Dockerized MISP
(Malware Information Sharing Platform)

A Degree Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

By

Dídac Cornet Arbós

In partial fulfilment

of the requirements for the degree in

TELECOMMUNICATIONS SYSTEMS ENGINEERING

Advisors:

Eloy Garcia Saro **Josep Pegueroles**

Barcelona, June 2018

Abstract

In an age where most companies offer as much services as they can on the Internet, any protection against cyber threats is not enough. That is why employee awareness campaigns about security best practices, correct network architecture designs, fine-tuned monitoring systems and adequate data treatment policies are of vital importance to companies. Any unexpected production disruption, unavailability of services or information leakage due to a cyberattack can be translated to significant losses at both economical and reputational level.

The aim of this project is to study feasibility of using a malware related information platform for sharing information assets between small, medium and large sized companies as alternative or complement of most advanced and expensive systems in the market in an open source software deployment. A company environment will be simulated in order to integrate Malware Information Sharing Platform with common security systems and study the behavior when a threat is detected.

Resum

En un món on les empresen ofereixen tants serveis com els és posible a través d'Internet, tota protecció contra amenaces de caire cibernètic és poca. És per això que campanyes de concienciació als empleats sobre seguretat i bones pràctiques, bons dissenys d'arquitectura de xarxa, sistemes de monitorització ben ajustats i polítiques de tractament de dades adequades són vitals per a les empreses. Qualsevol aturada inesperada de la cadena de producció, indisponibilitat dels serveis que s'utilitzen dia a dia o fugues d'informació degudes a un atac cibernètic es poden traduir en pèrdues importants tant a nivell econòmic com reputacional.

L'objectiu d'aquest projecte és estudiar la viabilitat d'utilitzar una plataforma d'informació relacionada amb malware i amenaces per compartir informació entre petites, mitjanes i grans empreses com a alternativa o complement als aplicatius més avançats i costosos del mercat mitjançant la utilització de software lliure. És simularà un entorn empresarial per tal d'integrar MISP (Malware Information Sharing Platform) amb altres sistemes de seguretat comuns i estudiar el comportament del sistema davant la detecció d'una amenaça.

Resumen

En una era donde las empresas ofrecen tantos servicios como pueden a través de Internet, toda protección contra amenazas de carácter cibernético es poca. Es por esto que campañas de concienciación a los empleados acerca de seguridad y buenas prácticas, buenos diseños de arquitectura de red, sistemas de monitorización bien ajustados y políticas de tratamiento de datos adecuadas son vitales para las empresas. Cualquier corte inesperado en la cadena de producción, indisponibilidad de los servicios utilizados día a día para el funcionamiento del negocio o fugas de información debidas a un ataque cibernético se pueden traducir en pérdidas importantes tanto a nivel económico como reputacional.

El objetivo de este proyecto es estudiar la viabilidad de utilizar una plataforma de información relacionada con malware y amenazas para compartir información entre pequeñas, medianas y grandes empresas como alternativa o complemento a los aplicativos más avanzados y costosos del mercado mediante la utilización de software libre. Se simulará un entorno empresarial para integrar MISP (Malware Information Sharing Platform) con otras herramientas de seguridad comunes y poder estudiar así el comportamiento del sistema ante la detección de una amenaza.

Acknowledgements

Thanks to all those who gave me help and support for developing this project and focused me in the best direction when problems arised.

Revision history and approval record

Revision	Date	Purpose
0	19/05/2018	Document creation
1	28/06/2018	Document revision

Written by:		Reviewed and approved by:	
Date	June 2018	Date	June 2018
Name	Dídac Cornet	Name	Josep Pegueroles
Position	Project Author	Position	Project Supervisor

Document distribution list

Name	E-mail
Dídac Cornet	didac_c4@hotmail.com
Eloy Garcia	egarciasaro@deloitte.es
Josep Pegueroles	josep.pegueroles@upc.edu

Table of contents

Tabla de contenido

List of figures

List of tables

1. Introduction

Due to rising digitalization of all services into a forced march in order not to lose business opportunities, companies are exposed to cyberattacks that put at risk both economy and reputation of these. Financing budget to protect IT assets often is less than necessary and it means higher risk to become a cyberattack victim that affect companies' interests. Even though large companies have enough resources to protect themselves, there are many others that cannot afford best security solutions in the market.

In this project, a low-cost solution for securing computer networks will be studied. MISP on Docker will be the software used to share information about malware and threats between different entities. In addition, this tool will allow to feed Intrusion Prevention and Detection System with new rules generated based on this shared information.

Once a threat is detected in one company and relevant related information introduced within the MISP community, the most identifying features of a threat will be shared with the rest of contributors allowing a fast security system update to protect the networks.

With the aim of implementing MISP software in companies, an image has been adapted to allow a fast, non-invasive, effective and easy deployment of MISP in any network.

System will be tested simulating network behavior in front of a threat detection in a company and extracting statistics about successfully detected or not detected attacks by IDS/IPS in order to demonstrate the utility and reliability of the proposed system.

All monitoring will be done with a Splunk SIEM acting as a Security Operation Center, centralizing alerts and showing system's performance.

1.1. Objectives

Due to complexity of the project, different objectives have been defined to face smaller goals easier to tackle.

These are the main goals to achieve:

- Create a MISP image based on Docker which will allow an easy and fast deployment of a MISP instance within a corporate network.
- Simulate a corporate network as testing environment.
- Deploy a MISP instance in corporate network using the created image.
- Connect the MISP instance in corporate network to a central instance.
- Integrate client's MISP instance with Suricata Intrusion Detection and Prevention System.
- Simulate cyberattacks generating random alerts within corporate network.
- Set up a Security Operations Center for monitoring client's network and alerts generated within it using Splunk as a SIEM.
- Test system in an attack/infection scenario and measure system performance.

The achievement of all these objectives will lead to demonstrate the feasibility of using the proposed security system for intrusion detection and prevention.

1.2. Scope

The project scope contemplates the development and customization of a MISP dockerfile for being deployed on Docker, design a standard corporate network with its firewall, IPS/IDS and employee's laptops subnet, integrate MISP with Suricata Intrusion Prevention System on corporate network, monitor alerts with a monitoring tool such as Splunk to be used a Security Operations Center (SOC), study whole system behavior in case of incident within the simulated client's network and obtain telemetry of number of detected attacks in front of generated ones.

Both network and virtual machines used in the project have been set up with the well-known virtualization software Oracle VirtualBox.

1.3. Project timeline

Project timeline has been defined focusing to objectives consecution. Separating the objectives has allowed developing and testing each part of the project individually in parallel without affecting the other parts and integrating them when all parts have been completed.

In the initial phase, deep research has been done in order to familiarize and gain knowledge about software used as Docker, MISP, Splunk, pfSense and Suricata. It has been essential to spent time on it due to some automations have been done customizing root program files and a modification without complete understanding could have resulted in unexpected future errors.

After this research phase, development stage has started. This stage is about installing all machines and software and learning to use them. Also, defining the network architecture and connections has been done during this phase.

Once all machines are on the system the customization part has started. It consisted in developing scripts for automate the updating of IPS rules, generate alerts and forward logs and synchronize MISP instances.

The final part consisted in deploy Splunk, a monitoring tool, to act as Security Operations Center centralizing all alerts generated on the system using log forwarders on the systems of interest and creating dashboards to visualize alerts and hardware performance to obtain the final conclusions.

1.3.1. Work Plan

Below, followed work plan is detailed with each main task to achieve and the time duration.

PROJECT WORK PLAN			
TASK	**Start date**	**Duration(days)**	**End date**
Decide Linux distribution to allocate MISP image	15/02/2018	5	20/02/2018
Create MISP dockerfile	21/02/2018	30	23/03/2018
Create MISP OVA (Open Virtual Appliance)	23/03/2018	7	30/03/2018
Simulate a company network with FW, IDS/IPS	16/03/2018	30	15/04/2018
MISP integration in company network	16/04/2018	30	16/05/2018
IPS/IDS rules generation and system automation	16/04/2018	30	16/05/2018
Testing system performance and tunning	17/05/2018	30	16/06/2018

Table 1. Project Work Plan

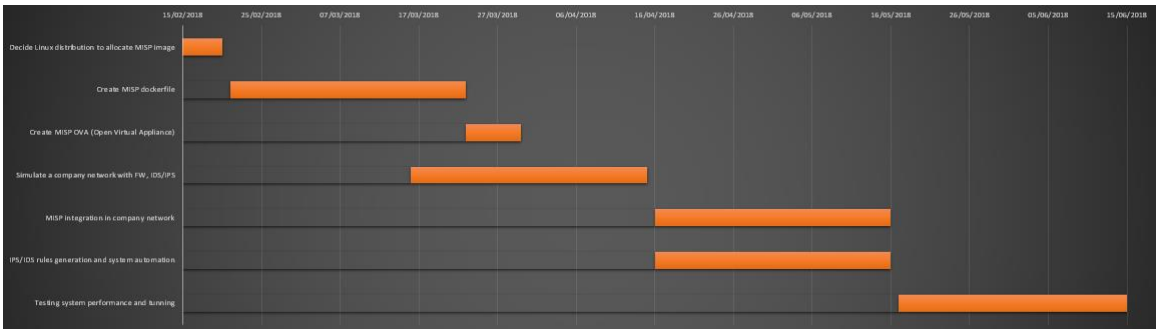


Figure 1. Project timeline – Gantt Diagram

1.3.2. Unforeseen circumstances

During the project, some problems raised. Next, some of them are listed:

- Major complexity in creation and configuration of MISP Dockerfile for Docker. Some errors in the installation process appeared due to the automation of installation process editing root program files.

- SNORT IPS/IDS on pfSense product limitations.
  Firstly, the software decided to be used as IPS was SNORT because more documentation can be found than for Suricata but the implementation of SNORT in pfSense doesn't allow to drop packets in real-time. The problem was detected once the program was installed and the offenses where detected and alerts generated but not blocked. It suposed to uninstall the program and use Suricata after spending some time learning to use the program and creating custom rules.

- MISP integration in company network.
  As client network is isolated from MISP Central, some firewall configurations had to be done to connect both platforms and that was not previously contemplated.

- For monitoring and data acquisition of the Suricata IPS/IDS system and the whole system, a Splunk server has been deployed that was not contemplated in the initial project plan.

2. State of the art

This section provides background knowledge about technologies used in the project.

2.1. Virtualized environment

The envionment virtualization process corresponds to creation of a virtual version of a technologic resource like an operating system, a hardware device, a computer, a network, etc.

Thus, using virtualization software in a physical machine (host) is posible to virtualize more machines (guests) and network devices in order to simulate a more complex infrastructure in a single machine.

The virtualization environment used in this project is Oracle VirtualBox.

This software provides necessary tools to simulate multiple networks and hosts properly isolated or communicated depending on needs.

Nowadays, other similar softwares are availilable like vmWare.

2.2. Docker

Docker is a development platform that allows complex applications creation separating the services required for these aplications in isolated containers. Thus, ligthweight, efficient and autosuficient applications can be developed and its behaivour is independent of the environment where they are executed. This allows corporations to update or install services in production environments in an agile way and minimizing risks. Another great advantage is that aplications are composed by continers that even though they don't depend on other conainers, they can be linked in order to if one container fails, coordinated restart of the rest of containers can be done, providing great reliability to these applications.

Unlike virtual machines, Docker doesn't virtualize all hardware and software needed to create a machine, but it uses the lower system requirements to start the services using the resources from the host where Docker is installed.

Docker containers are created from images. The file that defines the content of an image in Docker is Dockerfile. These images are software packages. Once the images are compiled they turn into containers.

Use of Docker applications is in raise last years in all kind of business.

2.3. MISP

MISP is the acronym for the open source software "Malware Information Sharing Platform". MISP is software designed for sharing, storing and correlating events related to cybersecurity threats.

Its main functionality is sharing intelligence information assets between organizations using this software, allowing a better, deeper and faster intrusion detection and prevention implementations and solutions. It also incorporates the feature for generating rules for IPS and IDS systems.

MISP is based in events that contain attributes identifying malware behaviors or significant features that describe specific attacks.

Currently, MISP is used to centralize all malware related information in a company serving as a database. In this project, capabilities to integrate MISP with IPS security systems will be explored.

2.4. Firewall

A firewall is a security device designed to allow or deny communications between multiple networks. It acts as a filter controlling all incoming and outgoing communicactions from a network to another basing its decisions on predefined rules.

Until now, firewall rules were based in two main criterias: filtering on layer 3 in OSI model (IP packet filtering) and filtering on layer 4 in OSI model (filters based on source/destination ports of communications or MAC addresses).

Currently, commercially named next-generation firewalls allow filtering on layer 7 in OSI model, basing its filters at application level.

The firewall used in this Project is pfSense, type software.

2.5. Intrusion Prevention and Detection Systems (IPS/IDS)

Intrusion Prevention and Detection Systems are security tools that monitor the events occurred in a system or a network searching for attempts to compromise the security of the system or network monitored.

IPS/IDS are based in predefined patterns that trigger alerts or drop traffic when a suspicious activity is detected.

Main difference between Intrusion Prevention Systems and Intrusion Detection Systems is that Prevention Systems can avoid an incident thanks to its active response capability for blocking traffic, while Intrusion Detection Systems are passive systems that can only alert when a suspicious event has been detected and they can not take action to stop an attack.

They are classified into two large groups:

- HIPS/HIDS (Host Intrusion Prevention/Detection Systems)

  They protect a unique server or host. They collect information in the system like files, logs, resources, etc, for subsequent analysis in search of coincidences with the anomalous patterns.

- NIPS/NIDS (Network Intrusion Prevention/Detection Systems)

  These systems protect systems network based. They analyze packets through the network in search of patterns that can suppouse or indicate a possible attack.

  Normally they work in real time and at TCP/IP level even though they can work at application level.

In this project, the IPS used is Suricata in Inline Mode, permitting reaction against detected threats.

2.6. Indicators of Compromise – IoCs

Indicators of compromise are extracted from a computer forensic investigation after an incident and are used to identify potencial malicious activity in a system or a network. They are the most significant features that characterize certain malicious software.

IOCs are introduced to MISP in the form of events and can be shared between organizations and used for generating intelligent rules for IPS and IDS systems.

Although there is currently no generally accepted standard data format for security teams to share Indicators of Compromise (IoCs) most used formats are XML and STIX.

In this project, IoCs are manually created in MISP platform in events form, which can be exported in multiple formats.

3. Methodology / Project development

Methodology followed during the project development is based on the achievement of specific objectives as part of the main purpose: demonstrate viability of integrating dockerized MISP instance with an Intrusion Prevention System.

The main milestones are:

- Create a MISP image based on Docker which will allow an easy and fast deployment of a MISP instance within a corporate network.
- Simulate a corporate network as testing environment.
- Deploy a MISP instance in corporate network using the created image.
- Connect the child MISP instance in corporate network to a central instance.
- Integrate client's MISP instance with Suricata Intrusion Detection and Prevention System.
- Simulate cyberthreats generating random alerts within corporate network.
- Set up a Security Operations Center for monitoring its network statistics and alerts generated within it using Splunk as a SIEM.
- Test system in an attack/infection scenario and measure system performance.

In this chapter all software and network designs as well as custom scripts developed to add automation functionalities in all the process will be explained.

3.1. Architecture overview

In order to integrate MISP within an organization, a standard corporate network has been simulated.

Next image shows all deployed servers, networks and security devices:

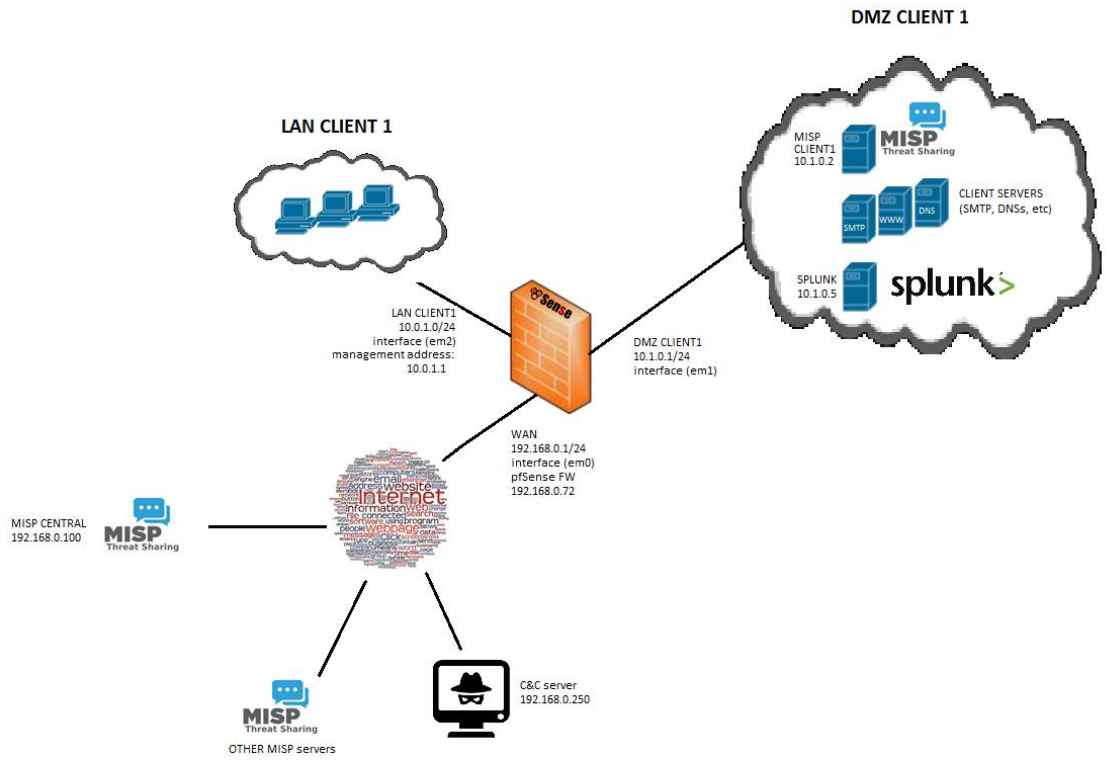


Figure 2. System architecture overview

On the left side, there is MISP-Central. This is the main MISP node where all information is centralized and distributed to the rest of clients' MISP instances that make up the community.

On the right side, there is the simulated corporate network. The elements that make up this network are:

- Firewall: software based pfSense firewall.
- IPS: Intrusion Prevention System based on Suricata software solution.
- LAN: subnet containing employee desktops.
- DMZ: subnet containing company internal servers for file storage, mail exchange, web servers, etc., and a MISP instance called MISP-Client1.

This MISP instance deployed in the organization's network has been isolated from the LAN and it can only be accessed by certain system administrators within the LAN, from IPS system and from MISP-Central from outside the network. This configuration has been done in client's firewall. Likewise, outgoing connections from MISP-Client1 to the Internet are only enabled to communicate with MISP-Central.

This entire configuration is done with firewall rules, NAT routing and configuring network interfaces for each host in VirtualBox.

Other elements as OTHER MISP instances represent the possibility to scale MISP community and the Command & Control server (C&C) was included in order to detect connections to a supposed malicious server.

A static routing has been established, ensuring the appropriate approach for a secure network perspective except for the LAN subnet, where DHCP (Dynamic Host Configuration Protocol) in pfSense LAN interface provides and assign IP addresses to employees laptops.

The corporate network consists of following subnets and hosts:

SUBNET	IP	MASK
WAN	192.168.0.72	255.255.255.0
LAN	10.0.1.1	255.255.255.0
DMZ	10.1.0.1	255.255.255.0

Table 2. Corporate network subnets

3.1.1. Resouces

These are all hosts and servers used during project development:

GUEST	OS	INTERFACE	IP	GATEWAY	USER/S
MISP-Central	Debian 8 (x64)	Eth0	192.168.0.100	192.168.0.1	root misp
pfSense FW	FreeBSD 11.1 - pfSense 2.4.2-RELEASE-p1 (x64)	WAN	192.168.0.72		admin
		LAN	10.0.1.1	192.168.0.1	
		DMZ	10.1.0.1		
MISP-Client1	Debian 8 (x64)		10.1.0.2	10.1.0.1	root misp
Splunk Client1	Ubuntu 17.10 (x64)	DMZ	10.1.0.5	10.1.0.1	root splunk
Desktop X Client1	Ubuntu 17.10 (x64)	LAN	10.0.1.[2-255]*	10.0.1.1	root desktopXclient1*
Ostinato Traffic Generator	Ubuntu 17.10 (x64)	LAN	10.0.1.222	10.0.1.1	root ostinato
C&C server	Ubuntu 17.10 (x64)	WAN	192.168.0.250	192.168.0.1	httpserver

Table 3. Resources

[*]: This host changes its IP address because is used to simulate multiple employee desktops to reduce the need of resources that would suppose create a virtual machine for each employee connected on LAN in further sections in the project.

3.1.2. Network configuration

Because security devices and host addresses on client's network side have been set statically, is necessary to define VLANs where each machine is connected. VirtualBox allow network interface configuration in order to reproduce private networks through the option "Internal Network" in network settings contextual menu. Multiple internal networks are allowed for a single host and each of these networks has its own name and is isolated from the rest and from physical host where VirtualBox software is running. To allow connections to the physical host, which interface allows connections to the Internet, another type of connection has been set up in "Bridged Adapter" mode.

Next figure shows the interfaces configured for pfSense firewall box (1 Bridged Adapter for WAN and 2 Internal Networks for DMZ_client1 and LAN_client1):

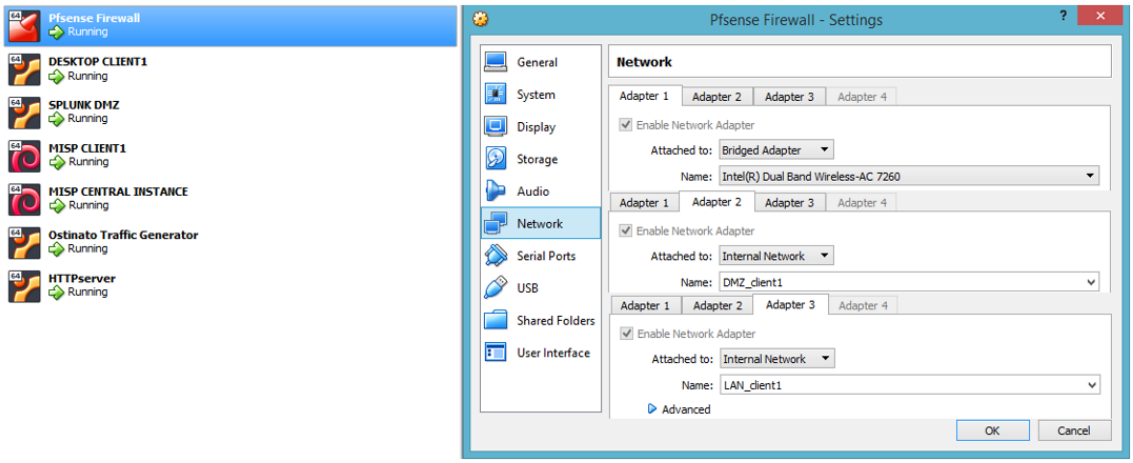


Figure 3. pfSense NIC configuration in VirtualBox

This is the configuration for DESKTOP CLIENT1, only connected to LAN:

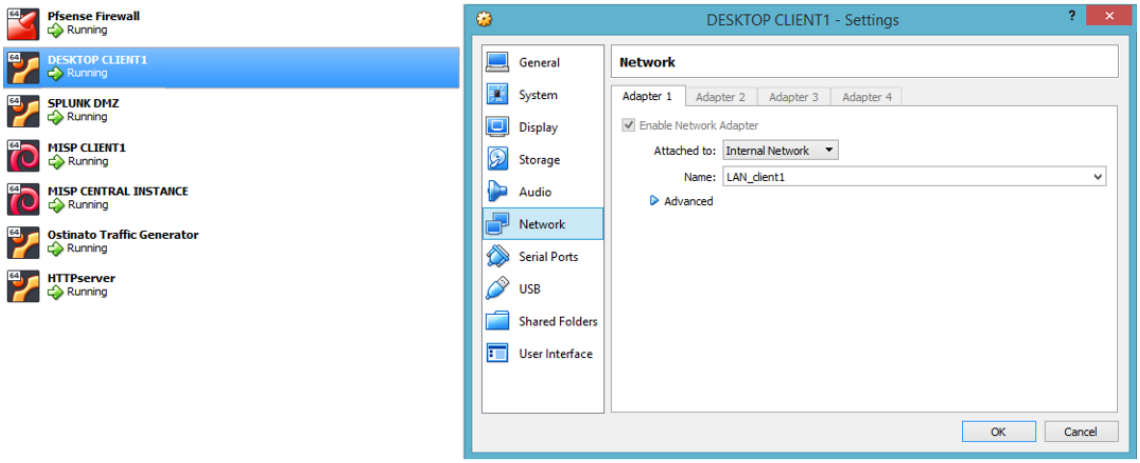


Figure 4. Desktop Client1 NIC configuration in VirtualBox

With this configuration, any host in LAN_client1, if connecting to the Internet will be redirected from the LAN_client1 gatway to the WAN in the pfSense firewall box. At this point is when traffic will be inspected for anomalous activity or threats by Intrusion Prevention and Detection System implemented with Suricata. The same routing and packet inspection is done in the DMZ_client1 inteface for hosts connected on it.

