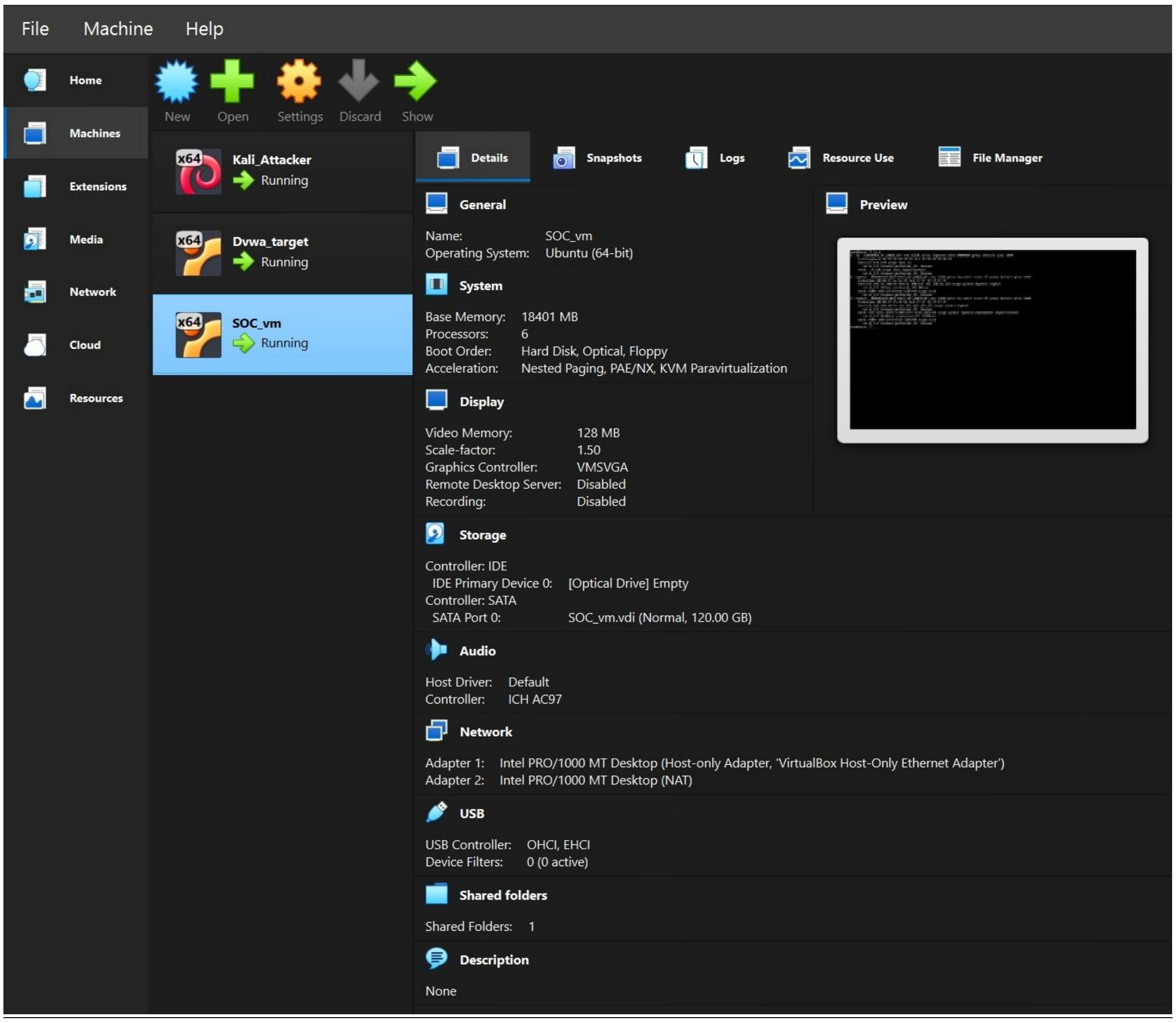
WORKFLOW

virtual box with all thre machines running :



Kali_Attacker
Running

Dvwa_target
Running

SOC_vm
Running

General

Name: SOC_vm
Operating System: Ubuntu (64-bit)

