

W24D4 – Esame del modulo 6 – Splunk

Premessa: in questo esercizio ho cercato di dimostrare la varietà e la creatività con le quali si possono realizzare delle query efficaci per analizzare i dati con Splunk, usando il search processing language. Ho scritto query semplici ed ho anche utilizzato l'intelligenza artificiale per generare delle query più complesse, dato che in questo modulo si parlava proprio dell'impatto dell'IA nel mondo della cybersecurity. Per questo, per ogni query richiesta nella consegna ci saranno varie soluzioni, alcune più semplici ed altre più elaborate, specificando quali sono le più precise e quali, invece, richiedono una maggiore attenzione per evitare falsi positivi. Ovviamente tutti i dati raccolti con le varie query possono concorrere ad una migliore comprensione di cosa avviene in un sistema.

Query numero 1

Proposta 1, più semplice:

Con "search" ho cercato tra i log di sicurezza i tentativi di accesso falliti a causa di una password non corretta, selezionando i campi per vedere il timestamp e richiedendo di mostrare lo user oltre all'indirizzo IP sorgente. La query usata è la seguente
Source="tutorialdata.zip:/*" sourcetype="secure.log" | search failed password | search user

The screenshot shows the Splunk web interface with the following details:

- URL:** 127.0.0.1
- Ricerca | Splunk 9.3.1**
- Query Bar:** source="tutorialdata.zip:/*" sourcetype="secure.log" | search failed password | search user
- Results Summary:** 18.406 eventi (prima di 01/11/24 10:44:49,000) Nessun campionamento degli eventi
- Event View:** Eventi (18.406) Pattern | Statistiche | Visualizzazione
- Timeline:** Formato timeline | Zoom indietro | Zoom area selezionata | Deseleziona | 1 ora per colonna
- Event List:** Shows a list of 18,406 events from 30/10/24 at 04:36:08.000. Each event entry includes the timestamp, user, host, and source information. For example:
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[5276]: Failed password for invalid user appserver from 194.8.74.23 port 3351 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[5258]: Failed password for invalid user testuser from 194.8.74.23 port 3626 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[3760]: Failed password for invalid user mongo from 194.8.74.23 port 2472 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[5981]: Failed password for invalid user desktop from 194.8.74.23 port 2285 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[5979]: Failed password for invalid user cyrus from 194.8.74.23 port 3417 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[2695]: Failed password for invalid user itadmin from 194.8.74.23 port 4692 ssh2
 - Thu Oct 30 2024 04:36:08 mailsv1 sshd[5331]: Failed password for invalid user snet from 194.8.74.23 port 4564 ssh2
- Bottom Buttons:** < Prec 1 2 3 4 5 6 7 8 Avanti >

Gli eventi trovati sono moltissimi, ne mostro alcuni.

Prendendo un singolo evento, si vedono tutti i dati richiesti, ovvero il timestamp, il nome utente, l'IP di origine e il motivo del fallimento del login, come richiesto dall'esercizio.

| 30/10/24 | Thu Oct 30 2024 04:36:08 mailsv1 sshd[5085]: Failed password for invalid user harrison from 194.8.74.23 port 3756 ssh2 | | |
|-----------------|--|--------------------------------------|--------|
| 04:36:08,000 | | | |
| Azioni evento ▾ | | | |
| Tipo | Campo | Valore | Azioni |
| Selezionato | <input checked="" type="checkbox"/> date_hour | 4 | ▼ |
| | <input checked="" type="checkbox"/> date_mday | 30 | ▼ |
| | <input checked="" type="checkbox"/> date_minute | 36 | ▼ |
| | <input checked="" type="checkbox"/> date_month | october | ▼ |
| | <input checked="" type="checkbox"/> date_second | 8 | ▼ |
| | <input checked="" type="checkbox"/> date_wday | wednesday | ▼ |
| | <input checked="" type="checkbox"/> date_year | 2024 | ▼ |
| | <input checked="" type="checkbox"/> date_zone | local | ▼ |
| | <input checked="" type="checkbox"/> host | 127.0.0.1 | ▼ |
| | <input checked="" type="checkbox"/> source | tutorialdata.zip:\mailsv1\secure.log | ▼ |
| | <input checked="" type="checkbox"/> sourcetype | secure.log | ▼ |
| Ora | <input checked="" type="checkbox"/> _time | 2024-10-30T04:36:08.000+01:00 | |
| Default | <input type="checkbox"/> index | main | ▼ |
| | <input type="checkbox"/> linecount | 1 | ▼ |
| | <input type="checkbox"/> punct | — - - - - | ▼ |
| | <input type="checkbox"/> splunk_server | DESKTOP-8HQQS01 | ▼ |

Splunk è un'ottima scelta per un SIEM perché consente di individuare dei pattern automaticamente, i quali vengono poi esaminati dagli analisti che decidono come intervenire in base alla situazione.

splunk>enterprise App ▾

Ricerca Analytics Set di dati Report Allarmi Dashboard

Nuova ricerca

```
source="tutorialdata.zip:*" sourcetype="secure.log" | search failed password | search user
```

✓ 18.406 eventi (prima di 01/11/24 14:29:52,000) Nessun campionamento degli eventi ▾

Eventi (18.406) Pattern Statistiche Visualizzazione

Più piccolo ————— Più grande

13 pattern basato su un esempio di 8.895 eventi

11,03% Tue <timestamp>www2 sshd[4356]: Failed password for invalid user postgres from 109.169.32.135 port 1129 ssh2
10,85% Thu <timestamp>www1 sshd[1089]: Failed password for invalid user local from 2.229.4.58 port 4910 ssh2
10,47% Sun <timestamp>www2 sshd[1408]: Failed password for invalid user operator from 178.19.3.199 port 3531 ssh2
9,66% Mon <timestamp>www2 sshd[1364]: Failed password for invalid user george from 223.5.16.102 port 3434 ssh2
9,41% Thu <timestamp>www2 sshd[1210]: Failed password for invalid user inet from 64.66.0.20 port 4086 ssh2
9,29% Thu <timestamp>www3 sshd[1370]: Failed password for invalid user administrator from 207.36.232.245 port 4438 ssh2
8,51% Wed <timestamp>www2 sshd[4068]: Failed password for invalid user vmware from 67.133.102.54 port 3575 ssh2
8,43% Sat <timestamp>www2 sshd[2309]: Failed password for invalid user appserver from 110.138.30.229 port 4431 ssh2
7,72% Fri <timestamp>www2 sshd[4855]: Failed password for invalid user administrator from 87.194.216.51 port 1569 ssh2
3,52% Fri <timestamp>www3 sshd[3352]: Failed password for invalid user mailman from 59.162.167.100 port 3607 ssh2

3% Thu <timestamp>mailsv1 sshd[5276]: Failed password for invalid user appserver from 194.8.74.23 port 3351 ssh2
2,81% Sat <timestamp>www3 sshd[2284]: Failed password for invalid user henri from 125.7.55.180 port 4192 ssh2
2,73% Wed <timestamp>www3 sshd[4351]: Failed password for invalid user email from 216.221.226.11 port 3047 ssh2

← C ⓘ 127.0.0.1:8000/it-IT/app/search/search?q=search%20source%3D"tutorialdata.zip%3A*%20sourcetype%3D"secure.log"

splunk>enterprise App ▾

Ricerca Analytics Set di dati Report Allarmi Dashboard

Nuova ricerca

```
source="tutorialdata.zip:*" sourcetype="secure.log" | search failed password | search user
```

✓ 18.406 eventi (prima di 01/11/24 14:29:52,000) Nessun campionamento degli eventi ▾

Eventi (18.406) Pattern Statistiche Visualizzazione

Più piccolo ————— Più grande

1 pattern basato su un esempio di 8.895 eventi

100% Thu <timestamp>mailsv1 sshd[5276]: Failed password for invalid user appserver from 194.8.74.23 port 3351 ssh2

Si ottengono risultati interessanti anche cambiando sourcetype. Per esempio, nella query source="tutorialdata.zip:/*" sourcetype="snort" | search failed password | search user ho scelto come sourcetype snort, un IDS/IPS che avevo trovato tra i sourcetype per la sicurezza e le reti nella configurazione iniziale per lanciare l'analisi dei dati di tutorialdata.

Query numero 2

Proposta numero 1 più precisa:

Una query precisa ed elaborata che si può utilizzare è la seguente:
Source="tutorialdata.zip:/*" sourcetype="secure.log" | rex "Accepted password for (?<user_id>\S+)" | where user_id="djohnson" | table _time user_id | rename _time as "Timestamp", user_id as "ID Utente" | sort - _time. In questo caso si usa il regex pattern per catturare, tramite una regular expression, lo username che riesce ad effettuare il login e in questo caso si specifica che l'username in questione è "djohnson". Qui ho visualizzato gli eventi e non la tabella. L'ID utente è stato selezionato anche tra i campi da mostrare, oltre al timestamp.

The screenshot shows the Splunk 9.3.1 interface with the following details:

- Search bar:** 127.0.0.1 > Niente | Splunk 9.3.1
- Search Query:** source="tutorialdata.zip:/*" sourcetype="secure.log" | rex "Accepted password for (?<user_id>\S+)" | where user_id="djohnson" | table _time user_id | rename _time as "Timestamp", user_id as "ID Utente" | sort - _time
- Results:** 725 eventi (prima di 01/11/24 12:20:40,000) Nessun campionamento degli eventi ▾
- Event View:** Eventi (725) Pattern Statistiche (725) Visualizzazione
- Timeline:** Formato timeline ▾ – Zoom indietro + Zoom area selezionata × Deselezioni
- Table View:** Elenco ▾ Formato 20 per pagina ▾
- Table Headers:** Nascondi campi, i, Ora, Evento
- Table Data:** The table lists 725 events, each containing the timestamp (e.g., 30/10/24 04:36:08.000), the event (e.g., Thu Oct 30 2024 04:36:08 mailsv1 sshd[54545]), and the user_id (e.g., djohnson). The events are sorted by timestamp in descending order.

