AUGUST 2018

MALWARE ANALYSIS MANAGEMENT

CERN Email Malware Management System

AUTHOR:

Shivam Kapoor

Computer Security (IT-DI-CSO)

SUPERVISOR:

Mihai Liviu Valsan

Computer Security (IT-DI-CSO)

CERN openlab

PROJECT SPECIFICATION

CERN is constantly being targeted with malware, with email being the primary attack vector. The CERN Computer Security Team in collaboration with the CERN Email Service Managers have deployed many in-depth measures in order to minimize the number of malicious emails reaching the mailboxes of CERN users. The cornerstone of this strategy is the use of FireEye EX[1] email security appliances. These appliances are performing behavioural analysis of all email attachments by detonating them inside a sandboxed environment and simulating user activity.

The malicious attachments are being quarantined based on the traces of malicious activity detected once they are opened. Still, this is a very quick analysis that does not offer a complete picture of the entire malware activity.

The goal of this project was to design, implement and deploy a framework for the automated analysis of quarantined files. The basic objectives were to:

- Download newly quarantined attachments from the FireEye EX appliances.
- Extract potentially malicious files from the attachments.
- Check fetched files to determine whether they have already been seen and analysed in the past. This is not always trivial as the exact same malware can be seen in thousands of different variations, by randomizing literals used inside the malicious code.
- Submit newly seen samples automatically for analysis in an advanced sandboxed environment.

Upon end of above analysis, the results were to be automatically processed in order to:

- Extract Indicators of Compromise (IOCs).
- Obtain extra context regarding each IOC from external sources.
- Cross check if the traces of the IOC match records in our threat intelligence platform.
- Preparation of a new security event to be added to our threat intel platform.

[1] FireEye EX Appliances - https://www.fireeye.com/solutions/ex-email-security-products.html/

ABSTRACT

Malware Analysis Management (M.A.M.) or the automated sandbox analysis of quarantined malware samples focuses on a detailed analysis of malware samples reaching CERN through email traffic. M.A.M. is a side process of the main email pipeline that listens to alerts posted by FireEye EX appliances when a malicious email is detected. Apart from sorting out new malware samples, M.A.M. utilizes advanced sandbox technologies like Joe Sandbox Cloud[2] to *deep analyse* the most interesting and important malware artefacts in order to gather detailed Indicators of Compromise (IOCs) information. In addition to the analysis and management of malware samples, CERN as a responsible member of the security threat intelligence community takes advantage of platforms like MISP[3] to share the threat intelligence gathered by the project. M.A.M., a real-time daemon running persistently on a dedicated VM, is now an addition to the other security and defence strategies deployed at CERN for email security.

[2] Joe Sandbox Cloud - https://www.joesecurity.org/joe-sandbox-cloud/
[3] Malware Information Sharing Platform (MISP) - http://www.misp-project.org/

ACKNOWLEDGEMENT

I would like to thank CERN and CERN Openlab for providing me with the opportunity to be a part of Openlab Summer Student 2018 Programme. The Computer Security Team for allowing me to work in the team and always being there for any help, no matter how small and elementary.

I want to especially thank Liviu Valsan for being a supportive supervisor and taking out his time to teach me security concepts and showing me the importance of secure code. Vincent Brillault for always being there and taking me out of dead ends to bring the project back to track. Without his office visits and whiteboard sessions, M.A.M. wouldn't be complete. Thank you, once again, for all the help and being the best mentors anyone could deserve.

This project would not be over without the support of Cristian Schuszter and Pascal Oser who always motivated and helped me at every step and checkpoint. To learn so much from them in such short time is the growth spurt I always needed. CERN wouldn't be home without them.

TABLE OF CONTENTS

PART A:

PROJECT DETAILS

INTRODUCTION

CERN, the European Organization for Nuclear Research is home to the largest particle physics laboratory in the world. With more than 2,600 scientific, technical, and administrative staff members working at CERN and a community of more than 12,000 users hosted on the organization's network[4], CERN is no stranger to cyber-attacks.

While there are various attack vectors seen at CERN, email is the most vulnerable vector for cyber-attacks. While security challenges around spam and phishing emails are ever-increasing, the major threat received through email services are disguised as file attachments, malicious URLs and in some cases malicious email bodies. While traditional email security equipped with anti-spam filters and other simple defence mechanisms is capable enough for detecting malicious URLs inside emails and stopping basic phishing attacks, it is not enough to stop advanced targeted ransomware and malicious binaries disguised to bypass the basic email defence strategies.

At CERN or any other organisation, email remains the prime medium to start an advanced attack or deliver malicious payload. M.A.M. revolves around the idea of creating a side process to the main email security pipeline to *deep analyse* the most interesting and important malware artefacts caught by the FireEye EX appliances and gather detailed Indicators of Compromise (IOCs) information about newly seen samples by utilizing sandbox technologies like the Joe Security Sandbox. M.A.M. also uses the open source threat intelligence platform MISP to share the outcome of the analysis.

As best explained in the web resource[5], "Indicator of Compromise (IOC) is an artefact observed on a network or in an operating system that with a high confidence level is associated with a security event. Typical IOCs are malware signatures, IP addresses, hashes of malware files, URLs, domain names of botnet command and control servers. After IOCs have been identified in a process of incident response and computer forensics, they can be used for early detection of future attack attempts using intrusion detection systems and antivirus software.".

[4] CERN Annual Report 2017 - https://cds.cern.ch/record/2624296/files/18030409_CERN_rapport_2017EN.pdf
[5] Wikipedia, Indicators of Compromise - https://en.wikipedia.org/wiki/Indicator_of_compromise

COMPONENTS USED

The three components that were utilized in the Malware Analysis Management (M.A.M.) project are the FireEye EX Appliances, the Joe Sandbox Cloud API and the MISP Threat Intelligence Sharing Platform. Each component acted as a separate independent module in the project architecture.

1. **FireEye EX Appliance –**

   "Email Security is an effective cyber threat protection solution that helps organizations minimize the risk of costly breaches by accurately detecting and immediately stopping advanced, targeted and other evasive attacks hiding in email traffic."[6]

   The CERN Computer Security Team in collaboration with the CERN Email Service Managers utilize FireEye EX appliances in the main email security pipeline to detect malware reaching CERN via email.

