

# BENCHMARK W24D4 - Domenico Vecchio

## Progetto:

Esercizio Progetto Importate su Splunk i dati di esempio "tutorialdata.zip":

- Crea una query Splunk per identificare tutti i tentativi di accesso falliti "Failed password". La query dovrebbe mostrare il timestamp, l'indirizzo IP di origine, il nome utente e il motivo del fallimento.
- Scrivi una query Splunk per trovare tutte le sessioni SSH aperte con successo. La query dovrebbe filtrare per l'utente "djohnson" e mostrare il timestamp e l'ID utente.
- Scrivi una query Splunk per trovare tutti i tentativi di accesso falliti provenienti dall'indirizzo IP "86.212.199.60". La query dovrebbe mostrare il timestamp, il nome utente e il numero di porta.
- Crea una query Splunk per identificare gli indirizzi IP che hanno tentato di accedere ("Failed password") al sistema più di 5 volte. La query dovrebbe mostrare l'indirizzo IP e il numero di tentativi.
- Crea una query Splunk per trovare tutti gli Internal Server Error.

Trarre delle conclusioni sui log analizzati utilizzando AI.

## Esecuzione dell'esercizio :

### Punto 1 :

#### Query eseguita :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Failed password"  
| rex "Failed password for (?<reason>invalid user)?\s*(?<user>\S+) from (?<src_ip>\S+) port  
(?<port>\d+)"  
| table _time src_ip user reason port
```

#### Spiegazione :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Failed password"
```

- Questo è il filtro iniziale: limita la ricerca agli eventi caricati dal file tutorialdata.zip sul host=Dominik-PC e che contengono la stringa **Failed password** (ossia messaggi di login SSH fallito).

```
| rex "Failed password for (?<reason>invalid user)?\s*(?<user>\S+) from (?<src_ip>\S+) port  
(?<port>\d+)"
```

- rex usa una regular expression per **estrarre campi** dal testo dell'evento.
- La regex contiene **gruppi nominati** (?<nomecampo>pattern):
  - (?<reason>invalid user)? → cerca letteralmente la stringa invalid user e la salva in reason. Il ? dopo la parentesi significa che il gruppo è **opzionale** (alcuni messaggi hanno invalid user X, altri hanno solo for root senza la parola invalid).
  - \s\* → matcha eventuali spazi dopo invalid user.
  - (?<user>\S+) → cattura il nome utente (una sequenza di caratteri non spazi).
  - from (?<src\_ip>\S+) → cattura l'IP sorgente (campo src\_ip).
  - port (?<port>\d+) → cattura il numero di porta (solo cifre).

| table \_time src\_ip user reason port

- Mostra in output solo le colonne richieste:
  - \_time → timestamp già interpretato da Splunk.
  - src\_ip, user, reason, port → i campi estratti con rex.

Nuova ricerca

source="tutorialdata.zip:\*" host="Dominik-PC" "failed password"  
 | rex "failed password for (?<reason>invalid user)?\s\*(?<user>\S+) from (?<src\_ip>\S+) port (?<port>\d+)"  
 | table \_time src\_ip user reason port

Intervallo temporale: Sempre

33.253 eventi (prima di 12/08/25 12:25:58,000) Nessun campionamento degli eventi

Processo

Modaltà intelligente

Eventi Pattern Statistiche (33.253) Visualizzazione

Mostra: 20 per pagina Formato Antepriima: on

_time	src_ip	user	reason	port
2025-08-09 06:38:23	123.30.108.208	jagger	invalid user	1454
2025-08-09 06:38:23	123.30.108.208	mailman	invalid user	4988
2025-08-09 06:38:23	123.30.108.208	henri	invalid user	2359
2025-08-09 06:38:23	123.30.108.208	root		2295
2025-08-09 06:38:23	128.241.228.82	helpdesk	invalid user	3210
2025-08-09 06:38:23	128.241.228.82	root		3117
2025-08-09 06:38:23	128.241.228.82	irc	invalid user	3428
2025-08-09 06:38:23	128.241.228.82	services	invalid user	2361
2025-08-09 06:38:23	128.241.228.82	prince		2481
2025-08-09 06:38:23	128.241.228.82	root		1078
2025-08-09 06:38:23	128.241.228.82	sys	invalid user	2687
2025-08-09 06:38:23	128.241.228.82	sunny	invalid user	4358
2025-08-09 06:38:23	128.241.228.82	icinga	invalid user	1547
2025-08-09 06:38:23	128.241.228.82	email	invalid user	2794

