# AN ANALISYS OF APPLICATION LOGS WITH SPLUNK : DEVELOPING AN APP FOR THE SYNTHETIC ANALYSIS OF DATA AND SECURITY INCIDENTS

Roberto Bruzzese

Freelancer, Rome, Lazio, Italy

***ABSTRACT***

*The present work aims to enhance the application logs of an hypothetical infrastructure platform, and to build an App that displays the synthetic data about performance, anomalies and security incidents synthesized in the form of a Dashboard. The reference architecture, with multiple applications and multiple HW distribution, implementing a Service Oriented Architecture, is a real case of which the details have been abstracted because we want to extend the concept to all architectures with similar characteristics.*

***KEYWORDS***

*Log Analysis, Data Analysis, Splunk, Security Incidents*

## 1. INTRODUCTION

In this chapter it is described the work done in the context of the experimentation on Splunk tool in analyzing Logs. The work starts from the initial point, the search for logs, and continues with the problems to face for manipulation of raw data, ending with the control rules for detecting web attacks.

### 1.1. Description of the Starting Point

The starting point of my trip is a collection of files, the logs, within a medium-high complexity application infrastructure. My aim is to make these objects becoming a resource, more properly an asset of the organization.

The logs inside the Application Infrastructure (for the description of the architecture see the Figure 27) are many and often scattered in the meanders of the machines and their directories, and are often not known at company level except by individual users.

They are often used by individual developers who use them for troubleshooting purposes, or by system administrators who use them for management purposes, or by security officers for control purposes.

However, there is a lack of a unified vision of the resource, a lack of cross analysis between the various logs and of an in-depth look at the information they contain.

It often happens that these files, sometimes very voluminous, are not even read, before being forgotten or completely deleted. For this reason, it has origin the present work with the need to try to enhance, through an approach of “exploration and discovery”, of what is hidden in those logs. Then my activity consisted in what can be called “log mining ”.

As being perfectly aware that logs, intended as a support to contain a record of the activities carried out in an information system, can come from very different sources (firewalls, servers, databases, networks, applications, etc..) and that my main purpose is to explore only the contents of logs of application origin.

## 1.2. Fragmentary Knowledge of Available Logs

Often every single developer knows the location of their own logs. You have to look for a little bit before you find what you are looking for, if, as in my case, you play a cross-cutting role in applications. In an Infrastructure like the one under consideration (which is based on the J2EE platform and which is built with a SOA approach , using IBM Websphere Application Server) , it is well known in advance the position and the format of the Application Server logs, since they are standardized.

As far as the logs of the single applications of which the infrastructure is composed are concerned, it could be searched in the directories of the respective applications, where the logs are produced.

These logs are the ones on which the attention is focused since they are of direct competence of a software development unit in which this work was carried out.

The evaluation of the format of the individual logs made in the first analysis let me believe my work could pursue several objectives:

(1) to proceed to an automatic extraction of targeted information
(2) measure quantities, evaluate trends
(3) detecting errors, reporting
(4) exploring relations between information

## 1.3. It Should be Decided to Adopt a Specific or General Solution ? Make or Buy ? Open Source or Commercial ?

A possible choice was to build a log analysis [1] solution starting from open source resources, freely available, and of limited cost.

In particular, the “grep ”commands could be used. This command is a Linux or Unix utility that searches file input patterns or regular expressions.

This type of command could allow us to focus on a single problem at a time and examine the results in detail.

However it seemed to be more targeted for small cases or for analysis that are carried out with a high degree of manual skills and also have a low “friendly” type degree of usability. It also seemed that the integration into an infrastructure was restrictive and cumbersome, and complex, above all in the case of multiple types of different logs.

In addition, there are the following factors that are detrimental to this “grep utility” tool:

- It is suitable when you know perfectly what you want to achieve, and you focus on very specific text, on the whole file.
- You should not overlook the time it takes to search for large files.
- It seems often endless as you search all over the file (which can be terabytes!).
- It is not easy for correlating searches on multiple log files at the same time, or even on different devices (it may be useful to extend the search on the Web server of the Web Farm, or on the HTTP server, or on a Firewall).

The latter is an important aspect in log management because, in a multi-application and multi-distributed infrastructure , the application log is produced on the machine on which the application is hosted and executed, and in the case of cross analysis of logs from different sources, it is necessary to centralize it as the next Figure 1:

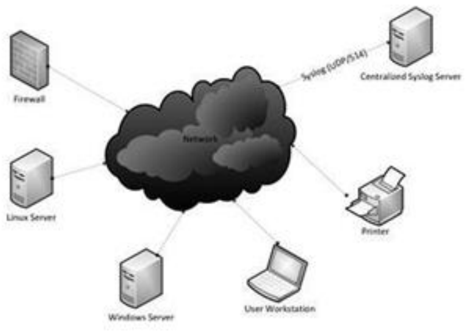


Figure 1: Centralization of Logs.

Splunk [2] commercial tool immediately seemed to me the most viable solution, providing this company the Enterprise distributions, freely downloadable, with licenses valid for 60 days, or, alternatively, Developer licenses, valid for 6 months.

This tool offers a series of features that reduce the time needed to analyze the logs on a periodic basis, as there are activatable files/directories monitors, which update the analysis. It also makes available the generation of alerts at the occurrence of predefined events, the generation of reports that allow corporate compliance, the centralization of the logs to be analyzed on which to make further actions or investigations. It should not be underestimated that several add-ons do exist, and also apps and APIs that allow Splunk to be integrated with the surrounding infrastructure.

Its use allows the easy centralization of log management, through different instances of log collector (Forwarders) , or log analyzer (Head Searches). It is possible to analyze multiple sources simultaneously, without affecting performance, as Splunk is based on the concept of data indexing (Indexer) that optimizes search times.

It is also easy to report and display in a Dashboard the summary data of the search results, thanks to a rich availability of graphs, which can be used from a web interface or even via a command line or web services.

## 2. SPLUNK FUNCTIONAL ARCHITECTURE

### 2.1. Splunk Functional Architecture

It is possible to install the version of Splunk Enterprise on a single instance of a machine, in such a way to have on a single site all the peculiarities of Splunk, namely “data input ”, “indexing ”, “search ”. In the case of more complex organizations, the typical architecture of Splunk is the type shown in the following Figure 2. The architecture is functional to the organization in which you want to install Splunk.

### 2.1.1. The Forwarders

Forwarders are 20 mb software distributions which can be installed directly on the site where the data need to be collected. They only monitor files or directories and send the collected data to Splunk instances (Enterprise type) that provide indexing of data.

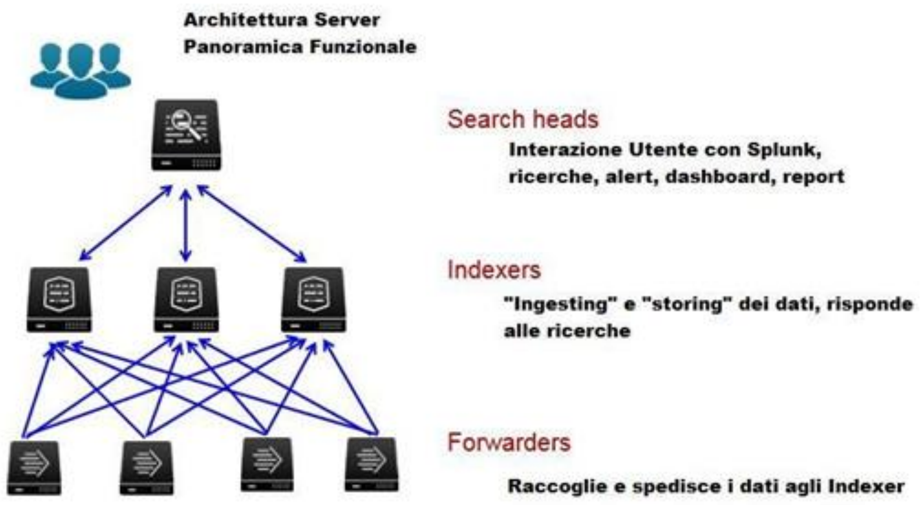

Architettura Server
Panoramica Funzionale
Search heads
Interazione Utente con Splunk, ricerche, alert, dashboard, report
Indexers
"Ingesting" e "storing" dei dati, risponde alle ricerche
Forwarders
Raccoglie e spedisce i dati agli Indexer

Figure 2: Splunk Functional Architecture Overview.

