incorporate hash to be protected against rainbow table attacks (TMTO).
It has also advantages as being more configurable and is more efficient against GPU attacks. It also seems better to use but the NIST recommendations are about PBKDF2.

On the other hand, the time complexity is not the same as for PBKDF2.
The problem with this algorithm is that they only consider the 72 firsts characters of the password for the key generation. It can seems a lot but even this url `https://translate.google.com/#fr/en/72%20caract%C3%A8re%2C%20ce%20n'est%20pas%20beaucoup` is 88 characters long. This mean is we add the token at the end (like in PKKDF2) it will not be included. On the other side, if we first put the token and the the data, an attacker would know that the 40 firsts characters will always be the same and moreover, there are only 32 characters left for the data which is not enough even for an IPv6.
As the goal was that the whole string has an impact to the key generated as well as the token to have an impact on all bits. The solution was thus obviously to hash the concatenation of the value and the token and then seed this fingerprint to the key generation function.

We also know that such key derivation function are aimed to have a time complexity that growth exponentially with the number of characters. Here every element can be considered as a 72 characters element (has the hash is longer) which means that with this technique, it can take really longer to check IPs in the system and so for bruteforcing as well. It thus need to be taken into consideration before choosing the number of iterations for small data.

4.3.3 Bloom filter

In this section, I will argue the choice for python-bloom library. This choice was actually a way more complicated than it could be as, I think it is not the fastest implementation.

[2] `https://cryptosense.com/parameter-choice-for-pbkdf2/`

But first, to understand my choice, we need to know what I really was searching for. I needed to find a way for avoiding brute forcing the database, but, on the other hand, I want it to be really fast for a common user. Thus Bloom filters already take car of the first part so, I only needed to find a fast implementation.
I also wanted the system to be storable in files without any additional informations.
So the first I've found was to use bloomd server with a python client. But, it would means that we would have to install all these things which seems not really interesting. But why not using redis to store data ? It is fast and there was a really fast implementation of bloom filters in python for redis (actually in c interfaced with python) call pyrebloom. But the code was really complicated and not easy to read. Moreover, I discovered that they are keeping additional information on keys to be able to remove elements. that, without forgetting the fact that it doesn't seemed savable in a file make me drop this implementation.
I've finally found the python-bloom library, It seems less efficient but well implemented, without any additional state and a way to save the bloom filter.

For the bloom filter, I've got different ideas, first, I need to implement the bloom filter as a tsv file. But, I also need it to be loaded each time. For this, I've added a special joker rule that is always loaded if it exists.
On this first implementation, I would like to have only one bloom filter for the whole data set. I could have implemented a bloom filter for attributes but as it is fast enough, it would use less memory and will be more efficient to use only one bloom filter.

4.3.4 Bloom filter used as to improve performance of Key Derivation Functions

Key derivation function are working to slow down a brute force attack. This is working really well as the complexity of a brute force attack time complexity increases exponentially with then length of the string to "crack". On the other hand, if we know exactly the data we are looking for and that there is a small set of different values. We cannot do anything against it.

So if we came back on a longer sized data like URLs, brute forcing them (if we are not looking especially for a value) is close to impossible and making the task easier for a user makes the task easier as well for the attacker. But it can still be nearly infeasible for an attacker to brute force the whole dataset while making the system a way more faster for the user.

With this idea in mind, I've used both type of implementations to create key derivation function improved with a bloom filter. The bloom filter avoid decrypting each rules if no rules contains the data we are looking for.
And, on the other side, when there is a match, rules can be used to get back the information needed to retrieve the event information from MISP.

Add an explanation on the message generated and how we could use it to find informations on an event

This is a serious advantage for matching. In function of the bloom filter false positive rate the system could me made a lot faster. So much faster that we could use it directly as an IDS that could updated time to time and that could work on computers, logs and that could work stand alone to generate reports on event that could affect a system.
The previous system was first aimed to be used like that but we a recommended number of iterations, the system was too slow to use in that way as data to check accumulates at a much bigger rate than the system need to look through all rules.

Thus increasing the speed can be done by decreasing the rate of false positive of the bloom filter but what if an attacker is only interesting in knowing the element belonging to the data set

?
The bloom filter could give him all the information he needs. This is the reason why we can not take a too small rate for the bloom filter.

