Recursos de pentesting

Introducció

Quan parlem de "test de penetració" (en anglès, "penetration test", d'on deriva l'expressió "**pentest**") ens estem referint a un conjunt d'activitats (sovint emmarcades dins d'una auditoria de seguretat més àmplia) que tenen com a objectiu identificar les vulnerabilitats d'una xarxa o sistema i explotar-les per tal de posar a prova els sistemes de seguretat existents. D'aquesta manera, es podran esmenar les falles trobades abans de què aquestes puguin ser aprofitades per un atacant no autoritzat.

> NOTA: És comú anomenar "**red team**" a l'atacant (autoritzat) que realitza el "pentest" mentre que l'"equip defensor" que protegeix amb diferents tècniques (ja siguin preventives o reactives) els recursos atacats se sol anomenar "**blue team**". En aquest sentit, "**purple teaming**" is when a red team (penetration testers) and the blue team (security operations centers) work together to improve threat detection: the red team would deploy attacks that the blue team would attempt to identify them; if an attack is missed, the blue team would need to identify the gap in their alerting tools.

Els pentests han de ser consensuats amb el client per a què siguin legals. És molt important acordar els objectius del pentest (obtenir informació confidencial?, col.lapsar algun servei?, prendre el control d'alguna màquina?, etc), el seu abast (quines IPs i serveis de l'empresa -com ara: web, correu, DNS, firewall/IDS/IPS...- s'auditaran i quines no?, si s'inclouran tècniques agressives com el "*phishing*" (via email o altres) contra els empleats o no?, etc) i la seva duració.

Queda clar, doncs, que cada pentest és diferent perquè dependrà de les necessitats de cada client: en alguns casos es tractarà simplement de verificar la seguretat d'una aplicació concreta mentre que en altres casos serà necessari realitzar proves de penetració contra un segment de xarxa complet. En qualsevol cas, encara que es tractin d'escenaris diferents, la metodologia ha seguir sol ser sempre molt semblant. De fet, al llarg dels anys han aparegut diferents estàndards promoguts per diverses empreses i organitzacions especialitzades en l'àmbit de la ciberseguretat que defineixen sengles metodologies (les quals tenen l'objectiu d'oferir al "red team" una planificació de treball completa, estructurada i sistematitzada que permeti realitzar el "pentesting" sense deixar res a la improvisació) que, tot i variar en petits detalls, "a grosso modo" recullen sempre els mateixos passos (el conjunt dels quals l'anomenarem "cicle de vida del pentesting") descrits a l'apartat que ve a continuació (després dels quadres següents).

Metodologies

*__PTES__ (http://www.pentest-standard.org). Wiki que representa una guia tècnica amb tot els detalls i recomanacions necessàries (incloent l'ús d'eines i comandes concretes) per realitzar tots els passos que integren un "pentesting" complet (des de la comunicació inicial amb el client, passant per l'obtenció d'informació de l'objectiu -de forma passiva o activa-, l'escaneig i explotació de les vulnerabilitats trobades, la realització de les tècniques de "post-explotació" adients -moviment lateral, escalada de privilegis, exfiltracions, etc- i l'ocultament de rastres, fins l'entrega de l'informe final).

*__OSSTMM__ (https://www.isecom.org/research.html). Document que descriu un mètode científic que abasta també tots els aspectes d'un procés de "pentesting" canònic amb procediments consistents (però fàcilment adaptables a les circumstàncies concretes de l'objectiu investigat en cada moment). Està enfocat més en el "què" que en el "com" i proporciona una referència per interpretar els resultats obtinguts i redactar-ne l'informe pertinent

*__ATT&CK__ (https://attack.mitre.org/matrices/enterprise) : Més que una metodologia en sí, ATT&CK és una (gran) base de dades públicament accessible plena de tàctiques, tècniques i software específic emprats pels "dolents", basada en observacions del món real. Ofereix una estructura molt completa, homogènia i ordenada dels tipus d'atacs reconeguts al món a partir d'una classificació per tipus de vulnerabilitats conegudes (i també pel diferent software ofensiu estudiat). És usada per desenvolupar, a partir dels comportaments observats, models d'amenaces i, a partir d'ells, metodologies concretes de protecció, detecció i mitigació.

La "gràcia" és que la informació disponible en ATT&CK és oferida en forma de matriu interactiva on cada columna representa una tàctica reconeguda (el "per què" de l'atac, el seu objectiu) i les cel·les que hi apareixen a cada columna representen les tècniques (el "com" de l'atac) que es corresponen a aquella tàctica en particular (filtrades, si es vol, per sistema operatiu, incloent de mòbils). Seleccionant un element concret d'aquesta matriu es podrà conèixer tot els detalls d'aquest element (es mostra una descripció molt exhaustiva de la tàctica/tècnica de l'atac i el seu context, una llista del vulnerabilitats i software -aquí anomenat també "procediments"- relacionats, un recull de les mitigacions possibles,...) i relacionar-lo amb altres elements similars de la matriu.

OWASP

OWASP (https://www.owasp.org) és una fundació internacional que desenvolupa de forma col.laborativa guies, eines i protocols d'actuació que qualsevol entitat/empresa podrà aplicar després en diferents àmbits de la seguretat informàtica (no només per fer "pentesting"). Va ser creada per diversos experts en aquest àmbit amb la finalitat de determinar i combatre les causes que fan que un software (ja sigui d'escriptori, mòbil, web...) sigui insegur, establint com a conseqüència mètodes de treballs segurs a l'hora de desenvolupar i implementar aplicacions. Entre les seves publicacions podem trobar la llista de les "**Top 10**" vulnerabilitats (o problemes de seguretat) web més importants/comunes (https://owasp.org/www-project-top-ten); l'objectiu d'aquesta llista és formar i conscienciar a tots els implicats en la seguretat per reduir els riscos i conseqüències.