La query 1 ha restituito esattamente quello che chiedeva l'esercizio:

- **\_time** → il timestamp dell'evento
- **src\_ip** → l'indirizzo IP di origine
- **user** → il nome utente usato nel tentativo
- **reason** → il motivo del fallimento (invalid user quando presente, vuoto quando non c'è)
- **port** → il numero di porta usato per il tentativo

## Punto 2 :

### Query eseguita :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Accepted password" "djohnson"  
| rex "Accepted password for (?<user>\S+) from (?<src_ip>\S+) port (?<port>\d+)"  
| table _time user src_ip port
```

### Spiegazione :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Accepted password" "djohnson"
```

Filtra i log provenienti dal file tutorialdata.zip sul host=Dominik-PC.  
Cerca solo eventi SSH con **accesso riuscito** ("Accepted password").  
Limita agli eventi che contengono il nome utente **djohnson**.

```
| rex "Accepted password for (?<user>\S+) from (?<src_ip>\S+) port (?<port>\d+)"
```

(?<user>\S+) → cattura il nome utente (qui sarà sempre djohnson).

(?<src\_ip>\S+) → cattura l'indirizzo IP sorgente della connessione.

(?<port>\d+) → cattura il numero della porta SSH utilizzata.

## Visualizzazione in tabella

Nuova ricerca

Salva come Crea vista tabella Chiudi

```
source=tutorialdata.zip:* host="Dominik-PC" "Accepted password" "djohnson"
| rex "Accepted password for (?<user>\S+) from (?<src_ip>\S+) port (?<port>\d+)"
| table _time user src_ip port
```

Intervallo temporale: Sempre

955 eventi (prima di 12/08/25 14:12:11,000) Nessun campionamento degli eventi

Processo

Eventi Pattern Statistiche (955) Visualizzazione

Mostra: 20 per pagina Formato Antepriema: on

< Prec

1 2 3 4 5 6 7 8 ...

Avanti >

_time	user	src_ip	port
2025-08-11 06:38:23	djohnson	10.3.10.46	2898
2025-08-11 06:38:23	djohnson	10.3.10.46	9913
2025-08-11 06:38:23	djohnson	10.3.10.46	5104
2025-08-11 06:38:23	djohnson	10.3.10.46	5279
2025-08-11 06:38:23	djohnson	10.3.10.46	8621
2025-08-11 06:38:23	djohnson	10.3.10.46	2459
2025-08-10 06:38:23	djohnson	10.3.10.46	5231
2025-08-10 06:38:23	djohnson	10.3.10.46	8733
2025-08-10 06:38:23	djohnson	10.3.10.46	9788
2025-08-10 06:38:23	djohnson	10.3.10.46	3684
2025-08-10 06:38:23	djohnson	10.3.10.46	7764
2025-08-10 06:38:23	djohnson	10.3.10.46	2506
2025-08-10 06:38:23	djohnson	10.3.10.46	4949
2025-08-10 06:38:23	djohnson	10.3.10.46	7385
2025-08-10 06:38:23	djohnson	10.3.10.46	5500

La query 2 ha restituito esattamente quello che chiedeva l'esercizio:

- **\_time** → data e ora dell'accesso riuscito.
- **src\_ip** → l'indirizzo IP di origine
- **user** → nome utente (djohnson).
- **port** → il numero di porta usato per il tentativo

