Ref. Ares(2020)2455137 - 08/05/2020

SPEAR

Secure and PrivaTE smArt gRid

(Grant Agreement No 787011)

D4.5 – SPEAR Smart Grid Database & Interfaces

2020-04-30

Version 1.1

Published by the SPEAR Consortium
Dissemination Level: Public

This project has received funding from the European Union’s Horizon 2020 research and innovation programme under Grant Agreement No 787011

Document Control Page

Document Details

Document Version	1.1
Document Owner	European Dynamics (ED)
Contributors	ED, REAL, UOWM, LUH
Work Package	WP 4 – Forensic Readiness and Privacy-Preserving
Deliverable Type	O
Task	Task 4.5 Distributed Database & Communication Interface
Document Status	Final
Dissemination Level	Public

Document History

Version	Author(s)	Date	Summary of changes
0.1	European Dynamics, REALAIZ	2019-12-04	Table of contents
0.3	European Dynamics, REALAIZ	2020-02-17	Ready for review by internal reviewers
0.4	European Dynamics, REALAIZ	2020-02-23	Updated version based on internal reviewers comments, ready to be submitted
1.0	European Dynamics, REALAIZ	2020-03-09	Final corrections on report. Ready for submission
1.1	European Dynamics, REALAIZ	2020-04-30	Final version addressing UOWM review comments

Internal Review History

Reviewed By	Date	Summary of Comments
Antonios Sarigiannidis (SH)	2020-02-27	Accepted with reservation. Comments to be addressed.
Igor Kotsiuba(PIMEE)	2020-02-14	Accepted with reservation. Comments to be addressed.

Table of Contents

List of Figures

List of Tables

Acronyms

Acronym	Description
ASN	Autonomous System Numbers
ELK	Elasticsearch, Logstash, and Kibana
ESM	Enterprise Security Monitoring
European Dynamics	ED
GDPR	General Data Protection Regulation
HTTPS	Hypertext Transfer Protocol Secure
IPsec	Internet Protocol Security
LUKS	Linux Unified Key Setup
NSM	Network Security Monitoring
NTP	Network Time Protocol
OSSIM	Open Source Security Information and Event Management
SG	Smart Grid
SIEM	Security Information and Event Management
SOCs	Security Operations Centres
SPEAR	Secure and PrivatE smArt gRid
SPEAR-ELK	SPEAR Elasticsearch, Logstash, and Kibana
SPEAR-FR	SPEAR Forensics Repository
SSH	Secure Shell
TLS	Transport Layer Security
UTC	Coordinated Universal Time
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
EVTX	Windows XML EventLog

Executive Summary

Given the large amounts of security event data (log files, flow-data, full content data and statistical data) required to be processed when conducting network forensic investigations and the computational resources needed to consume, parse and analyse them it becomes apparent that a big data storage and analysis platform is needed. This can be a whole distributed environment, given the large amounts of data to process and may require multiple external storage devices, to handle the large storage requirements. Commercial platforms that are up to the task are usually too expensive to be adopted by small- and medium-sized enterprises/organisations.

This document summarizes the logging architecture, presented in D4.2, which is able to support the smart grid network forensics process, by a) implementing an effective and secure storage for smart grid network forensic data and b) ingesting, processing, and analysing stored smart grid network forensic data.

Using multiple and distributed security probes in the network, will allow us to acquire smart grid network forensic data (log files, flow-data, full content data and statistical data) that will eventually be stored in a secured forensic evidence server, for long-term storage, namely the SPEAR Forensics Repository (SPEAR-FR). SPEAR-FR was built on top of open-source components such as cryptsetup[1,2] an open-source utility that allows creation of encrypted volumes based on dm-crypt[3] and Linux Unified Key Setup (LUKS), syslog-ng[4], softflowd[5], nfdump[6] and nfsen[7] toolset.

Regarding querying and analytics, this document describes how ELK[8] Stack, short for Elasticsearch, Logstash, and Kibana was incorporated into SPEAR, allowing us to ingest smart grid network forensic data stored in the SPEAR-FR, isolate and finally analyse them, thus addressing the computational resources needed during investigations and the time delays in processing log files. This pre-configured ELK Stack appliance of SPEAR-FR, provides all system administration and system engineering components, allowing forensic investigators to focus on the important aspects of any forensic task, which is to apply own intelligence and awareness when analyzing collected security event data.

[1] https://gitlab.com/cryptsetup/cryptsetup
[2] https://linux.die.net/man/8/cryptsetup
[3] https://gitlab.com/cryptsetup/cryptsetup/-/wikis/DMCrypt
[4] https://github.com/balabit/syslog-ng
[5] https://github.com/irino/softflowd
[6] https://github.com/phaag/nfdump
[7] http://nfsen.sourceforge.net/
[8] https://www.elastic.co/what-is/elk-stack

1 Introduction

One of the main goals of WP4 is to address the problem of secure storage of forensic data, the need of computational resources for investigation and the time delays in processing log files, required for cyber attacker attribution.

The logging architecture, presented in D4.2, and its implementation described in this document addresses the above requirements by: a) safely collecting and storing smart grid network forensic data in a dedicated remote centralized archival storage, for as long as possible, namely the **SPEAR Forensics Repository** (SPEAR-FR), and b) ingesting, processing, and analysing stored smart grid network forensic data. The latter is based on ELK Stack, a big data storage and analysis platform, that is incorporated into SPEAR and also described in this deliverable. ELK Stack continuously gains popularity due to its scalability and open-source components. Countless forensic teams and security operations centres (SOCs) are incorporating ELK Stack into their production environments, as it offers them a powerful platform capable to collect and process data from multiple data sources, while providing a set of tools to analyze collected data. Ingested data are stored in a centralized data store that can scale as data grows.

In order to support the task of cyber attacker attribution, Network Security Monitoring (NSM) tools[9] are used to provide context, intelligence and situational awareness of the monitored network. Although security event data can be collected and analysed, not all malicious activities can be identified even by Enterprise Security Monitoring (ESM) tools[10], including the four-layer SPEAR Security Information and Event Management (SIEM). While automation and correlation can enhance intelligence and assist us in identifying threat indicators, there is no replacement for human intelligence and awareness.

1.1 Deliverable Structure

The deliverable is structured in nine chapters:

- Chapter 1 is the introduction of the document.
- Chapter 2 analyses the system requirements related to secure transmission, storage and access of forensic data.
- Chapter 3 presents the architecture and related components, by defining and describing the decomposition of the SPEAR-FR system into components (ARCADE Component viewpoint [5]).
- Chapter 4 presents the details of the prototype implementation of both the SPEAR-FR and the ELK Stack of SPEAR-FR including installation prerequisites, software repositories and the various dashboards available in Kibana that can be used to support the analysis phase of the OSCAR methodology. A full description of the OSCAR methodology can be found in [3] page 17-22.
- Chapter 5 presents the unit tests done to demonstrate requirements compliance, in line with the SPEAR assessment methodology defined in D2.3.
- Chapter 6 draws conclusions.

1.2 Relation to other Tasks and Deliverables

The following deliverables support this deliverable:

[9] https://linuxsecurity.expert/security-tools/network-security-monitoring-tools
[10] https://www.dnsstuff.com/siem-tools

- D2.2 System Specifications and Architecture, which defines the functional and non-functional requirements of the SPEAR Platform, including the SPEAR Forensic Repository. It also presents the SPEAR Platform architecture.
- D2.3 Evaluation Strategy, which provides the assessment framework for the proposed SPEAR system, including guidelines for evaluating the security and privacy tools that will be developed within SPEAR project.
- D4.1 Forensic Law and Regulations, which defines the forensics strategies in SPEAR and identifies all the appropriate regulatory requirements for the SPEAR Forensic Readiness Framework (SPEAR FRF) including those related to the collection, preservation, and use of digital evidence sources.
- D4.2 Smart Network Forensics Specifications, which provides a smart grid network forensics methodology that is the result of incorporating the OSCAR methodology and relevant open source tools in order to ensure that necessary smart grid forensic information (evidence) can be collected, stored and used as legal evidence in court. The result of applying the suggested smart grid network forensics methodology to each SPEAR pilot, is that it allows them to become forensic ready by: a) identifying sources of evidence, b) prioritizing their collection, c) planning their acquisition, d) identifying how evidence will be transferred to the SPEAR-FR, including transfer protocols and procedures, e) identifying how evidences will be made available to a forensic investigator to support their analysis.
- D4.4, which proposes and develops a Privacy-Preserving Framework (PPF), based on the criteria of the Article 29 Data Protection Working Party (WP29) guidelines [1], [2] and compatible with the international standards on risk management (such as [ISO 31000]).

2 Analysis of SPEAR Forensic Repository Requirements

From D2.2 the functional and non-functional requirements of the SPEAR Forensic Repository. Table 2-1 shows these requirements and how they are addressed/realised by the prototype implementation of the SPEAR-FR which is the outcome of D4.5.

Table 2-1: Coverage of SPEAR requirements on Forensic Repository

Req ID	Description	How addressed
F39	**Forensic Data Collection** – The SPEAR platform must collect necessary forensic data to support forensic investigations	SPEAR-FR supports collection of: • Full content data using tcpdump[11], after configuring the switch "port mirroring" to forward all network traffic to SPEAR-FR. • Session data (router network flow statistics) using different probes. Probes usually come implemented in the operating system of the routers. If this is the case, they will be configured to export network flows to the nfdump collector that runs on the SPEAR-FR. However, to support scenarios (Hydro Power Plant Scenario) where the routing equipment cannot export network flows, the switch "port mirroring" will be used to forward all network traffic to the SPEAR-FR where the softflowd software probe is installed and configured to generate and export network flows to the nfdump collector. • Log files from the identified smart grid assets (clients) that are likely to relate to the investigation, through a syslog-ng server that accepts messages from authorized syslog-ng clients. • Security events published on the message bus, generated from the WP3 components, namely OSSIM Server, big data analytics and visual-aided IDS components, by creating a Kafka consumer and subscribing to the desired topics. Messages read are stored in a text file.
F40	**Forensic Data Transmission** – The network/transport protocol used for transferring the forensic data, it has to: a) be secured against eavesdropping, b) protect the integrity of the forwarded data against manipulation or lost messages and c) be able to deal with network outages	Regarding transferring of network flows a secured (Internet Protocol Security (IPsec), virtual private network (VPN) tunnel) line between the probe and the collector is used. Regarding transferring of pcap files[12] this is addressed through the use of secure protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Regarding transferring of logs from the identified smart grid assets (clients), that are likely to relate to

[11] https://www.tcpdump.org/
[12] https://fileinfo.com/extension/pcap

		the investigation, to the central forensics repository (SPEAR-FR), they will be protected against eavesdropping, by incorporating Transport Layer Security (TLS) certificates. Using certificates to authenticate both the syslog server and the clients will allow for mutual authentication. Regarding support for reliability and ability to deal with network outages, this is addressed through syslog-ng disk-based buffering that ensures reliability, while syslog-ng relays ensure logs buffering for short term, a few minutes or a few hours long outages (depending on the log volume).
F41	**Forensic Data Storage** – The SPEAR Forensic Repository must securely store collected forensic data	The received forensic data are securely stored in an encrypted disk, by creating a dm-crypt LUKS container, through the cryptsetup utility, thus preventing un-authorized disclosure of forensic data. The integrity of data at rest, which assures that the data have not been tampered, can be ensured via: a) appropriate network architecture ensuring that the subnet where SPEAR-FR is located is accessible only by specific users and IPs, supported by dedicated firewall rules, b) strict file and folder permissions (possibly backed up by Access Control Lists), c) dedicated Host Intrusion Detection System (HIDS) agents for File Integrity Monitoring (FIM)[13].
F42	**Forensic Data Access** – Access to the forensic data stored in the SPEAR-FR should be controlled	This is addressed through a well-defined and strict policy on how to decrypt the repository and make it available to the forensic investigator. This should be backed-up by a legal document (contract) that legally binds the investigator from releasing any private information found within the repository
F43	**Availability of forensic data** – SPEAR-FR must ensure the availability of forensic information; otherwise unavailability of data can become problematic, leading to overall service unavailability or degradation as the data owner is unable to access forensic data.	This is addressed through regular backups of SPEAR-FR data locations that come in the form of cryptographically verifiable copies. At the same time we should ensure that the restoration procedures work as expected.
F44	**Forensic Data Timeline** – SPEAR should address the problem of time skew between servers and the problem of timestamp format, to allow investigators build a comprehensive timeline	This is addressed by: a) synchronizing clocks on all systems to Network Time Protocol (NTP) or a similar system, b) standardizing time formats as much as possible, c) including complete, high-precision timestamps (full four-digit year) with time zone information, in the form of an offset, not the name of the time zone and d) normalizing timestamps to Coordinated Universal Time (UTC) as early as possible in the log chain.