System

Base Memory: 18401 MB
Processors: 6
Boot Order: Hard Disk, Optical, Floppy
Acceleration: Nested Paging, PAE/NX, KVM Paravirtualization

Display

Video Memory: 128 MB
Scale-factor: 1.50
Graphics Controller: VMSVGA
Remote Desktop Server: Disabled
Recording: Disabled

Storage

Controller: IDE
IDE Primary Device 0: [Optical Drive] Empty
Controller: SATA
SATA Port 0: SOC_vm.vdi (Normal, 120.00 GB)

Audio

Host Driver: Default
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (Host-only Adapter, 'VirtualBox Host-Only Ethernet Adapter')
Adapter 2: Intel PRO/1000 MT Desktop (NAT)

USB

USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

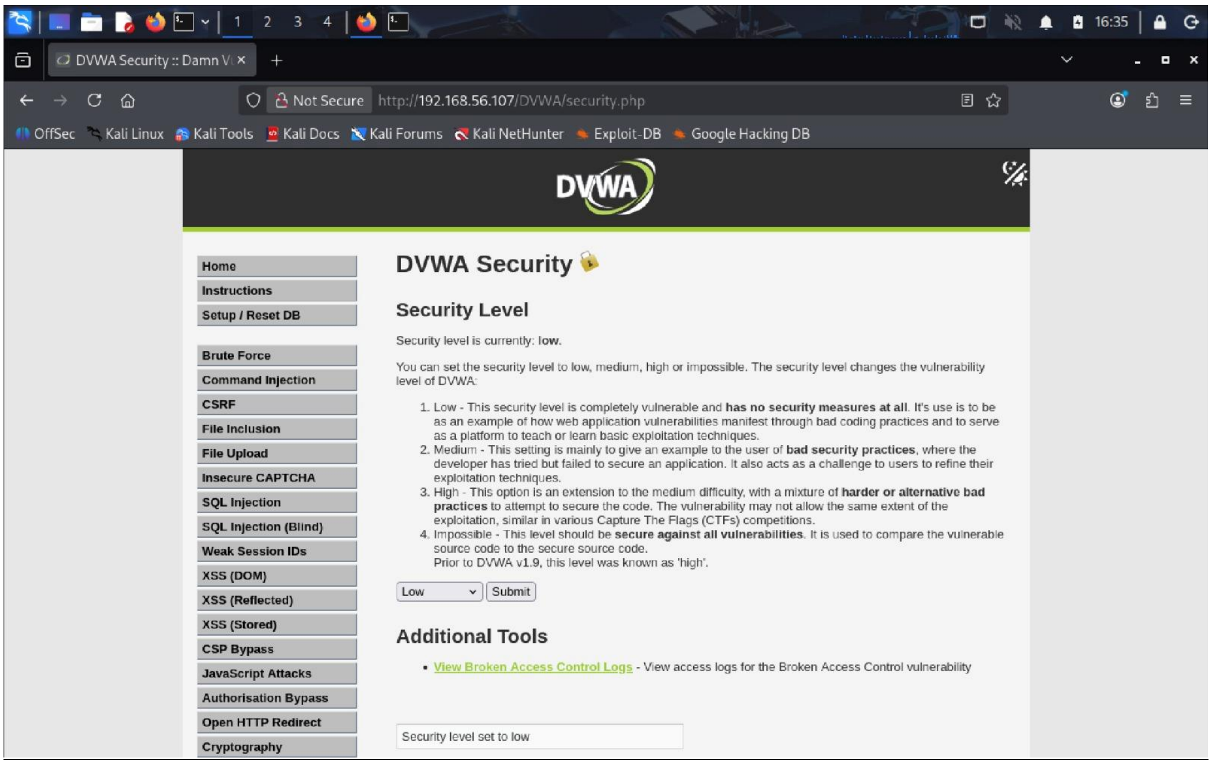
Shared folders

Shared Folders: 1

Description

None

2. DVWA web application loaded and security level set to low



DVWA Security

Security Level

Security level is currently: **low**.