NOTA: Existeix una llista similar per a les vulnerabilitats més importants a nivell d'API (https://owasp.org/www-project-api-security) i una altra per a vulnerabilitats a nivell d'aplicacions mòbils (https://owasp.org/www-project-mobile-top-10) . També existeix una llista de vulnerabilitats més importants relacionades amb aplicacions "server-less" (FaaS) però aquest no està mantinguda per OWASP sinó per l'empresa Palo Alto (https://github.com/puresec/sas-top-10)
NOTA: Com a contrapartida a la llista de vulnerabilitats "Top 10", OWASP també té publicat el document "**Top 10 Proactive Controls**" (https://owasp.org/www-project-proactive-controls), on es descriuen tot un seguit d'aspectes generals que es recomanen implementar en qualsevol projecte de desenvolupament de software per tal d'enfortir-lo davant d'aquestes.

Aquesta llista "Top 10" es basa en altres documents fonamentals generats per OWASP, com són:

*La guia **WSTG** (https://owasp.org/www-project-web-security-testing-guide). És el document publicat per OWASP més semblant al que seria una metodologia de "pentesting" (està enfocada, però, només a aplicacions i serveis web).

NOTA: També existeix la guia **MSTG** i (https://owasp.org/www-project-mobile-security-testing-guide), que proporciona una metodologia de "pentesting" on els processos, tècniques i eines estan enfocades, en canvi, específicament a aplicacions mòbils. En aquest sentit, OWASP també proporciona el projecte **MSTG-HP** (https://github.com/OWASP/MSTG-Hacking-Playground), el qual consisteix en un conjunt d'"apps" per Android i iOS insegures de forma intencionada per tal de fer-les servir com exemples de diferents vulnerabilitats explicades a la guia MSTG. En qualsevol cas, per saber tots els projectes OWASP relacionats amb la seguretat en l'àmbit de les aplicacions mòbils, es pot consultar la web "paraigües" https://owasp.org/www-project-mobile-security
NOTA: També existeix la guia **FSTM**, especialitzada en proporcionar consells de seguretat als desenvolupadors i enginyers relacionats amb l'avaluació de la seguretat en el "firmware" (https://scriptingxss.gitbook.io/firmware-security-testing-methodology)

*La guia **ASVS (**https://owasp.org/www-project-application-security-verification-standard) És un document molt complet, dirigit sobre tot als desenvolupadors, sobre bones pràctiques a l'hora de dissenyar, escriure i testejar aplicacions web de forma segura.

NOTA: També existeix **MASVS** (https://github.com/OWASP/owasp-masvs) , la guia equivalent a ASVS però enfocada al diseny, desenvolupament i testeig d'apps mòbils. D'altra banda, OWASP també proporciona el projecte **SKF** (https://owasp.org/www-project-security-knowledge-framework) el qual consisteix en un entorn web que conté classificats múltiples codis d'exemple en els que es poden posar en pràctica els diferents aspectes tractats a les guies ASVS i MASVS (fins i tot incorpora un conjunt d'aplicacions -en aquest cas noms web- vulnerables per a practicar).

La fundació OWASP, a més de publicar documentació tècnica, també promou el desenvolupament de projectes de software, tant de defensa (les anomenades "defenders"; és a dir, eines i llibreries que permeten detectar i contrarestar atacs al nivell d'aplicació) com d'atac (les anomenades "breakers"; és a dir, eines i llibreries que permeten realitzar proves de "pentesting"), així com també de desenvolupament (les anomenades "builders"; és a dir, eines i llibreries que permeten desenvolupar aplicacions de forma segura en múltiples llenguatges i plataformes). Com que la quantitat d'aquests projectes és molt gran, a l'enllaç https://owasp.org/projects es poden trobar classificats per ordre de rellevància i d'actualitzacions més freqüents els projectes més importants. Entre ells, podem destacar:

*__Zed Attack Proxy__ (https://www.zaproxy.org) Proxy HTTP/S i d'atac web que incorpora diferents mòduls (proxy HTTP/S MitM,"web spider", escanejador de vulnerabilitats, etc). Una alternativa a ZAP -però amb interfície web- és **OWTF** (https://owtf.github.io), el qual segueix els protocols d'actuació i vulnerabilitats desenvolupades a la documentació generada per OWASP, NIST i PTES

*__Nettacker__ (https://owasp.org/www-project-nettacker) Eina OSINT que permet descobrir nodes a la xarxa i recopilar informació sobre ells (serveis que executen, configuracions -mal- establertes, vulnerabilitats trobades, etc) per tal de generar un informe. Una altra eina OSINT que, en aquest cas permet trobar subdominis DNS publicats a diverses fons d'Internet, recerces WHOIS inverses, correlacions amb ASNs, etc, és **Amass** (https://owasp.org/www-project-amass)

*__ModSecurity CSR__ (https://coreruleset.org) . Conjunt de regles prefabricades llestes per incorporar ModSecurity (https://github.com/owasp-modsecurity/ModSecurity), un tallafocs specialitzat en tràfic HTTP/S (també anomenats genèricament "WAF", de "Web Application Firewall")

*__Dependency Track__ (https://dependencytrack.org) . Aplicació que analitza les dependències de tercers d'un projecte software i en troba les vulnerabilitats conegudes (obtingudes de la base de dades pública NVD) segons les versions detectades d'aquestes.

