LSI.pdf

snchez__

Legislación y Seguridad Informática

3º Grado en Ingeniería Informática

Facultad de Informática
Universidad de A Coruña

LSI

Elena Sánchez

elena.sanchezg@udc.es

LEGISLACIÓN Y SEGURIDAD INFORMÁTICA

Tema 1 y 2: Fundamentos y categorías de ataques.

GNS3 : Es un simulador, no emulador, gráfico de red con el que podemos diseñar topologías de red complejas.

Seguridad Informática incluída dentro de la Seguridad de la Información

VULNERABILIDAD
Debilidad de nuestro hardware o software, incluído el control de acceso, faltas de métodos o respuestas ante incidentes. Puede ser explotada por una amenaza.

Las debilidades son nuestras vulnerabilidades.
Posibles amenazas: malware, sniffing...

CVE - Common Vulnerabilities and Exposures - Listas de vulnerabilidades de todo tipo de plataformas que tenemos constancia. No aporta mucha información, les da un ID, una descripción y al menos una referencia pública.

NVD - National Vulnerability Database - Repositorio del gobierno de EEUU de datos de gestión de vulnerabilidades basados en estándares representados mediante el Protocolo de automatización de contenido de seguridad.

CVSS - Common Vulnerability Score System - Sistema puntuación gravedad vulnerabilidad (1-10).

CWE - Common Weakness Enumeration - Es una clasificación, en esta lista se mantienen clusters,que son clases de debilidades o vulnerabilidades.

CPE - Common Platform Enumeration - Una lista donde se registran distintos tipos de plataformas (sistemas,software,paquetes..).

OVAL -> Lenguaje que se utiliza para intercambio de información de este tipo de vulnerabilidades y listas.

Attack Vector (Network,Adjacent,Local,Physical) : Método que utiliza una amenaza para atacar un sistema.

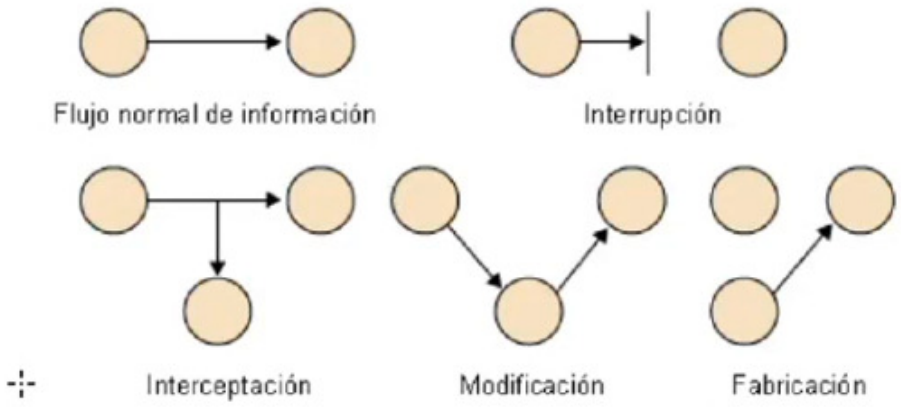
0-day: Vulnerabilidad no conocida, no sabemos cómo afrontar el problema. No tiene parches ni protección. (Ataques Zero-day: explotan una vulnerabilidad 0day).

Amenaza -> Posible peligro que podría explotar una vulnerabilidad. [Bots,Sniffers...]

Stuxnet: gusano/virus que espía y reprograma sistemas.

Ataque -> Acción que comprometa la seguridad, derivado de una amenaza. [MitM...]

CATEGORÍA DE AMENAZAS O ATAQUES



Una amenaza de seguridad puede caracterizarse modelando el sistema como un flujo de información desde una fuente a un destino.

Ataques capa 1 suelen ser de interrupción.

Ataques pasivos -> no afectan a los recursos del mismo, son muy difíciles de detectar.

Posible solución: cifrado

- **Análisis del tráfico**

Ataques activos -> alterar los recursos o afectar a su funcionamiento, objetivo es detectarlos.

- **Suplantación de identidad** (incluye otras formas de ataque activo)
- **Repetición**
- **Modificación de mensajes**
- **Interrupción del servicio**

Otra clasificación de ataques:

Sobre la Identidad
Sobre la Información
Sobre los servicios

XSS - Cross Site Scripting - Uno de los muchos ataques a aplicativo web; buscan vulnerabilidades en una aplicación web para introducir un script dañino y atacar su propio sistema.

OWASP -Open Web Application Security Project- Dedicado a la búsqueda y la lucha contra las vulnerabilidades en el software.

ATAQUES DE INTERRUPCIÓN

Deja de funcionar un punto del sistema, **detección inmediata**.

IDS → sistema que detecta intrusiones. (analiza todo el tráfico que pasa por ellos)
IPS → sistema de prevención de intrusos, pueden actuar a nivel de equipo para combatir amenazas.
SENSORES

DMZ - Zona desmilitarizada.
Es una red aislada que se ubica entre la red interna de una organización y una red externa. Red más expuesta, Los más habituales son sistemas de reglas.(??)

SNORT -> Sistema de prevención de intrusos en red (IPS) libre
Explotando una vulnerabilidad se puede provocar una **denegación de servicios**.

DoS - Ataque de denegación de servicio - Se generan mediante la saturación de los puertos con múltiples flujos de información, haciendo que el server se sobrecargue y no pueda seguir prestando su servicio.

DDoS [distribuído] -> generar un gran flujo de información desde varios puntos de conexión hacia un mismo punto de destino, la forma más común de ataque es a través de una red de bots.

Normalmente un proceso atiende a varias conexiones, entonces no sirve matar un proceso para matar una conexión, si quiero matar una conexión → tcpkill

Muchas veces no puedo diferenciar qué conexiones son buenas y cuáles malas.

Traffic Shaping: mecanismo de control del tráfico cuyo objetivo es evitar una posible sobrecarga en redes mal dimensionadas.
Mantengo todas las conexiones vivas pero gestiono el tráfico.
totalmente vinculado con QoS(quality of service)

IPTABLES uno de los paquetes para implementar un firewall
iptables gestiona, mantiene e inspecciona las reglas de filtrado de paquetes IPv4 a través de tablas.
extensión hashlimit -> Limita el flujo de paquetes considerando el origen y el destino.

Mallado de red de fibra óptica de [RECETGA]

RECETGA es una red basada en fibra oscura. Se conoce por fibra oscura a los circuitos de fibra óptica que están instalados pero que, por el contrario, están sin utilizar.

- Redundar nos proporciona seguridad. -> pej. separar un servidor en una red y otro en otra, así si falla uno seguimos teniendo el otro.

Un ejemplo de red es RTMS.

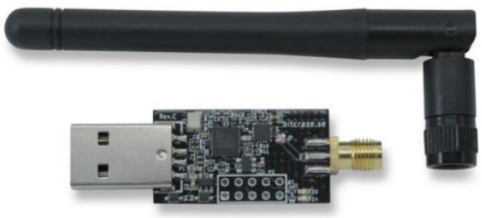
ALCATEL 1626LM usa dwdm protocol; transmiten la luz por la fibra óptica, ahí va empaquetado todo el tráfico.

Cables pasivos para que no seas detectado.

Cables UTP de solo recepcion y 'Network Taps'

Cables de solo recepción (‘receive-only cables’) o **cables ‘sniffing’** los cuales permiten a un ‘sniffer’ monitorizar el tráfico de red sin ser detectado.

BADUSBs (turtle,rubbertack...): son dispositivos hardware maliciosos que se hacen pasar por un HID “Human Interface Device”, llevan una capacidad de procesado para poder atacar una máquina.



para interceptar comunicaciones de dispositivos como teclados y ratones.

Port Security: Característica de los switches CISCO, se tiene una lista de direcciones MAC conectadas a cada puerto del switch, de esta manera sólo se le permite la comunicación en esa boca a estas direcciones. Si un dispositivo con otra dir. MAC se intenta comunicar port-security deshabilita el puerto.

estándar **802.11x** → Control de acceso a red basada en puertos.

Para que tengas conectividad de red tienes que autenticarte.

SAI/UPS - sistema de alimentación ininterrumpida.

Si hay cracking entramos en la categoría de **ataque por modificación**.

Ataque de interceptación -- ataque a la confidencialidad.