Proposta numero 2 completa e con dati in più:

La query che ho eseguito è la seguente: source="tutorialdata.zip:/*" host="127.0.0.1" sourcetype="secure.log" "Accepted password" "djohnson" | rex "sshd\[(<session_id>\d+)\]: Accepted password for (<user>\w+)" | where user="djohnson" | eval status="Success" | table _time user session_id status. Si vede l'utilizzo della funzione rex, che serve per estrarre dei campi da un evento grezzo sulla base di un'espressione regolare. Session ID ed user vengono quindi estratti quando viene rispettata la condizione indicata. Con eval viene aggiunto un nuovo campo per valutare se il login è andato a buon fine per questo utente e con table i dati vengono messi in una tabella con i campi richiesti dall'esercizio ben specificati nelle singole colonne. Si vede anche l'ID della sessione aperta, che va ad integrare quanto visto prima cercando lo User ID.

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** source="tutorialdata.zip:*" host="127.0.0.1" sourcetype="secure.log" "Accepted password" "djohnson" | rex "sshd\[(<session_id\>|)\]: Accepted password for (?>user\|w+)*| where user="djohnson" | eval status="Success"|" table _time user session_id status
- Results Summary:** 725 eventi (prima di 03/11/24 22:42:30,000) Nessun campionamento degli eventi *
- Event View:** Eventi (725) Pattern Statistiche (725) Visualizzazione
- Table Headers:** _time #, user #, session_id #, status #
- Table Data:** A list of log entries showing timestamp, user, session ID, and status for user djohnson.

Proposta numero 3 più semplice selezionando i campi:

La seguente query è invece molto più semplice e mostra I numerosi eventi rispondenti ai criteri di ricerca indicati. I campi selezionati mostrano il timestamp e l'ID utente. La query usata è la seguente: Source="tutorialdata.zip:/*" sourcetype="secure.log" | search sshd | search session opened for user djohnson

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** source="tutorialdata.zip:/*" sourcetype="secure.log" | search sshd | search session opened for user djohnson
- Results Summary:** 818 eventi (prima di 03/11/24 11:05,000) Nessun campionamento degli eventi *
- Event View:** Eventi (818) Pattern Statistiche Visualizzazione
- Timeline View:** Formato timeline • Zoom indietro + Zoom area selezionata X Deselezionata
- Table Headers:** _time #, Evento
- Table Data:** A list of log entries showing timestamp and event details for user djohnson.

Anche qui è possibile andare a controllare il singolo evento, vedendo tutti i campi richiesti, come il timestamp e l'ID utente.

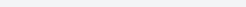
| 30/10/24 | Thu Oct 30 2024 04:36:08 msv1 | sshd[54980]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) | |
|------------------------|---|--|--------|
| 04:36:08.000 | | | |
| Azioni evento ▾ | | | |
| Tipo | Campo | Valore | Azioni |
| Selezionato | <input checked="" type="checkbox"/> date_hour | 4 | ▼ |
| | <input checked="" type="checkbox"/> date_mday | 30 | ▼ |
| | <input checked="" type="checkbox"/> date_minute | 36 | ▼ |
| | <input checked="" type="checkbox"/> date_month | october | ▼ |
| | <input checked="" type="checkbox"/> date_second | 8 | ▼ |
| | <input checked="" type="checkbox"/> date_wday | wednesday | ▼ |
| | <input checked="" type="checkbox"/> date_year | 2024 | ▼ |
| | <input checked="" type="checkbox"/> date_zone | local | ▼ |
| | <input checked="" type="checkbox"/> host | 127.0.0.1 | ▼ |
| | <input checked="" type="checkbox"/> source | tutorialdata.zip: mailsv1 secure.log | ▼ |
| | <input checked="" type="checkbox"/> sourcetype | secure.log | ▼ |
| | <input checked="" type="checkbox"/> uid | 0 | ▼ |
| Ora | <input checked="" type="checkbox"/> _time | 2024-10-30T04:36:08.000+01:00 | |
| Default | <input type="checkbox"/> index | main | ▼ |
| | <input type="checkbox"/> linecount | 1 | ▼ |
| | <input type="checkbox"/> punct | -_-[]{}_____=+= | ▼ |
| | <input type="checkbox"/> splunk_server | DESKTOP-8HIGQSD1 | ▼ |

Come sempre, è interessante controllare i pattern identificati da Splunk, che possono essere utili per il lavoro degli analisti.

source="tutorialdata.zip:*" sourcetype="secure.log" | search sshd | search session opened for user djohnson

✓ 818 eventi (prima di 01/11/24 17:03:40,000) Nessun campionamento degli eventi ▾

Eventi (818) Pattern Statistiche Visualizzazione

Più piccolo  Più grande

1 pattern basato su un esempio di 818 eventi

⚠ Un numero di eventi inferiore a 5.000 potrebbe generare pattern di bassa qualità. Provare a effettuare una ricerca in un intervallo temporale più ampio o con meno vincoli.

100% Thu <timestamp>mailru1 sshd[60445]: pam_unix(sshd:session): session opened for user djohnson by (uid=0)

Come visto per la query precedente, anche in questo caso si trovano informazioni interessanti cambiando sourcetype. Qui ho usato il sourcetype "ww1/secure" e la query è source="tutorialdata.zip:*" sourcetype="ww1/secure" | search sshd | search session opened for user djohnson.

Anche il sourcetype snort, visto precedentemente, consente di trovare maggiori informazioni. La query è identica a quella di prima, solo che cambia il sourcetype.

Nuova ricerca

source="tutorialdata.zip:/* sourcetype="snort" | search sshd | search session opened for user djohnson|

✓ 158 eventi (prima di 01/11/24 17:12:51,000) Nessun campionamento degli eventi ▾

Eventi (158) Pattern Statistiche Visualizzazione

Formato timeline ▾ — Zoom indietro + Zoom area selezionata × Deselezione 1 giorno per colonna

Elenco ▾ Formato 20 per pagina ▾

| Campi SELEZIONATI | Ora | Evento |
|--|-----------------------|---|
| # date_hour # date_minute # date_month # date_second # date_wday # date_year # date_zone # host # source # sourcetype | 30/10/24 04:36:08.000 | ... 10 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[5880]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) ... 33 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[87066]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) Thu Oct 30 2024 04:36:08 mailsv1 sshd[4907]: Failed password for invalid user irc from 194.8.74.23 port 1956 ssh2 ... 2 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[2005]: Failed password for invalid user itadmin from 194.8.74.23 port 4692 ssh2 Thu Oct 30 2024 04:36:08 mailsv1 sshd[68445]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) Mostra tutte le 63 righe date_hour = 4 date_minute = 30 date_month = october date_second = 8 date_wday = wednesday date_year = 2024 date_zone = local host = 127.0.0.1 source = tutorialdata.zip:mailsv1/secure.log sourcetype = snort |
| CAMPi INTERESSANTI | 30/10/24 04:36:08.000 | ... 4 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[53381]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) ... 39 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[97536]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) ... 64 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[68740]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) ... 62 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[54980]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) ... 77 lines omitted ... Thu Oct 30 2024 04:36:08 mailsv1 sshd[71798]: pam_unix(sshd:session): session opened for user djohnson by (uid=0) Mostra tutte le 257 righe date_hour = 4 date_minute = 30 date_month = october date_second = 8 date_wday = wednesday date_year = 2024 date_zone = local host = 127.0.0.1 source = tutorialdata.zip:mailsv1/secure.log sourcetype = snort |

+ Estrai nuovi campi

Query numero 3

Proposta numero 1 più completa e precisa:

Questa query consente di vedere gli accessi falliti da parte dell'IP 86.212.199.60, mostrando il timestamp, il nome utente e la porta. La seguente query riporta tutti i dati richiesti in formato di tabella, evidenziando per ogni colonna i dati cercati. La query è la seguente:

```
source="tutorialdata.zip:/*" host="127.0.0.1" sourcetype="secure.log" ("failed password" OR "Accepted password") | rex "password for (invalid user )?(?<user>\w+)" from (?<src_ip>86\.212\.199\.60) port (?<port>\d+) | eval status;if(searchmatch("failed password"), "Failed", "Success")| where src_ip="86.212.199.60" | table _time src_ip user port status
```

Viene utilizzato l'operatore booleano OR.

Ricerca | Splunk 9.3.1

127.0.0.1:8000/r/IT/app/search/search?q=search%20source%3D"tutorialdata.zip%3A%"%20host%3D"127.0.0.1"%20sourcetype%3D"secure.log"%20"failed%20password"%20OR%20"Accepted%20password"%20%"Failed", "Success")| where src_ip="86.212.199.60" | table _time src_ip user port status

✓ 54 eventi (prima di 03/11/24 22:47:47,000) Nessun campionamento degli eventi ▾

Eventi (54) Pattern Statistiche (54) Visualizzazione

20 per pagina ▾ Formato Anteprima ▾

| _time | src_ip | user | port | status |
|---------------------|---------------|----------|------|--------|
| 2024-10-29 04:36:05 | 86.212.199.60 | db | 2690 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | services | 1393 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | sync | 1695 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | admin | 3673 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | nginx | 1582 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | whos | 1635 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | mailman | 4139 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | mailman | 1954 | Failed |
| 2024-10-29 04:36:06 | 86.212.199.60 | rdb | 2650 | Failed |
| 2024-10-28 04:36:07 | 86.212.199.60 | whois | 4566 | Failed |
| 2024-10-28 04:36:07 | 86.212.199.60 | ftp | 2043 | Failed |
| 2024-10-28 04:36:07 | 86.212.199.60 | hsqldb | 2190 | Failed |
| 2024-10-28 04:36:07 | 86.212.199.60 | hammer | 1323 | Failed |
| 2024-10-28 04:36:06 | 86.212.199.60 | ncsd | 4022 | Failed |
| 2024-10-28 04:36:06 | 86.212.199.60 | games | 1763 | Failed |
| 2024-10-28 04:36:06 | 86.212.199.60 | noone | 1583 | Failed |
| 2024-10-28 04:36:06 | 86.212.199.60 | fpass | 3420 | Failed |
| 2024-10-28 04:36:06 | 86.212.199.60 | divine | 1059 | Failed |

Proposta numero 2 più semplice:

La query seguente è invece più semplice, perché ricerca l'IP specifico, lo user, la porta e i tentativi falliti, sfruttando il pipe per far in modo che l'output del comando precedente diventi l'input del secondo. Il timestamp è reso selezionando tutti i campi relativi al momento in cui si è verificato il dato evento. La query è la seguente: Source="tutorialdata.zip:/*" sourcetype="secure.log" | search 86.212.199.60 | search failed | search user | search port

The screenshot shows the Splunk interface with the following search command in the search bar:

```
source="tutorialdata.zip:/*" sourcetype="secure.log" | search 86.212.199.60 | search failed | search user | search port
```

The results table displays 35 events, all occurring on Oct 28, 2024, at various times between 04:36:00 and 04:46:00. Each event is a failed password attempt from user 'www1' or 'www2' on port 2999 or 4339. The details include the date, hour, minute, month, second, day of week (wednesday/tuesday), year (2024), zone (local), host (127.0.0.1), source (tutorialdata.zip:www1/secure.log or tutorialdata.zip:www2/secure.log), and sourcetype (secure.log). The user column shows 'www1' or 'www2' and the port column shows 2999 or 4339.