### 2.1.2. The Indexers

While the number of Forwarders is equivalent to the number of machines on which there are logs that need to be analyzed, the number of Indexers is calculated on the basis of a capacity plan depending on the size of the logs that are estimated to be handled. It is necessary to have approximately 100GB for each Indexer, in a configuration that meets the hardware requirements provided by Splunk. On the Indexers takes place data parsing and data indexing.

### 2.1.3. The Search Heads

As far as concerns the Search Heads, which are the input end points for user interactions, their number depends on the number of Users expected for that configuration. It should be noted that a Search Head User means both an interactive user and an REST application or also a web service user.

## 2.2. Initial Architecture

The architecture implemented in this project, is described in the following Figure 3.

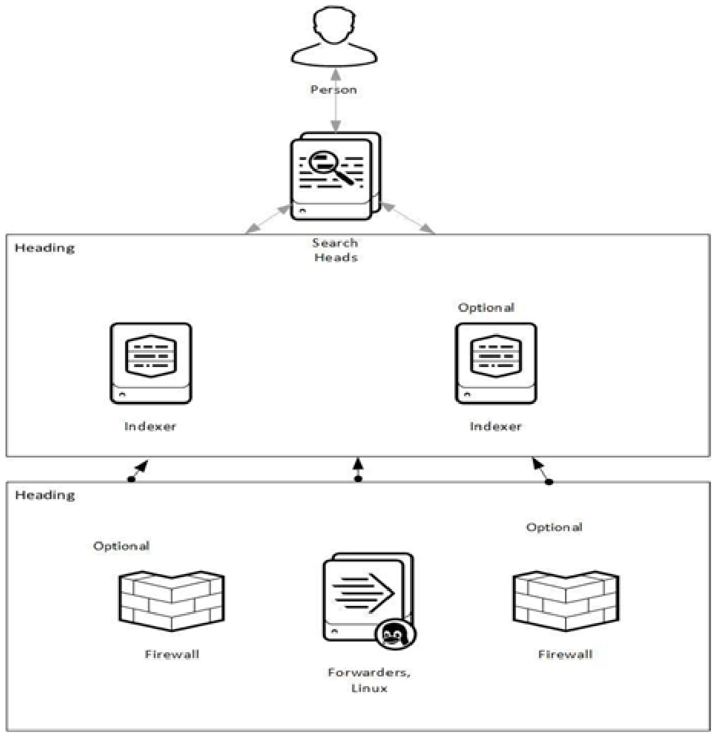


Figure 3: Initial Architecture.

This architecture has been realized using Virtual Machines created on the Aruba Cloud VPS. The instances were created by installing Splunk and transferring the logs which were withdrawed from the production environment of the Infrastructure. The instances of Splunk which were created are reported below :

**Universal Forwarder :** 1 of GB RAM, 1 single core CPU, 20 GB SDD, Linux system

**Indexer :** 2 GB RAM, 1 CPU single core, 40 GB SDD, Linux system

**Search Head :** 2GB RAM, 1 single core CPU, 40GB SDD, Linux system

Forwarder machines communicate with Indexer machines via a TCP connection on port 9997, through which they send data to the Indexer.

This minimum configuration is adequate to guarantee acceptable performance with a maximum of 5-10 logs, of about 300 mb size each and 1-2 million events.

## 2.3. HW System Requirements

The hardware requirements reported in Splunk Documentation necessary to be an adequate configuration to ensure efficient log management and searching are listed here below :

**Basic Level - Single Instance Host**

Intel x86 64-bit chip architecture 12 CPU cores a 2Ghz (at least 12GB RAM Standard 1Gb Ethernet NIC 800 IOPS (on average) Performance of Indexing Max : 20MB/sec Performance of Search Max : 50.000 events/sec

**Medium Level - Distributed Multi Hosts**

Instance Intel x86 64-bit chip architecture 12-16 CPU cores 2Ghz (at least) 12GB RAM Standard 1Gb Ethernet NIC 800 IOPS (on average)

**High Level - Distributed Multi Hosts**

Instance Intel 64-bit chip architecture 48 CPU cores at 2GHz (at least) 128GB RAM SSD 1Gb Ethernet NIC plus NIC optional SAN

## 2.4. Splunk processing: data pipeline

The initial architecture distinguishes the functional activities of Splunk, which take place as a data processing chain. Regardless of whether a single-instance or multi-instance architecture is adopted, the "data pipeline" follows three distinct stages:

1. Input: where the data is preloaded, divided into 64K blocks and annotated with the metadata: host, source, sourcetype. At this stage, the concept of event does not yet exist.
2. Parsing: In this phase Splunk segments the data into events, sets the timestamps, notes the metadata it obtains from the previous input phase, possibly transforms the event data, according to the transformation rules (regex).
3. Indexing: the events obtained by parsing are written to the disk in compressed and indexed form.
4. Search: this phase concerns the user’s access to the indexed data (in the form of search, reports, alerts, dashboards).

In the following Figure 4 is showed the data pipeline process :

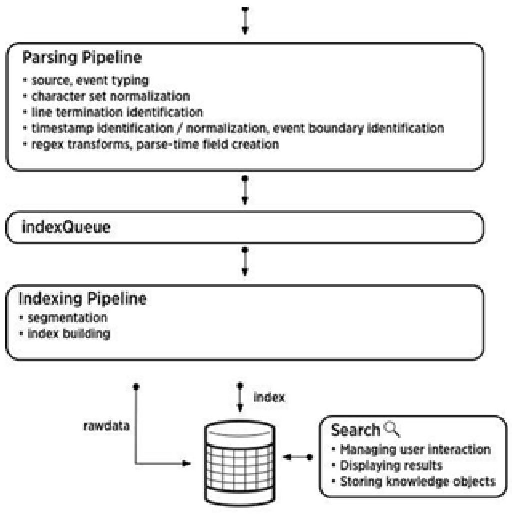

Parsing Pipeline
• source, event typing
• character set normalization
• line termination identification
• timestamp identification / normalization, event boundary identification
• regex transforms, parse-time field creation
indexQueue
Indexing Pipeline
• segmentation
• index building
rawdata
index
Search
• Managing user interaction
• Displaying results
• Storing knowledge objects

Figure 4: Data Pipeline.

Although in the present project there is a single source of data, it can be the most varied as Splunk is predisposed to all sources and formats (Figure 5).

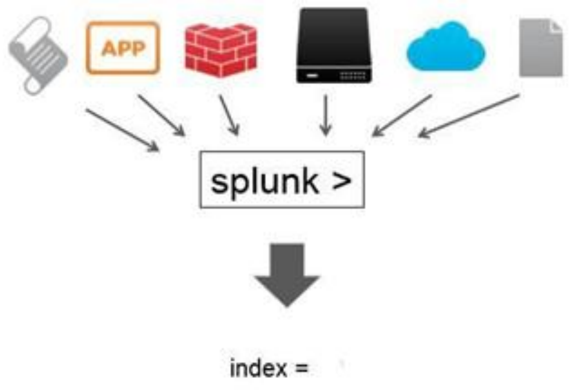


Figure 5: Heterogeneity of Data Sources.

## 3. INGESTION, EXTRACTION, ANALYSIS

### 3.1. Analysis of the Infrastructure Logs

In this project were analyzed sample portions of the Infrastructure Application Logs. In particular those related to a couple of applications.

Three main Indexes (for the sake of simplicity we will denominate them indexA, indexB and IndexC) have been created, on which all the searches have been carried out.

The set of search queries submitted to the system are still general and can be repeated in multiple contexts. In order to increase the generalization, the ease of use and the reusability of the search queries, the corresponding macros have been stored (although not reported in this paper).

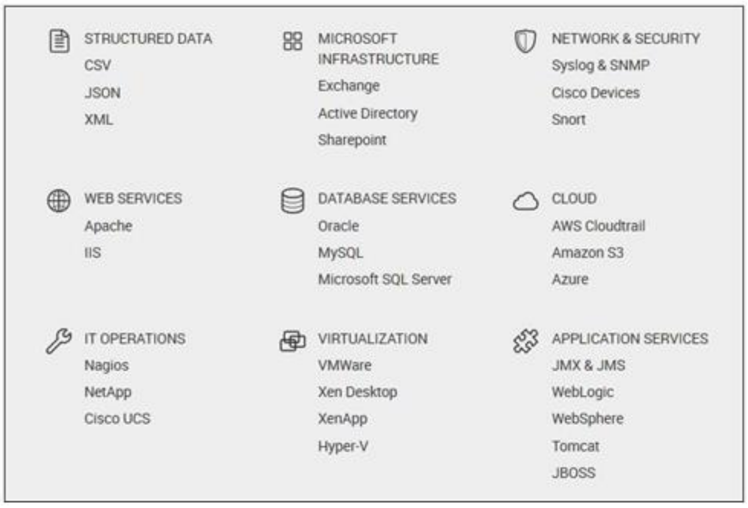
After loading, parsing, and indexing the data, it was necessary to pre-treat the data, in order to extract and better define the fields that are used in the subsequent stages of search.