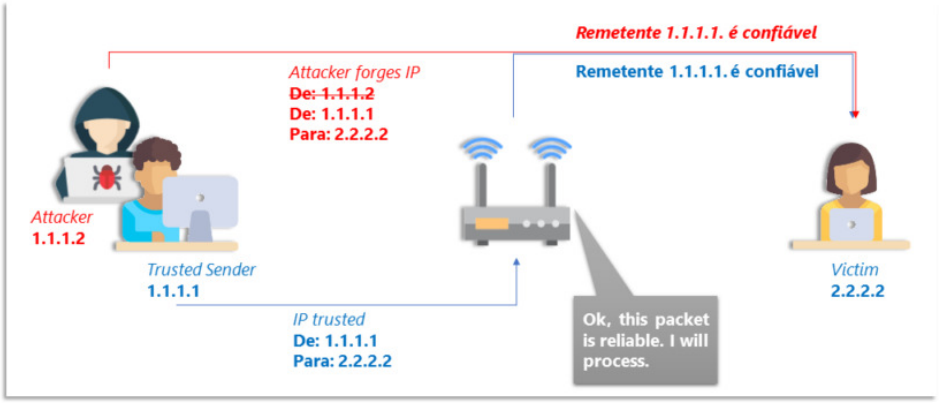
Por ejemplo: pinchar una línea, lectura de cabeceras...

Ataque de modificación -- ataque a la integridad (f.hash + firma digital principal mecanismo protección para garantizar integridad)

Por ejemplo: cambio de valores, sql injection, cracking, mod keyloggers..

Spoofing: suplantación. Hay 3 tipos:

- Mail Spoofing
- IP Spoofing (También MAC, DNS..)
- Smart-spoofing



ip spoofing

Ip Spoofing: imagina que un alumno inyecta paquetes y le cambias la Ip origen.

FAKE: falsear

- FAKE Mail..

recordatorio

IP dir 32 bits, IPv6 128 bits, MAC 48 bits

Ataque de generación o fabricación -- ataque contra la autenticidad.

fake emails, envenenamientos de dns (engaño por resolución dns), modificación de BD(sql injection), spoofing.

Herramientas interesantes de seguridad; inyectores de paquetes.
HPING3 -> permite analizar y ensamblar fácilmente paquetes TCP/IP, además de otros fines de seguridad como probar firewalls.
SCAPY -> herramienta de manipulación de paquetes.

Tema 3 (Information Gathering y La Trilogía)

INFORMATION GATHERING

Trilogía -- Host discovery, Port Scanning, Fingerprinting (SO,Servicios).

El primer paso para un ataque es la recopilación de información.

Footprinting (activo o pasivo) : Es el proceso de recogida de información. Muy importante.
Pasivo → Sin interactuar directamente con el objetivo: wikis, webs, blogs…
Los mecanismos **activos** interaccionan → todo lo que es Ingeniería Social, obtener información confidencial a través de la manipulación de usuarios legítimos.
SET - Social-Engineer Toolkit - es un marco de prueba de penetración de código abierto diseñado para la ingeniería social.

Scanning: Técnica invasiva. Interaccionamos con una máquina para ver como responde y así escanearla.

- Scanning pasivo: Como observar tráfico, nos da información valiosa como IP destino y origen, puertos y de paquetería**.**

Google Hack, Dorks: filtrar información en el buscador de Google, consultas de búsqueda que descubrieron sistemas vulnerables e información confidencial.

MALTEGO, NETGLUB → herramientas softwares.

OSINT - Open Source INTelligence - Manejo de fuentes abiertas, referencia al conjunto de técnicas y herramientas para recopilar información pública.

pastebin - servicios de copy paste, servicios que cualquiera puede usar de manera anónima para pastear info.

DNS probablemente me ofrece host discovery, ips, subdominios..

El siguiente paso es buscar los rangos de red.

NIC - Network Information Center- se encargan de llevar la gestión de cada dominio.
RIPE - redes IP europeas, se encarga de todo el direccionamiento IP (Asignación de bloques de direcciones IP..)
IP6 - nos da mucha información.
REDIRIS - organismo que gestiona la delegación del dominio ".es" en subdominios y direccionamiento. Hay jornadas entre ellas la gt y jt.

Por último la Identificación de máquinas (Hosts Discovery)

Descubrir las máquinas que hay en una red. Cuando sabemos que máquinas están "vivas" podemos realizar **Port Scanning** sólo sobre estas (o un subconjunto).
Realizarlo sobre todas las IPS del rango es muy costoso, especialmente en redes basadas en direccionamiento privado.
Tiene que ser lo más rápido y silencioso posible, también se conoce como **escaneo ping** o **descubrimiento de sistemas**.

Una forma básica de host discovery es enviar un ping a cada dir. IP de un rango y ver que dir. responden. El problema es que no todas las máquinas responden y es un proceso lento.

ping: muchas veces los filtramos, podría dar la máquina por muerta cuando está en funcionamiento.

Traceroute: puede hacer una ruta desde una máquina objetivo que me marque todos los routers por los que va pasando, seguramente los últimos serán firewalls.
Puede no funcionar si filtran el tráfico ICMP de manera que no llega la respuesta y no somos capaces de identificar el camino.

Robtex - Recolección de información sobre máquinas además de Log List.
Mira en el NIC y en el RIPE entre otras.

En los servidores web podemos ver su índice de reputación, evitando esos servidores podemos evitar páginas web que han estado involucradas en infecciones, malware, spam…

Uso de servicios DNS para recolectar información, en los servidores se guarda mucha información sobre IPv4 e IPv6.

En DNS solemos tener resolución directa e inversa.

- Servidor **primario**: mantienen de forma explícita la información de DNS, la BD de resoluciones nombre-IP..
- Servidor **secundario**: no mantienen nada, cada cierto tiempo se conectan al primario y hacen una transferencia de zona (recibir toda la info de un DNS server como si fuésemos una máquina secundaria) teniendo al final los dos la misma información.

Para gestionar un DNS y mantener la información miramos los **registros**:

- Registro A - registro de dirección, resolución directa.
- Registro AAAA - registro dirección IPv6. (Asignamos nombre a IPv6)
- Registro CNAME - registro de nombre canónico, alias.
- Registro NS - son para definir el servidor de nombres de un dominio.
- Registro MX - (mail exchange) indicamos los servidores de correo electrónico del dominio.
- Registro PTR - para resolución inversa, asignamos la IP a un nombre, contrario a A.
- Registro SOA - (start of authority) Name Server primario de la zona.

dnsenum	Nos devuelve: dir. host, name servers y servers de correo, intentará hacer una transferencia de zona
dnsmap	Registros dns por fuerza bruta o diccionarios, será la siguiente opción si no aceptan transferencia de zona ni registros PTR. Probaremos con todos los nombres del diccionario contra el servidor DNS, si alguno de ellos tiene resolución directa, nos canta todos los que aparecen. Un problema es que las peticiones a DNS suelen ser lentas, y eso en la fuerza bruta se va a notar mucho.
dnsrecon	Con -r, le pasamos un rango de IPs. Usará registros PTR, útil si no permiten transferencia de zona.. Esto nos aportará mucha información. Pero el host podría no usar registros PTR.
nslookup	Saber si el DNS está resolviendo correctamente los nombres y las IPs. Obtenemos la dirección IP conociendo el nombre, y viceversa.
dig	Realiza búsquedas en los registros DNS, a través de los nombres de servidores

DNS caché snooping

Poder investigar un servidor dns para saber si determinadas páginas están en caché.
dnssnoop -t snoop -n X.X.X.X dns -D <ficheros> (???chequear)
dnsrecon -t snoop - l <server> -D <ficheros>

- Modo **no recursivo** - Flag RD = 0.Sólo se resuelven peticiones que estén en caché.

NMAP -> realiza Host Discovery sobre cada máquina y sobre cada host “vivo” realizará Port Scanning.

Con -sP sólo realizará un ping scan.
--packet-trace permite ver los paquetes que envía nmap

Port Scanning: Testear de forma remota varios puertos para determinar en qué estado se encuentran.

Activo: se interactúa con un puerto de una máquina para ver como reacciona este.

Pasivo: observando la paquetería, si miramos las cabeceras podemos ver el puerto de origen/destino.

pej. Si vemos un puerto 80 abierto puede que tengamos un HTTP abierto, pero si cambiamos (por ejemplo, ssh) en *ssh.conf* el puerto 80, no tendríamos un HTTP si no un ssh.

"Reducir el número y complejidad de los servicios ofrecidos reduce la oportunidad de que los atacantes entren"

Un **puerto** es una **abstracción** usada para distinguir distintos canales de comunicación.
Al igual que las dir. IP identifican máquinas en una red los puertos **identifican aplicaciones** en una máquina.