2. **Joe Sandbox Cloud API –**

   "Joe Sandbox Cloud executes files and URLs fully automated in a controlled environment and monitors the behaviour of applications and the operating system for suspicious activities. All activities are compiled into comprehensive and detailed analysis reports.

   Analysis reports, containing key information about threats, enable cyber-security professionals to deploy, implement and develop appropriate defence strategies and protection mechanisms."[7]

   The CERN Computer Security Team utilize sandbox technologies such as Joe Security to *deep analyse* the most interesting and important malware samples

[6] FireEye EX Series Data Sheet - https://www.fireeye.com/content/dam/fireeye-www/products/pdfs/pf/email/fireeye-ex-series.pdf
[7] Joe Sandbox Cloud - https://www.joesecurity.org/joe-sandbox-cloud

reaching CERN. Joe Security Cloud provides a very thorough and detailed report as output.

3. **Malware Information Sharing Platform (MISP) –**

"MISP is a threat intelligence platform for sharing, storing and correlating Indicators of Compromise of targeted attacks, threat intelligence, financial fraud information, vulnerability information or even counter-terrorism information."[8]

A wide range of organisations rely heavily on threat intelligence gathered from multiple sources. They can benefit by getting involved in sharing and processing of threat intel which will ultimately increase the visibility into the threat landscape. Processed threat intel in correlated form is effective and actionable in real-time which helps organizations to detect and mitigate new and emerging threats.

CERN utilizes MISP to share threat intel and keep itself up to date with security events happening around the world.

[8] Malware Information Sharing Platform (MISP) - http://www.misp-project.org/features.html

ARCHITECTURE

M.A.M. is entirely written in Python and is a side branch from the main CERN email protection pipeline. The architecture for the project has three parts:

1. **Python Daemon –**

   The first part of the architecture runs a light-weight daemon on a dedicated virtual machine (VM) which listens to HTTP POST alerts sent by the FireEye appliances when they detect any malicious email. While emails with no malicious attachment are not interesting for this project, SQL records are still maintained for all detected emails.

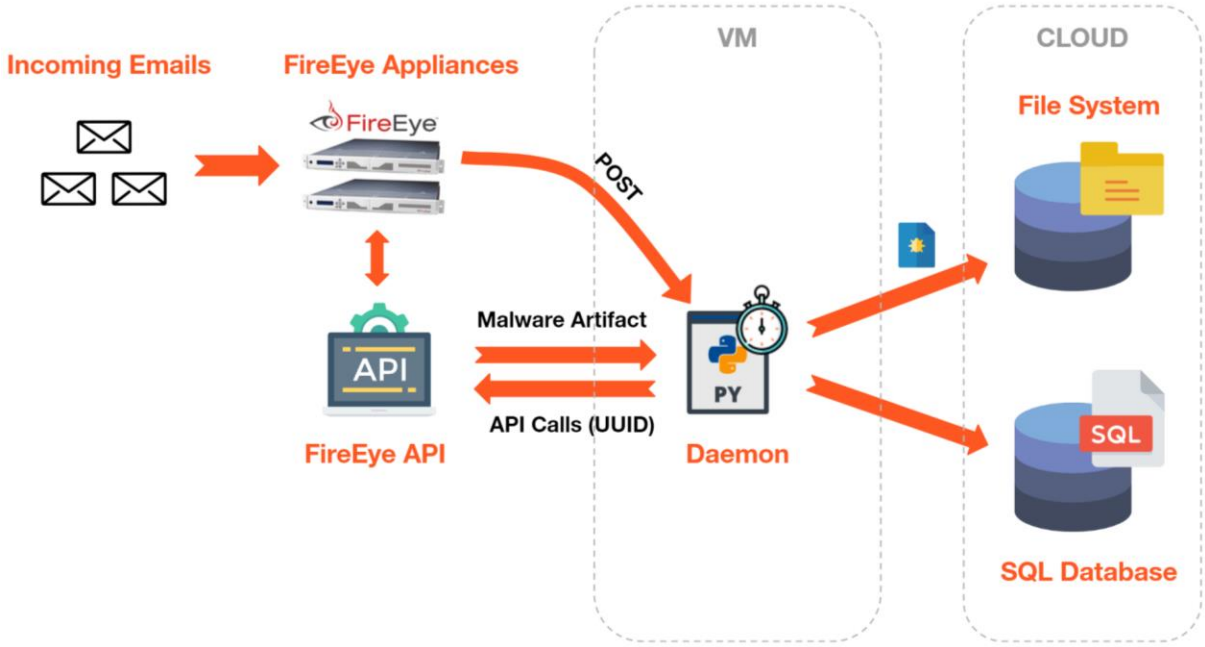


Figure 1: Part 1 of the system architecture showing the functioning of the daemon.[9]

The architectural diagram above shows the following:

- All incoming mails are passed through the FireEye EX appliances which perform behavioural analysis of email attachments by detonating them inside a sandboxed environment, sending *HTTP POST Alerts* for any malicious email detected;
- The *Python daemon* continuously listens for these HTTP POST Alerts;

[9] Acknowledgement: All icons made by DinosoftLabs, Freepik, Dimitry Miroliubov and Smashicons from www.flaticon.com.

- Once the HTTP POST Alert is received, the daemon authenticates itself to the FireEye API using standard login credentials. The daemon is optimized to reduce the number of API calls as discussed later in the report;
- After authentication, the daemon makes a call to the FireEye API with the Universally Unique Identifier (UUID) related to the alert and *retrieves the malware artefact* attached with the email;
- Once the artefact is retrieved, the HTTP POST Alert JSON, Malware artefact, and email body (if available) are stored in respective folders named as *SHA256_Filesize* to avoid collisions for similar hash;
- Important information related to the detected emails including file hashes, MIME Type, sender's information etc. are stored in an SQL database with four separate tables (see Appendix 1 for table schema and details).

2. MISP Threat Sharing Platform –

The second part of the architecture is an independent module which uses the HTTP POST Alert attributes to generate an unpublished MISP report with compound objects which can be manually published if the information is detailed enough.