### 3.2. Data Loading and Extraction

Data Loading into Splunk is an important step for the subsequent analysis and visualization phases. If the data are not correctly analyzed, time-stamped, and divided into events, it will be difficult in the next stages to carry out a ’proper analysis of the data and get insights into them’.

Since the data may come from many sources as web servers, application servers, operating systems, network security, services or applications (although in our case they will be exclusively of application origin) it is still important that the format makes all the subsequent analytical phases easily searchable.

In general, Splunk is equipped with internal facilities that recognize the most common formats from various sources (Figure 6) and then apply the preconfigured settings already stored. If Splunk recognizes the data source, it will apply the settings and definitions for that data source. The data formats can be divided into three types: **structured, semi-structured, and unstructured ones**.



STRUCTURED DATA
CSV
JSON
XML

MICROSOFT INFRASTRUCTURE
Exchange
Active Directory
Sharepoint

NETWORK & SECURITY
Syslog & SNMP
Cisco Devices
Snort

WEB SERVICES
Apache
IIS

DATABASE SERVICES
Oracle
MySQL
Microsoft SQL Server

CLOUD
AWS Cloudtrail
Amazon S3
Azure

IT OPERATIONS
Nagios
NetApp
Cisco UCS

VIRTUALIZATION
VMWare
Xen Desktop
XenApp
Hyper-V

APPLICATION SERVICES
JMX & JMS
WebLogic
WebSphere
Tomcat
JBOSS

Figure 6: Preconfigured Splunk Data Formats.

If structured formats occurs maybe additional settings need to be specified for the custom format in question. This can be done in the field extraction phase.

### 3.2.1. Application Logging

In the case of the application infrastructure under consideration, the logs considered are basically related to applications developed in-house. In this case, logging is delegated to developers for troubleshooting reasons, or for determining a certain level of quality service or determine system availability (through measurements of systems interruptions).

Most of the time they are just the developers who understand the meaning of the application logs as they are ultimately the only users to determine software bugs or make a fixing in case of malfunctions.

In the kind of infrastructure under consideration, which is a medium-high complexity system, with about 20 machines cooperating in the architecture, the resolution of application problems is a complex and time- consuming activity.

The adoption of a tool like Splunk, thanks to the use of queries and reports, alerts and messaging, can be of fundamental help to maintenance and operational activities.

This requires an activity of extraction of the fields, in order to fully analyze data, and thus making data to become an asset for the entire organization.

In fact, because it makes sense to believe that ’garbage in ... garbage out’, it cannot be expected to extract value if into Splunk is introduced only information of little value. In this sense we should follow as much as possible an approach in which there is a minimum gap between the development and operation of the system. Basically you have to try to get a log that is ’machine friendly’ and ’human readable’.

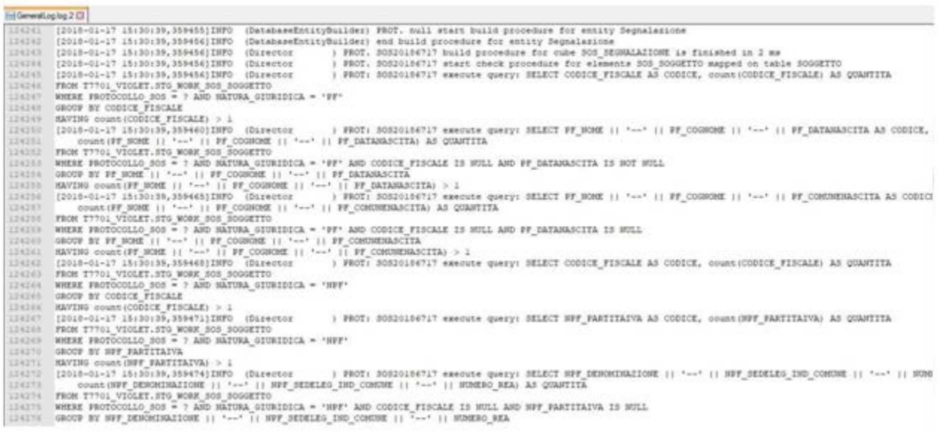
### 3.2.2. The Loggers

At the application level, the developer uses loggers (i.e. java log4j) that allow to write messages in multithreaded mode and in a typical standard format. In this way, if there is an exception, some peculiar aspects can be checked, while others are established and delegated to the Java framework.

In other words, the logger keeps track of what happened in the program, so if a crash or bug occurs, you can trace the cause.

We can say that the events produced by the application loggers are in the form of **’structured data’** while those that come from the proprietary applications developed ad hoc generally have a form of **’semi- structured’** data and in other cases of **’unstructured data’**. The latter may vary, and may also extend over several lines, while the event they report may be in a format without precise separators, and without the obligation of a start or an end clearly identified or reported explicitly.

The logs here reported and examined, relevant to the Infrastructure, are mostly structured data, although in some sections of the logs there may appear unstructured ones, as shown in Figure 7 :



```
[2018-01-17 15:30:39,359455]INFO  (DatabaseEntityBuilder) PROT. null start build procedure for entity Segnalazione
[2018-01-17 15:30:39,359456]INFO  (DatabaseEntityBuilder) end build procedure for entity Segnalazione
[2018-01-17 15:30:39,359456]INFO  (Director        ) PROT. SOS20186717 build procedure for cube SOS_SEGNALAZIONE is finished in 2 ms
[2018-01-17 15:30:39,359456]INFO  (Director        ) PROT. SOS20186717 start check procedure for elements SOS_SOGGETTO mapped on table SOGGETTO
[2018-01-17 15:30:39,359456]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT CODICE_FISCALE AS CODICE, count(CODICE_FISCALE) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'PF'
GROUP BY CODICE_FISCALE
HAVING count(CODICE_FISCALE) > 1
[2018-01-17 15:30:39,359460]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT PF_NOME || '--' || PF_COGNOME || '--' || PF_DATANASCITA AS CODICE,
    count(PF_NOME || '--' || PF_COGNOME || '--' || PF_DATANASCITA) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'PF' AND CODICE_FISCALE IS NULL AND PF_DATANASCITA IS NOT NULL
GROUP BY PF_NOME || '--' || PF_COGNOME || '--' || PF_DATANASCITA
HAVING count(PF_NOME || '--' || PF_COGNOME || '--' || PF_DATANASCITA) > 1
[2018-01-17 15:30:39,359465]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT PF_NOME || '--' || PF_COGNOME || '--' || PF_COMUNENASCITA AS CODICI
    count(PF_NOME || '--' || PF_COGNOME || '--' || PF_COMUNENASCITA) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'PF' AND CODICE_FISCALE IS NULL AND PF_DATANASCITA IS NULL
GROUP BY PF_NOME || '--' || PF_COGNOME || '--' || PF_COMUNENASCITA
HAVING count(PF_NOME || '--' || PF_COGNOME || '--' || PF_COMUNENASCITA) > 1
[2018-01-17 15:30:39,359468]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT CODICE_FISCALE AS CODICE, count(CODICE_FISCALE) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'NPF'
GROUP BY CODICE_FISCALE
HAVING count(CODICE_FISCALE) > 1
[2018-01-17 15:30:39,359471]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT NPF_PARTITAIVA AS CODICE, count(NPF_PARTITAIVA) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'NPF'
GROUP BY NPF_PARTITAIVA
HAVING count(NPF_PARTITAIVA) > 1
[2018-01-17 15:30:39,359474]INFO  (Director        ) PROT: SOS20186717 execute query: SELECT NPF_DENOMINAZIONE || '--' || NPF_SEDELEG_IND_COMUNE || '--' || NUM
    count(NPF_DENOMINAZIONE || '--' || NPF_SEDELEG_IND_COMUNE || '--' || NUMERO_REA) AS QUANTITA
FROM T7701_VIOLET.STG_WORK_SOS_SOGGETTO
WHERE PROTOCOLLO_SOS = ? AND NATURA_GIURIDICA = 'NPF' AND CODICE_FISCALE IS NULL AND NPF_PARTITAIVA IS NULL
GROUP BY NPF_DENOMINAZIONE || '--' || NPF_SEDELEG_IND_COMUNE || '--' || NUMERO_REA
```

Figure 7: Format of the Log indexA.