En TCP/UDP la conexión se identifica → IP+ P.origen // IP + P.destino

Podemos clasificar los puertos según la **IANA** - Internet Assigned Numbers Authority

- **Well-known ports:**
  reservados del **1 a 1023** registrados con la IANA para un servicio determinado.
  Algunos sistemas exigen privilegios especiales para asociar apps a estos puertos.
- **Puertos registrados:**
  1024 a 49151.
  Registrados con la IANA, la diferencia con los well-known es que no se exigen privilegios especiales.
- **Puertos dinámicos y/o privados:**
  49152 a 65535.
  Se usan dinámicamente o bien por servicios privados de una compañía

Finger printing: no hace referencia a personas o víctimas.

- SO (**OS Fingerprinting**): proceso de recopilación de información que permite saber de forma remota el sistema operativo y versión que corre en determinadas máquinas.
  [Cada SO responde de forma diferente a una variedad de paquetes].

Por ejemplo: TTL (tamaño de ventana), en Linux, Debian BSD es 64, en Windows 128 y en Cisco 255. Esto se puede averiguar haciendo un ping y viendo el TTL del paquete de respuesta.

- **Servicios**: orientado a identificar el servicio que corre en una máquina, que está corriendo y que versión.

  Con esa información, tanto a nivel de sistema operativo, como de servicios, podemos tratar de buscar vulnerabilidades para esos productos específicos, para tratar de explotarlas.

Por ejemplo al fingerprintear un puerto 80 de un servidor web, busca saber qué servidor está corriendo, por ejemplo un apache, y que versión de apache.

Activo → Interroga a la máquina, puede ser más o menos agresivo.
Se basa en enviar paquetes a la máquina y analizar sus respuestas.

Pasivo → No interroga directamente a la máquina.
Escuchamos la red y analizamos. Puede utilizar solicitudes legítimas para obtener tráfico. pej: solicitud de página web.

nmap

Mapa Red

Representación gráfica de la red a monitorizar, vemos los elementos que la conforman y el estado de los mismos,

BÚSQUEDA/ANÁLISIS DE VULNERABILIDADES

Se puede hacer de forma manual o con herramientas como:

- **openvas**
- **nessus**

nikto, OWASP ZAP y w3af están orientadas a aplicaciones web

Exploits: Ataques que aprovechan las vulnerabilidades. (o vulnerabilidades conocidas?)

Pentesting - Test de penetración - Atacar un sistema para identificar vulnerabilidades, fallos o errores para así poder prevenir ataques externos.

Manuales y Herramientas:

- **Metasploit**: proporciona información acerca de vulnerabilidades de seguridad y ayuda en tests de penetración. (Proyecto).
- **Cobalt Strike**: herramienta de seguridad legítima que permite a los equipos de seguridad emular la actividad de los ciberdelincuentes dentro de una red.
- Fat Rat: para desarrollar troyanos.
- Repositorios de búsqueda (NVE,CVE..) **shodan**.
- SEC: report Ataques de ingeniería social por SEC. (?preguntar?)

Immunity canvas: herramienta de prueba de penetración que incluye cientos de exploits.
Core impact: plataforma de pentesting.

Payloads : son módulos de los exploits.

[también están los "encoders" : códigos de cifrado para evasión de antivirus].
Cuando definimos un payload se hace de la manera más discreta posible (ofuscación).

Post Explotación:

Procesos de ocultación.
[escalado de privilegios : explotar cierta vulnerabilidad te da ciertos privilegios en una máquina → recolección de información,acceso a otras máquinas..]

pivoting - Busca usar una máquina ya comprometida, así tenemos más información, para desde ella atacar otra distinta.

Ofuscación

Distintos métodos y técnicas para coger un binario para que un antivirus lo vea como algo bueno y no como algo malo [que sea más difícil de detectar].

Modificaciones sobre el **payload**, el binario resultante lo va mandando, así va bajando el ratio de detección de antivirus hasta que ya no lo detecta.

msfpayload -> permite generar ejecutables con un payload en su interior.

msfencode -> le pasas distintos métodos de ofuscación para convertir a los binarios.
Que pase desapercibido el payload.

Ocultación

Flags PSH,URG,ECE,CWR -> temas de control de flujo.

mmap -P0 --p 11-200 pepe.abcd.org -> -p0 que haga **host discovery**

Si el pid se incrementa en uno el puerto está cerrado.

OWASP

Proyecto de código abierto dedicado a combatir causas que hacen que el software sea inseguro. Dentro del proyecto se distinguen distintas líneas de trabajo.

Se habla de seguridad en aplicativo web.

Uno de los proyectos que tienen es el **Owasp Top 10**: Básicamente sacan una versión cada 4 años, lo que hace es que el grupo analiza e investiga aplicaciones y entrega premios del 1 al 10.

1)	**Broken access control** → saltarte mecanismos de autenticación(?)
2)	**Cryptographic failures** → fallos a nivel de cifrado de la información.
3)	**Injection**
4)	**Insecure Design**
5)	**Security Misconfiguration** → referencias a temas de malas configuraciones (cabeceras http...) incluye XXE (xml external)
6)	**Vulnerable and Outdated components**
7)	**Identification and Authentication failures.**
8)	**Software and data integrity failures.**
9)	**Security Logging and Monitu¡oring failures**
10)	**Server-Side Request Forgery** → (SSRF) hacen una especie de pivoting

Los datos son manejados por un intérprete. Si los datos no están bien controlados proporcionan problemas de seguridad.
Para eso podemos usar técnicas de **encoding**.

Ciertos caracteres que pueden ser dañinos en los intérpretes se codifican de forma equivalente.

Por ejemplo cambiar la codificación del carácter “<” ya que puede formar parte de HTML, Scripts… a codificarse como &LT.

Escapado de datos: se busca introducir carácter de escape previo a los caracteres para que no sea malinterpretado.
Como una barra invertida “\” delante del carácter comillas “ que hace que se interprete como texto y no como delimitador de final de un entrecomillado.

Una correcta validación de las entradas aumenta la seguridad.

OWASP Project Application Security Verification Standard - Lista de requerimientos y pruebas de seguridad.

Interesante la sección de guías, con pasos para verificar el nivel de seguridad de la app web.

OWASP Top Ten Proactive Controls - Describe las categorías de control y en sí control más importantes que todo arquitecto y desarrollador debería incluir al 100% en cada proyecto.

OWASP ZAP - Búsqueda de vulnerabilidades en aplicativos web (como Nikto2)

Blind Elephant - Tiene muchos plugins para gestores de contenido.

Herramienta de fingerprinting para aplicativo web. Podemos obtener mucha información de los servidores web.

Nikto2 - Herramienta de escaneo de servidores web que se encarga de efectuar diferentes tipos de actividades tales como detección de malas configuraciones, vulnerabilidades en el servidor objetivo o analizar riesgos.

webshag - Llama a NikTo2 para hacer análisis de vulnerabilidades de aplicativo web.

EnableSecurity/wafw00f - Trata de determinar si en la comunicación con un servidor hay un firewall de por medio.

Fingerprinteará e intentará decir que tipo de waf hay. Web Application Firewall (WAF)

XSS - **Cross Site Scripting** - Es un tipo de ciberataque en el cual se buscan vulnerabilidades en una app web para introducir un script dañino y atacar su propio sistema.

En 2021 se integra en injection.

wget → Permite la descarga de contenidos desde servidores web.

Red Team y Blue Team [Purple Team]

Red Team busca vulnerabilidades de la organización mientras que Blue Team intenta proteger todo.

Purple team es un extra para coordinar a los dos equipos en caso de ser necesario.

mod security: Módulo de seguridad de Apache, actúa como firewall de aplicaciones web (WAF) y su trabajo es filtrar y bloquear las solicitudes HTTP sospechosas, pudiendo bloquear ataques de fuerza bruta, vulnerabilidades de cross scripting (XSS), ataques por inyección SQL (SQLi)...

Apache es un servidor web HTTP.

En *Windows*:

- **wmic**: parejo al comando **ps** de linux.
- **ipconfig**: es como ifconfig, si ejecutamos displaydns nos dará las direcciones que se han resuelto en la máquina a lo largo del tiempo.
  ipconfig -all sería parejo a **ifconfig -a**, devolviendo el conjunto de interfaces de red que tiene la máquina.
- **service**: service list full devuelve todo los servicios en su ejecución son sus ID.
  Sería un comando parecido a **ss** en linux.
- **netstat**: igual al de linux.

Iptables Mangle: Se encargan de modificar los paquetes, para ello tienen las opciones:

- **TOS**: Type Of Service, usado para definir el tipo de servicio de un paquete y se debe usar para definir cómo los paquetes deben ser enrutados, no para paquetes que vayan hacia Internet.
  La mayoría de los routers no hacen caso del valor de este campo o pueden actuar de forma imperfecta si se usan para su salida a Internet.
- **TTL**: Time To Live, cambia el campo de tiempo de vida de un paquete. Se puede usar para cuando no queremos ser descubiertos por ciertos proveedores de servicios de Internet (ISP) que sean demasiado “fisgones”.
- **MARK**: usado para marcar paquetes con valores específicos.