4.4 An other KDF

An open competition called Password Hashing Competition (PHC) started on 2013 to meet the needs of a standard password hashing algorithm.
It ended in 2015 by selecting the winner as Argon2 and also a special recognition to some other algorithms like Catena, Lyra2, yescrypt and Makwa.
Argon2 was designed in the University of Luxembourg and there is three versions:

- Argon2d maximizes resistance to GPU cracking attacks.
- Argon2i is optimized to resist side-channel attacks.
- Argon2id is a hybrid version out of Argon2d and Argon2i.

There are two available libraries that implements Argon2 which are passlib and argon2_cffi. This could be interesting to use it instead of bcrypt or pbkdf2 but unfortunately, the two existing libraries have directly integrated the salt generation.
The only way of using it for now would have been to take a salt of length zero and to generate the salt by myself.
But I'm not sure that it would keep it's security value doing so.

Chapter 5

Results

Finally, the most interesting part is coming, is it really working? Is the additional computation worth it? Can we trust this system with our data ? Could with improve the code for the computation time ?
This final chapter will try to answer those questions. With the code restructuring, I think I have succeeded in creating code easy to understand but there are still some tricky hacks that I will need to get rid of for make easier addons creation. This first part will be on understanding the used dataset, then I will focus on the code profiling to understand which parts could be improved.
Then I will explain a set of benchmark to try to show the interest of this system.
Discussion on the security of the techniques and their usability will serve as a nice conclusion.

5.1 Dataset

The objective was to make all data available if additional computation must be done or this work must be continued. For that, following the same reflection as with the starting screenshots, the data directly inside the virtual machine are TLP white and are available for every one as we can directly download them on the CIRCL website : `https://circl.lu/services/misp-training-materials/`.
Thus I've used these data instead of the one directly from an instance. Even if there are a lot less data here, the computations and analyze give the necessary idea.

5.2 Code Profiling

This section is divided in three parts, a deeper understanding of the code work flow is needed to understand where the time is lost during the execution. Then, the code will be profiled for two different executions, the first is aimed to compare the time of the key generation with only one iteration. That give a right comparison point between the different functions and the second will be with 1000 iterations to see how it occupies the time.

5.2.1 Code Flow

As previously explained, a deeper understanding of the different work flow of the code are important to understand the later profile.
This explanation will be separated into the read, match and crypto part as crypto is used by both.

ReadMisp

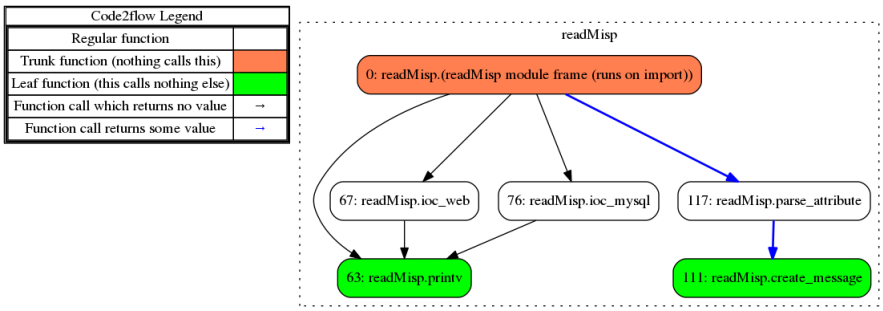


Figure 5.1: Code2flow: ReadMisp WorkFlow

ReadMisp (fig. 5.1) will first retrieve the data, either via the web API or the MySQL connection. The advantage of the MySQL is the speed for two reasons, first, there are no data transformation as we request the data as it is stored and secondly, in general, the MySQL server is accessible only from the server which means that there is less network delay.
On the flow chart, we can also see the function printv which is a call to print only if the program runs in verbose mode.
Once the IOCs available, the attributes are parsed and the message is created.
For the parsing is the real creation of the rule, which is done in several steps:

- Values from Misp are split and associated with their type in a dictionary.
- URL are normalized.
- The message is created.
- The Crypto package is called to create the rule.

In order to be modular, all the code for rule creation is separated. Each time that the program is started, it goes to the configuration file to see which Crypto module to load from the crypto package.
All modules need to implement the interface (interface.py) and then must be add in the choose_crypto.py.
For an example, the work flow for the PKKDF module is available on figure 5.3.

MatchRules

Match rules has two really different ways of working, try to match an element from the arguments (argument matching) or from redis if the pipeline system with logs is used.
The additional rangeip test is just for testing purpose and will be further explained in the benchmark sections.