Questi sono molti dei numerosi eventi registrati da Splunk. Nella seconda parte della consegna verranno analizzati tutti questi eventi dal punto di vista della sicurezza, per ora mi limito solo a riportare quanto trovato dalle ricerche con Splunk.

The screenshot shows the Splunk interface with the same search command as the previous one:

```
source="tutorialdata.zip:/*" sourcetype="secure.log" | search 86.212.199.60 | search failed | search user | search port
```

The results table displays 35 events, all occurring on Oct 28, 2024, at various times between 04:36:00 and 04:46:00. The details are identical to the first screenshot, showing failed password attempts from user 'www1' or 'www2' on port 2999 or 4339. The user column shows 'www1' or 'www2' and the port column shows 2999 or 4339.

Questo è uno degli eventi trovati, analizzato nel dettaglio. Si vede l'IP ricercato, la porta, l'utente coinvolto e tutti i dettagli temporali dell'evento.

The screenshot shows the Splunk search configuration interface. At the top, it displays the search query: `source=tutorialdata.zip;www3|secure.log`. Below this, there are sections for "Azioni evento" (Event Actions), "Tipo" (Type) set to "Campo" (Field), "Valore" (Value) set to "date_hour" with value "4", and "Ora" (Time) set to "2024-10-29T04:36:06.000+0100". There are also sections for "Default" (Default) settings like "index", "main", "linecount", "punct", and "splunk_server".

Si trovano risultati interessanti anche cambiando sourcetype, scegliendo "ww1/secure" invece di "secure.log". La query è come quella precedente.

The screenshot shows the Splunk search results for the query `source=tutorialdata.zip;ww1|secure`. The results list 475 events from October 30, 2024, at 04:36:08. The events are categorized under "Eventi (475)" and include fields like "Ora" (Time), "Evento" (Event), and "Campos" (Fields). The results show multiple failed password attempts from various users (e.g., agusuto, services, irc) on different ports (e.g., 2565, 4740, 1203, 3692) and hosts (e.g., 86.212.199.60).

Anche scegliendo "snort" come sourcetype si trovano risultati interessanti e rilevanti per gli analisti di sicurezza.

The screenshot shows the Splunk search results for the query `source=tutorialdata.zip;snort`. The results list 17 events from October 30, 2024, at 04:36:08. The events are categorized under "Eventi (17)" and include fields like "Ora" (Time), "Evento" (Event), and "Campos" (Fields). The results show multiple failed password attempts from various users (e.g., system, ventrilo, pmuser, irc, desktop) on different ports (e.g., 3385, 1465, 1203, 3518, 2690) and hosts (e.g., 86.212.199.60).

Proposta numero 3 più complessa:

Si potrebbe ricorrere anche alla seguente query per filtrare le ricerche tramite una regular expression e visualizzare i dati richiesti rinominando le colonne della tabella. La query è

```
source="tutorialdata.zip:*" sourcetype="secure.log" | search "86.212.199.60" | search "failed password" | rex "Failed password for (?:invalid user )?(?:<username>\S+) from (?<src_ip>\S+) port (?<port>\d+)" | table _time username port | rename _time as "Timestamp", username as "nome utente", port as "porta" | sort time. Qui ho visualizzato gli eventi e non la tabella.
```

The screenshot shows the Splunk interface with a search bar containing the query. Below the search bar, it says "54 eventi (prima di 02/11/24 21:25:08.000) Nessun campionamento degli eventi". The results are listed under the "Eventi (54)" tab. Each event row contains the timestamp, event ID, and a detailed log entry. The log entries show multiple failed password attempts from the IP 86.212.199.60 on various ports (2690, 1393, 1695, 3673, 1582, 1635, 4339) on different dates (Oct 28, Oct 29, Oct 30, Nov 1). The logs include fields like date_hour, date_minute, date_month, date_second, date_wday, date_year, date_zone, host, source, and src_ip.

Come visto prima, è sempre interessante esaminare i pattern che individua Splunk dalle query.

The screenshot shows the Splunk interface with a search bar containing the query. Below the search bar, it says "54 eventi (prima di 02/11/24 21:25:08.000) Nessun campionamento degli eventi". The results are listed under the "Eventi (54)" tab. A "Pattern" tab is selected, which includes a slider for "Piu piccolo" and "Piu grande". A note below the slider states: "1 pattern basato su un esempio di 54 eventi" and "Un numero di eventi inferiore a 5.000 potrebbe generare pattern di bassa qualità. Provare a effettuare una ricerca in un intervallo temporale più ampio o con meno vincoli." On the right side, there is a "Documentazione" section with a link "Fare clic su un pattern per maggiori informazioni."

Query numero quattro.

Proposta numero 1 completa:

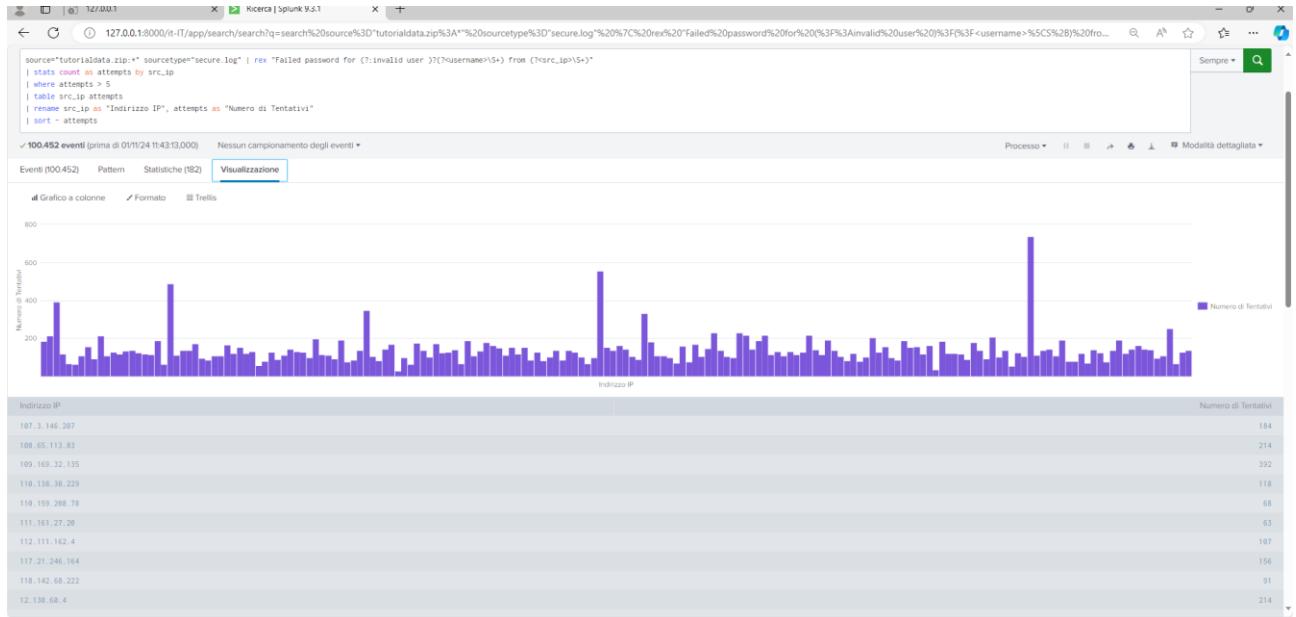
Questa query serve per trovare gli IP che hanno tentato di accedere al sistema per più di cinque volte, fallendo. Con la regular expression di rex vengono estratti I campi dello username e dell'IP sorgente, passando poi per il comando stats count che aggrega i dati contando il numero di tentativi falliti per IP alla condizione che superino i 5 tentativi falliti. I risultati vengono mostrati in una tabella, rinominando le colonne ed ordinando i tentativi di

accesso falliti. Ecco la query usata: `Source="tutorialdata.zip:/*" sourcetype="secure.log" | rex "Failed password for (?<invalid user >)?(?<username>\S+) from (?<src_ip>\S+)" | stats count as attempts by src_ip | where attempts > 5 | table src_ip attempts | rename src_ip as "Indirizzo IP", attempts as "Numero di Tentativi" | sort - attempts`

The screenshot shows a Splunk search interface with the following details:

- Search Query:** `source="tutorialdata.zip:/*" sourcetype="secure.log" | rex "Failed password for (?<invalid user >)?(?<username>\S+) from (?<src_ip>\S+)" | stats count as attempts by src_ip | where attempts > 5 | table src_ip attempts | rename src_ip as "Indirizzo IP", attempts as "Numero di Tentativi" | sort - attempts`
- Results:** 100.452 eventi (prima di 0/1/24 11:43:13,000) Nessun campionamento degli eventi.
- Panel:** Eventi (100.452) Pattern Statistiche (182) Visualizzazione
- Table Headers:** Indirizzo IP, Numero di Tentativi
- Data:** A list of IP addresses and their attempt counts, sorted by attempts in descending order. The top entries include 2.229.4.58 (141), 12.130.68.4 (214), 12.130.69.5 (109), 24.185.15.226 (100), and 27.1.11.11 (230).

Splunk offre anche strumenti di visualizzazione interessanti, mostrando i vari IP e i loro tentativi di accesso, rendendo più facile individuare visivamente gli IP più aggressivi che vanno magari bloccati.



