

Report finale M6 Analisi del malware e Splunk Manini Sara

Indice

Introduzione.....	2
Query n. 1.....	3
Query n. 2.....	6
Query n. 3.....	9
Query n. 4.....	11
Query n. 5.....	13
Analisi eventi query n. 1.....	15
Analisi eventi query n. 2.....	18
Analisi eventi query n. 3.....	22
Analisi eventi query n. 4.....	25
Analisi eventi query n. 5.....	28

Introduzione

In questa esercitazione si dovranno effettuare delle query specifiche tramite Splunk su un set di log chiamato tutorialdata.zip.

Successivamente verrà svolta un'analisi sui risultati ottenuti per poter trarre conclusioni sugli avvenimenti riportati dai log.

Prerequisiti:

- Installazione di Splunk Enterprise su vm o macchina host;
- Download e caricamento su Splunk del file tutorialdata.zip.

Contenuto generale di tutorialdata.zip

Il file contiene tre tipologie di log:

- access.log: contiene eventi di connessioni verso webserver;
- secure.log: contiene eventi di sicurezza come ad esempio tentativi di accesso;
- vendor_sales.log: contiene informazioni sulle vendite dei prodotti.

Consegna

Costruzione della query

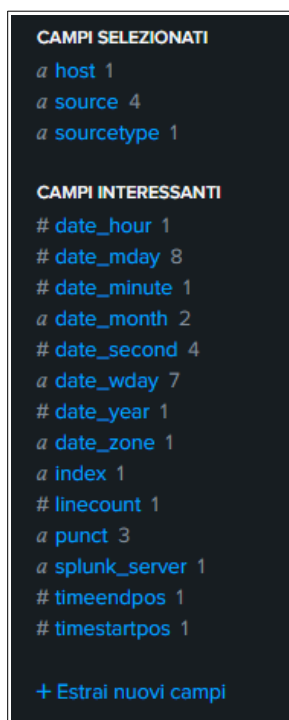
- ```
index="main"
source="tutorialdata.zip*"
| search "Failed password"
```

Andremo quindi a creare una query più precisa in modo da soddisfare la richiesta della consegna.

3

Come si può vedere dall'immagine sottostante, i campi indicizzati da Splunk non sono quelli richiesti.

Per ovviare al problema aggiungeremo alla query il parsing dei campi tramite regex.



### Sintassi della regex:

- Regex completa: `| rex "(?<failure_reason>Failed password for (?:invalid user )?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"`;
- `| rex`: Permette di inserire una regex.

### Sintassi della regex divisa per campi:

- `(?<failure_reason>Failed password for (?:invalid user )?(?<user>\S+))`: Crea il campo **failure\_reason** e inserisce tutto quello che matcha da “Failed password for” fino al nome dell’utente differenziando anche se l’utente inserito era valido o meno;
- `(?<user>\S+)`: crea il campo **user** ed estrae il nome utente che viene dopo;
- `from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})`: Crea il campo **src\_ip** che cattura un IPv4 sorgente (che nel log si trova dopo la parola “from”).

Adesso possiamo fare una ricerca più precisa inserendo “Failed password” come valore da cercare nel campo **failure\_reason** appena creato.

```
index="main"
source="tutorialdata.zip*"
| rex "(?<failure_reason>Failed password for (?:invalid user)?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| search failure_reason="Failed password"
```

Come possiamo vedere dall'immagine sottostante, ora sono presenti tutti i campi richiesti dalla consegna.

< Nascondi campi

Tutti i campi

CAMPI SELEZIONATI

a failure\_reason 100+

a host 1

a source 4

a sourcetype 1

a src\_ip 100+

a user 100+

CAMPI INTERESSANTI

# date\_hour 1

# date\_mday 8

# date\_minute 1

# date\_month 2

# date\_second 4

# date\_wday 7

# date\_year 1

# date\_zone 1

# index 1

# linecount 1

# punct 3

a splunk\_server 1

# timeendpos 1

# timestartpos 1

+ Estrai nuovi campi

Ora

Evento

05/08/25 04:07:07 mailsvl sshd[5276]: Failed password for invalid user appserver from 194.8.74.23 port 3351 ssh2

04:07:07:000

Azioni evento

Tipo

Selezionato

failure\_reason

host

source

sourcetype

src\_ip

user

Failed password for invalid user appserver

Navi

tutorialdata.zip:mailsvl/secure.log

www/secure

194.8.74.23

appserver

Azioni

Ora

Default

\_time

index

linecount

punct

splunk\_server

2025-08-05T04:07:07:000+02:00

main

1

...

Navi

>

05/08/25 04:07:07 mailsvl sshd[1039]: Failed password for root from 194.8.74.23 port 3768 ssh2

04:07:07:000

failure\_reason = Failed password for root | host = Navi | source = tutorialdata.zip:mailsvl/secure.log | sourcetype = www/secure | src\_ip = 194.8.74.23 | user = root

>

05/08/25 04:07:07 mailsvl sshd[5258]: Failed password for invalid user testuser from 194.8.74.23 port 3626 ssh2

04:07:07:000

failure\_reason = Failed password for invalid user testuser | host = Navi | source = tutorialdata.zip:mailsvl/secure.log | sourcetype = www/secure | src\_ip = 194.8.74.23 | user = testuser

Per avere un risultato ancora più pulito ed immediato per la lettura, andremo a creare una tabella contenente tutti i campi richiesti tramite la funzione table di Splunk.

```
index="main"
source="tutorialdata.zip*"
| rex "(?<failure_reason>Failed password for (?<invalid user >)?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| search failure_reason="Failed password*"
| table _time src_ip user failure_reason
```

La tabella risultante dalla query finale si presenterà in questo modo.

| _time               | src_ip      | user      | failure_reason                             |
|---------------------|-------------|-----------|--------------------------------------------|
| 2025-08-05 04:07:07 | 194.8.74.23 | appserver | Failed password for invalid user appserver |
| 2025-08-05 04:07:07 | 194.8.74.23 | root      | Failed password for root                   |
| 2025-08-05 04:07:07 | 194.8.74.23 | testuser  | Failed password for invalid user testuser  |
| 2025-08-05 04:07:07 | 194.8.74.23 | apache    | Failed password for apache                 |
| 2025-08-05 04:07:07 | 194.8.74.23 | mongodb   | Failed password for invalid user mongodb   |
| 2025-08-05 04:07:07 | 194.8.74.23 | mail      | Failed password for mail                   |
| 2025-08-05 04:07:07 | 194.8.74.23 | games     | Failed password for games                  |
| 2025-08-05 04:07:07 | 194.8.74.23 | desktop   | Failed password for invalid user desktop   |
| 2025-08-05 04:07:07 | 194.8.74.23 | nagios    | Failed password for nagios                 |
| 2025-08-05 04:07:07 | 194.8.74.23 | cyrus     | Failed password for invalid user cyrus     |
| 2025-08-05 04:07:07 | 194.8.74.23 | guest     | Failed password for invalid user guest     |
| 2025-08-05 04:07:07 | 194.8.74.23 | itmadmin  | Failed password for invalid user itmadmin  |
| 2025-08-05 04:07:07 | 194.8.74.23 | inet      | Failed password for invalid user inet      |
| 2025-08-05 04:07:07 | 194.8.74.23 | operator  | Failed password for invalid user operator  |

5

## Query n. 2

### Consegna

Scrivi una query Splunk per trovare tutte le sessioni SSH aperte con successo.  
La query dovrebbe filtrare per l'utente "djohnson" e mostrare il timestamp e l'ID utente.

