

REPORT PROGETTO M6 W24D4

ANALISI DEL MALWARE E SPLUNK

Federica Di Fiore
Cybersecurity Analyst PT

Indice

Introduzione.....	2
Obbiettivi del progetto.....	2
Strumenti utilizzati	2
ANALISI E RISULTATI.....	2
Query n° 1 - Identificazione dei tentativi di accesso falliti	2
Query n° 2 - Sessioni SSH riuscite per l'utente "djohnson"	4
Query n° 3 - Tentativi di accesso falliti provenienti dall'indirizzo IP "86.212.199.60"	5
Query n° 4 - Identificazione degli indirizzi IP con più di 5 tentativi di accesso falliti.....	6
Query n° 5 - Ricerca degli errori "Internal Server Error" (codice 500)	7
Conclusioni sull'analisi dei log con supporto AI	8

Introduzione

Il presente report documenta l'analisi di file di log di sistema (secure.log e access.log) utilizzando Splunk come strumento di indicizzazione e ricerca.

Lo scopo è identificare attività sospette, possibili intrusioni e problemi di stabilità del servizio, sfruttando query SPL mirate e tecniche di estrazione dati.

Obbiettivi del progetto

- Effettuare ricerche mirate su log di sicurezza e accesso.
- Identificare potenziali minacce, come attacchi brute-force o accessi non autorizzati.
- Evidenziare errori applicativi che possano compromettere la stabilità del sistema.
- Fornire raccomandazioni basate sull'analisi.

Strumenti utilizzati

- Splunk (motore di indicizzazione e ricerca log)
- Dataset forniti (recuperati dal file tutorialdata.zip):
- **secure.log** (tentativi di accesso SSH)
 - **access.log** (registrazioni HTTP, inclusi errori)

ANALISI E RISULTATI

Query n° 1 – Identificazione dei tentativi di accesso falliti

Per lo svolgimento del primo punto della consegna, è stata utilizzata la seguente query:

```
index=* source="secure.log" "Failed password"
| rex "Failed password for(<reason> invalid user)? (?<username>\w+) from (?<ip>\d{1,3}(\.?\d{1,3}){3}) port (?<port>\d+)"
| eval reason=if(isnotnull(reason), "invalid user", "wrong password")
| table _time ip username port reason
```

Questa query è stata realizzata per identificare tutti i tentativi di accesso falliti registrati nei log secure.log, filtrando in particolare quelli che contengono la stringa "Failed password", tipica dei messaggi SSH di autenticazione non riuscita.

1. **index=* source="secure.log" "Failed password"** > filtro iniziale, questo comando cerca tutti gli eventi nei log secure.log contenenti il messaggio "Failed password".

2. **rex "Failed password for(?<reason> invalid user\?) (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<port>\d+)" >** Il comando rex utilizza una regex per estrarre i seguenti campi:
 - **reason:** per determinare se l'utente è inesistente (invalid user) oppure se la password è errata.
 - **username:** il nome utente usato nel tentativo di login.
 - **ip:** l'indirizzo IP di origine del tentativo di accesso.
 - **port:** la porta da cui è stato effettuato il tentativo.
3. **eval reason=if(isnotnull(reason), "invalid user", "wrong password") >** questo passaggio assegna la ragione del fallimento in modo esplicito:
 - **"invalid user"** se il campo reason è stato rilevato
 - **"wrong password"** se il campo reason è nullo
4. **table _time ip username port reason >** Visualizza i risultati in formato tabellare, includendo:
 - **Timestamp** (_time)
 - **Indirizzo IP** (ip)
 - **Nome utente** (username)
 - **Porta** (port)
 - **Motivo del fallimento** (reason)

La query restituisce una tabella dettagliata che permette di individuare rapidamente i tentativi di accesso SSH falliti, distinguendo tra utenti inesistenti e password errate. Questo consente di identificare potenziali attacchi brute-force o di enumerazione degli utenti, e di agire prontamente per rafforzare la sicurezza.