Proposta numero 2 più precisa:

Anche la seguente query è molto efficace nel trovare gli IP che hanno tentato e fallito il login per più di cinque volte. Sempre con rex una regular expression estrae i tentativi di login non validi, registrando anche l'IP riportato nel formato standard di un IPv4. Con stats i dati vengono aggregate per contare I tentativi che superino I 5 fallimenti e I dati sono alla fine presentati in una tabella con due colonne, l'IP e i tentativi falliti. La query usata è la seguente: `source="tutorialdata.zip:/*" host="127.0.0.1" sourcetype="secure.log" "failed`

```
password" | rex "Failed password for (invalid user )?(?<user>\w+) from (?<src_ip>\d+\.\d+\.\d+\.\d+)" | stats count by src_ip | where count > 5 | table src_ip count.
```

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** source="tutorialdata.zip://" host="*127.0.0.1" sourcetype="secure.log" "failed password" | rex "Failed password for (invalid user)?(?<user>\w+) from (?<src_ip>\d+\.\d+\.\d+\.\d+)" | stats count by src_ip | where count > 5 | table src_ip count
- Results Summary:** 25.471 eventi (prima di 03/11/24 22:54:33.000) Nessun campionamento degli eventi *
- Event Count:** 25.471
- Table Headers:** src_ip #, count #
- Table Data:**

| src_ip # | count # |
|-----------------|---------|
| 107.3.146.207 | 184 |
| 188.65.113.83 | 213 |
| 189.169.32.135 | 391 |
| 110.138.30.229 | 118 |
| 110.159.288.78 | 68 |
| 111.161.27.20 | 62 |
| 112.111.162.4 | 185 |
| 117.21.246.164 | 155 |
| 118.142.68.222 | 90 |
| 12.130.60.4 | 214 |
| 12.130.60.5 | 189 |
| 121.254.179.199 | 125 |
| 121.9.245.177 | 119 |
| 123.118.73.195 | 134 |
| 123.196.113.11 | 137 |
| 123.30.108.208 | 125 |
| 124.168.192.241 | 117 |
| 125.17.14.100 | 115 |
| 125.7.55.180 | 188 |

Sempre utile ricordare che Splunk individua già automaticamente dei pattern sospetti e ricorrenti.

The screenshot shows a Splunk search interface with the following details:

- Search Bar:** source="tutorialdata.zip://" sourcetype="secure.log" | rex "Failed password for (?:invalid user)?(?<username>\S+)" | stats count as attempts by src_ip | where attempts > 5 | table src_ip attempts | rename src_ip as "Indirizzo IP", attempts as "Numero di Tentativi" | sort - attempts
- Results Summary:** 100.452 eventi (prima di 01/11/24 17:52:11,000) Nessun campionamento degli eventi *
- Event Count:** 100.452
- Table Headers:** Indirizzo IP, Numero di Tentativi
- Table Data:**

| Indirizzo IP | Numero di Tentativi |
|---------------|---------------------|
| 91.205.189.15 | 51,7% |
| 81.11.191.113 | 6,54% |

Anche le statistiche generate automaticamente sono interessanti.

The screenshot shows a Splunk search interface with the following details:

- Query:**

```
source="tutorialdata.zip:*" sourcetype="www/secur*" | rex "Failed password for (?<invalid user>)(?<username>\$*) from (?<src_ip>\$*)"
| stats count as attempts by src_ip
| where attempts > 5
| table src_ip attempts
| rename src_ip as "Indirizzo IP", attempts as "Numero di Tentativi"
| sort - attempts
```
- Results:** 181,572 eventi (prima di 09/11/24 18:03:01,000) Nessun campionamento degli eventi.
- Table View:** Shows a list of IP addresses and their attempt counts. The top entries are:

| Indirizzo IP | Numero di Tentativi |
|-----------------|---------------------|
| 107.146.287 | 1252 |
| 108.65.113.83 | 1166 |
| 109.169.32.135 | 2376 |
| 110.138.30.229 | 717 |
| 110.159.280.78 | 563 |
| 111.161.27.28 | 376 |
| 112.111.162.4 | 584 |
| 117.21.246.164 | 863 |
| 118.142.68.222 | 437 |
| 12.138.69.4 | 1095 |
| 12.138.69.5 | 691 |
| 121.254.179.199 | 829 |
| 121.9.245.177 | 720 |

Query numero 5.

Proposta numero 1 precisa:

La query che segue permette di trovare gli Internal Server Error, marcati con il codice di errore 500. Con l'operatore booleano vengono filtrati gli eventi che contengono il codice 500 oppure la dicitura “Internal Server Error”, mentre con il comando eval si crea un nuovo campo per chiarire lo status e i dati vengono rappresentati in una tabella che mostra, nelle colonne, i campi indicati. La query usata è la seguente: Source=“tutorialdata.zip:” host=“127.0.0.1” sourcetype=“access_combined_wcookie” (“Internal Server Error” OR “500”) | eval status=“Internal Server Error” | table _time src_ip status_code error_message status.

The screenshot shows a Splunk search interface with the following details:

- Query:**

```
source="tutorialdata.zip:/*" host="127.0.0.1" sourcetype="access_combined_wcookie" ("Internal Server Error" OR "500") | eval status="Internal Server Error" | table _time src_ip status_code error_message status
```
- Results:** 2.343 eventi (prima di 03/11/24 22:59:48,000) Nessun campionamento degli eventi.
- Table View:** Shows a list of events with columns: _time, src_ip, status_code, error_message, and status. The status column consistently shows "Internal Server Error". Some rows also show "Error Windows". The table includes many rows of timestamped entries like:

| _time | src_ip | status_code | error_message | status |
|---------------------|--------|-------------|---------------|-----------------------|
| 2024-10-30 18:18:59 | | | | Internal Server Error |
| 2024-10-30 18:18:59 | | | | Internal Server Error |
| 2024-10-30 18:18:59 | | | | Internal Server Error |
| 2024-10-30 18:18:55 | | | | Internal Server Error |
| 2024-10-30 18:18:55 | | | | Internal Server Error |
| 2024-10-30 17:42:03 | | | | Internal Server Error |
| 2024-10-30 17:42:03 | | | | Internal Server Error |
| 2024-10-30 17:42:03 | | | | Internal Server Error |
| 2024-10-30 17:42:03 | | | | Internal Server Error |
| 2024-10-30 17:42:00 | | | | Internal Server Error |
| 2024-10-30 17:17:00 | | | | Internal Server Error |
| 2024-10-30 17:17:00 | | | | Internal Server Error |
| 2024-10-30 17:15:13 | | | | Internal Server Error |
| 2024-10-30 17:15:13 | | | | Internal Server Error |
| 2024-10-30 17:08:43 | | | | Internal Server Error |
| 2024-10-30 17:08:43 | | | | Internal Server Error |
| 2024-10-30 16:54:07 | | | | Internal Server Error |

Proposta numero 2 più semplice:

La query che segue è più semplice ma ugualmente efficace, perché cerca i codici 500, legati all'Internal Server Error. È questa query: Source=“tutorialdata.zip:”

sourcetype="secure.log" | search 500. Il sourcetype qui è di nuovo secure.log, legato alla sicurezza.

The screenshot shows a Splunk search interface with the following details:

- Search Query:** source="tutorialdata.zip" * sourcetype="secure.log" | search 500
- Event Count:** 781 eventi (prima di 01/11/24 12:50:54,000) Nessun campionamento degli eventi ▾
- Panel Headers:** Processo, Sempre, Modальность детализации ▾
- Event List Headers:** Evento, Formato, 20 per pagina ▾
- Event Details:** The list includes several entries, such as:
 - 198.35.1.75 - [30/Oct/2024:18:18:59] "GET /cart.do?action=addtocart&itemID=EST-1&JSESSIONID=SD0185L2FF4ADFF530899 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 host: 127.0.0.1 source: tutorialdata.zip.lwwn2access.log sourcetype: secure.log
 - 198.35.1.75 - [30/Oct/2024:18:18:55] "GET /product.screen?productId=BVS-G81&JSESSIONID=SD0185L2FF4ADFF530899 HTTP/1.1" 500 2089 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-14" "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/536.5 (KHTML, like Gecko) Chrome/19.0.1084.46 Safari/536.5" 378 host: 127.0.0.1 source: tutorialdata.zip.lwwn2access.log sourcetype: secure.log
- Page Navigation:** 1 / 1 Rec 1 2 3 4 5 6 7 8 ... Avanti >

Gli eventi sono molti e la timeline aiuta a capire quando si sono verificati più frequentemente.

Sempre

source="tutorialdata.zip" & sourcetype="secure.log" | search 500

781 eventi (prima di 07/10/24 12:50:54,000) Nessun campionamento degli eventi ▾

Eventi (130) Pattern Statistiche Visualizzazione

Formato timeline ▾ Zoom indietro + Zoom area selezionata × Deselezione 24 ott 2024 25 ott 2024 1 giorno per colonna