### Costruzione della query

Anche in questo caso partiamo da una query generica che comprenda però una ricerca per il valore "djohnson", come richiesto.

```
index="main"
source="tutorialdata.zip*"
| search "djohnson"
```

I log risultanti dalla ricerca si presentano come da immagine sottostante.

Come si può notare il campo **uid** che identifica l'ID utente è già presente, abbiamo però di nuovo la stessa problematica dei campi mancanti per poter eseguire la query.

Il passo successivo sarà quindi quello di andare a formulare una regex adeguata per aggiungere ciò che manca.

The screenshot shows the Splunk search results interface. On the left, there are two panels: 'CAMPI SELEZIONATI' (Selected Fields) and 'CAMPI INTERESSANTI' (Interesting Fields). The main panel displays a list of search results for the query. The first result is expanded, showing a configuration table for the event.

| Tipo        | Campo         | Valore                              | Azioni |
|-------------|---------------|-------------------------------------|--------|
| Selezionato | host          | Navi                                |        |
|             | source        | tutorialdata.zip.*mailsv/secure.log |        |
|             | uid           | 0                                   |        |
| Ora         | _time         | 2025-08-05T04:07:00.000+02:00       |        |
| Default     | index         | main                                |        |
|             | linecount     | 1                                   |        |
|             | punct         | ... [ ] { } ( ) (=)                 |        |
|             | sourcetype    | www1/secure                         |        |
|             | splunk_server | Navi                                |        |

The search results list shows the following events:

- 05/08/25 04:07:00:000 Thu Aug 05 2025 04:07:07 mailsv1 sshd[60445]: pam\_unix(sshd:session): session opened for user djohnson by (uid=0)
- 05/08/25 04:07:00:000 Thu Aug 05 2025 04:07:07 mailsv1 sshd[87066]: pam\_unix(sshd:session): session opened for user djohnson by (uid=0)
- 05/08/25 04:07:00:000 Thu Aug 05 2025 04:07:07 mailsv1 sudo: djohnson : TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bin/su

*Sintassi della regex:*

- **Regex completa:** | rex "pam\_unix\(((?<protocol>[^\:]+):session\): (?<session\_result>session [^\ ]+) for user (?<username>\S+)"

*Sintassi della regex divisa per campi:*

- `pam_unix\((?<protocol>[^\:]+):session\)`: cerca partendo da “pam unix” ed inserisce nel campo **protocol** ciò che segue fermandosi prima di “session”;
- `(?<session_result>session [^\ ]+)`: crea il campo **session\_result** prendendo la parola “session” e la parola seguente e così determinando lo stato della sessione;
- `for user (?<username>\S+)`: cerca partendo da “for user” ed inserisce nel campo **username** la parola che segue.

Possiamo ora inserire nella ricerca i valori richiesti nei campi appena creati.

```
index="main"
source="tutorialdata.zip*"
| rex "pam_unix\((?<protocol>[^\:]+):session\): (?<session_result>session [^\]+ for user (?<username>\S+)"
| search username="djohnson" session_result="session opened*" protocol="sshd"
```

Il risultato della query sarà il seguente.

< Nascondi campi

Tutti i campi

CAMPI SELEZIONATI

a host 1

a protocol 1

a session\_result 2

a source 4

# uid 1

a username 1

CAMPI INTERESSANTI

# date\_hour 1

# date\_mday 8

# date\_minute 1

a date\_month 2

# date\_second 4

a date\_wday 7

# date\_year 1

a date\_zone 1

a index 1

# linecount 1

a punct 1

a sourcetype 1

a splunk\_server 1

# timeendpos 1

# timestartpos 1

+ Estrai nuovi campi

i

Ora

▼

05/08/25

04:07:07,000

Evento

Thu Aug 05 2025 04:07:07 mailsv1 sshd[60445]: pam\_unix(sshd:session): session opened for user djohnson by (uid=0)

Azioni evento ▼

| Tipo        | <input checked="" type="checkbox"/> Campo            | Valore                               | Azioni |
|-------------|------------------------------------------------------|--------------------------------------|--------|
| Selezionato | <input checked="" type="checkbox"/> host ▼           | Navi                                 | ▼      |
|             | <input checked="" type="checkbox"/> protocol ▼       | sshd                                 | ▼      |
|             | <input checked="" type="checkbox"/> session_result ▼ | session opened                       | ▼      |
|             | <input checked="" type="checkbox"/> source ▼         | tutorialdata.zip..mmailsv/secure.log | ▼      |
|             | <input checked="" type="checkbox"/> uid ▼            | 0                                    | ▼      |
|             | <input checked="" type="checkbox"/> username ▼       | djohnson                             | ▼      |
| Ora ⌕       | _time ▼                                              | 2025-08-05T04:07:07.000+02:00        |        |
| Default     | <input type="checkbox"/> index ▼                     | main                                 | ▼      |
|             | <input type="checkbox"/> linecount ▼                 | 1                                    | ▼      |
|             | <input type="checkbox"/> punct ▼                     | _____[:f)_____(*                     | ▼      |
|             | <input type="checkbox"/> sourcetype ▼                | www1/secure                          | ▼      |
|             | <input type="checkbox"/> splunk_server ▼             | Navi                                 | ▼      |

>

05/08/25

04:07:07,000

Thu Aug 05 2025 04:07:07 mailsv1 sshd[87866]: pam\_unix(sshd:session): session opened for user djohnson by (uid=0)

host = Navi | protocol = sshd | session\_result = session opened | source = tutorialdata.zip..mmailsv/secure.log | uid = 0 | username = djohnson

>

05/08/25

04:07:07,000

Thu Aug 05 2025 04:07:07 mailsv1 sshd[5860]: pam\_unix(sshd:session): session opened for user djohnson by (uid=0)

host = Navi | protocol = sshd | session\_result = session opened | source = tutorialdata.zip..mmailsv/secure.log | uid = 0 | username = djohnson

Come fatto in precedenza, andiamo a creare una tabella con la seguente sintassi.

```
index="main"
source="tutorialdata.zip*"
| rex "pam_unix\((?<protocol>[^\:]+):session\): (?<session_result>session [^\]+) for user (?<username>\S+)"
| search username="djohnson" session_result="session opened*" protocol="ssh"
| table _time username uid protocol session_result
```

La tabella creata si presenterà nel seguente modo.

| _time ↕             | username ↕ ✓ | uid ↕ ✓ | protocol ↕ | session_result ↕ |
|---------------------|--------------|---------|------------|------------------|
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson     | 0       | sshd       | session opened   |



## Query n. 3

### Consegna

Scrivi una query Splunk per trovare tutti i tentativi di accesso falliti provenienti dall'indirizzo IP "86.212.199.60". La query dovrebbe mostrare il timestamp, il nome utente e il numero di porta.

### Costruzione della query

Partiamo sempre da una query generica per poter studiare la struttura del log prima di raffinare ulteriormente la query.

Assicuriamoci però di comprendere i valori "86.212.199.60" e "Failed password", come richiesto.

```
index="main"
source="tutorialdata.zip*"
| search "86.212.199.60" "Failed password"
```

I log si presentano in questo modo, come visto per le query precedenti, il prossimo passo sarà quello di creare la regex adatta che ci permetta di indicizzare i campi richiesti dalla consegna.

| Ora                   | Evento                                                                                                                  |
|-----------------------|-------------------------------------------------------------------------------------------------------------------------|
| 05/08/25 04:07:07.000 | Thu Aug 05 2025 04:07:07 mailsv1 sshd[5728]: Failed password for invalid user agushto from 86.212.199.60 port 3692 ssh2 |
| 05/08/25 04:07:07.000 | Thu Aug 05 2025 04:07:07 mailsv1 sshd[4843]: Failed password for invalid user tomcat from 86.212.199.60 port 1464 ssh2  |
| 05/08/25 04:07:07.000 | Thu Aug 05 2025 04:07:07 mailsv1 sshd[5718]: Failed password for invalid user desktop from 86.212.199.60 port 3518 ssh2 |

### Sintassi della regex:

Per la creazione di questa regex è stata modificata quella che abbiamo usato per la query n.1 in modo che potesse filtrare anche i campi richiesti da questa consegna.