OSfuscate: Permite al usuario seleccionar un sistema operativo para emular e implementa esa selección cambiando los valores del registro.

Una de las partes importantes es descubrir **rutas y firewalls**.

ping -> un gran problema es que la gran parte de las organizaciones filtran el tráfico, así que la mayoría no te van a responder.

traceroute -> va dando todas las rutas por donde va pasando. Es una manera de localizar firewalls.

Si saca ****… es porque el tráfico está filtrado, el servidor pasa de nosotros.

Si trabajamos con traceroute a nivel de TCP o UDP (-T/-U):

-T → paquetes SYN. Muestra todos los routers, firewalls…

Si salen asteriscos es porque hay algunas respuestas filtradas o la misma máquina.

En lugar de usar paquetes ICMP manda SYN con TTL (saca la ruta entera, con uno normal no por eso muestra ****)

En *windows* es tracert.

Hablamos de **firewalls**, hay de muchos tipos, por ejemplo:

- **Firewall tipo router**: es muy habitual encontrar un firewall implementado en un router.

  Access List -> en un router cisco puedo definir qué ips pueden mandar etc..
- A nivel de NAT -> Hace Nating, podemos implementar reglas de filtrado.
- **Firewall de control de estado**: A mayores comprueba el estado de todas las conexiones que se establecen y garantiza que se sigue el protocolo.
- **Firewalls transparentes**: No tienen ips, como si fuese una caja por donde pasa el tráfico.

  Si haces un traceroute no te aparece nada, porque a nivel ip no existe.
  Lo ideal es configurar un firewall en modo transparente.

  Report Diego Villar
- **Firewalls a nivel de capa de aplicación**: Normalmente los veremos con las siglas de WAF (Web Application Firewall). Hay distintos paquetes como:

  - mod Security

Se ponen entre las máquinas cliente y los servidores web, de tal forma que tratan de detectar los distintos ataques contra las aplicaciones web (además de pararlos).

- **Firewalls de nueva generación**: Trabajan en el modelo ISO/OSI en todas las capas.

nsslab.com es muy interesante porque si tienes una empresa ver estos informes cunde.

Realizan pruebas de evasión, para ver si los firewalls pueden detectar las técnicas de evasión que les aplican, solo dos lo han conseguido: **paloalto** y **watchwark**

En los firewalls tiene que estar todo **filtered** [ni IP, ni puertos abiertos/cerrados], en ocasiones algunos proporcionan otros servicios como VPN pero no es recomendable.

Más herramientas:

Creepy - pequeño script que coge fotos y si tienen geoposicionamiento en los datos te ubica las fotos en un mapa.

Metagoofil - trabaja con metadatos de ficheros que hay por internet, sacando así mucha información.

Foca - **Fingerprinting Organizations with Collected Archives** - utilizada principalmente para encontrar metadatos e información oculta en los documentos que examina.

Hay muchos temas relacionados con el estudio de la información..

Twitonomy, Tweetreach -> especializadas en twitter.

Spiders, Crapers, Scrapers and Hardening

Son aplicables a los motores de búsqueda.

- **Spidering** → recorrer todo el árbol de una web (o conjunto) y descargar toda la información.
- **Cravers** → realizan análisis semántico, un web craver recorre la web y obtiene de ella lo que le interesa.
  Google o Yahoo: recorren las webs que visitan para indexar esas páginas.
- **Scrapping** → recorre webs buscando datos concretos, un ejemplo son los típicos scrapers que buscan por determinadas webs el precio de determinados productos.
- **Hardening** → endurecimiento, vinculado a securizar algo.

Buscadores recorren árboles de los servidores web e indexan su contenido. Este sistema se suele usar en análisis de vulnerabilidades. (Proceso de Spidering)

También se usan en temas de análisis de mercados y clientes.

Fiddler - Servidor proxy para depurar código HTTP

WhatWeb - Hace fingerprinting a nivel de web.

Netcraft - Ofrece análisis de cuota de mercado de servidores y alojamiento web, incluyendo la detección del tipo de servidor web y de sistema operativo.

Tema 4: Ocultación y privacidad.

Están totalmente vinculadas, hay que tener en cuenta que la ocultación no es sólo ocultar la ip.

NAT - Network Address Translation - Que redes de ordenadores que usan IPs privadas se conecten a Internet usando una única dir.IP pública.

No ofrece ni privacidad ni ocultación.

Nateado de IP: Navegamos por la misma red pública por la que va NAT.

[Ni privacidad ni ocultación].

> Normalmente la privacidad la pierdes al ejecutar un script por la web al transmitir directamente la información requerida por este

Proxys: Máquina intermedia a la que se le hacen peticiones y te las proporciona.
En algunas organizaciones tienes que salir a través de un proxy.

Podemos establecer los mecanismos de seguridad que consideremos oportunos a todos los niveles.

Podemos navegar a través de un proxy o conectar varios, cuantos más proxys haya más difícil será el rastreo, aquí sí que obtenemos **privacidad**.

Por temas de rendimiento podemos emplear Proxys Web.

Web-Based Proxies

Sitios web que permiten acceder al contenido de 3º sin configuración en el cliente.
Proporciona **privacidad y confidencialidad** puesto que ocultan la IP del usuario y la mayoría cifran el tráfico entre usuario y proxy.
Desventajas: Algunas páginas pueden no mostrarse correctamente.

Squidt - Uno de los paquetes habituales para montar un proxy.

x_forwarded_for - Cada vez que hay un salto, en la cabecera se añade la IP origen proxy1, proxy2… Se guarda todo el camino, guardando la IP destino y de todas las máquinas por las que pasa.
ni ocultación ni privacidad.

Apache - **mod_extract_forwarded** - Vale de proxy además de servidor. Se le pueden cargar varios módulos.

remote_addr

Existen dos tipos de proxys:

- **Proxy HTTP/HTTPS** → para navegación.

- **Proxy Socks** → para todos los servicios (multiservicio).
  Puede funcionar con muchos protocolos, que también incluyen HTTP.

Hay que tener cuidado con los proxys ya que si son de dudosa fiabilidad nos pueden llegar ataques puesto que todo nuestro tráfico va a pasar por él.

Especial cuidado si no está cifrado como HTTP.

Open Proxies

El cliente debe configurar los datos, después todo es "transparente" para el usuario.

VPN

Para privacidad, un poco la misma idea que los proxys. Si en VPN no se registran ni trazan las conexiones se crea un túnel, estando más seguro.
Todo el tráfico va cifrado.

Va a abrir un túnel cifrado en Capa 3 desde tu PC al VPN. Autentificación del cliente y del servidor, por tanto, si por encima de la capa mete cualquier aplicación que no sea segura, se va a securizar en capa 3.

Si usamos VPNs normalmente el rendimiento baja.

[*Nino*] - Usar la IP de otro profesor desde su despacho, con eso no llega, le pueden detectar. Hay que cambiar la MAC a la del profe y el cableado para la salida, básicamente ocultación de cada capa.

Cuidado con los detectores de ARP Spoofing al ver el cambio de IP en la misma MAC.

- **Proxy transparente**: no da ni privacidad ni anonimato, usan X-Forwarded-For, quedando todo traceado, IP desde la que te conectas y máquinas por las que pasas. Estos proxys se suelen utilizar para temas de aceleración, caching de páginas web...

- **Proxy anónimo**: da cierto nivel de anonimato, sin X_Forwarded_For.

- **Proxy élite/high anonymous**: No usan X_Forwarded_For ni otras cabeceras poco seguras, con más grado de privacidad que los anónimos.

- **Proxies ruidosos**: Te ocultan, y usan X_Forwarded_For con información falsa creando ruido.

- **Web Proxy Server**

Proxy checker para verificar la validez de los servidores proxy.

Configuración máquina Linux.

Distintas variables de entorno:

http_proxy
https_proxy
ftp_proxy

#export http_proxy =http://server-ip:port/ → Si yo configuro estas variables, las peticiones http o los **repositorios de actualización de paquetes** me las harán a través de este proxy.

Ficheros de configu8ración:

/etc/profile: podemos definir variables que se configurarán a cada usuario cuando hacen login en las máquinas.

/etc/bash.bashrc: fichero de configuración para los usuarios que usen el intérprete de comandos bash (se puede modificar en el fichero de passwords).

$HOME/.bashrc: Es el fichero anterior pero específico para un usuario. Si a un usuario le ponemos aquí un proxy puerto, ese usuario usará ese proxy.