The screenshot shows a Splunk search interface. At the top, the search bar contains the following query:

```
index= source="secure.log" "Failed password"
| rex "Failed password for(?<reason> invalid user\? ) (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<port>\d+)"
| eval reason=if(isnotnull(reason), "invalid user", "wrong password")
| table _time ip username port reason
```

Below the search bar, the interface shows 16,952 events. The results are displayed in a table with the following columns: _time, ip, username, port, and reason. The table is sorted by _time in descending order.

_time	ip	username	port	reason
2025-08-03 04:07:07	67.133.102.54	nagios	1213	wrong password
2025-08-03 04:07:07	67.133.102.54	rdp	3856	invalid user
2025-08-03 04:07:07	67.133.102.54	testuser	3526	invalid user
2025-08-03 04:07:07	88.191.145.142	vmware	2438	invalid user
2025-08-03 04:07:07	88.191.145.142	yp	4782	invalid user
2025-08-03 04:07:07	88.191.145.142	ben	4842	invalid user
2025-08-03 04:07:07	88.191.145.142	mail	4626	wrong password
2025-08-03 04:07:07	88.191.145.142	administrator	4711	invalid user
2025-08-03 04:07:07	88.191.145.142	root	2747	wrong password
2025-08-03 04:07:07	88.191.145.142	noone	2695	invalid user
2025-08-03 04:07:07	88.191.145.142	jabber	1839	invalid user
2025-08-03 04:07:07	88.191.145.142	root	2342	wrong password
2025-08-03 04:07:07	88.191.145.142	gitosis	3681	invalid user
2025-08-03 04:07:07	88.191.145.142	admin	1332	invalid user
2025-08-03 04:07:07	88.191.145.142	inet	3047	invalid user
2025-08-03 04:07:07	88.191.145.142	informix	2122	invalid user
2025-08-03 04:07:07	88.191.145.142	guest	4590	invalid user
2025-08-03 04:07:07	88.191.145.142	bfuser	1632	invalid user
2025-08-03 04:07:07	88.191.145.142	operator	4998	invalid user
2025-08-03 04:07:07	221.207.229.6	perl	4860	invalid user

Figura 1 Risultato query n°1

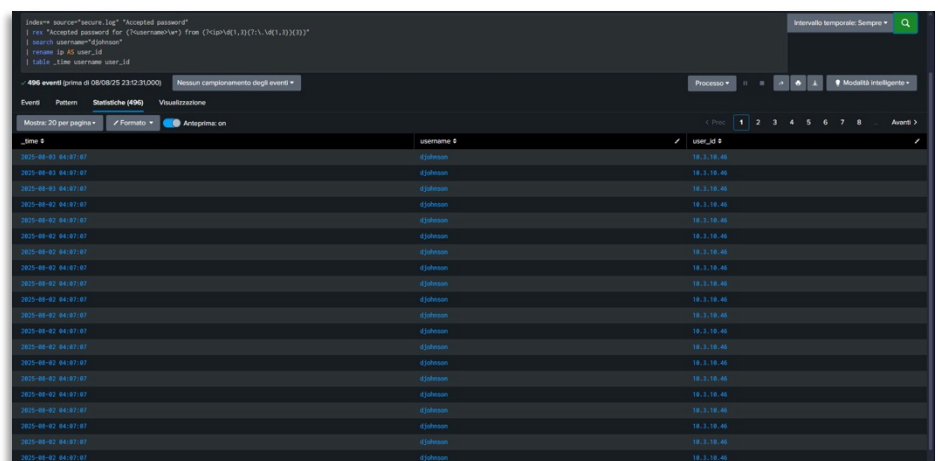
Query n°2 - Sessioni SSH riuscite per l'utente "djohnson"

```
index=* source="secure.log" "Accepted password"
| rex "Accepted password for (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3})"
| search username="djohnson"
| rename ip AS user_id
| table _time username user_id
```

Questa query è stata sviluppata per identificare tutte le sessioni SSH aperte con successo da parte dell'utente **djohnson**, visualizzando per ciascuna il **timestamp** e l'**indirizzo IP** di origine.