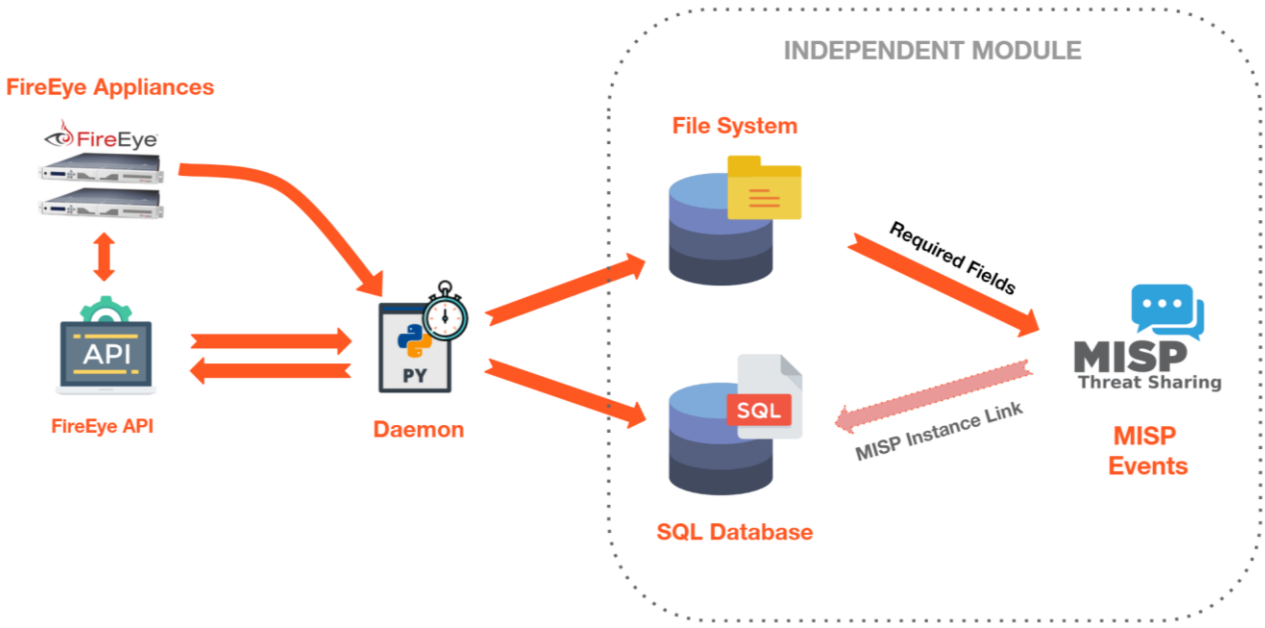


Figure 2: Part 2 of system architecture showing the functioning of MISP as an independent module.

The architectural diagram above explains the following:

- All the SQL records are shown on the M.A.M. dashboard from which the interesting ones can be selected by a human actor, e.g. a security analyst, in order to create a *MISP event*;
- The required fields are fetched from the HTTP POST Alert to be added as objects in the MISP Report;
- If the event is created for a detected email, the *MISP instance link* is stored in the SQL table for future reference. (see Appendix 2 for an example MISP event).

3. Joe Security Cloud Sandbox –

The third part of the architecture uses the Joe Security Sandbox technology to detonate interesting malware samples inside dedicated VMs in order to obtain a very detailed report for further analysis by the security analyst. Furthermore, if the threat intelligence is interesting and worth sharing with the community, a MISP event is created which can be published after review. Joe Security can create analysis reports in 33 different formats. (See Appendix 3 for details)

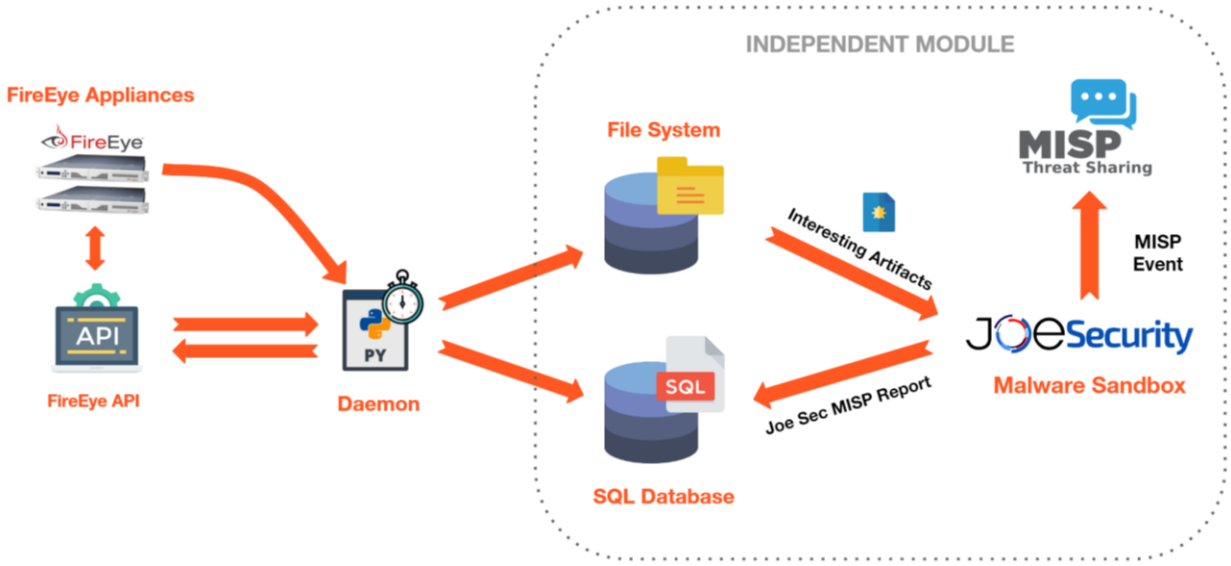


Figure 3: Part 3 of architecture showing the functioning of Joe Security Sandbox analysis as an independent module.

The architectural diagram above explains the following:

- From the dashboard, after the malware is found to be interesting by e.g. the security analyst the sample can be sent to *Joe Security Sandbox* for detailed analysis;

- The detailed report, once obtained, can be added as an unpublished MISP event;
- The MISP event can be manually published if the information is detailed enough.

HUMAN ACTOR - MOTIVATION

Throughout the architecture a human actor has been mentioned who is taking important decisions like:

- Which malware samples are interesting and important for further analysis?
- Which MISP events are detailed and important enough to be published?
- Should the malware be sent to Joe Security Sandbox for detailed analysis?