/etc/environment: Configura variables y lo que le pongas, incluso para procesos que no tiene ni asignados él.

Para ir a localhost o máquinas de nuestro segmento vamos directos, no por el proxy.

Si queremos configurar proxy socks modificaremos el fichero **/etc/socks.conf**

```
direct 128.0.0.1 255.255.255.255
direct x.x.x.x 255.255.254.0
sockd @=x.x.x.x
```

WIFI

crackeo de passwords - rompemos el handshake entre un cliente y un punto de acceso wifi usando diccionarios mediante fuerza bruta, capturando ese handshake.

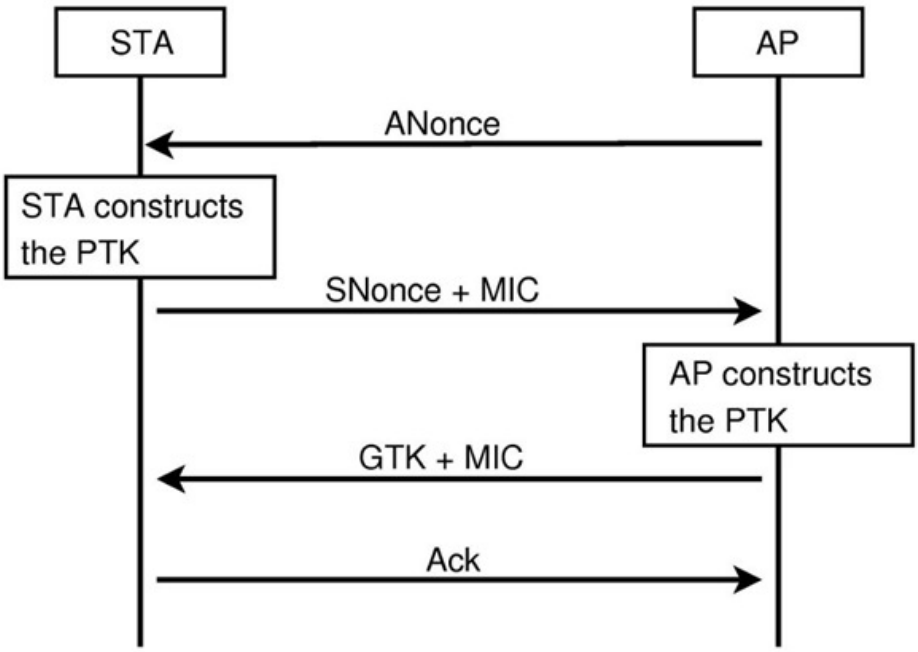
Si la clave es muy larga y con diferentes símbolos romper ese handshake mediante fuerza bruta es casi imposible.

Aircrack → herramienta para crackear que trae una serie de utilidades. Entre ellas está la opción de poner la tarjeta wifi en modo monitor (recoge todo el tráfico de la red).

Airodump-ng: captura de paquetes wireless 802.11.

Handshake

4 primeros mensajes encriptados del proceso de conexión entre el cliente y el punto de acceso wifi.



handshake que hemos robado de una máquina.
STA - Máquina cliente.
AP - Access Point.

- **PMK** (Pairwise Master Key): transformación de la clave que usamos para acceder a ese punto de acceso.
- **PTK** (Pairwise Transit Key): Encriptación para el tráfico unicast. Cadena de información.
- **GTK** (Group Temporal Key): Encriptación tráfico broadcast y multicast entre el AP y los clientes. Para cada AP hay un GTK distinto.
- **GMK** (Group Master Key) : Crea el GTK
- **ANonce**: primer mensaje del handshake, un número aleatorio.
- **SNonce**: número aleatorio de respuesta al ANonce.
- **MIC** (Message Integrity Code) : Hash del PTK.

En una **WPA** (Wi-Fi Protected Access)

1º Mensaje: AP envía al cliente su ANonce. El cliente crea el PTK porque tiene todos los datos que necesita.

2º Mensaje: El cliente envía su SNonce al AP con su MIC, la cúal sirve para que el AP reconozca que el mensaje es realmente de ese cliente.

Ahora el AP tiene todo lo que necesita para crear el PTK. (Junto las MAC y el GTK, genera su Hash y lo compara con la MIC recibida, si son iguales se permite la conexión)

3º Mensaje: El AP envía el GTK porque va a ser su nuevo cliente, este lo recibe y lo instala.

4º Mensaje: El cliente envía a el AP un ACK confirmando que todo fué OK.

PTK = PMK + ANONCE + SNONCE + MAC(AA) + MAC(SA)

Cuando robamos el handshake en modo monitor a una máquina que se enlaza a tu AP le robamos todo este tráfico.

Después de generar una lista de posibles contraseñas, empleamos aircrack-ng

Coge el handshake capturado y separa la MIC de los demás parámetros que equivalen al PTK, después se corre la lista con las posibles contraseñas y se junta con los demás parámetros y chequea si la MIC que se obtiene de la combinación es igual a la MIC original.

Este proceso lleva mucho tiempo.

WPS

"abrir" la red WiFi que genera el router durante un periodo corto de tiempo, lo normal es conectarse a través de un PIN.
Inseguro.
Se hacían ataques de fuerza bruta para recuperar la WPA/WPA2 “passphrases”.

MAC

Para enlazar con un AP también se puede usar la MAC como comprobante.
Herramientas crackeo:

Wifite - Se encarga de ver todos los AP en alcance, si es WPA u otra y busca maneras de crackearlas.

Fluxion - Herramienta Ingeniería Social. Crea un AP de manera que cuando una máquina se enlace a este punto lo redireccionamos a una página web dónde le solicitamos las credenciales de la red wifi.

Hashcat - Crackeo de contraseñas. Corre sobre CPUs y GPUs, también se ha usado con computación distribuída, usando varios PCs.

Las GPUs se han usado numerables veces para el crackeo de passwords debido a su gran potencia.

WPA/WPA2

KRAKATTACK

Ataque de reinstalación de claves, engaña al cliente para que re-instale una clave ya usada.

Crea un AP falso al que se conectará el cliente. 4 way handshake - bloqueamos el ACK del 4º mensaje y re-insertamos el 3º mensaje.

Se hace un RST de los números aleatorios, se puede inyectar,descifrar o falsear la paquetería.

WAP3

Ataque **dragonblood**: acceder desde el canal lateral, sin necesidad de conocer la contraseña.

[parcheado]

Características:

- Renueva el handshake, usando el algoritmo Dragonfly, que utiliza Diffie-Hellman con curvas elípticas.
- Las claves que se generan entre el cliente y el AP tienen una ventana temporal muy pequeña, es decir, se restablecerán unas nuevas en periodos de tiempo muy pequeños.

  Esto dejaría prácticamente inútiles los ataques de reinyección (kackattack) y cualquier crackeo del handshake.
- Se fortalece a nivel criptográfico la longitud de claves (128 bits → 192 bits).
- Ya no es tan sencillo hacer ataques de deautenticación contra las plataformas que están enlazadas a un AP debido a que las tramas de gestión están cifradas.

Borrado seguro

Con rm se puede recuperar, el software de análisis forense es capaz de recuperar información de RAM, de dispositivos de almacenamiento..

Herramientas:

- srm → Secure Remove
- Distros booteables como **dwan**.

  Sobreescriben varias veces dónde está la información, así evitamos que queden trazas.
- disk dump → sobreescribe un fichero encima de otro.
- sfill → borrado en discos duros del espacio libre (no asignado)

  En los discos duros pudo existir información almacenada con anterioridad. sfill lo elimina de manera segura.
- sswap → borrado seguro del espacio de swapping y particiones swapping.
- snem → borrado seguro de la RAM.

Con el **accounting** activado queda registrada toda la información sobre lo que se ha hecho en la máquina, los ficheros de **log** dejan traceado desde una máquina.

-history

DNS Leaks

Fugas de DNS. Nos proporciona consejos para evitar las fugas, como el uso de VPNs y configurar los firewalls para que el tráfico que no vaya por esa VPN sea dropeado.

Lightbeam
Addon para firefox, nos permite ver como usan nuestra información gráficamente.

¿Cómo buscan los clientes los AP?

Primeramente el cliente envía paquetes “Probe Request” de tipo broadcast a cualquier AP que se encuentre cerca, la finalidad es la de localizar y obtener los SSID de los AP.
Los AP responden con paquetes “Probe Response” que contienen información sobre él.
A continuación el cliente inicia el proceso de autenticación y asociación.