You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:

1. Low - This security level is completely vulnerable and **has no security measures at all**. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of **bad security practices**, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of **harder or alternative bad practices** to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be **secure against all vulnerabilities**. It is used to compare the vulnerable source code to the secure source code.
   Prior to DVWA v1.9, this level was known as 'high'.

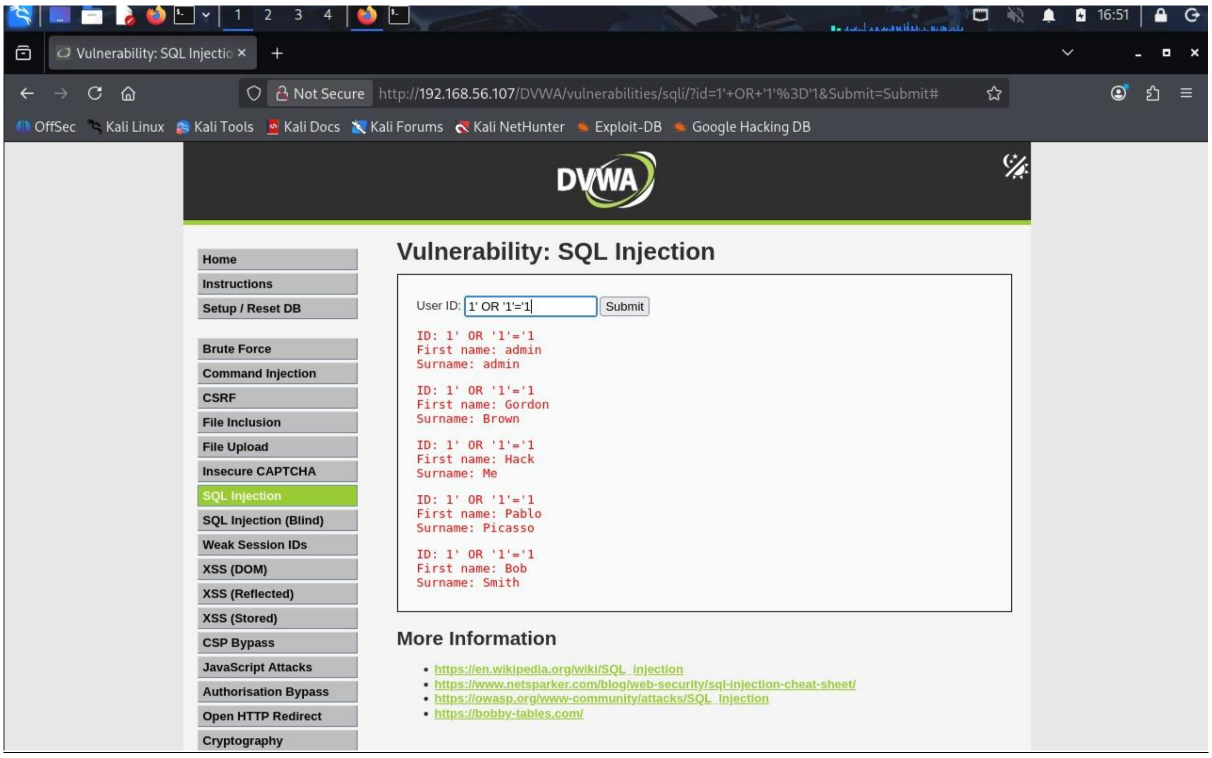
Low Submit

Additional Tools

- View Broken Access Control Logs - View access logs for the Broken Access Control vulnerability

Security level set to low

proves vulnerable target is intentionally exposed



Vulnerability: SQL Injection

User ID: 1' OR '1'='1 Submit

```
ID: 1' OR '1'='1
First name: admin
Surname: admin

ID: 1' OR '1'='1
First name: Gordon
Surname: Brown

ID: 1' OR '1'='1
First name: Hack
Surname: Me

ID: 1' OR '1'='1
First name: Pablo
Surname: Picasso

ID: 1' OR '1'='1
First name: Bob
Surname: Smith
```