- Regex completa: `| rex "(?<failure_reason>Failed password for (?<invalid_user>invalid user )?(?<user>\S+)) from (?<src_ip>\d{1,3}(\.\d{1,3}){3}) port (?<src_port>\d+)"`

## Sintassi della regex divisa per campi:

Per i campi `failure_reason`, `user` e `src_ip` abbiamo riutilizzato le parti della regex del quesito n.1, di seguito la sintassi per il campo `src_port`.

- `port (?<src_port>\d+)`: cerca “port” nel log, crea il campo **src\_port** e cattura ciò che segue.

Possiamo quindi aggiungere alla ricerca i valori per i campi **failure\_reason** e **src\_ip**.

```
index=main
source="tutorialdata.zip:*"
| rex "(?<failure_reason>Failed password for (?<invalid user >?)?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<src_port>\d+)"
| search failure_reason="Failed password*" src_ip="86.212.199.60"
```

I log risultanti dalla query si presenteranno nel seguente modo.

The screenshot shows the Splunk interface with search results for failed password attempts. The left sidebar lists fields like `failure_reason`, `host`, `source`, `src_ip`, and `src_port`. The main panel displays event details for a specific event (05/08/25 04:07:0000) where the failure reason is "Failed password for invalid user agusto" from IP 86.212.199.60 on port 3692. Below this, a table lists multiple events with columns for `_time`, `failure_reason`, `host`, `source`, `src_ip`, and `src_port`.

| Tipo        | Campo          | Valore                                  | Azioni |
|-------------|----------------|-----------------------------------------|--------|
| Selezionato | failure_reason | Failed password for invalid user agusto |        |
|             | host           | Navi                                    |        |
|             | source         | tutorialdata.zip:\malls\secure.log      |        |
|             | src_ip         | 86.212.199.60                           |        |
|             | src_port       | 3692                                    |        |
| Evento      | user           | agusto                                  |        |
| Ora         | _time          | 2025-08-05T04:07:00.000+02:00           |        |
| Default     | index          | main                                    |        |
|             | linecount      | 1                                       |        |
|             | punct          |                                         |        |
|             | splunk_server  | Navi                                    |        |

| _time               | failure_reason                           | host | source                             | src_ip        | src_port |
|---------------------|------------------------------------------|------|------------------------------------|---------------|----------|
| 05/08/25 04:07:0000 | Failed password for invalid user tomcat  | Navi | tutorialdata.zip:\malls\secure.log | 86.212.199.60 | 1464     |
| 05/08/25 04:07:0000 | Failed password for invalid user desktop | Navi | tutorialdata.zip:\malls\secure.log | 86.212.199.60 | 3518     |

Anche in questo caso andiamo a creare una tabella riassuntiva dei risultati chiesti dalla consegna utilizzando la sintassi contenuta nell’immagine sottostante.

```
index=main
source="tutorialdata.zip:*"
| rex "(?<failure_reason>Failed password for (?<invalid user >?)?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<src_port>\d+)"
| search failure_reason="Failed password*" src_ip="86.212.199.60"
| table _time src_ip user src_port
```

Di seguito il risultato della query.

| _time               | src_ip        | user     | src_port |
|---------------------|---------------|----------|----------|
| 2025-08-05 04:07:07 | 86.212.199.60 | agusto   | 3692     |
| 2025-08-05 04:07:07 | 86.212.199.60 | tomcat   | 1464     |
| 2025-08-05 04:07:07 | 86.212.199.60 | desktop  | 3518     |
| 2025-08-05 04:07:07 | 86.212.199.60 | yp       | 2856     |
| 2025-08-05 04:07:07 | 86.212.199.60 | mail     | 1054     |
| 2025-08-05 04:07:07 | 86.212.199.60 | apache   | 2630     |
| 2025-08-05 04:07:07 | 86.212.199.60 | services | 4740     |
| 2025-08-05 04:07:07 | 86.212.199.60 | irc      | 1203     |
| 2025-08-05 04:07:07 | 86.212.199.60 | mysql    | 4802     |
| 2025-08-05 04:07:07 | 86.212.199.60 | pmuser   | 1775     |
| 2025-08-05 04:07:07 | 86.212.199.60 | ventrilo | 1465     |
| 2025-08-05 04:07:07 | 86.212.199.60 | system   | 3305     |

## Consegna

## Costruzione della query

Andiamo ad aggiungere una ricerca con il valore “Failed password” nel campo **failure\_reason**.

I log si presenteranno in questo modo.

11

Andiamo a svolgere il resto della consegna, modificando la query come segue:

- | stats count by src\_ip: conta i tentativi che esegue ogni ip;
- | where count > 5: come da consegna inserisce solo gli ip che hanno effettuato più di cinque tentativi;
- | sort - count: ordina i risultati della colonna count in ordine decrescente.

```
index="main"
source="tutorialdata.zip*"
| rex "(?<failure_reason>Failed password for (?<invalid user >)(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<src_port>\d+)"
| search failure_reason="Failed password*"
| stats count by src_ip
| where count > 5
| sort - count
```

Il risultato sarà la seguente tabella, contenente due colonne, una per gli IP sorgente e una per il numero di tentativi di accesso falliti, con in cima l'IP che ne ha fallito il maggior numero.

| src_ip ↕       | count ↕ |
|----------------|---------|
| 87.194.216.51  | 948     |
| 211.166.11.101 | 743     |
| 128.241.220.82 | 622     |
| 109.169.32.135 | 515     |
| 194.215.205.19 | 514     |
| 216.221.226.11 | 433     |
| 188.138.40.166 | 297     |
| 65.19.167.94   | 286     |
| 107.3.146.207  | 282     |
| 95.130.170.231 | 279     |
| 223.205.219.67 | 274     |
| 27.1.11.11     | 273     |
| 27.35.11.11    | 270     |
| 59.162.167.100 | 267     |
| 91.210.104.143 | 253     |
| 27.101.11.11   | 251     |
| 108.65.113.83  | 249     |

## Query n. 5

### Consegna

Crea una query Splunk per trovare tutti gli Internal Server Error.

### Costruzione della query

Per costruire questa query è bastato inserire all'interno della stessa il codice che identifica gli errori interni di un server e cioè il codice 500.

La sintassi è come segue:

- `| where status="500"`: cerca solamente il codice 500 nel campo **status**.

```
index="main"
source="tutorialdata.zip*"
| where status="500"
```

Per avere dei risultati più parlanti sono stati aggiunti i seguenti campi:

- **clientip**: mostra l'IP sorgente;
- **method**: mostra se la richiesta è una GET o una POST;
- **status**: mostra il response code.

I log si presentano come segue.