Si ponemos nuestra tarjeta en modo monitor podemos capturar esos Probe Request, viendo que WiFis está intentando enlazar una máquina y perfilar una máquina en función de estas wifis.

Las WIFIs se pueden geolocalizar (**wigle**)

Se puede ocultar la WiFi al público (que no emita SSID), esto es llamado defensa por ocultación. Pero te pueden detectar debido al AP y esnifando paquetería.

SSID (identificador de red SSID) es el nombre público de una WLAN que sirve para diferenciarla de otras redes.

Redes de anonimato

Muchas siguen el concepto de proxy pero enlazando máquinas, quedando oculto.
Gran parte basadas en **onion routing**.

> Se va a degradar el rendimiento porque la comunicación no va directa entre cliente-servidor, sino que pasa por varias máquinas encriptando la información.

- Freenet
- Invisible Internet Project (I2P)
- Tor
- JAP
- Retroshare
- Tarzan

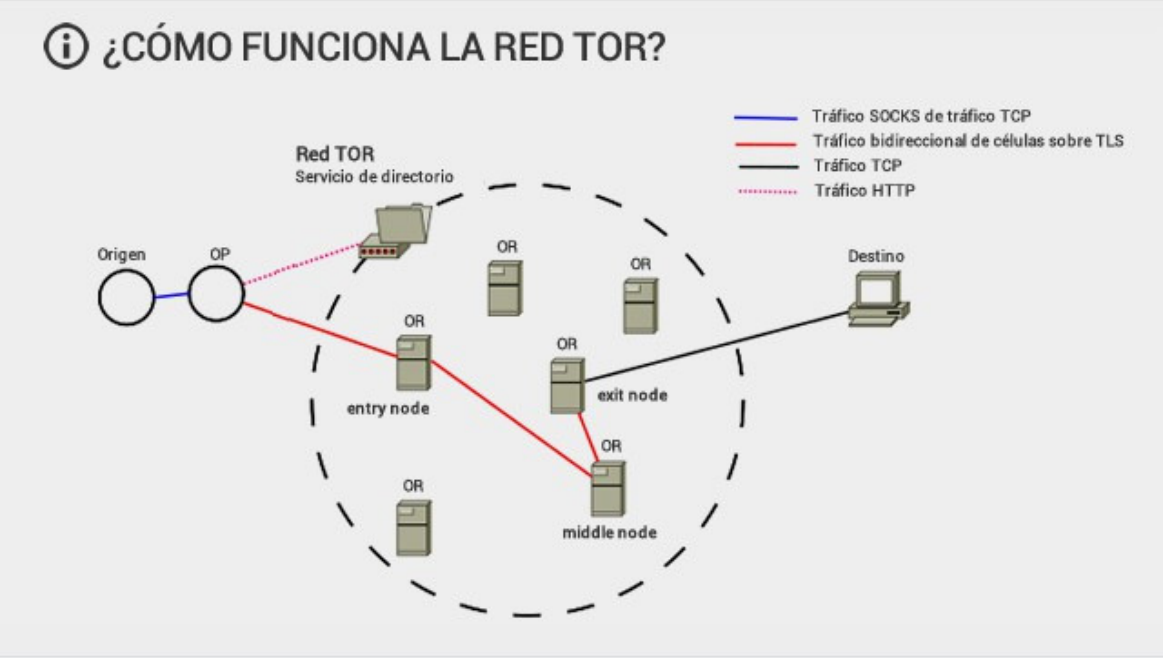
TOR [de onion project].

Entra en funcionamiento el **onion proxy**, el cual adapta la paqueteria de mi maquina para encaminar el tráfico.
Lo habitual es que esté integrado en nuestra máquina.

Los **onion routers**, por defecto se seleccionan 3 nodos, así cuando se abre la conexión no se hace directamente, la única ip que ven es el exitnode de la red tor.

> (El último nodo que se selecciona = exit node).

Los distintos nodos tienen una clave pública y una privada.
Por eso con 3 nodos no se puede reconstruir el camino origen.



OP(Onion Proxy) → Coge paquetería normal y habla con Tor.

> Normalmente IP Origen y OP son lo mismo, estando el OP normalmente montado en nuestra máquina.

OR(Nodos Tor) → Hay varios distribuídos por internet, podemos convertir nuestra máquina a uno de ellos.

Hay un servicio de directorios, que es público. Tiene información de los nodos que forman la red Tor con sus distintos parámetros.

Un nodo sólo conoce al antecesor y al sucesor.

entry node: no sabe a dónde va.

el siguiente ya no sabría de dónde vendría.

No es un cifrado extremo a extremo. [Examen]

Conexiones TLS securizadas. Una conexión TLS es lo que normalmente usamos al establecer una clave HTTPS (o TLS), securizando y cifrando mediante una clave de sesión.

Así cada par de máquinas tendrá una conexión TLS donde todo va cifrado.

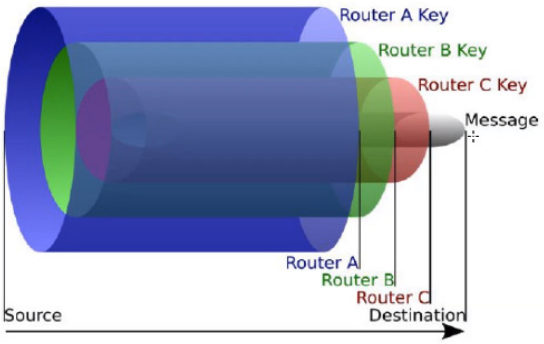
Entre la **salida y destino no va securizado**, ya que si hacemos una conexión de la máquina de salida a la destino por HTTP ahí va a haber tráfico HTTP no seguro. Por tanto este tráfico es tal cual el de la petición original.

El último nodo tiene la paquetería tal cual, tanto de ida como de vuelta.

A mayores para securizar el nodo de origen establece dos claves con cada uno de los nodos (entrada, medio y salida). Una clave para el tráfico de ida y otra para el de vuelta.

Funciona de forma que una vez que la máquina origen y el entry node tienen ya una conexión securizada la máquina origen genera un par de claves (para un lado y para otro) y las cifra con la clave pública del nodo de entrada, y la manda por el enlace TLS.

Así estas claves solo pueden descifrarlas los correspondientes nodos, de esta manera tendré claves establecidas, a mayores de la securización TLS, con los nodos que forman mi circuito de comunicación.



El mensaje se cifra con cada **Key**, de aquí el ejemplo de las capas de una cebolla.

El azul es la del nodo de entrada, de forma que se lo mandamos y lo descifra, lo mismo para el verde que es la de intermedio y finalmente el rojo lo descifra el nodo de salida.

Queda el mensaje como se creó originalmente para entregarlo a la capa de salida.

Además de Proxys, VPNs y redes como Tor, no solo llega con esto para nuestra privacidad.

Scripts de las webs - Si un javascript te pide la IP u otros datos y se la das, ya estás perdiendo todo el anonimato.

Tor browser nos protege de ello (integra el onion proxy), además dificulta los fingerprinting para obtener huellas digitales.

Metrics → Información sobre los onion routers de la red Tor.

Tails → SO.

Stem → librería python para realizar apps torificadas.

Orbot → proyecto que proporciona anonimato en Internet para usuarios Android.

Tortilla → crea interfaces de red virtuales que podemos asignar a nuestras máquinas virtuales.

Meterle un driver a nuestras NIC de manera que enlazas las nic virtuales a tortilla. Forma de proporcionar torificación a nuestras máquinas virtuales.

Whonix → podemos montar un router virtual con una serie de máquinas virtuales configuradas por ese gateway para que vayan por la red Tor.

Tor Socks → torificar conexiones.

ADBlock → Addon para navegadores que bloquea anuncios, trackers...

NoScript → Extensión que bloquea tecnologías de scripting, deja activar y desactivar java, flash…

Tiene una lista blanca para permitir la ejecución de guiones informáticos de ciertos sitios.

NoMiner → Hace lo mismo pero bloqueando scripts que intenten usar tu máquina para el minado de criptomonedas.

HTTPS Everywhere → Forzar al navegador a usar HTTPS, no HTTP.

TIPOS DE REDES

Deep Web

Lo que generan las páginas dinámicas se guarda en la deep web, no es accesible al público de forma convencional.

- **Deep Web Onion**: se accede desde Tor u otras pasarelas.

Pseudo Dominio .onion, sólo son accesibles desde una red Tor (ahora hay pasarelas). Subdominio que redirige a máquinas ocultas en la red Tor.

- **Darknet**
- Redes out proxy: Si montas **Tor browser** puedes acceder.
- Redes ip proxy: Se generan “comunidades” dentro de la red, para enlazarte tienen que “invitarte” o certificarte, como Freenet.