For redis matching, the only difference is that a pool of process running a redis matching process is used.
Each process is actually polling the redis queue for new logs and is then feeding them to the matching system.
In the other system, the arguments are directly seeded to the matching system.

The matching system is working in different steps, first, in function of the argument types, it will load and cache the needed rules.

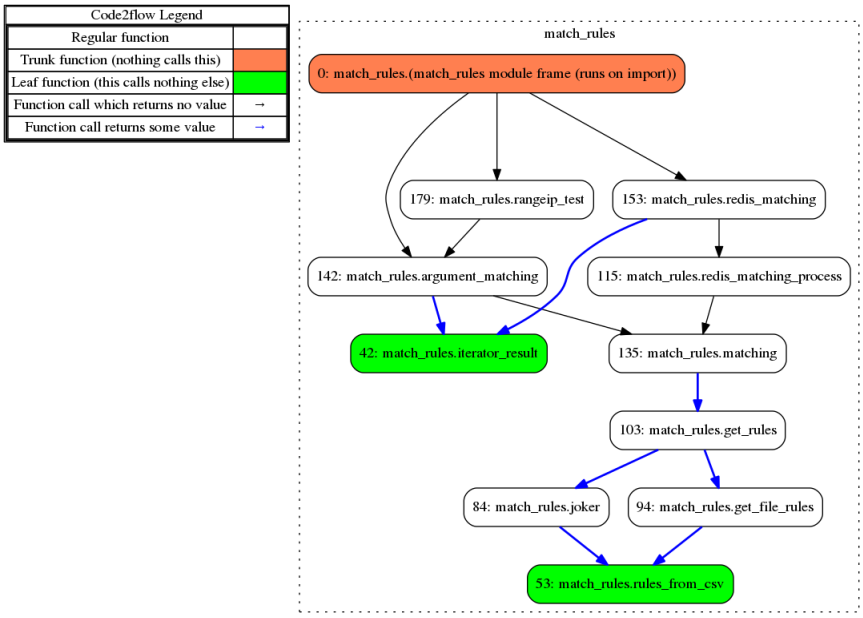


Figure 5.2: Code2flow: MatchRules WorkFlow

There are two types of rules, the standards one are divided into files but, as I've added systems like bloom filter, it was easier to have a special file always loaded that I called Joker.

Once the rules loaded, for each attributes to check, the crypto module will try to decrypt each rules up to find or not a match.

Crypto Module PBKDF2

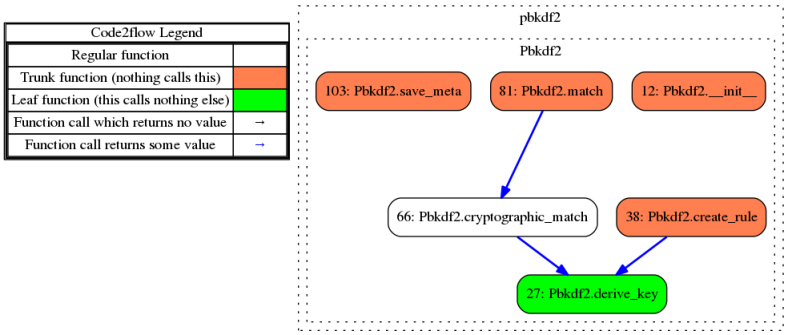


Figure 5.3: Code2flow: PBKDF2 Crypto Module WorkFlow

The crypto modules are responsible for creating and matching rules. In the particular case of the PBKDF2 module, it generate the keys and encrypt the message or try to decipher it.
I've additionally add the bloom filter chart as well to see the difference is small as they inherit the same interface.

As the types of file can differ from modules to modules, this is also the crypto module wich is responsible to save data and meta data files.
This part can take a lot of time as for saving the elements (as well as getting them) because we

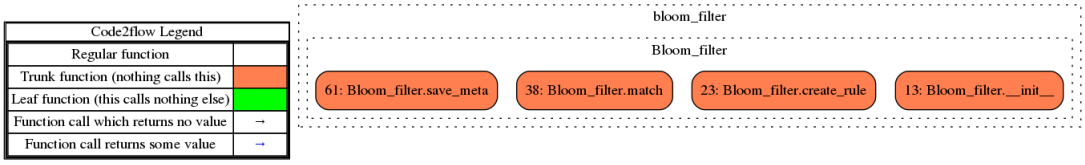


Figure 5.4: Code2flow: Bloom Filter Crypto Module WorkFlow

cannot afford problems inside the files like special characters to appear. So for that the idea is to use base64 encoding for the encrypted message, the salt (or nounce) and also the cipher-check argument (used for faster match checking).