NOTA: Un altre projecte amb objectius similars (però que és independent d'OWASP) és **Snyk** (https://snyk.io). En comptes de la NVD, però, fa servir una base de dades de vulnerabilitats pròpia, disponible a https://snyk.io/vuln

*__Security Shepherd__ (https://owasp.org/www-project-security-shepherd). Aplicació web/mòbil vulnerable, especialment dissenyada per practicar habilitats de "pentesting". D'altra banda, hi ha moltes més plataformes de tipus web expressament vulnerables pensades per poder-hi explotar vulnerabilitats de forma legal, com **JuiceShop** (https://owasp.org/www-project-juice-shop) ; **WebGoat** (https://owasp.org/www-project-webgoat) ; **DVWA** (http://dvwa.co.uk) o **AltoroJ** (https://github.com/AppSecDev/AltoroJ), entre d'altres (de fet, hi ha un llistat molt més exhaustiu a https://owasp.org/www-project-vulnerable-web-applications-directory, on s'inclouen, a més de plataformes web, també sistemes complets...alguns es detallen al quadre següent)

NOTA: Un altre projecte interessant d'OWASP és **IoTGoat** (https://github.com/OWASP/IoTGoat), el qual, en comptes de ser una aplicació web vulnerable, és un firmware OpenWRT (com els de molts "routers" domèstics). També hi ha https://github.com/OWASP/NodeGoat (que representa una aplicació Node vulnerable) o ; https://github.com/OWASP/Serverless-Goat (que representa una aplicació "serverless" vulnerable)

A més de les anteriors, existeixen moltes altres aplicacions web lliures (més enllà del projecte OWASP) que són dissenyades de forma vulnerable a propòsit per tal de poder practicar amb elles tècniques de "pentesting". A continuació es llisten unes quantes:

https://github.com/webpwnized/mutillidae
https://github.com/hummingbirdscyber/Vulnerable-Web-Application
https://github.com/opendns/Security_Ninjas_AppSec_Training
https://google-gruyere.appspot.com/start
http://www.itsecgames.com

A més d'aplicacions web vulnerables individuals, també hi ha força distribucions Linux completes que inclouen vàries aplicacions web vulnerables tot-en-un. Exemples són:

https://sourceforge.net/projects/websecuritydojo/files
https://sourceforge.net/projects/metasploitable/files/Metasploitable2
https://github.com/rapid7/metasploitable3
https://github.com/chuckfw/owaspbwa
https://github.com/SamuraiWTF/samuraiwtf

Es pot trobar una llista de més distribucions d'aquest tipus a https://www.amanhardikar.com/mindmaps/Practice.html i també a https://github.com/commixproject/commix/wiki/Command-Injection-Testbeds

Vulnerabilitats

A continuació es mencionaran els tipus de vulnerabilitats més coneguts (de natura web, ja que en general són les més exposades), molts dels quals també apareixen al "Top 10" d'OWASP (de fet, aquests tipus de vulnerabilitats estan associats a diferents tipus d'atacs que es poden veure llistats a https://www.owasp.org/index.php/Category:Attack):

*__Cross-Site Scripting (XSS)__ (https://www.owasp.org/index.php/Cross-site_Scripting_(XSS)): Permite, utilizando los parámetros de entrada de la aplicación, modificar y añadir código Javascript a la misma. Son vulnerabilidades que se encuentran en el servidor, pero que están destinadas a atacar al cliente: generalmente, se necesita que este siga un enlace de la aplicación, en el que se ha modificado algún parámetro, para permitir añadir código, que se ejecutará en el navegador del cliente. Normalmente la intención será un robo de cookies del cliente, predicción del id de sesión, etc. El cliente verá que al seguir el enlace llega realmente al sitio web que quería: no hay suplantación del mismo. Podemos agrupar los XSS en dos grupos, los *XSS reflejados* (donde el código modificado se elimina al cargar la página de nuevo; está basado en la URL y, por tanto, al recargar la página, se elimina el mismo) y los *XSS persistentes* (donde el código modificado queda almacenado en la web)

La forma de corregir esta vulnerabilidad es filtrar y validar todas las entradas de la aplicación. No se debe utilizar nunca una variable que se recibe desde el cliente, confiando en que ésta tendrá un valor correcto. Los lenguajes de programación incluyen diferentes funciones que permiten filtrar el contenido de las variables, dependiendo de dónde las vayamos a utilizar. Para ver más en concreto cómo se podría hacer esto en un servidor PHP, se puede consultar por ejemplo https://diego.com.es/ataques-xss-cross-site-scripting-en-php o también https://ctf101.org/web-exploitation/cross-site-scripting/what-is-cross-site-scripting o también https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.md

*__Cross Site Request/Reference Forgery (CSRF)__ (https://www.owasp.org/index.php/Cross-Site_Request_Forgery_(CSRF): Esta vulnerabilidad es una evolución de los XSS donde un cliente anómalo se hace pasar por un cliente legítimo (utilizando datos parciales del mismo) para "robarle" la sesión. Esta vulnerabilidad está presente en formularios. Así pues, cuando estos se envían al servidor, es necesario asegurarse que la petición es legítima y debemos asegurarnos que el cliente ha realizado la petición, realizando los pasos previos necesarios. La forma más común para eliminar esta vulnerabiliad o, al menos, mitigarla, es la inclusión de tockens dinámicos. En los actuales FrameWorks, suelen incluirse mecanismos para añadir esta protección a los formularios, de una forma muy sencilla. En algunos, por ejemplo, Laravel (utilizado para desarrollo de aplicaciones en PHP), basta añadir una etiqueta a la plantilla del formulario, para que se añada al mismo el tocken anti-csrf. Un ejemplo muy básico de en qué consiste se puede ver en https://ctf101.org/web-exploitation/cross-site-request-forgery/what-is-cross-site-request-forgery/

NOTA: Existen diferents técnicas XSS según el lugar donde se consiga inyectar el código deseado. Algunas de ellas son: DOM Cross Site Scripting (DOM XSS), Cross Site Flashing (XSF), Cross Frame Scripting (XFS), Cross Zone Scripting (XZS), Cross Agent Scripting (XAS), Cross Referer Scripting (XRS), etc

*__Local File Inclusion (LFI):__ Estos tipos de ataques se aprovechan de alguna vulnerabilidad concreta en el servidor para poder subir en él ficheros (normalmente binarios) y así ejecutarlos desde el propio servidor. De esta manera se podría ejecutar, por ejemplo, un visor de logs, o un servidor ssh/consola Meterpreter, o un simple descargador de otros programas más avanzados (alojados remotamente en un servidor propiedad del "hacker" de tipo FTP o HTTP), etc. Un ejemplo de código PHP vulnerable se puede encontrar aquí: https://en.wikipedia.org/wiki/File_inclusion_vulnerability