1. **index=* source="secure.log" "Accepted password"** > filtro iniziale, questo comando cerca nei log **secure.log** tutti gli eventi che indicano un accesso SSH andato a buon fine (ovvero quando viene visualizzato il messaggio "Accepted password").
2. **rex "Accepted password for (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3})"** > Il comando **rex** utilizza un'espressione regolare per estrarre i seguenti dati:
username: l'utente che ha effettuato con successo il login SSH.
ip: l'indirizzo IP da cui è stata avviata la connessione.
3. **search username="djohnson"** > Viene applicato un filtro per visualizzare esclusivamente gli eventi relativi all'utente **djohnson**.
4. **rename ip AS user_id** > L'indirizzo IP viene rinominato come **user_id**, in modo da rispettare le specifiche richieste nella traccia.
5. **table _time username user_id** > Viene restituita una tabella contenente:
 - **Timestamp dell'accesso** (_time);
 - **Nome utente** (username);
 - **IP di origine** (user_id).

La query restituisce un elenco di tutte le sessioni SSH correttamente avviate dall'utente **djohnson**, mostrando per ciascuna il momento esatto del login e l'indirizzo IP da cui è stato effettuato. Questa informazione è utile per monitorare i comportamenti di accesso e rilevare eventuali anomalie legate all'account.



_time	username	user_id
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40
2023-08-01 04:00:01	djohnson	10.1.10.40

Figura 2 Risultato query n°2

Query n° 3 - Tentativi di accesso falliti provenienti dall'indirizzo IP "86.212.199.60"

```
index=* source="secure.log" "Failed password" "86.212.199.60"
| rex "Failed password for( invalid user)? (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<port>\d+)"
| table _time, username, port
```

Questa query è stata costruita per individuare tutti i tentativi di autenticazione SSH falliti provenienti dallo specifico indirizzo IP 86.212.199.60. I dati restituiti includono:

- **Timestamp** dell'evento,
- **Nome utente** usato nel tentativo di accesso,
- **Numero di porta** utilizzata.

1. **index=* source="secure.log" "Failed password" "86.212.199.60" >** Questo comando cerca nei file secure.log tutti gli eventi che contengono contemporaneamente:
 - "Failed password" (indicatore di login fallito)
 - l'indirizzo IP 86.212.199.60
2. **rex "Failed password for(invalid user)? (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3}) port (?<port>\d+)" >** Con rex, si estrae:
 - **username:** il nome utente utilizzato nel tentativo di login fallito
 - **ip:** l'indirizzo IP (già noto in questo caso)
 - **port:** la porta su cui è avvenuto il tentativo di connessioneL'opzione (invalid user)? viene usata per gestire correttamente entrambi i tipi di errore SSH:
 - "invalid user" (utente inesistente)
 - "wrong password" (utente esistente, password errata)
3. **table _time username port >** Il risultato mostra per ogni tentativo:
 - l'orario dell'evento (_time)
 - il nome utente usato (username)
 - la porta di accesso (port)

La query consente di monitorare i tentativi di attacco brute-force o scansioni automatizzate da parte dell'indirizzo IP 86.212.199.60. Identificando nomi utente e porte usate, è possibile rafforzare le difese del sistema, attuare contromisure (es. blocco IP o configurazione fail2ban), e migliorare la rilevazione degli attacchi SSH.