The screenshot shows the Splunk search results interface. On the left, there are two panels: 'CAMPI SELEZIONATI' (Selected Fields) and 'CAMPI INTERESSANTI' (Interesting Fields). The 'Selected Fields' panel lists fields like clientip, host, method, source, sourcetype, and status. The 'Interesting Fields' panel lists fields like action, bytes, categoryid, date\_hour, date\_minute, date\_month, date\_second, date\_weekday, date\_year, date\_zone, file, ident, index, itemid, jsessionid, and responsecode. The main panel displays a single log entry for an HTTP 500 error. The log entry is as follows:

| Ora                   | Evento                                                                                                                                                                                                                                                                                                                 |
|-----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 05/08/25 18:18:59,000 | 198.35.1.75 - - [05/Aug/2025:18:18:59] "GET /cart.do?action=addtocart&itemId=EST-13&jSESSIONID=SD10SL2FF4ADFF53099 HTTP/1.1" 500 2324 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 645 |

Below the log entry, there is a table showing the fields and their values for the selected event. The table has three columns: 'Tipo', 'Campo', 'Valore', and 'Azioni'.

| Tipo        | Campo      | Valore                                                        | Azioni |
|-------------|------------|---------------------------------------------------------------|--------|
| Selezionato | clientip   | 198.35.175                                                    |        |
|             | host       | Navi                                                          |        |
|             | method     | GET                                                           |        |
|             | source     | tutorialdata.zip_wwwfaccess.log                               |        |
|             | sourcetype | access_combined_wcookie                                       |        |
|             | status     | 500                                                           |        |
| Evento      | jSESSIONID | SD10SL2FF4ADFF53099                                           |        |
|             | action     | addtocart                                                     |        |
|             | bytes      | 2324                                                          |        |
|             | categoryid | NULL                                                          |        |
|             | file       | cart.do                                                       |        |
|             | ident      | -                                                             |        |
|             | itemid     | EST-13                                                        |        |
|             | other      | 645                                                           |        |
|             | referrer   | http://www.buttercupgames.com/category.screen?categoryId=NULL |        |

Come per le altre query, andiamo a creare una tabella con le informazioni più salienti.

```
index="main"
source="tutorialdata.zip*"
| where status="500"
| table _time clientip method status
```

La tabella quindi mostrerà la data, l'IP sorgente, il metodo della richiesta e il response code che corrisponde all'internal server error.

| _time ↕             | clientip ↕      | method ↕ | status ↕ |
|---------------------|-----------------|----------|----------|
| 2025-08-05 18:18:59 | 198.35.1.75     | GET      | 500      |
| 2025-08-05 18:18:55 | 198.35.1.75     | GET      | 500      |
| 2025-08-05 17:42:03 | 125.89.78.6     | POST     | 500      |
| 2025-08-05 17:17:00 | 194.146.236.22  | POST     | 500      |
| 2025-08-05 17:15:13 | 121.254.179.199 | POST     | 500      |
| 2025-08-05 16:54:07 | 76.89.103.115   | GET      | 500      |
| 2025-08-05 16:33:02 | 59.36.99.70     | GET      | 500      |
| 2025-08-05 16:05:47 | 201.42.223.29   | GET      | 500      |
| 2025-08-05 15:44:56 | 188.173.152.100 | GET      | 500      |
| 2025-08-05 15:26:14 | 187.231.45.62   | POST     | 500      |

# Analisi eventi query n. 1

Impostando come range temporale “Sempre”, la query fornisce un totale di 33.069 eventi, distribuiti su 8 giorni dal 29 Luglio al 5 Agosto.



Aggiungendo alla query “| stats count by \_time”, vengono raggruppati i risultati per data. Possiamo notare che i tentativi vengono effettuati sempre alle 04:07 di tutti i giorni in questione, con una minima variazione sui secondi. Ciò significa che l’attività è stata eseguita molto probabilmente tramite uno script schedulato per essere eseguito ogni giorno alla stessa ora, fuori dall’orario lavorativo standard.

| _time               | count |
|---------------------|-------|
| 2025-08-02 04:07:05 | 2553  |
| 2025-08-03 04:07:05 | 2270  |
| 2025-08-04 04:07:05 | 2197  |
| 2025-08-05 04:07:05 | 1683  |
| 2025-08-01 04:07:05 | 1399  |
| 2025-08-03 04:07:06 | 1331  |
| 2025-08-04 04:07:07 | 1324  |
| 2025-07-31 04:07:05 | 1312  |

Ora andremo a modificare la query in modo che la regex estragga anche l’**hostname** su cui sono stati effettuati i tentativi di accesso. Aggiungiamo quindi alla regex la seguente parte:

- (?<hostname>\S+)\s+sshd(?:\[\d+\])?:\s\*: crea il campo **hostname** ed estrae il valore prima di “sshd”.

```
index="main"
source="tutorialdata.zip*"
| rex "(?<hostname>\S+)\s+sshd(?:\[\d+\])?:\s*(?<failure_reason>Failed password for (?<invalid user >)?(?<user>\S+)) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3})"
| search failure_reason="Failed password"
```

Così facendo sarà possibile visualizzare gli **hostname** target coinvolti.

| Valori  | Conteggio | %       |
|---------|-----------|---------|
| www1    | 8.746     | 26,448% |
| www3    | 8.220     | 24,857% |
| mailsv1 | 8.111     | 24,528% |
| www2    | 7.992     | 24,168% |

Di seguito i primi 10 IP maggiormente coinvolti, classificati tutti come clean da fonti OSINT (abuseipdb) tranne per l'IP 95.130.170.231 che ha una segnalazione per brute force.

| Primi 10 valori | Conteggio | %      |  |
|-----------------|-----------|--------|--|
| 87.194.216.51   | 948       | 2,867% |  |
| 211.166.11.101  | 743       | 2,247% |  |
| 128.241.220.82  | 622       | 1,881% |  |
| 109.169.32.135  | 515       | 1,557% |  |
| 194.215.205.19  | 514       | 1,554% |  |
| 216.221.226.11  | 424       | 1,282% |  |
| 65.19.167.94    | 286       | 0,865% |  |
| 188.138.40.166  | 283       | 0,856% |  |
| 107.3.146.207   | 282       | 0,853% |  |
| 95.130.170.231  | 279       | 0,844% |  |

Andando ad analizzare l'utenza più colpita possiamo vedere che gli attaccanti hanno cercato di effettuare login su root o altre utenze con privilegi elevati.

Questo può essere riconducibile ad uno script di brute force automatico che effettua tentativi su un dizionario di possibili nomi utente e password.

| Primi 10 valori                                | Conteggio | %      |  |
|------------------------------------------------|-----------|--------|--|
| Failed password for root                       | 1.493     | 4,515% |  |
| Failed password for invalid user administrator | 1.020     | 3,084% |  |
| Failed password for invalid user admin         | 938       | 2,836% |  |
| Failed password for invalid user operator      | 923       | 2,791% |  |
| Failed password for mail                       | 753       | 2,277% |  |
| Failed password for invalid user mailman       | 752       | 2,274% |  |
| Failed password for invalid user irc           | 644       | 1,947% |  |
| Failed password for invalid user email         | 626       | 1,893% |  |
| Failed password for games                      | 601       | 1,817% |  |
| Failed password for invalid user sys           | 586       | 1,772% |  |



## Conclusione eventi query n. 1

In base ai dati raccolti e alla conseguente analisi, possiamo concludere che i tentativi di accesso sono stati eseguiti probabilmente tramite una botnet controllata da un server C2.