This latter type of log is much more complicated because all knowledge must be extracted manually and it requires collaboration and information exchange between the Splunk engineer and the programmer or vendor. In such cases, if Splunk is allowed to operate automatically, it will try to extract stuff that it wrongly believes to be data, and the whole process will not bring added value in the following phases of analysis and reporting.

In such type of logs, in order to be able to carry out a correct extraction of significant fields, it must taken into account the **’event breaking’**.

To make sure that we are loading or extracting correctly, we must put order in the initial chaos and this usually begins by separating an event from the next one, by means of a line or a character. There are many ways in which you can separate events from each other (see Splunk /Admin/Propsconf Documentation).

### 3.2.3. Data Extraction

The step mentioned above is called **’event extraction’**. It consists of identifying the line or group of lines that constitutes the information relating to an action. The **’extraction of a field’** consists in identifying precisely what can be considered a field, for its sequential position in a list of values, assuming the field is not already in the form of a key=value. This is a process for the that Splunk administrator, which creates value in the data sets.

### 3.2.4. Metadata fields

It should be noted that there are three most important default fields: host, source, and sourcetype which are also called metadata. They describe where the event originated.

**'host'** – it indicates the name of the host, the device, the IP address, or the domain name of the network from which the event originates.

**'source'** – it indicates the name of the File,¸ stream, or any other input from which the event originates. In the case that the data comes from directories and files specified by the configuration files, the source value is the path, e.g. /archive/server1/var/log/messages.0 or /var/log/. If the data source was based on data from a network, such as a protocol or port, it would be UDP:514.

**'sourcetype'** – this field stands for the format of the data of the source from which it originates. This value determines how the data will be formatted. Other default fields include date and time fields, which add more search 'granularity' to event timestamps. Finally Splunk Enterprise adds other default fields classified as 'internal' fields.

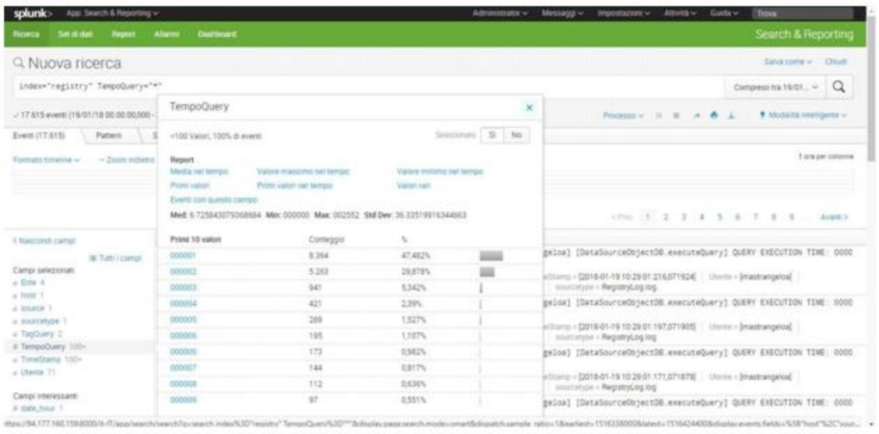
## 4. SEARCH ON APPLICATION LOGS

### 4.1. Examples of Searches on Indexes

Once the meaningful fields have been extracted, using the interactive extractor and the regular expressions, the search query can be run. This is possible using SPL (Search Processing Language) and after the extraction of of the relevant fields. Here below will be reported the main searches compiled for the present work.

### 4.1.1. Query 1 : SQL Query Times

This search query asks the system, to list the first 10 time intervals spent in running the sql queries of the application, sorted by percentage distribution, from the first, most frequent, to the last. As it can be seen from the result of the query in Figure 8 , 47% of SQL queries last 1 second; about 29% last 2 seconds.



```
index="registry" TempoQuery="*"
```

TempoQuery

>100 Valori, 100% di eventi

Med: 6.725843079368684 Min: 000000 Max: 002552 Std Dev: 36.33519916344663

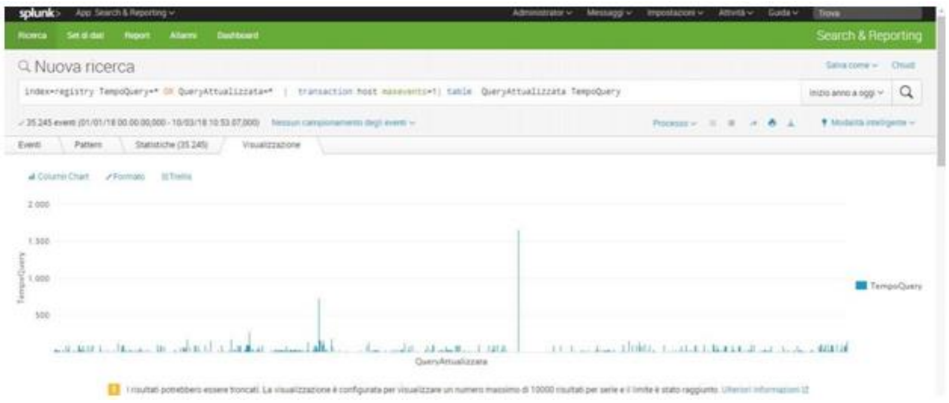
| Primi 10 valori | Conteggio | % |
|---|---|---|
| 000001 | 8.364 | 47,482% |
| 000002 | 5.263 | 29,878% |
| 000003 | 941 | 5,342% |
| 000004 | 421 | 2,39% |
| 000005 | 269 | 1,527% |
| 000006 | 195 | 1,107% |
| 000000 | 173 | 0,982% |
| 000007 | 144 | 0,817% |
| 000008 | 112 | 0,636% |
| 000009 | 97 | 0,551% |

```
geloa] [DataSourceObjectDB.executeQuery] QUERY EXECUTION TIME: 0000
```

Figure 8: results of Query 1

### 4.1.2. Query 2 : SQL Query Times (actualized)

In this query, carried out on the log, we asked the system to make a 'transaction', that is to group events having in common certain characteristics. In particular, it was asked to display all the queries on the axis of the abscissae, and the relative times taken to run that query, on the axis of the ordinates. By passing the mouse on the single line of the histogram, the query in question can be displayed. You can see, in the Figure 9, for the purpose of determining possible anomalies, a query whose time is more than 1500 ms.



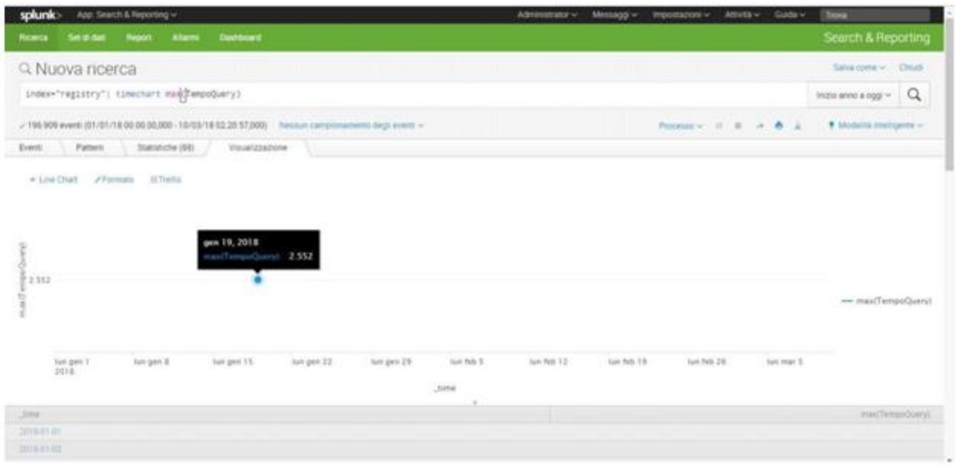
```
index=registry TempoQuery=* OR QueryAttualizzata=* | transaction host maxevents=1| table QueryAttualizzata TempoQuery
```

```
✓ 35.245 eventi (01/01/18 00.00.00,000 - 10/03/18 10.53.07,000)
```

Figure 9: Results of Query 2

### 4.1.3. Query 3 : SQL Query Times (anomalies)

In this query shown in the Figure 10, the maximum value recorded on all queries ran on the system is asked. This value is enough high to be considered an ’outlier’.