5.2.2 Profile of ReadMisp for 1 iteration

We thus need to verify that generating the keys and encrypting/decrypting data is much more time consuming than everything else even with one iteration.
We should also do especially close attention to the normalization of URLs. Normalizations could be inefficient as I don't know how the library is implemented and moreover, I'm additionally iterating for searching regex after that.

For profiling the code, I've had some difficulties, first, I tried to use the vprof profiler as it seems to be the best and it was said that it supports python3.
But there were a lot of errors and I first didn't succeed in using it. I've thus tried to use a line_profiler (only working with python2) then profiling which was really complicated to used. I finally managed to use vprof with a little modification (to force it using python3) and I was really impressed by this tool that analyze everything. We can have graphs of the function utilizations, a line by line profiler but also a memory analyzer which can be really useful.

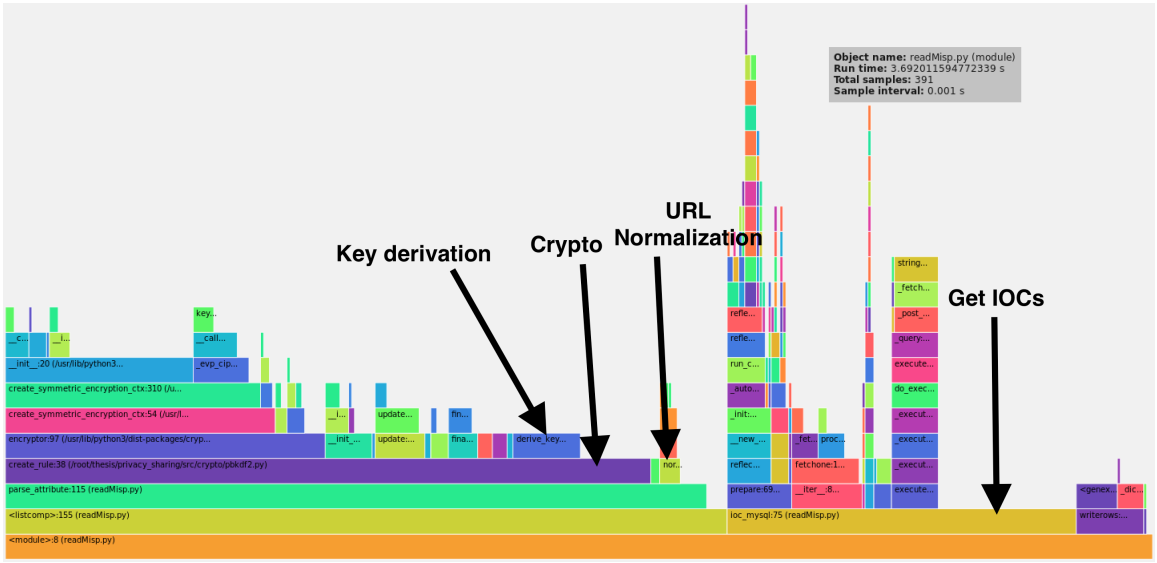


Figure 5.5: Vprof - Flame Chart Profile of readMisp PBKDF2(1 iteration)

This graph 5.5 shows that getting the IOCs from MySQL takes approximately one third of the time. And there, we also see that two thirds of this section are used directly for Mysql but

the one third left is used for ordering and small modification of the format of some values. This could be slightly improved but this would not increase the average performance as we can see with 1000 iterations.
Then writing data to files (Rightmost dark blue) takes time but cannot be improved significantly. The first question about the normalization time is answered and we see that it shows that normalizations of URLs is already a small amount of time compared to one iteration for the key derivation.
We also see that in this particular case, the major part of the time is taken by the encryption of the messages.