_time	username	port
2025-07-28 04:02:07	guest	2222
2025-07-28 04:02:07	testlog	4040
2025-07-28 04:02:07	test	2178
2025-07-28 04:02:07	sumy	2765
2025-07-28 04:02:07	root	1581
2025-07-28 04:02:07	mail	4535
2025-07-28 04:02:07	mailman	2541
2025-07-28 04:02:07	ventrilo	1685
2025-07-28 04:02:07	tomcat	1135
2025-07-28 04:02:07	sumy	4044
2025-07-28 04:02:07	infomex	3648
2025-07-28 04:02:07	administrator	2558
2025-07-28 04:02:07	appserver	4079
2025-07-28 04:02:07	britany	4039
2025-07-28 04:02:07	shoucast	2227
2025-07-28 04:02:07	nagios	5562
2025-07-28 04:02:07	oracle	2411
2025-07-28 04:02:07	nmap	2381
2025-07-28 04:02:07	mail	2713
2025-07-28 04:02:07	testuser	1622

Figura 3 Risultato query n°3

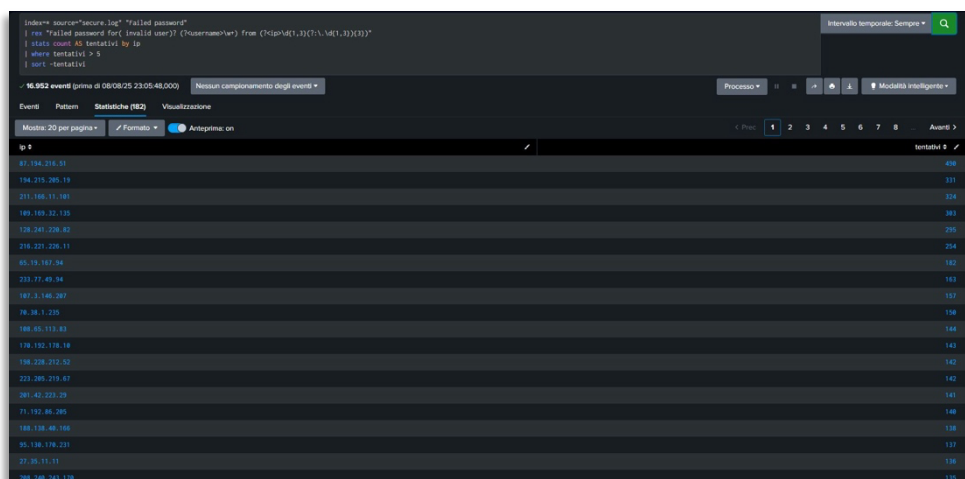
Query n° 4 - Identificazione degli indirizzi IP con più di 5 tentativi di accesso falliti

```
index=* source="secure.log" "Failed password"
| rex "Failed password for( invalid user)? (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3})"
| stats count AS tentativi by ip
| where tentativi > 5
| sort -tentativi
```

Questa query è stata costruita per identificare gli indirizzi IP che hanno tentato di accedere al sistema SSH senza successo **più di 5 volte**, evidenziando possibili attacchi brute-force o attività sospette.

1. **index=* source="secure.log" "Failed password"** > Vengono selezionati tutti gli eventi nel file secure.log che riportano un tentativo di accesso fallito via SSH.
2. **rex "Failed password for(invalid user)? (?<username>\w+) from (?<ip>\d{1,3}(?:\.\d{1,3}){3})"** > Utilizzando una regex, si estrae:
 - o **ip**: l'indirizzo IP da cui proviene il tentativo di login fallito
 - o **username**: il nome utente usato (non mostrato nel risultato ma utile in altri contesti)
3. **stats count AS tentativi by ip** > Questo comando conta il numero totale di eventi per ogni IP e salva il risultato nella colonna tentativi.
4. **where tentativi > 5** > Limita i risultati ai soli IP che hanno superato i 5 tentativi falliti.
5. **sort -tentativi** > Gli IP vengono ordinati in base al numero di tentativi, dal più alto al più basso.