NOTA: Even without the ability to upload and execute code, a LFI attack can be done because an attacker can still perform a __Directory Traversal / Path Traversal__ walk using it as a LFI attack as follows: http://example.com/?file=../../../../etc/passwd In this example, an attacker can get the contents of the "/etc/passwd" file that contains a list of users on the server. Similarly, an attacker may leverage the Directory Traversal vulnerability to access log files (for example, Apache access.log or error.log), source code, and other sensitive information. This information may then be used to advance an attack. To prevent this, you should effectively filter any user input. More information in https://www.acunetix.com/websitesecurity/directory-traversal/ An example of this attack can be seen here: https://ctf101.org/web-exploitation/directory-traversal/what-is-directory-traversal/

*__Command Injection__ (https://www.owasp.org/index.php/Command_injection): Estos tipos de ataques tienen como objetivo la ejecución de comandos arbitrarios en el sistema operativo host a través de una aplicación vulnerable. Estos ataques son posibles cuando una aplicación pasa datos inseguros proporcionados por el usuario sin una debida validación (formularios, cookies, encabezados HTTP, etc.) a un shell del sistema (por tanto, los comandos del sistema operativo proporcionados por el atacante generalmente se ejecutan con los privilegios de la aplicación vulnerable). Por supuesto, existen herramientas que automatizan esto como por ejemplo, **Commix** (https://github.com/commixproject/commix) Ejemplos de aplicaciones vulnerables se pueden encontrar en https://www.owasp.org/index.php/Command_Injection y un ejemplo en https://ctf101.org/web-exploitation/command-injection/what-is-command-injection

*__SQL Injection__ (https://www.owasp.org/index.php/SQL_Injection): Este tipo de ataques permiten acceder y manipular la BBDD subyacente tras la aplicación web. La idea es modificar las consultas que hace la aplicación a la base de datos, aprovechando las entradas de usuario a la aplicación, para poder obtener y/o modificar datos que a priori no deberían poderse consultar/alterar. En este sentido, hay que tener en cuenta quelos motores de BBDD, tienen sus propias tablas para almacenar metadados (por ejemplo, las bases de datos que hay, qué tablas tiene cada BBDD, las columnas y tipo de cada una, información de usuarios, etc), las cuales serán el objetivo más goloso, generalmente. Un ejemplo es https://ctf101.org/web-exploitation/sql-injection/what-is-sql-injection

Como ejemplo, explicaremos cómo se realizaría un SQLi de tipo UNION. En este tipo de ataque lo primero que debe hacerse es negar la consulta original, para que esa no sea la que devuelva la información; por lo tanto, se debe añadir una condición que siempre sea falsa, así por ejemplo: *SELECT ID, Nombre FROM Clientes WHERE ID=1 AND 1=0* A partir de aquí, comprobamos cuántas columnas devuelve la consulta original. Esto es importante, porque los SQLi deben devolver exactamente el mismo número de columnas. Para ello, vamos probando mostrando un valor constante, luego dos, etc. hasta que veamos que tenemos un resultado.

http://dominio/index.php?id=1 AND 1=0 UNION SELECT 1
http://dominio/index.php?id=1 AND 1=0 UNION SELECT 1,2
...

Seguidamente hay que intentar obtener información de los nombres de tablas que hay en la BBDD, luego de los nombres de columnas por cada tabla y, una vez tengamos esa información, podremos ir extrayendo aquello que necesitemos. Dependiendo del motor de BBDD que estemos atacando, podremos hacer uso de funciones predefinidas que lleve incorporadas; por ejemplo, en MySQL existen las funciones *user()* (que devuelve el usuario que ejecuta las consultas en la aplicación) o *database()* (que devuelve el nombre de la base de datos usada), etc: *http://dominio/index.php?id=1+and+1=0+union+select+1,2,user(),database()*

Otro escenario común es utilizar un consulta para validar la autentificación de un usuario. Se consulta en la BBDD si el nombre de usuario y password existen y, si devuelve un registro, entonces se le permite el acceso. Si alguna de estas variables es vulnerable a un SQLi, se puede forzar la consulta para que siempre devuelva algo, de manera que podremos saltar el login de la aplicación.

NOTA: Hay que tener presente que la SQL injection se hará con los permisos que disponga el usuario de la BBDD con el que se haya ejecutado la consulta en cuestión.

Por supuesto, existen herramientas que automatizan esto. Por ejemplo, **SQLMap** (http://sqlmap.org) , nos permite automatizar los SQLis y, una vez detectado un parámetro vulnerable, nos permite extraer información de una manera muy cómoda. Incluso nos permitirá abrir una shell en el servidor, si esto es posible con el usuario actual, subir ficheros, realizar elevación de privilegios, etc.

Para prevenir los SQLi, se deben parametrizar las consultas y es necesario filtrar y comprobar el valor de las entradas. También es muy importante restringir al máximo los permisos del usuario con el que la aplicación se conecta a la BBDD y, por supuesto, para cada BBDD, utilizar un usuario diferente. Esto es muy importante y, aunque es una mala práctica usar una misma instancia del motor de BBDD para diferentes aplicaciones, es un escenario muy común. Por este motivo, si un usuario de la BBDD tiene permisos sobre varias BBDD de diferentes aplicaciones y una de éstas presenta esta vulnerabilidad, el atacante podría obtener los datos del resto de aplicaciones. Además de esto, es muy importante ocultar los errores provocados por consultas en BBDD; esto es porque la forma de detectar un SQLi es intentar forzar un error (la típica ‘) y, una vez se comprueba que existe, si se muestra al usuario, éste puede extraer información, como el motor de BBDD usado, etc., con lo que se le facilitará la realización del ataque. Más consejos hay en
https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.md

NOTA: D'atacs "injection" n'hi ha de diferents tipus:
"**LDAP Injection**" (https://www.owasp.org/index.php/LDAP_injection)
"**XPATH Injection**" (https://www.owasp.org/index.php/XPATH_Injection)
"**Log Injection**"/"**Log Poisoning**" (https://www.owasp.org/index.php/Log_Injection), etc

NOTA: Para encontrar más recursos (libros, documentacion, "cheatsheets", herramientas, webs, cursos, laboratorios, etc) sobre cómo implementar estos ataques y más, se puede consultar https://github.com/infoslack/awesome-web-hacking .
NOTA: A https://owasp.org/www-community/vulnerabilities es troba un recull oficial més complet i exhaustiu dels diferents tipus de vulnerabilitats web existents i les maneres d'evitar-les/protegir-les/eliminar-les.