1 day

Elenco ▾ Formato 20 per pagina ▾

1 2 3 4 5 6 7 Avanti >

< Nascondi campi ▾ CAMPI SELEZIONATI ▾ ▪ Tutti i campi ▾ ▪ host 1 ▪ source 3 ▪ sourcetype 1 ▾ CAMPI INTERESSANTI ▾ ▪ category 5 ▪ categoryId 5 ▪ date_hour 24 ▪ date_mday 1 ▪ date_min 53 ▪ date_month 1 ▪ date_sec 52 ▪ date_yday 1 ▪ date_year 1 ▪ date_zone 1 ▪ index 1 ▪ itemid 14 ▪ JSESSIONID 100+ ▪ InetSocketAddress 1 ▪ punct 5 ▪ punct2 32 ▪ spurlk_server 1 ▪ timestamppos 6 ▪ Estrarri nuovi campi ▾

| # | Ora | Evento |
|---|-----------------------|--|
| 1 | 24/10/24 23:54:42.000 | 27.35.11.11 -- [24/Oct/2024:23:54:42] "GET /product/screen?productId=SF-BVS-01&SESSIONID=\$D951FF560FF13435 HTTP/1.1" 500 1729 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-01" "Mozilla/4.0 (compatible; MSIE 7.8; Windows NT 5.1; .NET CLR 1.1.422; InfoPath 1; MS-RTC LH B)" 329 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 2 | 24/10/24 23:44:51.000 | 27.96.128.8 -- [24/Oct/2024:23:44:51] "POST /oldlink?itemID=EST-17A75E5530ND+\$D561FF3ADFF13381 HTTP/1.1" 500 3355 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-01" "Googlebot/2.1 (http://www.googlebot.com/bot.html)" 716 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 3 | 24/10/24 23:36:20.000 | 99.61.68.226 -- [24/Oct/2024:23:36:21] "GET /product/screen?productId=SF-BVS-01&SESSIONID=\$D45LFF44DF13337 HTTP/1.1" 500 1666 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; .NET 4.0; Trident/5.0; .NETCLR/4.0.2)" 376 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 4 | 24/10/24 23:40:07.000 | 233.74.94 -- [24/Oct/2024:23:34:07] "POST /cart.do?action=view&itemId=EST-17A75E5530ND+\$D45LFF2A0DF13320 HTTP/1.1" 500 1199 "http://www.buttercupgames.com/product.screen?productId=SF-BVS-01" "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2.28) Gecko/20100819 Firefox/3.12.28 (.NET CLR 3.5.3077; .NET4.0C)" 195 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 5 | 24/10/24 23:17:26.000 | 66.69.195.226 -- [24/Oct/2024:23:17:26] "GET /cart.do?action=addtocart&itemId=EST-14A75E5530ND+\$D561FF9ADFF13247 HTTP/1.1" 500 2085 "http://www.buttercupgames.com/category.screen?categoryId=NULL" "Mozilla/5.0 (Windows NT 6.1; .NET4.0.2)" 716 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 6 | 24/10/24 23:07:08.000 | 66.228.218.8 -- [24/Oct/2024:23:07:08] "GET /cart.do?action=view&itemId=EST-18A75E5530ND+\$D55L1FF1A0DF13184 HTTP/1.1" 500 2673 "http://www.buttercupgames.com/cart.do?action=view&itemId=EST-18" "Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 5.1; Trident/5.0; .NET CLR 3.5.3077; .NET4.0.2)" 447 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 7 | 24/10/24 22:43:55.000 | 67.240.18.18 -- [24/Oct/2024:22:43:55] "GET /oldlink?itemID=EST-16A75E5530ND+\$D5810FF5ADF13072 HTTP/1.1" 500 1108 "http://www.buttercupgames.com/oldlink?itemID=EST-16" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 1.0.3705.3705; .NET CLR 1.1.4720.2572)" 536 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 8 | 24/10/24 22:32:35.000 | 217.152.146 -- [24/Oct/2024:22:32:35] "GET /oldlink?itemID=EST-13A75E5530ND+\$D575L4FF9ADFF12992 HTTP/1.1" 500 338 "http://www.buttercupgames.com/oldlink?itemID=EST-13" "Mozilla/5.0 (compatible; NetcraftSurveyAgent/1.0; cc-prepass; http://www.buttercupgames.com/oldlink?itemID=EST-13)" 761 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |
| 9 | 24/10/24 22:00:23.000 | 10.0.347.127 -- [24/Oct/2024:22:00:23] "GET /oldlink?itemID=EST-10A75E5530ND+\$D5810FF5ADF13072 HTTP/1.1" 500 256 "http://www.buttercupgames.com/oldlink?itemID=EST-10" "Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR 1.0.3705.3705; .NET CLR 1.1.4720.2572)" 536 host: 127.0.0.1 source: tutorialdata.zip:/www3/access.log sourcetype: secure.log |

Come sempre, è possibile analizzare nel dettaglio il singolo evento.

Si possono sempre controllare i pattern individuati automaticamente da Splunk.

The first screenshot shows the search results for 'source=tutorialdata.zip:* sourcetype=secure.log' with 781 events found. It includes a summary table, event details, and a visualization section. A warning message indicates that fewer than 5,000 events might result in low-quality patterns.

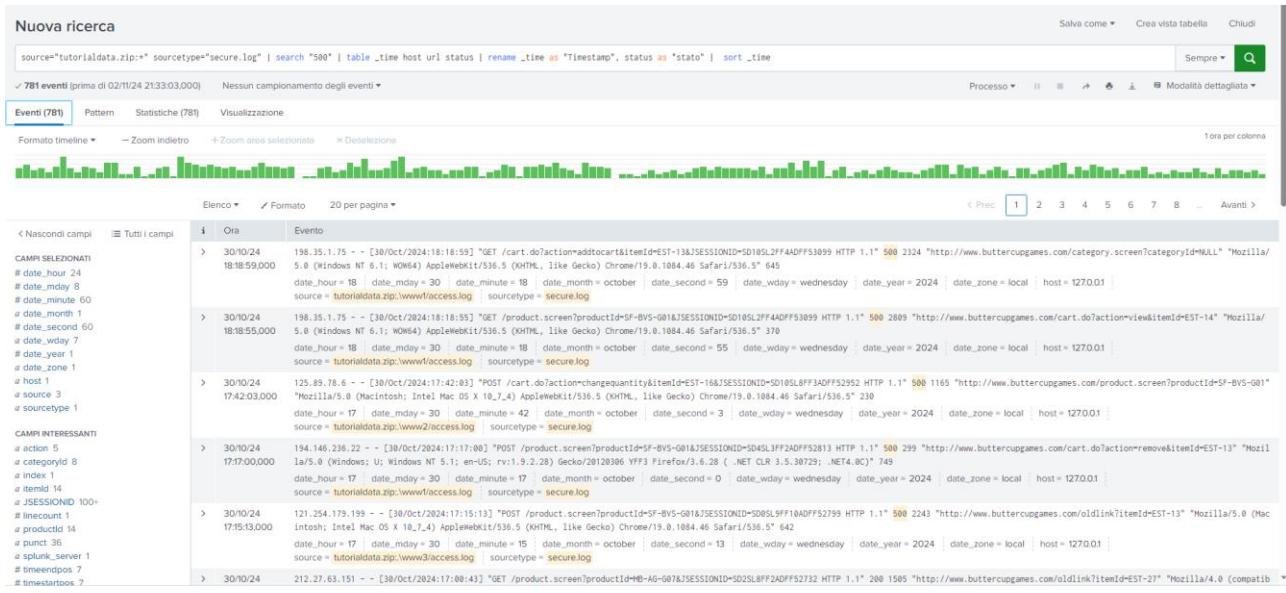
The second screenshot shows the same search results for a different timestamp, with 153 events found. It also includes a summary table, event details, and a visualization section. A similar warning message is present.

Questa è la query più semplice, eseguita cambiando sourcetype e scegliendo "snort".

This screenshot shows the search results for 'source=tutorialdata.zip:* sourcetype=snort'. It displays a detailed log entry from a network capture. The log entry shows a sequence of HTTP requests and responses between a client and a server, illustrating how Splunk can analyze network traffic using the 'snort' sourcetype.

Proposta numero 3 completa:

Si potrebbe usare anche la query `source="tutorialdata.zip:*` `sourcetype="secure.log"` | `search "500"` | `table _time url status` | `rename _time as "Timestamp"`, `status as "stato"` | `sort _time`. I dati possono essere visualizzati come tabella ma anche come eventi.



Analisi dei log con ChatGPT.

Ho ovviamente analizzato gli eventi man mano che eseguo le query e poi li ho fatti esaminare all'intelligenza artificiale, che ha confermato le mie ipotesi iniziali. Ho esaminato prima i singoli eventi trovati con le cinque query dell'esercizio e poi ho fatto analizzare gli screenshot di tutte le cinque query a ChatGPT, per avere una visione d'insieme e un'analisi complessiva. Inizierò con l'esame dei singoli eventi per poi passare all'analisi globale di tutta la situazione. Inserisco anche gli screenshot dei prompt che ho usato e delle risposte che ho ottenuto con ChatGPT, che uso sempre in inglese.

Query numero uno.

Dai log esaminati con la prima query emergono moltissimi tentativi di login SSH falliti, usando password errate. Salta all'occhio il fatto che le porte cambino sempre, molto probabilmente per evitare il rate limiting e tentare più porte possibili e dai timestamp si vede che i tentativi di login avvengono con tempistiche molto ristrette se non in simultanea. Questo può far pensare ad un attacco automatizzato, anche perché vengono provati degli username piuttosto comuni, come testuser o nomi molto diffusi. Per questo motivo, si potrebbe pensare ad un attacco di brute force su SSH, realizzato in maniera anche automatica per sfruttare delle credenziali deboli e poco elaborate. Se alcuni nomi utente ed account vengono bersagliati più spesso, è possibile che l'attaccante abbia già raccolto informazioni per scegliere il target. Tra le misure che si possono attuare immediatamente per limitare l'attacco, figurano il blocco degli indirizzi IP coinvolti maggiormente, fermare temporaneamente le connessioni SSH ed implementare con maggiore attenzione le rate limiting policies per rallentare questi tentativi di login pressanti.

I am examining some logs using Splunk, to find evidence of cyberattacks and other malicious activities. What happened on the basis of these screenshots?



The Splunk logs you've shared indicate repeated failed login attempts from various IP addresses, which suggests a brute-force attack or credential-stuffing attempt. Here's a detailed analysis based on the data:

1. **Failed Login Attempts:** Each log entry contains the phrase "Failed password for invalid user," followed by a username and the originating IP address. This indicates multiple unsuccessful SSH login attempts, likely targeting non-existent or disabled user accounts on the server.
2. **Source of the Attacks:**
 - **IPs:** The IP addresses attempting these logins are mostly from two ranges, 194.8.74.23 and 203.45.206.135. This is a common pattern in brute-force attacks where a set of IPs, often compromised or proxies, repeatedly try to access a system.
 - **Port Numbers:** The ports vary with each attempt (e.g., 3351, 3626, 2472), which may indicate the attacker is using different SSH connections to try each login, potentially to bypass rate-limiting.
3. **Timestamp and Frequency:**
 - The timestamps (04:36:08) are identical across the logs shown, which suggests that these login attempts were automated, possibly launched in quick succession. The frequency and

login attempts were automated, possibly launched in quick succession. The frequency and simultaneous attempts suggest the use of an automated script or botnet.

