

SQL Injection

Lucas Gabriel Oliveira Lima - 231003406

Vinicius da Silva Araujo - 221001981

July 2025

Resumo

Este relatório apresenta uma simulação de ataque de SQL Injection em ambiente controlado, utilizando infraestrutura provisionada na AWS via Terraform. São abordados os conceitos do ataque, a arquitetura da simulação, o uso da ferramenta *sqlmap* para automação do ataque e mecanismos de detecção como Snort e Wazuh. O objetivo é demonstrar vulnerabilidades comuns em aplicações web e estratégias para mitigação e monitoramento.

1 Introdução

A segurança de aplicações web é um tema de grande relevância na área de tecnologia da informação, especialmente diante do crescente número de ataques cibernéticos que exploram vulnerabilidades em sistemas de gerenciamento de banco de dados. Um dos ataques mais comuns e perigosos é o *SQL Injection*, que consiste na inserção maliciosa de comandos SQL em campos de entrada de aplicações, com o objetivo de manipular ou acessar dados sensíveis de forma não autorizada.

Este relatório apresenta uma simulação prática de um ataque de SQL Injection, utilizando uma infraestrutura automatizada para provisionamento de servidores, configuração de banco de dados e aplicação web vulnerável. O ambiente foi desenvolvido com o objetivo de demonstrar, de forma controlada, como a exploração dessa vulnerabilidade pode comprometer a integridade e a confidencialidade dos dados, além de ressaltar a importância da adoção de boas práticas de desenvolvimento seguro. O código-fonte do provisionamento da infraestrutura, e dos setups do webserver e do atacante estão disponíveis no repositório <https://github.com/Vini-ara/Sql-Injection-Simulation>.

Ao longo do relatório, serão abordados os conceitos fundamentais do ataque, a arquitetura da solução implementada, os scripts utilizados para automação do ambiente e os resultados obtidos durante a simulação. O objetivo é proporcionar uma compreensão clara dos riscos associados ao SQL Injection e das medidas necessárias para mitigar esse tipo de ameaça. Será mostrado também, os resultados que obtivemos a cada etapa da realização do experimento.

2 Infraestrutura da Simulação

Para simular o ataque, foi criada uma infraestrutura na AWS utilizando o Terraform. O ambiente é composto por quatro instâncias principais:

- **Web Server:** Hospeda a aplicação vulnerável, configurada via scripts shell e Docker, nela também está o Snort e o Wazul Agent. O Snort fará o papel de detectar tráfego suspeito que segue determinadas regra, gerando logs de alerta, ao ser configurado no modo IDS e de barrá-los também, quando configurado no modo IPS. O Wazul Agent terá o papel de acessar esses logs e enviá-los para o Wazul Server.
- **Banco de Dados:** Instância separada rodando PostgreSQL, provisionada com Docker Compose.
- **Servidor Atacante:** Instância dedicada para execução dos ataques, equipada com ferramentas como *sqlmap*.
- **Wazul Server:** Instância separada rodando o Wazul Server, onde irá ocorrer a análise e correlação dos logs recebidos do Wazul Agent.

