Overview

Wyatt Pangerl, Kevin Tröll, Jona Monette

The goal of our tool set is to combat SQL injection attacks executed using SQLmap by implementing **Snort** firewall rules to filter SQLmap packets from incoming network traffic. SQLmap network traffic has unique, identifiable headers and thus can be filtered out with simple firewall rules.

[GIT repository]

Snort Rule #1

Steps we followed to create the rule:

- Created rule file in /etc/snort/rules/ (rule in the git repo)
- Added rule to /etc/snort/snort.conf to enable it
- Started a web server and collected web traffic on port 80, while running a SQLmap scan on the web server, using Wireshark (.pcap file in the git repo)
- Ingested the pcap file into snort using this command:
    - `sudo snort -r sqlmap.pcap -c /etc/snort/snort.conf`
- Examined the snort conf file to see the matched rule (screenshot in the git repo)

The Snort rule:

When SQLmap attempts to connect to a site, it will display "sqlmap" in the connection headers for the useragent. It will be displayed twice. Once at the beginning of the

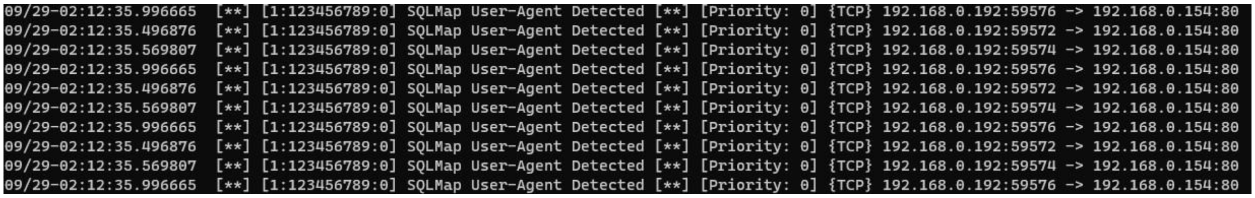
header, and towards the end of the header in a URL. This rule also looks at all ports in case an application is running on another port other than port 80.

```
alert tcp any any -> any any (msg:"SQLMap User-Agent Detected";
content:"User-Agent"; nocase; http_header; pcre:"/sqlmap/i"; sid:123456789;)
```

The rule checks for the SQLmap header and blocks the packet if it matches it.

Proof of Value #1

The following screenshot shows the snort rule detecting SQLmap traffic:



```
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
09/29-02:12:35.496876  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59572 -> 192.168.0.154:80
09/29-02:12:35.569807  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59574 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
09/29-02:12:35.496876  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59572 -> 192.168.0.154:80
09/29-02:12:35.569807  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59574 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
09/29-02:12:35.496876  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59572 -> 192.168.0.154:80
09/29-02:12:35.569807  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59574 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
```

Snort alert breakdown:

1. Timestamp: The timestamp indicates when the alert was generated. It follows the format “MM/DD-HH:MM:SS.”

2. Alert Rule: Snort uses rules to define conditions that trigger an alert when matched. The rule is enclosed in square brackets, like ‘[1:123456789:01 SQLMap User-Agent Detected]’.

Components of the rule:

- - [1:123456789:01]: This is the rule ID, a unique identifier for the rule.
- - SQLMap: This is a message or description associated with the rule.
- - User-Agent Detected: This indicates the condition that triggered the alert.

3. Priority: Priority levels can help classify the severity of the alert.

4. Source and Destination IP and Port: The source and destination IP addresses and port numbers involved in the network communication that triggered the alert. For example:

- 192.168.0.192:59576 => 192.168.0.154:80: This represents traffic from source IP 192.168.0.192, source port 59576, going to destination IP 192.168.0.154, and destination port 80

Snort Rule #2

Steps we followed to create the rule:

- Created rule file in /etc/snort/rules/ (rule in the git repo)
- Added rule to /etc/snort/snort.conf to enable it
- Started a web server and collected web traffic on port 80, while running a SQLmap scan on the web server, using Wireshark (.pcap file in the git repo)
- Ingested the pcap file into snort using this command:
    - `sudo snort -r sqlmap.pcap -c /etc/snort/snort.conf`
- Examined the snort conf file to see the matched rule (screenshot in the git repo)

The Snort rule:

This rule behaves similarly to the previous rule. When SQLmap attempts to connect to a site, it repeats a portion of the header “`Connection:\s*close`” twice.

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Connection: close header detected"; flow:to_server,established; content:"Connection: close"; nocase; pcre:"/Connection:\s*close.*Connection:\s*close/i"; sid:1000001;)
```

The rule checks for the repeated content in the SQLmap header and blocks the packet if it matches it.

Proof of Value #2

The below screenshots show the snort rule working on the sqlmap traffic, and the second screenshot shows what sqlmaps connection headers look like, note the 2 connection headers.