## Punto 3 :

### Query eseguita :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Failed password" "86.212.199.60"  
| rex "Failed password for (?<user>\S+) from 86\.212\.199\.60 port (?<port>\d+)"  
| table _time user port
```

### Spiegazione :

source="tutorialdata.zip:\*" host="Dominik-PC" "Failed password" "86.212.199.60"

Usa solo i log del file tutorialdata.zip e dell'host Dominik-PC.  
Cerca solo eventi di login fallito che contengono l'IP 86.212.199.60.

| rex "Failed password for (?<user>\S+) from 86\.212\.199\.60 port (?<port>\d+)"

(?<user>\S+) → cattura il nome utente usato nel tentativo.

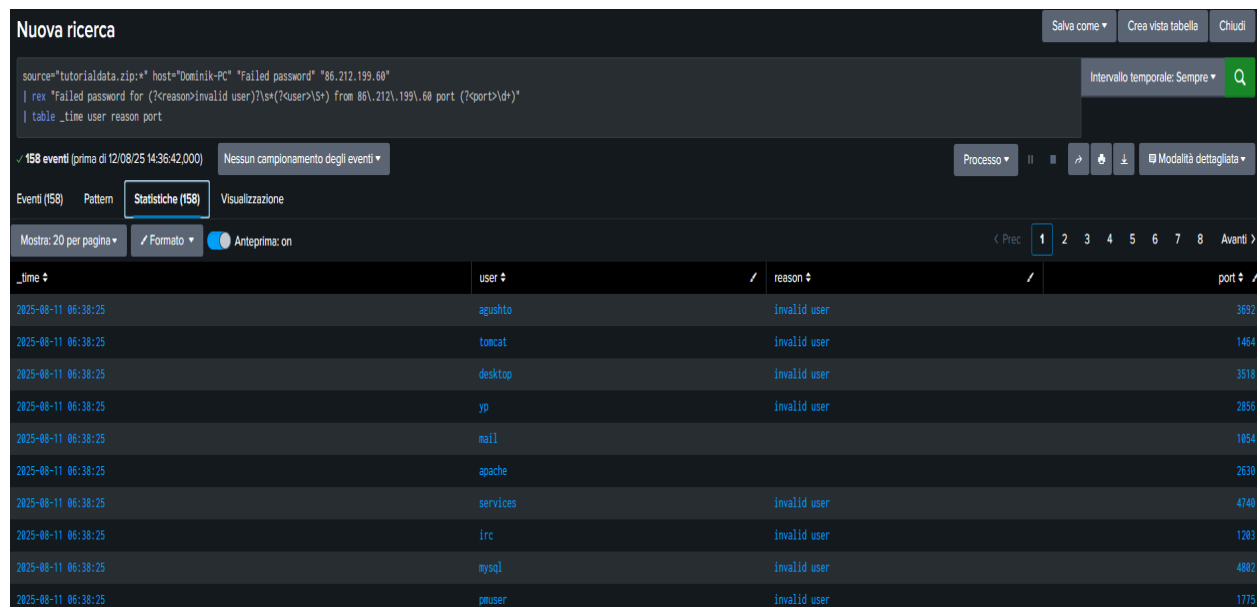
(?<port>\d+) → cattura il numero della porta usata.

86\.212\.199\.60 → l'IP è scritto in regex, quindi i punti sono \..

| table \_time user port

Mostra solo i tre campi richiesti: timestamp, nome utente, numero porta.

### Visualizzazione in tabella :



The screenshot shows the Splunk search results interface. At the top, the search bar contains the query: `source="tutorialdata.zip:*" host="Dominik-PC" "Failed password" "86.212.199.60" | rex "Failed password for (?<reason>invalid user)?\s+(?<user>\S+) from 86\.212\.199\.60 port (?<port>\d+)" | table _time user reason port`. The results are displayed in a table with 158 events. The table has four columns: `_time`, `user`, `reason`, and `port`. The data shows multiple failed login attempts for various users (agushto, tomcat, desktop, yp, mail, apache, services, irc, mysql, pmuser) from the IP 86.212.199.60 on 2025-08-11 at 06:38:25. The reasons for failure are all "invalid user".

_time	user	reason	port
2025-08-11 06:38:25	agushto	invalid user	3692
2025-08-11 06:38:25	tomcat	invalid user	1464
2025-08-11 06:38:25	desktop	invalid user	3518
2025-08-11 06:38:25	yp	invalid user	2856
2025-08-11 06:38:25	mail		1054
2025-08-11 06:38:25	apache		2630
2025-08-11 06:38:25	services	invalid user	4740
2025-08-11 06:38:25	irc	invalid user	1203
2025-08-11 06:38:25	mysql	invalid user	4802
2025-08-11 06:38:25	pmuser	invalid user	1775

La query 3 ha restituito esattamente quello che chiedeva l'esercizio:

- **\_time** → data e ora dell'accesso riuscito.
- **src\_ip** → l'indirizzo IP di origine
- **user** → prende l'utente anche se preceduto da invalid user.
- **port** → rimane sempre estratto correttamente.

Dai log risulta che l'IP 86.212.199.60 ha tentato l'accesso SSH più volte, sia con utenti validi (root, sync, mail) che con utenti inesistenti, usando porte varie (1695, 3683, ecc.). Questo comportamento è compatibile con un attacco di tipo **brute force**.

## Punto 4 :

### Query eseguita :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Failed password"
| rex "Failed password for .* from (?<src_ip>\S+)"
| stats count AS total_attempts by src_ip
| where total_attempts > 5
| sort - total_attempts
```