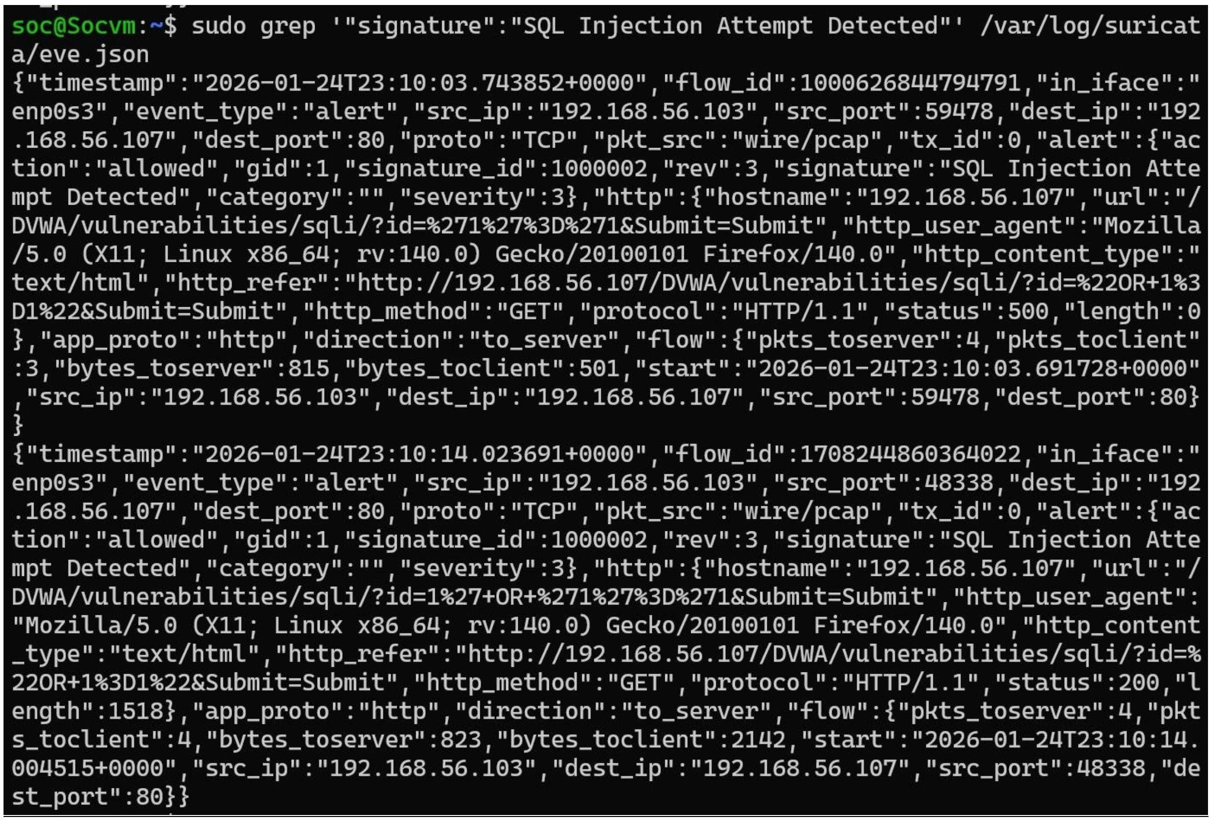
More Information

- https://en.wikipedia.org/wiki/SQL_injection
- https://www.netsparker.com/blog/web-security/sql-injection-cheat-sheet/
- https://owasp.org/www-community/attacks/SQL_Injection
- https://bobby-tables.com/

DVWA showing dumped database records

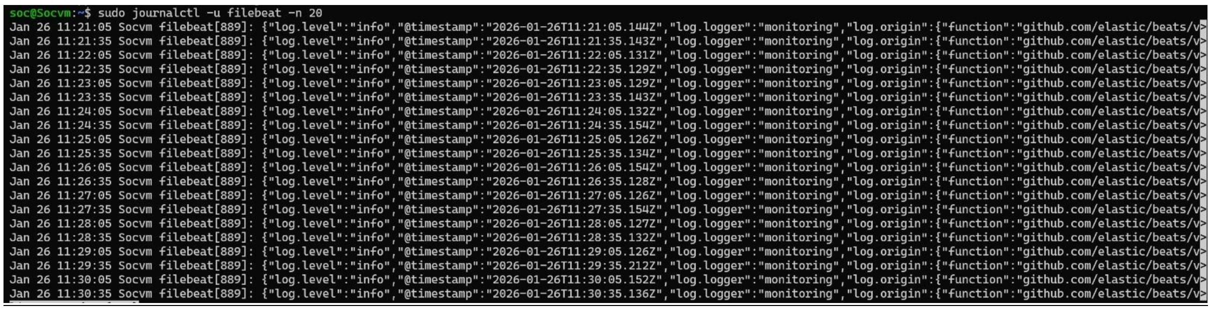
Purpose: Proves successful sql attack execution