For Splunk and MISP-Client1, only one network adapter has been configured to connect it to DMZ (Demilitarized Zone), where usually corporate servers are located:

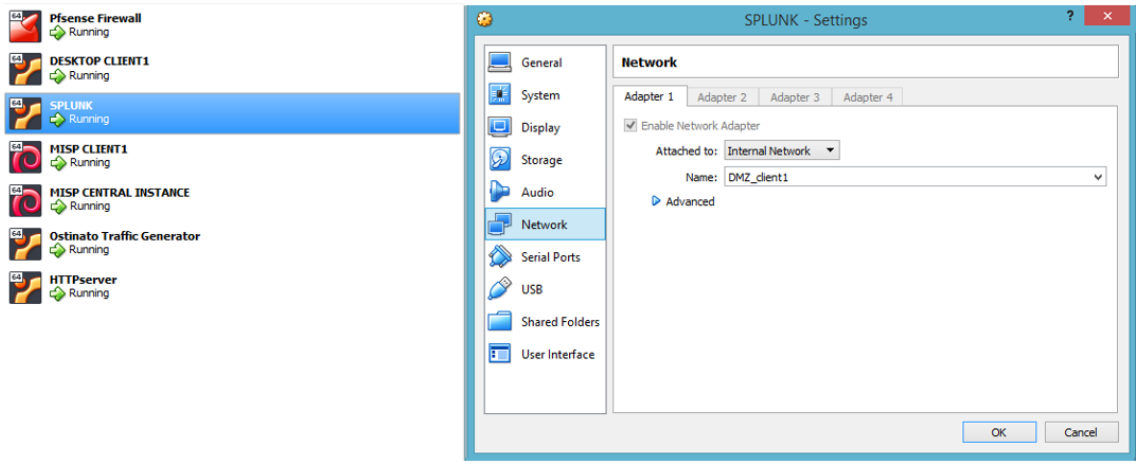


Figure 5. Splunk NIC configuration in VirtualBox

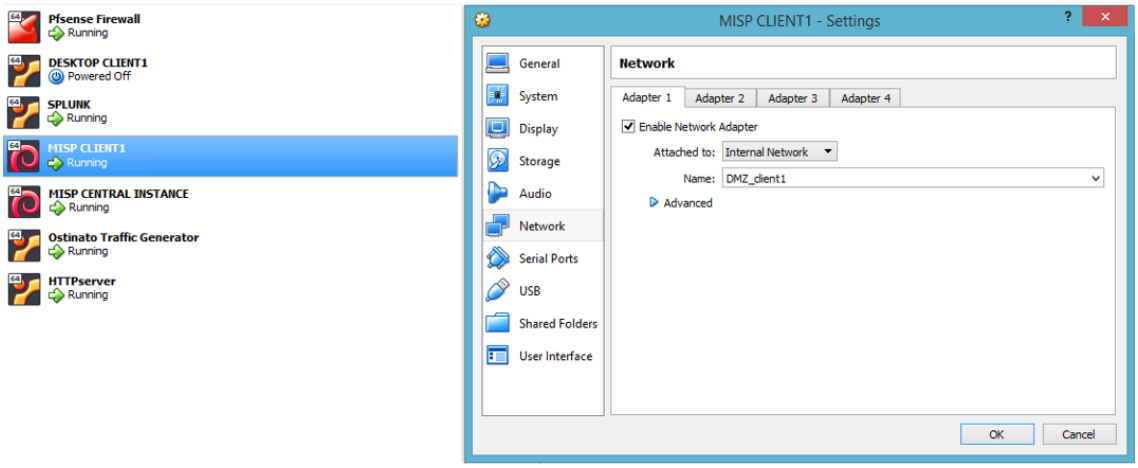


Figure 6. MISP-Client1 NIC configuration in VirtualBox

For MISP-Central instance, a bridge network adapter has been configured in order to represent it is connected to the Internet:

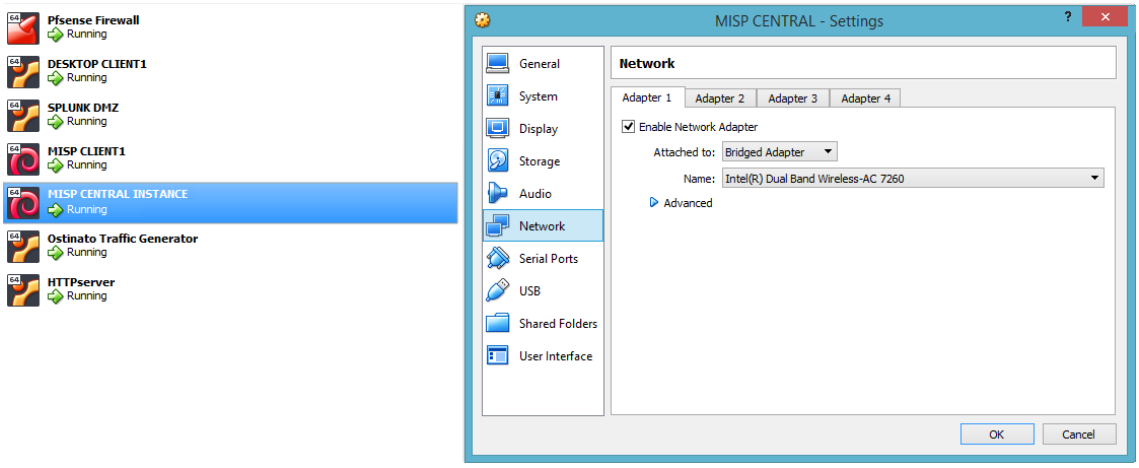


Figure 7. MISP-Central NIC configuration in VirtualBox

Ostinato traffic generator has been connected to LAN_client1 in order to generate traffic to the pfSense box from this network for simulating high traffic load.

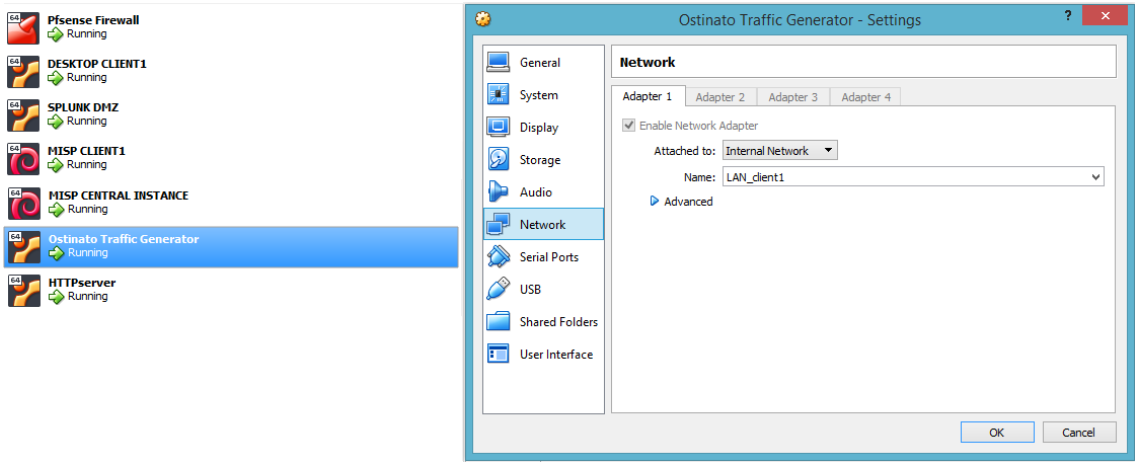


Figure 8. Ostinato Traffic Generator NIC configuration in VirtualBox

HTTP server has been configured with bridge network adapter to represent it is connected to the Internet. This server will be used for simulating a malicious Command and Control server for testing purposes.

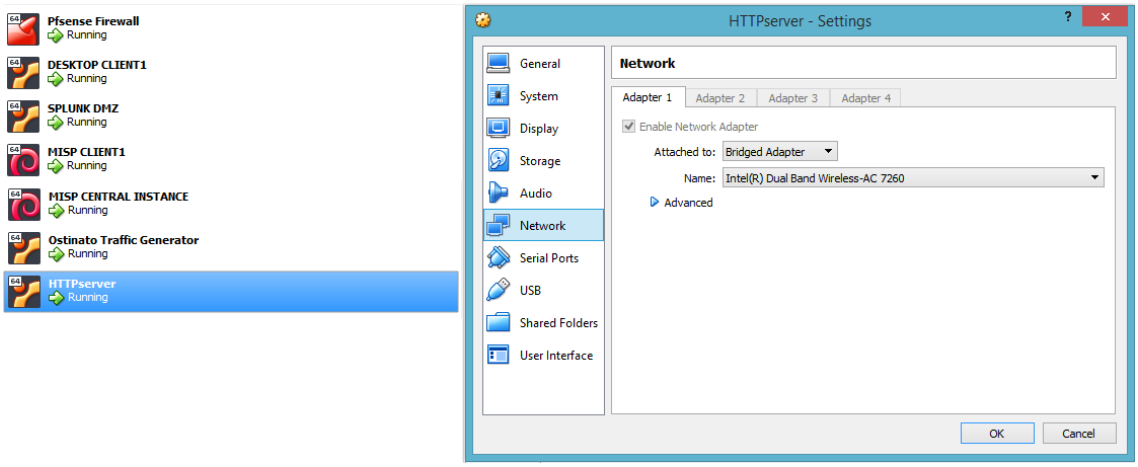


Figure 9. HTTPserver NIC configuration in VirtualBox

3.2. MISP

MISP or Malware Information Sharing Platform is developed as free software by a group of developers from CIRCL (Computer Incident Response Center Luxembourg) but also the Belgian Ministry of Defence and NATO NCIRC.

This platform allows for sharing, storing and correlating Indicators of Compromise of cyber attacks. Multiple organizations can share information trhough MISP to help improving counter-mesasures used against detected attacks and set-up preventive actions and detection.

Among the most outstanding features, are:

- Facilitate the storage of technical and non-techical information about malware and attacks.
- Create relations between malware and its attributes.
- Store data in a structured format (allowing automated use of the database to feed detection systems or forensic tools).
- Generate rules for Network Intrusion Detection Systems (NIDSs).

The spirit of MISP is to collaborate between organizations to share knowledge in the form of indicators of compromise and observable data.

3.2.1. MISP Dockerfile

With the aim of having an easy and fast deployment method, an image of MISP for Docker has been created.

In Docker, MISP is composed by two containers: misp and db.

Docker-compose is the tool used to link conainters and will be used to communicate the MISP platform running in misp container with the database in mysql container.

For developing this part, the host allocating Docker is a Debian 8 Linux server.

As starting point, MISP image from official github repository has been downloaded and has been adapted to separate it in two conainers: misp and mysql, creating a multiconainer application. This has been done due to the Docker best practices state that decoupling applications into multiple containers make it easier to scale horizontally and reuse containers. In case the database is running out of free space, another container can be used to increase the memory size. Also, for load balancing reasons this is the recommended implementation.

To build and execute the application tool docker-compose is need. This will allow restarting automatically whole application in case of one container fails.

The content of docker-compose.yaml defines application container relations between them and the relations to the host where Docker is running. This is the content of this file:

```
version: '2'

services:

  db:

    image: mysql:5.6

    restart: always

    volumes:

      - /var/lib/misp-db:/var/lib/mysql

      - ./db/MYSQL.sql:/tmp/MYSQL.sql

      - ./db/feed.sh:/docker-entrypoint-initdb.d/feed.sh
```

```
    env_file:
      - ./env.txt
  misp:
    build: ./misp
    restart: always
    volumes:
      - /dev/urandom:/dev/random
    env_file:
      - ./env.txt
    environment:
      - SALT=amvyuLcCgfjD7UpziwnD1a03PLm4Ea7rpRt5KjO0ZDYa
    ports:
      - "443:443"
    depends_on:
      - db
```

As can be seen, there are the two conainers making up the application inside services section.

These are the main clauses appearing in this file:

- services: in this section is placed each container that will be executed. We have two conainers: db and misp.
    - misp contains the web application
    - db contains the database
- image: specifies the image file from which the conainer will be built. Has to be an image available on Docker's GitHub repository.
- build: is the path to the creation context where the configuration options are defined and will be applicated when booting the image of a container.

For misp container, relative path ./misp point to the creation context of MISP image. This path contains the Dockerfile used to create the image.

For db container, build is no needed because image states to the image will be pulled from online repository and it will contain its dockerfile.

- volumes: this clause maps directories from containers to the host executing Docker (local_directory:container_directory). This way, is posible to store or make data persistent due to information contained within a container only exist within it and all data is deleted when conainer is stopped.
- env_file: this variable points to a file path with environment variables accesible from all conainers composing the aplication. These variables are accesible when building the container and creating the image and during the execution of it.
- ports: is used to map ports from host allocating Docker to containers.
- restart: this clause is optional and will restart a conainer in case it stops.
- depends_on: define a dependency with other services. If container which is depending to is stoped, this conainer also will be stopped until the other is up again.

3.2.2. MISP creation context

The creation context contains all necessary files to build a container. As seen before, a Dockerfile file is needed and it defines all content in an image and all processes to be executed when building and booting the container.

MISP creation context is composed by a Dockerfile file (Dockerfile), an environment variables file (env.txt) and an executable bash file (run.sh).

During container creation, Dockerfile will be responsible to install all necessary paquets for correct installation and functioning of MISP platform on a Linux Ubuntu OS, create the configuration files for the apache server allocating MISP web interface and map the files located on host to the container.

Environment variables file contains all variables that are need during the installation process:

- MYSQL_ROOT_PASSWORD: MySQL password for user root

- MYSQL_MISP_PASSWORD: MySQL password for user misp
- MISP_ADMIN_EMAIL: mail address used for message encryption GPG
- MISP_ADMIN_PASSPHRASE: password associated to GPG key
- MISP_BASE_URL: MISP access URL

Moreover, during image building, Dockerfile will call the script run.sh, responsible of SSL certificates generation and pass the configuration parameters contained in env.txt to the container.

An extra file has been added for entering initial setup parameters needed by misp creation context from an interactive shell istead of manipulate these three explained files directly. This file is misp_install script, explained later in this section.

In first deployment of the instance, all packets specified in dockerfile have to be downloaded so Internet connection from the development host is needed.

After this first boot, an OVA file has been generated from VirtualBox with option Export Virtual Appliance. An Open Virtualized Appliance (OVA) is a package that contains a virtual machin for deployment. It contains different files:

- .vmdk: virtual machin disk files
- .mf: manifest file
- .ovf: meaning Open Virtualized Format, is an open standard for packaging and distributing virtualized services

This OVA contains all needed packages for installation already downloaded, and Docker images created, and will be used for deploying MISP instances with no need of Internet connection. At same time, this will reduce the amount of time needed for deploying a MISP instance in a production environment as well as no need of communications modifications to deploy it.

3.2.3. MySQL creation context

MySQL creation context provides the necessary files for container creation of MySQL 5.7 database. All information contained in MISP platform will be stored in this container and

mapped at the same time to the host allocating the application in order to make persistent data introduced in MISP.

3.2.4. MISP installer

To facilitate the intallation of a MISP instance to any corporation, the installation process has been automated as much as possible. Bash script named install_misp has been saved in /bin directory for initial application configuration and boot.

Once we power on the MISP.ova created in a virtualized environment as VirtualBox or vmWare, typing misp_installation, installation process starts. The wizard asks the parameters to configure the instance like IP address for allocate MISP, information to generate SSL certificate for the server, email, passwords for MySQL database, listening ports and others. These parameters are placed in environment variables file (env.txt) for feed docker-compose.yaml and run.sh when booting the instance and place them into configuration files.

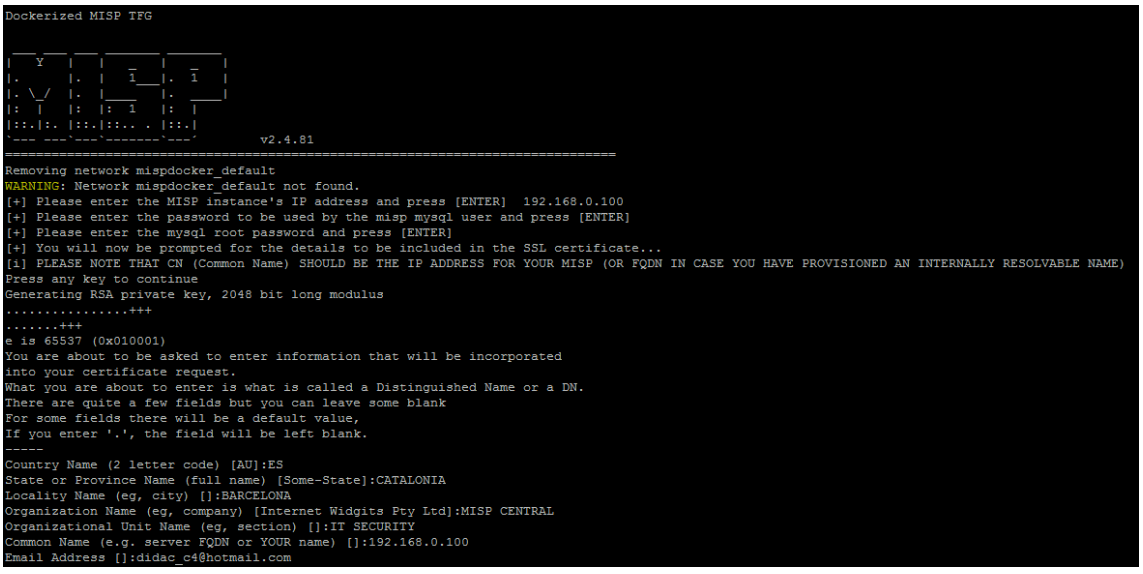


Figure 10. MISP installation wizard

Once the wizard is completed, the instance will be booted and can be accessed from a browser in the specified address. For MISP-Central instance, the site is accessible in https://192.168.0.100.

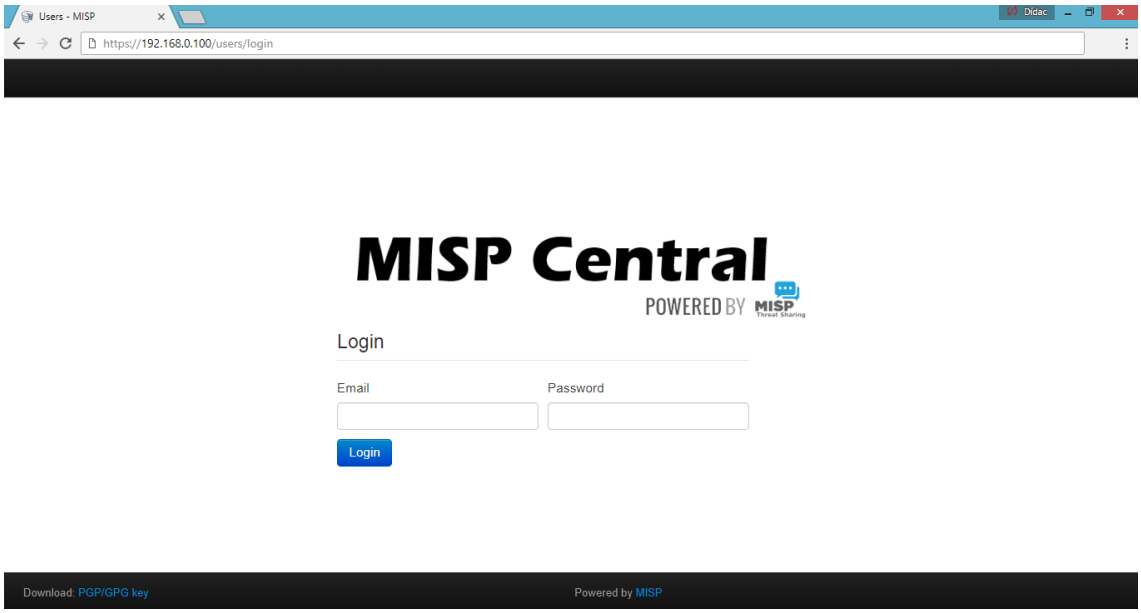


Figure 11. MISP-Central login page

When first log in, user will be prompted to change default credentials for new ones.

3.2.5. MISP SSL certificates

Server certificates are generated during the installation process and are self-signed. This type of certs is not validated with any third. In a public scenario, certificates signed by a CertificateAuthority should be used to grant security. The CAs issues digital certificates that certify the ownership of a public key by the named subject of the certificate.

The installation wizard prompt user to introduce all information needed for certificate generation.