D'altra banda, per estar al dia de les diferents vulnerabilitats concretes que es van descobrint a diari es poden consultar els següents butlletins informatius/bases de dades:

*__CVE de Mitre__ (http://cve.mitre.org/cve ; https://www.cvedetails.com). Es basa en l'enviament de vulnerabilitats descobertes per empreses informàtiques reconegudes, tan de tipus generalistes (Microsoft, Apple, Google, Amazon, Cisco, etc) com de l'àmbit de la ciberseguretat. Aquesta base de dades serveix per omplir una altra base de dades, la **NVD** ("National Vulnerability Database"), que és la base de dades gestionada pel govern nord-americà. (https://www.us-cert.gov/ncas/alerts ; https://nvd.nist.gov ; https://kb.cert.org/vuls). En qualsevol cas, cada vulnerabilitat ve identificada pel seu codi numèric "CVE" ("Common Vulnerability and Exploit"), d'allí el nom de la base.

*__Exploit-DB__ (https://www.exploit-db.com ; https://github.com/offensive-security/exploitdb). Es base en l'enviament de la comunitat. Per realitzar recerques es pot utilitzar la seva interfície web o també la seva comanda de terminal *searchsploit*

NOTA: Exemples d'ús de la comanda searchsploit són els següents:

```
searchsploit "linux Kernel" #Example
searchsploit apache mod_ssl #Other example
searchsploit -m 7618 #Paste the exploit in current directory
searchsploit -p 7618[.c] #Show complete path
searchsploit -x 7618[.c] #Open vi to inspect the exploit
searchsploit --nmap file.xml #Search vulns inside an nmap xml result
```

Altres bases de dades (que recullen vulnerabilitats de les anteriors i d'altres orígens) són:

https://www.vulncode-db.com
https://vuldb.com
https://cxsecurity.com
https://sploitus.com
https://seclists.org/bugtraq ; https://seclists.org/fulldisclosure
https://packetstormsecurity.com
https://www.securityfocus.com/bid
https://www.symantec.com/security-center/threats
https://vulners.com/search?query=order:published ; https://vulners.com/help
https://exploits.shodan.io
https://securitytrails.com/blog/what-is-cve
https://wpvulndb.com (especializada en vulnerabilitats Wordpress)
https://securitytrails.com/blog/top-exploit-databases (llista de exploits més que de vulnerabilitats)
https://securelist.com (portal de notícies sobre ciberseguretat on s'informa de noves vulnerabilitats)

Escaneig de vulnerabilitats (amb Nmap)

Existeixen varis escanejadors de vulnerabilitats que, aprofitant la informació de les bases de dades anteriors, el que fan és realitzar exploracions contra un determinat objectiu indicat per tal d'intentar detectar si aquest objectiu pateix alguna de les vulnerabilitats registrades en alguna de les bases de dades emprades. Aquestes eines poden ser molt útils, per tant, per tenir la informació necessària per, a partir de l'informe obtingut, saber escollir l'exploit adient per la vulnerabilitat detectada (i així poder "entrar" finalment a la màquina-víctima). D'entre els escanejadors de vulnerabilitats genèrics més importants podem destacar:

Nikto2 (https://cirt.net/Nikto2)
Greenbone SE (https://www.greenbone.net/en/technology) -Abans conegut com OpenVAS -
ZAP (https://www.zaproxy.org) -A més de proxy HTTP també és escanejador de vulnerabilitats-
Wpscan (https://wpscan.com/wordpress-security-scanner) -Específic per vulnerabilitats Wordpress-

De totes formes, a més dels anteriors, un programa que també es pot fer servir com a escanejador de vulnerabilitats és el nostre conegut Nmap. Nmap incorpora un motor d'scripting propi anomenat NSE ("Nmap Scripting Engine", https://nmap.org/book/nse.html) que proporciona la capacitat d'interpretar scripts escrits en el llenguatge Lua i que fan ús de llibreries especialitzades en l'automatització de tasques de descoberta i enumeració de xarxa, d'escaneig de ports/SOs/vulnerabilitats/backdoors i fins i tot d'explotació de vulnerabilitats. De fet, el propi Nmap incorpora "de sèrie" un conjunt d'scripts ja prefabricats llestos per fer servir (per saber on es troben ubicats, es pot executar la comanda *locate *.nse*, per exemple (a Fedora estan dins de la carpeta "/usr/share/nmap/scripts"). Concretament, per fer-ne ús d'aquests scripts podem afegir alguns dels següents paràmetres a la comanda *nmap -Pn -A -p nºportAProvar x.x.x.x* :

--script tipusScript, unaltreTipus,... : Executa tots els scripts prefabricats que siguin del tipus indicat. Els tipus predefinits són:

discovery : Scripts dissenyats per descobrir més informació sobre la xarxa indicada
broadcast : Scripts dissenyats per trobar nous hosts a la xarxa
vuln : Scripts dissenyats per trobar vulnerabilitats al host indicat
auth : Scripts dissenyats per trobar vulnerabilitats al host indicat relacionades amb diversos mètodes d'autenticació
dos : Scripts dissenyats per trobar vulnerabilitats a possibles a atacs DoS
fuzzer : Scripts dissenyats per trobar vulnerabilitats a atacs "fuzzing" (és a dir, per saber com respón a entrades inesperades/aleatòries per part del client)
malware : Scripts dissenyats per detectar la presència de malware al host indicat
brute : Scripts dissenyats per provar contrasenyes per força bruta contra el host indicat
exploit : Scripts dissenyats per explotar una vulnerabilitat concreta al host indicat
external : Categoria genèrica que indica que l'script envia tràfic a un tercer (un SGBD...)
intrusive : Categoria genèrica que indica que l'script pot afectar greument el host
safe : Categoria genèrica que indica que l'script no afectarà al funcionament del host
default : Selecció d'scripts més comuns. Equivalent a escriure el paràmetre *-sC*