```
09/29-02:12:35.996665  [**] [1:1000003:0] HTTP Connection: close header dete
cted [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**]
[Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
```

```
GET / HTTP/1.1
Host: spoutingwhale.requestcatcher.com
Connection: close
Accept: */*
Accept-Encoding: gzip,deflate
Cache-Control: no-cache
Connection: close
User-Agent: sqlmap/1.4.4#stable (http://sqlmap.org)
```

Snort alert breakdown:
1. Timestamp: The timestamp indicates when the alert was generated. It follows the format “MM/DD-HH:MM:SS.”

2. Alert Rule: Snort uses rules to define conditions that trigger an alert when matched. The rule is enclosed in square brackets, like ‘[1:1000003:0 HTTP Connection: close header detected]’.

Components of the rule:

- - [[1:1000003:0]: This is the rule ID, a unique identifier for the rule.
- - HTTP Connection: This is a message or description associated with the rule.
- - close header Detected: This indicates the condition that triggered the alert.

3. Priority: Priority levels can help classify the severity of the alert.

4. Source and Destination IP and Port: The source and destination IP addresses and port numbers involved in the network communication that triggered the alert. For example:

- 192.168.0.192:59576 => 192.168.0.154:80: This represents traffic from source IP 192.168.0.192, source port 59576, going to destination IP 192.168.0.154, and destination port 80

Script - Snort Rule Deployment

After creating our two rules, we determined a good addition to these would be creating a script for deploying them automatically. The script assumes a functioning snort install.

```
  GNU nano 2.9.3                          deploy_snort_rules.sh

#!/bin/bash
# Function to deploy Snort rules
deploy_rules() {
    echo "$1" | sudo tee -a /etc/snort/rules/local.rules > /dev/null
}
# Function to restart Snort
restart_snort() {
    sudo systemctl restart snort
}
# Snort rule 1
rule1='alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Connection: close header detected"; f$
# Snort rule 2
rule2='alert tcp any any -> any any (msg:"SQLMap User-Agent Detected"; content:"User-Agent"; nocase; ht$
# Deploy rules
deploy_rules "$rule1"
deploy_rules "$rule2"
# Restart Snort
restart_snort
echo "Snort rules deployed and Snort restarted."
```

Steps and components to create the script:

1- We defined that this script should be run in the Bash shell.

2- We created a function "*deploy_rules()*" that takes the first argument passed to the function and appends it to the "*/etc/snort/rules/local.rules*", where the file used by Snort for custom rules is located.

3- We created a "*restart_snort()*" function that uses a command to restart the Snort service.

4- We defined the snort rules #1 and #2 that were created previously.

5- “*deploy_rules*” call the functions for rule #1 and #2, passing the rules as arguments.

6- *“restart_snort”* function to restart the Snort Service with a confirmation message

7- We created the script executable by using the command *“chmod +x deploy_snort_rules.sh”*.

It is important to note that this script assumes you have administrative privileges (‘sudo’) to modify the Snort’s configuration file and restart the service.

Proof of Value #3

The following show the successful deployment of the script to a fresh Snort install.

```
americaninseoul@wntrxsldr:/etc/snort/rules$ cat local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
americaninseoul@wntrxsldr:/etc/snort/rules$ clear
```

Snort local.rules file before executing the script

```
americaninseoul@wntrxsldr:~/Desktop$ touch deploy_snort_rules.sh
americaninseoul@wntrxsldr:~/Desktop$ sudo chmod +x deploy_snort_rules.sh
[sudo] password for americaninseoul:
americaninseoul@wntrxsldr:~/Desktop$ sudo nano deploy_snort_rules.sh
americaninseoul@wntrxsldr:~/Desktop$ ./deploy_snort_rules.sh
Snort rules deployed and Snort restarted.
```

Script deploying snort rules

```
americaninseoul@wntrxsldr:/etc/snort/rules$ cat local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"Connection: close header detected"; flow:to_server,e
stablished; content:"Connection: close"; nocase; pcre:"/Connection:\s*close.*Connection:\s*close/i"; sid:100000
1;)
alert tcp any any -> any any (msg:"SQLMap User-Agent Detected"; content:"User-Agent"; nocase; http_header; pcre
:"/sqlmap/i"; sid:123456789;)
americaninseoul@wntrxsldr:/etc/snort/rules$
```

Snort local.rules file after executing the script

```
===================================================================================
Action Stats:
     Alerts:            6 ( 12.500%)
     Logged:            6 ( 12.500%)
     Passed:            0 (  0.000%)
Limits:
      Match:            0
      Queue:            0
        Log:            0
      Event:            0
      Alert:            0
Verdicts:
      Allow:           46 (100.000%)
      Block:            0 (  0.000%)
    Replace:            0 (  0.000%)
  Whitelist:            0 (  0.000%)
  Blacklist:            0 (  0.000%)
     Ignore:            0 (  0.000%)
      Retry:            0 (  0.000%)
===================================================================================
```

Snort Verbose output showing that there were alerts triggered

```
snort@snort:~$ date
Fri Dec  1 02:11:33 AM UTC 2023
snort@snort:~$ tail /var/log/snort/snort.alert.fast
09/29-02:12:35.496876  [**] [1:1000003:0] HTTP Connection: close header detected [**] [Priority: 0] {TCP} 192.168.0.192:59572 -> 192.168.0.154:80
09/29-02:12:35.496876  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59572 -> 192.168.0.154:80
09/29-02:12:35.569807  [**] [1:1000003:0] HTTP Connection: close header detected [**] [Priority: 0] {TCP} 192.168.0.192:59574 -> 192.168.0.154:80
09/29-02:12:35.569807  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59574 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:1000003:0] HTTP Connection: close header detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
09/29-02:12:35.996665  [**] [1:123456789:0] SQLMap User-Agent Detected [**] [Priority: 0] {TCP} 192.168.0.192:59576 -> 192.168.0.154:80
snort@snort:~$
```

Snort logs alerting to SQLmap traffic