4. Invalid Usernames:

- The usernames being tried (e.g., appserver, testuser, mongod, guest) appear generic and are typical of default accounts on various systems. Attackers often try a list of commonly used or default usernames, as they may exist on improperly secured servers.

What This Indicates:

These repeated failed login attempts are strong indicators of a brute-force attack, where an attacker is trying numerous username-password combinations to gain unauthorized access. The attack aims to exploit weak passwords or default credentials.

Recommended Actions:

1. **Immediate Blocking:**
 - Block the offending IP ranges (194.8.74.23, 203.45.206.135) at the firewall level or configure SSH to deny connections from these sources temporarily.
2. **Enhanced Authentication:**
 - Implement multi-factor authentication (MFA) on SSH logins if possible.
 - Enforce strong passwords and consider using SSH keys for authentication instead of passwords.
3. **Rate-Limiting:**
 - Configure rate-limiting or delay mechanisms on SSH to slow down repeated login attempts.

3. Rate-Limiting:

- Configure rate-limiting or delay mechanisms on SSH to slow down repeated login attempts.

4. Monitoring and Alerts:

- Set up alerts in Splunk to notify your team of similar login attempts or failed login patterns that might indicate an attack in progress.
- Consider expanding logging to capture additional context, such as geolocation data of IPs, to understand if this is a localized or widespread attack.

5. Log Analysis:

- Continue reviewing logs for any successful logins from these IP addresses or unusual activity involving privileged accounts, as this may indicate a compromise.

These steps will help mitigate the current attack and strengthen defenses against future attempts.

Query numero due.

Questa query dimostra numerosi accessi SSH effettuati da parte dell'account djohnson avvenuti in tempi ristretti se non simultaneamente. È difficile pensare che un utente singolo legittimo faccia tutti questi accessi contemporaneamente; quindi, piuttosto si potrebbe pensare che l'account del nome utente djohnson sia stato compromesso da parte di un attaccante. Si potrebbe trattare di un attacco automatizzato, realizzato magari con degli script per mantenere la connessione. I login di djohnson possono essere collegati con i tentativi di brute force rilevati con la query precedente, per cui gli attaccanti sono riusciti a compromettere l'account con lo username djohnson. È necessario effettuare maggiori controlli sul possibile account compromesso, per esempio provando a geolocalizzare l'IP, resettando la password o bloccando temporaneamente l'account, specialmente se è un account che aveva dei privilegi particolari, per cui la sua compromissione diventa ancora più pericolosa.



Based on the logs you've provided, here's an analysis of potential security concerns:

1. Suspicious Login Attempts for User "djohnson":

- The first two screenshots show multiple successful SSH sessions opened for the user "djohnson" from the same source IP address (10.3.10.46) within a very short timeframe (04:36:08).
- These events suggest that either:
 - The user "djohnson" is performing legitimate activity, logging in multiple times rapidly, or
 - There is a possible security issue, where an attacker has compromised the "djohnson" account and is establishing multiple sessions to maintain persistent access.

2. Repetitive Access from the Same IP Address:

- All successful logins originate from 10.3.10.46 . If this is not a typical IP for "djohnson" or if it's from an unexpected or unusual location, it could be evidence of unauthorized access.
- The numerous connections from a single IP in a short period could indicate automated login behavior, which is often seen in malicious scenarios where attackers use scripts or bots to maintain connections.



3. Correlation with Prior Brute-Force Attempts:

3. Correlation with Prior Brute-Force Attempts:

- In your earlier screenshots, we saw multiple failed login attempts from different IP addresses targeting various usernames. If the account "djohnson" was one of those usernames, it's possible that the attacker successfully brute-forced this account, leading to the successful login attempts observed here.
- Even if "djohnson" was not explicitly targeted, it's possible that a compromised IP or user from an external brute-force attempt gained access through a different attack vector, such as a weak or reused password.

Potential Indicators of Malicious Activity

- **Rapid Session Creation:** Multiple sessions for "djohnson" created in a very short timeframe is atypical for a legitimate user and could suggest malicious activity.
- **Consistent IP and Ports Variability:** While the IP remains the same (`10.3.10.46`), the ports used in each login attempt differ, suggesting an effort to bypass detection or potential rate-limiting mechanisms.
- **Unusual User Behavior:** If "djohnson" does not typically log in from this IP or at this frequency, it might indicate unauthorized access.

Recommended Actions

1. Verify the Legitimacy of Activity:

- Check with "djohnson" or the system owner to confirm if they recognize the IP `10.3.10.46` and if these login activities are legitir ↴.
- Review historical logs to determine if "djohnson" has previously logged in from this IP.
- Review historical logs to determine if "djohnson" has previously logged in from this IP.

2. Password Reset and Account Lockdown:

- If the activity is suspicious, reset "djohnson's" password and temporarily lock the account to prevent further unauthorized access.
- Review and strengthen password policies if necessary.

3. Monitor and Correlate:

- Set up alerts for future login attempts by "djohnson" or other sensitive accounts, particularly for logins from unusual IP addresses or at unusual times.
- Correlate this activity with firewall logs to see if there is any traffic from `10.3.10.46` on other ports or to other internal systems, indicating lateral movement.

4. Implement Multi-Factor Authentication (MFA):

- If feasible, implement MFA for SSH access to add an additional layer of security, making it more difficult for an attacker to gain access even if they acquire valid credentials.

Conclusion

The logs provide **strong indicators of potentially malicious access** to the "djohnson" account. Multiple, rapid successful logins from the same IP, especially if atypical for the user, often suggest account compromise. Proceed with caution by verifying with the user and, if necessary, locking down the account and tightening security around SSH access.

Query numero tre.

Questi log sembrano confermare l'ipotesi di un attacco brute force su servizi SSH, poiché si vede un singolo IP che tenta l'accesso con vari username, alla ricerca di trovare una password debole o una configurazione non sicura per poter entrare. L'attacco potrebbe

essere effettivamente automatizzato, perché i tentativi di login avvengono nell'arco di più giorni allo stesso orario, facendo pensare ad uno script realizzato ad hoc per questo attacco. Le porte cambiano, facendo pensare ad una strategia per aggirare il rate limiting. La persistenza di un singolo indirizzo IP che fa numerosi tentativi di login su diverse porte con credenziali e username comuni fa venire in mente anche un attacco di tipo credential-stuffing, per rubare credenziali, avere accesso al sistema e trovare dati. Per far fronte a questa minaccia, è utile bloccare l'IP 86.212.199.60 o inserirlo in una blacklist, verificare la configurazione di SSH, modificare le policy per le password per richiedere password più forti, abilitare la multi factor authentication e rinforzare le strategie di rate limiting.



Examine these security logs analyzed with Splunk. What can be inferred from these logs? Are there the evidences of any malicious activity or cyberthreat?



These Splunk logs show strong evidence of malicious activity, specifically a **brute-force attack** targeting SSH services. Here's an analysis of what these logs indicate:

Observations from the Logs

1. Failed SSH Login Attempts:

- The logs display repeated failed login attempts with the message "Failed password for invalid user," targeting a variety of usernames (`db`, `services`, `admin`, `nginx`, `whois`, etc.).
- The usernames being used are generic and commonly associated with default accounts on various systems. Attackers often attempt to use these generic usernames to exploit weak or default passwords, as they may exist on systems with poor security configurations.

2. Single Source IP Address:

- All of these failed attempts originate from the same IP address, `86.212.199.60`, suggesting a focused attack from this source.
- This IP appears to be systematically probing for different usernames, which is a typical pattern in brute-force attacks, where attackers try a wide range of user accounts to find one with a weak password.



3. Frequent and Systematic Login Attempts:

- The attack is consistent across multiple days, targeting SSH services at almost the same time (04:36) each day. This level of persistence is common in automated attacks, where a script or botnet is programmed to perform login attempts at set intervals.
- This systematic approach indicates an automated tool, rather than a human, which aligns with common brute-force tactics.

4. Variety of Ports:

- The logs show login attempts across various ports (e.g., 2690, 3673, 1582, 4566), all aimed at the SSH service. The use of varied ports could indicate either:
 - A tactic to bypass simple rate-limiting or detection mechanisms, or
 - A misconfiguration in logging or a different SSH configuration across hosts.
- The attacker's strategy of rotating ports may help evade detection if certain ports are monitored more closely than others.

Malicious Activity Indicators

This pattern of failed login attempts is **strongly indicative of a brute-force attack** aimed at guessing passwords for different accounts. Key indicators include:

- High volume of failed login attempts for various usernames.
- Persistence from a single IP over multiple days.
- Use of common or default usernames often targeted in credential-stuffing or brute-force attacks.
↓

Potential Cyberthreat Implications

- If the attacker successfully brute-forces a valid account with a weak password, they could gain unauthorized access to your system.
- With SSH access, the attacker could potentially exfiltrate data, install malicious software, or pivot to other systems within the network.
- This type of attack could be part of a larger, more coordinated effort to infiltrate systems with weak security settings.

Recommended Actions to Mitigate This Threat

1. Block the Attacker's IP:

- Immediately block IP address 86.212.199.60 at the firewall level or within your SSH configuration to prevent further login attempts from this source.

2. Enforce Strong Passwords:

- Review and enforce strong password policies, especially for any commonly used or default usernames. Disable or remove default accounts if they are unnecessary.

3. Enable Multi-Factor Authentication (MFA):

- If possible, enable MFA for SSH access. This would provide an additional layer of security, making it significantly harder for an attacker to gain access even if they guess a password.

4. Configure Rate-Limiting or IP Banning:

- Implement fail2ban or similar software to automatically block IPs that exhibit suspicious behavior, such as multiple failed login attempts within a short period.
↓

4. Configure Rate-Limiting or IP Banning:

- Implement fail2ban or similar software to automatically block IPs that exhibit suspicious behavior, such as multiple failed login attempts within a short period.

5. Monitor for Unusual Login Activity:

- Set up alerts in Splunk for repeated failed login attempts, especially from unknown IPs or for common usernames. This proactive measure can help identify brute-force attacks early.