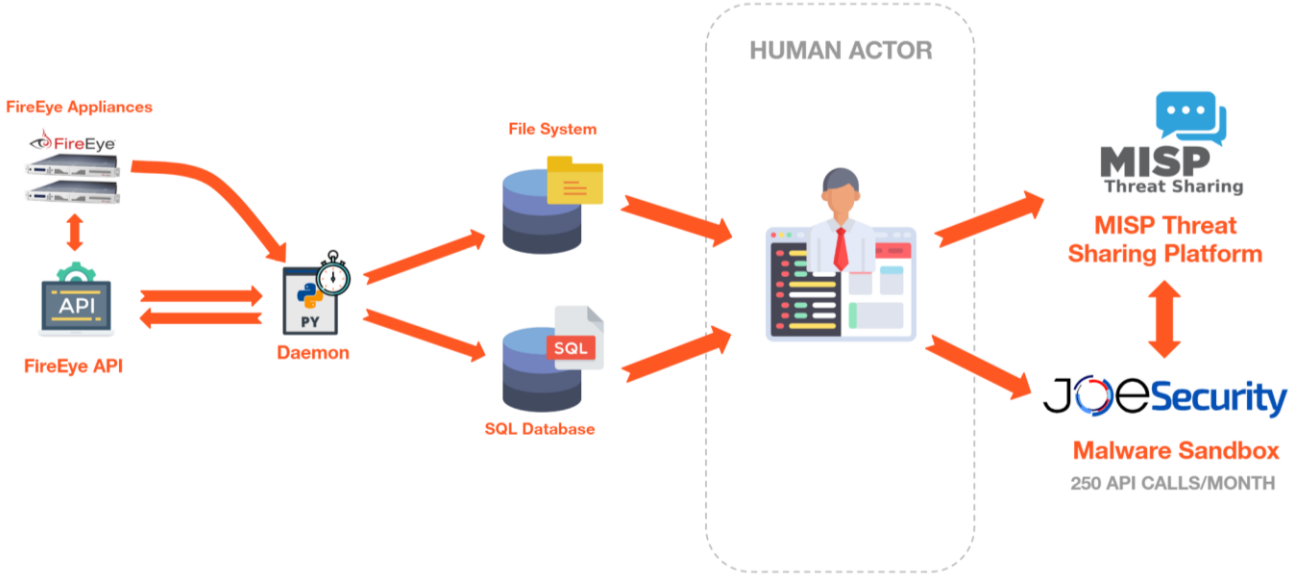


Figure 4: Dashboard is added in the architecture to provide interface.

While Joe Security Sandbox is a very important tool which helps us to unleash the power of deep malware analysis, it is restricted to only a certain number of daily and monthly API calls. Therefore, there is a need for a human actor to restrict the API calls to only the important and interesting malware samples.

For this purpose, M.A.M. uses a basic dashboard made entirely using FLASK to provide an interface to the human actor to carry out the above tasks with the click of a button. The dashboard has two views:

1. **The list view –**

   This view lists all the malware entries with specific columns including UUID, file name, MIME type, malware type and date, as basic information. The human actor can press *View* under the *More Information* menu to switch to table view. After review, *Submit to MISP* and *Submit to JoeSec* can be used for the respective course of action.

   Once the MISP report is generated, the report can be accessed by clicking on *See Event in MISP* button. Similarly, for the Joe Security generated MISP report.

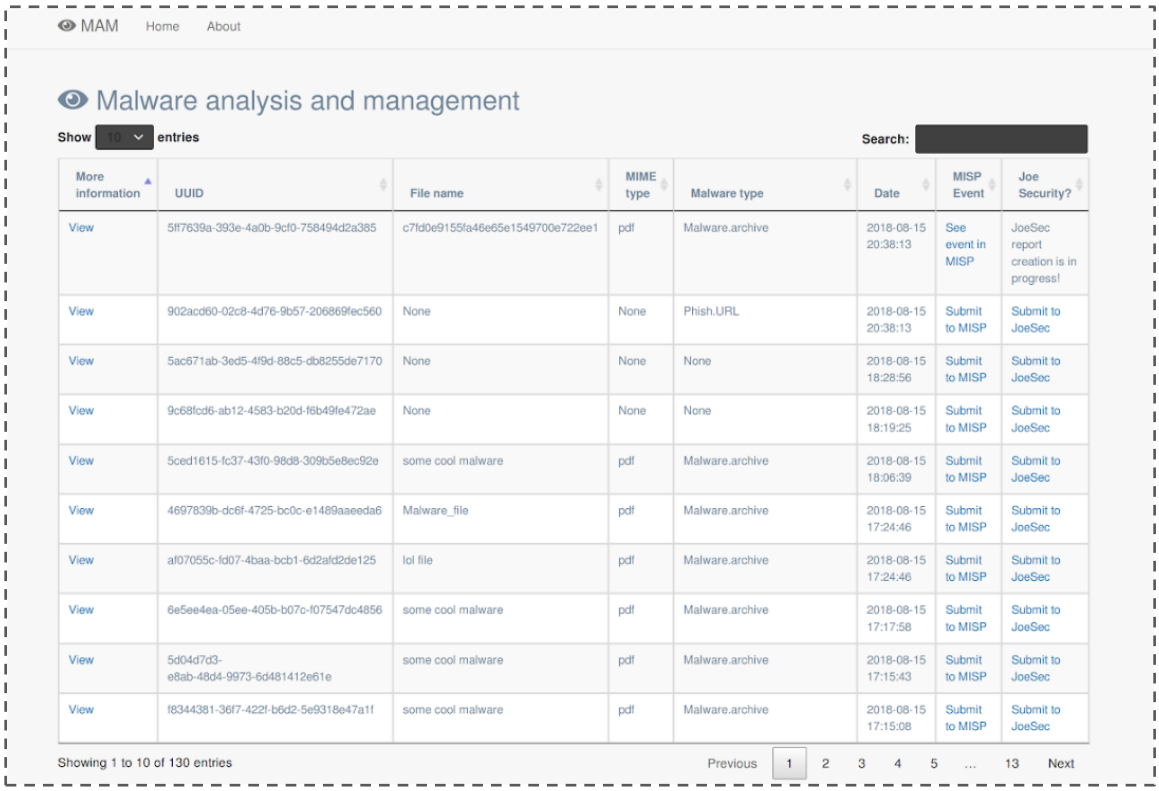


Figure 5: List view screenshot of M.A.M. dashboard entirely made in FLASK.