For MISP-CENTRAL instance, the certificate is generated using OpenSSL software with next values:

- Country Name (2 letter code): ES
- State or Province Name: CATALONIA
- Locality Name: Barcelona
- Organization Name: MISP CENTRAL
- Organizational Unit Name: CYBERSECURITY
- Common Name: 192.168.0.100
- Email Address: didac_c4@hotmail.com

These values are used by misp_install script to generate the certificate using the command:

```
$openssl req -x509 -subj "/C=ES /C=CATALONIA /L=Barcelona/O=MISP CENTRAL

/OU=CIBERSECURITY /CN=192.168.0.100" -nodes -days 3650 -newkey rsa: 2048 –

keyout misp.key -out misp.crt –batch
```

This is the self-signed certificate for MISP CENTRAL server:

Figure 12. MISP CENTRAL SSL certificate

This is the MISP Client1 certificate:

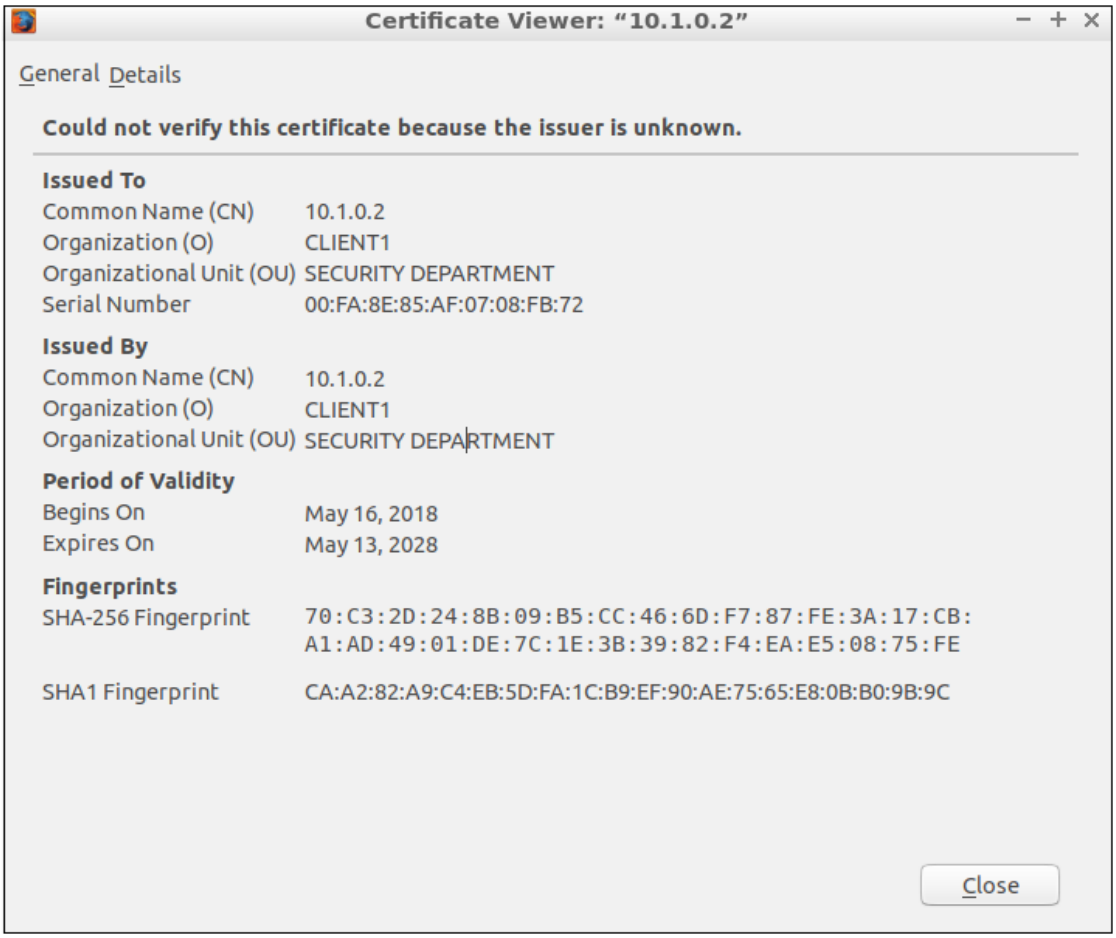


Figure 13. MISP CLIENT1 SSL certificate

Certificates are used to authenticate MISP servers between them, explained in the next section.

3.2.6. Connecting MISP instances

In order to connect multiple MISP instances between them, once the MISP-Central and MISP-Client1 have been deployed, is necessary to set up the connection for allow communication between them.

In the interest to push and pull events between instances, an authentication key is shared. This key will be obtained creating a synchronization user in MISP-Central instance to allow connections from MISP-Client1. Morover, client certificates have to be imported.

The following picture shows the concept how different MISP instances could tie together

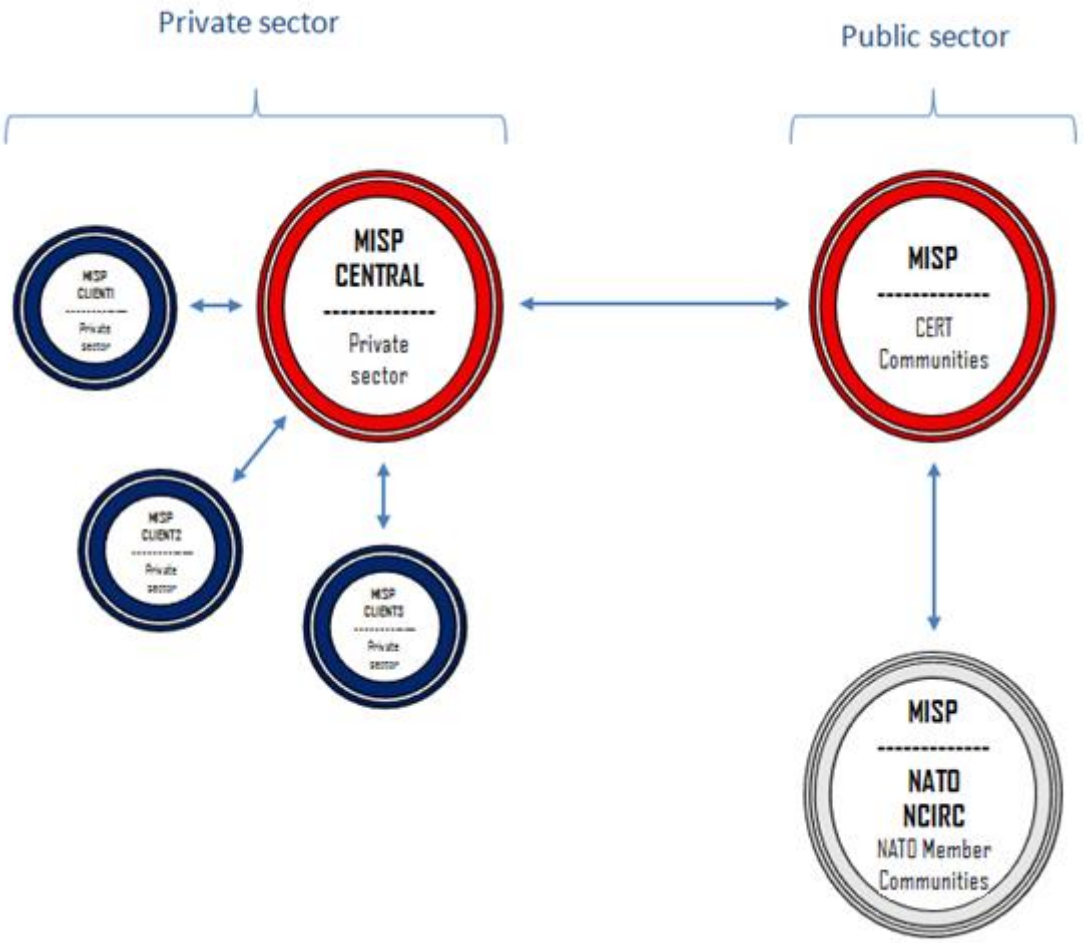


Figure 14. MISP network topology

In the private sector side are private companies using a MISP instance to centralize its information cyber security assets, normally used by IT security team and private users who collaborate with communities and/or take advantage of them.

In the public side, there are communities (groups of MISP's users or organizations) that are visible from the Internet and is possible to connect them using the MISP API, MISP user-interface or synchronize private MISP instances to fetch events. Each community has its own rules to join them and its own policy to collect/distribute its information. Below are some of the most important existing communities:

- CIRCL MISP Community (Computer Incident Response Center Luxembourg): is a government-driven initiative designed to provide a systematic response facility to computer security threats and incidents.
- CiviCERT MISP Community: is an umbrella organization formed by the partnership between Internet Content and Service Providers, Non-Governmental Organizations and individuals that contribute some of their time and resources in in order to globally improve the security awareness of civil society.
- NATO MISP Community: NATO Community is shared with some nations as well as the NCIRC (NATO Computer Incident Response Capability), which is responsible for coordination of cyber defence activities with nations and international organizations.

In this project, the MISP's network structure created is star-shaped, with a primary node MISP-Central and MISP-Client1 as child node. More child nodes could be added.

We will create a private community to share information about threats and malware between different companies. A community is composed of the local organizations on a MISP server and the remote organizations connected by the sync users.

Both primary and secondary instances, MISP-Central and MISP-Client1, will send and receive events from each other as shown in next figure:

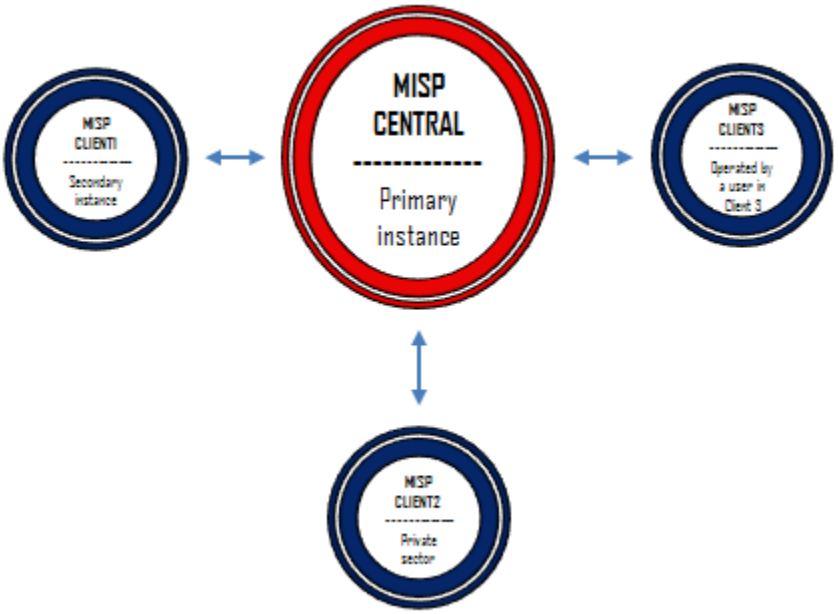


Figure 15. MISP instances network topology

All information is collected in MISP-Central, revised and redistributed to the rest of clients depending on the distribution group the information is included.

Next image shows the main steps to follow to configure a connection:

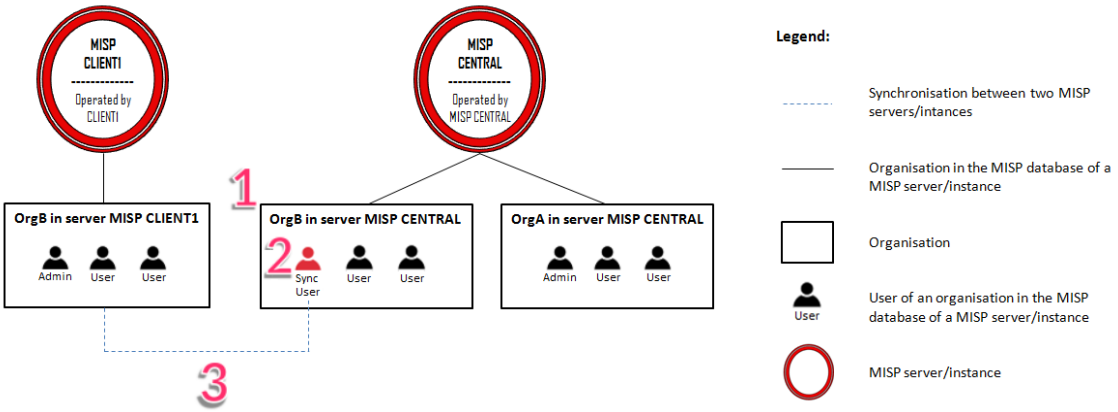


Figure 16. Synchronization setup process

- **Step 1:** Add MISP Client1 (OrgB in figure) as a local organization on MISP-Central server.

- **Step 2:** Add a *Sync User* (syncuser@OrgB.mispcentral) in the Client1 organization on MISP-Central server.
- **Step 3:** Set up a *Sync Server* on MISP Client1 server using the key (called Authkey) from the *Sync User* (syncuser@OrgB.mispcentral) previously created on MISP-Central server.

3.2.6.1. Syncrhonization user

Next, configuration required is shown:

1. Create syncronization user in MISP-Central.
   Option under Administration tab > Add user.
   Next page is shown:

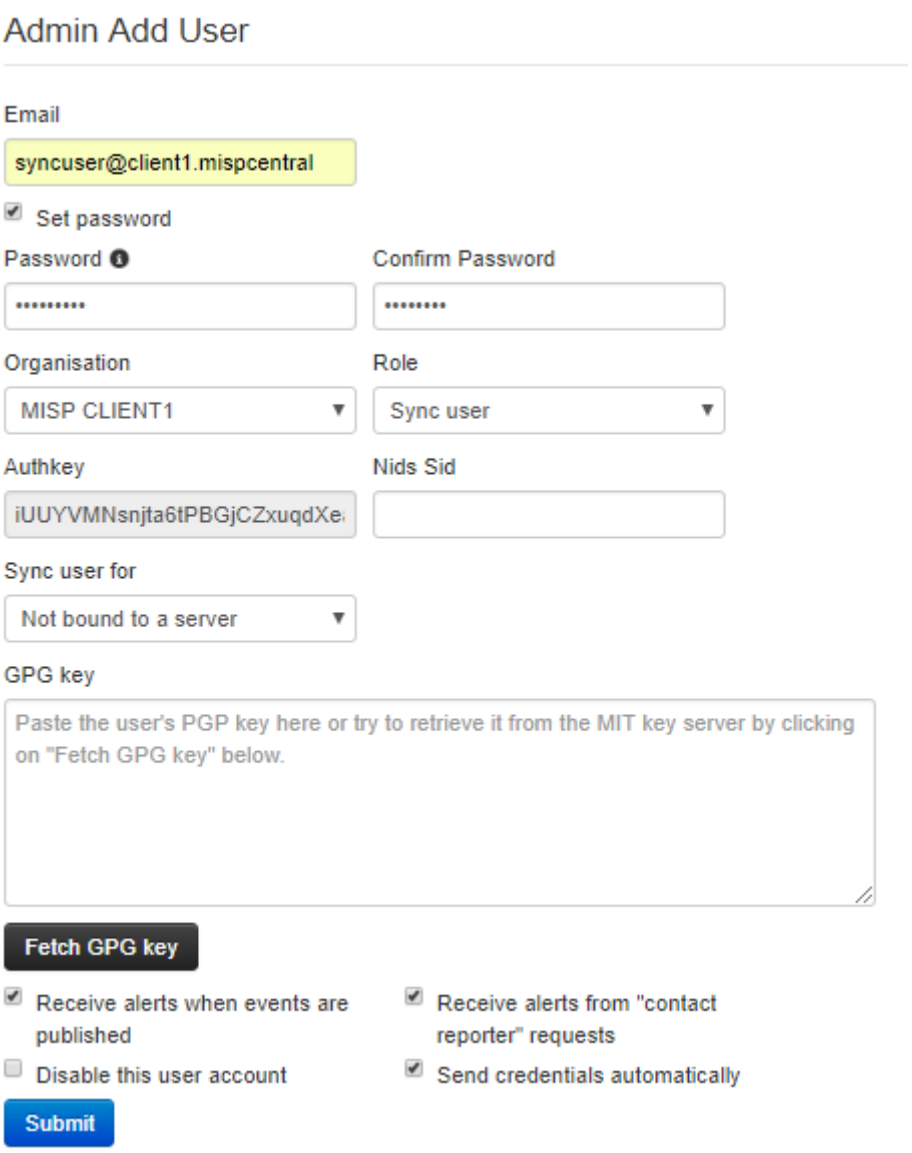


Figure 17. MISP-CENTRAL - Synchronization user configuration

The associated role for synchronization user syncuser@client1.mispcentral is 'Sync user'. We can define email/username to clearly identify the user and identify from which organization is member.

Previously to create the user the Organization MISP-Client1 has been added through option Add Organization under Administration tab. Further, configurations in Client's firewall have been done to allow connectivity from MISP-Central to MISP-Client1, located in DMZ zone.

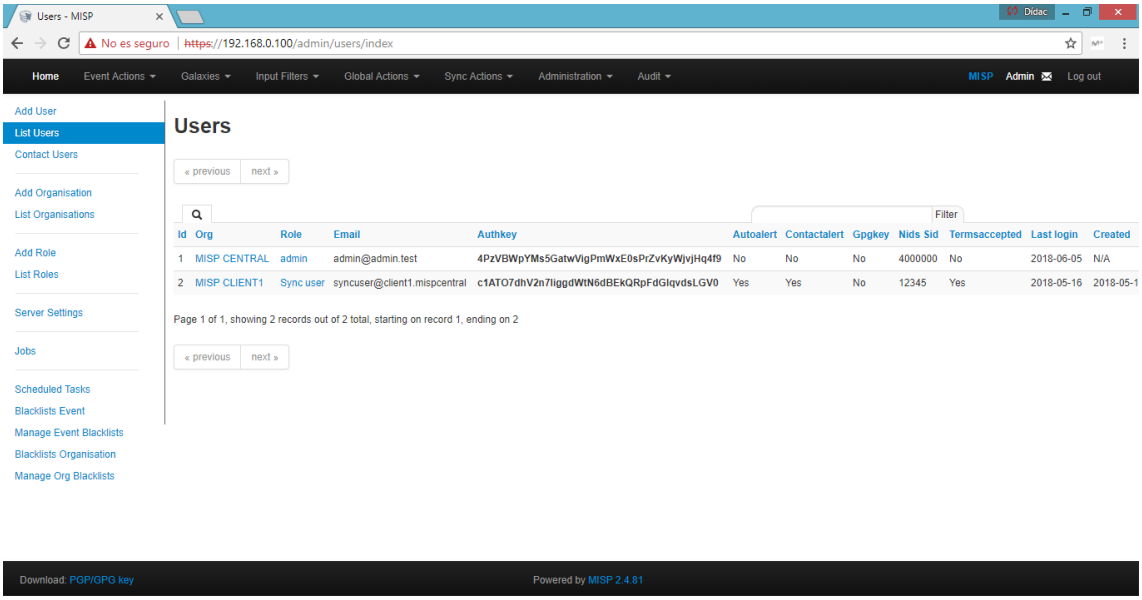


Figure 18. MISP CENTRAL Users list

As we can see, each user has its Authkey, used later in MISP-Client1 when adding a remote synchronization server.

3.2.6.2. Synchronization server

In order to create a new connection to a remote server, MISP-Central in this case, we address to Sync Actions tab > List Server > New Server in MISP-Client1 instance.

Next, configuration done is shown:

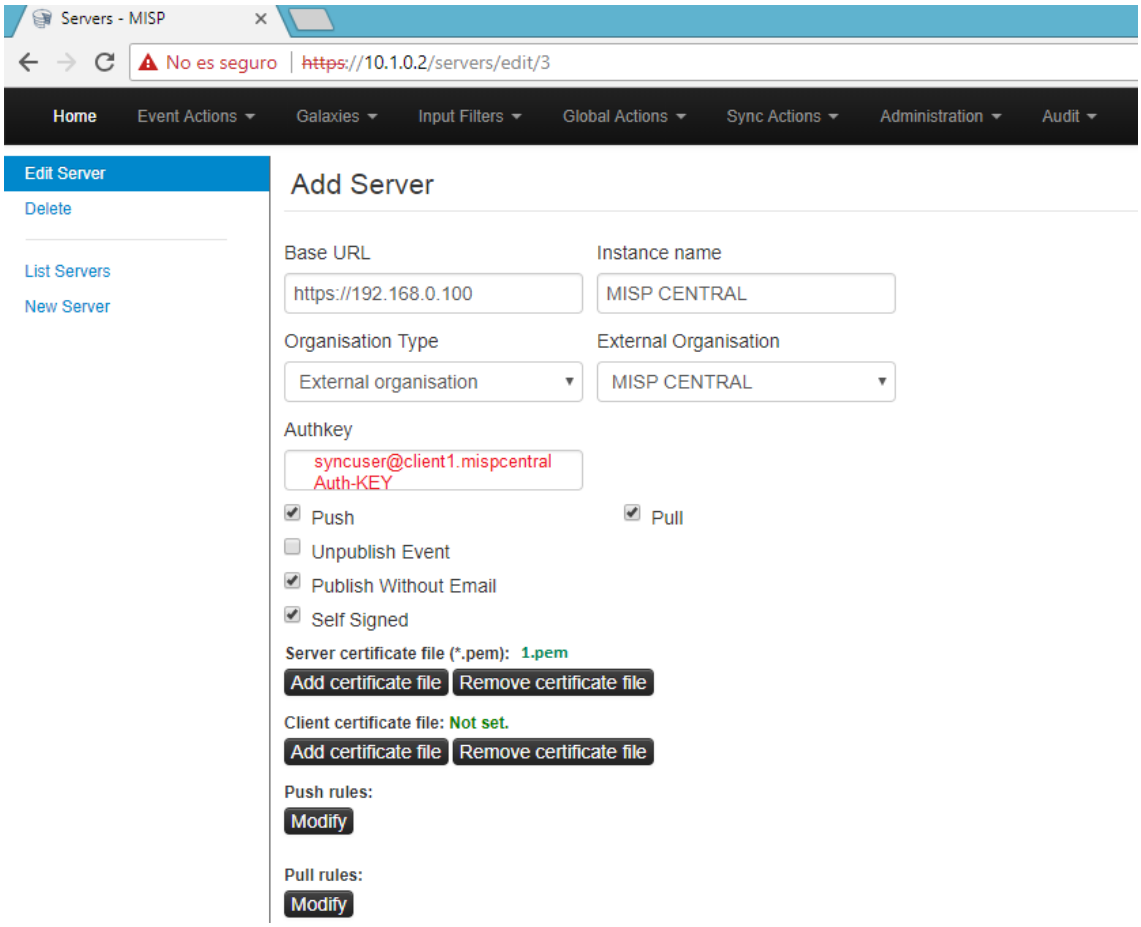


Figure 19. MISP CLIENT1 - Sychronization server configuration

These parameters are required:

- Base URL: remote server URL (MISP CENTRAL).
- Organization: Organization name owning the remote server.
- Authkey: authentication key of syncrhronization user created previously on the remote instance (syncuser@client1.mispcentral).
- Push: allow push events to remote server.
- Pull: allow pull events from remote server.
- Self Signed: allow synchronization with remote instances with self-signed certificate.
- Certificate File: remote server certificate in PEM format.

To obtain the certificate of remote instance (MISP CENTRAL) in PEM format is necessary to access from browser to the MISP user interface of the instance and export the certificate.

PEM is defined in RFC's 1421 through 1424, as a container format that may include just the public certificate (such as with Apache installs, and CA certificate files /etc/ssl/certs) or may include an entire certificate chain including public key, private key, and root certificates. Confusingly, it may also encode a CSR (e.g. as used here) as the PKCS10 format can be translated into PEM. The name is from Privacy Enhanced Mail (PEM), a failed method for secure email but the container format it used lives on and is a base64 translation of the x509 ASN.1 keys.