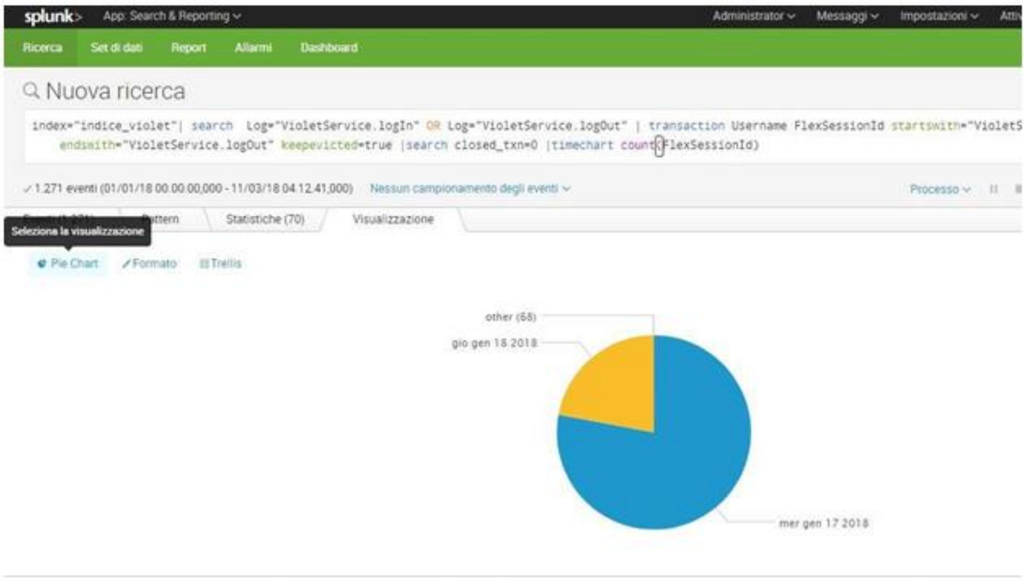
```
index="registry"| timechart max(TempoQuery)
```

```
✓ 196.909 eventi (01/01/18 00.00.00,000 - 10/03/18 02.20.57,000)
```

Figure 10: Results of Query 3

### 4.1.4. Query 4: Incomplete Transactions on Services

In the following search query, carried out on the index log, it was asked to display all incomplete transactions, i.e. all those that have a service login that does not match a service logout, and/or viceversa. In the pie chart in the Figure 11 , there are 1271 total events found, distributed on two dates, 17/1/2018 and 18/1/2018.



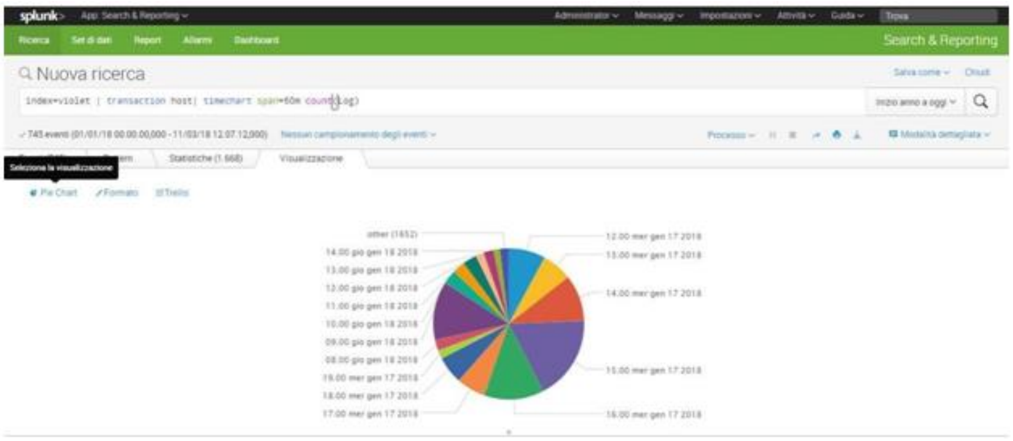
```
index="indice_violet"| search  Log="VioletService.logIn" OR Log="VioletService.logOut" | transaction Username FlexSessionId startswith="VioletS
    endswith="VioletService.logOut" keepevicted=true |search closed_txn=0 |timechart count(FlexSessionId)
```

✓1.271 eventi (01/01/18 00.00.00,000 - 11/03/18 04.12.41,000)

Figure 11: Results of Query 4

### 4.1.5. Query 5 : Measure of Application Load : 'transaction'

In this query (Figure 12), an estimation of the application load is inferred, by assuming that the "Log" field, gives us an approximation of this measure. Since we can't directly measure CPU cycles, the number of service operations, indicated by "Log" field, divided by time slot, is likely one of the possible load measurements.



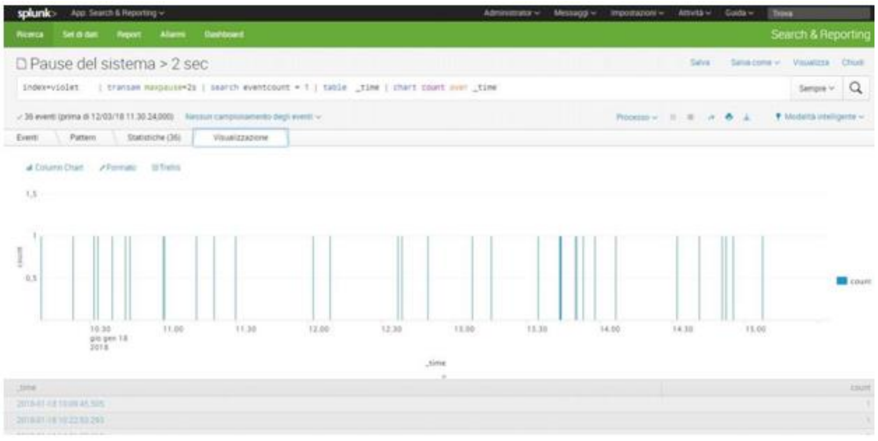
```
index=violet | transaction host| timechart span=60m count(Log)
```

✓745 eventi (01/01/18 00.00.00,000 - 11/03/18 12.07.12,000)

Figure 12: Results of Query 5

### 4.1.6. Query 6 : Application Load Measurement : 'System pauses'

In the following query carried out on the log (Figure 13) , it was asked to highlight all system pauses longer than 2 seconds, and report them with respect to the time on the abscissae axis. The pauses are deduced from the event timestamp. The interpretation given to the result of the query is that there are periodic instants of time, coinciding with the hours or half hours, in which there are breaks greater than 2 seconds. It can be supposed that these application pauses may coincide with higher-level tasks, which the system or application must perform in those deadlines.



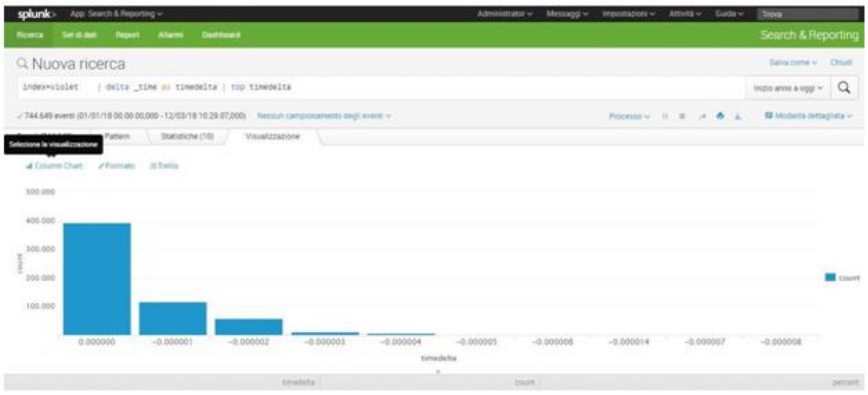
```
index=violet    | transam maxpause=2s | search eventcount = 1 | table _time | chart count over _time
```

✓ 36 eventi (prima di 12/03/18 11.30.24,000)

Figure 13: Results of Query 6

### 4.1.7. Query 7 : Application Load Measurement : ’events concentration’

This query also can be considered a measure of the application effort of our system, as it is a distribution of the time intervals between two successive events. As you can see in Figure 14 , there are more events at a distance of 0 ms, in respect to those at 0.1 ms, and to those at 0.2 ms time distance. This fact highlights a big commitment of the system also, and probably, in a multithreading mode.



```
index=violet    | delta _time as timedelta | top timedelta
```