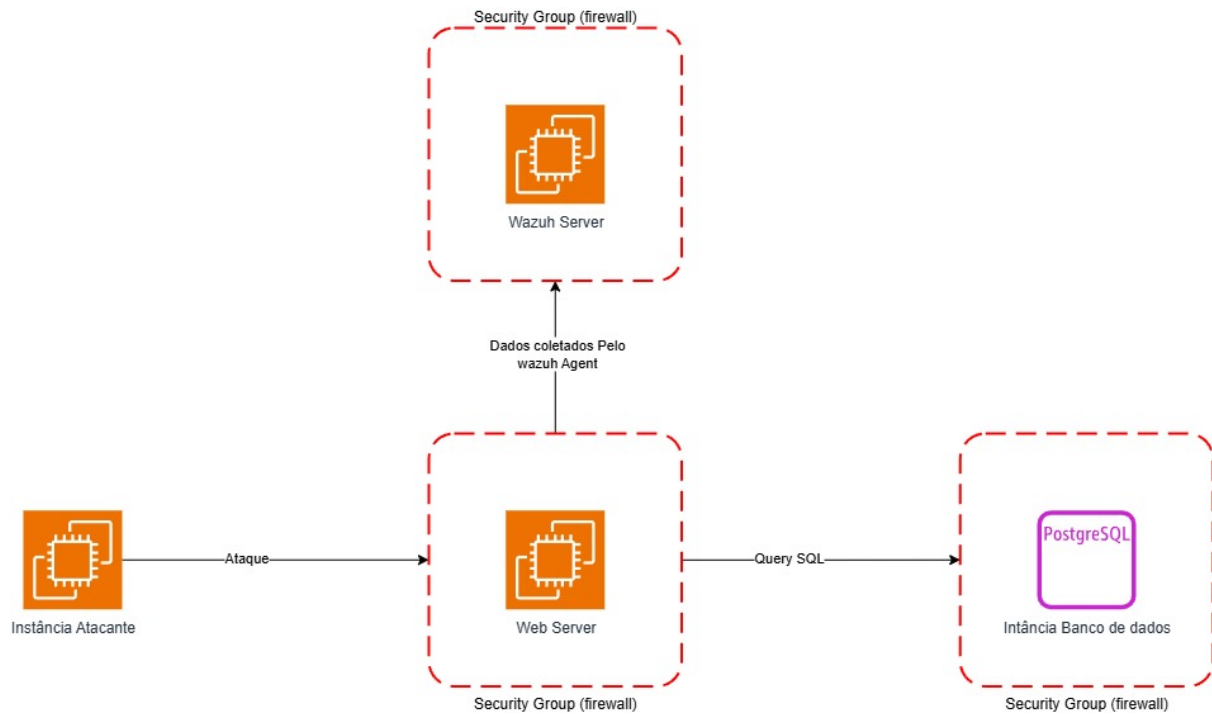


Figura 1: Diagrama da infraestrutura

Os arquivos `.tf` definem recursos como VPC e sub-redes, para garantir que as instâncias possam se comunicar, grupos de segurança (firewalls), de forma que somente as portas necessárias sejam acessíveis, chaves SSH e associações de rotas. Scripts automatizam a configuração dos servidores, instalação de dependências e inicialização dos containers. O uso do Terraform garante reprodutibilidade e controle sobre o ambiente, facilitando a simulação e análise dos ataques.

3 Sqlmap

O **sqlmap** é uma ferramenta automatizada de código aberto projetada para detectar e explorar vulnerabilidades de SQL Injection em aplicações web. Amplamente utilizada por profissionais de segurança e pesquisadores, o sqlmap facilita a identificação de falhas em bancos de dados acessíveis por meio de interfaces web, permitindo a execução de comandos maliciosos e a extração de informações sensíveis.

A ferramenta opera realizando requisições HTTP para endpoints vulneráveis, testando diferentes tipos de injeção e automatizando o processo de exploração. No contexto deste projeto, o sqlmap foi utilizado para simular ataques contra um servidor web, conforme exemplificado no script `attack.sh`.

Esse comando instrui o sqlmap a analisar o endpoint fornecido, identificar possíveis vulnerabilidades e executar o ataque de forma automatizada. O parâmetro `--batch` permite que o processo ocorra sem interação manual, tornando o teste mais eficiente e repetível.

Além disso, o script de configuração (`serverConfig.sh`) garante que o sqlmap esteja instalado e pronto para uso no ambiente de ataque, reforçando a importância da preparação adequada do ambiente para simulações realistas.

Ao realizar o ataque a sistema vulnerável, obtivemos o seguinte resultado:

```

[01:11:01] [INFO] GET parameter 'findEmail' appears to be 'PostgreSQL > 8.1 stacked queries (comment)' injectable
it looks like the back-end DBMS is 'PostgreSQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
for the remaining tests, do you want to include all tests for 'PostgreSQL' extending provided level (1) and risk (1) values? [Y/n] Y
[01:11:01] [INFO] testing 'Generic UNION query (NULL) - 1 to 20 columns'
[01:11:01] [INFO] automatically extending ranges for UNION query injection technique tests as there is at least one other (potential) technique
found
[01:11:01] [INFO] 'ORDER BY' technique appears to be usable. This should reduce the time needed to find the right number of query columns. Autom
atically extending the range for current UNION query injection technique test
[01:11:01] [INFO] target URL appears to have 2 columns in query
[01:11:01] [INFO] GET parameter 'findEmail' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
GET parameter 'findEmail' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 48 HTTP(s) requests:
---
Parameter: findEmail (GET)
  Type: stacked queries
  Title: PostgreSQL > 8.1 stacked queries (comment)
  Payload: findEmail=teste';SELECT PG_SLEEP(5)--

  Type: UNION query
  Title: Generic UNION query (NULL) - 2 columns
  Payload: findEmail=teste' UNION ALL SELECT NULL,(CHR(113)||CHR(98)||CHR(107)||CHR(120)||CHR(113))||(CHR(112)||CHR(74)||CHR(107)||CHR(107)||C
HR(68)||CHR(115)||CHR(104)||CHR(107)||CHR(117)||CHR(97)||CHR(104)||CHR(66)||CHR(90)||CHR(67)||CHR(76)||CHR(84)||CHR(84)||CHR(83)||CHR(114)||CHR(
77)||CHR(103)||CHR(70)||CHR(117)||CHR(120)||CHR(111)||CHR(105)||CHR(78)||CHR(69)||CHR(67)||CHR(102)||CHR(81)||CHR(99)||CHR(76)||CHR(99)||CHR(103
)||CHR(67)||CHR(118)||CHR(88)||CHR(89)||CHR(116))||(CHR(113)||CHR(120)||CHR(98)||CHR(122)||CHR(113))-- rh6y
---
[01:11:01] [INFO] the back-end DBMS is PostgreSQL
web application technology: Express
back-end DBMS: PostgreSQL