### Spiegazione :

```
source="tutorialdata.zip:*" host="Dominik-PC" "Failed password"
```

Cerca solo eventi di login fallito nei log caricati.

```
| rex "Failed password for .* from (?<src_ip>\S+)"
```

Cattura l'indirizzo IP dopo la parola from.

```
| stats count AS total_attempts by src_ip
```

Conta quante volte ogni IP compare nei tentativi falliti.

```
| where total_attempts > 5
```

Mostra solo gli IP che hanno effettuato più di 5 tentativi.

| sort - total\_attempts

Ordina in ordine decrescente di numero di tentativi.

**Visualizzazione in tabella :**

Nuova ricerca

source="tutorialdata.zip:\*" host="Dominik-PC" "Failed password"

| rex "Failed password for .\* from (?<src\_ip>S+)"

| stats count AS total\_attempts by src\_ip

| where total\_attempts > 5

| sort - total\_attempts

Intervallo temporale: Sempre

Salva come Crea vista tabella Chiudi

33.253 eventi (prima di 12/08/25 15:08:56,000)

Nessun campionamento degli eventi

Processo

Modalità dettagliata

Eventi (33.253) Pattern Statistiche (182) Visualizzazione

Mostra: 20 per pagina Formato Anteprima: on

< Prec 1 2 3 4 5 6 7 8 ... Avanti >

src_ip	total_attempts
87.194.216.51	948
211.166.11.101	743
128.241.220.82	622
109.169.32.135	515
194.215.205.19	514
216.221.226.11	433
188.138.40.166	297
65.19.167.94	286
107.3.146.207	282
95.130.170.231	279
223.205.219.67	274
27.1.11.11	273
27.35.11.11	270
59.162.167.100	267

La query 4 ha restituito esattamente quello che chiedeva l'esercizio:

L'IP **87.194.216.51** ha effettuato **948 tentativi falliti**, un numero estremamente alto, tipico di un attacco di tipo brute force.

Anche altri IP (211.166.11.101, 128.241.220.82, ecc.) mostrano centinaia di tentativi indicazione di scansioni automatizzate.

## Punto 5 :

### Query eseguita :

```
source="tutorialdata.zip:*" host="Dominik-PC" status=500  
| table _time clientip method uri_path status
```

### Spiegazione :

```
source="tutorialdata.zip:*" host="Dominik-PC" status=500
```

Cerca solo eventi nei log caricati dal file tutorialdata.zip sull'host Dominik-PC.  
status=500 filtra le richieste HTTP che hanno restituito codice **500**, ovvero **Internal Server Error**.

```
| table _time clientip method uri_path status
```

\_time → timestamp dell'evento.

clientip → indirizzo IP del client che ha generato la richiesta.

method → metodo HTTP usato (es. GET, POST).

uri\_path → percorso della risorsa richiesta.

status → codice HTTP restituito (500).

### Visualizzazione in tabella :

Nuova ricerca

Salva come Crea vista tabella Chiudi

source="tutorialdata.zip:\*" host="Dominik-PC" status=500  
| table \_time clientip method uri\_path status

Intervallo temporale: Sempre

733 eventi (prima di 12/08/25 15:45:07,000) Nessun campionamento degli eventi

Processo

Eventi (733) Pattern Statistiche (733) Visualizzazione

Mostra: 20 per pagina Formato Anteprima: on

< Prec 1 2 3 4 5 6 7 8 ... Avanti >

_time	clientip	method	uri_path	status
2025-08-11 18:18:59	198.35.1.75	GET	/cart.do	500
2025-08-11 18:18:55	198.35.1.75	GET	/product.screen	500
2025-08-11 17:42:03	125.89.78.6	POST	/cart.do	500
2025-08-11 17:17:00	194.146.236.22	POST	/product.screen	500
2025-08-11 17:15:13	121.254.179.199	POST	/product.screen	500
2025-08-11 16:54:07	76.89.103.115	GET	/cart.do	500
2025-08-11 16:33:02	59.36.99.78	GET	/oldlink	500
2025-08-11 16:05:47	201.42.223.29	GET	/category.screen	500
2025-08-11 15:44:56	188.173.152.100	GET	/category.screen	500
2025-08-11 15:26:14	187.231.45.62	POST	/product.screen	500
2025-08-11 15:09:37	92.46.53.223	GET	/category.screen	500