[13] https://cybersecurity.att.com/documentation/usm-appliance/ids-configuration/file-integrity-monitoring.htm

F45	**Data Protection Impact Assessment (DPIA)** – DPIA is a process designed to describe the processing, assess the necessity and proportionality of a processing and to help manage the risks to the rights and freedoms of natural persons resulting from the processing of personal data (by assessing them and determining the measures to address them)	DPIAs are important tools for accountability, as they help controllers not only to comply with requirements of the General Data Protection Regulation (GDPR), but also to demonstrate that appropriate measures have been taken to ensure compliance with the Regulation. The DPIA methodology followed in SPEAR is presented in deliverable D4.4, identifying the data processing activities, including the purpose of the processing, types of personal data stored, data retention periods and security measures implemented.

3 SPEAR Forensic Architecture

3.1 Architecture Overview

As already presented in D4.2 the logging architecture is based on the UK National Cyber Security Centre four step program for an effective logging capability to support the network forensics task [4]. The resulting distributed architecture is shown in Figure 3-1 and allows us to: a) safely collect and store smart grid network forensic data in a dedicated remote centralized archival storage, for as long as possible, namely the **SPEAR Forensics Repository** (SPEAR-FR) and b) ingest, process, and analyse stored smart grid network forensic data.

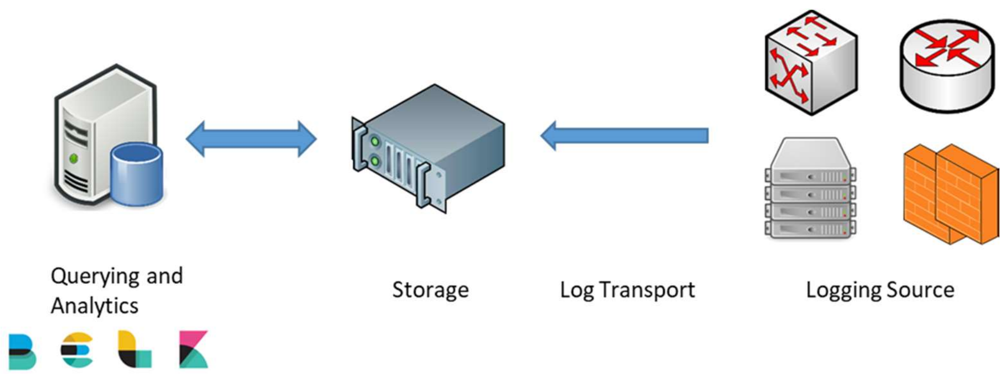


Figure 3-1: Centralised Logging Architecture

As shown above, identified smart grid assets (clients) that are likely to relate to the investigation, transmit forensic data to the forensics repository. The transport protocol is dictated by the logging source and the service that ingests the forensic data. Transferring of network flows is done over a secured line between the probe and the collector. Transferring of pcap files is done over secure protocols, such as SSH. Transferring of logs from the identified clients, is done using syslog-ng and over an encrypted (TLS Encryption) and authenticated (Mutual Authentication) channel between the clients and the server. Moreover, syslog-ng disk-based buffering is employed to ensure reliability, while syslog-ng relays ensure logs buffering for short term, a few minutes or a few hours long outages (depending on the log volume).

Received forensic data are securely stored in an encrypted filesystem, by creating a dm-crypt LUKS container, though the cryptsetup open source utility.

Regarding querying and analytics, it is based on ELK Stack, it authenticates users and allows searches to be performed on the smart grid network forensic data set.

More details about the components are described in the sections below.

3.2 Component Model

3.2.1 Storage

SPEAR-FR collects the following forensic data:

- Full content data, through tcpdump that is based on libpcap[14] software library. This is achieved by configuring the switch "port mirroring" to forward all network traffic to SPEAR-FR.
- Session data (network flow statistics), through an nfdump collector which is running on SPEAR-FR. This is achieved by: a) configuring routers to export network flows statistics to the nfdump collector, or b) by configuring softflowd software probe to generate and export network flows statistics to the nfdump collector.

[14] https://sourceforge.net/projects/libpcap/

- Log files, through a syslog-ng server that accepts messages from authorized syslog-ng clients.
- Security events published on the message bus, through a Kafka consumer that stores read messages on SPEAR-FR.

All received smart grid network forensic data are securely stored in an encrypted filesystem, on the SPEAR-FR server, by creating a dm-crypt container with LUKS extension, though the cryptsetup open source utility, thus preventing un-authorized disclosure of data.

3.2.2 Querying and Analytics

As already presented, the querying and analytics is based on ELK Stack. The following section includes a description of the ELK Stack core components, core data sources, directories that Filebeat inputs are crawled for new smart grid network forensic data. It also presents the importance of data normalisation and data enrichment.

3.2.2.1 Core Components

ELK Stack is a distributed (cluster-based) easily horizontally scalable data storage, indexing, and searching platform. We should think it as a document centric database. It is based on three primary components: a) **Elasticsearch**[15] component, which is a large, cluster capable, storage, full-text search and analysis engine that is based on Apache Lucene, b) **Logstash**[16] which is the ingest engine that reads in, adds and manipulates the data that we are parsing, c) **Kibana**[17] that provides a web-based dashboard for visualising and interacting with the collected security event data. On top of that, we have log shippers, called **Beats**[18]. These are lightweight agents (tiny pieces of code), installed on edge hosts, that are designed to ship one specific kind of data. In SPEAR we will use **Filebeat**[19] that is designed to ship logs and generally file contents.

Since all security event data are located in the SPEAR-FR and since within SPEAR we perform post-incident forensics, the following classic architecture of ELK is used.

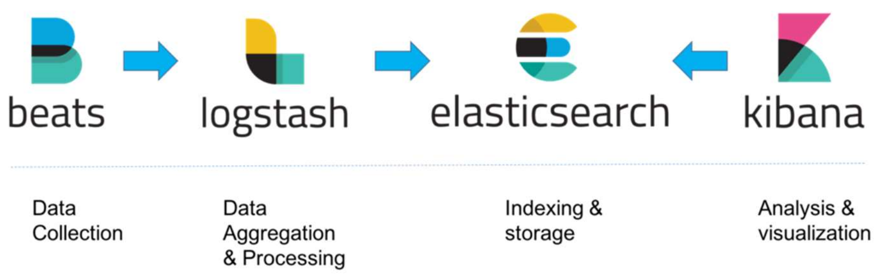


Figure 3-2: ELK architecture[20]

If we need to have a full production-grade instance this will require additional technologies such as Kafka, RabbitMQ, or Redis for resiliency nginx for security, multiple Elasticsearch nodes, multiple Logstash instances, an archiving mechanism, an alerting plugin and a full replication for high availability.

However, as Logstash requires JVM to run, and this can cause significant memory consumption, we should make use of Logstash monitoring API[21] or the monitoring UI within Kibana.

[15] https://www.elastic.co/elasticsearch/
[16] https://www.elastic.co/logstash
[17] https://www.elastic.co/kibana
[18] https://www.elastic.co/products/beats
[19] https://www.elastic.co/products/beats/filebeat
[20] https://logz.io/learn/complete-guide-elk-stack/
[21] https://www.elastic.co/guide/en/logstash/current/monitoring-logstash.html

3.2.2.2 Data Sources

Core data sources include:

- **Log data**. Although the goal is to ingest/parse as many as possible log data formats, due to the fact that there is a large variety of them, this suggests that it won't be possible to accommodate every single one. Since we are focusing on those log formats that are going to help us out, in typical types of forensic analysis workflow, we focused on:
  - **Syslog**. Here we should mention that the difficulty is not just the syslog format, which is a semi-standardized log format, but parsing various different SSH log types, NTP logs and Dynamic Host Configuration Protocol (DHCP)/Domain Name System (DNS) log types. ELK Stack of SPEAR-FR solves this problem by incorporating multiple parsers, while also being able to handle web server logs in a number of different formats, whether it comes directly from a web or proxy server.
- **NetFlow**. ELK Stack of SPEAR-FR incorporates native NetFlow v5 visibility.

Regarding windows logs, such as Windows XML EventLog (EVTX) files, these are ingested/parsed by forwarding them through a syslog pipeline, using snare[22].

For all of those major data sources ELK Stack of SPEAR-FR can ingest them in two different fashions:

a) First of all **live data**, such as when we have a security operations use case. Live data comes in the form of syslog transactions, Filebeat or Elastic Beat transactions.
b) Secondly and most important for our study, it supports the forensic use case. To that extend the ELK Stack of SPEAR-FR can consume **existing files**. This means that security event data safely stored in the SPEAR-FR can be consumed without the need to have a live aggregator operational during the period of interest or during a compromise. This is not always going to be feasible, otherwise we wouldn't have post-incident forensics!

3.2.2.3 Loading Data

For regular logs, we should drop the files into the proper directory. This directory depends on the data that we are feeding, hence:

- Syslog-formatted data should be placed in the */logstash/syslog/* directory. However, we have to keep in mind that since syslog does not reflect the year of a log entry, logstash has been configured to look for a year value in the path of the file.
- Apache logs in common, combined, or vhost-combined formats should be placed in the */logstash/httpd/* directory
- Logs from the Passive DNS utility should be placed in the */logstash/passivedns/* directory.
- CSV files generated by the **Plaso**[23] should be placed in the */logstash/plaso/* directory.

Once files are placed, Filebeat inputs are always looking for these locations, trying to pick up any new logs that show up.

On the other hand, if we are looking at archived NetFlow output, we have to do some post-processing steps. Unfortunately, NetFlow output comes in a post-incident format and as such requires parsing. To ingest *nfcapd*[24] NetFlow log storage format, we have developed a helper script that should be used prior to placing them into the */logstash/nfarch/* directory.

To load pcap files like before we have to do some post-processing steps. To ingest pcap files we should first distil the pcap to nfcapd-compatible NetFlow data. Then we should use nfdump command to create an

[22] https://www.snaresolutions.com/
[23] https://github.com/log2timeline/plaso
[24] https://www.systutorials.com/docs/linux/man/1-nfcapd/

ASCII output in a format the ELK Stack appliance can understand. The ASCII output like before should then be placed in the */logstash/nfarch/* directory for parsing.

3.2.2.4 Data Ingest

Since data come in very different formats, it should be normalised, to allow for structured queries. This comes in the form of a standardised naming scheme across all of our log sources. A proposed field name standardisation is presented here[25]. The fact that we can now query all of our data sources with the same field, provides a consistent use case that in turns enables flexible dashboards and easier queries.

A big benefit that Logstash provides is **data enrichment**[26]. Therefore, for each IP address the ELK Stack appliance adds GeoIP data as well as Autonomous System Numbers (ASN) data[27], or network owners, from local databases. Therefore, we are not doing any off system look ups. ELK Stack appliance also supports dynamic field creation that adds an extract value to the data, by being able to query in a slightly different way. All of these enrichments are done at run/query time, which means that they do not take extra space in the databases. For example, we can have a new dynamic field called *total_bytes* that is the result of *source_bytes* + *destination_bytes*.

Document tagging is another Logstash feature, allowing each record in the ElasticSearch database, which is called document, to receive tags of any numbers (an array of full tags). These tags are helpful because they can tell us how that record was parsed, enabling troubleshooting, load times categorization and easy filtering. Basically, it traces its way through the parsing pipeline.

3.3 Interfaces Model

Since SPEAR-FR does not interact with other SPEAR components, there are no interfaces to present. However, if components need to communicate with the ELK Stack components of SPEAR-FR, relevant APIs should be used. Their documentation is available below:

a) Elasticsearch REST-API (https://www.elastic.co/guide/en/elasticsearch/reference/6.7/release-notes-6.7.2.html)
b) Kibana REST-API (https://www.elastic.co/guide/en/kibana/6.7/api.html)
c) Logstash REST-API (https://www.elastic.co/guide/en/logstash/6.7/monitoring.html)

[25] https://www.elastic.co/guide/en/ecs/current/ecs-guidelines.html
[26] https://www.elastic.co/guide/en/elasticsearch/reference/master/ingest-enriching-data.html
[27] https://www.elastic.co/guide/en/logstash/current/lookup-enrichment.html

4 Prototype Implementation

4.1 Storage

4.1.1 Prerequisites and Installation

Regarding the storage of forensic data the hardware and operating system prerequisites are:

- A 2-core processor
- 4GB RAM Memory
- 500GB of disk space or more

The software prerequisites include:

- Centos 7 Operative System (OS)
- self-signed certificates to be able to encrypt syslog data in transit using TLS encryption
- softflowd
- syslog-ng
- nfdump
- Kafka consumer

Installation and configuration scripts are provided in the unit testing section.