```

Figura 2: Ataque a um sistema vulnerável com sqlmap

4 Detecção: Snort e Wazuh

A detecção de intrusões é um componente essencial na defesa de sistemas contra ataques cibernéticos, como o SQL Injection. Ferramentas especializadas permitem identificar padrões suspeitos de tráfego e atividades maliciosas, possibilitando respostas rápidas e eficazes. Nesta seção, serão abordadas duas soluções amplamente utilizadas: Snort e Wazuh.

4.1 Snort

Snort é um sistema de detecção de intrusões em rede (NIDS) de código aberto, capaz de analisar pacotes em tempo real e identificar tentativas de ataque por meio de regras pré-definidas. Para ataques de SQL Injection, o Snort pode ser configurado com regras específicas que detectam padrões comuns em requisições HTTP, como comandos SQL malformados ou tentativas de manipulação de parâmetros. Ao identificar uma possível ameaça, o Snort pode gerar alertas, registrar logs detalhados e até mesmo bloquear o tráfego suspeito, contribuindo para a proteção do ambiente.

Para nosso ataque, utilizamos o snort com as seguintes regras:

Listing 1: Ataque de sqlmap

```

1 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: Bypass autenticacao - ' OR '1'='1
  "; uricontent:" ' OR '1'='1"; nocase; sid:100001; rev:1;)
2
3 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: Coments '--'; uricontent:"--";
  nocase; sid:100002; rev:1;)
4
5 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: OR 1=1"; uricontent:"OR 1=1";
  nocase; sid:100003; rev:1;)
6
7 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: UNION SELECT"; uricontent:"UNION
  SELECT"; nocase; sid:100004; rev:1;)
8
9 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: SELECT * FROM"; uricontent:"
  SELECT * FROM"; nocase; sid:100005; rev:1;)
10
11 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: xp_cmdshell detectado"; content:"
  xp_cmdshell"; nocase; sid:100006; rev:1;)
12
13 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: WAITFOR DELAY (blind SQLi)";
  uricontent:"WAITFOR DELAY"; nocase; sid:100007; rev:1;)
14

```

```

15 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: CHAR/ASCII obfuscation";
    uricontent:"CHAR("; nocase; sid:100008; rev:1;)
16
17 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: OR TRUE--"; uricontent:"OR TRUE--
    "; nocase; sid:100009; rev:1;)
18
19 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: DROP TABLE tentativa"; uricontent
    : "DROP TABLE"; nocase; sid:100010; rev:1;)
20
21 alert tcp any any -> $HOME_NET 80 (msg:"SQLi: Express o regex OR X=X"; pcre:"
    /(\\%27)|(\\')|(\\-\\-)|(\\%23)|(#)/i"; sid:100011; rev:1;)