2. **More Information view –**

   More information about a detected malicious email can be checked by clicking *View* on the list view for each entry. The view shows details from all tables for that particular record.

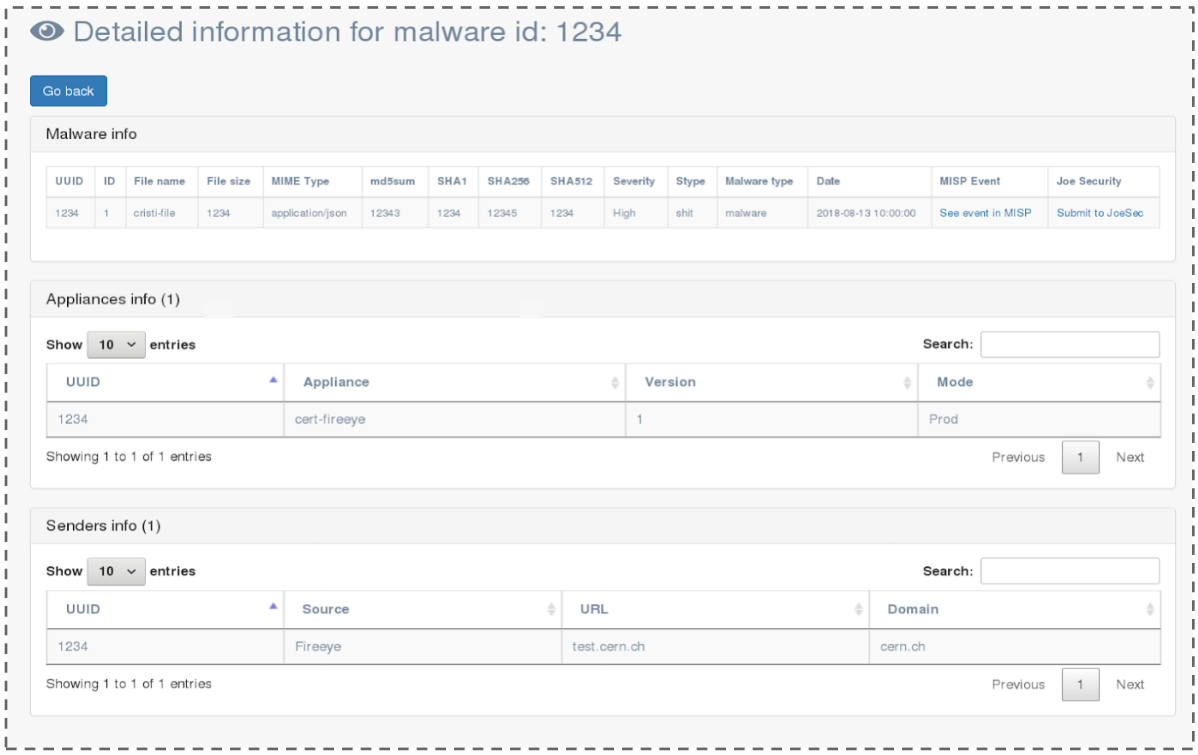


Figure 6: More information can be viewed for a record by clicking on *View* on *List View*.

CURRENT WORKFLOW

HTTP POST Alerts from FireEye EX appliances can be challenging to process by only taking into account the malware type as it is hard to distinguish which malicious alert can contain malware samples and which are without attachment. In the course of this project, 10 different Malware Types were seen with only 3 of them containing malware artefacts.

Malware type names can be random and seldom unseen. With no way of deciding if an alert can contain malware artefacts, the daemon is made to be generic by adopting the following workflow (see Appendix 4 for the malware type list):

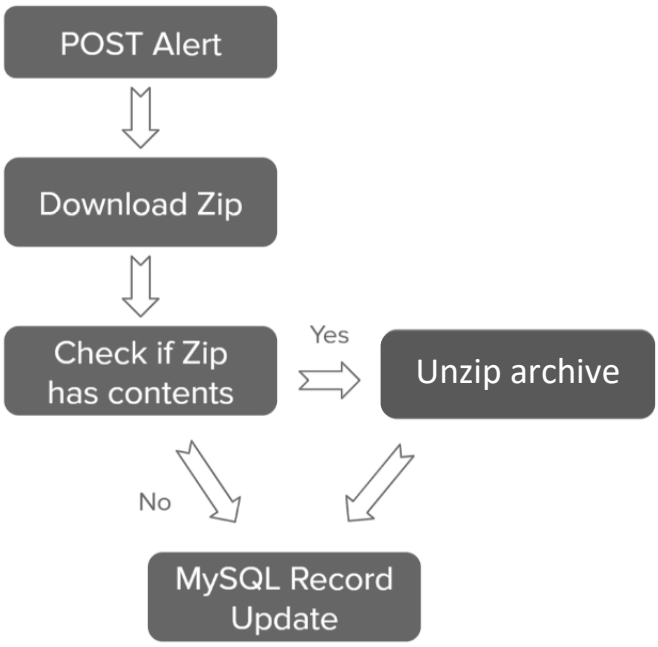


Figure 7: Workflow diagram for python daemon.

- Irrespective of malware type, an API call is being made to the FireEye EX appliances to retrieve the malware content for each HTTP POST Alert;
- The archive is then checked for content inside. If the archive contains any file, the archive is unzipped, and the contents are saved in an output directory as *SHA256_FileSize;*
- Another problem is to recognize which file inside the zip is a malware artefact as they are named randomly too. To solve this, SHA256 for each file is matched with the SHA256 hash received in HTTP POST alert;

- When the hash matches, the malware artefact is renamed to *MalwareFile* so it can be easily accessed for future use without storing each malware file name separately;
- SQL records are inserted for each alert even if it the zip doesn't contain any files.

This way there is no need to hardcode for each malware type and change the code accordingly every time a new malware type is seen.

OPTIMISED DAEMON WORKFLOW

The daemon used in M.A.M. is entirely written in Python using BaseHTTPServer and SimpleHTTPServer to host a basic web server which is primarily designed to listen to HTTP POST Alerts from the FireEye EX appliances and perform a required set of tasks.