Duckduckgo

Buscador centrado en la privacidad y el no traceado.

Torch

Indexador de webs .onion, hay más indexadores.
Se puede acceder hasta a facebook.onion

Herramientas de seguridad:

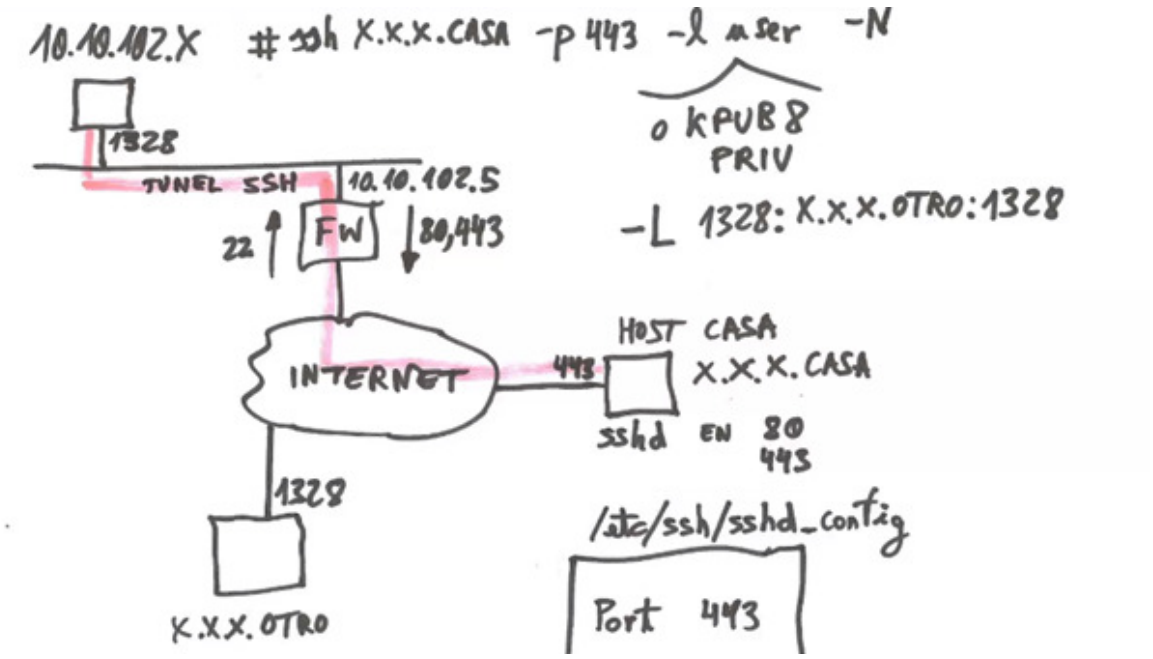
SET - Deja hacer todo tipo de cosas en el ámbito de la ingeniería social.

BeEF - Framework de explotación de navegadores.

HONEY - Máquinas trampa (honeypot, ojo a prácticas).

Honeyd
Es un daemon que permite a un usuario configurar y ejecutar múltiples hosts virtuales en una red informática (crear máquinas virtuales con determinados servicios y configurarlas para que parezcan distintas máquinas con determinado SO o servicio).
Para colocarla en la red a modo de HONEY.

¿Cómo podemos saltarnos un firewall de filtrado de paquetes?



El firewall sólo deja entrar el 22 (ssh) hacia “dentro” y hacia “fuera” el 80 y 443 (HTTP, HTTPS).
En casa podemos montar un sshd que escuche en el puerto 443 en vez del 22 configurando la variable port en el fichero **/etc/ssh/sshd.config.**

Entramos por ssh con **-p 443**

Si queremos acceder a otra máquina distinta de internet, **x.x.x.otro**, y resulta que en el puerto **1328** de esa máquina hay cierto servicio al que quiero acceder desde la red debian 10.10.102.0

Podemos desde la máquina Debian abrir un **túnel ssh** contra la máquina de casa con:

ssh x.x.x.cada -p 443 -l user -N -L 1328:x.x.x.otro:1328

-N → para que no nos asigne un shell.
-L → es redireccionar mi puerto localhost 1328. [a través de la línea roja]
el firewall lo va a dejar pasar ya que es para el puerto 443 de la máquina de casa.
Desde la máquina de casa lo reenvía al puerto 1328 de la máquina **x.x.x.otro**

Para ver el puerto 1328 de la otra máquina desde nuestra máquina 10.10.102.x tendremos que conectarnos al puerto 1328 de localhost.

Con un firewall que realice inspección de tráfico y controle la paquetería nos pueden dropear.

Dyndnss

Servicio DNS dinámico.
Te asigna un nombre de dominio, de tal forma que se mantiene actualizado.
Si tu máquina de casa cambia de IP por DHCP automáticamente se actualiza ese servicio de DNS dinámico con la nueva IP asignada.
De tal forma que si creo un túnel contra el nombre dominio siempre me lo va a crear contra la IP de casa.

Se puede montar via socks, un proxy socks con túneles SSH.
El túnel de antes con **-d 0.0.0.0:8080**, abriremos un túnel desde la máquina de prácticas hasta casa sólo que ahora en lugar de redireccionar un determinado puerto mi máquina local está funcionando como un proxy sock.

Iodine

Tunelizador DNS para saltarse los portales cautivos (son esos portales que encontramos en hoteles. Cuando te enlazas a la wifi del hotel te asigna IP, máscara…Entras a la web y te aparece un portal cautivo, donde tienes que poner un login y password para entrar).

La solución son los DNS, ya que antes de entrar en el portal, haces peticiones DNS.
Los paquetes DNS tienen un pequeño payload de 512 bytes, pues lo que hace es navegar usando estos paquetes DNS, cambiando la configuración DNS apuntando a iodine.
Iodine lo que hará será hacer peticiones DNS y al otro lado empaquetar las páginas en el payload.

> La conexión será lenta pero funciona.

Navegaremos por la web con paquetes DNS gracias a Iodine.
En los paquetes de las páginas vendrá la info de la web en el payload del paquete DNS.

Corkscrew

Tunelizador con SSH a través de un proxy.
Ese proxy solo resuelve peticiones HTTP. Para que esto funcione el servidor proxy tiene que tener el **HTTP CONNECT METHOD**, que forwardeará conexiones TCP.

Port Knocking

Los servicios más delicados son los que escuchan en red: detección, password guessing, DoS…
Con Knocking(Daemon) **/etc/nocking.conf** podemos editar una configuración "secreta" para que sólo puedan usar máquinas clientes nocking un puerto que hayamos tirado, sería una secuencia para los paquetes.

Así desde una máquina cliente podremos enviar un **knockd** a la IP de la máquina y le pasamos esa combinación secreta, eso hará que envíe un SYN a los puertos con ese número.
Por tanto ejecutará un comando asociado (configurable en su fichero de configuración) como por ejemplo que levante SSH y configure una regla en los firewalls (iptables y wrappers) que solo dejen pasar a esta máquina por SSH.
Podemos **securizar** para que X servicios se abran solo cuando sea necesario.

> **knock** port-knock cliente → envía paquetes TCP/UDP a cada puerto especificado creando una secuencia knock especial en el servidor.
> **knockd** port-knock server.

NX Server

Realiza conexiones remotas X11 muy rápidas, lo que permite a los usuarios acceder a escritorios remotos de Linux o Unix incluso bajo conexiones lentas como las realizadas con módem.
Puede ir por SSH o por su propio protocolo.

NX Client Server, por ejemplo: **Server** 10.10.102.xxx **Cliente** mi pc

Web knocking

Podemos tener un servidor web configurado para que no tenga acceso a determinadas IP o a determinadas zonas del árbol de webs para abrirle a solo a quien queremos.

Tema 5: Interceptación

Sniffing: Podemos hacer **análisis de red**.

Un sniffer es un programa que captura las tramas de una red de computadoras.

- **Protocolos no seguros** →Información tal cual [HTTP].
- **Protocolos seguros** → Información cifrada.

Método Basic: métodos de autenticación que no dan seguridad.

Detección y prevención de intrusiones - cogen paquetería de la red y la analiza para buscar posibles incidentes.

Snort
Suricata

Los sniffers también se usan en temas de análisis de seguridad en red (**auditorías de seguridad en red**) e incluso se usa en **análisis forenses post-mortem**, para ver lo que ha hecho una máquina que ya ha muerto. Esto se suele hacer sobre ficheros pcap.

[PCAP es un formato de archivo utilizado por las aplicaciones para monitorear el tráfico de red]

Mirrorear puertos (sniffing sano) -- Un port-mirror nos sirve para poder capturar tráfico que entra y sale de una interfaz sobre un Switch.
La captura se realiza conectando un sniffer (wireshark por ejemplo) en una interfaz X llamada "mirror".