```

No webserver, ao realizar o ataque, o snort apontou o seguinte nos logs:

```

07/20-01:10:51.269462 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:39262 -> 10.0.1.231:80
07/20-01:10:51.269462 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:39262 -> 10.0.1.231:80
07/20-01:10:56.290192 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:39278 -> 10.0.1.231:80
07/20-01:10:56.290192 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:39278 -> 10.0.1.231:80
07/20-01:10:56.294530 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36166 -> 10.0.1.231:80
07/20-01:10:56.294530 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36166 -> 10.0.1.231:80
07/20-01:11:01.311924 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36182 -> 10.0.1.231:80
07/20-01:11:01.311924 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36182 -> 10.0.1.231:80
07/20-01:11:01.317897 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36194 -> 10.0.1.231:80
07/20-01:11:01.317897 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36194 -> 10.0.1.231:80
07/20-01:11:01.324414 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36214 -> 10.0.1.231:80
07/20-01:11:01.324414 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36214 -> 10.0.1.231:80
07/20-01:11:01.359811 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36204 -> 10.0.1.231:80
07/20-01:11:01.359811 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36204 -> 10.0.1.231:80
07/20-01:11:01.392219 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36216 -> 10.0.1.231:80
07/20-01:11:01.392219 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36216 -> 10.0.1.231:80
07/20-01:11:01.427443 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36228 -> 10.0.1.231:80
07/20-01:11:01.427443 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36228 -> 10.0.1.231:80
07/20-01:11:01.454217 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36234 -> 10.0.1.231:80
07/20-01:11:01.454217 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36234 -> 10.0.1.231:80
07/20-01:11:01.491292 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36244 -> 10.0.1.231:80
07/20-01:11:01.491292 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36244 -> 10.0.1.231:80
07/20-01:11:01.570744 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36292 -> 10.0.1.231:80
07/20-01:11:01.570744 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36292 -> 10.0.1.231:80
07/20-01:11:01.580648 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36314 -> 10.0.1.231:80
07/20-01:11:01.580648 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36314 -> 10.0.1.231:80
07/20-01:11:01.588975 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36306 -> 10.0.1.231:80
07/20-01:11:01.588975 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36306 -> 10.0.1.231:80
07/20-01:11:01.597596 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36316 -> 10.0.1.231:80
07/20-01:11:01.597596 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36316 -> 10.0.1.231:80
07/20-01:11:01.607347 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36318 -> 10.0.1.231:80
07/20-01:11:01.607347 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36318 -> 10.0.1.231:80
07/20-01:11:01.615447 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36348 -> 10.0.1.231:80
07/20-01:11:01.615447 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36348 -> 10.0.1.231:80
07/20-01:11:01.622669 [**] [1:100002:1] SQLi: Coments '--' [**] [Priority: 0] {TCP} 10.0.1.250:36334 -> 10.0.1.231:80
07/20-01:11:01.622669 [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:36334 -> 10.0.1.231:80

```

Figura 3: Logs do snort, configurado em modo IDS para detecção

4.2 Wazuh

Wazuh é uma plataforma de segurança que integra funcionalidades de detecção de intrusões, análise de logs e monitoramento de integridade. Diferente do Snort, o Wazuh atua tanto na análise de eventos de rede quanto na inspeção de atividades em sistemas operacionais e aplicações. Para SQL Injection, o Wazuh pode correlacionar logs de servidores web e bancos de dados, identificando comportamentos anômalos, como consultas SQL inesperadas ou falhas de autenticação. Além disso, o Wazuh oferece dashboards e relatórios que facilitam a visualização dos incidentes, permitindo uma resposta mais eficiente às tentativas de invasão.

Essas ferramentas foram configuradas para analisar o tráfego entre as instâncias e gerar alertas em caso de detecção de comandos SQL maliciosos, contribuindo para a resposta rápida a incidentes.