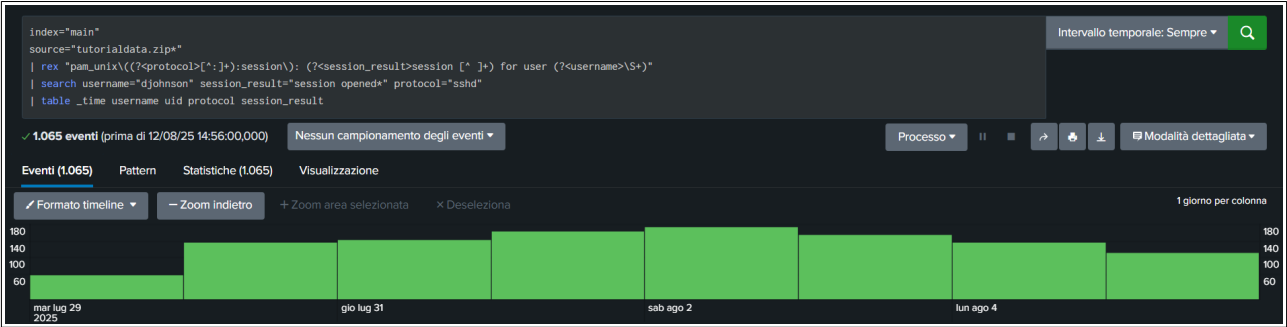
L'attacco viene eseguito tramite uno script automatizzato che esegue richieste di login utilizzando un dizionario con possibili username e password.

## Azioni immediate da intraprendere

- Instaurare una regola sul firewall che blocchi attacchi massivi in un breve arco temporale;
- Scansionare i server afflitti dall'attacco;
- Implementare regole di password sicure;
- Disattivare utenze amministrative con nome comune (root, admin, administrator);
- Utilizzare dove possibile l'MFA.

# Analisi eventi query n. 2

Impostando come range temporale “Sempre”, la query fornisce un totale di 1.065 eventi, distribuiti su 8 giorni dal 29 Luglio al 5 Agosto.



La prima cosa da prendere in esame è l’orario in cui sono state aperte le sessioni ssh. Anche in questo caso l’orario di esecuzione risulta essere sempre le 04:07 di tutti i giorni con un numero di tentativi al secondo che fa pensare più ad uno script che ad un utente umano. Dalla tabella sottostante possiamo anche vedere che lo uid della sessione è 0, cioè root.

| _time ↕             | username ↕ | uid ↕ | protocol ↕ | session_result ↕ |
|---------------------|------------|-------|------------|------------------|
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |
| 2025-08-05 04:07:07 | djohnson   | 0     | sshd       | session opened   |

Eseguiamo quindi una query generica, immettendo nel campo di ricerca solo il nome dell’utente, con la sintassi presente nell’immagine subito sotto.

```
index="main"
source="tutorialdata.zip*"
| search "djohnson"
```

Andiamo ad applicare alla query appena scritta il campo **COMMAND** ed il valore “/bin/su”.

COMMAND

1 Valore, 100% di eventi

Selezionato

Si

No

Report

Primi valori

Primi valori nel tempo

Valori rari

Eventi con questo campo

| Valori  | Conteggio | %    |
|---------|-----------|------|
| /bin/su | 203       | 100% |

Così facendo avremo modo di analizzare dei log che riportano che l'utente djohnson ha eseguito il comando "su" ovvero il comando che nei sistemi a base Linux permette ad un utente che possiede i privilegi di root di eseguire comandi con permessi amministrativi.

05/08/25  
04:07:07,000

Thu Aug 05 2025 04:07:07 mailsv1 sudo: djohnson ; TTY=pts/0 ; PWD=/home/djohnson ; USER=root ; COMMAND=/bin/su

Azioni evento ▾

| Tipo        | <input checked="" type="checkbox"/> | Campo           | Valore                              | Azioni |
|-------------|-------------------------------------|-----------------|-------------------------------------|--------|
| Selezionato | <input checked="" type="checkbox"/> | COMMAND ▾       | /bin/su                             | ▾      |
|             | <input checked="" type="checkbox"/> | PWD ▾           | /home/djohnson                      | ▾      |
|             | <input checked="" type="checkbox"/> | USER ▾          | root                                | ▾      |
|             | <input checked="" type="checkbox"/> | host ▾          | Navi                                | ▾      |
|             | <input checked="" type="checkbox"/> | source ▾        | tutorialdata.zip:\mailsv\secure.log | ▾      |
|             | <input checked="" type="checkbox"/> | sourcetype ▾    | www1/secure                         | ▾      |
| Evento      | <input type="checkbox"/>            | TTY ▾           | pts/0                               | ▾      |
| Ora ⚙       |                                     | _time ▾         | 2025-08-05T04:07:07,000+02:00       |        |
| Default     | <input type="checkbox"/>            | index ▾         | main                                | ▾      |
|             | <input type="checkbox"/>            | linecount ▾     | 1                                   | ▾      |
|             | <input type="checkbox"/>            | punct ▾         | _____::_____:._:=/_.:_:=/_:=//      | ▾      |
|             | <input type="checkbox"/>            | splunk_server ▾ | Navi                                | ▾      |

Aggiungiamo “password” nella ricerca generica che abbiamo appena fatto.

```
index="main" source="tutorialdata.zip*" | search "djohnson" "password"
```

In questo modo avremo come risultato dei log che presentano i tentativi di accesso in ssh falliti e riusciti dall'utenza, in tutti i log l'indirizzo IP sorgente è 10.3.10.46.  
Come possiamo notare l'IP è un privato, ciò significa che le connessioni arrivavano dalla rete interna.

|   |                          |                                                                                                                                                                                                   |
|---|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[54545]: Accepted password for djohnson from 10.3.10.46 port 5143 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[90328]: Accepted password for djohnson from 10.3.10.46 port 3914 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[52473]: Accepted password for djohnson from 10.3.10.46 port 5449 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[96461]: Accepted password for djohnson from 10.3.10.46 port 3041 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[1269]: Accepted password for djohnson from 10.3.10.46 port 2652 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure  |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[92618]: failed password for djohnson from 10.3.10.46 port 9930 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure   |
| > | 05/08/25<br>04:07:07,000 | Thu Aug 05 2025 04:07:07 mailsrv1 sshd[56370]: failed password for djohnson from 10.3.10.46 port 2187 ssh2<br>host = Navi source = tutorialdata.zip:\mailsv/secure.log sourcetype = www1/secure   |

## Conclusioni eventi query n. 2

Visto il numero troppo elevato di connessioni ssh simultanee da parte dell'utente djohnson in orari non lavorativi possiamo ipotizzare che l'utenza sia stata compromessa e che venga utilizzata per aprire un gran numero di sessioni ssh parallele in un tentativo di saturare il sistema.

L'utilizzo di un IP privato implica una possibile compromissione di una macchina interna tramite malware o tramite sessione remota.

L'utente djohnson è inoltre nel gruppo sudo, è infatti possibile ottenere i permessi amministrativi tramite il comando "su" (o "sudo su"), azione che l'attaccante ha effettuato.

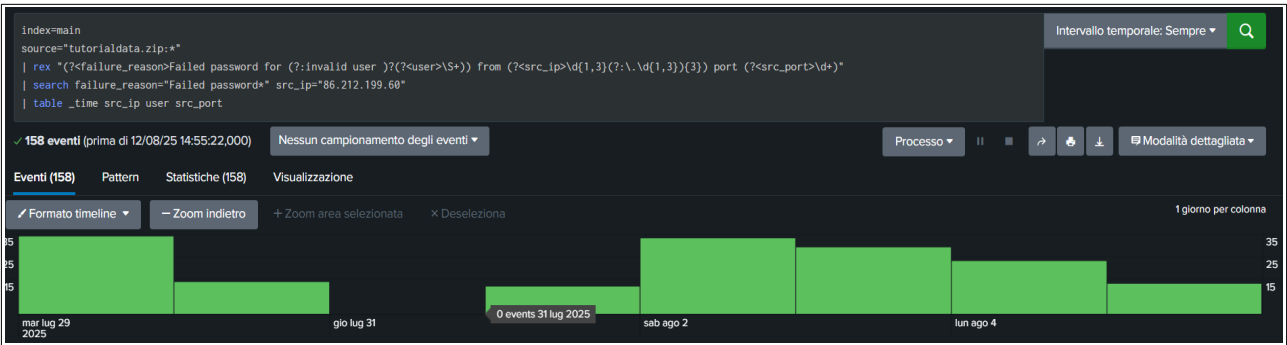
## Azioni immediate da intraprendere

- Trovare la macchina con l'IP 10.3.10.46 assegnato ed eseguire sia una scansione completa che un isolamento del sistema;
- Bloccare l'IP 10.3.10.46;
- Disattivare l'utenza djohnson e cambiare le credenziali (implementando MFA).