There are certain steps taken to reduce the number of API calls made to the FireEye Cloud API. To access the FireEye API, a secure communication session must be maintained by authenticating the device with a FireEye API token which, according to the documentation, expires after 15 minutes of inactivity. The authentication workflow between the FireEye API and the Python daemon follows the following steps:

- Whenever an HTTP POST Alert is received, the daemon checks which appliance sent the alert and then sends an authentication request to the FireEye API for that particular appliance. The authentication request is sent with the username and password;
- If the authentication succeeds, the FireEye API replies with a *FireEye API Token* which is then used to maintain a secure session.

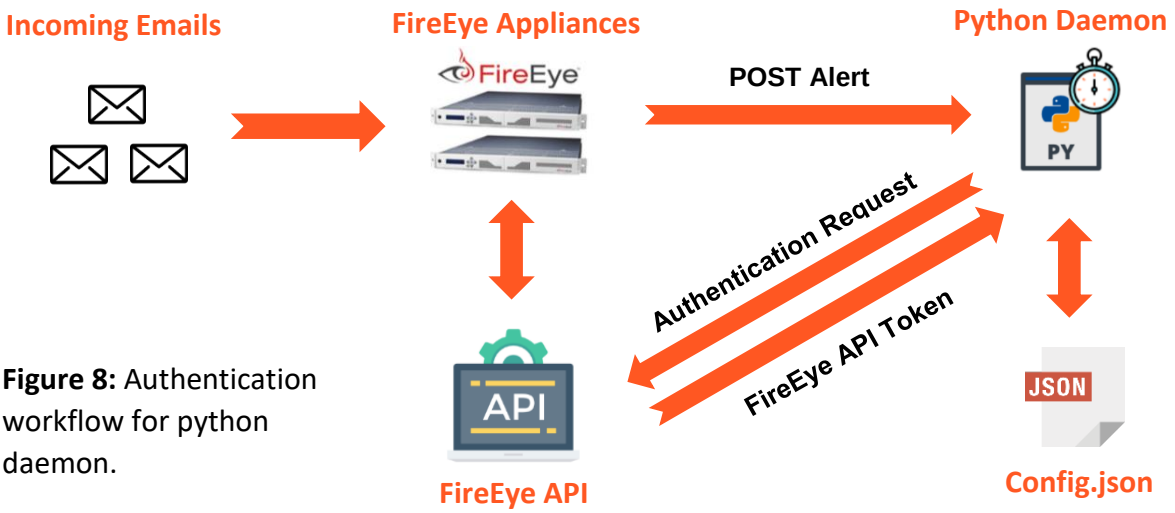


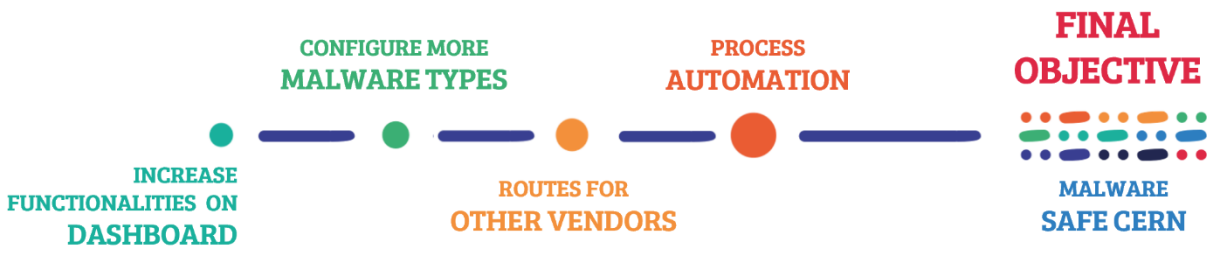
Figure 8: Authentication workflow for python daemon.

As seen from past results, there are sometimes waves of malicious email which makes it highly inconvenient and costly to authenticate and initiate a new secure session with every alert when a token can survive for only 15 minutes. Therefore, to overcome this, when the FireEye API Token is retrieved, it is saved in *Config.json.*

Now when a new alert is received, the token is checked for validity. If the token is still valid, it can be directly used without requesting for a new token by going through whole authentication process. This reduces the number of API calls.

FUTURE WORK

While M.A.M.'s first prototype is complete and running, there is scope for improvements and additions to the project:



1. **Increase functionalities on Dashboard –**
   M.A.M. is equipped with a very basic dashboard made entirely with FLASK. While the dashboard is enough to complete the desired tasks, more functionalities can be added including:
   - A statistics area to show the count of malwares detected in a day, week, month, etc;
   - Another statistics area to show the count of Joe Security API calls exhausted per month and per day and how many are left respectively;
   - An HTML view for the HTTP POST Alert JSON canvas for every record.

2. **Configure more malware types –**
   As there was no list of malware types provided in the FireEye documentation, a total of only ten malware types were recorded throughout the length of the project out of which only three contained malware artefacts. This poses a problem as new malware types keep originating out of the blue which may or may not contain malware artefacts.

   In short, even though the workflow for the daemon is generic, there is a need to tune it more for a fault tolerant system.

3. **Support for other vendors –**
   Currently the daemon is only configured to listen for alerts coming from FireEye EX appliances. Although the daemon can be easily scaled to listen to more FireEye EX appliances, more routes can be added to listen to other vendors in future.

4. **Process automation –**
   Right now, there is no intelligent way to replace the human actor in the process. But at some point, the whole process can be automated to carry out the whole operation seamlessly without any human interaction.

   To decide which malware artefact is important and interesting for detailed analysing, however, can be a challenging objective.

5. **Daemon downtime –**
   In case of any downtime, there should be an automated process to check the alerts for the downtime period and retrieve all the malware artefacts for that time span. This will help keep a record of all malicious emails detected by FireEye appliances.

6. **Email alerts –**
   In case of a targeted email attack wave, if the number of similar malicious emails exceeds a particular value, email alerts can be sent to the monitoring authorities who can take needed action.

7. **Contacting Joe Security –**
   Although Joe Security Cloud sandbox offers to report in 33 different formats, CERN Computer Security Team is only interested in MISP format. Unfortunately, the quality of the MISP Report generated by Joe Security is subpar industry standards. The report doesn't contain group/compounded objects and have objects like `filename|SHA1` which is just filename concatenated with SHA1.