<u>“This is the raw Suricata IDS alert generated in JSON format. It confirms SQL injection detection at the sensor level.” :</u>



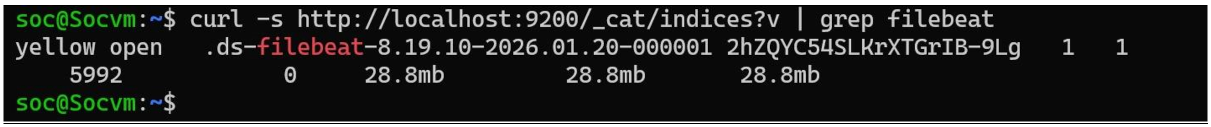
```
soc@Socvm:~$ sudo grep '"signature":"SQL Injection Attempt Detected"' /var/log/suricat
a/eve.json
{"timestamp":"2026-01-24T23:10:03.743852+0000","flow_id":1000626844794791,"in_iface":"
enp0s3","event_type":"alert","src_ip":"192.168.56.103","src_port":59478,"dest_ip":"192
.168.56.107","dest_port":80,"proto":"TCP","pkt_src":"wire/pcap","tx_id":0,"alert":{"ac
tion":"allowed","gid":1,"signature_id":1000002,"rev":3,"signature":"SQL Injection Atte
mpt Detected","category":"","severity":3},"http":{"hostname":"192.168.56.107","url":"/
DVWA/vulnerabilities/sqli/?id=%271%27%3D%271&Submit=Submit","http_user_agent":"Mozilla
/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0","http_content_type":"
text/html","http_refer":"http://192.168.56.107/DVWA/vulnerabilities/sqli/?id=%22OR+1%3
D1%22&Submit=Submit","http_method":"GET","protocol":"HTTP/1.1","status":500,"length":0
},"app_proto":"http","direction":"to_server","flow":{"pkts_toserver":4,"pkts_toclient"
:3,"bytes_toserver":815,"bytes_toclient":501,"start":"2026-01-24T23:10:03.691728+0000"
,"src_ip":"192.168.56.103","dest_ip":"192.168.56.107","src_port":59478,"dest_port":80}
}
{"timestamp":"2026-01-24T23:10:14.023691+0000","flow_id":1708244860364022,"in_iface":"
enp0s3","event_type":"alert","src_ip":"192.168.56.103","src_port":48338,"dest_ip":"192
.168.56.107","dest_port":80,"proto":"TCP","pkt_src":"wire/pcap","tx_id":0,"alert":{"ac
tion":"allowed","gid":1,"signature_id":1000002,"rev":3,"signature":"SQL Injection Atte
mpt Detected","category":"","severity":3},"http":{"hostname":"192.168.56.107","url":"/
DVWA/vulnerabilities/sqli/?id=1%27+OR+%271%27%3D%271&Submit=Submit","http_user_agent":
"Mozilla/5.0 (X11; Linux x86_64; rv:140.0) Gecko/20100101 Firefox/140.0","http_content
_type":"text/html","http_refer":"http://192.168.56.107/DVWA/vulnerabilities/sqli/?id=%
22OR+1%3D1%22&Submit=Submit","http_method":"GET","protocol":"HTTP/1.1","status":200,"l
ength":1518},"app_proto":"http","direction":"to_server","flow":{"pkts_toserver":4,"pkt
s_toclient":4,"bytes_toserver":823,"bytes_toclient":2142,"start":"2026-01-24T23:10:14.
004515+0000","src_ip":"192.168.56.103","dest_ip":"192.168.56.107","src_port":48338,"de
st_port":80}}
```