3.2.6.3. Testing connection and troubleshooting

Once the configuration is finished, we are able to test the connection with Connection test option with 'Run' button in Sync Actions > List Servers:

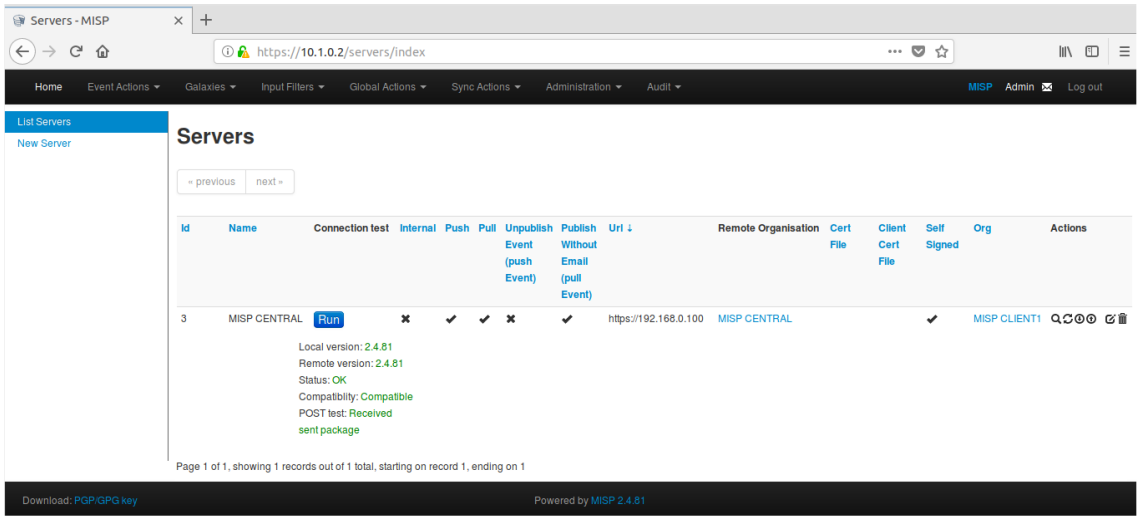


Figure 20. MISP synchronization servers list

If all configuration and communitations are correct, test results will show MISP version of local and remote instances, the compatibility and the result of the test.

In case the connection fails, next are some quick steps to locate the error:

- Try to connect with the Sync User account created directly to the remote server to ensure the password is still valid and the API key is valid. Check the role ‘sync user’ has been correctly assigned.
- Check the server certificate file is correct.
- Check the error message displayed under the Jobs tab when executing a pull or a push initiated manually from servers list.
- If connection issues do a traffic capture to find out more where is the problem.

Now, we are able to share information between MISP-Central and MISP-Client1.

3.3. pfSense firewall

pfSense is an open-source software-based firewall running on a FreeBSD Linux distribution. It has been selected because it allows expanding its functionalities adding ready-to-use packages through its packaging system. These extra features go from managing proxy servers, configure watchdogs, DNS servers and the most important one for the aim of this project: packet inspection and filtering with Suricata.

The installation is carried out with an ISO image in Oracle VirtualBox. This image can be downloaded from its official site (pfSense 2.4.2 release).

3.3.1. Initial setup

When pfSense machine is booted for the first time, basic configuration is needed after completing the initial wizard for language, time zone and disk partitions. The main configuration consists on define and assign addresses to the interfaces. The set up is done through options 1-Assign interfaces and 2-Set interface(s) IP address.

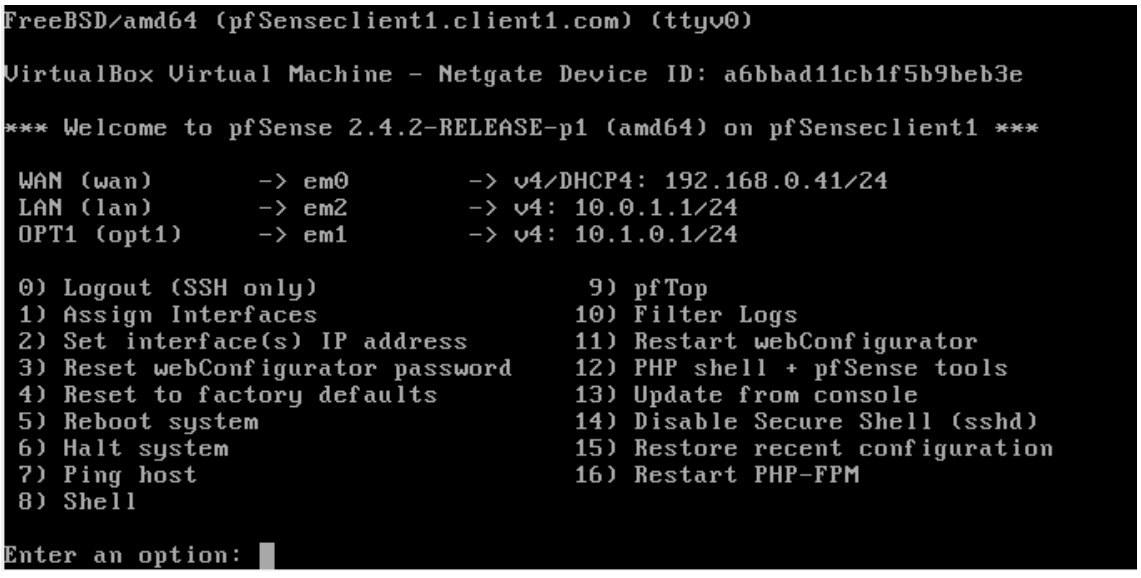


Figure 21. pfSense main console

The interfaces shown in the menu are the same as the ones defined for the machine in VirtualBox.

WAN interface is a bridge adapter with Internet connectivity; LAN and OPT are the internal networks LAN_client1 and DMZ_client1, respectively.

After this configuration, webConfigurator or web user interface can be accessed only from a host connected to LAN network (10.1.0.1/24). We can do all firwall and packages configuration from there.

In order to access pfSense webConfigurator, an Ubuntu Desktop machine has been deployed in VirtualBox with an interface adapter connected to LAN_client1.

This is the login site accessed with browser on http://10.1.0.1:

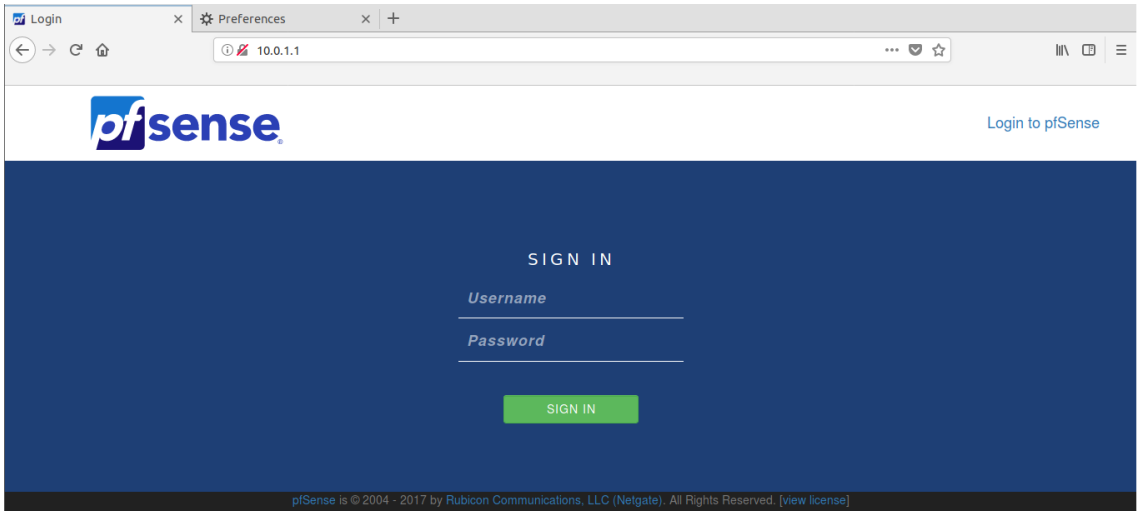


Figure 22. pfSense login page

3.3.2. pfSense webConfigurator

Once user interface is accessed, Status Dashboard is displayed.

By default, it shows general System Information as the firewalls hostname, installed version, CPU and memory usage, system time but other dashboards widgeds can be added.

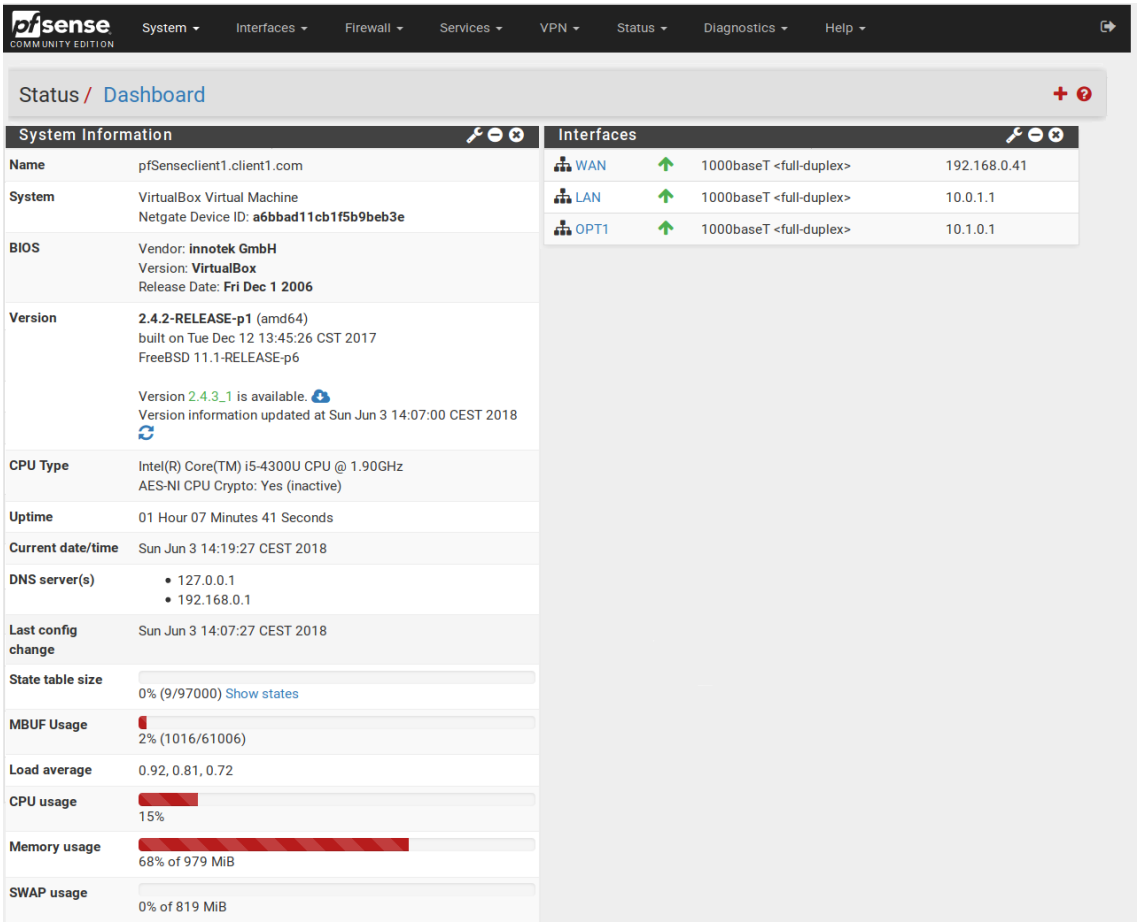


Figure 23. pfSense status dashboard

All options are available from its top toolbar:

a. System (General setup, Package Manager, User Manager)
b. Interfaces (Interfaces setup, names, IP addresses assignment)
c. Firewall (Aliases configuration, FW rules, NAT)
d. Services (NTP, DNS, DHCP, SNMP, packages)
e. VPN (VPN access configuration)
f. Status (event information, sytem logs, FW logs, traffic statistics)
g. Diagnostics (system troubleshooting tools)

At this point we will install Suricata package, our packet filtering IPS/IDS.

3.3.3. Suricata package installation

Suricata is an OpenSource Network Intrusion System (NIDS), which is developed by the Open Information Security Foundation (OISF). The project's intention is to secure the network via the inherent transparency of OpenSource.

Suricata it's a daemon that sniff network traffic and use rules to generate alert messages and drop packets. Besides, we will configure Suricata as an Intrusion Prevention System (IPS), which will give us the extra feature to block traffic that is highly probably malicious according to the configured rules.

The installation is accomplished using the pfSense Package Manager under System options.

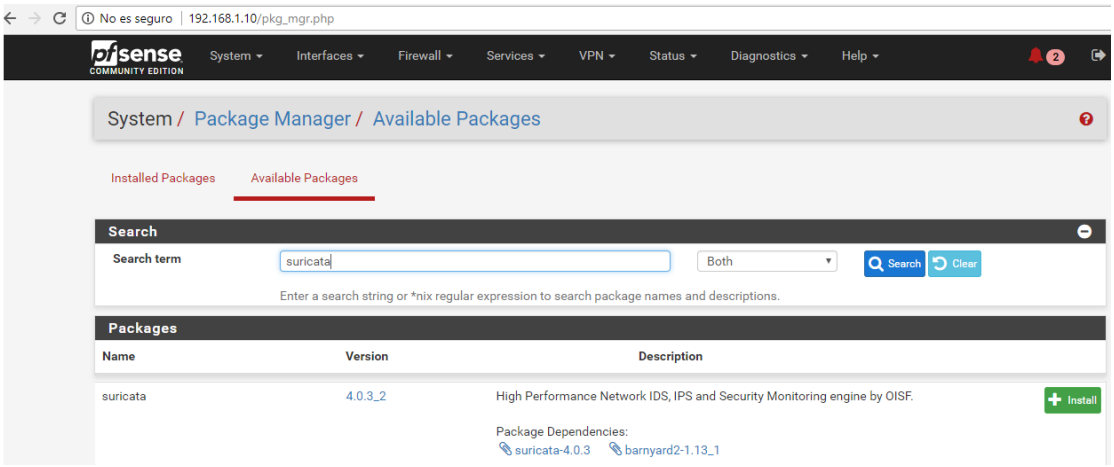


Figure 24. pfSense Package Manager

Once installed, we can access all Suricata options through Services tab > Suricata.

This software will allow us to inspect all traffic that flows from/to client's network. For this purpose, we need to configure Suricata on the different network interfaces we have.

Next, is explained how to configure an interface:

1. In Suricata menu, select Interfaces > Add.
2. General settings
   2.1. Select the interface to configure in the drop-down box.
   2.2. Enter a description for the interface which is going to be monitored.

3. Logging settings

Here we can select the level and the information kind to be logged. A normal configuration could be:

3.1. Check 'Send Alerts to System Log'

This option will log all alerts triggered on this interface in System Log.

4. EVE Output settings

This option allows creating a log file with more details of alerts and traffic detected in JSON format. Is a very powerful option to log the events and filter them in an easy manner whit the fields composing the JSON file.

5. Alert and Block Settings

5.1. Select Inline Mode in 'IPS Mode'

This option is important because will allow us to block packets in addition to alert only adding the feature of real-time blocking for the Intrusion Prevention System. When activating this option, internal mechanism act duplicating the interface and the traffic traversing the main one is passed to the copy after applying Suricata rules in real time.

*The unmentioned options mean that they remain per Default.

We need to configure as much interfaces as we want to monitor (WAN, LAN and DMZ) and it's posible to configure multiple Suricata instances in the same interface each one controlled by its own rules.

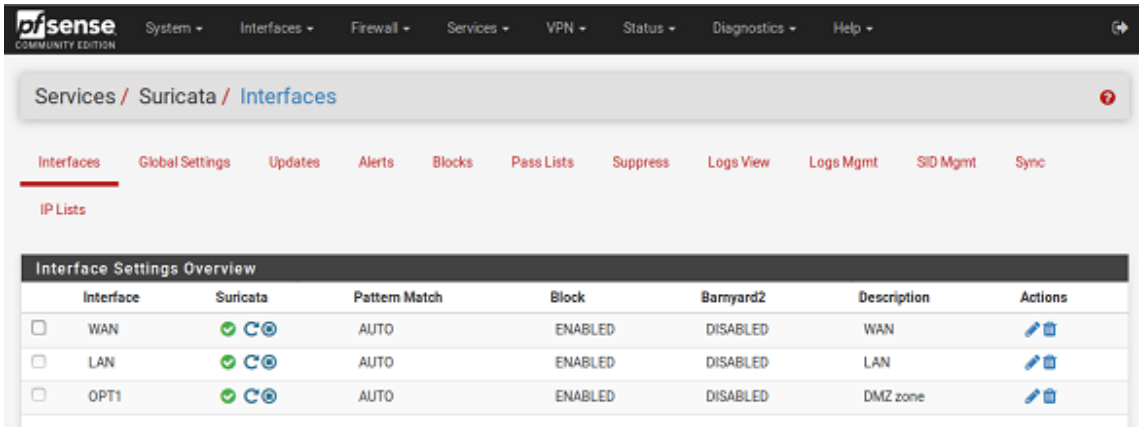


Figure 25. Suricata interfaces

In this project Suricata rules will be imported from MISP instance. Other rule sources are installed by default but will be deactivated.

3.3.4. Firewall rules

In order to permit traffic between specific hosts, some firewall rules had been set up.

From the pfSense Web configurator, firstly only accessible from LAN, next rules have been created:

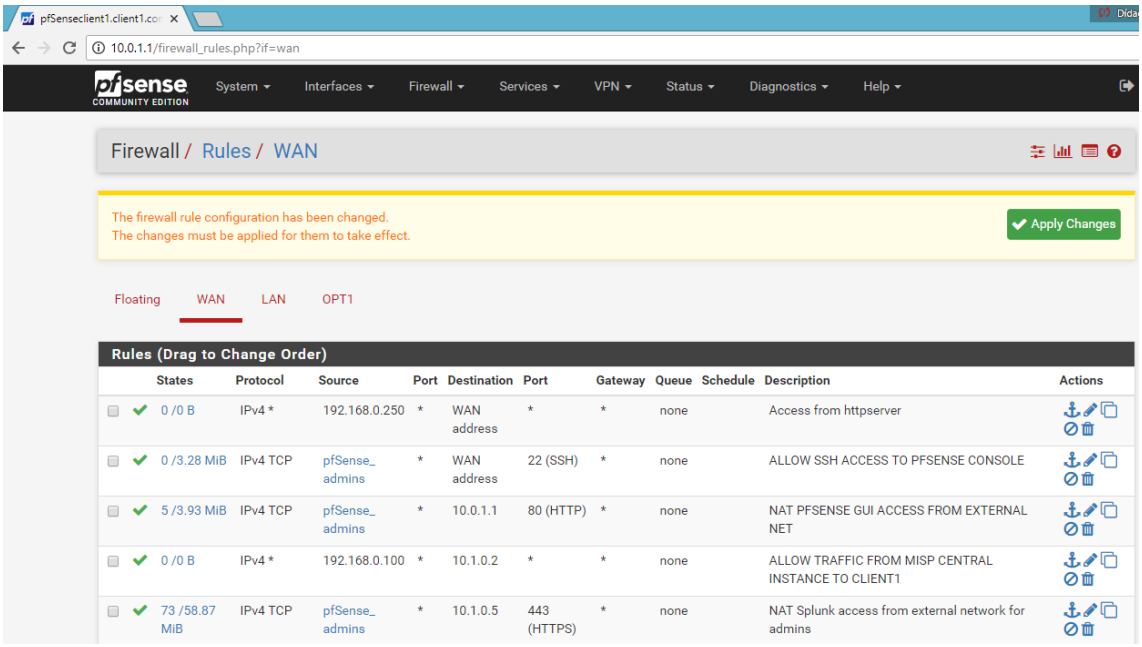


Figure 26. Firewall rules

First rule has been enabled due to the need of allowing connectivity to a remote server for testing purposes, simulating connections to a malicious server.

Second rule has been created to allow remote connection via SSH to pfSense host by admins in order to be accessible externally in case of need/emergency. This pfSense administrators are

contained in an alias list called pfSense_admins. For security, only predefined IP's within this alias can access via SSH.

There is another rule to allow connections between MISP instances due to host MISP Client1 is isolated from Internet, and connectivity for synchronization was needed.

The rules which Description starts with NAT mean that are created from a NAT rule. This type of rule allows traffic redirection from one IP/port to a different one. Using this rule, Splunk user interface is accessible from external networks from address https://192.168.0.72, the pfSense WAN address.

3.3.5. Suricata customization

In order to feed our Suricata IPS/IDS with latests threats published in MISP, we need to customize some files of Suricata software.

The file misp.rules has been added to Suricata configuration files for adding automatic rules load feature when updating or restarting pfSense.

Next steps are necessary to accomplish this:

1. Edit suricata_yaml_template.inc
   Add to /usr/local/pkg/suricata/suricata_yaml_template.inc the line
   misp.rules on rule-files section:

```
[2.4.2-RELEASE][root@pfSenseclient1.client1.com]/usr/local/pkg/suricata:
# IPS Mode Configuration
{$suricata_ips_mode}

legacy:
  uricontent: enabled

default-rule-path: {$suricatacfgdir}/rules
rule-files:
 - {$rules_files}
 - misp.rules
classification-file: {$suricatacfgdir}/classification.config
reference-config-file: {$suricatacfgdir}/reference.config
```

Figure 27. Suricata_yaml_template.inc customization

   This file is used by Suricata each time an interface configuration is updated from webConfigurator or pfSense is restarted.

2. Create *misp.rules* file in each interface folder where we want to apply rules exported from MISP.

   This file contains the rules that will be checked every time a packet is inspected. Suricata will check the rules from top to bottom till a rule is matched. When new rules are available in MISP Client1, these rules will be places into this file.

Other files that can be modified are classification.config and reference.config located in each interface directory. These files allow configuring custom classification types for alerts and different reference tags. Another important file is custom.rules, accessible from the web user interfaces to add rules manually and can be used for testing new rules.

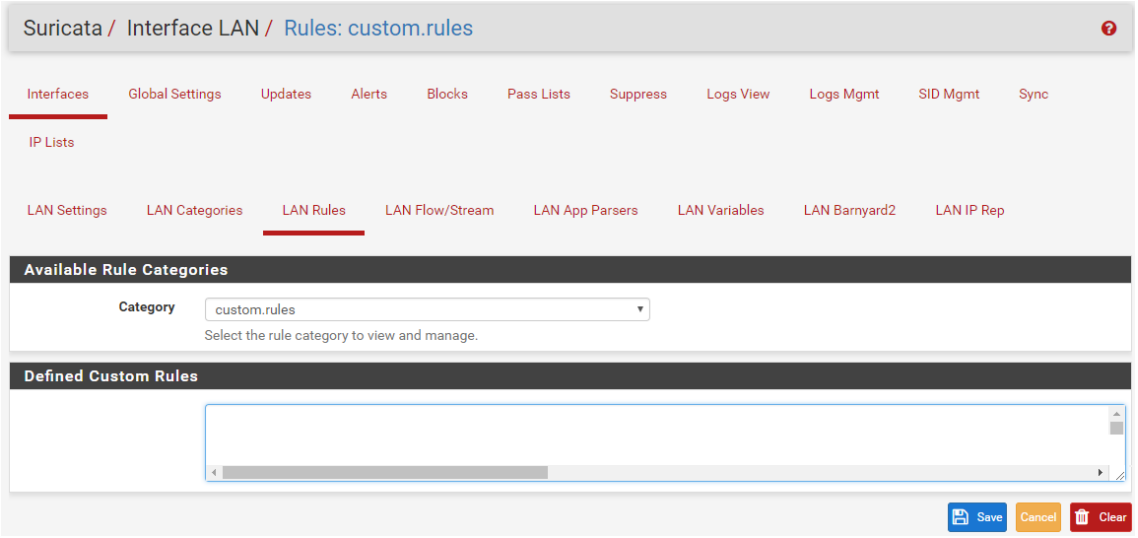


Figure 28. Suricata custom rules UI

3.3.6. Suricata update_rules.sh script

The process of rules update has been automated.