4.2 Querying and Analytics

4.2.1 Prerequisites and Installation

To support querying and analytics the hardware and operating system prerequisites are:

- Centos 7 Operative System
- At least a 4-core processor
- 16GB RAM Memory

However, if we are to deploy the ELK Stack of SPEAR-FR in a production environment we should choose a machine with 64GB of RAM, a couple of Terabytes HDD and 24-cores that eventually should allow us to reach larger numbers (reports indicate at least half-a-billion records). So it becomes apparent that in terms of scalability it all depends on the available infrastructure. Of course there is always the possibility to have a cluster allowing us to reach even higher numbers.

4.2.2 Repository

The pre-configured ELK Stack appliance of SPEAR-FR can be found at a private github repository managed by European Dynamics: https://github.com/european-dynamics-rnd/spear-elk.

4.2.3 Dashboards

To support the analysis phase of the OSCAR methodology presented in D4.2 we use Kibana, that allows to visualize, search, monitor and interact with our data across the Elastic Stack.

4.2.3.1 Syslog Dashboard

Below we present the syslog dashboard that allows us to visualise and search into our syslog data and enable us to drill down into details. On top, we see the number of events per time unit and this is dynamic. Of course, we can zoom in any time of interest. In the second row the two pie graphs, since we are parsing syslog data, they contain the source host name, as well as the program and those are going to be identified based on what the syslog log event is (two fields standardised in the syslog interface).

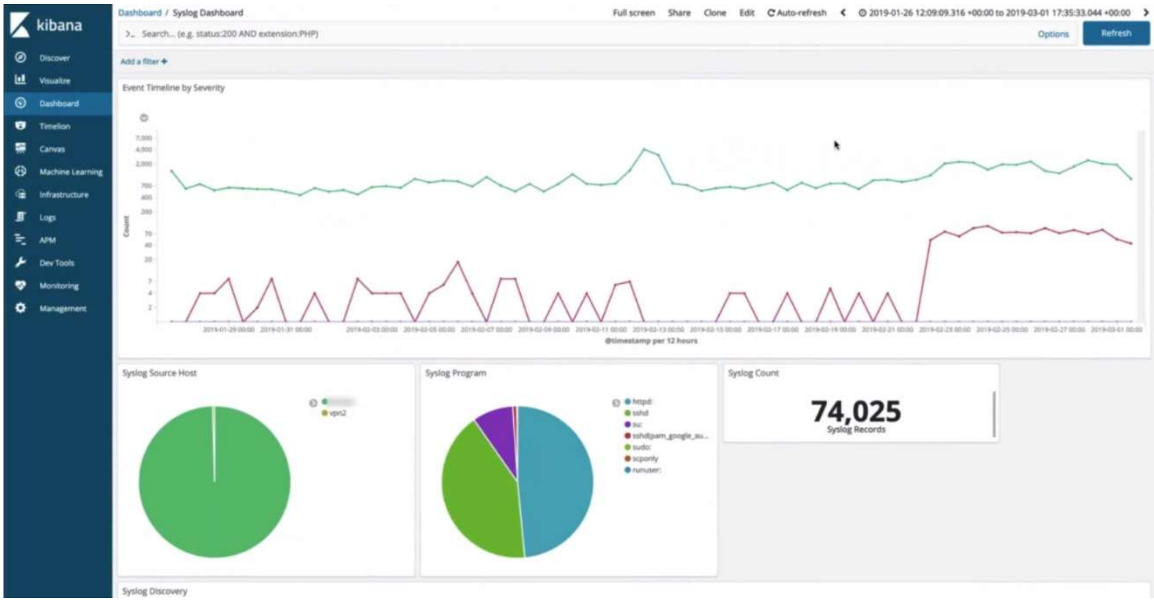


Figure 4-1: Log specific dashboard (syslog dashboard)

As already mentioned dashboards are interactive and Elastic provides means to interact with them. As we hover the mouse over those slices of the pie charts, it will actually provide immediate feedback on what the content is and how much of it there is. Therefore, in the figure below, we see that hovering over the "su" log source program, we have little over six thousand entries, coming from that log source, which is almost 9%.

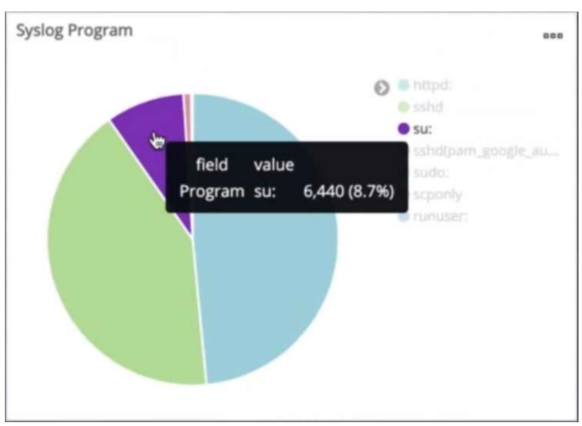


Figure 4-2: Kibana Dynamic and Interactive Dashboards

If we now click on that slice all the way at the very top of the dashboard, it is going to actually build one of these filter boxes. It is going then to filter content, thus providing us with a quick way to filter through millions of entries.

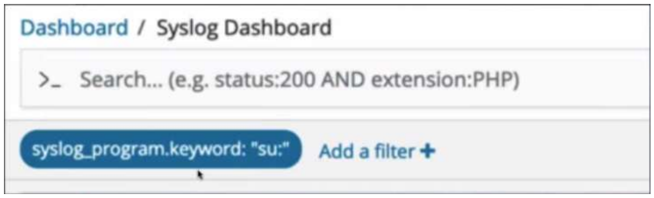


Figure 4-3: Kibana filter

However, hovering over the filter expression, it now presents 5 different icons, as shown below. The check box allows to enable/disable that filter, the pin icon determines whether this filter will last across all different dashboards, the minus magnifying glass is going to invert the filter, the trash icon allows to delete the filter and the edit icon, allows to edit/tweak the filter expression.

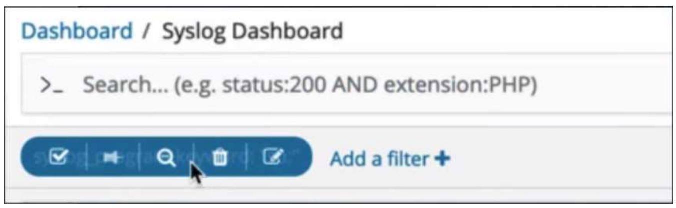


Figure 4-4: Kibana filter box

Inverting a filter is a helpful function as it allows to partition out what we have just been looking at and look at everything else.

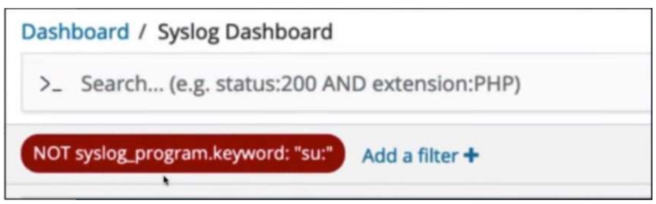


Figure 4-5: Kibana filter box (inverting)

One of the great things about Elastic is the way it creates the mapping in its storage engines, by segmenting up or tokenizing the strings. This means that any word can be searched easily, which can be a really powerful tool during forensic investigations. Instead of typing a filter, we can simply type in strings and take advantage of Kibana data discovery functions. Applying this free form search will eventually return all different type of log entries that contain that specific string. So not only we can narrow down our scope, without using a filter, since now we are using a search string, but Kibana can visualize and highlight where it was found. This is really convenient during forensic investigations because it allows to identify, based on broad searches, which records are in scope and then explore these records, see what they contain and see which fields contain the string that we are interested in. This is useful in cases where we are not sure what we are looking at or when we are starting with a weak lead. Simply typing in the weak lead and without having to know the structure of the data, we will be able to search for that string in any field possible. Here we can either use Kibana or Lucene syntax.

4.2.3.2 NetFlow Dashboard

The next figure presents the dashboard from our NetFlow data, which as already mentioned in D4.2, is a statistical summary (no content, since we are not loading pcap data). In this figure, we can see that we are loading a short period of time worth of data and specifically we are loading 5 days' worth of NetFlow that amounts to about 30Gbytes of traffic.

Figure 4-6: Log specific dashboard (NetFlow dashboard)

The summaries and the different colour in these graphs, represent the different protocols and in our case purple is ICMP, green is UDP and blue is TCP. Moreover, because the y-axis visualisation is in logarithmic scale, we can get a clear idea of where even the small data points are. So as we see, we can visualise the ICMP part that does not dwarf by the large data transactions of the TCP part.

Scrolling down, we see the source and destination categories as the following figures presents. On the left, we see source and destination IP addresses allowing to see who are the consumers and generators in terms of traffic. To the right is source and destination ports going from L3 addressing to L4. In the middle, we have got the maps that provides us with an idea of the geolocation of the source and destination of the traffic and is presented in a heat-map form. This is incredibly helpful when we want to get an idea, a broad understanding of traffic patterns. However, we should be aware that IP-based geolocation is not perfect, however, it is going to be enough for us to get some broad trends.

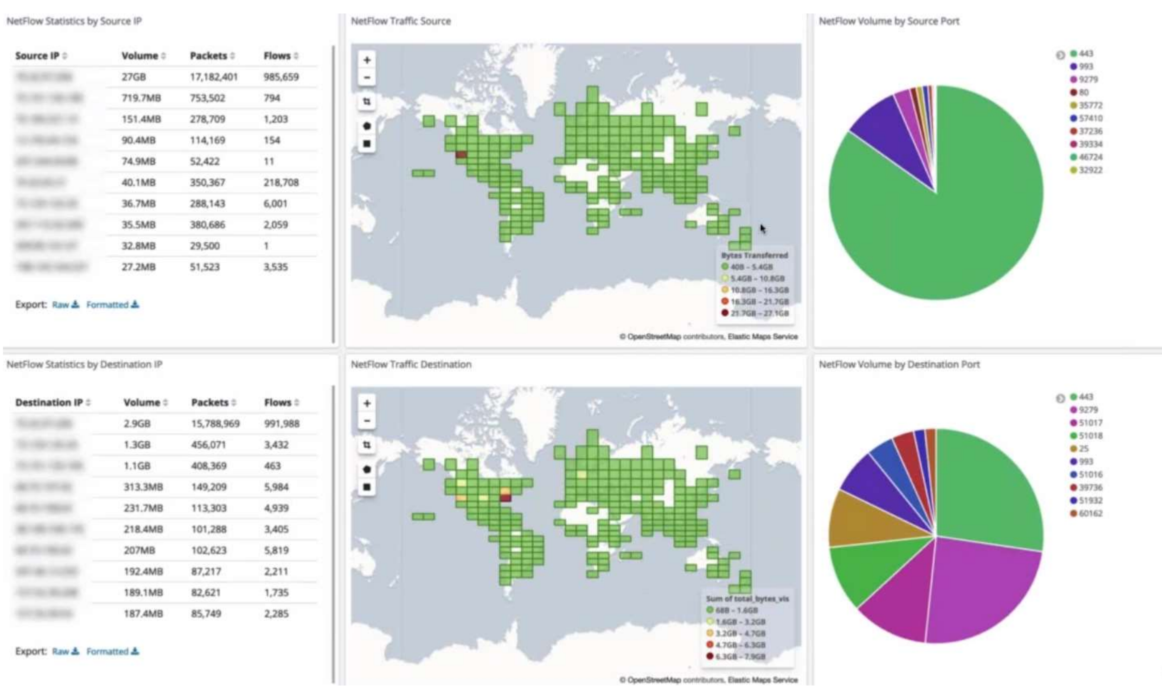


Figure 4-7: NetFlow dashboard (source and destination categorisation)

4.2.3.3 Webserver Logs (HTTPD) Dashboard

The next figure presents the dashboard from our webserver logs. On the top line, we see the request methods, while the bottom line shows the response codes, which like the previous dashboards are fully interactive and searchable. These are helpful during forensic investigations as it allows us to visualize patterns of traffic and anomalies. If we see something that does not match the expected patterns, then it becomes an investigative anomaly that we should try and figure out. The ASN numbers are also displayed, identifying where that traffic is coming from.

Figure 4-8: Log specific dashboard (HTTPD dashboard)

Scrolling down we visualise the same information based on source host (vhost) name, as well as source IP address geolocation, followed by user agents which can be invaluable for trying to characterize behaviour in our environment, if this is available. If we are looking at https logs that are being generated from our own server, it is going to be helpful. If we are looking at traffic coming from a proxy server, we are not going to get that information from an encrypted data flow. But if we are intercepting with a TLS proxy the ELK Stack of SPEAR-FR will be able to handle that.

At the very bottom like the previous dashboards we have got the discovery panel, which allows us a full exploration of all these fields that we might be interested in.