<u>filebeat shipping logs from suricata to elasticsearch:</u>



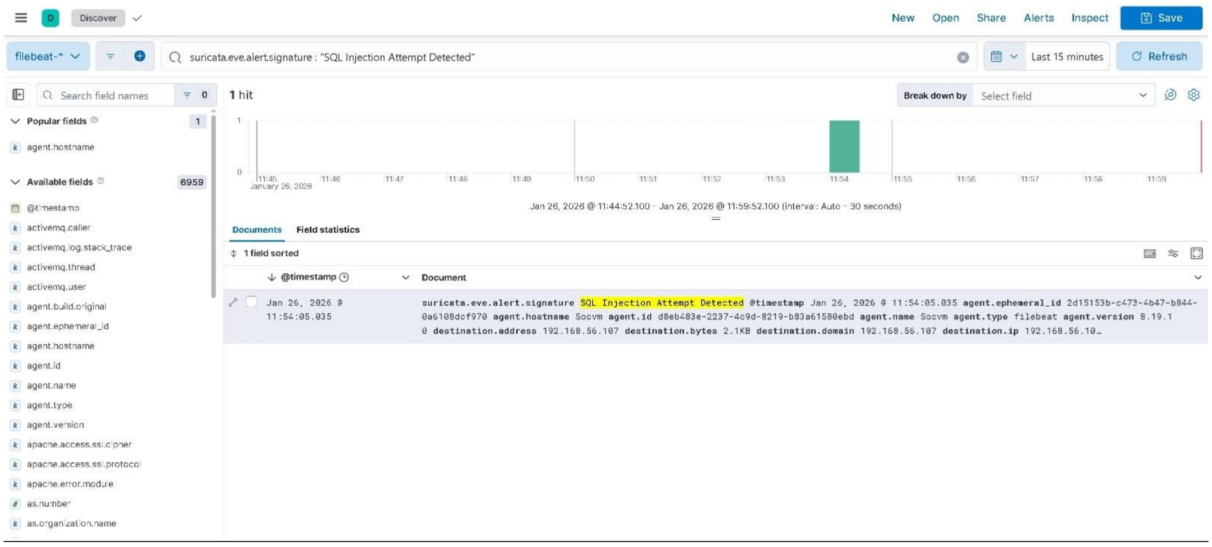
```
soc@Socvm:~$ sudo journalctl -u filebeat -n 20
Jan 26 11:21:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:21:05.144Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:21:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:21:35.143Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:22:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:22:05.131Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:22:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:22:35.129Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:23:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:23:05.129Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:23:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:23:35.143Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:24:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:24:05.132Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:24:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:24:35.154Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:25:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:25:05.126Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:25:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:25:35.134Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:26:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:26:05.154Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:26:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:26:35.128Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:27:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:27:05.126Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:27:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:27:35.154Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:28:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:28:05.127Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:28:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:28:35.132Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:29:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:29:05.126Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:29:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:29:35.212Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:30:05 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:30:05.152Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
Jan 26 11:30:35 Socvm filebeat[889]: {"log.level":"info","@timestamp":"2026-01-26T11:30:35.136Z","log.logger":"monitoring","log.origin":{"function":"github.com/elastic/beats/v>
```

<u>Logs indexed in Elasticsearch :</u>



```
soc@Socvm:~$ curl -s http://localhost:9200/_cat/indices?v | grep filebeat
yellow open   .ds-filebeat-8.19.10-2026.01.20-000001 2hZQYC54SLKrXTGrIB-9Lg   1   1
    5992            0     28.8mb          28.8mb        28.8mb
soc@Socvm:~$
```

Alert Verification and Evidence Collection :