La query fornisce un elenco degli indirizzi IP più aggressivi nei confronti del sistema, ordinati per numero di tentativi falliti. Questo tipo di analisi è fondamentale per attività di **threat hunting** e può supportare l'implementazione di misure difensive come il blocco degli IP sospetti tramite firewall o strumenti come **fail2ban**.



ip	tentativi
87.194.210.51	208
194.215.245.19	121
211.186.11.181	124
189.169.22.135	201
128.241.228.82	205
218.221.226.11	204
65.19.187.94	182
233.77.49.94	183
187.3.146.287	187
78.38.1.225	188
188.85.113.83	144
178.192.178.18	143
198.228.212.52	142
223.285.219.67	142
285.47.223.29	141
71.192.86.285	140
188.158.48.188	138
95.138.178.201	137
211.89.111.11	136
288.148.243.178	135

Figura 4 Risultato query n°4

Query n° 5 - Ricerca degli errori “Internal Server Error” (codice 500)

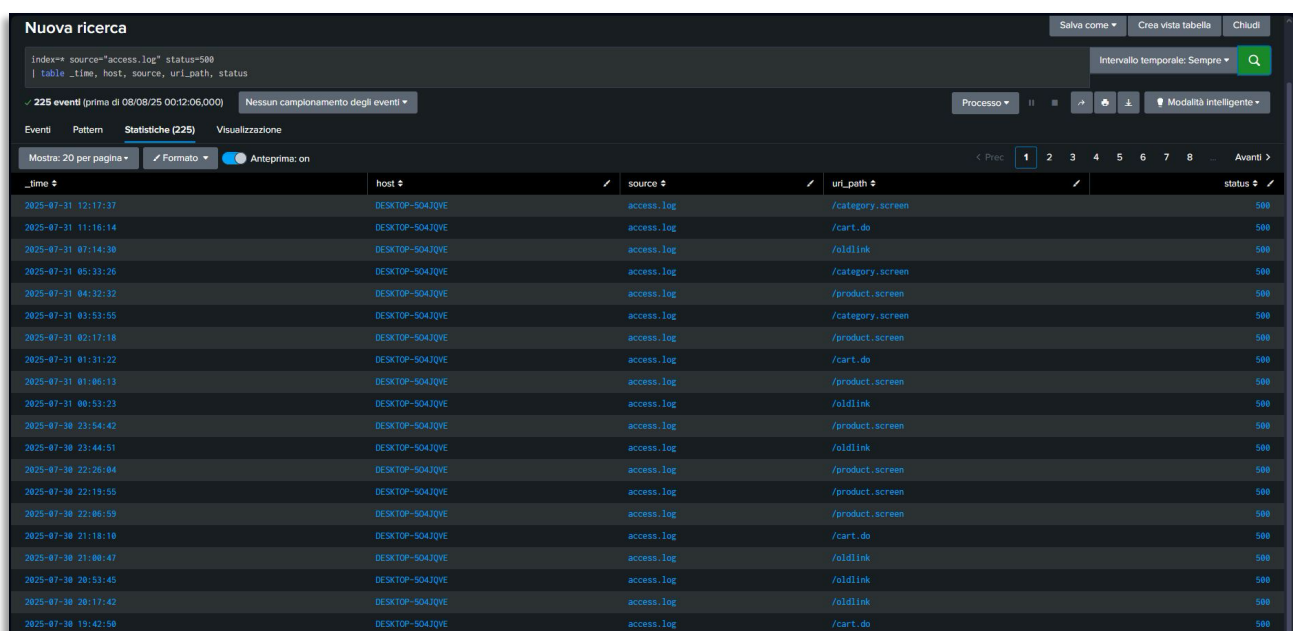
```
index=* source="access.log" status=500
| table _time, host, source, uri_path, status
```

Questa query è stata realizzata per identificare **tutti gli errori HTTP di tipo 500 Internal Server Error** all'interno dei file **access.log**. Questo tipo di errore indica che il server ha riscontrato una condizione imprevista che gli ha impedito di soddisfare la richiesta dell'utente.