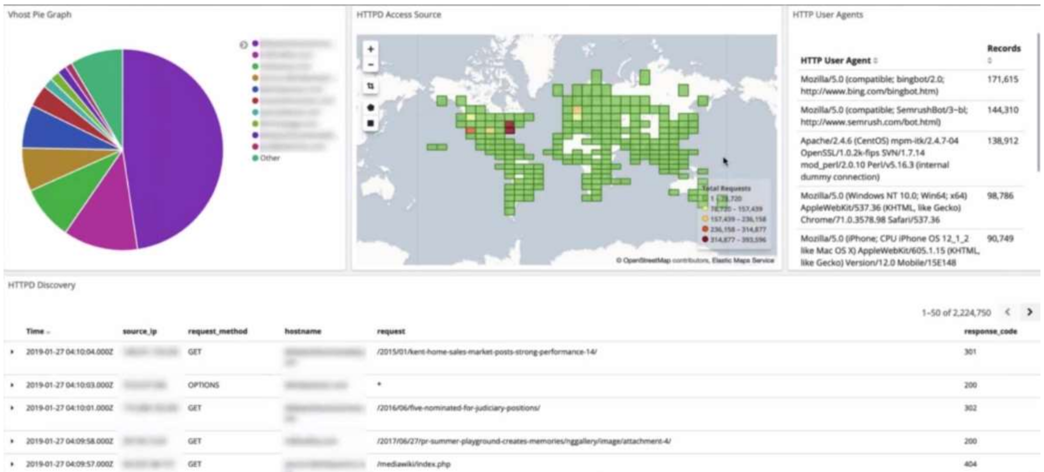


Figure 4-9: Webserver dashboard (source and destination categorisation)

4.2.3.4 Login Activity Dashboard

The next figure presents a dashboard that integrates NetFlow and various different types of logging data. This dashboard correlates NetFlow data along the top row (same graphs we saw before) based on the log events themselves. So the second entry reflects the fact that we have a timeline based on syslog data and this includes, in this specific example, just SSH login records, that as shown it includes almost 800 unsuccessful logins! This could indicate some kind of a brute force attack, which is not uncommon within internet connected machines, but at least it provides us with the ability to see any kind of ratios between what was or what was not present, in terms of the login results success or failure. The source map allows to visualize those events themselves.

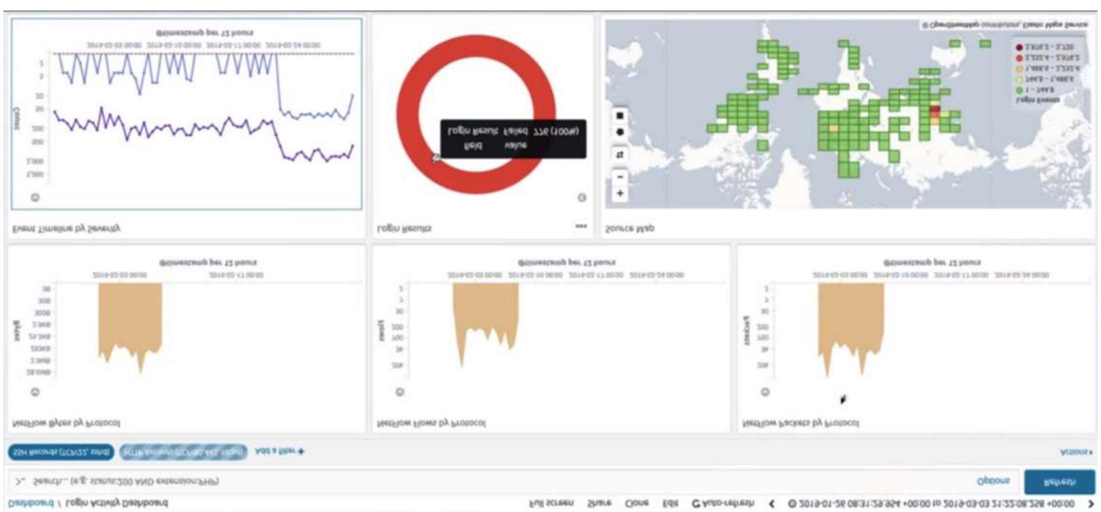


Figure 4-10: Log specific dashboard (NetFlow and various logging data dashboard)

5 Unit Testing

According to the assessment methodology defined in D2.3, section 4.1.1 states that: “Unit test plans will be developed during the implementation phase of the project. All individual units of the SPEAR solution will be tested to determine if they are operational and if they meet their specifications.”

In this section the unit test cases for the developed components, are defined, executed and their results presented. They include references to the system functional and non-functional requirements as defined in D2.2.

<table>
<tr><td>Test Case ID</td><td colspan="2">FR_01</td><td>Component</td><td>softflowd</td></tr>
<tr><td>Description</td><td colspan="4">Install and configure softflowd, on a pilot machine, to generate and export network flows</td></tr>
<tr><td>Req ID</td><td colspan="2">F39</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td colspan="2">ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="4">Centos 7 server on pilot premises</td></tr>
<tr><td colspan="5">Test steps</td></tr>
<tr><td>1</td><td colspan="4">Ensure that we have a few utilities installed on the server to satisfy the dependencies<pre>
yum install libtool automake autoconf python-devel libpcap-devel
</pre></td></tr>
<tr><td>2</td><td colspan="4">Copy the softflowd compressed source files:<pre>
cd /root/
wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/softflowd/softflowd-0.9.9.tar.gz
</pre>Decompress them<pre>
tar -zxvf softflowd-0.9.9.tar.gz
</pre></td></tr>
<tr><td>3</td><td colspan="4">Run the configuration script that checks whether we have the relevant programs dependencies in place and where those binaries are on our system.<pre>
cd softflowd-0.9.9
./configure
configure: creating ./config.status
config.status: creating Makefile
config.status: WARNING: 'Makefile.in' seems to ignore the --datarootdir setting
config.status: creating config.h
</pre></td></tr>
<tr><td>4</td><td colspan="4">Run the make utility to build a binary executable ready to install<pre>
make
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o softflowd.o softflowd.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o log.o log.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o netflow1.o netflow1.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o netflow5.o netflow5.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o netflow9.o netflow9.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o freelist.o freelist.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o convtime.o convtime.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o strlcpy.o strlcpy.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o strlcat.o strlcat.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o closefrom.o closefrom.c
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o daemon.o daemon.c
gcc -o softflowd softflowd.o log.o netflow1.o netflow5.o netflow9.o freelist.o convtime.o strlcpy.o strlcat.o closefrom.o daemon.o -lpcap
gcc -g -O2 -DFLOW_SPLAY   -DEXPIRY_RB   -I.   -c -o softflowctl.o softflowctl.c
gcc -o softflowctl softflowctl.o convtime.o strlcpy.o strlcat.o closefrom.o daemon.o -lpcap
</pre></td></tr>
<tr><td>5</td><td colspan="4">Install the application</td></tr>
</table>

```
# make install
[ -d /usr/local/sbin ] || \
    ./mkinstalldirs /usr/local/sbin
[ -d /usr/local/share/man/man8 ] || \
    ./mkinstalldirs /usr/local/share/man/man8
/usr/bin/install -c -m 0755 -s softflowd /usr/local/sbin/softflowd
/usr/bin/install -c -m 0755 -s softflowctl /usr/local/sbin/softflowctl
/usr/bin/install -c -m 0644 softflowd.8 /usr/local/share/man/man8/softflowd.8
/usr/bin/install -c -m 0644 softflowctl.8 /usr/local/share/man/man8/softflowctl.8
```

Input data

List Network Interfaces

ip link show

Test whether we can see relevant messages

```
# softflowd -D -v 9 -i eth0 -n 10.250.100.16:9995 -T full
```

In the above example, the switches used are explained:

-D	Debug mode, which bring this to the foreground
-v 9	Version 9 of NetFlow
-i eth0	The interface to listen on number
-n 10.250.100.16:9995	The target host IP address and port number of the collector/analyser
-T full	All protocols

However, we should run softflowd in the background by removing the –D switch

Result

```
[root@rnd-web ~]# softflowd -D -v 9 -i eth0 -n 10.250.100.16:9995 -T full
Using eth0 (idx: 0)
softflowd v0.9.9 starting data collection
Exporting flows to [10.250.100.16]:palace-4
ADD FLOW seq:1 [10.250.21.112]:19328 <> [10.250.73.28]:22 proto:6
ADD FLOW seq:2 [10.250.73.28]:80 <> [10.250.156.12]:56632 proto:6
ADD FLOW seq:3 [10.250.73.28]:80 <> [10.250.156.12]:56712 proto:6
ADD FLOW seq:4 [10.250.73.28]:80 <> [10.250.156.12]:56711 proto:6
ADD FLOW seq:5 [10.250.73.28]:80 <> [10.250.156.12]:56713 proto:6
ADD FLOW seq:6 [10.250.73.28]:80 <> [10.250.156.12]:56715 proto:6
ADD FLOW seq:7 [10.250.73.28]:80 <> [10.250.156.12]:44431 proto:6
ADD FLOW seq:8 [10.250.73.28]:80 <> [10.250.156.12]:16768 proto:6
ADD FLOW seq:9 [10.250.73.28]:80 <> [10.250.156.12]:56728 proto:6
ADD FLOW seq:10 [10.250.73.28]:80 <> [10.250.156.12]:56730 proto:6
ADD FLOW seq:11 [10.250.73.28]:80 <> [10.250.156.12]:56731 proto:6
ADD FLOW seq:12 [10.250.73.28]:80 <> [10.250.156.12]:10633 proto:6
```

Test Case Result

Achieved

<table>
<tr><td>Test Case ID</td><td>FR_02</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">Enable softflowd application so that we can stop/start and restart it like a service and have this enabled after the server has had a reboot</td></tr>
<tr><td>Req ID</td><td>F39</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">Installed and configured softflowd probe on server at pilot premises.</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Create file /etc/init.d/softflowd and add the following entries to it:
<pre>
#! /bin/bash
#
chkconfig: 2345 80 30
description: SoftFlow Deamon Service
BEGIN INIT INFO
Provides: SOFTFLOWD
Short-Description: Start/Stop/Restart SOFTFLOWD TCP Flow Probe
END INIT INFO
#
SOFTFLOWD This init.d script is used to start SOFTFLOWD.
#
SOFTFLOWD=/usr/local/sbin/softflowd
VERSION="9"
INTERFACE="eth0"
COLLECTOR="10.250.100.16"
CPORT="9995"
PID_FILE="/var/run/softflowd.pid"
OPTIONS="-v ${VERSION} -i ${INTERFACE} -n ${COLLECTOR}:${CPORT} -T full -p
${PID_FILE}"

start_SOFTFLOWD() {
${SOFTFLOWD} ${OPTIONS} > /dev/null &
return 1
}

stop_SOFTFLOWD() {
if [-f ${PID_FILE}]; then
kill `cat ${PID_FILE}` 2>1 /dev/null
\rm ${PID_FILE}
fi
}

########
case "$1" in
</pre></td></tr>
</table>

```
start)
echo -n "Starting SOFTFLOWD"
start_SOFTFLOWD;
echo " Done."
;;

stop)
echo -n "Stopping SOFTFLOWD"
stop_SOFTFLOWD;
echo " Done."
;;

restart)
echo -n "Restarting SOFTFLOWD"
stop_SOFTFLOWD;
sleep 1
start_SOFTFLOWD;
echo " Done."
;;

*)

echo "Usage: /etc/init.d/SOFTFLOWD {start|stop|restart}"
exit 1
esac
exit 0
```

2	Change the file permissions: `# chmod 755 /etc/init.d/softflowd`
3	Make the script a loadable initialisation script as part of the “service <application name> start” function by adding this with the chkconfig command: `# chkconfig --add softflowd`
Input data	Start the service: `# systemctl start softflowd.service`

Result	Check service status
	`# systemctl status softflowd.service`
	`● softflowd.service - LSB: Start/Stop/Restart SOFTFLOWD TCP Flow Probe`
	`Loaded: loaded (/etc/rc.d/init.d/softflowd; bad; vendor preset: disabled)`
	`Active: active (running) since Wed 2020-03-11 16:26:58 EET; 4s ago`
	`Docs: man:systemd-sysv-generator(8)`
	`Process: 30695 ExecStop=/etc/rc.d/init.d/softflowd stop (code=exited, status=0/SUCCESS)`
	`Process: 30703 ExecStart=/etc/rc.d/init.d/softflowd start (code=exited, status=0/SUCCESS)`
	`CGroup: /system.slice/softflowd.service`
	`└─30705 /usr/local/sbin/softflowd -v 9 -i eth0 -n 10.250.100.16:2055 -T full -p /var/run/softflowd.pid`