# Analisi eventi query n. 3

Impostando come range temporale “Sempre”, la query fornisce un totale di 158 eventi, distribuiti su 8 giorni dal 29 Luglio al 5 Agosto.

Come si può vedere dal grafico sottostante il giorno 31 Luglio non presenta eventi.



Come possiamo vedere dalla tabella seguente, sia il fatto che i tentativi di accesso sono così ravvicinati nell’orario (orario non lavorativo standard), sia che sono stati fatti tutti dallo stesso IP ma con username diversi, ci porta a pensare ad un attacco a dizionario eseguito tramite script.

È stato aggiunto il campo **failure\_reason** in modo che si potesse discernere tra i tentativi di login su utenti esistenti e su quelli che invece non sono presenti nel sistema.

È stata inoltre fatta una ricerca tramite fonti OSINT (abuseipdb) sull’IP 86.212.199.60 e non è risultata nessuna segnalazione.

| _time ↕             | src_ip ↕      | ✓ | user ↕   | ✓ | failure_reason ↕                          | src_port ↕ | ✓ |
|---------------------|---------------|---|----------|---|-------------------------------------------|------------|---|
| 2025-08-05 04:07:07 | 86.212.199.60 |   | agushto  |   | Failed password for invalid user agushto  | 3692       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | tomcat   |   | Failed password for invalid user tomcat   | 1464       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | desktop  |   | Failed password for invalid user desktop  | 3518       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | yp       |   | Failed password for invalid user yp       | 2856       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | mail     |   | Failed password for mail                  | 1054       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | apache   |   | Failed password for apache                | 2638       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | services |   | Failed password for invalid user services | 4748       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | irc      |   | Failed password for invalid user irc      | 1203       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | mysql    |   | Failed password for invalid user mysql    | 4802       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | pmuser   |   | Failed password for invalid user pmuser   | 1775       |   |
| 2025-08-05 04:07:07 | 86.212.199.60 |   | ventrilo |   | Failed password for invalid user ventrilo | 1465       |   |

Per verificare se i tentativi sono andati a buon fine, la query è stata modificata in modo da contenere nella regex un campo che differenziasse tra i tentativi riusciti e quelli falliti.

La sintassi è la seguente:

- (?<request\_status>\S+ password): crea il campo **request\_status** e cattura qualsiasi risultato della password.

Per eseguire la ricerca per tutti i tentativi di accesso diversi da un accesso non riuscito, è stata usata la seguente sintassi:

- request\_status!="failed\*": la query cerca nei log qualsiasi evento con **request status** diverso da “failed” e qualsiasi cosa segua.

Il risultato di questa query sarà 0 log trovati, questo perché l'attacco a dizionario proveniente dall'IP in questione non è andato a buon fine.

The screenshot shows a search interface with a dark theme. At the top, a query is entered in a text box: `index=main source="tutorialdata.zip:*" | rex "(?<request_status>\S+ password) for (?<invalid user >)?(?<user>\S+) from (?<src_ip>\d{1,3}(?<.\d{1,3}>){3}) port (?<src_port>\d+)" | search request_status!="failed*" src_ip="86.212.199.60"`. To the right of the query box is a dropdown menu for 'Intervallo temporale: Sempre' and a green search button with a magnifying glass icon. Below the query box, there is a status bar showing '0 eventi (prima di 11/08/25 21:25:04,000)' and a dropdown for 'Nessun campionamento degli eventi'. To the right of this are buttons for 'Processo', a pause icon, a square icon, a refresh icon, a download icon, and a 'Modalità dettagliata' dropdown. Below the status bar is a navigation bar with tabs: 'Eventi (0)', 'Pattern', 'Statistiche', and 'Visualizzazione'. The main content area below the navigation bar is empty and displays the text 'Nessun risultato trovato.'

## **Conclusioni eventi query n. 3**

Dalle informazioni ricavate possiamo concludere che l'IP 86.212.199.60 ha tentato di trovare delle credenziali valide per accedere ai sistemi tramite un attacco a dizionario che però non ha avuto esito positivo.

## **Azioni immediate da intraprendere**

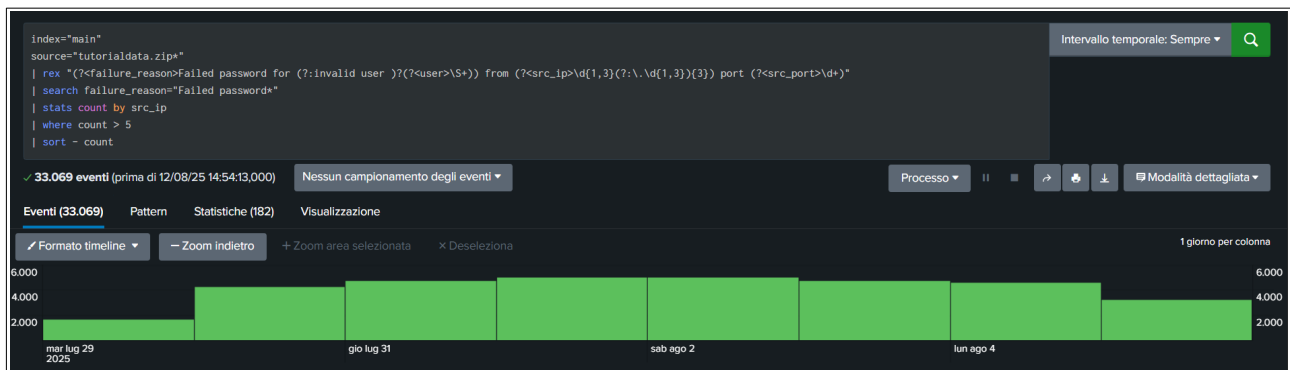
- Bloccare l'IP 86.212.199.60;
- Per gli utenti che sono risultati come validi al momento dell'attacco, effettuare un cambio password preventivo ed applicare MFA.



## Analisi eventi query n. 4

Come per la query n.1, impostando come range temporale “Sempre”, avremo un totale di 33.069 eventi, distribuiti su 8 giorni dal 29 Luglio al 5 Agosto.

La differenza sta nella parte “Statistiche” dove verranno mostrati solamente gli IP che hanno effettuato più di 5 login falliti.



Di seguito vengono riportati gli IP con maggiore attività, prendendo in esame l'intero arco temporale.

Questi saranno gli IP che verranno presi in analisi.

| src_ip          | count |
|-----------------|-------|
| 87.194.216.51   | 948   |
| 211.166.111.101 | 743   |
| 128.241.220.82  | 622   |
| 109.169.32.135  | 515   |
| 194.215.205.19  | 514   |
| 216.221.226.11  | 433   |
| 188.138.40.166  | 297   |

Per avere invece un conteggio giornaliero diviso per IP, possiamo aggiungere alla query la seguente funzione:

- `| eval day=strftime(_time, "%d-%m-%y")`: crea il campo **day** e formatta il campo **\_time** prendendo in considerazione solo il giorno, il mese e l'anno.