1. **index=* source="access.log" status=500 >** Recupera tutti gli eventi nel file access.log in cui il campo status è uguale a 500, corrispondente a un errore di tipo “Internal Server Error”.
2. **| table _time host source uri_path status >** Mostra in formato tabellare:
 - **_time**: la data e ora in cui si è verificato l'errore;
 - **host**: il nome del dispositivo o server che ha generato l'evento;
 - **source**: la sorgente del log (in questo caso access.log);
 - **uri_path**: la risorsa (es. pagina o endpoint) che ha causato l'errore;
 - **status**: il codice HTTP dell'evento (500).

La query restituisce l'elenco dettagliato delle richieste che hanno generato un errore HTTP 500, indicando **quale pagina o funzione ha causato l'errore, quando, e su quale host**.

Questa analisi è utile per identificare problemi ricorrenti lato server, pagine non funzionanti o mal configurate, e per supportare il debugging e il miglioramento dell'infrastruttura web.



_time	host	source	uri_path	status
2025-07-31 12:17:37	DESKTOP-SO4JQE	access.log	/category.screen	500
2025-07-31 11:16:14	DESKTOP-SO4JQE	access.log	/cart.do	500
2025-07-31 07:14:30	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-31 05:33:26	DESKTOP-SO4JQE	access.log	/category.screen	500
2025-07-31 04:32:32	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-31 03:53:55	DESKTOP-SO4JQE	access.log	/category.screen	500
2025-07-31 02:17:18	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-31 01:31:22	DESKTOP-SO4JQE	access.log	/cart.do	500
2025-07-31 01:06:13	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-31 00:53:23	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-30 23:54:42	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-30 23:44:51	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-30 22:26:04	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-30 22:19:55	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-30 22:06:59	DESKTOP-SO4JQE	access.log	/product.screen	500
2025-07-30 21:18:10	DESKTOP-SO4JQE	access.log	/cart.do	500
2025-07-30 21:00:47	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-30 20:53:45	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-30 20:17:42	DESKTOP-SO4JQE	access.log	/oldlink	500
2025-07-30 19:42:50	DESKTOP-SO4JQE	access.log	/cart.do	500

Figura 5 Risultato query n°5

Conclusioni sull'analisi dei log con supporto AI

L'analisi condotta sui file di log `secure.log` e `access.log` tramite query Splunk ha permesso di identificare con precisione pattern e anomalie legate sia alla sicurezza del sistema che alla stabilità delle applicazioni web.

In particolare:

- Sono stati rilevati **numerosi tentativi di accesso SSH falliti**, provenienti da indirizzi IP specifici e in alcuni casi con frequenza elevata (>5 tentativi), evidenziando potenziali attacchi brute-force o attività di enumerazione utenti.
- È stato possibile isolare **le sessioni SSH riuscite** di utenti specifici, fornendo informazioni utili per il monitoraggio e l'audit degli accessi.
- L'analisi dei log web (`access.log`) ha individuato diversi errori **HTTP 500 – Internal Server Error**, permettendo di associare ogni evento alla risorsa richiesta e all'host coinvolto, facilitando così attività di debug e manutenzione.

L'integrazione di strumenti di **Intelligenza Artificiale** nel processo di analisi ha potenziato la capacità di individuare correlazioni e tendenze nei dati.

Attraverso algoritmi di rilevamento automatico di anomalie e clustering comportamentale, l'AI può:

- Identificare schemi ricorrenti nei tentativi di accesso malevoli;
- Prevedere potenziali escalation di attacco sulla base di trend storici;
- Fornire alert proattivi per mitigare tempestivamente le minacce.

Questi risultati dimostrano come l'uso combinato di **Splunk** e **AI** consenta non solo di effettuare indagini retrospettive sui log, ma anche di migliorare la postura di sicurezza, passando da un approccio reattivo a uno **proattivo e predittivo**.