Test Case Result	Achieved

<table>
<tr><td>Test Case ID</td><td colspan="2">FR_03</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="4">SPEAR-FR can collect exported network flows</td></tr>
<tr><td>Req ID</td><td colspan="2">F39</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td colspan="2">ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="4">On pilot either router with NetFlow export capability of installed softflowd, exporting flows to SPEAR-FR VM (in the ED test environment IP is 10.250.100.16).</td></tr>
<tr><td colspan="5">Test steps</td></tr>
<tr><td>1</td><td colspan="4">Install rrdtool-devel
<code># yum install rrdtool-devel</code></td></tr>
<tr><td>2</td><td colspan="4">Download the latest code from github
<code># cd /opt</code>
<code># wget http://downloads.sourceforge.net/project/nfdump/stable/nfdump-1.6.13/nfdump-1.6.13.tar.gz</code></td></tr>
<tr><td>3</td><td colspan="4">Compile nfdump while in the "/opt/nfdump-1.6.13" directory:
<code># tar -zxvf nfdump-1.6.13.tar.gz</code>
<code># cd /opt/nfdump-1.6.13</code>
<code># ./configure --prefix=/opt/nfdump --enable-nfprofile --enable-nftrack --enable-sflow</code>
<pre>* Many thanks for using nfdump tools
* You may want to subscribe to the nfdump-discuss and/or
* nfsen-discuss mailing list:
* http://lists.sourceforge.net/lists/listinfo/nfdump-discuss
* http://lists.sourceforge.net/lists/listinfo/nfsen-discuss
* Please send bug reports back to me: phaag@users.sourceforge.net
* or to one of the lists.</pre><code># autoreconf</code>
<code># make</code>
<code># sudo make install</code></td></tr>
<tr><td>Input data</td><td colspan="4">1. <code># ps axo command | grep '[n]fcapd'</code>
2. <code># lsof -Pni | grep nfcapd</code>
3. <code># tcpdump -n -v dst port 9995</code></td></tr>
<tr><td>Result</td><td colspan="4">1. <code># /opt/nfdump/bin/nfcapd -w -D -p 9030 -u netflow -g apache -B 200000 -S 1 -P /data/nfsen/var/run/p9030.pid -z -I rnd-web-softflowd -l /data/nfsen/profiles-data/live/rnd-web-softflowd</code>
2. <code># nfcapd 14790 netflow 4u IPv4 1626345 0t0 UDP *:9030</code>
3. <code>listening on eth0, link-type EN10MB (Ethernet), capture size 65535</code>
<code>14:14:01.426775 IP 10.250.73.28.35829 > 10.250.100.16.iop: UDP, length 312</code>
<code>14:15:01.185508 IP 10.250.73.28.35829 > 10.250.100.16.iop: UDP, length 408</code>
<code>14:16:01.944233 IP 10.250.73.28.35829 > 10.250.100.16.iop: UDP, length 168</code></td></tr>
<tr><td>Test Case Result</td><td colspan="4">Achieved</td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>FR_04</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">Install and configure nfsen. This is an optional test visualising collected netflows</td></tr>
<tr><td>Req ID</td><td>F39</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">On pilot either router with NetFlow export capability of installed softflowd, exporting flows to SPEAR-FR VM (in the ED test environment IP is 10.250.100.16).
On SPEAR-FR server installed nfdump toolset</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Install the following packages for CentOS 7
<code># yum install -y httpd php wget gcc make rrdtool-devel rrdtool-perl perl-MailTools perl-Socket6 flex byacc perl-Sys-Syslog perl-Data-Dumper</code>
<code># yum install -y autoconf automake apache php perl-MailTools rrdtool-perl perl-Socket6 perl-Sys-Syslog.x86_64 policycoreutils-python tcpdump</code>
<code># echo "date.timezone = Europe/Belgrade " > /etc/php.d/timezone.ini yum update -y</code></td></tr>
<tr><td>2</td><td colspan="3">Set user for web interface and dump files
<code># useradd netflow</code>
<code># usermod -a -G apache netflow</code></td></tr>
<tr><td>3</td><td colspan="3">Create required folders
<code># mkdir -p /data/nfsen</code>
<code># mkdir -p /var/www/html/nfsen</code></td></tr>
<tr><td>4</td><td colspan="3">Download the latest code from github
<code># cd /opt</code>
<code># wget https://sourceforge.net/projects/nfsen/files/stable/nfsen-1.3.8/nfsen-1.3.8.tar.gz</code>
<code># tar -zxvf nfsen-1.3.8.tar.gz</code>
<code># cd nfsen-1.3.8</code></td></tr>
<tr><td>5</td><td colspan="3">Edit configuration file to make sure all variables are set correctly:
<code># cd etc</code>
<code># cp nfsen-dist.conf nfsen.conf</code>
<code># vi nfsen.conf</code>

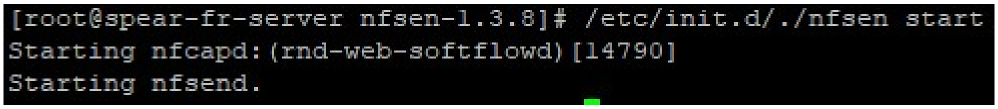
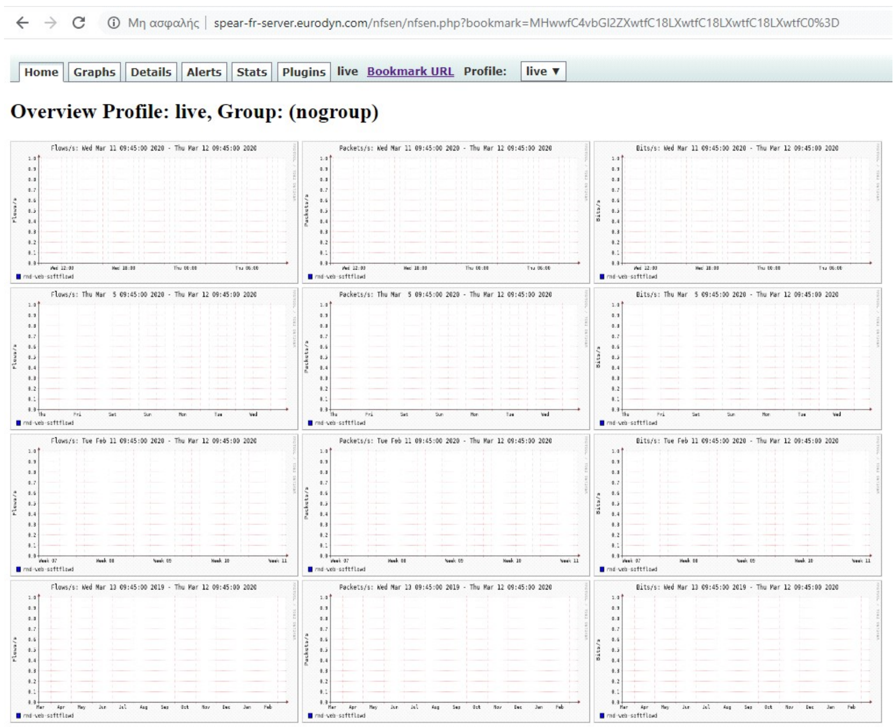
<code>$BASEDIR= "/data/nfsen";</code>
<code>$HTMLDIR = "/var/www/html/nfsen";</code>
<code>$PREFIX = '/opt/nfdump/bin';</code>

<code>$WWWUSER = "apache";</code>
<code>$WWWGROUP = "apache";</code></td></tr>
<tr><td>6</td><td colspan="3">Add our host to the file to allow for collection, my %sources looks like this:
In the example below we have two valid sources with different ports and different colors.
<code>%sources = (</code></td></tr>
</table>

	`'rnd-server-softflowd'    => { 'port' => '9030', 'col' => '#0000ff', 'type' => 'netflow' },` `);`

7	Run the perl installation script to install nfsen
	`# cd ..`
	`# ./install.pl etc/nfsen.conf`
	Press enter to accept the default path.
	`# Perl to use: [/usr/bin/perl]`
	
	Ignore any Errors since we did not configure any flows at this point.

8	Optionally we can make it start at boot:
	`# vi /etc/init.d/nfsen`
	
	And add this to the file:
	`#!/bin/bash`
	`#! #chkconfig: - 50 50`
	`#description: nfsen`
	
	`DAEMON=/data/nfsen/bin/nfsen`
	
	`case "$1" in`
	`start)`
	`$DAEMON start`
	`;;`
	`stop)`
	`$DAEMON stop`
	`;; status)`
	`$DAEMON status`
	`;;`
	`restart)`
	`$DAEMON stop`
	`sleep 1`
	`$DAEMON start`
	`;;`
	`*)`
	`echo "Usage: $0 {start|stop|status|restart}"`
	`exit 1`
	`;;`
	`esac`
	
	`exit 0`
	then make script executable
	`# chmod +x /etc/init.d/nfsen`

	Start the nfsen deamon: `# /etc/init.d/./nfsen start` 
Input data	Once we have configured our NetFlow source (see previous use cases), we should start seeing data in ~5-10 minutes. Navigate to http://spear-fr-server.eurodyn.com/nfsen/nfsen.php
Result	
Test Case Result	Achieved

Test Case ID	FR_05	Component	SPEAR-FR
Description	SPEAR-FR can collect log files from the identified smart grid assets (clients).		
Req ID	F39	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	SPEAR-FR and SPEAR-PPF (client) Centos 7 servers		
Test steps			

1

Enable and install the Extra Packages for Enterprise Linux (EPEL) repository, since it contains many useful packages, which are not included in RHEL. A few dependencies of syslog-ng are available in this repo.

```
# wget https://dl.fedoraproject.org/pub/epel/epel-release-latest-7.noarch.rpm
# rpm -Uvh epel-release-latest-7.noarch.rpm
```

2

Install syslog-ng both on SPEAR-FR and the identified smart grid assets/servers

```
# cd /etc/yum.repos.d/
# wget https://copr.fedorainfracloud.org/coprs/czanik/syslog-ng324/repo/epel-7/czanik-syslog-ng324-epel-7.repo
# yum install syslog-ng
# systemctl enable syslog-ng
```

3

Configure Syslog-ng on SPEAR-FR:

- SSH to SPEAR-FR server
- Make a copy of the syslog-ng.conf to /etc/syslog-ng/conf.d

```
# cd /etc/syslog-ng
# cp /etc/syslog-ng/syslog-ng.conf /etc/syslog-ng/conf.d/spear.conf
```

- Adapt the spear syslog-ng "spear.conf" configuration:

```
source spear_ppf_source {
    network(
        ip(0.0.0.0)
        port(6514)
    );
};

log {
       source(spear_ppf_source);
       destination(d_spear_ppf_camunda);
};

destination d_spear_ppf_camunda {
       file("/var/log/spear/spear_ppf");
};
```

- Update /etc/syslog-ng/syslog-ng.conf to include spear.conf

```
@include "/etc/syslog-ng/conf.d/*.conf"
```

<table>
<tr><td></td><td>Configure Syslog-ng on client (SPEAR-PPF):

SSH to client (SPEAR-PPF server)
Make a copy of the syslog-ng.conf to /etc/syslog-ng/conf.d
<code># cd /etc/syslog-ng</code>
<code># cp /etc/syslog-ng/syslog-ng.conf /etc/syslog-ng/conf.d/spear.conf</code>
Adapt the spear syslog-ng configuration:
<pre>
destination spear_fr_destination {
    network("spear-fr-server.eurodyn.com"
        port(6514)
    );
};

log {
    source(s_spear_ppf_access_logs);
    destination(spear_fr_destination);
};

source s_spear_ppf_access_logs {
    file("/var/log/httpd/spear-ppf-access.log" follow-freq(1));
};
</pre>
Update /etc/syslog-ng/syslog-ng.conf to include spear.conf
<code>@include "/etc/syslog-ng/conf.d/*.conf"</code>
</td></tr>
<tr><td>Input data</td><td>Start syslog-ng on both the SPEAR-FR server and the client (SPEAR-PPF).
<code># systemctl start syslog-ng</code>
Access either the PIA or the “Forensic Readiness Process” applications</td></tr>
<tr><td>Result</td><td>Navigate on the SPEAR_FR server to
<code># cd /var/log/spear</code>
We should see a new file, namely spear_ppf. If we open it we should see all log messages from the "/var/log/httpd/spear-ppf-access.log” location of the client</td></tr>
<tr><td>Test Case Result</td><td>Achieved</td></tr>
</table>

Test Case ID	FR_06	Component	SPEAR-FR
Description	SPEAR-FR can collect security events published on the message bus		
Req ID	F39	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	SPEAR-FR Centos 7 server		

Test steps

1

Import the required libraries:

```
from kafka import KafkaConsumer
import os, sys
import configparser
```

2

Create the Kafka Consumer: Kafka-python library (pip install Kafka-python) is used. For the creation of the Consumer, the following items are required:

- The CA certificate
- The consumer key
- The consumer certificate
- The password of the certificate