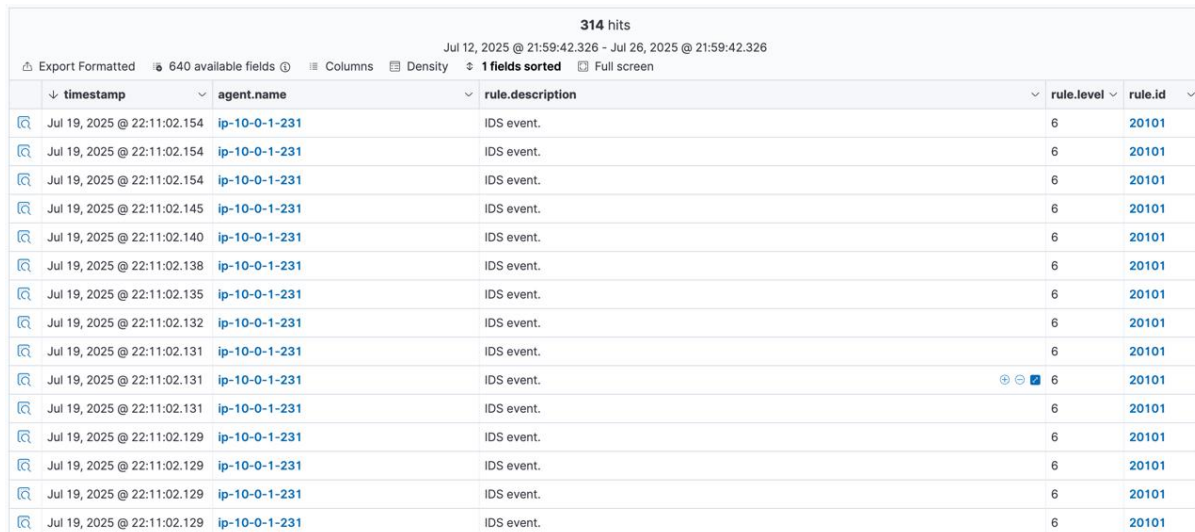
4.2.1 Wazuh Agent

O Wazuh Agent é instalado no nosso webserver. Ele coleta informações locais, como logs do sistema, alterações em arquivos e eventos de segurança, enviando esses dados ao Wazuh Server para análise. O agente é leve e pode ser executado em diferentes sistemas operacionais, garantindo ampla cobertura na detecção de ameaças.

4.2.2 Wazuh Server

O Wazuh Server é responsável por coletar, analisar e correlacionar os dados enviados pelos agentes. Ele centraliza as informações, aplica regras de detecção e gera alertas para os administradores. Por ser uma ferramenta muito pesada, tivemos que configurá-la para uma instância separada e com mais memória (no caso, uma t3.medium da AWS) do que as outras instâncias.

Ao realizar o ataque, o Wazul Server apontou o seguinte:



The screenshot shows the Wazuh dashboard interface. At the top, it indicates '314 hits' for the period 'Jul 12, 2025 @ 21:59:42.326 - Jul 26, 2025 @ 21:59:42.326'. Below this, there are controls for 'Export Formatted', '640 available fields', 'Columns', 'Density', '1 fields sorted', and 'Full screen'. The main table displays a list of IDS events. Each row contains a timestamp, an agent name (all are 'ip-10-0-1-231'), a rule description (all are 'IDS event.'), a rule level (all are '6'), and a rule ID (all are '20101').

timestamp	agent.name	rule.description	rule.level	rule.id
Jul 19, 2025 @ 22:11:02.154	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.154	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.154	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.145	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.140	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.138	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.135	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.132	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.131	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.131	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.131	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.129	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.129	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.129	ip-10-0-1-231	IDS event.	6	20101
Jul 19, 2025 @ 22:11:02.129	ip-10-0-1-231	IDS event.	6	20101

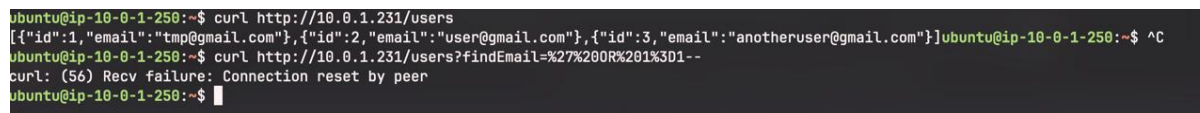
Figura 4: Dados do Wazul Server após o ataque

5 Mitigações

O Snort, além de atuar como IDS, pode ser configurado no modo IPS (Intrusion Prevention System), permitindo que ele intervenha diretamente no tráfego de rede.

No modo IPS, o Snort analisa os pacotes e, ao identificar padrões de ataque definidos em suas regras, pode descartar ou modificar pacotes suspeitos antes que alcancem o destino. Isso impede que comandos maliciosos sejam executados no banco de dados, protegendo a aplicação contra explorações conhecidas. O uso do Snort no modo IPS é uma estratégia eficaz para mitigar ataques de SQL Injection, atuando de forma proativa na proteção dos sistemas e reduzindo o risco de comprometimento dos dados.