   Joe Security has to be contacted for the same to improve the report format.

RESULTS AND CONCLUSION

Malware Analysis Management or M.A.M. is a real time daemon persistently running on a dedicated virtual machine which retrieves malware artefacts with each HTTP POST Alert sent by the FireEye EX appliances and maintains SQL records of them. M.A.M., currently, is programmed to provide useful insights and detailed analysis on malware reaching CERN through email traffic and possibly through other means in future.

With M.A.M., data is taken from the HTTP POST Alerts. A database record is being formed and maintained even if the alert contains no malware artefact. These records can be useful in performing analytics and possibly in future can help automate the whole process. Replacing the human actor with an automated process remains a challenge to the project.

M.A.M. is definitely a small but significant incremental step towards a “malware safe” CERN. It started as a Python daemon listening to alerts coming from FireEye EX appliances and eventually integrated with the Joe Security Cloud Sandbox service and the MISP framework. M.A.M. is now an addition to other security and defence strategies deployed at CERN for email security. In conclusion, M.A.M. was successful in completing a major part of project specification and has opened the way to more improvements in CERN’s email security. The project was also successful in recognizing the shortcomings in FireEye EX documentation and Joe Security report formats which can be communicated to the respective companies for future improvements.

With M.A.M., deep analysis of malwares is now just a click away.

PART B:
APPENDIXES

APPENDIX 1: SQL Tables Schema

As explained in the architecture section, database records are saved for each alert even if the alert contains no malware artefact. This data can be useful in future analysis or in case of new features to be added in dashboard. There are four SQL tables used in this project to save various types of data. The information is extracted from the HTTP POST Alert with the UUID as primary and referential key for each table:

- **Malware Info –** Contains the basic information regarding a detected malware such as MIME type, file hashes, file name, file size, severity etc;
- **Appliance Info –** Contains information about which appliance detected the malicious email;
- **Senders Info –** Contains information about the source of the email and related data;
- **Mail Info –** Contains email related information including email subject, queue id etc.

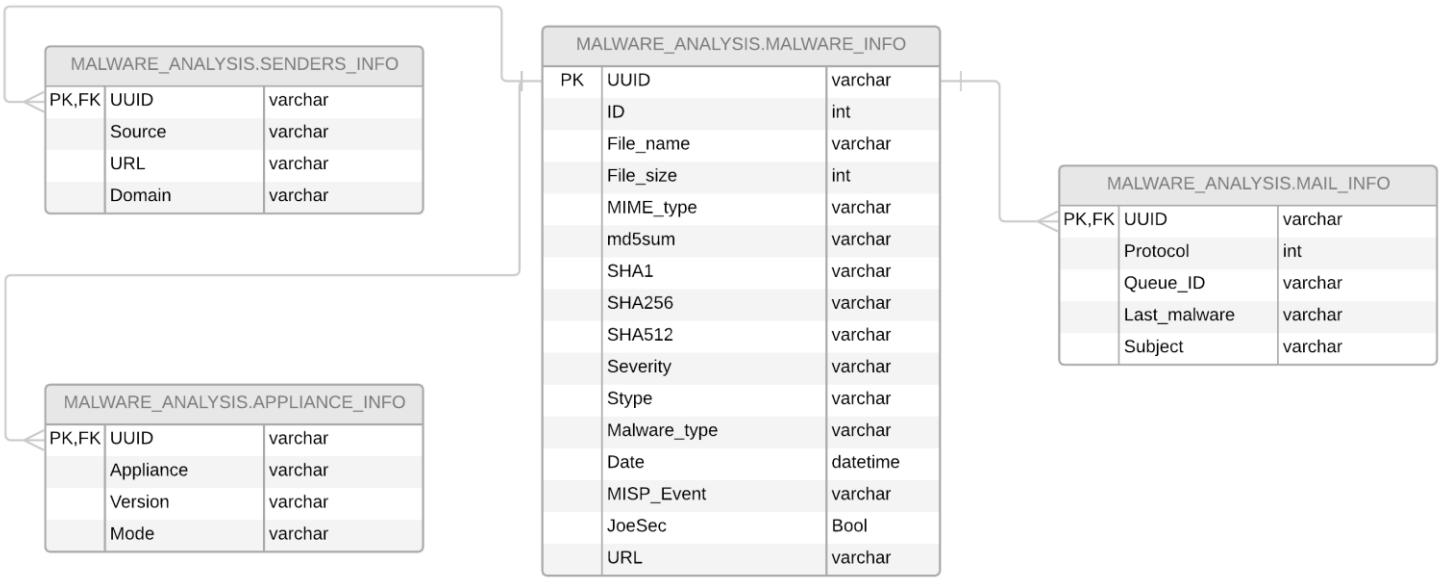


Figure 9: ERD diagram for MySQL Tables in M.A.M.

APPENDIX 2: MISP Events

CERN uses MISP to share security events and keep up to date with security events happening around the world. To publish the threat intelligence gathered, the CERN Computer Security Team creates MISP Events and shares them with the connected communities. A MISP event has two parts:

- **Event details –**

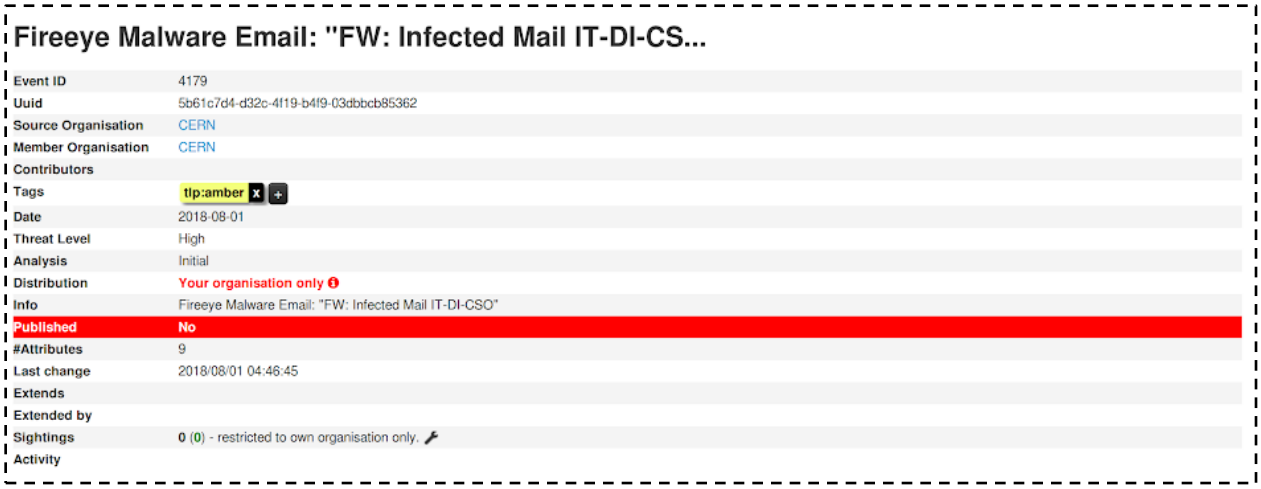


Figure 10: Details for a MISP event.