`configparser` library in Python can be used to set the mentioned items.

```
cafile = config['DEFAULT']['kafka_ca_file']
consumer_certfile = config['DEFAULT']['kafka_cert_file']
consumer_keyfile = config['DEFAULT']['kafka_key_file']
consumer_pass = config['DEFAULT']['kafka_pass']

producer = KafkaConsumer(group_id='spear_consumer',
      bootstrap_servers='[{0}:{1}]'.format('kafka', 9092),
      security_protocol='SSL',
      ssl_ciphers='ALL',
      ssl_check_hostname=False,
      ssl_cafile=cafile,
      ssl_certfile=consumer_certfile,
      ssl_keyfile=consumer_keyfile,
      ssl_password=consumer_pass)
```

In order to use `configparser` , a file called "config.ini" is needed:

```
[DEFAULT]
kafka_ca_file=/<path_of_certificates>/CARoot.pem
kafka_cert_file=/<path_of_certificates>/BDAC_consumer-certificate.pem
kafka_key_file= /<path_of_certificates>/BDAC_consumer-key.pem
kafka_pass= <password>
```

Note: Replace the marked parts with the path of your certificates

3

Subscribe the consumer to desired topics :

```
consumer.subscribe(['schneider_operational_topic'])
```

<table>
<tr><td></td><td colspan="2">

Note: The list of the available topics is

	Topic name	Description
Streaming Bus	spear_topic	Network packets captured
	flows_topic	Netflow data
	vets_operational_topic	Operational data from hydro power use case
	schneider_operational_topic	Operational data from substation use case
	spear_PPC_Operational_topic	Operational data from combined IAN and HAN use case
	certh_operational_battery_topic	Operational data about battery from smart house use case
	certh_operational_electricity1stfloor_topic	Operational data about 1st floor electricity from smart house use case
	certh_operational_electricityGRfloor_topic	Operational data about ground floor electricity from smart house use case
	certh_operational_electricityhome_topic	Operational data about electricity from smart house use case
	certh_operational_pv_topic	Operational data about PV from smart house use case

</td></tr>
<tr><td>4</td><td colspan="2">

Read the messages and do the necessary process. For instance, here there is an example of reading 50 messages and writing them to a text file.

```
i = 0
    for message in consumer:
        text_file = open('schneider_operational_kafka.txt' + str(message.partition), "a+")
        text_file.write(str(message) + '\n')
        i += 1
        if i == 50:
        # We stop when we read 50 messages
        Break
```

</td></tr>
<tr><td>5</td><td colspan="2">

Close the consumer

```
consumer.close()
```

</td></tr>
<tr><td colspan="2">Input data</td><td>Access the text file schneider_operational_kafka.txt</td></tr>
<tr><td colspan="2">Result</td><td>We should see inside all messages stored</td></tr>
<tr><td colspan="2">Test Case Result</td><td>Achieved</td></tr>
</table>

Test Case ID	FR_07	Component	SPEAR-FR
Description	For data in transit TLS should encrypt syslog messages, exchanged between the syslog server and the clients.		
Req ID	F40	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	SPEAR-FR and SPEAR-PPF (client) Centos 7 servers and TLS certificates (spear-fr-server_cacert.pem, spear-fr-server_serverkey.pem, spear-fr-server_cacert.pem, spear-ppf-server_clientcert.pem, spear-ppf-server_clientkey.pem)		
Test steps			

1	Configure Syslog-ng on SPEAR-FR:
	• SSH to spear-fr server
	• Create directories cert.d and ca.d under /etc/syslog-ng
	`# cd /etc/syslog-ng`
	`# mkdir cert.d ca.d`
	• Copy spear-fr-server_servercert.pem and spear-fr-server_serverkey.pem to cert.d.
	`# cp /root/SPEAR_FR_CA/spear-fr-server_servercert.pem cert.d/`
	`# cp /root/SPEAR_FR_CA/spear-fr-server_serverkey.pem cert.d/`
	• Copy spear-fr-server_cacert.pem to ca.d
	`# cp /root/SPEAR_FR_CA/spear-fr-server_cacert.pem ca.d/`
	• issue the following command on the certificate:
	`# cd ca.d/`
	`# openssl x509 -noout -hash -in spear-fr-server_cacert.pem`
	• The result is a hash (for example f3734642), a series of alphanumeric characters based on the Distinguished Name of the certificate.
	• Create a symbolic link to the certificate that uses the hash returned by the previous command and the .0 suffix.
	`# ln -s spear-fr-server_cacert.pem f3734642.0`
	• Make a copy of the syslog-ng.conf to /etc/syslog-ng/conf.d
	`# cp /etc/syslog-ng/syslog-ng.conf /etc/syslog-ng/conf.d/spear.conf`
	• Adapt the spear syslog-ng configuration:
	`source spear_ppf_tls_source {`
	`    network(`
	`        ip(0.0.0.0)`
	`        port(6514)`
	`        transport("tls")`
	`        tls(`
	`            key_file("/etc/syslog-ng/cert.d/spear-fr-server_serverkey.pem")`
	`            cert_file("/etc/syslog-ng/cert.d/spear-fr-server_servercert.pem")`
	`            ca_dir("/etc/syslog-ng/ca.d")`		

```
            )
        );
    };

    log {
            source(spear_ppf_tls_source);
            destination(d_spear_ppf_camunda);
    };

    destination d_spear_ppf_camunda {
            file("/var/log/spear/spear_ppf");
    };
```

- Update /etc/syslog-ng/syslog-ng.conf to include spear.conf

```
@include "/etc/syslog-ng/conf.d/*.conf"
```

Configure Syslog-ng on client (for example SPEAR-PPF):

- SSH to client (SPEAR-PPF server)
- Create directories cert.d and ca.d under /etc/syslog-ng

```
# cd /etc/syslog-ng
# mkdir cert.d ca.d
```

- Copy spear-ppf-server_clientcert.pem and spear-ppf-server_clientkey.pem from the SPEAR-FR server to cert.d on the client server.

```
# scp spear-ppf-server_clientkey.pem root@spear-ppf-server:/etc/syslog-ng/cert.d/
# scp spear-ppf-server_clientcert.pem root@spear-ppf-server:/etc/syslog-ng/cert.d/
```

- Copy spear-fr-server_cacert.pem from the SPEAR-FR server to ca.d on the client server

```
# scp spear-fr-server_cacert.pem root@spear-ppf-server:/etc/syslog-ng/ca.d/
```

- issue the following command on the certificate (on the client VM):

```
# cd ca.d/
# openssl x509 -noout -hash -in spear-fr-server_cacert.pem
```

- The result is a hash (for example f3734642), a series of alphanumeric characters based on the Distinguished Name of the certificate.
- Create a symbolic link to the certificate that uses the hash returned by the previous command and the .0 suffix.

```
# ln -s spear-fr-server_cacert.pem f3734642.0
```

- Make a copy of the syslog-ng.conf to /etc/syslog-ng/conf.d

```
# cp /etc/syslog-ng/syslog-ng.conf /etc/syslog-ng/conf.d/spear.conf
```

- Adapt the spear syslog-ng configuration:

```
destination spear_fr_tls_destination {
    network("spear-fr-server.eurodyn.com"
        port(6514)
        transport("tls")
        tls(
            ca_dir("etc/syslog-ng/ca.d")
```

<table>
<tr><td></td><td colspan="2">

```
            key_file("/etc/syslog-ng/cert.d/spear-ppf-
server_clientkey.pem")
            cert_file("/etc/syslog-ng/cert.d/spear-ppf-
server_clientcert.pem")        )
    );
};

log {
    source(s_spear_ppf_access_logs);
    destination(spear_fr_tls_destination);
};

source s_spear_ppf_access_logs {
    file("/var/log/httpd/spear-ppf-access.log" follow-freq(1));
};
```

- Update /etc/syslog-ng/syslog-ng.conf to include spear.conf

```
@include "/etc/syslog-ng/conf.d/*.conf"
```

</td></tr>
<tr><td>Input data</td><td colspan="2">Re/start syslog-ng on both the SPEAR-FR server and the client (SPEAR-PPF).
<code># systemctl restart syslog-ng</code>
Access either the PIA or the “Forensic Readiness Process” applications</td></tr>
<tr><td>Result</td><td colspan="2">Navigate on the SPEAR_FR server to
<code># cd /var/log/spear</code>
On the server side, tail the file, where logs are arriving ("/var/log/spear/spear_ppf"). We should see logs from the client (spear-ppf)</td></tr>
<tr><td colspan="2">Test Case Result</td><td>Achieved</td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>FR_08</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">Provide support for reliability and ability to deal with network outages.</td></tr>
<tr><td>Req ID</td><td>F40</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">SPEAR-FR and SPEAR-PPF (client) Centos 7 servers and TLS certificates (spear-fr-server_cacert.pem, spear-fr-server_serverkey.pem, spear-fr-server_cacert.pem, spear-ppf-server_clientcert.pem, spear-ppf-server_clientkey.pem)</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Update Syslog-ng on client (for example SPEAR-PPF):
SSH to spear-fr server
Adapt the spear syslog-ng configuration:
<pre>
destination spear_fr_tls_destination {
    network("spear-fr-server.eurodyn.com"
        port(6514)
        disk-buffer(
                    mem-buf-size(10000)
                    disk-buf-size(2000000)
                    reliable(yes)
                    dir("/tmp/disk-buffer")
)
        transport("tls")
        tls(
            ca_dir("etc/syslog-ng/ca.d")
            key_file("/etc/syslog-ng/cert.d/spear-ppf-
server_clientkey.pem")
            cert_file("/etc/syslog-ng/cert.d/spear-ppf-
server_clientcert.pem")   )
    );
};
</pre></td></tr>
<tr><td>Input data</td><td colspan="3">Re/start syslog-ng on both the SPEAR-FR server and the client (SPEAR-PPF).
<code># systemctl restart syslog-ng</code>
Disable network interface of SPEAR-FR to emulate a network outage. In the meantime access the PIA application
After 2-3 minutes enable network interface of SPEAR-FR</td></tr>
<tr><td>Result</td><td colspan="3">Navigate on the SPEAR_FR server to
<code># cd /var/log/spear</code>
On the server side, tail the file, where logs are arriving ("/var/log/spear/spear_ppf"). We should see logs from the client (spear-ppf)</td></tr>
<tr><td>Test Case Result</td><td colspan="3">Achieved</td></tr>
</table>

<table>
<tr><td colspan="2">Test Case ID</td><td>FR_09</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td colspan="2">Description</td><td colspan="3">SPEAR-FR must securely store collected forensic data in an encrypted disk.</td></tr>
<tr><td colspan="2">Req ID</td><td>F41</td><td>Priority</td><td>S</td></tr>
<tr><td colspan="2">Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td colspan="2">Pre-condition(s)</td><td colspan="3">SPEAR-FR Centos 7 server</td></tr>
<tr><td colspan="5">Test steps</td></tr>
<tr><td>1</td><td colspan="4">Create a file which will act as our storage device, using the ubiquitous dd command and the /dev/random pseudo-device. This way we will write random data, which should mimic the encrypted data that will actually be written to it:

dd if=/dev/urandom of=/root/spear-fr bs=1M count=512
chown -R root:root /root/spear-fr
chmod -R 700 /root/spear-fr</td></tr>
<tr><td>2</td><td colspan="4">Create a LUKS partition within the file (LUKS Container).
NOTE: We need to provide a password that will be needed to decrypt the data.

cryptsetup -y luksFormat /root/spear-fr

If we check out the file now, we can see that it is now known as a LUKS encrypted file:
file /root/spear-fr
/root/spear-fr: LUKS encrypted file, ver 1 [aes, xts-plain64, sha256] UUID: 9f2ba4e7-5875-470f-84ac-4c82078799f0
[root@spear-fr-server spear-fr-files]# file /root/spear-fr
/root/spear-fr: LUKS encrypted file, ver 1 [aes, xts-plain64, sha256] UUID: 9f2ba4e7-5875-470f-84ac-4c82078799f0</td></tr>
<tr><td>3</td><td colspan="4">Open the container.
NOTE: We must provide the password we set before for the file, which is needed to decrypt it.

cryptsetup luksOpen /root/spear-fr spear-fr-volume

The above command will open the LUKS device, and maps it to “spear-fr-volume“, by creating a file at /dev/mapper/spear-fr-volume. This basically opens the file as a local loopback device so that the rest of the system can now handle the file as if it were a real device.</td></tr>
<tr><td>4</td><td colspan="4">Create and mount the file system:

mkfs.ext4 -j /dev/mapper/spear-fr-volume

The above command will create a filesystem written on top of our LUKS container that is contained in our file.</td></tr>
<tr><td>5</td><td colspan="4">Logically to mount the device:
• Create mount location:
mkdir /mnt/spear-fr-files
• Mount filesystem
mount /dev/mapper/spear-fr-volume /mnt/spear-fr-files/</td></tr>
</table>

	• Give read, write permissions only to root:
	`# chown -R root:root /mnt/spear-fr-files/`
	`# chmod -R 700 /mnt/spear-fr-files/`
	
	Test whether can see the mounted filesystem as part of our available filesystems:
	`# df -h`
	should return:
	`Filesystem                        Size  Used Avail Use% Mounted on`
	`/dev/mapper/spear-fr-volume       486M  2.3M  459M   1% /mnt/spear-fr-files`

6	Write data to this location, and it will be placed, encrypted, in the file.
	Create a file inside and add some content:
	
	`# cd /mnt/spear-fr-files`
	`# echo 'SPEAR' >> spear-ppf-messages`

7	When finished collecting evidences unmount the device and close the LUKS file again to encrypt the data at rest, store the checksum of the drive and cryptographically verify it.
	