NOTA: Es poden escriure expressions booleanes amb els operadors *and*,*or* i *not* per triar tipus d'scripts de manera més completa (un exemple seria "(default or broadcast) and not ssh-*")

--script nomScript,unaltreScript,... : Executa l'script/s concret/s indicat/s. Es poden escriure comodins ("*") per indicar varis scripts a la vegada. Si algun d'aquests scripts necessités d'algun paràmetre per funcionar, el seu nom<->valor respectiu es pot indicar amb el paràmetre addicional *--script-args nom1=valor1,nom2=valor2,...* (o bé *--script-args-file /ruta/fitxer* , on "/ruta/fitxer" representa un fitxer on estan escrits les parelles nom<->valor necessàries)

--script-help nomScript : Mostra l'ajuda per l'script concret indicat (funcionalitat, ús, trucs...). La llista completa de tots els scripts oficials i la seva respectiva pàgina d'ajuda es pot trobar a https://nmap.org/nsedoc

--script-updatedb : Actualitza els scripts disponibles, incorporant-ne de nous si n'hi hagués (bé perquè els haguem afegit nosaltres manualment dins de la carpeta d'scripts o bé perquè s'hagin afegit al repositori remot oficial, descarregant-se en aquest cas).

La taula següent mostra alguns exemples d'scripts NSE comuns (una descripció més detallada es pot trobar mitjançant el paràmetre *–script-help* mencionat anteriorment, la qual indicarà la URL exacta sota https://nmap.org/nsedoc on es troba la descripció més detallada de l'script en qüestió, incloent també una explicació de cadascun dels seus paràmetres (obligatoris o opcionals) i exemples d'ús

Nom	Explicació
ip-forwarding	Detecta si el dispositiu remot té el reenviament IP o "connexió a Internet compartit" habilitat (això ho aconsegueix enviant un paquet ICMP "echo-request" a un objectiu determinat usant el destí indicat com a porta d'enllaç predeterminada).
sniffer-detect	Comprova si el destí indicat (que cal que estigui a la pròpia LAN) té la seva targeta de xarxa en mode promiscu
dhcp-discover	Descobreix els servidors DHCP presents a la xarxa local (això ho aconsegueix enviant sol·licituds DHCPINFORM per obtenir tots els paràmetres de configuració però sense assignar-se cap adreça nova)
whois-ip	Intenta recuperar informació administrativa sobre la xarxa a la què pertany la IP indicada
whois-domain	Intenta recuperar informació administrativa sobre el nom de domini indicat
asn-query	Relaciona l'adreça IP indicada amb el seu números AS
dns-brute	Intenta enumerar els subdominis DNS existents provant de resoldre'ls contra una llista de subdominis comuns.
dns-recursion	Comprova si el servidor DNS indicat permet consultes recursives
http-put	Penja un fitxer local a un servidor web remot mitjançant el mètode HTTP PUT. Cal especificar el nom del fitxer i la ruta URL amb els arguments respectius
http-methods	Esbrina quins mètodes HTTP estan disponibles en un servidor HTTP (enviant una petició OPTIONS)
http-enum	Enumera els directoris utilitzats per aplicacions web populars per tal de descobrir-ne la presència. Compte perquè la prova estàndard inclou proves de més de 2000 ruta, cosa que significa que els logs del servidor web tindran més de 2000 entrades que no s'han trobat (HTTP 404)
http-brute	Realitza un atac de força bruta de contrasenyes amb autenticació bàsica http, digest i ntlm
http-title	Mostra el títol de la pàgina predeterminada d'un servidor web
http_robots.txt	Mostra el contingut de l'eventual fitxer "robots.txt" trobat al/s servidor/s web indicat/s
http-headers	Realitza una sol·licitud HEAD per a la carpeta arrel ("/") d'un servidor web i mostra les capçaleres HTTP retornades
http-slowloris	Realitza un atac DoS de tipus Slowloris
ssl-cert	Mostra el certificat TLS del servidor indicat.
ssl-enum-ciphers	Inicia diferents connexions TLS amb el destí indicat, provant cada vegada un xifratge o un compressor nou mentre es registra si el destí l'accepta o el rebutja. El resultat final és una llista de tots els xifrats i compressors que un servidor accepta, classificats per fortalesa.
ssl-heartbleed	Detecta si el servidor és vulnerable al bug Heartbleed d'OpenSSL
smb-brute	Intenta endevinar combinacions de nom d'usuari/contrasenya SMB.
smb-os-discovery	Intenta determinar el sistema operatiu, nom, domini, grup de treball i hora actual d'un servidor SMB
smb-enum-shares	Intenta llistar els recursos compartits obertament per un servidor SMB. Això és interessant per un atacant perquè pot haver-hi fitxers privats compartits o, si s'hi pot escriure, pot ser un bon lloc per deixar caure un troià o infectar un fitxer que ja hi és.
smb-enum-users	Intenta enumerar tots els comptes d'usuari que existeixen en un sistema Windows remot, amb tanta informació com sigui possible
*smb-vuln-**	Comprova l'existència alguna vulnerabilitat relacionada amb servidors de tipus SMB d'entre un conjunt conegut donat (i l'explota, si és el cas)
mysql-audit	Comprova que la configuració de seguretat del servidor MySQL compleixi el document de referència CIS MySQL
mysql-info	Identifica detalls d'un servidor MySQL com ara el nº de protocol i versió, l'identificador de fil, l'estat, les funcions i la contrasenya.
mysql-databases	Intenta enumerar les bases de dades existents en un servidor MySQL
mysql-users	Intenta enumerar els comptes d'usuari existents en un servidor MySQL
*mysql-vuln-**	Comprova l'existència alguna vulnerabilitat relacionada amb servidors de tipus MySQL d'entre un conjunt conegut donat (i l'explota, si és el cas)
smtp-enum-users	Intenta enumerar els comptes d'usuari existents en un servidor SMTP mitjançant peticions VRFY, EXPN o RCPT TO
*smtp-vuln-**	Comprova l'existència alguna vulnerabilitat relacionada amb servidors de tipus SMTP d'entre un conjunt conegut donat (i l'explota, si és el cas)