Ao adicionar a configuração do snort para o modo IPS, obtivemos os seguintes resultados:



The terminal screenshot shows a user attempting to connect to a web service. The first command is 'curl http://10.0.1.231/users', which returns a JSON array of user data. The second command is 'curl http://10.0.1.231/users?findEmail=%27%20OR%201%3D1--', which results in a 'curl: (56) Recv failure: Connection reset by peer' error.

```
ubuntu@ip-10-0-1-250:~$ curl http://10.0.1.231/users
[{"id":1,"email":"tmp@gmail.com"}, {"id":2,"email":"user@gmail.com"}, {"id":3,"email":"anotheruser@gmail.com"}]ubuntu@ip-10-0-1-250:~$ ^C
ubuntu@ip-10-0-1-250:~$ curl http://10.0.1.231/users?findEmail=%27%20OR%201%3D1--
curl: (56) Recv failure: Connection reset by peer
ubuntu@ip-10-0-1-250:~$
```

Figura 5: Tentativa de execução do ataque partindo do atacante

```

=====
Action Stats:
  Alerts:          17 ( 37.778%)
  Logged:          17 ( 37.778%)
  Passed:           0 (  0.000%)
Limits:
  Match:           0
  Queue:           0
  Log:             0
  Event:           0
  Alert:           0
Verdicts:
  Allow:           34 ( 75.556%)
  Block:           0 (  0.000%)
  Replace:         0 (  0.000%)
  Whitelist:       0 (  0.000%)
  Blacklist:       11 ( 24.444%)
  Ignore:          0 (  0.000%)
  Retry:           0 (  0.000%)
=====

```

Figura 6: Resultado do ataque ao usar modo IPS do Snort

```

07/20-02:50:09.446300 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46092 -> 10.0.1.231:80
07/20-02:50:09.502172 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46108 -> 10.0.1.231:80
07/20-02:50:09.502650 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46108 -> 10.0.1.231:80
07/20-02:50:09.502693 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46108 -> 10.0.1.231:80
07/20-02:50:09.504988 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46110 -> 10.0.1.231:80
07/20-02:50:09.505317 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46110 -> 10.0.1.231:80
07/20-02:50:09.505385 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46110 -> 10.0.1.231:80
07/20-02:50:09.507947 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46126 -> 10.0.1.231:80
07/20-02:50:09.508538 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46126 -> 10.0.1.231:80
07/20-02:50:09.508538 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:46126 -> 10.0.1.231:80
07/20-02:52:55.061802 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:52:56.115485 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:52:57.139484 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:52:58.163499 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:52:59.187507 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:53:00.211577 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:53:02.259607 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:53:06.291497 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:52964 -> 10.0.1.231:80
07/20-02:53:12.651339 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:12.651839 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:12.651840 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:12.665148 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:12.665402 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:12.665769 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:39764 -> 10.0.1.231:80
07/20-02:53:26.199931 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:26.200317 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:26.200576 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:26.213825 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:26.214085 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:26.214467 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35982 -> 10.0.1.231:80
07/20-02:53:31.298336 [Drop] [**] [1:100003:1] SQLi: OR 1=1 [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80
07/20-02:53:31.298336 [Drop] [**] [1:100003:1] SQLi: Comments '--' [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80
07/20-02:53:31.298336 [Drop] [**] [1:100011:1] SQLi: Expressão regex OR X=X [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80
07/20-02:53:31.297924 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80
07/20-02:53:31.298309 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80
07/20-02:53:31.298336 [**] [1:10001:1] Dropping HTTP [**] [Priority: 0] {TCP} 10.0.1.250:35988 -> 10.0.1.231:80

```

Figura 7: Logs dos pacotes barrados no arquivo snort.alert.fast

6 Conclusão

Por meio da simulação realizada neste projeto, foi possível compreender o funcionamento do ataque de SQL injection, suas consequências e as principais técnicas utilizadas para explorá-lo, como o uso da ferramenta *sqlmap*. Além disso, foram abordadas estratégias de defesa e monitoramento, destacando o papel de sistemas de detecção de intrusões como Snort e Wazuh, que auxiliam na identificação e resposta a atividades suspeitas.

Portanto, este estudo mostra que a segurança contra SQL Injection depende não apenas de ferramentas específicas, mas de uma arquitetura robusta, processos bem definidos e constante atualização frente às novas ameaças. A conscientização e o treinamento das equipes de desenvolvimento e operação são fundamentais para criar sistemas mais seguros e resilientes.

Referências

- [1] OWASP Foundation. *SQL Injection*. Disponível em: https://owasp.org/www-community/attacks/SQL_Injection
- [2] sqlmap Project. *Automatic SQL Injection and Database Takeover Tool*. Disponível em: <http://sqlmap.org>
- [3] Snort. *Network Intrusion Prevention and Detection System*. Disponível em: <https://www.snort.org>
- [4] Wazuh. *Open Source Security Platform*. Disponível em: <https://wazuh.com>