La query 5 ha restituito esattamente quello che chiedeva l'esercizio:

Gli errori 500 si verificano su più endpoint, in particolare:

- /cart.do (sia GET che POST) → possibile malfunzionamento della gestione del carrello.
- /product.screen → problemi nella visualizzazione o gestione dei prodotti.
- /oldlink → probabile URL obsoleto o non più supportato.

Più indirizzi IP coinvolti → non è un problema legato a un singolo utente, ma un'anomalia lato server.

Alcuni errori avvengono con richieste **POST** → potrebbero essere causati da input non validi o malformati.

## Conclusioni AI sui log analizzati

L'analisi dei log provenienti da tutorialdata.zip evidenzia **due macro-aree di criticità**: sicurezza SSH e stabilità dell'applicazione web.

### 1 – Sicurezza SSH

- **Attività anomala di brute force:**
  - Molti indirizzi IP hanno effettuato centinaia di tentativi falliti, con picchi oltre le 900 connessioni da un singolo IP (87.194.216.51).
  - I tentativi includono utenti inesistenti (invalid user) e utenti privilegiati (root), indicando attacchi di dizionario mirati.
- **Utente djohnson sotto osservazione:**
  - Login riusciti multipli dallo stesso IP (10.3.10.46) in intervalli ravvicinati, con porte sorgente variabili.
  - Possibile accesso legittimo automatizzato **oppure** compromissione con uso di script.

- **Indirizzo IP mirato (86.212.199.60):**

- Tentativi falliti con utenti sia validi sia fittizi → tipico comportamento di enumerazione utenti prima di un brute force.

**AI Insight:** il pattern temporale dei tentativi falliti seguito da connessioni riuscite, se proveniente dallo stesso IP, può indicare un attacco che ha avuto successo. Andrebbe verificata la correlazione temporale tra i log dei punti 1, 2 e 3.

## **2 – Stabilità applicativa e possibili exploit**

- **Errori HTTP 500 ricorrenti** su endpoint critici:

- /cart.do e /product.screen → collegati a funzionalità di e-commerce, quindi potenzialmente impattanti sulle vendite o sulla user experience.
- /oldlink → potrebbe essere un link deprecato, ma il fatto che generi errori può esporre informazioni sull'applicazione.

- Gli IP sorgente variano, includendo sia indirizzi potenzialmente legittimi sia possibili fonti malevole.
- Alcuni errori 500 derivano da richieste **POST**, che possono essere veicolo per attacchi come SQL injection o Command injection se l'input non è correttamente validato.

**AI Insight:** la sovrapposizione temporale tra tentativi SSH falliti e picchi di errori HTTP 500 può indicare **attacchi coordinati** volti a individuare vulnerabilità multiple nello stesso arco temporale.

## **3 – Rischio complessivo**

**Livello di rischio SSH:** Alto → tentativi costanti e massivi da più IP distribuiti globalmente.

**Livello di rischio Web:** Medio-Alto → errori 500 su endpoint critici con metodi POST richiedono indagine immediata.

**Possibile minaccia combinata:** alcuni IP potrebbero essere coinvolti sia in tentativi SSH che in richieste web malevole.

#### 4 – Raccomandazioni AI

1. **Mitigazione brute force:** blocco IP dopo N tentativi falliti, rate limiting su porta SSH, uso di chiavi invece delle password.
2. **Hardening SSH:** disabilitare accesso root, cambiare porta SSH, abilitare 2FA.
3. **Analisi endpoint web:** logging dettagliato input/output su /cart.do e /product.screen, revisione codice e fix bug.
4. **Correlazione log:** implementare ricerca automatizzata che unisce log SSH e HTTP per identificare IP coinvolti in più tipologie di attacco.
5. **Alerting AI-driven:** configurare regole che sfruttano pattern storici per segnalare attività anomale in tempo reale.