Un complement a les eines anteriors, especialment en el cas de voler atacar servidors web, són els anomenats "busters". Aquestes eines s'encarreguen de trobar subcarpetes i/o fitxers amagats sota les rutes que hi puguin haver en les URLs oferides pel servidor web, de manera que poguem trobar material inicialment no pensat per ser publicat però que ho estigui de forma involuntària. La manera de funcionar d'aquestes eines normalment es basa en anar repassant una llista de paraules ("diccionari") que contindrà els noms més habituals de carpetes o fitxers que es poden trobar en un servidor web ("/backup", "wp-login.php", etc) per veure si efectivament hi són (normalment buscant-hi també variacions ("/backup.xxx", /b4ckup", etc). Exemples d'aquestes eines (entre d'altres més antiquades com "Dirb" o "Dirbuster" són:

Wfuzz (https://github.com/xmendez/wfuzz)
Fuff (https://github.com/ffuf/ffuf)
Gobuster (https://github.com/OJ/gobuster) -Inclou també recerques de subdominis-

CTFs

Un CTF ("Capture the flag") és una sèrie de desafiament informàtics enfocats a la seguretat que pretenen posar a prova els nostres coneixements i possibilitar d'aprendre noves tècniques. Podeu trobar més informació a https://ctfd.io/whats-a-ctf

Se solen classificar en dos grans tipus: els CTFs "Jeopardy" i els d'"Atac i defensa". Els primers són una competició on diferents aspirants han de resoldre un conjunt de problemes abans que els demés (o la quantitat major d'ell en un determinat temps finit). Each challenge is designed so that when the competitor solves it, a small piece of text or "flag" is revealed. The flag is then submitted to a website or scoring engine in exchange for points. The amount of points rewarded is typically relative to the perceived difficulty of the challenge. Se solen dividir en les següents categories (podeu trobar més informació a https://ctf101.org) :

*Anàlisi forense: cal descobrir la informació que guarden diferents tipus d'imatges de memòria RAM, discos durs o captures de xarxa.
*Criptografia: cal desxifrar un determinat missatge secret
*Esteganografia: cal descobrir el missatge que s'amaga dins d'una fotografia, so o vídeo
*Explotació: cal descobrir una vulnerabilitat i explotar-la adientment. Pot ser binària, web, etc
*Enginyeria inversa: cal esbrinar com funciona un determinat software (de Windows o Linux) a partir de l'anàlisi del seu contingut binari mitjançant el seu desamblatge estàtic i depuració dinàmica
*Reconeixement : cal trobar la informació demanada a partir de diferents llocs d'Internet
*Programació: cal desenvolupar un programa/script que realitzi una determinada tasca

Als CTFs d'"atac i defensa", teams are each given the same set of vulnerable server software. Teams are to setup & audit this software before the competition. At the start of the competition, teams will connect their servers to an isolated network to join the CTF. Within this network, teams will launch attacks against each others servers hoping to exploit the vulnerabilities they've found. Likewise, teams will need to properly patch their software so that it is protected against these exploits and functions normally. Teams receive points for extracting flags, properly defending their flags, and keeping their servers operating normally.

Participant als CTFs es poden adquirir diferents coneixements, com ara:

-Manipulació i decodificació de HTML/CSS/JS
-Sintaxi i elements bàsics de llenguatges de programació/scripting com: Python, Bash, Perl, C, etc
-Sintaxi i elements bàsics d'Assembler juntament amb eines de depuració i enginyeria inversa
-Protocols de xarxa a tots els nivells OSI
-Sintaxi i limitacions SQL
-Criptografia pràctica tant simètrica com asimètrica (GPG, TLS ...)
-Linux (escalada de privilegis, crides al sistema, càrrega de llibreries, fitxers de configuració...)
-Comandes multidisciplinars (*tr, nc, tcpdump/tshark, strings, base64, xxd*, etc)
-Eines específiques (Hashcat, Radare2, Bettercap, Metasploit, BeEF, ZAP, StegHide, forenses, etc)
-...

A continuació es llista un seguit d'enllaços a diferents CTFs genèrics que estan oberts públicament tot l'any (també coneguts com "wargames" en contraposició als CTFs que només s'obre durant uns certs dies, moltes vegades associats a la celebració en paral.lel de conferències de seguretat...per estar a l'aguait de quan se celebren aquests darrers es pot consultar https://ctftime.org, que és una agenda de CTFs mundial):

https://w3challs.com/challenges
https://www.root-me.org/challenges
https://ringzer0team.com/challenges
https://www.newbiecontest.org/index.php?page=challenges
https://www.enigmagroup.org/pages/challenges
https://hack.me/explore
https://www.hacksplaining.com/lessons
https://www.hackthissite.org/missions
https://defendtheweb.net
https://pwnable.tw/challenge
http://pwnable.kr/play.php
http://smashthestack.org/wargames.html
https://www.hellboundhackers.org
http://www.try2hack.nl
http://www.gameofhacks.com
https://backdoor.sdslabs.co/challenges
https://avatao.com
http://hackers.udg.edu/ctf
https://www.revolutionelite.co.uk (reptes detectivescos)
https://legacy.hackerexperience.com (videojoc)
http://www.pwnadventure.com (videojoc)
https://www.hacking-lab.com (cal descarregar una iso amb una VPN especial configurada per accedir al laboratori)
https://lab.pentestit.ru (cal tenir l'OpenVPN engegat per entrar a una VPN concreta)
https://365.csaw.io (cal tenir l'OpenVPN engegat per entrar a una VPN concreta)
https://public.attackdefense.com (cal subscripció)
https://labs.secadvise.com (cal subscripció)
https://www.majorleaguecyber.org (cal subscripció)
Més CTFs: http://www.wechall.net/active_sites