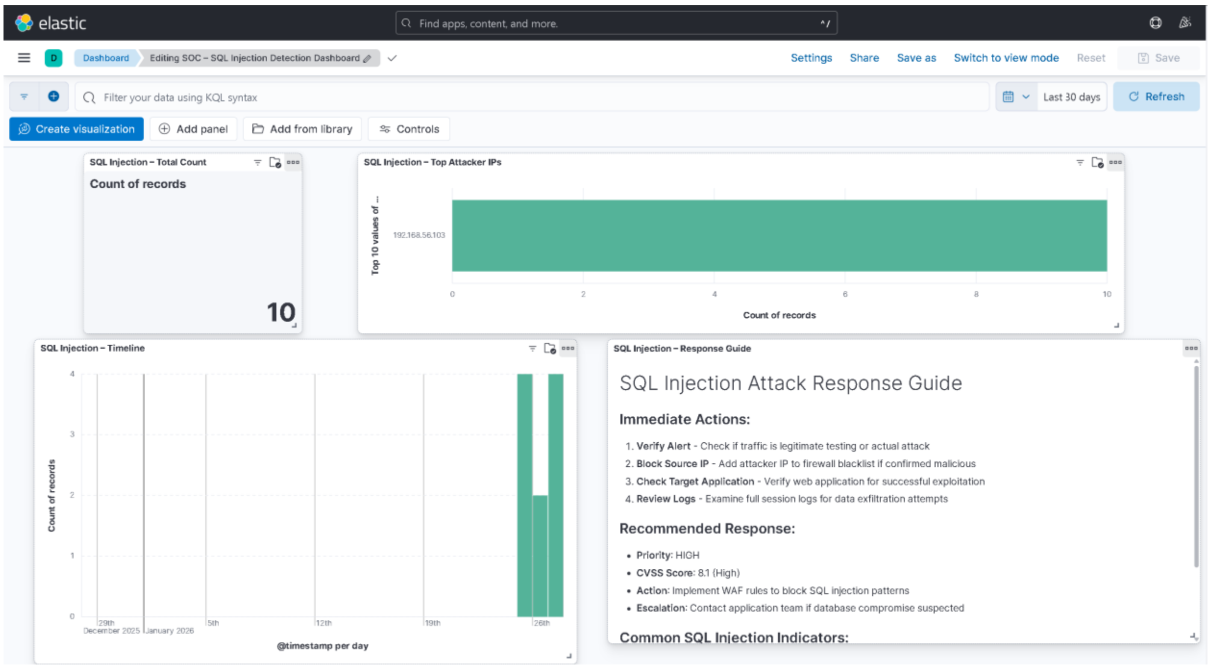


suricata.eve.alert.signature : "SQL Injection Attempt Detected"

1 hit

Jan 26, 2026 @ 11:44:52.100 - Jan 26, 2026 @ 11:59:52.100 (interval: Auto - 30 seconds)

```
@timestamp                Document
Jan 26, 2026 @            suricata.eve.alert.signature SQL Injection Attempt Detected @timestamp Jan 26, 2026 @ 11:54:05.035 agent.ephemeral_id 2d15153b-c473-4b47-b844-
11:54:05.035              0a6108dcf970 agent.hostname Socvm agent.id d8eb483e-2237-4c9d-8219-b83a61580ebd agent.name Socvm agent.type filebeat agent.version 8.19.1
                          0 destination.address 192.168.56.107 destination.bytes 2.1KB destination.domain 192.168.56.107 destination.ip 192.168.56.10…
```



SQL Injection Attack Response Guide

Immediate Actions:

1. **Verify Alert** - Check if traffic is legitimate testing or actual attack
2. **Block Source IP** - Add attacker IP to firewall blacklist if confirmed malicious
3. **Check Target Application** - Verify web application for successful exploitation
4. **Review Logs** - Examine full session logs for data exfiltration attempts

Recommended Response:

- **Priority**: HIGH
- **CVSS Score**: 8.1 (High)
- **Action**: Implement WAF rules to block SQL injection patterns
- **Escalation**: Contact application team if database compromise suspected

Common SQL Injection Indicators:

Each detected SQL injection attempt generated a structured alert within Suricata's eve.json log file. These alerts contained contextual metadata including source IP address, destination IP address, HTTP request parameters, and timestamp. A representative alert record was examined in Kibana's Discover interface to validate the integrity and completeness of the detection pipeline. This step confirms that attack telemetry is preserved end-to-end and is suitable for forensic analysis and incident response workflows.