The script update_rules.sh written in bash is responsible to download latest IPS rules in MISP-Client1 platform and import them to Suricata IPS.

Basically, the function of this script is to download latest rules on MISP through its API and compare with the existing misp.rules on Suricata interfaces directories. If files are different

it indicates that there had been changes in MISP events database and then latest rules are loaded to Suricata and each interface is restarted for changes to take effect.

MISP has an API that allows most actions availilable in web UI. One functionality is designed to automatically generate signatures for intrusion detection systems. To enable signature generation for a given attribute in an event, 'Signature for IDS' field of the attribute must be set to 'Yes'. Not all attributes are applicable for signature generation, but the most important ones are: IP, domains, host names, user agents and hashes of file artifacts. To make this functionality available for automated tools an authentication key is used. This makes it easier for script tools to access the data without further form-based-authentication.

This script is programmed to run every minute through crontab, the task scheduler for UNIX systems. To add the task to crontab, cron package has been installed from pfSense Package Manager in web interface.

Update_rules.sh script can be found in Apenix.

From update_rules.sh, MISP API call is done with next command:

```
$curl -o /usr/local/etc/suricata/misp_rules_last_download/misp.rules –

insecure --header "Authorization: CF0j4VURzbHMdyGFvOlJGxaGHoAejApygFsPIvAJ" –

-header "Accept: application/json" --header "Content-type: application/json"

https://10.1.0.2/events/nids/suricata/download
```

Authorization key can be found in MISP UI, in section Automation under Event Actions tab:

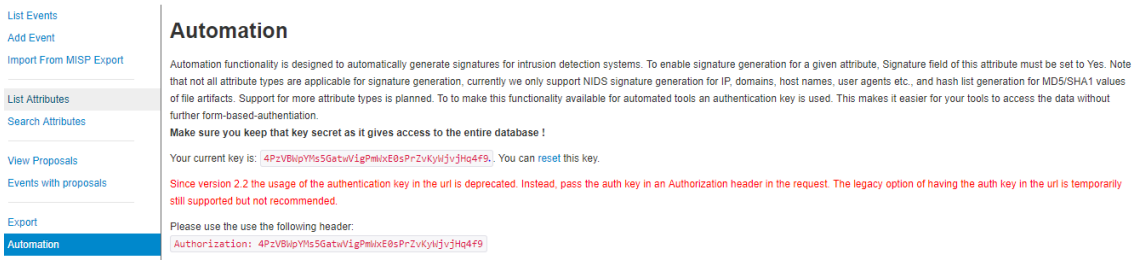


Figure 29. MISP Authorization key for API requests

3.4. Splunk SIEM

In order to monitor all alerts generated within the client's network, a Splunk server has been deployed on DMZ subnet. Splunk is a powerful platform for analyzing machine data gathered into indexes that can be rapidly searched from its web interface.

This tool allows to generate dashboards and reports with the events happened in the network and measure performance of pfSense machine. Due the possibility to trigger alerts using conditions, Splunk oftenly is used as SIEM (Security Information and Event Management), but in this project, alerts will be triggered from IPS and forwarded to Splunk.

In case of incident, Splunk allows to consult all events related to the incident like connections from/to certain IP, time stamps, actions made by firewall, etc.

3.4.1. Initial setup

First step for setting up Splunk is to install it. After downloading the Enterprise free version 7.1.1 from official site, the process consists on expand the tar file into an appropriate directory as /opt with the following command:

```
$tar xvzf splunk_package_name.tgz -C /opt
```

Once the installation has finished, Splunk is accessible in the IP address where has been installed on port 8443.

Splunk service has been configured to start at server boot time. For this, next command has been executed:

```
$/opt/bin/splunk enable boot-start
```

In order to facilitate access to the user interface without specifying any port when accessing from browser, SSL (HTTPS) access has been enabled for Splunk web and NAT rule has been enabled to redirect traffic from port 443 to 8443 internally in the server's iptables firewall.

To enable SSL access, in Settings -> Server settings -> General settings, the option Enable SSL (HTTPS) in Splunk web has been marked to yes.

Next figure shows the NAT configuration in the server:

```
root@splunk:/# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target     prot opt source               destination
REDIRECT   tcp  --  anywhere             anywhere             tcp dpt:https redir ports 8443
```

Figure 30. Splunk NAT HTTPS redirection

The command used to add this rule is:

```
$iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8443
```

After this initial setup and boot, Splunk can be accessed in https://10.1.0.5.

This address is restricted to be available only from certain addresses within the LAN subnetwork through rules in pfSense firewall.

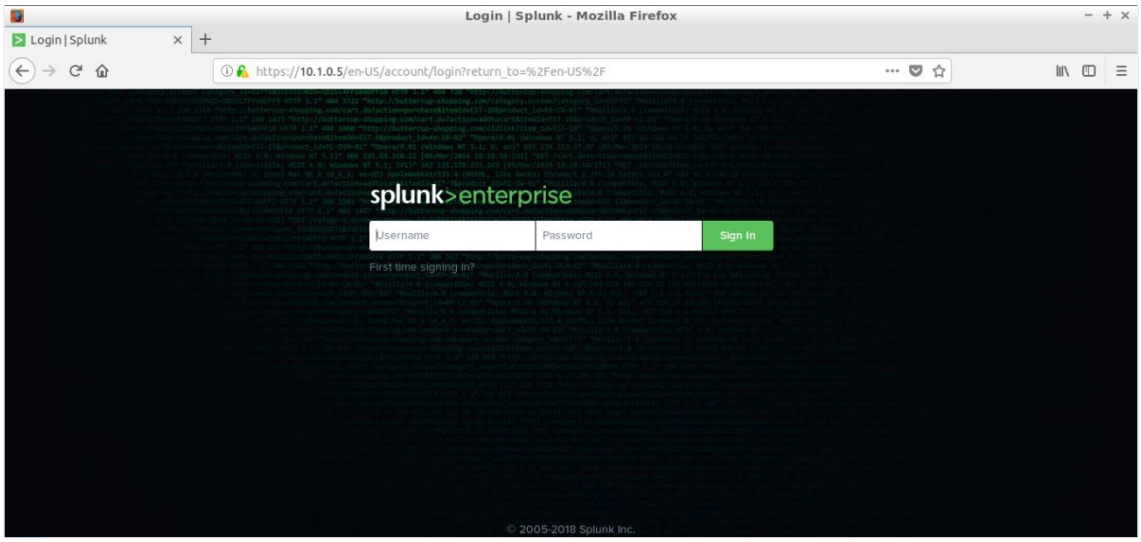


Figure 31. Splunk login page

Once Splunk is running, we will proceed to configure data inputs.

3.4.2. Splunk indexes

Indexes are used to classify different event types in Splunk for indexing data.

Next indexes have been created in section Settings -> Indexes:

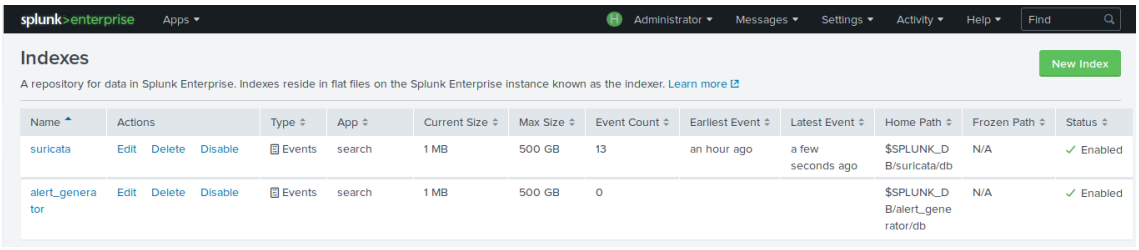


Figure 32. Splunk Indexes

Also, sourcetypes are defined in order to determine the structure of data types as JSON, CSV or custom delimited text files. In this case, our logs are all in JSON format. Thus, next sourcetypes have been configured in section Settings -> Sourcetypes:

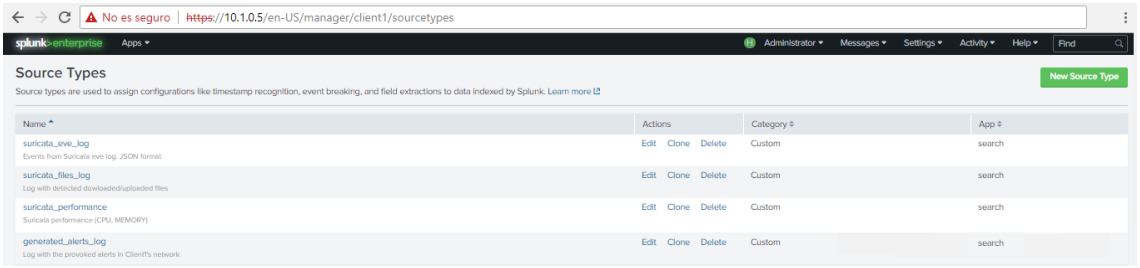


Figure 33. Splunk sourcetypes

Index name	Content	Sourcetypes	Source
suricata	Events from pfSense box related to alerts triggered in interfaces and stats of pfSense	suricata_eve_log suricata_files_log suricata_performance	pfSense
alert_generator	Generated alerts from script alert_generator	generated_alerts_log	Employee Desktops

Table 4. Splunk Indexes Description

After this, a listener starts on port 9999, where log files are sent to be indexed. The configuration is done through Splunk web in Settings -> Forwarding and receiving -> Configure receiving.

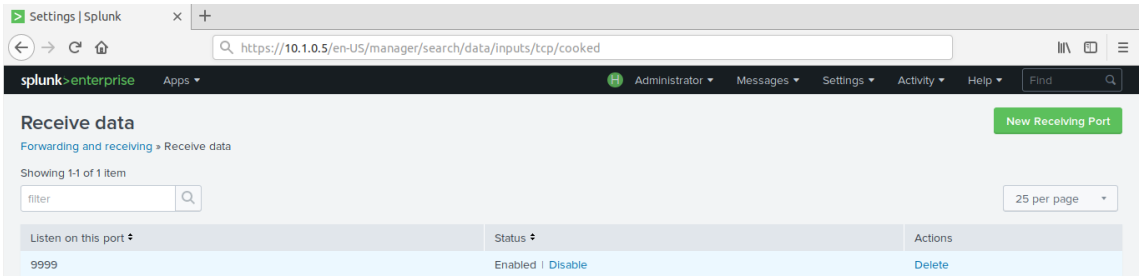


Figure 34. Splunk input listeners

All events of interest will be sent to this port and indexed to Splunk as explained in next section.

3.4.3. Event forwarding

Now Splunk is ready to receive events, event forwarders will be configured on data sources of interest.

All events will come from pfSense host, where the information of triggered alerts and detected threats is located. Also, to control the number of provoked alerts and compare them with detected by the system itself, each time a suspicious action is done in client's network the event is logged in the alerts_generator log, also located in pfSense machine for ease this count as attackers' addresses change every alert_generator.sh script execution and would suppose a different source to index to Splunk every time.

The tool used to forward log files from any source to Splunk is Splunk Universal Forwarder 7.1.1. This tool is free and maintained by Splunk developers and can be installed on any OS.

To install this software, we have to open a shell in pfSense box through option 8 in console (See Figure19. pfSense main console).

The package has been downloaded in a Desktop machine and trasfered to this host through SCP transfer due to register and loggin to Splunk official site is needed for download the program. The version for 2.6+ kernel Linux distributions for 64-bit system is used in tgz package format.

Once the package is stored in pfSense host, the installation is performed using next command:

```
$tar xvzf splunkforwarder.tgz -C /opt
```

For configuring this tool, there are two files to be edited:

- Inputs.conf
- Output.conf

Inputs.conf contains the directories to be monitored by splunkforwarder and when changes occur, these files are sent to Splunk.

The eve.json files for each monitored interface containing all detected events in Suricata IPS are sent and tagged accordingly to Splunk. The clause index will forward the events to specified index created previously and sourcetype can be used later for filtering events in Splunk searches. _meta tag adds metadata to events.

Alerts.log and stats.log have been disabled because same information is found in eve.json log.

Configuration done for this file is shown in next figure:

```
[default]
host = pfSenseclient1.client1.com

[monitor:///var/log/suricata/suricata_em040915/eve.json]
disabled = false
sourcetype = suricata_eve_log
index = suricata
_meta = interface::wan_client1

[monitor:///var/log/suricata/suricata_em158956/eve.json]
disabled = false
sourcetype = suricata_eve_log
index = suricata
_meta = interface::dmz_client1

[monitor:///var/log/suricata/suricata_em247621/eve.json]
disabled = false
sourcetype = suricata_eve_log
index = suricata
_meta = interface::lan_client1

[monitor://var/log/alert_generator/alert_generator.log]
disabled = false
sourcetype = generated_alerts_log
index = alert_generator
host = pfsenseclient1_LAN_NETWORK

[monitor:///var/log/suricata_performance/suricata_performance.log]
disabled=false
sourcetype = suricata_performance
index = suricata

[monitor:///var/log/suricata/suricata_em247621/files-json.log]
disabled = false
sourcetype = suricata_files_log
index = suricata
```

Figure 35. Splunk Universal Forwarder - Inputs.conf

Outputs.conf contains the configuration for remote servers where logs set in inputs.conf will be sent. Next figure shows the content set for this file:

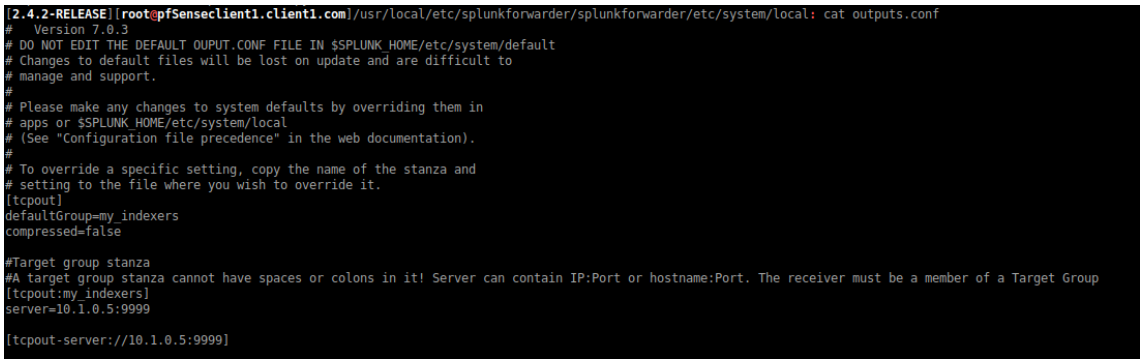


Figure 36. Splunk Universal Forwarder - Outputs.conf

After changes on these files, a restart of the service is needed. Can be accomplished using next command:

```
$/usr/local/etc/splunkforwarder/splunkforwarder/bin/splunk restart
```

When restart is completed, we are ready for events search and treatment with Splunk web.

3.4.5. Splunk web interface

Now all data inputs are ready, Spluk web is used to navigate and filter events for monitoring purposes and statistics extraction for analysis.

Splunk is organized in applications. For all searches, reports and dasbhbords related with Client1's network, Client1 Security Operation Center application has been created under application tab.

Now, this application appears on the left panel or dropdown application menu in the web:

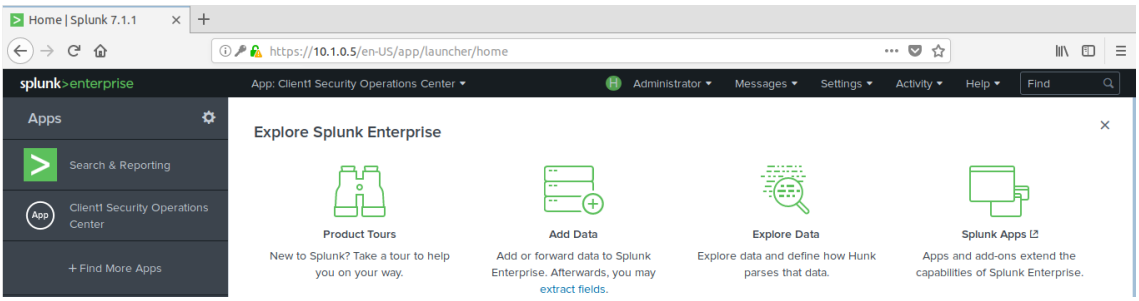


Figure 37. Splunk web – main page

In next section, search function will be explained, essential for event management.

3.4.6. Splunk searches

The Search app is the application for navigating the data in Splunk platform. Once a search is done, the query or results can be saved to make up dashboards or reports.

Basic structure of a Splunk search is:

```
Index=”suricata” sourcetype=”suricata_eve_log” | stats latest(src_ip),

latest(dest_ip) by timestamp
```

The result of this search are events contained in index Suricata of sourcetype Suricata_eve_log and shows columns src_ip and dest_ip differentiating events by its timestamp.

The parts that make up a search are:

- Index: index where events pertain
- Soucetype: the type of the events
- Stats: indicates the start of the search
- Where: clause to filter events

As example, the search used to monitor pfSense CPU utilization is:

```
index=suricata sourcetype=suricata_performance | stats latest(idle_CPU) as

CPU_IDLE by timestamp | eval "%CPU USE"=(100-CPU_IDLE) | fields timestamp,

"%CPU USE"
```

As CPU value retrieved from pfSense logs is the percentage of idle CPU, eval clause is used to do the calculation 100-CPU_IDLE in order to obtain the utilization.

Following example is the search use to group events by type and count them:

```
index="alert_generator" host="pfsenseclient1_LAN_NETWORK" | stats

latest(attack) as attack by timestamp | stats count by attack
```

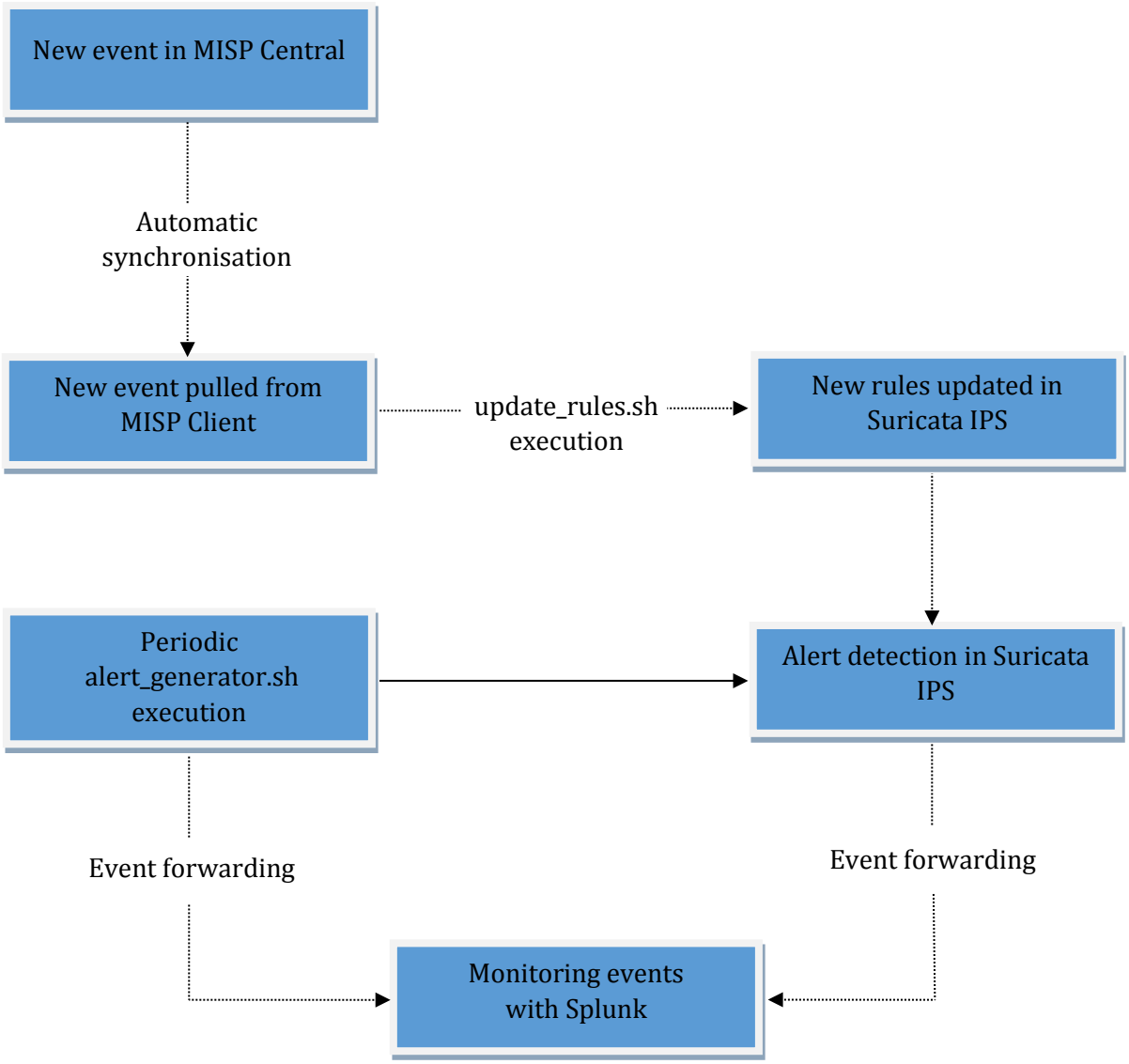
More examples of Splunk searches can be found in Appendix.

3.5. Testing environment

In order to test the whole system, next process has been designed:

- Generate continuous TCP traffic in the corporate network at different throughput for more realistic modelling.
- Generate Suricata IPS rules from update_rules script using information contained in MISP-Client1.
- Generate alerts within the corporate network using alert_generator script.
- Detect alerts with Suricata IPS.
- Monitor pfSense performance and provoked vs detected alerts in Splunk.

This is the flowchart of the testing process:


New event in MISP Central
Automatic synchronisation
New event pulled from MISP Client
update_rules.sh execution
New rules updated in Suricata IPS
Periodic alert_generator.sh execution
Alert detection in Suricata IPS
Event forwarding
Event forwarding
Monitoring events with Splunk

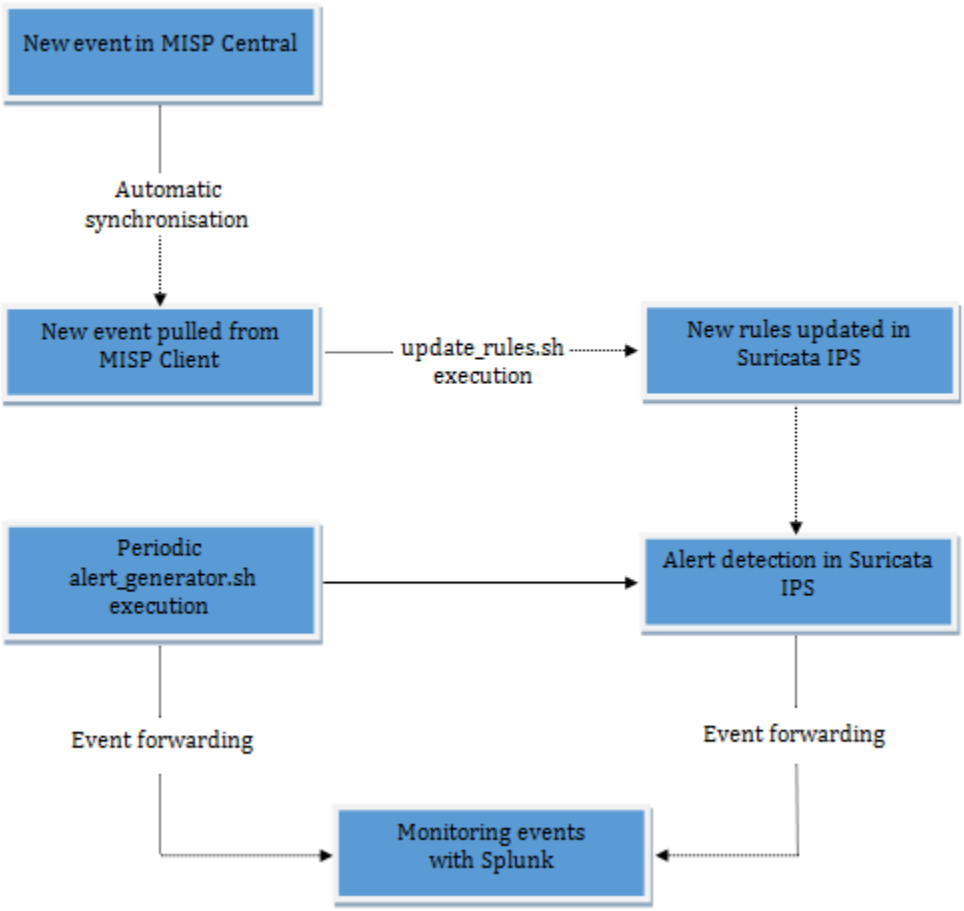


Figure 38. Testing flowchart

This test will determine the correct functioning of the proposed system and will show statistics about detected threads and generated threads and also the load supported by Suricata packet inspector.