- **MISP Objects and Attributes –**

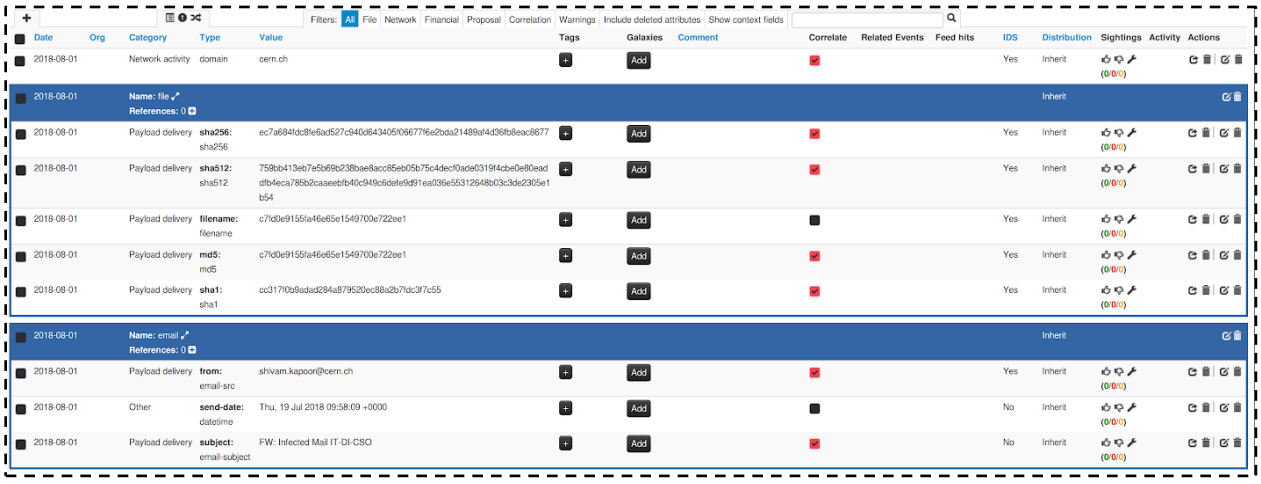


Figure 11: Correlated objects in a MISP event.

In this example, the event is populated with compound objects which makes it easier to correlate between Indicators of Compromise.

APPENDIX 3: Joe Security Reports

Joe Security Sandbox is a very important tool when it comes to performing deep analysis of malware samples. It detonates malware samples in a controlled, sandboxed environment and monitors the behaviour of applications and that of the operating system.

Joe Security provides detailed reports in 33 different formats which allow professionals to develop appropriate defence strategies and protection mechanisms. In M.A.M., the default report is fixed to MISP but can be changed to the following 33 formats by doing a small change in the configuration file*:*

FORMAT	CLASSIFICATION
HTML	Human readable reports
Light HTML	Human readable reports
Executive	Human readable reports
PDF	Human readable reports
Class HTML	Human readable reports
XML	XML reports
Light XML	XML reports
Class XML	XML reports
Cluster XML	XML reports
irXML	XML reports
JSON	JSON reports
JSON Fixed	JSON reports
Light JSON	JSON reports
Light JSON Fixed	JSON reports
irJSON	JSON reports
irJSON Fixed	JSON reports

FORMAT	CLASSIFICATION
Shoots	Other reports
OpenIOC	Other reports
MAEC	Other reports
MISP	Other reports
Graph Reports	Other reports
Mem Strings	Other reports
Bin Strings	Other reports
Sample	Other reports
Cookbook	Other reports
Bins	Other reports
Unpackpe	Other reports
Unpack	Other reports
Ida	Other reports
Pcap	Other reports
Pcap Slim	Other reports
Mem Dumps	Other reports
YARA	Other reports

APPENDIX 4: Malware Types

A malware type is defined by the signature-less detonation chamber ("MVX engine") inside the FireEye EX appliance that detects the behaviour of the malicious attachment.

Malware types can be difficult to process sometimes as seldom they are totally unique and unseen. This makes it hard to distinguish between those alerts which contain malware artefacts, and those that don't. As there is n o documentation available for all types, it becomes a challenge to code a generic way to process the alerts. Also, sometimes, the FireEye API raises unusual alerts containing list of malwares and not a dictionary which was also eventually coded in the project very late as it was never seen before.

The daemon, at the moment, can process all type of alerts and these are the Malware Types seen in the course of this project:

Malware Type	Artefact Present	Possible Description
Malware.archive	Yes	Malicious file attachment. MIME type may include PDF, DOCX etc.
Malware.Binary	Yes	Malicious binary file attachment.
Malware.Binary.FEC2	Yes	Multiple malicious binary file attachments.
Phish.URL	No	Phishing URL included in email.
FE_EMAIL_BODY_BAD_URL_WPADM_PHP	No	Email body contains malicious URL.
FE_EMAIL_MALICIOUS_BODY_501581	No	Malicious email body content.
FE_EMAIL_PHISH_FORMAT_O365_4	No	Email resembles phishing email.
FE_EMAIL_EMOTET_CAMPAIGN_TREND_1	No	Email resembles Emotet campaign emails.
FE_EMAIL_MALURL_FORMAT_HANCKLR_2	No	Email looks like it may contain malicious URLs. (Maybe using Machine Learning)
Phishing.Exfiltrate	No	Phishing link which performs exfiltration of data once opened.

MALWARE ANALYSIS MANAGEMENT

CERN Email Malware Management System