✓ 744.649 eventi (01/01/18 00.00.00,000 - 12/03/18 10.29.07,000)

Figure 14: Results of Query 7

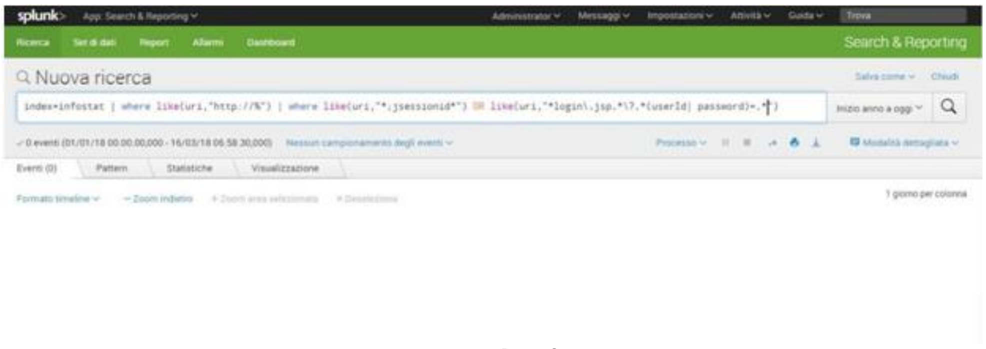
### 4.1.8. Query 8: Broken Authentication and Session Management Control

This query is genuinely a security check to ensure against attacks such as ’Broken Authentication and Session Management’. This security check is done in the web server log file and could be useful for both external and internal access verification. There is a type of attack of this type called ’session fixation attack’ that occurs by passing on the URL, in the request, the session identifiers preset and invalid. In addition, in the search query, neither the user nor the password should be passed in full, as this constitutes an exposure of sensitive data in the URLs logged into the proxie web or stored in the cache. The security control contains the following control strings:

1) /login\.jsp.*\?.*(userId|password)=./

Which is the login.jsp script followed by ? , user and password

2) /;jsessionid=./



```
index=infostat | where like(uri,"http://%") | where like(uri,"*;jsessionid*") OR like(uri,"*login\.jsp.*\?.*(userId| password)=.*")
```

✓ 0 eventi (01/01/18 00:00:00,000 - 16/03/18 06:58:30,000) Nessun campionamento degli eventi

Figure 15: Results of Query 8

In the following pages will be reported further control rules against possible attacks, according to the OWASP classification, potentially taking place in the web farm, and recorded on the logs.

## 5. QUERY EXECUTION TIMES AND PERFORMANCE CONSIDERATIONS

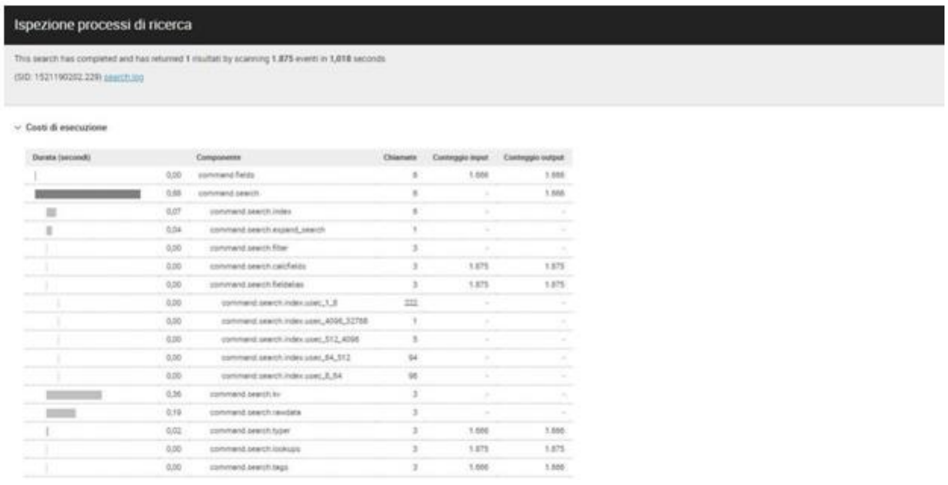
Here below are reported some execution times in order to evaluate Splunk performances and its degree of interactivity with the user. Obviously it should be taken into account that the machines have been designed with very low power compared to what is required by the HW requirements. This is because of cost limitation reasons, and also because of the stress test of the system was not in the main objectives of the present work.

Not all search queries ran in this research prototype are the same in terms of power and resource requirements. For example, the commands such as ’cluster’, ’transaction’, ’correlated’, ’transam’, as well as the search combined with pipes in sequence, absorb a lot of CPU and therefore increase waiting times, which rarely go above a minute even for the most complex queries.

Queries can be optimized to increase performance. In addition, if called in program mode, they are further optimized as they do not engage the web interface.

To examine the time taken just inspect the process performed for the query, as shown below. For the execution of the query : “ERROR" the inspection of the process reports the following times:

1,018 seconds for the execution of the query :



This search has completed and has returned 1 risultati by scanning 1.875 eventi in 1,018 seconds

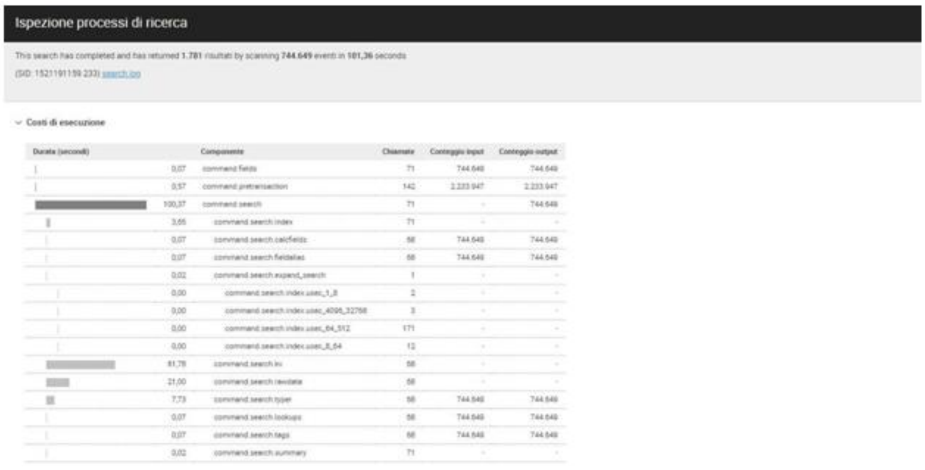
(SID: 1521190202.229) search.log

| Durata (secondi) | Componente | Chiamate | Conteggio input | Conteggio output |
|---|---|---|---|---|
| 0,00 | command.fields | 6 | 1.666 | 1.666 |
| 0,88 | command.search | 6 | - | 1.666 |
| 0,07 | command.search.index | 6 | - | - |
| 0,04 | command.search.expand_search | 1 | - | - |
| 0,00 | command.search.filter | 3 | - | - |
| 0,00 | command.search.calcfields | 3 | 1.875 | 1.875 |
| 0,00 | command.search.fieldalias | 3 | 1.875 | 1.875 |
| 0,00 | command.search.index.usec_1_8 | 222 | - | - |
| 0,00 | command.search.index.usec_4096_32768 | 1 | - | - |
| 0,00 | command.search.index.usec_512_4096 | 5 | - | - |
| 0,00 | command.search.index.usec_64_512 | 94 | - | - |
| 0,00 | command.search.index.usec_8_64 | 96 | - | - |
| 0,36 | command.search.kv | 3 | - | - |
| 0,19 | command.search.rawdata | 3 | - | - |
| 0,02 | command.search.typer | 3 | 1.666 | 1.666 |
| 0,00 | command.search.lookups | 3 | 1.875 | 1.875 |
| 0,00 | command.search.tags | 3 | 1.666 | 1.666 |

Figure 16: Error Process Inspection

to execute the query: 'CaricoApplicativoLog' the process inspection reports the following times:

101.36 seconds for the execution of the query :



**Ispezione processi di ricerca**

This search has completed and has returned **1.781** risultati by scanning **744.649** eventi in **101,36** seconds