Next sections explain the preparation of the testing environment in more depth.

3.5.1. Generating IPS rules

In order to detect attacks, firstly, rules for IPS have to be generated from Events in MISP with attributes in those events marked for IDS rules export.

Next events have been created in MISP-Central instance that will be automatically pulled from MISP-Client1. Once these events pertain to this instance, update_rules script explained in section Suricata update_rules.sh script will update the IPS with those new rules using MISP API.

Events created are:

- **Remote connection attempt**

  This alert is triggered when a connection to port 22 TCP is stablished. This port is used to connect remotely to a computer.

  Rule:

```
alert tcp any any -> any 22 (msg: "Remote connection attempt – TCP 22 SSH";
classtype:trojan-activity; sid:4000102; rev:1; priority:2;)
```

  The event created in MISP contains an attribute of the type 'Network activity – port' as shown in next image:

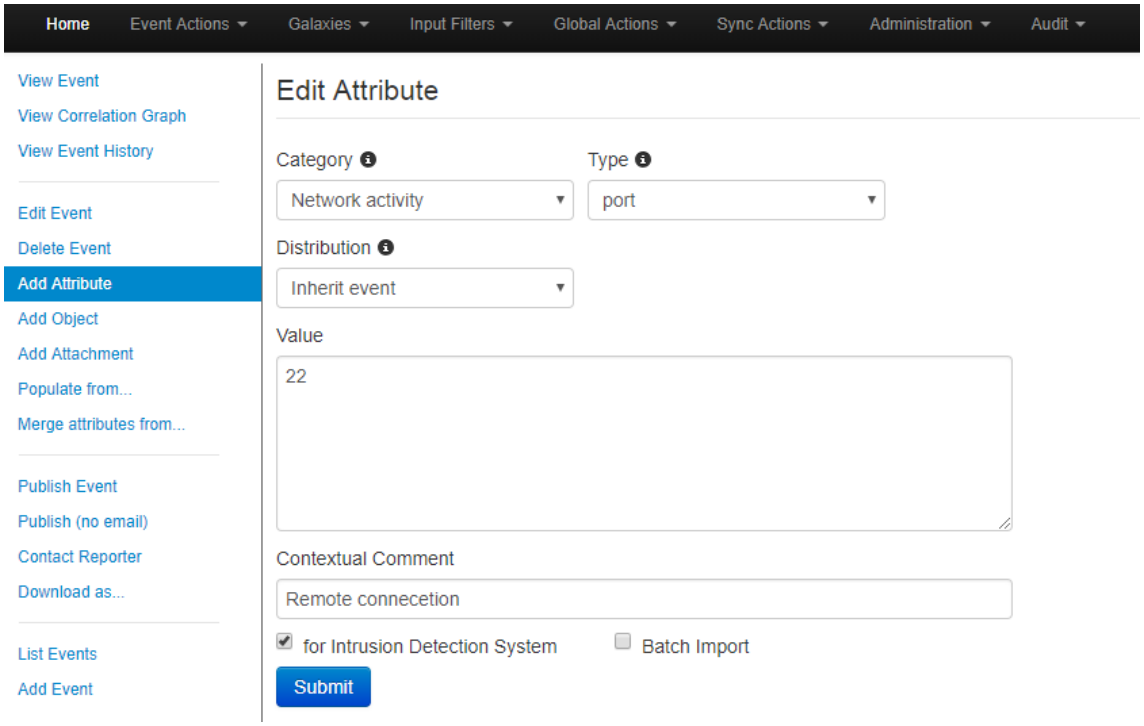


Figure 39. MISP Event – attribute Network activity > port

- **Connections to a known Command and Control server (C2C)**

  This alert is triggered when an incoming or outgoing communication to a specific IP identified as malicious is detected.

  Rule:

```
alert ip 192.168.0.250 any -> $HOME_NET any (msg: "MISP e6 [] Incoming From
IP: 192.168.0.250"; classtype:bad-unknown; sid:4000101; rev:1; priority:1;
reference:url,https://192.168.0.200/events/view/6;)
```

The event created in MISP contains attributes of the type 'Network activity – ip-dst/src' as shown in next image:

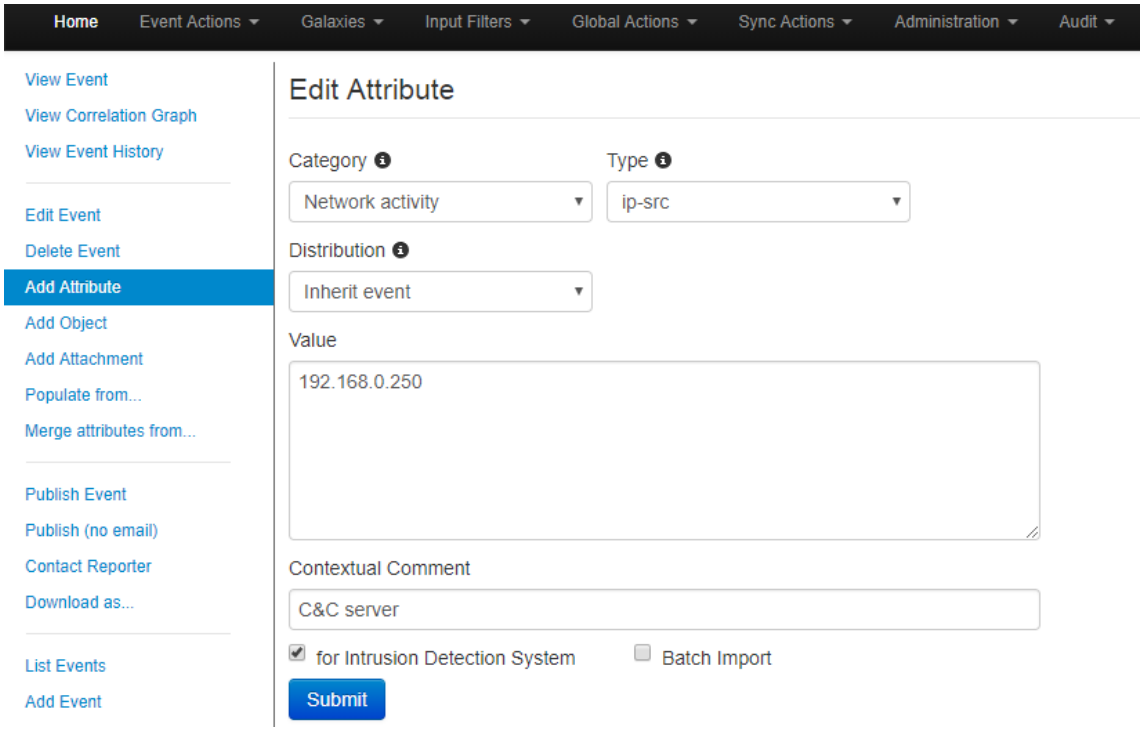


Figure 40. MISP Event – attribute Network activity > ip-src

- **File download tagged as malicious by hash**

  This alert is triggered when a file hash is contained in a blacklist generated from MISP. This list is updated when script update_rules is executed, obtaining all hashes

introduced in MISP and will alert and block the download if detected. In addition, a copy of the downloaded file and a file with information extracted from the file (Filename, Size, MD5, download time, Source IP) will be logged in the files directory of the interface where alert is triggered in Suricata host (/usr/local/etc/suricata/{suricata-interface}/files/). These files can be used as samples by analysts for malware reversing in investigations.

Rule:

```
reject tcp any any -> any any (msg:"Malicious file - blacklisted MD5
hashdetected";filemd5:/usr/local/etc/suricata/misp_hashes_last_download/md5_b
lacklist.txt; filestore; classtype:suspicious-file; sid:4005002; rev:1;)
```

The event created in MISP contains an attribute of the type 'Artifacts dropped – md5'.

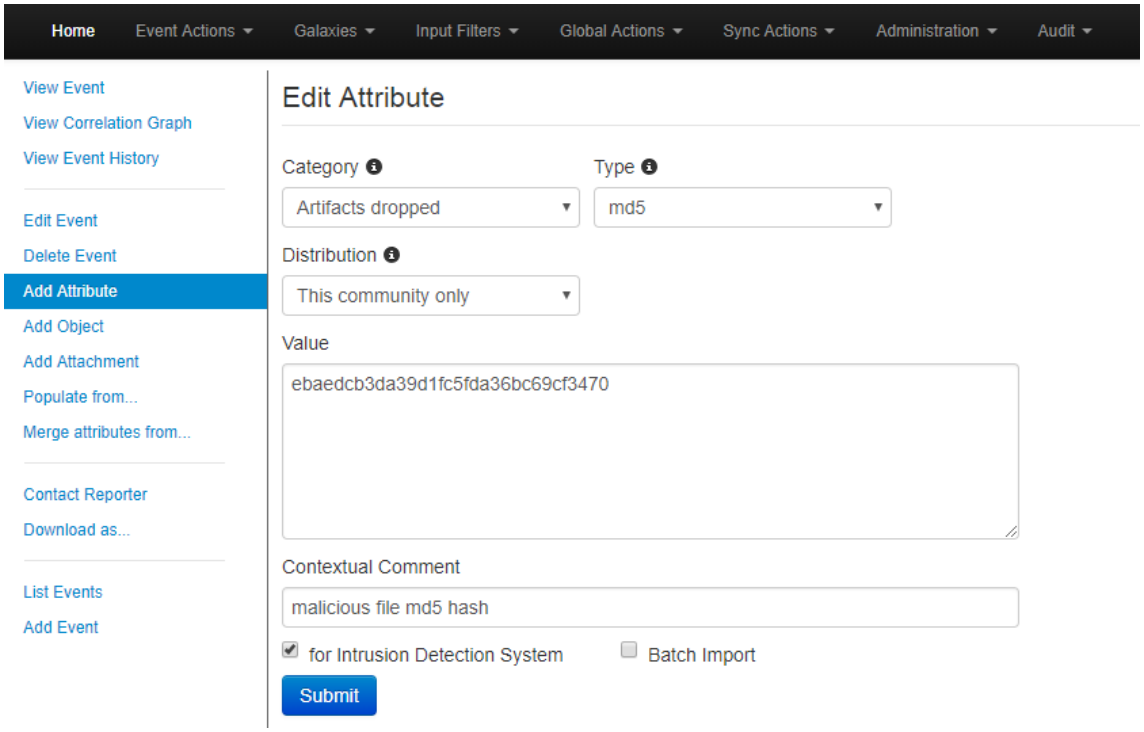


Figure 41. MISP Event – attribute Artifacts dropped > md5

- **Possible network regonaisance**

This alert is triggered when a ping request or response packet is sent from/to external network in order to detect recognition attacks.

Rule:

```
alert icmp $HOME_NET any <> $EXTERNAL_NET any (msg:"ICMP PACKET FROM LAN";
classtype:network-scan; sid:4000003; rev:1;)
```

This rule is configured from the web UI interface and saved in the rule files custom.rules.

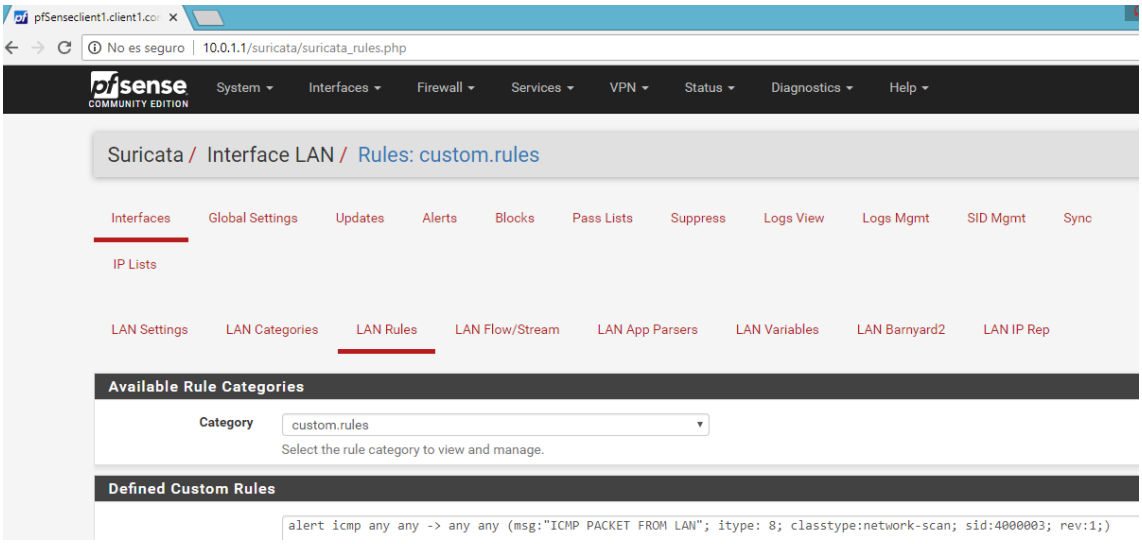


Figure 42. Suricata custom.rules editor

- **Potencially malicious site connection**

  This alert is triggered when a connection to a site identifyied as malicious or malware related is done.

  Rule:

```
alert http $HOME_NET any -> $EXTERNAL_NET any (msg: "MISP e3 [] Outgoing URL:
facebook.com"; flow:to_server,established; content:"maliciousurl.com";
fast_pattern; nocase; http_uri; tag:session,600,seconds; classtype:trojan-
activity; sid:4000051; rev:1; priority:1;
```

```
reference:url,https://10.1.0.2/events/view/3;)
```

The event created in MISP contains an attribute of the type 'Network activity – url'.

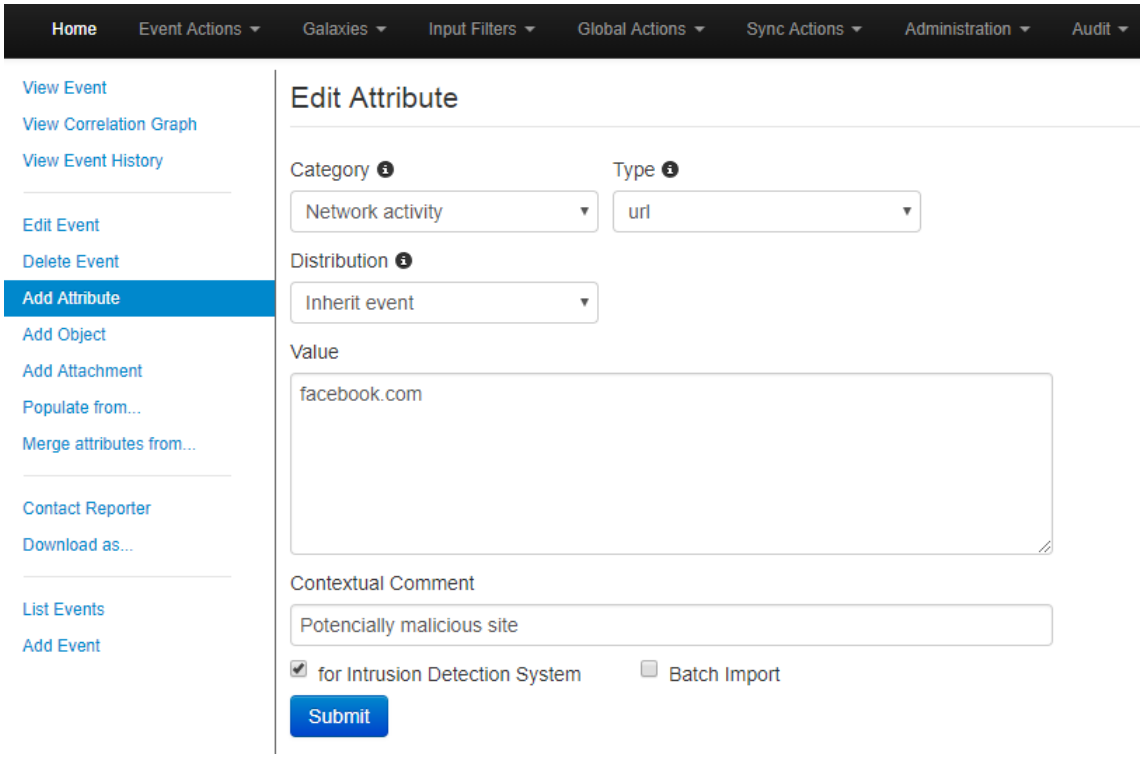


Figure 43. MISP Event – attribute Network activity > url

Once these events are created in MISP-Central, automatically, MISP-Client1 will pull these events due to the configuration of synchronization between MISP instances done previously and next time update-rules script is executed, these events will be transformed into IPS rules and loaded to Suricata IPS in the client's network, ready for detecting traffic matching with these rules.

3.5.2. Generating TCP traffic

In order to test the system under different workloads, different sustained loads of data had been inserted to corporate network to simulate activity on LAN subnet and force Suricata to inspect many packets. The aim of this is obtain Suricata's performance in packets inspected or dropped/non-inspected under different work loads in a more realistic experiment.

For this purpose, a traffic generator tool is needed. These programs are essential for design, testing and development of networks and they can manage most of standard communication protocols.

After some research, selected traffic generator choosen was Ostinato due this tool allows exactly what I wanted to do. It enables network packet creation, generation and sending to any available machine interface. Used through a friendly GUI and working in multiplo OSs. For non-availiable options in GUI, Python API is availiable, but for the purpose of the project will not be necessary to use it.

Ostinato is open-source software licensed under GNU GPLv3. It can be downloaded from the Ostinato official web page for free.

To run Ostinato during tests, Ostinato Traffic Generator virtual machine has been added in VirtualBox on top of Ubuntu 17.1 connected on LAN subnet in the corporate network.

The tool is installed using the command:

```
$sudo apt-get install ostinato
```

To start the user interface, command 'ostinato' has to be executed as superuser due to the program needs to take control over network interface card.

This is the GUI (Graphical User Interface):

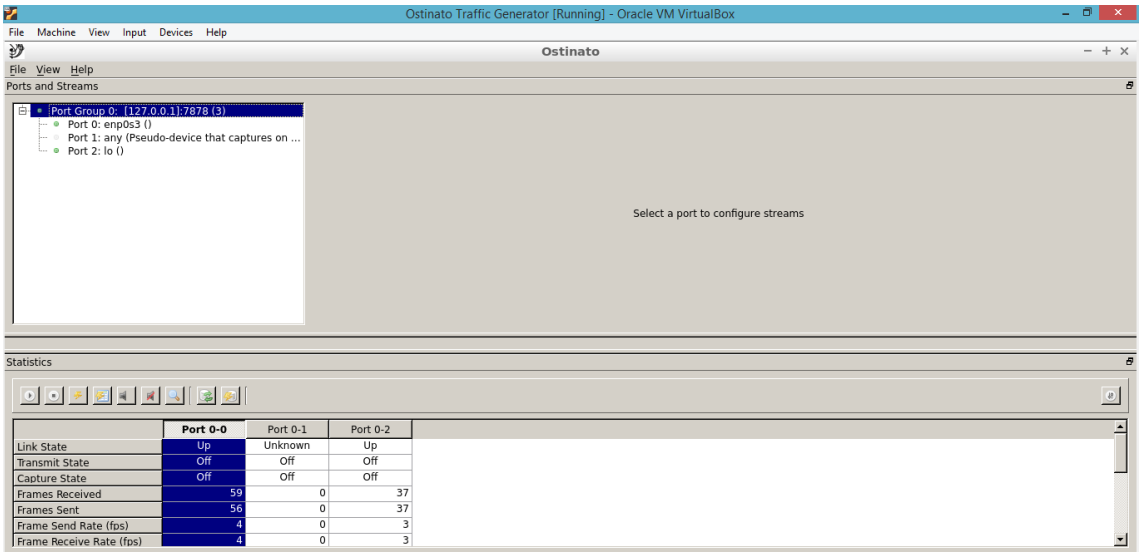


Figure 44. Ostinato GUI

Once in the interface, we should select an interface from the left panel. This port is used to send the crafted packets. When selected, Streams menu appears next to Port and Streams menu:

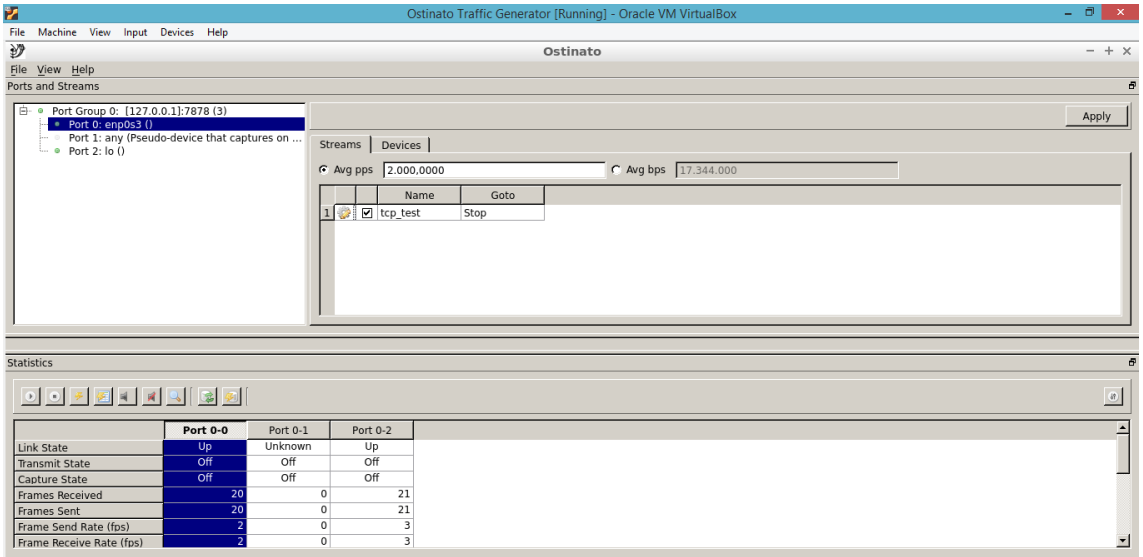


Figure 45. Ostinato GUI 2

In this new menu, right click for adding a new stream, assign a name and edit it.