6. Review SSH Configuration:

- Restrict SSH access to trusted IPs or use VPNs for remote access to minimize exposure of SSH to the internet.
- Consider using non-standard SSH ports (although this is not a standalone security measure, it can reduce exposure to automated scans).

Conclusion

These logs provide clear evidence of a **sustained brute-force attack** against your SSH service.

Implementing the recommended actions can help mitigate this attack and prevent similar threats in the future. It is essential to act quickly to block the attacking IP and strengthen your defenses against unauthorized access.



Query numero quattro.

Questi log rafforzano l'interpretazione che si tratti di un attacco di brute force e fanno pensare anche ad un attacco di tipo credential stuffing, per cui si tenta di accedere ai sistemi con credenziali rubate o già individuate. Si tratta di un attacco di brute force distribuito, perché gli IP coinvolti sono moltissimi e da reti diverse, per cui diventa più difficile contrastare l'attacco. Sembra anche un attacco persistente ed automatizzato, perché gli IP provano numerose volte gli accessi e raggiungono picchi anche importanti di attività. Per difendersi da un attacco di questo tipo, in aggiunta alle misure già elencate precedentemente come bloccare gli IP più aggressivi, si possono usare tecnologie come fail2ban, che bloccano automaticamente gli IP che falliscono il login dopo un certo numero di tentativi, senza quindi che un esperto di sicurezza debba intervenire. Si può attivare anche la geolocalizzazione degli IP, per capire se l'attacco proviene da una specifica regione geografica e si può anche fare una verifica sugli account, per capire se qualche account è già stato compromesso per bloccarlo prima che faccia danni e resettare le credenziali compromesse.

Analyze now these log data from Splunk. Can you detect a cyberthreat or a malicious pattern?



Yes, these Splunk data visualizations show evidence of a **brute-force attack** or **credential-stuffing attack** on the system. Here's an analysis of the patterns observed:

Key Indicators of Malicious Activity

1. High Number of Failed Login Attempts by IP Address:

- The logs display a list of source IPs with a high number of failed login attempts. For instance, several IPs have over 100 failed attempts, with some reaching as high as 392. This pattern is typical of automated brute-force attacks, where multiple usernames and passwords are systematically attempted to gain unauthorized access.
- The query used (`where attempts > 5`) filters for IP addresses with more than five failed login attempts, indicating that only persistently attempting IPs are shown, further reinforcing the likelihood of malicious activity.

2. Visualization of Failed Attempts by IP:

- The bar chart in the second screenshot shows the distribution of failed attempts by each IP address. There are noticeable spikes for certain IPs, indicating that some addresses are attempting login far more aggressively than others.
- The peaks in failed attempts suggest concentrated efforts from specific IPs, potentially

- The peaks in failed attempts suggest concentrated efforts from specific IPs, potentially using automated scripts or bots to break into the system by cycling through different credentials.

3. Persistent and Distributed Attack Pattern:

- The IPs listed come from various networks, implying that the attack might be distributed across multiple locations or routed through various proxy servers or compromised machines. Distributed brute-force attacks are harder to block as they use multiple IP addresses, avoiding detection through typical rate-limiting or IP-blocking mechanisms.

What This Pattern Implies

The observed pattern of high-volume, failed login attempts from multiple IPs strongly suggests a coordinated **brute-force attack**. Attackers are likely trying to guess valid username-password pairs, potentially to gain unauthorized SSH access. This can lead to significant security risks, as successful entry could allow the attacker to:

- Access sensitive data,
- Pivot to other systems within the network,
- Install malware or launch further attacks from within.

Recommended Defensive Actions

To mitigate this threat, consider the following steps:

1. Block Aggressive IPs at the Firewall:

- Immediately block the IPs with the highest number of failed attempts at the firewall level to prevent further attacks from these sources.

2. Rate Limiting and IP Blocking (Intrusion Prevention):
 - Implement rate limiting on login attempts and use tools like fail2ban to automatically block IPs after a certain number of failed login attempts.
3. Enable Multi-Factor Authentication (MFA):
 - Add MFA for SSH or other critical services. This makes it significantly harder for attackers to gain access even if they guess a valid password.
4. Check for Compromised Accounts:
 - Investigate if any of the brute-force attempts were successful by reviewing login success logs for the same period. Any compromised accounts should have their passwords reset, and an incident response process should be initiated if unauthorized access is confirmed.
5. Set Up Alerts in Splunk:
 - Configure Splunk to alert security personnel whenever there is a high number of failed login attempts from any single IP or when login attempts exceed a normal threshold, so that future attacks can be detected early.
6. Consider Geolocation Analysis:
 - Use IP geolocation to determine if login attempts are coming from regions where legitimate access is unlikely. This can help inform decisions about blocking specific regions or setting up additional access restrictions.

Conclusion

The data shows a clear **brute-force attack** pattern, with multiple IPs attempting a high number of failed logins. Implementing the recommended steps will help mitigate this threat and reduce the risk

Conclusion

The data shows a clear **brute-force attack** pattern, with multiple IPs attempting a high number of failed logins. Implementing the recommended steps will help mitigate this threat and reduce the risk of unauthorized access due to credential guessing. This proactive approach to blocking suspicious IPs and setting up alerts will strengthen the system's resilience against future brute-force attempts.

Query numero cinque

Un internal server error con codice 500 così ripetuto potrebbe implicare la scoperta e lo sfruttamento di una vulnerabilità del server da parte degli attaccanti, i quali provocano l'errore interno al server che, di conseguenza, non riesce a rendere disponibili le risorse e i servizi richiesti. La frequenza di queste richieste fa pensare all'utilizzo di un bot o di uno script automatico, che magari sta testando il server in una fase iniziale di scansione e ricerca delle vulnerabilità. Le richieste provengono da normali browser, da browser meno utilizzati ed anche da strumenti come Googlebot, che fanno pensare ad un user-agent spoofing, ossia una tattica che usa un crawler legittimo per bypassare meccanismi di sicurezza. Questo pattern può far pensare ad un DDoS a livello applicativo, per andare a saturare le risorse del server, oppure si potrebbe trattare di un vulnerability scanning o di un tentativo di reconnaissance per studiare se vari tipi di attacchi sono fattibili. Si potrebbe avere a che fare anche con un caso di web scraping, con tentativi automatici di monitorare i prezzi e le quantità dei prodotti dello shop online (si vedano i numerosi "addtocart" evidenziati anche nei pattern automatici di Splunk) oppure con un tentativo per sfruttare una vulnerabilità specifica scoperta del server. Tra le misure di difesa che si possono implementare vi sono i controlli contro i bot, la verifica delle configurazioni del server e l'applicazione delle patch di aggiornamento.



Examine these screenshot from security logs analyzed with Splunk. Is there evidence of a cyberthreat or malicious activity?



The logs you've shared provide several indications of potentially **malicious or suspicious activity**, specifically focused on repeated HTTP `500` errors across various product and category pages on the website. Here's an analysis of what these logs might imply:

Key Observations

1. HTTP 500 Status Code Responses:

- HTTP `500` errors indicate **internal server errors**. They usually occur due to server-side issues, such as application bugs, server misconfiguration, or overload. Repeated `500` errors in this context may suggest:
 - An application vulnerability that is being triggered, intentionally or unintentionally.
 - A malfunction in the web application, potentially leading to degraded user experience or exposing server information.

2. High Frequency of Requests:

- The logs show multiple requests to various endpoints, particularly for actions like:
 - `GET /cart.do?action=addtocart`, `GET /product.screen`, and `POST /oldlinkitem`.
- The high frequency of requests to the `pages`, especially with the `500` response status, could indicate that a **bot or automated tool** is probing the site's functionality, potentially in

could indicate that a **bot or automated tool** is probing the site's functionality, potentially in a **reconnaissance phase or vulnerability scanning**.

3. Varied User-Agents:

- The requests come with various `User-Agent` strings, including:
 - Common browsers like `Chrome`, `Firefox`, `Safari`.
 - Older and less common agents, such as `Mozilla/4.0` and `MSIE 7.0`.
 - Notably, one request appears to use `Googlebot/2.1`, which could be a legitimate crawler or a **spoofed user-agent** to mask bot activity.
- The diversity in user-agent strings could indicate **user-agent spoofing**, a tactic often used in automated attacks to bypass detection mechanisms that block or limit repeated requests from a single user-agent.

4. Repeated Requests to Specific Product IDs:

- The requests target specific `productId` or `itemId` values (`SF-BVS-G01`, `EST-13`, `EST-18`, etc.). This could indicate that:
 - The attacker is focusing on certain products to identify vulnerabilities, such as parameter tampering vulnerabilities.
 - There might be an attempt to exploit application logic related to product or cart operations (e.g., adding items to the cart or manipulating product details).

Potential Cybersecurity Implications

The pattern observed could indicate one or more of the following malicious activities:

1. Application Layer DDoS Attack:
 - If the volume of these requests is very high, it could be an application layer DDoS (Denial of Service) attack, aimed at overwhelming specific application functions and causing legitimate user requests to fail with `500` errors.
2. Vulnerability Scanning or Reconnaissance:
 - The repeated requests to different endpoints and item IDs, along with varied user-agents, suggest that an automated tool may be probing the application for vulnerabilities. This could be an attempt to identify misconfigurations, weak parameters, or vulnerabilities like **SQL injection**, **local file inclusion (LFI)**, or **cross-site scripting (XSS)**.
3. Bot Activity for Web Scraping or Data Extraction:
 - If the activity is not necessarily malicious but focused on gathering product or pricing data, it could be a bot performing **web scraping**. While this may not directly harm the application, it could lead to unauthorized data collection, putting proprietary information at risk.
4. Potential Exploitation of Server-Side Vulnerabilities:
 - Since `500` errors indicate server-side issues, these could be triggered by attempts to exploit server vulnerabilities. Attackers might be testing how the application handles malformed requests or edge cases, which could be a prelude to exploiting an identified weakness.