	`# umount /mnt/spear-fr-files`
	`# cryptsetup luksClose spear-fr-volume`
	`# find /root/spear-fr -type f -print0 | xargs -0 md5sum >> /root/spear-fr.md5`
	`# md5sum -c /root/spear-fr.md5`

Input data	Login as a root user in the spear-fr server and try to access/view/edit the directory/file Login as another user in the spear-fr server and try to access/view/edit the directory/file

Result	We should be able to view the directory/file contents
	We should get "Permission denied" message:
	
	`[elk_user@spear-fr-server ~]$ cd /mnt/spear-fr-files/`
	`-bash: cd: /mnt/spear-fr-files/: Permission denied`

Test Case Result	Achieved

Test Case ID	FR_10	Component	SPEAR-FR
Description	For data at rest SPEAR-FR must ensure data integrity via strict file and folder permissions.		
Req ID	F41	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	Encrypted and mounted drive/file where forensic data is stored (see FR_01)		
Test steps			
1	Connect (SSH) to the SPEAR-FR server as a non-root user		
Input data	Try to access/view/edit the directory/file where forensic data are located		

Result	"Permission denied" message:
	`[elk_user@spear-fr-server ~]$ cd /mnt/spear-fr-files/`
	`-bash: cd: /mnt/spear-fr-files/: Permission denied`		

Test Case Result	Achieved		

Test Case ID	FR_11	Component	SPEAR-FR
Description	SPEAR-FR should ensure that access to the forensic data is controlled.		
Req ID	F42	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	Encrypted and unmounted drive/file where forensic data is stored		
Test steps			
1	Connect (SSH) to the SPEAR-FR server		
Input data	Try to open the LUKS device (container): `cryptsetup luksOpen /root/spear-fr spear-fr-volume`		
Result	User is prompted for password. Hence only authorized users can access data		
Test Case Result	Achieved		

<table>
<tr><td>Test Case ID</td><td>FR_12</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">Ensure availability of forensic data.</td></tr>
<tr><td>Req ID</td><td>F43</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">LUKS device (container) where forensic data is stored, is open and mounted</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Setup daily backups of SPEAR-FR data locations, using a cron job
<code># cp -rp /mnt/spear-fr/ /mnt/spear-fr-files-backup/</code></td></tr>
<tr><td>Input data</td><td colspan="3">Store the checksum of all the files inside the backup directory into <file>.md5
<code># find /mnt/spear-fr-files-backup/ -type f -print0 | xargs -0 md5sum >> /mnt/spear-fr-files-backup.md5</code></td></tr>
<tr><td>Result</td><td colspan="3">Cryptographically verify the copy
<code># md5sum -c /mnt/spear-fr-files-backup.md5</code>
Should return
<code>/mnt/spear-fr-files-backup/spear_ppf_access: OK</code>
If data were modified this should return:
<code>/mnt/spear-fr-files-backup/spear_ppf_access: FAILED</code>
<code>md5sum: WARNING: 1 computed checksum did NOT match</code></td></tr>
<tr><td>Test Case Result</td><td colspan="3">Achieved</td></tr>
</table>

<table>
<tr><td>Test Case ID</td><td>FR_13</td><td>Component</td><td>SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">Ensure availability of forensic data (alternative approach)</td></tr>
<tr><td>Req ID</td><td>F43</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">Unmounted and encrypted drive/file where forensic data is stored</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Setup daily backups of SPEAR-FR data locations, using a cron job
<code># cp -rp /root/spear-fr /root/spear-fr-backup</code></td></tr>
<tr><td>Input data</td><td colspan="3">Store the checksum of the drive
<code># find /root/spear-fr-backup -type f -print0 | xargs -0 md5sum >> /root/spear-fr-backup.md5</code></td></tr>
<tr><td>Result</td><td colspan="3">Cryptographically verify the copy
<code># md5sum -c /root/spear-fr-backup.md5</code>
Should return
<code>/root/20200214_spear-fr-backup: OK</code></td></tr>
<tr><td>Test Case Result</td><td colspan="3">Achieved</td></tr>
</table>

Test Case ID	FR_14	Component	SPEAR-FR
Description	Ensure that the restoration of the cryptographically verifiable copies of the forensic data work as expected.		
Req ID	F43	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	Backup of the open, mounted drive where forensic data is stored		
Test steps			

1	Cryptographically verify the copy
	`# md5sum -c /mnt/spear-fr-files-backup.md5`
	Should return
	`/mnt/spear-fr-files-backup/spear_ppf_access: OK`		

Input data	Access data location `# cd /mnt/spear-fr-files-backup`		
Result	Verify that all files are there `# ls -la`		
Test Case Result	Achieved		

Test Case ID	FR_15	Component	SPEAR-FR
Description	Ensure that the restoration of the cryptographically verifiable copies of the forensic data work as expected.		
Req ID	F43	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	Backup of the closed (thus encrypted) and unmounted drive/file where forensic data is stored		
Test steps			

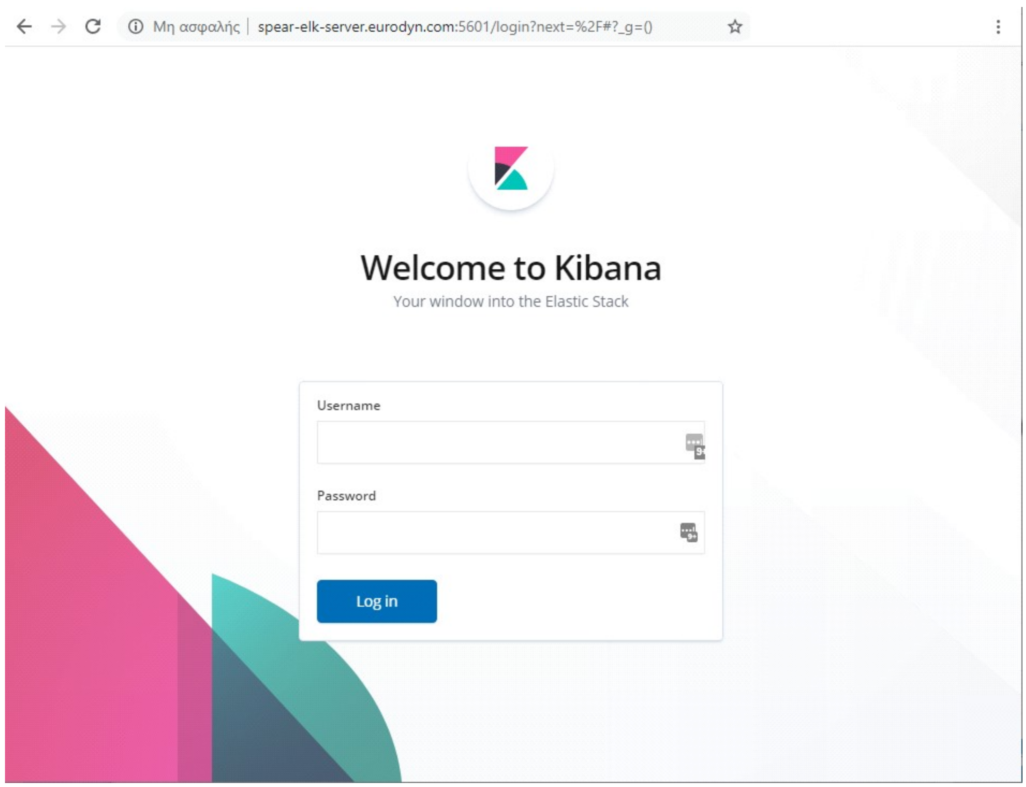
1	Cryptographically verify the copy
	`# md5sum -c /root/20200214_spear-fr-backup.md5`
	Should return
	`/root/20200214_spear-fr-backup: OK`		

2	Open the container. NOTE: We must provide the encryption password. `# cryptsetup luksOpen /root/20200214-spear-fr-backup 20200214-spear-fr-backup-volume` The above command will open the LUKS device, and maps it to “20200214_spear-fr-backup-volume “, by creating a file at /dev/mapper/20200214_spear-fr-backup-volume.		

3	Logically to mount the device:
	• Create mount location:
	`# mkdir /mnt/20200214-spear-fr-files`
	• Mount filesystem
	`# mount /dev/mapper/20200214-spear-fr-volume /mnt/20200214-spear-fr-files/`		

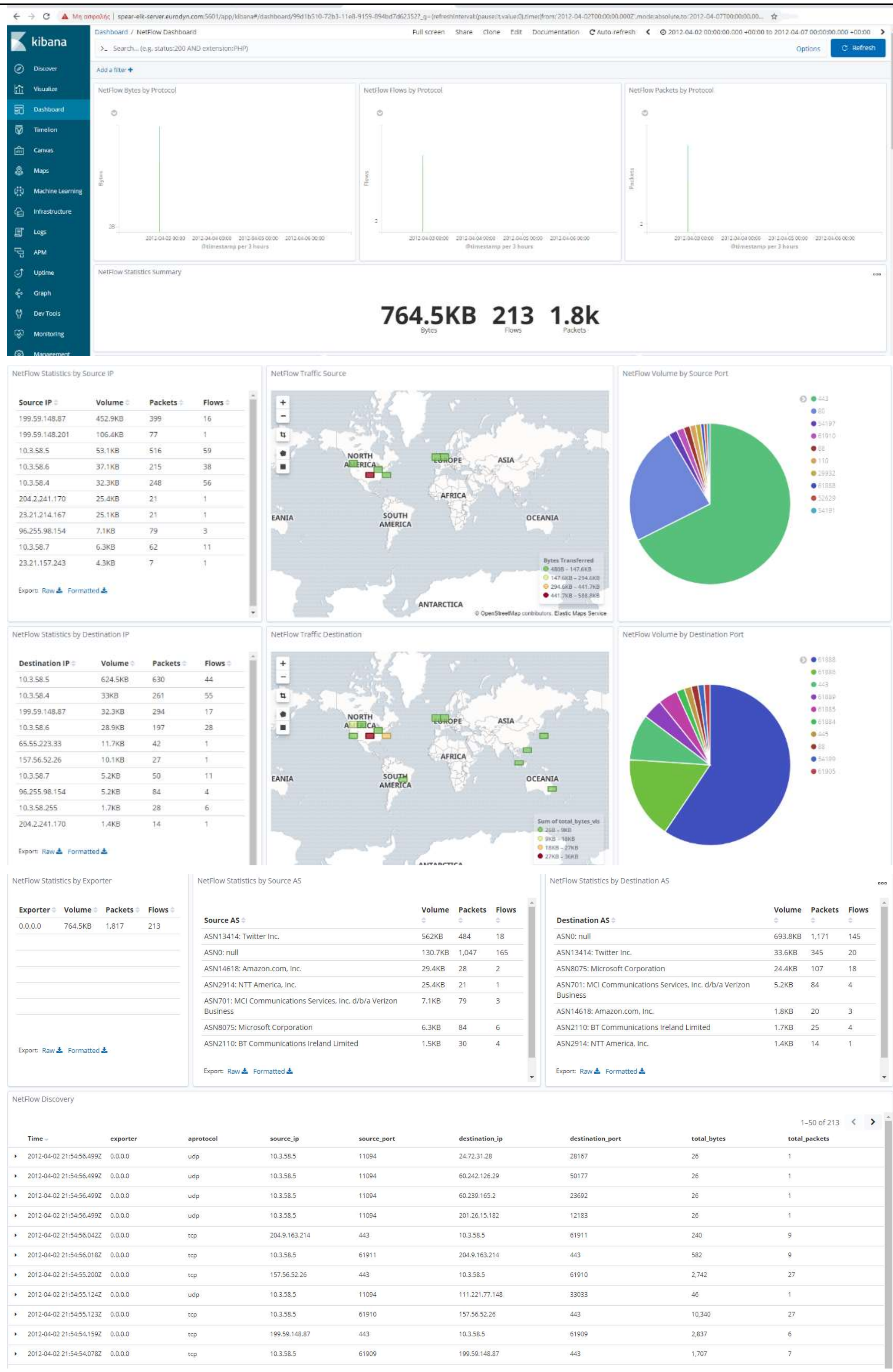
Input data	Access data location `# cd /mnt/20200214-spear-fr-files`		
Result	Verify that all files are there `# ls -la`		
Test Case Result	Achieved		