Quindi analizzando la tabella di cui sotto, possiamo notare che l'IP che ha generato più rumore è l'87.194.216.51: il giorno 02 Agosto con 233 tentativi di login falliti, il giorno 03 Agosto 156 tentativi, il giorno 30 Luglio 154 tentativi e così via durante tutta la finestra temporale.

Il primo passo sarà quello di analizzare gli IP più rumorosi su fonti OSINT (abuseipdb) per controllarne la reputazione.

Nessuno degli IP presi in esame risulta segnalato.

Ovviamente però il comportamento risultante è sospetto in quanto tutti questi tentativi di login falliti in un solo giorno non possono essere stati eseguiti da un utente reale.

| day      | src_ip         | count |
|----------|----------------|-------|
| 02-08-25 | 87.194.216.51  | 223   |
| 02-08-25 | 211.166.11.101 | 170   |
| 03-08-25 | 87.194.216.51  | 156   |
| 30-07-25 | 87.194.216.51  | 154   |
| 31-07-25 | 87.194.216.51  | 147   |
| 03-08-25 | 211.166.11.101 | 129   |
| 04-08-25 | 109.169.32.135 | 129   |
| 02-08-25 | 128.241.220.82 | 118   |

Sarà quindi nostra cura andare ad analizzare più da vicino il comportamento di questi IP per capire in che modo e se hanno impattato i sistemi.

La query che andremo ad utilizzare ha nel corpo due regex già viste in precedenza, in più è stato aggiunto il campo **protocol** e nel campo **search** abbiamo messo gli IP già individuati come più rumorosi.

```
index="main"
source="tutorialdata.zip*"
| rex "(?<request_status>\S+ password) for (?<invalid user>)?(?<user>\S+) from (?<src_ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<src_port>\d+) (?<protocol>\S+)"
| rex "(?<failure_reason>Failed password for (?<invalid user>)?(?<user>\S+))"
| search src_ip IN ("87.194.216.51", "211.166.11.101", "128.241.220.82", "109.169.32.135", "194.215.205.19", "216.221.226.11", "188.138.40.166")
| table _time src_ip user request_status failure_reason src_port protocol
```

La tabella risultante ci mostrerà i campi **\_time**, **src\_ip**, **user**, **request\_status**, **failure\_reason**, **src\_port** e **protocol**, tutte informazioni utili all'analisi.

Analizzando la tabella possiamo vedere che i tentativi di accesso sono stati fatti sempre in orari non lavorativi standard a distanza di pochi secondi.

Nessuno dei tentativi di accesso da qualsiasi IP preso in esame è andato a buon fine.

Tutti gli IP in questione, stanno probabilmente effettuando degli attacchi brute force a dizionario tramite script automatizzati.

La porta sorgente cambia ad ogni tentativo, ciò significa che ogni volta viene instaurata una nuova sessione.

Il protocollo usato è sempre ssh, quindi le connessioni vengono eseguite probabilmente sulla porta standard 22.

| _time               | src_ip        | user              | request_status  | failure_reason                                     | src_port | protocol |
|---------------------|---------------|-------------------|-----------------|----------------------------------------------------|----------|----------|
| 2025-08-05 04:07:07 | 87.194.216.51 | testing           | Failed password | Failed password for invalid user testing           | 3228     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | mantis            | Failed password | Failed password for invalid user mantis            | 4506     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | elena_andubasquet | Failed password | Failed password for invalid user elena_andubasquet | 3363     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | squid             | Failed password | Failed password for squid                          | 4206     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | mailman           | Failed password | Failed password for invalid user mailman           | 1227     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | mailman           | Failed password | Failed password for invalid user mailman           | 4550     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | info              | Failed password | Failed password for invalid user info              | 1858     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | mailman           | Failed password | Failed password for invalid user mailman           | 3366     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | apache            | Failed password | Failed password for apache                         | 3026     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | nginx             | Failed password | Failed password for invalid user nginx             | 1455     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | inet              | Failed password | Failed password for invalid user inet              | 3329     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | testing           | Failed password | Failed password for invalid user testing           | 4810     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | desktop           | Failed password | Failed password for invalid user desktop           | 2903     | ssh2     |
| 2025-08-05 04:07:07 | 87.194.216.51 | grumpy            | Failed password | Failed password for grumpy                         | 1982     | ssh2     |

## Conclusioni eventi query n. 4

Come già visto in precedenza, anche in questo caso si tratta probabilmente di attacchi brute force a dizionario da botnet.

Gli attacchi sono automatizzati tramite script schedulato per attivarsi tutti i giorni intorno alle 04:00 e testano username e password comuni sul sistema target.

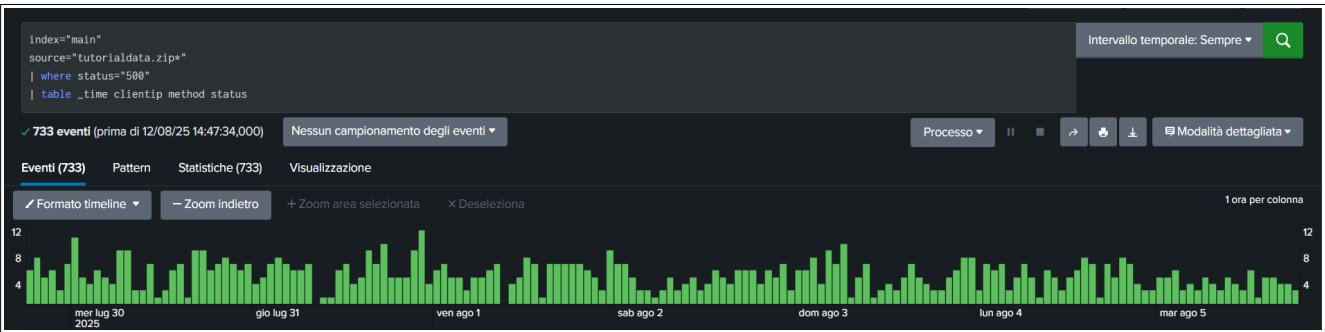
Le connessioni sono sempre tramite ssh e quindi probabilmente viene usata una porta standard per il servizio e cioè la 22.

## Azioni immediate da intraprendere

- Bloccare gli IP più rumorosi;
- Limitare l'accesso in ssh ai soli utenti autorizzati ed assicurarsi che questi utenti abbiano una password sicura, MFA ed un username non comune;
- Per gli utenti che sono risultati come validi al momento dell'attacco, effettuare un cambio password preventivo ed applicare MFA.

# Analisi eventi query n. 5

Mettendo come intervallo di tempo “Sempre”, i log totali risultanti dalla query sono 733 in un intervallo di tempo che va dal 29 Luglio al 05 Agosto.