(SID: 1521191159.233) search.log

Costi di esecuzione

| Durata (secondi) | Componente | Chiamate | Conteggio input | Conteggio output |
|---|---|---|---|---|
| 0,07 | command.fields | 71 | 744.649 | 744.649 |
| 0,57 | command.pretransaction | 142 | 2.233.947 | 2.233.947 |
| 100,37 | command.search | 71 | - | 744.649 |
| 3,65 | command.search.index | 71 | - | - |
| 0,07 | command.search.calcfields | 58 | 744.649 | 744.649 |
| 0,07 | command.search.fieldalias | 68 | 744.649 | 744.649 |
| 0,02 | command.search.expand_search | 1 | - | - |
| 0,00 | command.search.index.usec_1_8 | 2 | - | - |
| 0,00 | command.search.index.usec_4096_32768 | 3 | - | - |
| 0,00 | command.search.index.usec_64_512 | 171 | - | - |
| 0,00 | command.search.index.usec_8_64 | 12 | - | - |
| 81,78 | command.search.kv | 58 | - | - |
| 21,00 | command.search.rawdata | 58 | - | - |
| 7,73 | command.search.typer | 58 | 744.649 | 744.649 |
| 0,07 | command.search.lookups | 58 | 744.649 | 744.649 |
| 0,07 | command.search.tags | 68 | 744.649 | 744.649 |
| 0,02 | command.search.summary | 71 | - | - |

Figure 17: CaricoApplicativoLog Process Inspection

# 6. THE DASHBOARD AND THE WEB

## 6.1. The Infrastructure Application Dashboard

A Dashboard has been created (Figure 18) , that summarizes the data obtained from previous search queries, inserting the results in quadrants that rapresent the number of errors or warnings, the performances, the application load, the security alerts. The idea is that these values could eventually be summarized in a single value, the Key Performance Indicator (KPI), which indicates the overall performance of the Infrastructure application system and should be linked to the Service Level Agreements established by the organization, to internal and external users.

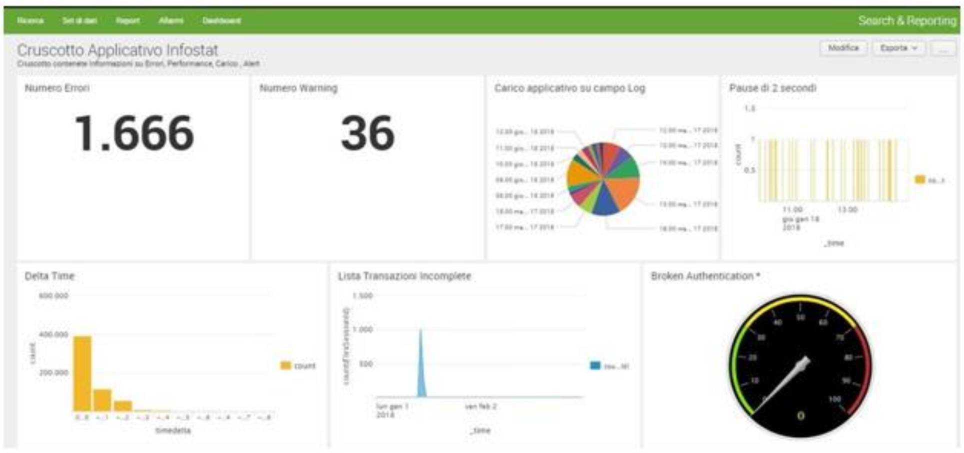


Figure 18: Infrastructure Application Dashboard

## 6.2. Bringing Splunk to the Web

The Dashboard created so far is a synthesis of information extracted from the logs of the Infrastructure application. As it has been realized, it suffers from a certain staticity and therefore the aim of the work was to make its use more interactive in order to allow those users who consult it to have more freedom of action and deepening or drilling into its values. It was considered also as an important point the portability aspect and the easy installation on any machine of the infrastructure. For this reason it was decided to make it an app.

An app in Splunk is nothing more than a set of configuration files.

Even if the dashboard is a way of rendering a lot of information in a synthetic , visual and strategic way, without the need to use SPL commands, it is important to give the user the possibility to explore it further.

For these reasons this APP was created which has its own navigation menu and which displays the application dashboard.

The app, set in a package, has been evaluated as meeting the criteria defined by Splunk with the AppInspect control, and therefore ensures its quality and robustness standards.

It can be activated by clicking on the icon of the Splunk apps installed and it has been 'branded' by inserting the logo of the Department of Informatics. By clicking on this icon the application dashboard, shown in the following image, can be accessed (Figure 19)

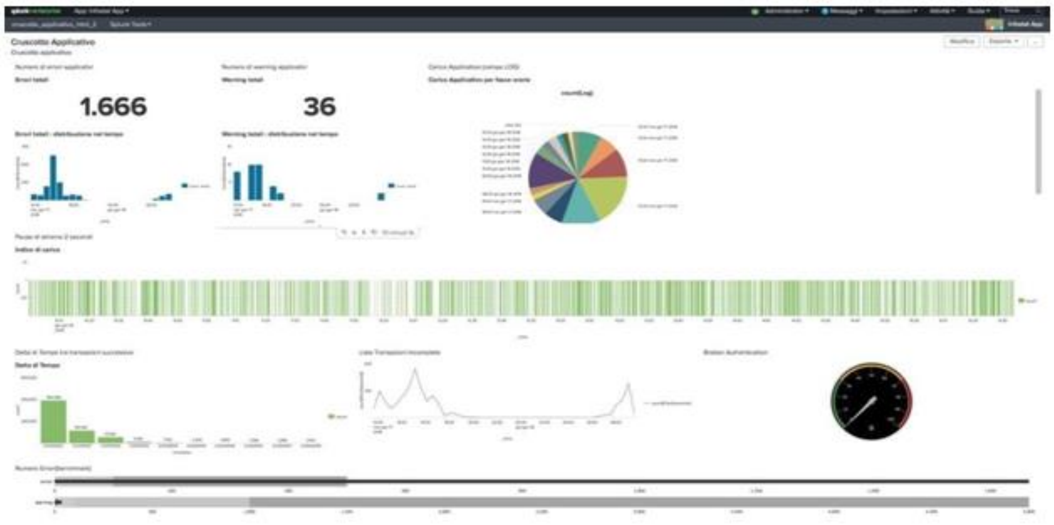


Figure 19: Dynamic Application Dashboard

## 7. HACKER'S ATTACK CONTROL RULES

### 7.1. Control Rules for the Safety of the Infrastructure

The infrastructure architecture we have taken under consideration in our research project is a distributed one, with multiple applications cooperating. It is a Service Oriented Architecture and implements parallelism for the purpose of load balancing and increase of reliability. The Infrastucture is divided into a web farm front, i.e. the component exposed to the Internet, and a server farm, i.e. the internal back-end. The logs analyzed so far are only a very small part of the logs of the Infrastructure. This choice has been made because the logs selected have been considered relevant to the chosen portion of time (two days), and for they were extracted from a single machine, since the attention was limited to a couple of applications, the ones under developers responsibility, who are committed to troubleshooting activities. Only application logs were examined because this was the context in which the author’s work takes place, being himself an application developer.

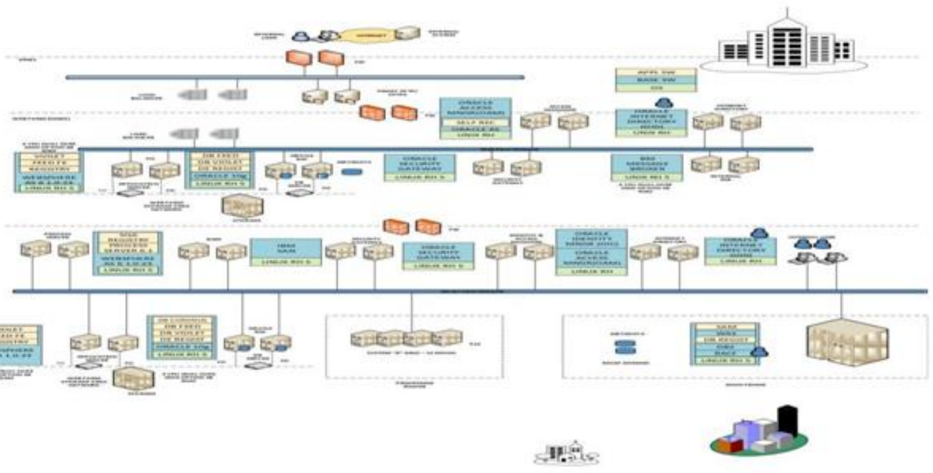


Figure 20: Infrastructure Architecture

The total size of the logs analyzed is about 2GB, covering hundreds of millions of events. The analysis which was carried out , mainly revealed anomalies, and strings were also scanned in order to identify fraudulent authentication attempts.