3 streams have been created and saved to use for testing:

- 1Mb/s stream
- 5Mb/s stream

- 20Mb/s stream

These streams have as destination the pfSense address and inject 1064bit TCP packets at different packet rates to achieve the desired throughputs for stressing pfSense box when inspecting traffic.

Is possible to concatenate streams to run secuentially one after the other. To do so, we configure each stream to GOTO next when finishes.

Next picture shows the example of running 10 seconds each configured stream:

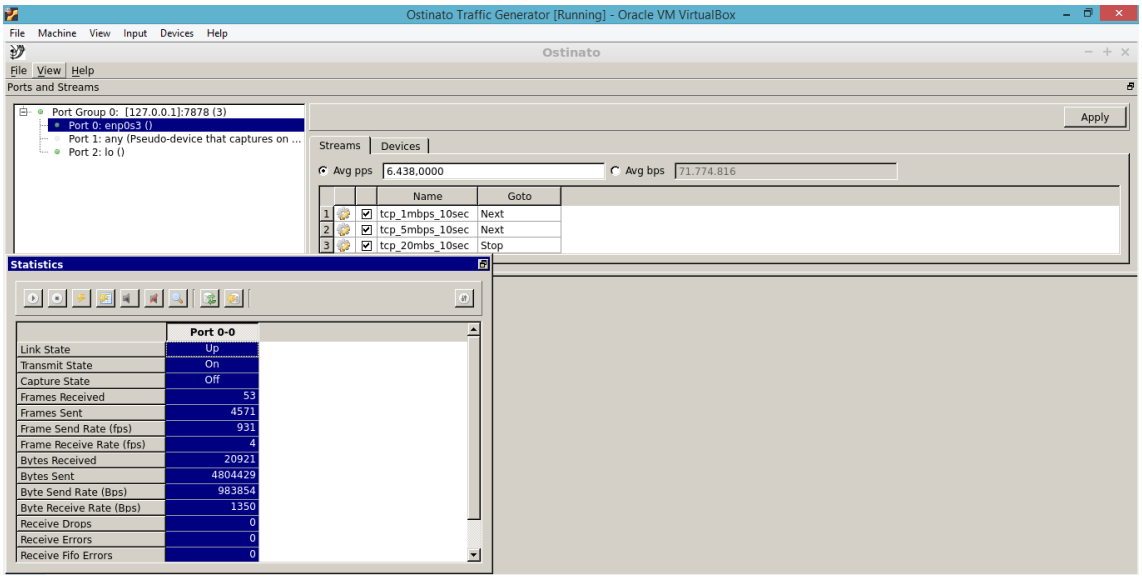


Figure 46. Ostinato streams chain

In pfSense dashboard we can observe the generated traffic entering the LAN interface on pfSense box:

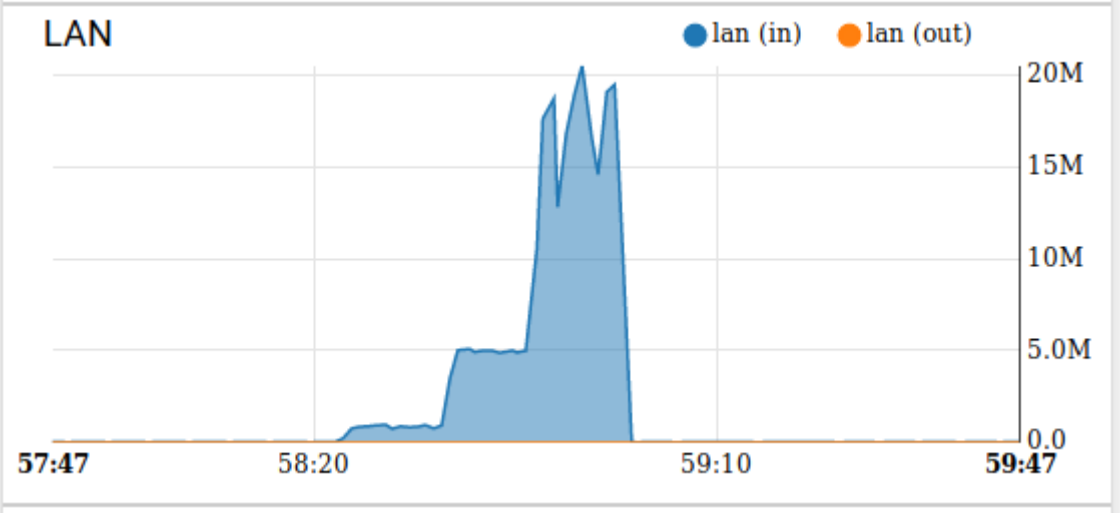


Figure 47. Osinato network traffic generated

3.5.3. Generating alerts

In order to test generated IPS rules for Suricata through update_rules.py script execution, script alert_generator.sh has been developed.

Alert_generator is executed every minute from a computer in the Client1 LAN subnet from a crontab task. When executed, IP address is reset to simulate multiple employee desktops within the LAN subnet and a random attack is generated. Parallel, a log is generated for each provoked attack and sent to Splunk in order to compare the number of generated attacks against detected ones by Suricata.

According with the Suricata rules previously created, 5 different attacks have been configured to test correct detection:

- SSH session started.
- Connection to a potencially malicious IP (known Command and Control server)
- Malicious file download identified by MD5 hash
- Network recognition
- Malicious site connection

3.5.4. Monitorig system

To monitor system behavior, two dashboards have been added to application Client1

Security Operation Center in Splunk:

- pfSense Performance

- Alerts

pfSense Performance dashboard

This dashboard contains graphs about the CPU utilization, available memory and inspected/non-inspected packets.

The objective of monitoring CPU occupation is to see if the pfSense machine is well dimensioned or, on the contrary, machine is running out of processment power or memory in some situation.

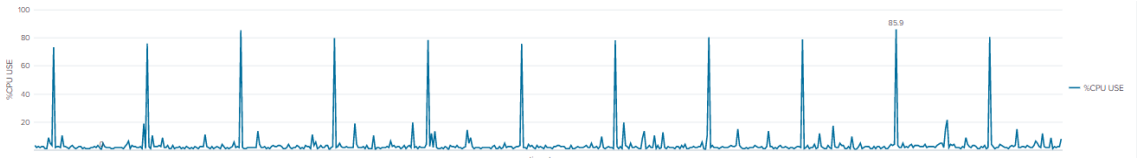


Figure 48. Splunk – pfSense Dashboard - CPU utilization

The available memory can be used for system administrators to detect if pfSesne box is running out of RAM memory.

Figure 49. Splunk -fSense Dashboard - Available memory

To obtain these statistics, a script called suricata_performances_stats.sh has been developed. This script is executed from a crontask, configured from the pfSense Web Configurator with package Cron as shown in next image:

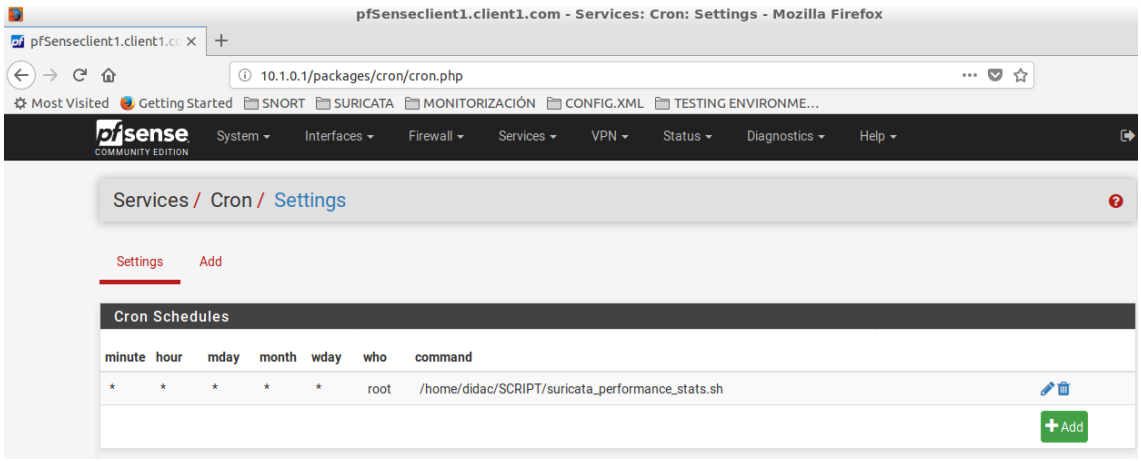


Figure 50. pfSense - suricata_performance_stats.sh Crontab

Every minute on the dot, the script is executed and obtains statistics every 6 seconds and logs them to a log file. This log is forwarded to Splunk following the same process explained in section 3.4.3. Event forwarding and is used to plot the graphs using searches and dashboards. Script Suricata_perfromcance_stats.sh and log sample Suricata_performance.log can be found in Appendix.

Graph with packet inspection statistics will serve to detect if packets are being directly transferred and not inspected by pfSense, meaning a loss of reliability on the system. This statistics are extracted from internal pfSense log called eve.json, containing information about alerts and statistics. Event type is “stats”, in from index=suricata sourcetype=suricata_eve_json.

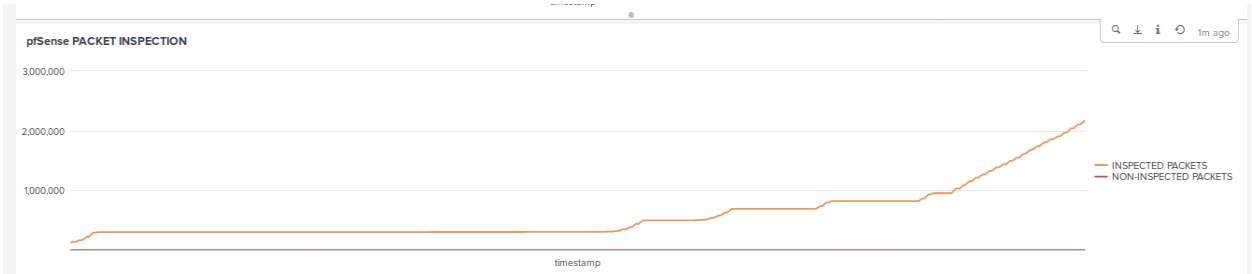


Figure 51. pfSense - Packets inspected/non-inspected graph

Alerts dashboard

This dashboard is used to monitor the alerts generated from Alert Generator and alerts detected by Suricata IPS.

The graphics on the top show the intencionally generated threats using script alert_generator.sh. First panel is a pie chart with percentages depending on threat type; the second one shows the total amount of each type of thread; the third contains info about the source and time when were generated.

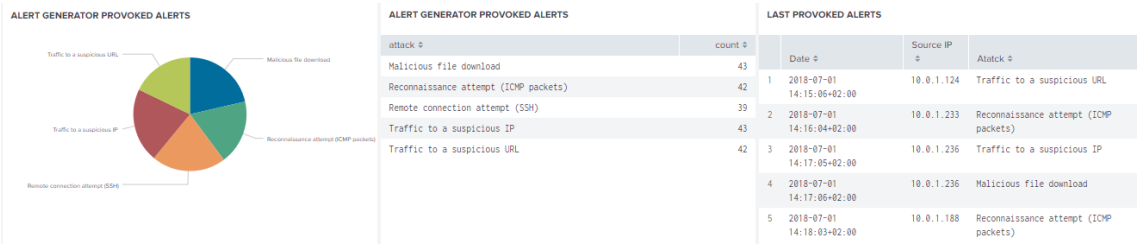


Figure 52. pfSense - Alerts dashboard - Provoked alerts panels

On the central part, detected threats in Suricata are represented. As in the graphics from Alert Generator, we have a chart pie, the count of detected events and a panel with more information with the time, source IP of the event and the action made by firewall (allow or block).

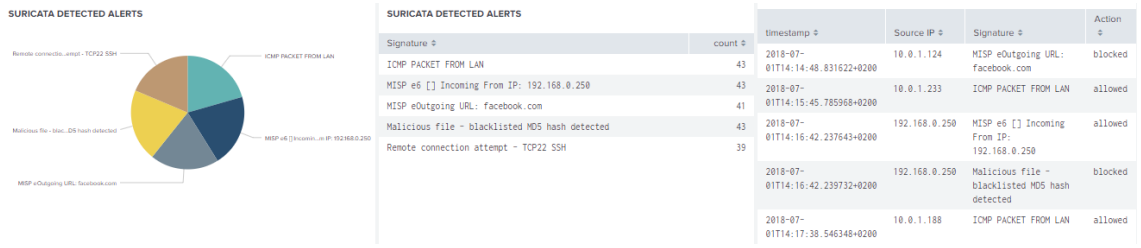


Figure 53. pfSense - Alerts dashboard - Suricata detected alerts panels

On the inferior part, we have a table with detailed information about the alerts triggered by Suricata with extended information as time, GID or Group Identifier (number identifying what subsystem of Suricata generates the event), SID or Signature Identifier (unique number identifying a rule), severity of the alert, category, signature, source and destination IPs, source and destination ports, interface where event was detected, communication protocol and action made by firewall.

timestamp	GID	SID	Severity	Category	Signature	Source IP	Source Port	Source Interface	Destination IP	Destination Port	Protocol	Action
2018-07-01T14:14:48.831622+0200	1	4000011	2	A Network Trojan was detected	MISP eOutgoing URL: facebook.com	10.0.1.124	34624	em2	31.13.90.36	443	TCP	blocked
2018-07-01T14:15:45.785968+0200	1	4000003	3	Detection of a Network Scan	ICMP PACKET FROM LAN	10.0.1.233		em2	8.8.8.8		ICMP	allowed
2018-07-01T14:16:42.237643+0200	1	4000101	1	Potentially Bad Traffic	MISP e6 [] Incoming From IP: 192.168.0.250	192.168.0.250	80	em2+	10.0.1.236	45148	TCP	allowed
2018-07-01T14:16:42.239732+0200	1	4005002	3	Suspicious File download detected	Malicious file - blacklisted MD5 hash detected	192.168.0.250	80	em2+	10.0.1.236	45148	TCP	blocked
2018-07-01T14:17:38.546348+0200	1	4000003	3	Detection of a Network Scan	ICMP PACKET FROM LAN	10.0.1.188		em2	8.8.8.8		ICMP	allowed
2018-07-01T14:18:35.972157+0200	1	4000011	2	A Network Trojan was detected	MISP eOutgoing URL: facebook.com	10.0.1.131	54666	em2	157.240.1.35	443	TCP	blocked
2018-07-01T14:19:32.155725+0200	1	4000102	2	A Network Trojan was detected	Remote connection attempt - TCP22 SSH	10.0.1.218	47654	em2	192.168.0.19	22	TCP	allowed
2018-07-01T14:20:28.647295+0200	1	4000003	3	Detection of a Network Scan	ICMP PACKET FROM LAN	10.0.1.45		em2	8.8.8.8		ICMP	allowed
2018-07-01T14:21:25.916622+0200	1	4000011	2	A Network Trojan was detected	MISP eOutgoing URL: facebook.com	10.0.1.192	53560	em2	31.13.90.36	443	TCP	blocked
2018-07-01T14:22:22.089460+0200	1	4000003	3	Detection of a Network Scan	ICMP PACKET FROM LAN	10.0.1.109		em2	8.8.8.8		ICMP	allowed
2018-07-01T14:23:18.476553+0200	1	4000011	2	A Network Trojan was detected	MISP eOutgoing URL: facebook.com	10.0.1.170	60636	em2	157.240.1.35	443	TCP	blocked

Figure 54. pfSense - Alerts dashboard - Suricata detected alerts panels

Last panel shows detected malicious files downloads with source and destination addresses, filenames, md5 hash, incoming port and action made by firewall.

DETECTED MALICIOUS FILES

timestamp	src_ip	dstip	http_referer	http_url	magic	md5_hash	srcport	state
07/01/2018-14:16:42.239732	192.168.0.250	10.0.1.236	<unknown>	/mensual_budget.pdf		ebaedcb3da39d1fc5fda36bc69cf3470	80	CLOSED
07/01/2018-14:26:08.300520	192.168.0.250	10.0.1.217	<unknown>	/mensual_budget.pdf		ebaedcb3da39d1fc5fda36bc69cf3470	80	CLOSED
07/01/2018-14:27:05.657313	192.168.0.250	10.0.1.51	<unknown>	/mensual_budget.pdf		ebaedcb3da39d1fc5fda36bc69cf3470	80	CLOSED

Figure 55 . pfSense - Alerts dashboard - Suricata detected malicious files

All Splunk searches used for generating dashboards can be found in Appendix.

4. Results

After configuring monitoring system explained in previous section, two performance tests had been performed:

- Low traffic load test
- 20 Mbps traffic load test

Test is performed with a pfSense configuration of 2.4GHz processor single core and 1024MB of RAM memory.

Low traffic load test

In this test, no extra traffic apart from generated with Alert generator has executed.

This is Suricata performance obtained from dashboards:

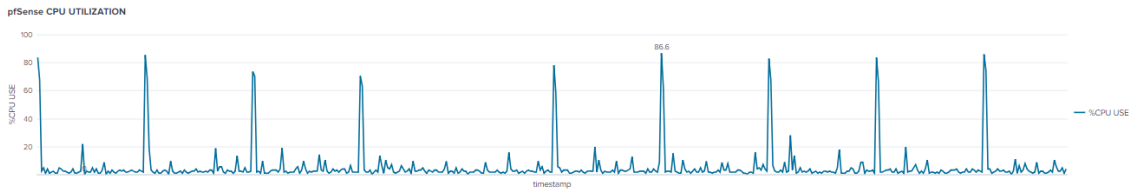


Figure 56. Results with low traffic load - CPU utilization

Figure 57. Results with low traffic load - Available memory

CPU utilization remains below 20% and available memory between range 50-75 Mbytes. Peaks observed are caused by internal Suricata subsystem updates done every 5 minutes.

In packet inspection panel we can observe no packet loss:

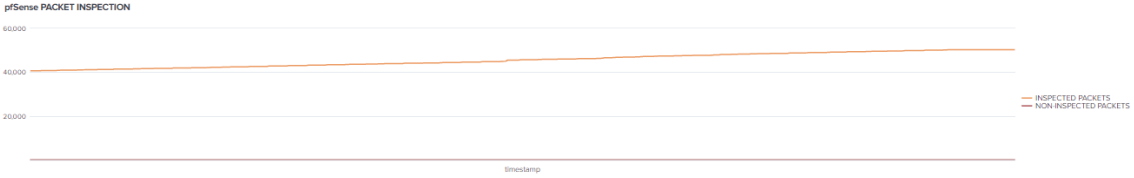


Figure 58. Results with low traffic load - Packets inspected/non-ispected

In Alerts dashboard, we can observe that all generated events where successfully detected by Suricata Intrusion Prevention System, generating corresponding alerts and blocking connections according with defined IPS rules.

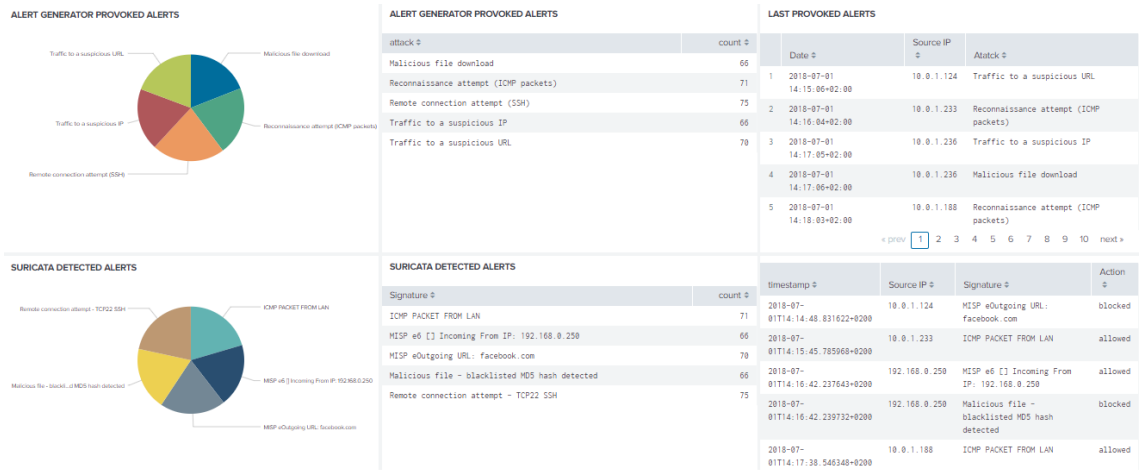


Figure 59. Results with low traffic load - Alerts dashboard

With all results commented, we can determine 100% of success for low traffic conditions test.

High traffic load test

In this test, additional random traffic was generated within the Client1's network for modelling a real environment using traffic injector explained in Generating TCP traffic section.

Next results were obtained from Splunk dashboards:

Below are shown statistics about pfSense performance under high traffic load conditions:

CPU utilization fluctuates between 60-100% while pfSense is receiving and inspecting 20Mbps sustained flow.

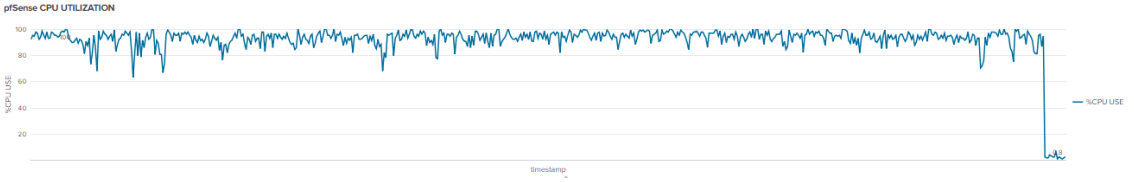


Figure 60. High traffic load test - pfSense CPU utilization

Although CPU utilization is very high (80-99.8%), all packets have been inspected with 0 packet drops (packets going through pfSense without being inspected) of a total of 25.000.000 packets.

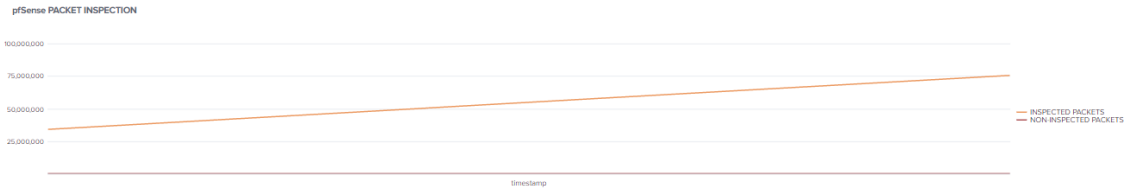


Figure 61. High traffic load test- pfSense inspected/non-inspected packets

Memory never was below 40MB. However, decreasing trend observed indicates that in case of more traffic load, system could run out of memory. Because of this, 2GB RAM memory is recommended memory to provision the pfSense server for a load higher than 20Mbps traffic inspection.