Possiamo vedere che i tentativi sono stati eseguiti interamente sul server ecommerce appartenente al dominio <http://www.buttercupgames.com>.

| referer_domain                                                                 |                        |             |
|--------------------------------------------------------------------------------|------------------------|-------------|
| 1 Valore, 100% di eventi                                                       |                        |             |
| Selezionato <input checked="" type="checkbox"/> Sì <input type="checkbox"/> No |                        |             |
| Report                                                                         |                        |             |
| Primi valori                                                                   | Primi valori nel tempo | Valori rari |
| Eventi con questo campo                                                        |                        |             |
| Valori                                                                         | Conteggio              | %           |
| <a href="http://www.buttercupgames.com">http://www.buttercupgames.com</a>      | 733                    | 100%        |

Di seguito i primi 10 IP coinvolti.  
Come primo IP notiamo l'87.194.216.51, che abbiamo già incontrato nell'analisi precedente.

| clientip                                                                       |                        |             |
|--------------------------------------------------------------------------------|------------------------|-------------|
| >100 Valori, 100% di eventi                                                    |                        |             |
| Selezionato <input checked="" type="checkbox"/> Sì <input type="checkbox"/> No |                        |             |
| Report                                                                         |                        |             |
| Primi valori                                                                   | Primi valori nel tempo | Valori rari |
| Eventi con questo campo                                                        |                        |             |
| Primi 10 valori                                                                | Conteggio              | %           |
| <a href="#">87.194.216.51</a>                                                  | 19                     | 2,592%      |
| <a href="#">125.89.78.6</a>                                                    | 15                     | 2,046%      |
| <a href="#">211.166.11.101</a>                                                 | 13                     | 1,774%      |
| <a href="#">109.169.32.135</a>                                                 | 11                     | 1,501%      |
| <a href="#">128.241.220.82</a>                                                 | 11                     | 1,501%      |
| <a href="#">188.138.40.166</a>                                                 | 10                     | 1,364%      |
| <a href="#">69.175.97.11</a>                                                   | 10                     | 1,364%      |
| <a href="#">201.28.109.162</a>                                                 | 9                      | 1,228%      |
| <a href="#">107.3.146.207</a>                                                  | 8                      | 1,091%      |
| <a href="#">125.7.55.180</a>                                                   | 8                      | 1,091%      |

Visto che sappiamo che l'IP in questione ha già compiuto azioni sospette, andiamo ad analizzarlo più nello specifico.  
Applichiamo alla query dei filtri per il domain, l'IP e lo status.

```
index="main" source="tutorialdata.zip*"
| search referer_domain="http://www.buttercupgames.com" clientip="87.194.216.51" status=500
```

Il risultato sono 19 log in totale, che evidenziano un comportamento sospetto:

- **JSESSIONID**: Il valore cambia quasi a ogni richiesta, segno che ogni sessione è nuova o che il client non mantiene una sessione persistente (o forza sessioni multiple);
- **method**: sia GET che POST;
- **referer**: spesso da pagine interne (oldlink, category.screen) con parametri a volte sospetti (ad esempio categoryId=NULL);
- **useragent**: vari e cambiati a ogni richiesta, questo cambio continuo è probabilmente attribuibile a script automatizzati che simulano browser diversi.

```
> 05/08/25 10:50:07,000 87.194.216.51 - - [05/Aug/2025:10:50:07] "GET /oldlink?itemId=EST-14&JSESSIONID=SD0SL7FF5ADFF50854 HTTP 1.1" 500 220 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.55.3 (KHTML, like Gecko) Version/5.1.5 Safari/534.55.3" 576
JSESSIONID = SD0SL7FF5ADFF50854 clientip = 87.194.216.51 host = Navi method = GET referer = http://www.buttercupgames.com/category.screen?categoryId=NULL referer_domain = http://www.buttercupgames.com source = tutorialdata.zip:/www2/access.log sourcetype = access_combined_wcookie status = 500 uri_path = /oldlink user = - useragent = Mozilla/5.0 (Macintosh; Intel Mac OS X 10_6_8) AppleWebKit/534.55.3 (KHTML, l...

> 05/08/25 06:49:30,000 87.194.216.51 - - [05/Aug/2025:06:49:30] "POST /product.screen?productId=SF-BVS-G01&JSESSIONID=SD5SL8FF9ADFF50030 HTTP 1.1" 500 3286 "http://www.buttercupgames.com/oldlink?itemId=EST-7" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.28) Gecko/20120306 YFF3 Firefox/3.6.28 (.NET CLR 3.5.30729; .NET4.0C)" 184
JSESSIONID = SD5SL8FF9ADFF50030 clientip = 87.194.216.51 host = Navi method = POST referer = http://www.buttercupgames.com/oldlink?itemId=EST-7 referer_domain = http://www.buttercupgames.com source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie status = 500 uri_path = /product.screen user = - useragent = Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.28) Gecko/20120306 Y...

> 05/08/25 02:34:29,000 87.194.216.51 - - [05/Aug/2025:02:34:29] "GET /oldlink?itemId=EST-11&JSESSIONID=SD8SL10FF6ADFF48710 HTTP 1.1" 500 1414 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)" 758
JSESSIONID = SD8SL10FF6ADFF48710 clientip = 87.194.216.51 host = Navi method = GET referer = http://www.buttercupgames.com/category.screen?categoryId=NULL referer_domain = http://www.buttercupgames.com source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie status = 500 uri_path = /oldlink user = - useragent = Mozilla/5.0 (compatible; YandexBot/3.0; +http://yandex.com/bots)

> 04/08/25 17:52:06,000 87.194.216.51 - - [04/Aug/2025:17:52:06] "GET /cart.do?action=addtocart&itemId=EST-19&JSESSIONID=SD2SL3FF1ADFF46010 HTTP 1.1" 500 3300 "http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-19" "Opera/9.20 (Windows NT 6.0; U; en)" 601
JSESSIONID = SD2SL3FF1ADFF46010 clientip = 87.194.216.51 host = Navi method = GET referer = http://www.buttercupgames.com/cart.do?action=addtocart&itemId=EST-19 referer_domain = http://www.buttercupgames.com source = tutorialdata.zip:/www1/access.log sourcetype = access_combined_wcookie status = 500 uri_path = /cart.do user = - useragent = Opera/9.20 (Windows NT 6.0; U; en)
```

Togliendo il filtro dell'IP ed aggiungendo quello per il **referer** "categoryId=NULL", possiamo vedere che svariati IP hanno tentato di effettuare questo tipo di richiesta.

```
index="main" source="tutorialdata.zip*"
| search referer_domain="http://www.buttercupgames.com" status=500 referer="http://www.buttercupgames.com/category.screen?categoryId=NULL"
| stats count by clientip
```

Il risultato sono 186 eventi eseguiti da 116 IP diversi, con l'IP 87.194.216.51 che ha effettuato la stessa richiesta cinque volte.

Questo tipo di eventi è attribuibile ad un tentativo di iniezione SQL, che ha generato un errore interno del server che non è riuscito ad elaborare la richiesta.

Eventi (186)

Pattern

Statistiche (116)

Visualizzazione

Mostra: 100 per pagina

Formato

Anteprima: on

< Prec

1

2

Avanti >

| clientip       | count |
|----------------|-------|
| 87.194.216.51  | 5     |
| 107.3.146.207  | 4     |
| 128.241.220.82 | 4     |
| 175.44.24.82   | 4     |
| 198.35.2.120   | 4     |
| 211.166.11.101 | 4     |
| 109.169.32.135 | 3     |

## Conclusioni eventi query n. 5

I log analizzati mostrano uno scenario in cui alcuni IP hanno tentato di effettuare delle iniezioni SQL sul server di e-commerce del dominio “buttercupgames.com”

Le richieste hanno generato degli errori interni del server (codice 500) che non ha potuto elaborare correttamente la richiesta.

Lo scopo dei potenziali attaccanti potrebbe anche essere quello di generare volutamente questo tipo di errore nel server per provocare disservizio.

## Azioni immediate da intraprendere

- Bloccare tutti gli IP che hanno generato la richiesta **referer** “Id=NULL”;
- Assicurarsi che gli input lato utente siano correttamente sanitizzati;
- Applicare una regola di rate limiting sul firewall in modo da rallentare gli attaccanti;
- Bloccare gli user-agent riconducibili a bot malevoli.