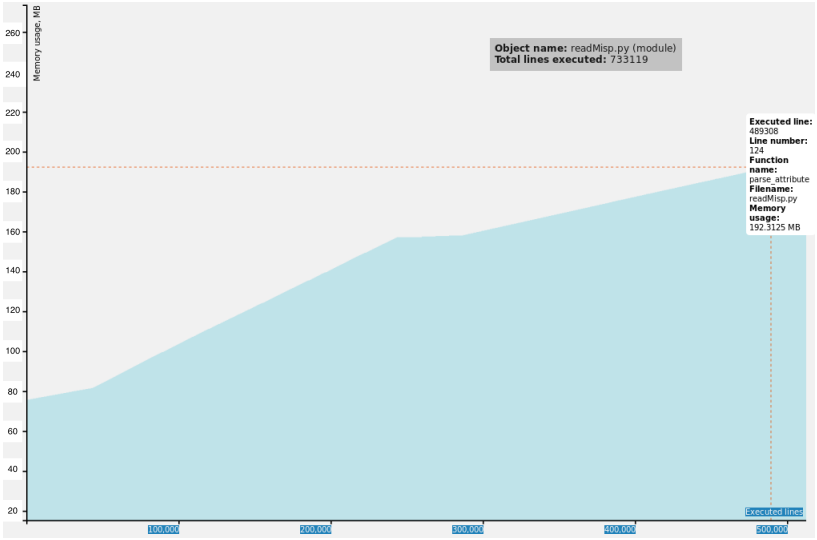


Figure 5.6: Vprof - Memory Chart Profile of readMisp PBKDF2(1 iteration)

The memory graph we can see that the highest level (on the right) is approximately at 200MB used for 27663 rules. This does not vary with the number of iterations.

Analyze the increase of memory in function of the number of rules !

5.2.3 Profile of ReadMisp for 1000 iteration

The goal of this work is to slow down the system, which means that a lot of iteration will be used.

5.2.4 Profile of the bloomy_pbkdf2 module

The first step is to look for the additional computing time for creating the rules and as we can see on figure **??**. The key derivation function is still more computationally intensive than the creation of the bloom filter.

Then it could be interesting to compare the flow when there is a match and then when the Bloom filter can directly say that the element is not in the set of IOCs.

Add picture her but I was not able to do so as my computer is not working anymore

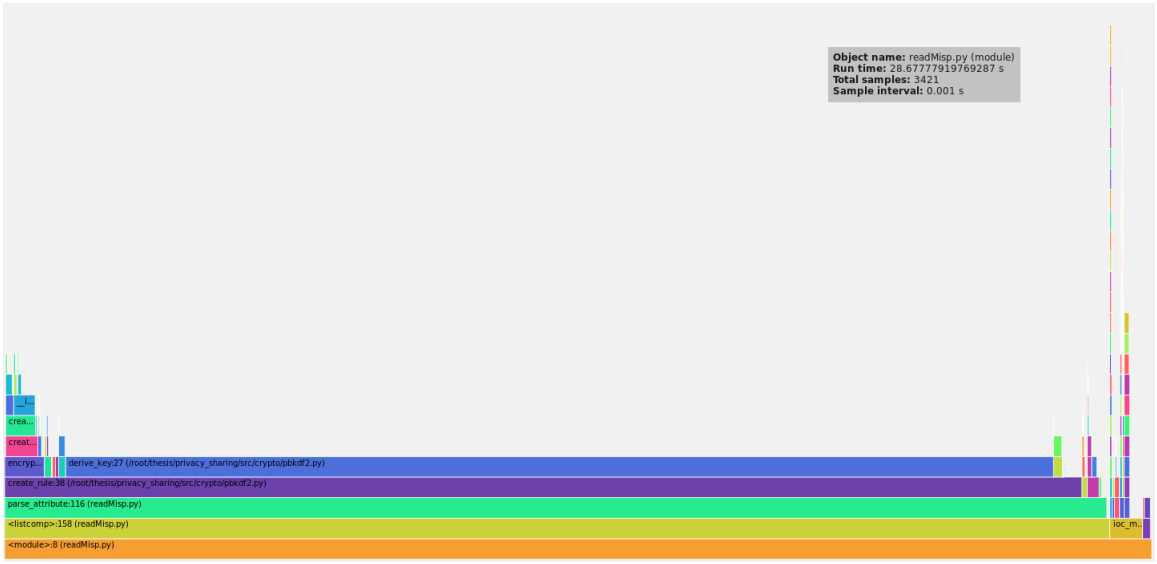


Figure 5.7: Vprof - Flame Chart Profile of readMisp PBKDF2(1000 iteration)