Figure 62. pfSense high traffic load test – Available RAM

Observing Alerts dashboard, next results were obtained:

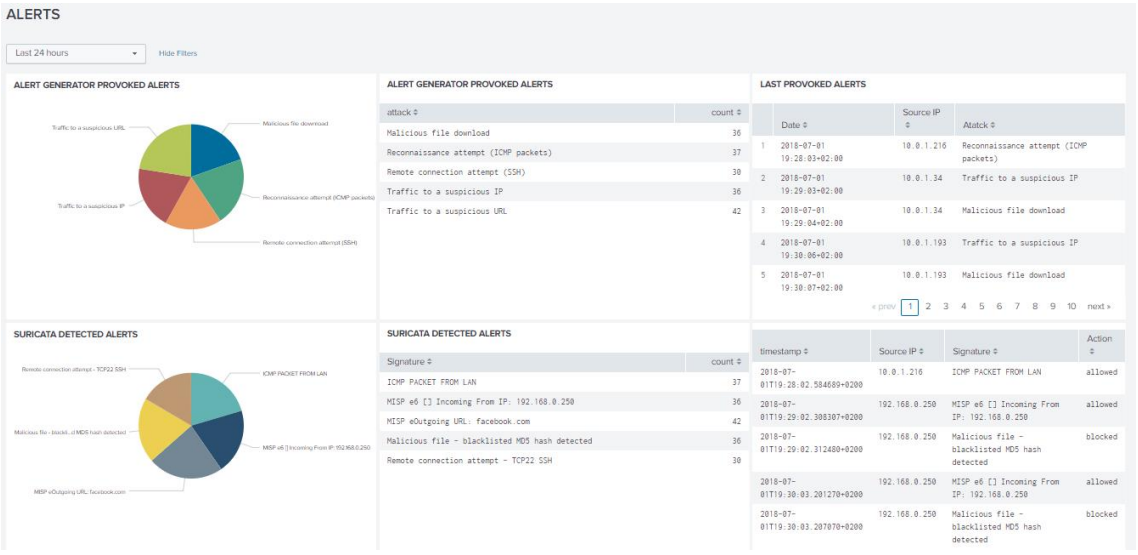


Ilustración 1. pfSense high traffic load test – Alerts dashboard

Number of detected events in Suricata coincides with number of generated alerts with alert_generator script, indicating 100% correct threat detection for the proposed test.

With results in hand, the following conclusions have been drawn:

- Proposed system can be used to secure a network.
- Creating events in MISP is relatively simple and the rules generated with it, although they are not optimal, can serve as valuable assets.
- MISP can serve as security systems feed and not just as source of information to consult and distribute.
- MISP API can be used not only to integrate it with Suricata, but many other vendors products and is possible to develop more features easily.
- MISP can help to get the maximum out of the information about malware and threats without unmanageable complexity.
- MISP network is scalable and the more people use it, the more information is available.
- Suricata IPS/IDS worked as expected and has been demonstrated that is a powerful open-source tool to be used as complement to conventional firewalls.
- Splunk is a consolidated tool for event management and can be used as a SIEM for Security Operation Centers. For low data volumes (500MB per day), is free licensed and the possibilities to big data analysis and representation are endless.

5. Budget

The main expense in the project has been the worktime invested by the author developing the project because all software tools are open source with no cost.

The use of open source tools allows both to reduce the cost of the project and to add high added value on it due to its scalability, flexibility and adaptability depending on needs and ensure the financial viability in exchange for effort to understand and maintain this kind of systems.

Next table shows the budged for the project:

Concept	Description	Units	Price x Unit (€)	Amount (€)
Computer	Computer used for developing the project	1	900	900
Worktime	Invested hours to project development	500	12	6000
Electric power consumption	Hours of electricity consumption during development of the project	500	0,05	25
Splunk License*	Software liecense	0	1700	0
	TOTAL PROJECT COST	**6925€**		

Table 4. Project budget

*Splunk license has been considered as cost 0 due to license is needed only to process more than 500MB/day of data with Splunk and this limit has not been reached.

6. Environment impact

Due to this is a technological project, important direct environment impacts are not visible.

Nevertheless, because the integration of proposed system uses existing resources in companies due to software proposed can be installed to an existing hardware used for other services with sufficient space and processing power, it doesn't suppose an extra cost in terms of energy. Also, the basis of the idea of sharing information between companies induces assossiative and self-sustaining behavior.

7. Conclusions and future development:

After results obtained in tests, has been demonstated that the proposed system can be deployed both in production environments of small and in large companies and that has possibilities for improvement, adaptation and scalation.

All tools used for developing the project are tools that are currently used in many companies and served me to learn and to discover the potential they have.

Contemplating future developments, one interesting implementation could be to inspect encrypted data with HTTPS protocol (decrypting it before Suricata interface like a legal Man in the Middle Attack) and automate the generation of IPS rules from MISP for deep packet inspection using regular expressions and detecting patterns in traffic.

8. Bibliography

- "Misp User Guide", Belgian Ministry of Defence (CERT), CIRCL Computer Incident Response Center Luxembourg, Iklody IT Solutions, NATO NCIRC, Cthulhu Solutions, CERT-EU.
  [Online] Available:
  https://www.circl.lu/doc/misp/
  Accessed on February-June 2018

- "Suricata User Guide", Andreas Herz, Andreas Moe, Anne-Fleur Koolstra, Christophe Vandeplas, Darren Spruell, David Cannings, David Diallo, David Wharton, Eric Leblond, Haris Haq, Ignacio Sanchez, Jason Ish, Jason Taylor, Josh Smith, Ken Steele, Les Syv, Mark Solaris, Martin Holste, Mats Klepsland, Matt Jonkman, Michael Bentley, Michael Hrishenko, Nathan Jimerson, Nicolas Merle, Peter Manev, Philipp Buehler, Rob MacGregor, Russel Fulton, Victor Julien, Vincent Fang, Zach Rasmor.
  [Online] Available:
  http://suricata.readthedocs.io/en/suricata-4.0.4/
  Accessed on May 2018

- Victor Julien, "File extraction in Suricata", Inliniac bloc in 2011.
  [Online] Available:
  https://blog.inliniac.net/2011/11/29/file-extraction-in-suricata/
  Accessed on May 2018

- Snort User Guide
  [Online] Available:
  http://manual-snort-org.s3-website-us-east-1.amazonaws.com/
  Accessed on April 2018

- Saad Hafeez, "Deep Packet Inspection using Snort". *In the Islamia University of Bahawalpur, A Report Submitted in Partial Fulfillment of the Requirements for the Degree of MASTER OF ENGINEERING in the Department of Electrical and Computer Engineering*.

- Shalvi Srivastava, Sweta Anmulwar, Dr.A.M.Sapkal, Tarun Batra, Anil Gupta, Vinodh Kumar "Evaluation of Traffic Generators over a 40Gbps link". *In 2014 Asia-Pacific Conference on Computer Aided System Engineering (APCASE)*.
  [Online] Available:
  https://www.researchgate.net/publication/290304763_Evaluation_of_traffic_generators_over_a_40Gbps_link
  Accessed on April 2018

- Alessio Botta, Alberto Dainotti, Antonio Pescapé. "A tool for the generation of realistic network workload for emerging networking scenarios". *In 2012 University of Napoli Federico II, Department of Computer Engineering and Systems, Via Claudio 21, I-80125*

Napoli, NA, Italy.
[Online] Available:
http://wpage.unina.it/a.botta/pub/COMNET_WORKLOAD.pdf
Accessed on April 2018

- José Llopis Polvoreda, "SISTEMA DE MONITORIZACIÓN DEL IDS SNORT". *In Trabajo Fin de Grado Ingeniería Informática, Universitat Politècnica de Valencia, 2016/2017*.
  [Online] Available:
  https://riunet.upv.es/bitstream/handle/10251/88474/LLOPIS%20-%20Sistema%20de%20monitorizaci%C3%B3n%20del%20IDS%20Snort.pdf?sequence=1
  Accessed on April 2018

- Ostinato User Guide
  [Online] Available:
  https://userguide.ostinato.org/
  Accessed on June 2018

- Splunk User Guide
  [Online] Available:
  docs.splunk.com/Documentation/Splunk/7.0.2/
  Accessed on May-June 2018

9. Appendix

All scripts developed during the project can be found in this section.

Script update_rules.sh

```
#!/usr/local/bin/bash

#This script compares the newests with the previous misp.rules file

#cmp return values: 0-files are identical; 1-files are different; >1 an error occurred


#Variable to control if a previous file exists

file_found=0


#file1 is the route of the new file (last download from misp)

curl -o /usr/local/etc/suricata/misp_rules_last_download/misp.rules --insecure --header
"Authorization: CF0j4VURzbHMdyGFvOlJGxaGHoAejApygFsPIvAJ" --header "Accept:
application/json" --header "Content-type: application/json"
https://10.1.0.2/events/nids/suricata/download

file1=/usr/local/etc/suricata/misp_rules_last_download/misp.rules


#file2 is the route of the md5 hashes of malicious files (last download from misp)

curl -o /usr/local/etc/suricata/misp_hashes_last_download/md5_blacklist.txt --insecure --
header "Authorization: CF0j4VURzbHMdyGFvOlJGxaGHoAejApygFsPIvAJ" --header "Accept:
application/json" --header "Content-type: application/json"
https://10.1.0.2/events/hids/md5/download

file2=/usr/local/etc/suricata/misp_hashes_last_download/md5_blacklist.txt


#suricata main path

suricata_path=/usr/local/etc/suricata


for dir in `ls -d -1 $suricata_path/suricata_*/rules`;do
        file_found=0
        for file in `ls $dir`;do
                rules_file_path=$dir/$file1
```

```
                if [[ "$rules_file_path" == *"misp.rules" ]]; then
                        file_found=1
                        cmp $file1 $rules_file_path
                        #Case new downladed rules file different than old one -> update rules in Suricata -> store a copy in /old directory
                        if [ $? -eq 1 ]; then
                                echo "INFO: Files are different. Update+restart needed."
                                date=$(date +%Y_%m_%d-%H:%M:%S)
                                if [ -d $dir/old/ ]; then
                                        cp $rules_file_path $dir/old/misp_until_update_$date.rules
                                        rm $rules_file_path
                                        cp $file1 $dir/misp.rules
                                else
                                mkdir $dir/old/
                                cp $rules_file_path $dir/old/misp_until_update_$date.rules
                                cp $file1 $dir/misp.rules
                                fi

                                #Reboot suricata rules file. Obtain pids before with script or function...
                                echo "Restarting Suricata..."
                                process_pattern=$(dirname $dir)
                                pid=$(ps aux | awk -v pat="$process_pattern" '$0 ~ pat { print $2;exit; }')
                                echo "Sending restart signal to pid $pid"
                                #echo 'kill -USR2 '$pid

                        else
                        echo "INFO: New rules are equal to old ones. No update+restart needed."
```

```
                        fi
                fi
        done

        #Case if no misp.rule file encountered -> Create the new rule file
        if [ $file_found -eq 0 ]; then
                echo "No misp.rule file existing in directory."
                cp $file1 $dir/misp.rules
        fi
done
```

Script alert_generator.sh

```
#!/bin/bash
#This script generate random alerts simulating users behaivour

#Logging system
#set LOGFILE to the full path desired to store de log. Ensure write permissions there
#set RETAIN_NUM_LINES to the maximum number of lines to be retained at the beginning of program execution
#execute 'logsetup' once at the beginning of your script
#execute 'log' as many times as you liki to log to LOGFILE

#Variables
change_ip=/home/desktop1client1/scripts/change_ip.sh
LOGFILE=/home/desktop1client1/scripts/logs/alert_generator.log
RETAIN_NUM_LINES=100
interfaces_file=/etc/network/interfaces
pfsense_ip=192.168.0.78

#Logging functions
```

```
function logsetup {
TMP=$(tail -n $RETAIN_NUM_LINES $LOGFILE 2>/dev/null) && echo "${TMP}" > $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1
}

function log {
echo "{\"timestamp\":\"$(date --rfc-3339=seconds)\",$*}" >> $LOGFILE
}

#Change IP to simulate specific random user
function change_ip {
new_host=$(shuf -i2-254 -n1) #Radom number between range 2-254
new_ip="10.0.1."$new_host

#sed -i per a fer el replace en el propi input_file
sed -i -E -e 's/(address )(.*)/\1'$new_ip'/g' $interfaces_file
#cat $interfaces_file
ip addr flush enp0s3
systemctl restart networking
}

#Main
logsetup
change_ip
#Select randomly which action user will do to simulate an attack. Radom number between range 1-4
#RANDOM ATTACK
attack_number=$(shuf -i1-4 -n1)
```

```
case $attack_number in
1)
attack='"Reconnaissance attempt (ICMP packets)"'
timeout 3 ping -c1 8.8.8.8 -q &>/dev/null
log '"attack":'$attack',"src_ip":"'$new_ip'\""

;;
2)
attack='"Traffic to a suspicious URL"'
#timeout 3 wget -O /dev/null -q facebook.com
timeout 3 wget facebook.com -o /dev/null
log '"attack":'$attack',"src_ip":"'$new_ip'\""

;;
3)
attack='"Remote connection attempt (SSH)"'
timeout 2 ssh 192.168.0.19
log '"attack":'$attack',"src_ip":"'$new_ip'\""

;;
4)
attack='"Traffic to a suspicious IP"'
#timeout 3 wget http://192.168.0.250 -o /dev/null
timeout 2 wget http://192.168.0.250:80/mensual_budget.pdf -P
/home/desktop1client1/scripts/downloaded_files/ -q > /dev/null 2>&1
log '"attack":'$attack',"src_ip":"'$new_ip'\""
sleep 1
attack='"Malicious file download"'
log '"attack":'$attack',"src_ip":"'$new_ip'\""
esac
```

```
#echo "Resultat final: "$attack

#echo "$new_ip"


sshpass -p 'mispdcapass' scp /home/desktop1client1/scripts/logs/alert_generator.log
root@$pfsense_ip:/var/log/alert_generator
```

Script suricata_performance_stats.sh

```
#!/usr/local/bin/bash

#Logging system setup

#Variables

LOGFILE=/var/log/suricata_performance/suricata_performance.log

RETAIN_NUM_LINES=100

declare -i COUNTER

COUNTER=0

#Logging functions

function logsetup {

        TMP=$(tail -n $RETAIN_NUM_LINES $LOGFILE 2>/dev/null) && echo "${TMP}" >
$LOGFILE

        exec > >(tee -a $LOGFILE)

        exec 2>&1

}


function log {

        echo "{\"timestamp\":\"$(date "+%Y-%m-%dT%H:%M:%S")\",$*}" >> $LOGFILE

}


logsetup


while [ $COUNTER -le 10 ]; do
```

```
        params=$(top -b -d2 -s1 -n 1 | sed -n '13,14p')
        cpu=$(echo "$params" | sed -n '1p' | awk '{print $10}')
        mem=$(echo "$params" | sed -n '2p' | awk '{print $12}')
        idle_CPU=$(echo "$cpu" | sed 's/%//')
        free_MEMORY=$(echo "$mem" | sed 's/M//')
        log '"idle_CPU":"'$idle_CPU'","free_MEMORY":"'$free_MEMORY'"'
        COUNTER=($COUNTER+1)
        sleep 6
done
```

Script misp_install.sh

```
# AUTOMATION SCRIPT FOR MISP INSTANCE DEPLOYMENT
# Dockerized MISP TFG - UNIVERSITAT POLITECNICA DE CATALUNYA

clear
echo -e "\n\nDockerized MISP TFG\n"
echo -e ''' __ __ __ _____ _____\n|  Y  |  |  _  |  _  |\n|.     |. |  1__|. 1  |\n|. \_/  |. |___
|. ___|\n|: |  |: |: 1  |: |\n|::.|:. |::.|::... . |::.|\n`--- ---`---`-------`---´\t
v2.4.81\n=========================================================
================================'''

#Places us in the proper directory for Docker Compose automation
cd /home/misp/docker/misp-docker/

#Making sure no residual containers exist
docker-compose down

#Clean Docker Compose env file
cp env.default.txt env.txt

#Input variables for installation with minimal checks
```

```
while true; do
   read -p "[+] Please enter the MISP instance's IP address and press [ENTER] $(echo $' ')"
ip
   ips=${ip//"."/"\\\\."}
   case $ip in
      *.*.*.* ) sed -i "s=baseurl=http:\\\/\\\/$ips=" env.txt; break;;
      *) echo "[!] Please enter a well formed IP address.";;
   esac
done
while true; do
   read -sp "[+] Please enter the password to be used by the misp mysql user and press
[ENTER] $(echo $' \n')" misppass
   echo -ne "\n"
            case $misppass in
      "" ) echo "[!] Please enter non-blank password";;
       * ) sed -i "s/"misppass"/$misppass/" env.txt; break;;
   esac
done
while true; do
   read -sp "[+] Please enter the mysql root password and press [ENTER] $(echo $' \n')"
rootpass
            echo -ne "\n"
   case $rootpass in
      "" ) echo "[!] Please enter non-blank password";;
       * ) sed -i "s/"rootpass"/$rootpass/" env.txt; break;;
   esac
done


#Places us in the directory where compose expects to find the SSL cert and RSA key.
rm /home/misp/docker/misp-docker/misp/resources/cert/*
cd /home/misp/docker/misp-docker/misp/resources/cert
```

echo "[+] You will now be prompted for the details to be included in the SSL certificate..."

echo "[i] PLEASE NOTE THAT CN (Common Name) SHOULD BE THE IP ADDRESS FOR YOUR MISP (OR FQDN IN CASE YOU HAVE PROVISIONED AN INTERNALLY RESOLVABLE NAME)"

read -n1 -r -p "Press any key to continue" key

openssl genrsa -out misp.key 2048

openssl req -new -key misp.key -out misp.csr -nodes

openssl x509 -req -days 3650 -in misp.csr -signkey misp.key -out misp.crt

docker-compose build

docker-compose up -d

Log file sample eve.json


```
{"timestamp":"2018-07-
01T22:17:02.269971+0200","in_iface":"em2","event_type":"alert","src_ip
":"10.0.1.29","dest_ip":"8.8.8.8","proto":"ICMP","icmp_type":8,"icmp_c
ode":0,"alert":{"action":"allowed","gid":1,"signature_id":4000003,"rev
":1,"signature":"ICMP PACKET FROM LAN","category":"Detection of a
Network
Scan","severity":3},"payload":"vzY5WwAAAAB\/YgcAAAAAABAREhMUFRYXGBkaGx
wdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=","stream":0,"packet":"CAAnd9bZC
AAnW+ZPCABFAABUq11AAEABdB8KAAEdCAgICAgAeItBrAABvzY5WwAAAAB\/YgcAAAAAAB
AREhMUFRYXGBkaGxwdHh8gISIjJCUmJygpKissLS4vMDEyMzQ1Njc=","packet_info":
{"linktype":1}}
{"timestamp":"2018-07-
01T22:16:02.707012+0200","flow_id":1889915373796294,"in_iface":"em2+",
"event_type":"alert","src_ip":"192.168.0.250","src_port":80,"dest_ip":
"10.0.1.201","dest_port":34956,"proto":"TCP","tx_id":0,"alert":{"actio
n":"blocked","gid":1,"signature_id":4005002,"rev":1,"signature":"Malic
ious file - blacklisted MD5 hash detected","category":"Suspicious File
download
detected","severity":3},"http":{"hostname":"192.168.0.250","url":"\/me
nsual_budget.pdf","http_user_agent":"Wget\/1.19.1 (linux-
gnu)","http_content_type":"application\/pdf","http_method":"GET","prot
ocol":"HTTP\/1.1","status":200,"length":21},"app_proto":"http","payloa
d":"SFRUUC8xLjAgMjAwIE9LDQpTZXJ2ZXI6IFNpbXBsZUhUVFAvMC42IFB5dGhvbi8zLj
YuMw0KRGF0ZTogU3VuLCAwMSBKdWwgMjAxOCAyMDoxNjowMiBHTVQNCkNvbnRlbnQtdHlw
ZTogYXBwbGljYXRpb24vcGRmDQpDb250ZW50LUxlbmd0aDogMjENCkxhc3QtTW9kaWZpZW
Q6IEZyaSwgMjIgSnVuIDIwMTggMjE6MDY6MTAgR01UDQoNCnRoaXMgaXMgbWFsaWNpb3Vz
Li4uCg==","stream":1,"packet":"CAAnW+ZPCAAnd9bZCABFAABJCRxAAD8GZSjAqAD
```

6CgAByQBQiIy6j2DsvIBXJoAZAOuaQwAAAQEICp4lxYmzGnXQdGhpcyBpcyBtYWxpY2lvd
XMuLi4K","packet_info":{"linktype":1}}
{"timestamp":"2018-07-
01T22:08:02.758064+0200","flow_id":1316983884932099,"in_iface":"em2","
event_type":"drop","src_ip":"10.0.1.7","src_port":35444,"dest_ip":"157
.240.1.35","dest_port":443,"proto":"TCP","drop":{"len":378,"tos":0,"tt
l":64,"ipid":41305,"tcpseq":531130477,"tcpack":1830640807,"tcpwin":229
,"syn":false,"ack":true,"psh":true,"rst":false,"urg":false,"fin":false
,"tcpres":0,"tcpurgp":0},"alert":{"action":"blocked","gid":1,"signatur
e_id":4000011,"rev":1,"signature":"MISP eOutgoing URL:
facebook.com","category":"A Network Trojan was
detected","severity":2}}

Log file sample Suricata_performance.log

{"timestamp":"2018-07-
01T22:12:57","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:01","idle_CPU":"97.6","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:04","idle_CPU":"96.9","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:08","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:11","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:15","idle_CPU":"93.7","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:22","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:29","idle_CPU":"96.9","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:36","idle_CPU":"96.9","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:43","idle_CPU":"97.7","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:50","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:13:57","idle_CPU":"97.7","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:01","idle_CPU":"96.9","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:04","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:08","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:11","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:15","idle_CPU":"99.2","free_MEMORY":"102"}

{"timestamp":"2018-07-
01T22:14:22","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:29","idle_CPU":"69.5","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:36","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:43","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:50","idle_CPU":"95.3","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:14:57","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:01","idle_CPU":"21.9","free_MEMORY":"103"}
{"timestamp":"2018-07-
01T22:15:04","idle_CPU":"63.3","free_MEMORY":"103"}
{"timestamp":"2018-07-
01T22:15:08","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:11","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:15","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:22","idle_CPU":"98.4","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:29","idle_CPU":"92.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:36","idle_CPU":"96.1","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:43","idle_CPU":"86.7","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:50","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:15:57","idle_CPU":"97.6","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:16:01","idle_CPU":"99.2","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:16:04","idle_CPU":"97.7","free_MEMORY":"102"}
{"timestamp":"2018-07-
01T22:16:08","idle_CPU":"98.4","free_MEMORY":"102"}

10. Glossary

Next, a list of all acronyms appearing in this document can be found.

CA: Certificate Authority

C&C server: Command and Control server

CIRCL: Computer Incident Response Center Luxembourg

CSV: Comma Separated Values

DHCP: Dynamic Host Configuration Protocol

DMZ: Demilitarized Zone

DNS: Domain Name System

GUI: Graphical User Interface

LAN: Local Area Network

FW: firewall

GW: gateway

HIDS: Host Intrusion Detection System

IDS: Intrusion Detection System

IOC: Indicator of Compromise

IPS: Intrusion Prevention System

ISO: International Organization for Standarization

IT: Information Technlogy

MISP: Malware Information Sharing Platform

NAT: Network Address Translation

NATO: North Atlantic Treaty Organization

NCIRC: NATO Computer Incident Response Capability

NIDS: Network Intrusion Detection System

NIPDS: Network Intrusion Prevention and Detection System

NTP: Network Time Protocol

OISF: Open Information Security Foundation

OS: Operating System

OSI model: Open Systems Interconnection model

OVA: Open Virtual Appliance

SIEM: Security Information and Event Management

SNMP: Simple Network Management Protocol

SOC: Security Operation Center

VM: Virtual Machine

WAN: Wide Area NetworkA list of all acronyms and the meaning they stand for.