Recommended Actions

1. Analyze and Patch Vulnerabilities:
 - Investigate the root cause of these `500` errors by reviewing application error logs and server configurations. Look for any unhandled exceptions, resource exhaustion, or other vulnerabilities that could be causing these errors.
 - If an application vulnerability is identified, work with the development team to patch it as soon as possible.
2. Implement Rate Limiting and Bot Mitigation:
 - Set up rate limiting to block IP addresses or user-agents that exceed a reasonable threshold of requests within a specific timeframe. This can help mitigate both DDoS attacks and aggressive bot activity.
 - Consider using bot protection solutions (e.g., CAPTCHA challenges or bot detection services) to identify and block automated requests.
3. Monitor for IPs with High Error Rates:
 - Track IP addresses that generate a high frequency of `500` errors and consider blocking or challenging these IPs, as they may be associated with suspicious activity.
 - Since `500` errors can be a side effect of parameter tampering or injection attacks, keep a close eye on requests with unusual parameters or repetitive access to sensitive endpoints.
4. Check for User-Agent Spoofing:
 - Review user-agents to detect patterns or inconsistencies that indicate spoofing. It may be beneficial to apply stricter rules on outdated or uncommon user-agents, especially those that are not typically used to access ' application.'

5. Enable Alerts for Anomalous Activity:

- Configure Splunk to alert on anomalies such as surges in 500 errors, unusual user-agent patterns, or spikes in requests to specific endpoints. This can help you detect and respond to potential reconnaissance or attacks early.

Conclusion

The logs show potential malicious activity or bot-driven probing against the website, evidenced by repeated requests resulting in 500 errors across various endpoints. This pattern could be part of a vulnerability scan, bot scraping, or even an attempt to exploit application vulnerabilities.

Implementing the recommended actions will help mitigate these risks and improve your defenses against both automated and targeted attacks.

Analisi complessiva dei log con ChatGPT.

Ho fatto esaminare tutti i log a ChatGPT per avere un'analisi complessiva ed ha confermato degli aspetti che avevo notato anch'io. Tuttavia certe intuizioni che ho avuto non sono state riportate anche dall'intelligenza artificiale, che si conferma come un valido supporto per rendere più efficiente il lavoro ma che non sostituisce completamente l'intuito umano di un analista. È ovviamente stata confermata l'ipotesi di un attacco brute force su accessi SSH, come dimostrano i numerosi accessi falliti da parte di molti indirizzi IP, alla ricerca di trovare le credenziali giuste per accedere ai sistemi. I molti accessi effettuati dall'account djohnson possono far pensare ad un account compromesso da cui poi muoversi nel sistema. Si potrebbe trattare anche di un'azione di reconnaissance ed enumerazione dei servizi, che serve ad un attaccante per raccogliere informazioni riguardo ai servizi attivi, ai processi e alle vulnerabilità che possono poi essere sfruttate in un secondo momento durante l'attacco vero e proprio. Si ha a che fare con un attacco di brute force distribuito, realizzato con più dispositivi su più reti e anche con dei bot per automatizzare le richieste. Si conferma infatti la presenza di bot automatizzati che svolgono attività malevole con http sul server, per esempio nell'ambito dello scraping, per cui vengono controllati automaticamente i prezzi e le quantità dei prodotti effettuando numerose richieste http per simulare il comportamento degli utenti che acquistano online. Non va infatti dimenticato che tutorialdata.zip simula i log di uno shop online, per cui i bot eseguono numerose azioni automatiche per il controllo dei prezzi dei prodotti, anche per attività fraudolente. Si conferma anche l'ipotesi di un attacco di credential stuffing, per cui gli attaccanti usando credenziali già compromesse oppure rubate per accedere ad SSH e i numerosi login falliti sono il segno di tutti i tentativi già effettuati con le credenziali compromesse trovate precedentemente. È importante intervenire subito, perché un attaccante potrebbe tentare una SQL injection avendo sfruttato una vulnerabilità legata all'input utente che ha portato ai numerosi internal server error oppure l'attaccante potrebbe trovare un modo per sfruttare le API del sito web, con azioni come "addtocart" e "changequantity". Come già spiegato prima, serve implementare il rate limiting per evitare ripetuti tentativi di login e sono utili degli account lockouts per bloccare automaticamente un account, magari compromesso, dopo un certo numero di login falliti. Si può ricorrere anche a fail2ban oppure implementare un IDS o IPS per il rilevamento rapido di minacce legate a questi IoC, come i ripetuti login falliti e l'altissimo numero di richieste in poco tempo. È sempre utile mantenere la sicurezza delle password, in modo che siano sufficientemente complesse anche grazie all'implementazione di una policy di gruppo ed abilitare l'autenticazione a più fattori. Visto che SSH è stato particolarmente bersagliato, serve pensare ad un hardening di SSH, ad esempio disabilitando gli accessi con privilegi di root ad SSH, migliorando la sicurezza dei meccanismi di autenticazione SSH e spostando

SSH su porte non standard, che non vengano trovate immediatamente dagli attaccanti. Serve configurare correttamente anche il firewall ed implementare una politica seria di logging e monitoring, avvalendosi di un SIEM o ancora meglio di un SOAR o comunque analizzando frequentemente i log, anche grazie a Splunk, con il quale si possono settare degli alert in caso di pattern sospetti come quelli trovati con questo esercizio.

I am examining security logs using Splunk. Analyze the screenshots provided and explain in details the type of attack, cyberthreat and malicious activity shown in these logs. What can be inferred from these screenshots, what are the indicators of compromise, which threat is being faced and what measures can be taken to protect the system against this type of attack?



From the provided screenshots, it appears that the logs capture a pattern of attempted unauthorized access, specifically brute-force attacks, targeting SSH (Secure Shell) services on various user accounts and ports. Here's a detailed analysis of the attack and recommended countermeasures.

1. Type of Attack

- **Brute-Force Attack on SSH:** The repeated "Failed password for invalid user" messages, with attempts targeting a wide range of usernames (such as `appserver`, `mongodb`, `desktop`, `admin`, etc.), indicate a brute-force attack. The attacker is systematically attempting various usernames and possibly passwords in an effort to gain unauthorized access to the SSH service.
- **Successful Logins:** There are instances indicating successful login attempts (e.g., "session opened for user `djohnson`"), suggesting that either valid credentials were discovered or that legitimate users are accessing the system amidst these attacks.
- **Distributed Sources:** The screenshots show addresses from different locations attempting access, suggesting either a **distributed brute-force attack** (possibly involving botnets) or

access, suggesting either a **distributed brute-force attack** (possibly involving botnets) or repeated, automated attacks from multiple compromised hosts.

2. Indicators of Compromise (IoCs)

- **Repeated Failed Login Attempts:** The logs show numerous failed login attempts across various user accounts. This indicates a brute-force attack trying multiple username and password combinations.
- **IP Addresses of Interest:** Specific IP addresses appear frequently with failed login attempts, such as `194.8.74.23`, `86.212.199.60`, `203.45.206.135`, and others listed in your data. These IPs are likely part of the attacker's network.
- **Frequent Use of Non-Standard Usernames:** The attackers are attempting login with usernames that are either generic (like `admin`, `testuser`, `mongodb`) or random (like `ypguser`, `harrison`), which may not correspond to real users on the system.
- **High Number of Requests:** In the last screenshot, we see a histogram showing the volume of attempts from each IP address. Spikes from particular IP addresses or clusters indicate persistent attackers, further supporting the brute-force hypothesis.

3. Threats Being Faced

- **Credential Stuffing and Brute-Force Attacks:** The attacker is attempting to use brute-force techniques to discover valid credentials. If successful, this could lead to unauthorized access, data exfiltration, or lateral movement within the network.
- **Potential Distributed Attack:** Given the different IP addresses involved, this may indicate a

- **Potential Distributed Attack:** Given the different IP addresses involved, this may indicate a coordinated distributed attack, possibly by a botnet.
- **Account Compromise:** A successful brute-force attack on SSH can lead to account compromise, providing the attacker with shell access to the server. This is a critical threat as SSH access can allow for extensive control over the system.

4. Recommended Measures to Protect the System

- **Implement Rate Limiting or Account Lockouts:** Set up rate limiting to prevent rapid repeated login attempts. Configure account lockouts after a certain number of failed login attempts to slow down brute-force attacks.
- **Use Multi-Factor Authentication (MFA):** Require MFA for SSH access where feasible. This adds an additional layer of security, making it significantly harder for attackers to gain access even if they guess valid usernames and passwords.
- **Implement IP-Based Blocking or Geo-Blocking:** Block IP addresses after a threshold of failed login attempts. Geo-blocking can be used to deny access from regions where no legitimate users are expected.
- **Use Fail2ban or Similar Intrusion Prevention Tools:** Fail2ban is a tool that can monitor log files and ban IP addresses after multiple failed login attempts, automatically preventing brute-force attacks.
- **Require Strong Password Policies:** Enforce strong password requirements to make brute-force attacks less effective. Educate users on creating complex passwords.

- **SSH Hardening:**
 - **Disable SSH Root Login:** Configure SSH to deny root logins and only allow access to authorized users.
 - **Use Key-Based Authentication:** Prefer public key authentication over password-based logins for SSH access.
 - **Change the Default SSH Port:** Move SSH to a non-standard port to reduce exposure to automated attacks targeting default ports.
- **Log Monitoring and Alerts:** Configure Splunk to send real-time alerts when suspicious activity, such as multiple failed login attempts from the same IP or different IPs in a short span, is detected.
- **Network Segmentation and Firewalls:** Limit SSH access to trusted IP addresses only, using firewalls or other network controls.

5. Conclusion

- The logs indicate an ongoing brute-force attack on SSH services, possibly from multiple sources (distributed attacks), attempting to compromise user accounts through repeated login attempts.
- Key IoCs include multiple failed logins for various usernames, high-frequency login attempts from specific IP addresses, and logs showing both failed and successful login events.
- To defend against this type of attack, a layered security approach should be adopted, combining access control policies, account security hardening, monitoring, and automated defense tools like Fail2ban and MFA.

Proactive Steps for Future Protection

- **Regular Audits:** Continuously monitor and audit SSH access logs to identify abnormal patterns early.
- **Threat Intelligence Feeds:** Integrate threat intelligence feeds to block known malicious IPs in real time.
- **User Awareness and Training:** Ensure users understand the importance of strong passwords and MFA. They should also be vigilant against phishing and credential theft, as these could indirectly lead to brute-force success.

By implementing these protections, the system can be hardened against brute-force attacks and potential account compromise, safeguarding sensitive assets accessible through SSH.