Test Case ID	FR_16	Component	SPEAR-PPF
Description	SPEAR-FR must assess: a) the purpose of the processing, b) the types of personal data stored and d) the retention period.		
Req ID	F45	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)			
Test steps			
1	Access the DPIA tool, either though the integrated platform or directly at https://spear-ppf.eurodyn.com/pia/#/home		
Input data	Identify the data processing activities, including the purpose of the processing, types of personal data stored, data retention periods and security measures implemented		
Result	Review, validate and sign		
Test Case Result	Achieved		

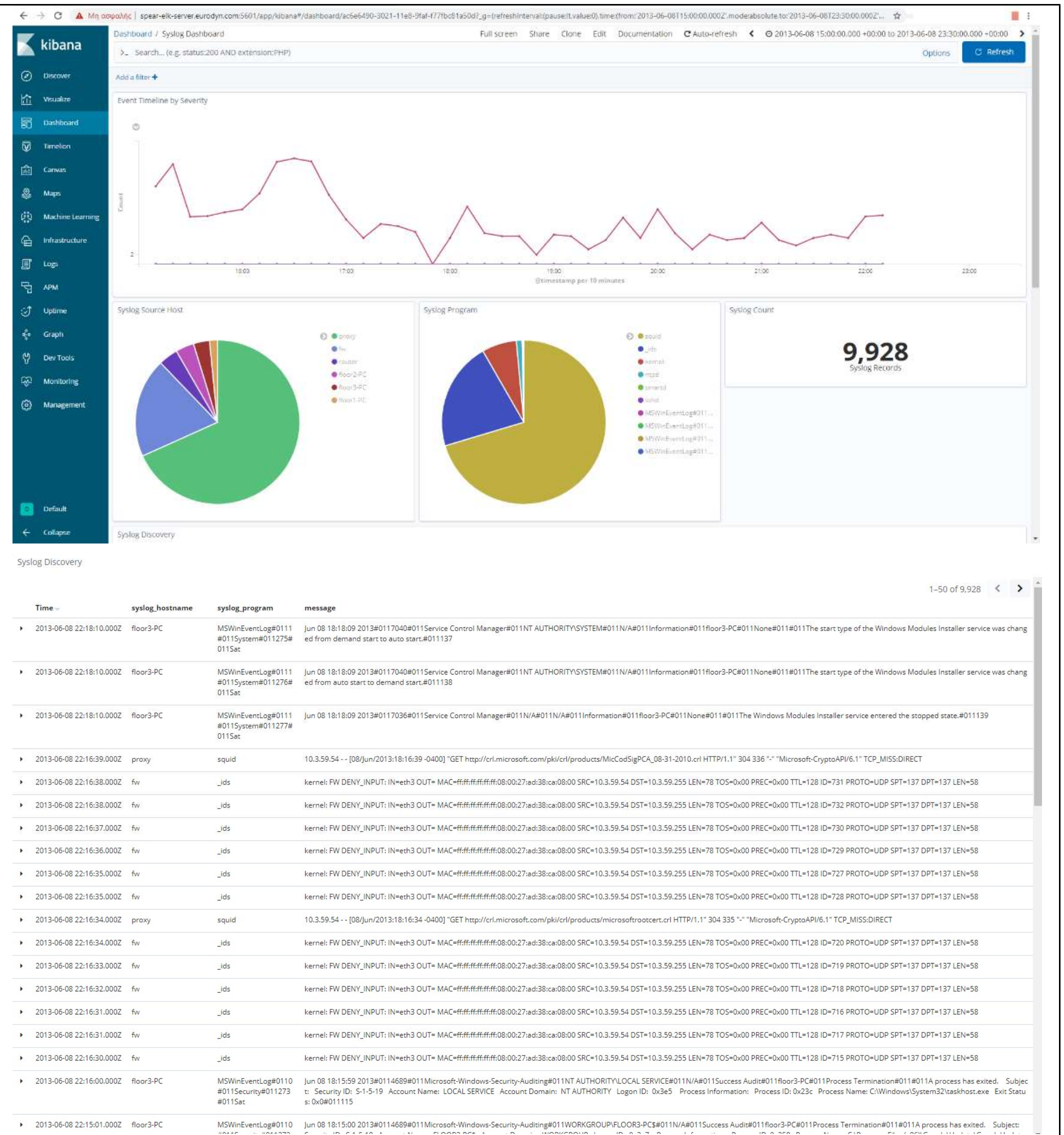
Test Case ID	FR_17	Component	ELK Stack of SPEAR-FR
Description	ELK Stack of SPEAR-FR should authenticate users.		
Req ID	F45	Priority	S
Prepared by	ED, REAL	Tested by	ED, REAL
Pre-condition(s)	ELK Stack VM of SPEAR-FR is up and running on forensic investigator machine		
Test steps			
1	Access the ELK Stack of SPEAR-FR (from within the ED test environment this is available at http://spear-elk-server.eurodyn.com:5601/)		
Input data	Navigate to http://spear-elk-server.eurodyn.com:5601/		
Result	Forensic investigator is presented with login screen 		
Test Case Result	Achieved		

<table>
<tr><td>Test Case ID</td><td>FR_18</td><td>Component</td><td>ELK Stack of SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">ELK Stack of SPEAR-FR should ingest archived NetFlow data.</td></tr>
<tr><td>Req ID</td><td>F45</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">ELK Stack VM of SPEAR-FR is up and running on forensic investigator machine. SPEAR-FR encrypted disk/partition copy is received (spear-fr and spear-fr.md5).</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Place copy of the encrypted partition under root and cryptographically verifies it
<code># md5sum -c /root/spear-fr.md5</code>
Should return
<code>/root/spear-fr: OK</code></td></tr>
<tr><td>2</td><td colspan="3">Open the LUKS device (container):
<code>cryptsetup luksOpen /root/spear-fr spear-fr-volume</code></td></tr>
<tr><td>3</td><td colspan="3">Create and mount the file system:
<code># mkfs.ext4 -j /dev/mapper/spear-fr-volume</code></td></tr>
<tr><td>4</td><td colspan="3">Logically to mount the device:
Create mount location:
<code># mkdir /mnt/spear-fr-files</code>
Mount filesystem
<code># mount /dev/mapper/spear-fr-volume /mnt/spear-fr-files/</code>
Give read, write permissions only to root:
<code># chown -R root:root /mnt/spear-fr-files/</code>
<code># chmod -R 700 /mnt/spear-fr-files/</code></td></tr>
<tr><td>5</td><td colspan="3">To ingest existing NetFlow evidence, parse them with the nfdump2spear-elk.sh script

<code># nfdump2spear-elk.sh -r /mnt/spear-fr-files/path/to/netflow/nfcapd.202002190000 -w /logstash/nfarch/inputfile_1.txt</code></td></tr>
<tr><td>6</td><td colspan="3">Access the ELK Stack of SPEAR-FR (from within the ED test environment this is available at http://spear-elk-server.eurodyn.com:5601/)</td></tr>
<tr><td>7</td><td colspan="3">Access the “NetFlow Dashboard”</td></tr>
<tr><td>Input data</td><td colspan="3">Select the time period of interest</td></tr>
<tr><td>Result</td><td colspan="3">User is presented with the search results</td></tr>
</table>

	
Test Case Result	Achieved

<table>
<tr><td>Test Case ID</td><td>FR_19</td><td>Component</td><td>ELK Stack of SPEAR-FR</td></tr>
<tr><td>Description</td><td colspan="3">ELK Stack of SPEAR-FR should ingest archived syslog data.</td></tr>
<tr><td>Req ID</td><td>F45</td><td>Priority</td><td>S</td></tr>
<tr><td>Prepared by</td><td>ED, REAL</td><td>Tested by</td><td>ED, REAL</td></tr>
<tr><td>Pre-condition(s)</td><td colspan="3">ELK Stack VM of SPEAR-FR is up and running on forensic investigator machine. SPEAR-FR encrypted disk/partition copy is received (spear-fr and spear-fr.md5).</td></tr>
<tr><td colspan="4">Test steps</td></tr>
<tr><td>1</td><td colspan="3">Place copy of the encrypted partition under root and cryptographically verifies it
<code># md5sum -c /root/spear-fr.md5</code>
Should return
<code>/root/spear-fr: OK</code></td></tr>
<tr><td>2</td><td colspan="3">Open the LUKS device (container):
<code>cryptsetup luksOpen /root/spear-fr spear-fr-volume</code></td></tr>
<tr><td>3</td><td colspan="3">Create and mount the file system:
<code># mkfs.ext4 -j /dev/mapper/spear-fr-volume</code></td></tr>
<tr><td>4</td><td colspan="3">Logically to mount the device:
• Create mount location:
<code># mkdir /mnt/spear-fr-files</code>
• Mount filesystem
<code># mount /dev/mapper/spear-fr-volume /mnt/spear-fr-files/</code>
• Give read, write permissions only to root:
<code># chown -R root:root /mnt/spear-fr-files/</code>
<code># chmod -R 700 /mnt/spear-fr-files/</code></td></tr>
<tr><td>5</td><td colspan="3">Copy archived syslog data into the /logstash/syslog/ directory
<code># cp /mnt/spear-fr-files/path/to/syslog/ /logstash/syslog/</code></td></tr>
<tr><td>6</td><td colspan="3">Access the ELK Stack of SPEAR-FR (from within the ED test environment this is available at http://spear-elk-server.eurodyn.com:5601/)</td></tr>
<tr><td>7</td><td colspan="3">Access the “NetFlow Dashboard”</td></tr>
<tr><td>Input data</td><td colspan="3">Select the time period of interest</td></tr>
<tr><td>Result</td><td colspan="3">User is presented with the search results</td></tr>
</table>

	
Test Case Result	Achieved

6 Conclusions

This deliverable summarized the logging architecture, presented in D4.2, which was able to support the smart grid network forensics process, by a) implementing an effective and secure storage for smart grid network forensic data and b) ingesting, processing, and analysing stored smart grid network forensic data.

The development of the SPEAR Forensics Repository was based on the result of the analysis of the SPEAR Forensics Repository requirements provided in this deliverable (Section 2).

Collection and long-term storage of forensic data (log files, flow-data, full content data and statistical data) is handled by a dedicated and secure forensic evidence server, namely the SPEAR Forensics Repository (SPEAR-FR). SPEAR-FR was built on top of open-source components such as cryptsetup, syslog-ng, softflowd, nfdump and nfsen.

Regarding querying and analytics, the deliverable described how the pre-configured ELK Stack appliance of SPEAR-FR, was incorporated into SPEAR, allowing us to ingest smart grid network forensic data stored in the SPEAR-FR, isolate and finally analyse them, thus addressing the computational resources needed during investigations and the time delays in processing log files. This pre-configured ELK Stack appliance allowed forensic investigators to focus on the important aspects of any forensic task, which is to apply own intelligence and awareness when analyzing collected security event data.

Furthermore, the deliverable explained the details of the software prototypes developed for each of the components, i.e. the SPEAR-FR and the ELK Stack of SPEAR-FR. The prototype description included installation and configuration details, configuration and the references to the artefact repositories.

Finally the deliverable also presented the results of the SPEAR assessment/unit tests performed over the logging architecture components namely the SPEAR-FR and the ELK Stack of SPEAR-FR, thus showing compliance to the requirements defined in WP2.

References

[1] ARTICLE 29 DATA PROTECTION WORKING PARTY, WP 248, Guidelines on Data Protection Impact Assessment (DPIA) and determining whether processing is “likely to result in a high risk” for the purposes of Regulation 2016/679, 2017, https://ec.europa.eu/newsroom/document.cfm?doc_id=44137

[2] ARTICLE 29 DATA PROTECTION WORKING PARTY, WP250rev.01, Guidelines on Personal data breach notification under Regulation 2016/679, 2018, https://ec.europa.eu/newsroom/article29/document.cfm?doc_id=49827

[3] Network Forensics, Tracking Hackers through Cyberspace, by Sherri Davidoff and Jonathan Ham, 2012, ISBN-13: 978-0-13-256471-7

[4] Introduction to logging for security purposes, Laying the groundwork for incident readiness, UK National Cyber Security Center, July 2018

[5] Stav, E., S. Walderhaug, and U. Johansen, ARCADE - An Open Architectural Description Framework. December 2013, SINTEF ICT. Available at: http://www.arcade-framework.org/wp-content/uploads/2013/12/ARCADE-Handbook.pdf