There is a distinction between the log of a front end and the log of a back end , since as the former has information about connections from external clients (Http), the latter mainly operates connections with databases or web services (SOAP). Obviously the logs of the applications in the web farm can allow to identify the attacks coming from outside, while the logs of the applications in the server farm allows to identify attacks form inside.

In the case of attacks from outside, because the field of intervention is level 7 of the OSI architecture (that is the application layer), and because the information used to determine security attacks consists of logs, the rules that will be activated will be based on the ’uri’ field , in order to identify the major security attacks as cataloged by the organization **OWASP**. These type of attacks, in general, have the purpose of sabotage, or theft of information, or the compromise of the integrity of the data itself. In this case data can be changed and not only stolen. It should be assumed that, in addition to the application logs, should also be scanned the access logs of the Application Server.

### 7.1.1. Cross Site Scripting (XSS) Control Rule

The XSS attack can be identified by detecting HTML tags relevant to the execution of scripts in the uri field. Therefore it is a matter of entering a uri field search with the following SPL commands:

**’index=indexC | where like(uri,’http:/%’) | where like(uri,’*<*>*</*>’) | chart count’**

this search should identify all suspicious requests like : <script>alert(’XSS’)</script> or <h1> alert(’XSS’)</h1>

An alternative method is to explicitly list expressions that can activate active codes or javascript. The SPL control rule in this case should be :

**’index=indexC | where like(uri,’http:/%’) | where like(uri,’*javascript*’) like(uri,’*vbscript*’) OR like(uri,’*applet*’) like(uri,’*script*’) like(uri,’*frame*’) |chart count’**

### 7.1.2. Cross Site Request Forgery (CSRF) Control Rule

There are many applications that are vulnerable to CSRF attack. They are those based on the automatic submission of credentials, as session cookies. The CSRF attack, which therefore presupposes an application with vulnerabilities (as an example we suppose it is hosted on the site www.example.com), can consist of putting a post in a forum with a link of the following type:

**’<img src=https:/www.example.comtrx.do? amt=100&toAcct=1234 width=’0’ height=’0’>’**

The request will start from the forum site and will be directed to the destination site (trying to download the image size 0).

The attack therefore comes from the forum site, at the implicit request of the attacker, who does not perform the action directly.

To identify this attack in an access log file, is possible through the use of the referer. In the log, in fact, it will be present the referrer, that the client reports to be the site from which the reference or the link to the resource, requested. The one which the GET has started. From the examination of the log relative to an application that suffers this type of attack we can note something that usually does not happen, that means that the URL (or URI) request, that carries out the transaction, and the referrer, are belonging to two different sites. The SPL commands will be as follows :

**’index=indexC | where like(uri,’http:/%’) | where IP != referer’**

The referrer value will also be an IP as a result of a lookup table or a workflow action that translates the domain into its IP address.

### 7.1.3. SQL Injection Control Rule

Let’s consider one of the most recurrent injections, even if many others can be identified. Injection is made using the characters () or (), which are delimiters for queries and comments, and also the character (#). The SPL commands used to check the attack will be :´

**’index=indexC | where like(uri,’http:/%’) | where like(uri,’*’*’) OR like(uri,’*(userId)=*’)’ )| OR like(uri,’*(password)=’1’=’1’-’) | chart count’**

A possible attack vector, among many possible, could be as follows: ’) or ’1’=’1. Of course there are endless variations of these values as 1’ or 1<2. The constant part is the single apex followed by the term OR.

### 7.1.4. Mischievous File Execution Check Rule

Since applications can allow the user to provide a filename, or part of a filename, this constitutes a vulnerability in case the input is not validated. Therefore, the attacker can manipulate the filename to run a system program or an external URL. For example, the attacker may attempt to upload an executable file, or a file that calls other files to run or parse. The SPL commands to chech this attack will be :

**’index=indexC | where like(uri,’http:/%’) | where like(uri,’*.jsp*’) OR like(uri,’*.xml*’) | chart count’**

### 7.2. Control Points

Attacks can be identified in different areas of the Infrastructure architecture, and in each of these areas the available logs can be used. For example logs originating from firewall, IDS, DBMS and application firewalls (WAF) can be used to complete the knowledge about an attack.

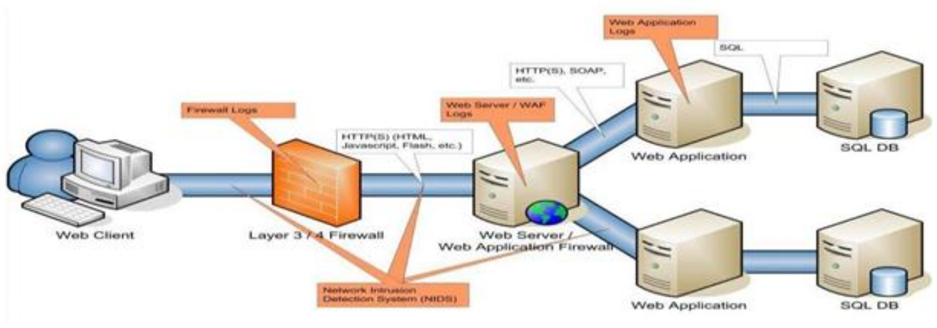


Figure 21: Infrastructure Architecture Control Points

All these rules can only be written if the attack vectors are known before writing down the rules. In this way, we can identify the attacks considered by OWASP to be among the 10 most critical attacks for the applications exposed on the Internet [2]. On the contrary, an approach to identifying statistical anomalies does not require prior knowledge of attack vectors. The nature of the work carried out anyway is experimental and indicative of the potential of the Splunk tool under investigation. It cannot be considered in terms of a product ready for the production environment.

## 8. CONCLUSIONS

My work begun with the aim of enhancing the application log files, thus making them company assets. An asset implies adequate protection mechanisms, professional skills that deal with it, adequate training, standards of writing and compliance. In this work can be assessed how much value can be extracted at an informative level from the analysis of the logs, exclusively considering those logs of applicative nature. Backwards, the work retraced the reverse path of the synthesis of the data analyzed, to make them visible and usable from the perspective of Business Analytics. It was observed if it is more useful in the process of log analysis to “look at the tree or the forest”, and thanks to a tool as powerful as versatile and scalable as Splunk it was concluded that both could be done. Forensic analysis is no longer an investigative activity at Sherlock Holmes way of doing, that is, using a magnifying glass to go deeper and deeper into the details that can trace the anomalous event or fraudulent compromise, but it must be considered an overview. A limit of the present work at the moment is that the analysis is based only on the information contained in the logs and not on all other sources such as online traffic data, scripts execution, API calls, commands, which would provide a certain vision even more complete, but at the moment outside the range of action of the presented project.

## ACKNOWLEDGEMENTS

Special thanks should be given to Dr. M. Bernaschi, Dirigente Tecnologo” of the National Research Council of Italy and also Adjunct Professor of Computer Science at Rome University

"Sapienza", my research project supervisor for his professional guidance and valuable support , his useful and constructive recommendations on this project.

## REFERENCES

[1] Towards structured log analysis, by Dileepa Jayathilake, from International Joint Conference on Computer Science and Software Engineering (JCSSE), Publication Year: 2012, Page(s): 259 – 264

[2] A. Krishna, Splunk Admin & Architect: Complete Tutorials + 30 Days Lab, Udemy, Online.

[3] Exploring Splunk search processing language (SPL) primer and cookbook, by David Carasso, Splunk's Chief Mind, CITO Research New York, NY , 2012

[4] R. Meyer, Detecting Attacks on Web Applications from Log Files, SANS Institute , 2008, Roger Meyer

[5] Splunk and the SANS Top 20 Critical Security Controls, Mapping Splunk Software to the SANS Top 20 CSC Version 4.1, 2014 by Splunk Inc.

[6] Splunk Inc, Splunk Machine Learning Toolkit User Guide 3.1.0. 2018

## AUTHORS

**Dr. Roberto Bruzzese** is an employee of a private institute under Italian public law. He graduated in Computer Science from the University of Bari in 1986. He won a scholarship for scientific research in 1988 and worked for three years as a researcher at Tecnopolis Csata N.O., Valenzano, Italy (1990). Since 1991 he has been working as a developer in the Department of Informatics, of a private institute under Italian public law in Rome, Italy. He subsequently obtained his specialization in Web Technologies at the University of L'Aquila (2015), and a second specialization in Cybersecurity at the University of Rome, La Sapienza. His current fields of study are Internet of Things, Malware Analysis, Ethical Hacking.