Enfocat a cracking: (és un forum): https://0x00sec.org
Enfocat a cracking: http://reversing.kr/challenge.php
Enfocat a Assembler: https://microcorruption.com/login
Enfocat a Assembler: https://io.netgarage.org
Enfocat a exploits: https://exploit.education
Enfocats a XSS: https://alf.nu/alert1 ; https://xss-game.appspot.com ;
https://www.google.com/about/appsecurity/learning/xss/index.html
Enfocat a criptografia: https://www.mysterytwisterc3.org/en
Enfocat a criptografia avançada: https://cryptopals.com
Enfocat a shell: http://overthewire.org/wargames/natas .
Altres CTFs interessants d'aquesta mateixa web són "hes2010", "abraxas" o "monxla"

D'altra banda, més enllà de realitzar o participar en CTFs més o menys genèrics, també existeix la possibilitat de practicar les nostres habilitats de "pentesting" contra màquines virtuals específicament dissenyades per ser vulnerables (però que no necessàriament han d'oferir aplicacions de tipus web com les que vam llistar anteriorment tipus "Dojo" o "SamuraiWTF" sinó que poden ser de qualsevol tipologia). En aquest sentit, cal mencionar **VulnHub** (https://www.vulnhub.com), el qual és un lloc web comunitari on es concentra un gran catàleg de màquines virtuals (algunes més realistes que altres, altres més senzilles que altres, algunes més enfocades a un aspecte de la seguretat que altre, etc) que, un cop descarregades i posades en marxa (mitjançant VirtualBox, normalment) ofereixen sistemes que poden ser "hackejats" i explotats de

forma legal per aprendre'n i practicar les diverses tècniques relacionades (escaneig de ports i vulnerabilitats en serveis, enumeració de directoris, atacs via SQLInjection, LFI, XSS, execució d'exploits, etc, etc, etc). Cada màquina virtual és un repte diferent (cal llegir el seu enunciat) però en la majoria de vegades es tracta de reptes "*boot2root*": és a dir, aconseguir convertir-se en l'usuari root del sistema a partir d'una escalada de privilegis després d'haver accedit (indegudament) al sistema. Moltes vegades també forma part del repte aconseguir diferents "flags", que no són més que continguts secrets ubicats en fitxers secrets, els quals generalment caldrà desxifrar d'alguna forma.

La mateixa idea però una mica "més professional" (tot i que en aquest cas no deixen descarregar la màquina vulnerable sinó que l'has d'executar "online" des de la seva pròpia plataforma) són els serveis **HackTheBox** (https://www.hackthebox.eu) i **TryHackMe** (https://tryhackme.com). Aquests dos serveis ofereixen, previ registre gratuït, també un gran catàleg de màquines per explotar, tot i que si s'hi abona una quota mensual es té accés a més màquines i altres funcionalitats extra.

NOTA: La web de TryHackMe ofereix una sèrie d"itineraris" proposats que suggereixen provar diferents màquines concretes (de més fàcil a més difícil) especialitzades en un cert àmbit d'especialització per tal d'adquirir les habilitats concretes relacionades. Concretament podem destacar:

https://tryhackme.com/module/web-hacking-1
https://tryhackme.com/room/encryptioncrypto101
https://tryhackme.com/room/ccpentesting
https://tryhackme.com/room/linuxprivesc
https://tryhackme.com/module/basic-computer-exploitation
https://tryhackme.com/module/threat-and-vulnerability-management
https://tryhackme.com/module/security-operations-and-monitoring
https://tryhackme.com/module/threat-emulation
https://tryhackme.com/module/incident-response-and-forensics
https://tryhackme.com/module/malware-analysis
https://blog.tryhackme.com/free_path

NOTA: A continuació es llisten diversos recursos interessants (consells, eines, tècniques, etc) per preparar un bon CTF:

https://www.vulnhub.com/resources
https://github.com/ctfs/resources
https://trailofbits.github.io/ctf
http://captf.com
https://resources.infosecinstitute.com/tools-of-trade-and-resources-to-prepare-in-a-hacker-ctf-competition-or-challenge
http://www.hackplayers.com/2015/09/herramientas-y-recursos-para-preparar-en-CTF.html
https://github.com/SandySekharan/CTF-tool
https://github.com/apsdehal/awesome-ctf
https://github.com/bt3gl/My-Gray-Hacker-Resources (recursos hacking en general)
https://asecuritysite.com (miscelània)
https://github.com/Gallopsled/pwntools (software específic per realitzar tasques comunes a CTFs)

NOTA: Si hom vol muntar un CTF propi (amb registre d'usuaris, reptes per nivells, estadístiques, etc) existeix software que ajuda en aquesta tasca, com ara https://github.com/ctfd7ctfd (els quals també ofereixen un servei hosted). Altre software similar és https://github.com/ucsb-seclab/ictf-framework , https://github.com/easyctf/openctf , https://github.com/facebook/fbctf o https://github.com/Nakiami/mellivora D'altra banda, si es vol generar VMs vulnerables, es pot fer servir aquest software de cara a muntar un repte "boot2root": https://github.com/cliffe/SecGen

NOTA: També és digne de menció l'existència dels "bug bunties", que són plataformes que remuneren econòmicament a les persones que trobin vulnerabilitats desconegudes fins llavors. Alguns dels més importants són:

https://www.intigriti.com/public
https://www.hackerone.com/product/bounty
https://www.bugcrowd.com/product/bug-bounty
https://www.yeswehack.com/en/faq.html
https://www.synack.com