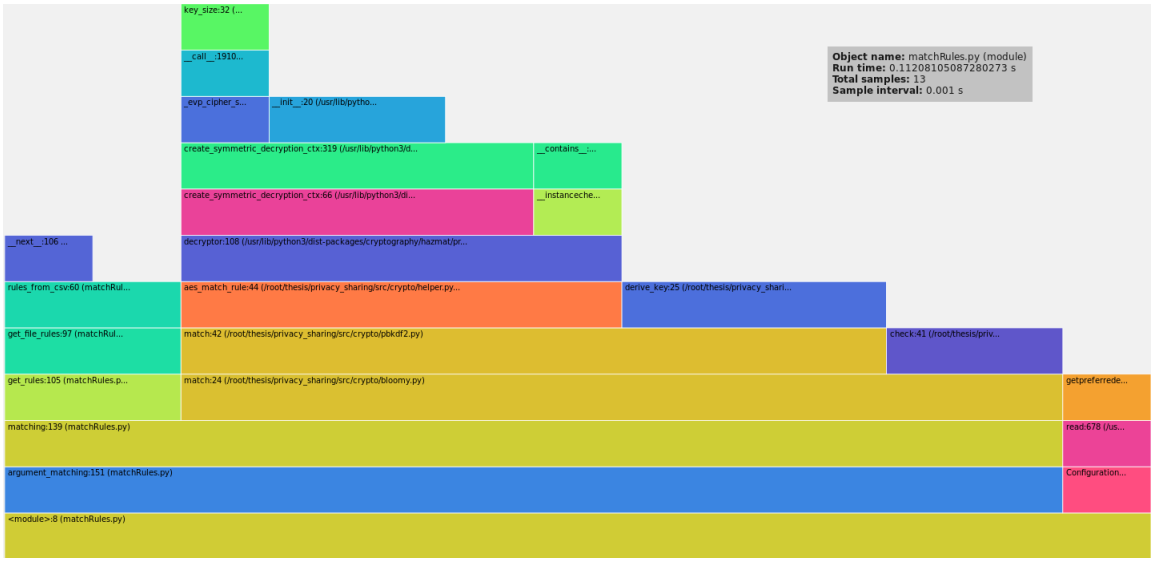


Figure 5.8: Vprof - Flame Chart Profile of matchRules bloomy_pbkdf2 (1 iteration) with a match

5.3 Benchmarking

The ideas have been explained, in the first few sections, then the implementation had been explained followed by profiles that help understand how the behavior of the system and how the computation power is used inside the system. Now it is time to see how it is working with real data, what would be the right parameters to use and why these choices. Moreover, this is in this section that we could analyse if the goal had been totally reached, reached for some part and decide if it can or cannot be used in real implementations.
This is important as we are aware of the benefits that it could have but we must stay aware of the risk of that exposure on which a system like this one could lead.

Do all needed analyzes; ajouter brute force de string avec bcrypt => explain expo and why not here

Figure 5.9: Vprof - Flame Chart Profile of matchRules bloomy_pbkdf2 (1 iteration) with no match in the bloom filter

5.4 Security Discussion

Must be dimensioned carefully !!! => puis expliquer pourquoi c'est trop cool et que l'idée fonctionne vraiment bein mm dans les domaines où on était pas sur

5.5 Further Work

- Sightings
- additional crypto systems
- additional benchmark
- ...

Bibliography

[1] About taxii. https://taxiiproject.github.io/about/.

[2] Awesome threat intelligence. https://github.com/hslatman/awesome-threat-intelligence.

[3] Url normalization. wikipedia. https://en.wikipedia.org/wiki/URL_normalization.

[4] A. Dulaunoy, A. I. Misp core format. https://tools.ietf.org/html/draft-dulaunoy-misp-core-format-01.

[5] Agrawal, R., Evfimievski, A., and Srikant, R. Information sharing across private databases. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data* (2003), ACM, pp. 86–97.

[6] Barnum, S. Standardizing cyber threat intelligence information with the structured threat information expression (stix^{TM}). *MITRE Corporation 11* (2012).

[7] Barnum, S., Martin, R., Worrell, B., and Kirillov, I. The cybox language specification. *draft, The MITRE Corporation* (2012).

[8] Connolly, J., Davidson, M., and Schmidt, C. The trusted automated exchange of indicator information (taxii). *The MITRE Corporation* (2014).

[9] Cover, T. M., and Thomas, J. A. Elements of information theory, 1991.

[10] Dwork, C. Differential privacy: A survey of results. In *International Conference on Theory and Applications of Models of Computation* (2008), Springer, pp. 1–19.

[11] ENISA. Good practice guide network security information exchange.

[12] ENISA. Eisas – european information sharing and alert system for citizens and smes.

[13] FIRST. Traffic light protocol (tlp) first standards definitions and usage guidance — version 1.0. https://www.first.org/tlp, June 2016.

[14] Fransen, F., Smulders, A., and Kerkdijk, R. Cyber security information exchange to gain insight into the effects of cyber threats and incidents. *e & i Elektrotechnik und Informationstechnik 132*, 2 (2015), 106–112.

[15] Freudiger, J., De Cristofaro, E., and Brito, A. E. Controlled data sharing for collaborative predictive blacklisting. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment* (2015), Springer, pp. 327–349.

[16] Johnson, C., Badger, L., and Waltermire, D. *Guide to cyber threat information sharing (draft).* 2014.

[17] Johnson, C., Badger, L., Waltermire, D., Snyder, J., and Skorupka, C. Guide to cyber threat information sharing. *NIST Special Publication 800* (2016), 150.

[18] Krawczyk, H. Cryptographic extraction and key derivation: The hkdf scheme. Cryptology ePrint Archive, Report 2010/264, 2010. http://eprint.iacr.org/2010/264.

[19] Lee, S. H., Kim, S. J., and Hong, S. H. On url normalization. In *International Conference on Computational Science and Its Applications* (2005), Springer, pp. 1076–1085.

[20] Li, Y., Tygar, J., and Hellerstein, J. Private matching. *Computer Security in the 21st Century* (2005), 25–50.

[21] Lincoln, P., Porras, P. A., and Shmatikov, V. Privacy-preserving sharing and correlation of security alerts. In *USENIX Security Symposium* (2004), pp. 239–254.

[22] Machanavajjhala, A., Kifer, D., Gehrke, J., and Venkitasubramaniam, M. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD) 1*, 1 (2007), 3.

[23] Malik, J. Threat intelligence sharing: The only way to combat our growing skills gap. *Information Security Magazine* (jun 2016).

[24] Mohaisen, A., Al-Ibrahim, O., Kamhoua, C., Kwiat, K., and Njilla, L. Rethinking information sharing for actionable threat intelligence. *arXiv preprint arXiv:1702.00548* (2017).

[25] Parekh, J. J., Wang, K., and Stolfo, S. J. Privacy-preserving payload-based correlation for accurate malicious traffic detection. In *Proceedings of the 2006 SIGCOMM workshop on Large-scale attack defense* (2006), ACM, pp. 99–106.

[26] Sauerwein, C., Sillaber, C., Mussmann, A., and Breu, R. Threat intelligence sharing platforms: An exploratory study of software vendors and research perspectives.

[27] Swamidass, S. J., and Baldi, P. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. *Journal of chemical information and modeling 47*, 3 (2007), 952–964.

[28] Sweeney, L. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 10*, 05 (2002), 557–570.

[29] Turan, M. S., Barker, E., Burr, W., and Chen, L. Nist: Special publication 800-132, recommendation for password-based key derivation. *Computer Security Division Information Technology Laboratory. http://csrc. nist. gov/publications/nistpubs/800-132/nist-sp800-132. pdf* (2010).

[30] Van Acker, S., Hausknecht, D., Joosen, W., and Sabelfeld, A. Password meters and generators on the web: From large-scale empirical study to getting it right. In *Proceedings of the 5th ACM Conference on Data and Application Security and Privacy* (2015), ACM, pp. 253–262.

[31] van de Kamp, T., Peter, A., Everts, M. H., and Jonker, W. Private sharing of iocs and sightings. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security* (2016), ACM, pp. 35–38.

[32] Wagner, C., Dulaunoy, A., Wagener, G., and Iklody, A. Misp: The design and implementation of a collaborative threat intelligence sharing platform. In *Proceedings of the 2016 ACM on Workshop on Information Sharing and Collaborative Security* (2016), ACM, pp. 49–56.

[33] Wang, K. *Network payload-based anomaly detection and content-based alert correlation.* PhD thesis, Columbia University, 2006.

[34] White, G., and Harrison, K. State and community information sharing and analysis organizations. In *Proceedings of the 50th Hawaii International Conference on System Sciences* (2017).

[35] Xu, D., and Ning, P. Privacy-preserving alert correlation: a concept hierarchy based approach. In *21st Annual Computer Security Applications Conference (ACSAC'05)* (2005), IEEE, pp. 10–pp.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**