Capa 2: VLANs

Antes realizábamos subnetting en las redes, estas compartían el medio y estaban conectadas mediante routers. En los routers declarábamos las políticas de filtrado y seguridad.

Actualmente tenemos redes de **medios switcheados**, no hay problemas de ancho de banda al tener switches.
De estos medios surgieron las **VLAN**, las máquinas no tienen que competir por el ancho de banda.

Una VLAN es básicamente una extensión. Podemos tener un conmutador de red(switch) dónde cada puerto se puede asignar a una VLAN.

redes lógicas independientes en una misma red física.

Un **conmutador** en la FIC: 3 puertos la **VLAN 1**, en otros la **VLAN 2**.
Otro conmutador en otro edificio separado geográficamente por routers por medio tener una serie de puertos pertenecientes a la **VLAN 1.**

PROTOCOLOS

- **802.1Q**: Nos permite truncar los puertos e interconectar conmutadores.
  Las tramas a nivel ethernet las taggeará para saber a qué VLAN pertenecen.
  - Puerto tranqueado: puerto con 802.1Q, se ocupa de propagar las tramas.

  Por los puertos 802.1Q se van a meter las tramas con el tab de la VLAN a la que pertenecen.
  Se propagarán a distintos conmutadores y estos verán a qué VLAN pertenecen y ya los van a distribuir en esta o en otro conmutador.
  Nos permite distribuir y tener VLANs, a través de puertos truncados.

- **DTP**: [Dynamic Trunking Protocol] podemos hacer lo mismo.

En 802.1Q tenemos una serie de ataques:

Vlan Hopping

Son muy peligrosos.
Permiten a una VLAN tener acceso o sniffar tráfico de otra VLAN.
Más grave que un ARP Spoofing donde robamos paquetería a una máquina. Estamos hablando de VLANs, podríamos saltar a una intranet por capa 2, sin pasar por router.

Existen dos tipos de ataques:

- **switch spoofing**: desde una máquina normal metemos tramas ethernet tageadas con el protocolo 802.1Q.

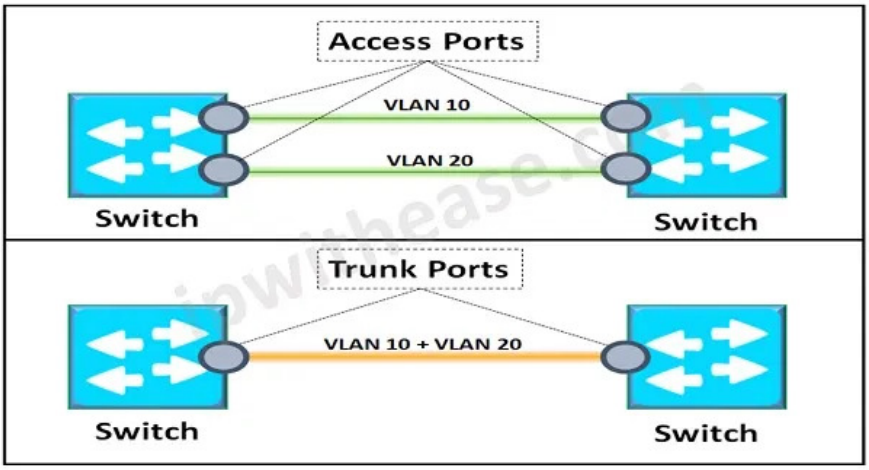
  Si los conmutadores de red de la empresa no están correctamente configurados el resto de los conmutadores pueden llegar a pensar que la máquina es otro conmutador.

Por lo cual los otros conmutadores me van a reenviar paquetería del resto de tramas para poder distribuirlas.
Por tanto podría pegar saltos a otras VLANs y robo de paquetería (o modificar paquetería, etc..).

La **protección** ante esto viene de configurar los conmutadores de red de forma que no gestionen “trunks” de forma automática.
Si alguien intenta tagear desde una máquina que está conectada a un puerto del conmutador tal cual, el conmutador no va a gestionar ahí trunks, impidiendo que ese ordenador se vuelva un conmutador.

Un trunk port puede recibir tráfico de una o más VLANs diferenciadas.
Mientras que un Access port sólo forma parte de una VLAN y normalmente se usa para la comunicación con dispositivos como PCs, portátiles e impresoras.

Un **puerto de acceso** transporta tráfico hacia y desde **sólo** la VLAN especificada que se le ha asignado. A diferencia de un **trunk port**, no entregará tags de identificación exclusivas (etiquetas 802.1Q o ISL) porque la VLAN destinada a él está preasignada



- **double tagging:** desde nuestra máquina las tramas ethernet que metemos en la red se le mete un doble taggeado. [taggeo los frames con dos tags de dos vlans]

  Irán taggeadas a nivel conmutador por la red a la que pertenecen pero a mayores se les inyectará un tag de VLAN. De tal forma que el conmutador va a coger la trama, va a mirar el taggeado ethernet enviándolo a los conmutadores que sea y lo eliminará.
  Los conmutadores de destino van a coger este segundo taggeado que le hemos metido y lo van a procesar. Por tanto, podemos pegar un salto de VLAN.

Este ataque **sólo funciona** si el puerto de nuestras máquinas atacantes están asociados a la VLAN nativa del conmutador.

La **protección** se basa en asegurar que los puertos de las máquinas no estén asociadas a la VLAN nativa, lo normal es que estas sólo tengan los puertos truncados por los que se distribuyen las tramas entre conmutadores.

Estos ataques son muy serios porque **se saltan los firewalls de nivel 3 para arriba**, que filtran IP destino, etc.

yersinia → Ataca a un buen número de protocolos como el STP, el DTP, el VTP, DHCP, 802.1Q…

Es la herramienta por excelencia para el ataque en capa 2.

STP → Emite tramas BPDU solo que se cambia la MAC por una más pequeña.
Lo que consigues es hablar STP y convertirte en el nodo raíz [te llegará todo el tráfico, si lo dropeas harías un ataque de **denegación de servicio**].

STP [Spanning Tree Protocol] - Protocolo para analizar bucles redundantes.

Muchas organizaciones no tienen securizados los STP.
Suele venir configurado para que se difunda a toda la red de la organización.

Su objetivo es **evitar bucles**.

Normalmente a las troncales que se enlazan las redes de fibras ópticas tienen caminos redundantes, por si hay una caída de una línea.
El problema de tener muchos caminos redundantes es que un paquete acabe en un bucle.

El protocolo STP lo “hablan” los conmutadores, estos generan un árbol a partir de la estructura previa de forma que ya no hay bucles.

Usan tramas BPDU (Bridge Protocol Data Units), por defecto las redes suelen aceptar y difunden las BPDU a toda la red, por tanto hay organizaciones que desde una máquina cualquiera puedes recibir tramas BPDU.
Con yersinia podrías conectarte y responder, comunicándote por STP con los conmutadores de una determinada red.

Esto es un **problema de seguridad muy grande**, esa máquina está hablando con el backbone de la organización.

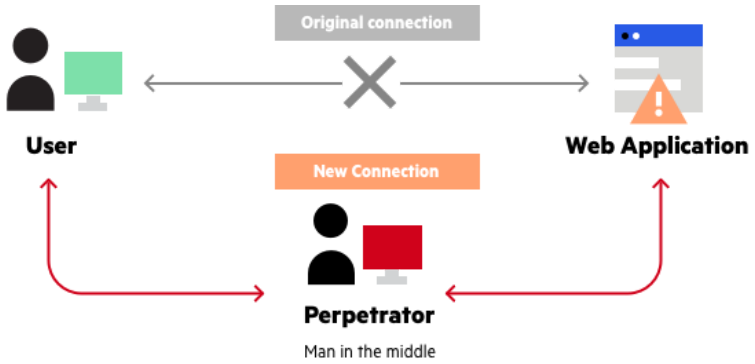
Algunos ataques [de los 6] son:

- DoS → Con yersinia intentamos convertirnos en el **root bridge** (cima del árbol STP) del **backbone** (una de las principales conexiones de internet) de toda la red.
  Aprovechando el campo de **identificador de prioridad** de las tramas BPDU, el que tiene la más baja es el nombrado root bridge.
  Yersinia manda un BPDU cada dos segundos con la misma prioridad que el root bridge con una mac ligeramente menor. Si el STP ve que las prioridades están empatadas le asigna el rol de root bridge a aquella que tenga la **menor MAC**.

Así el resto de conmutadores mandarán el tráfico por mí. Si queremos hacer un DoS solo tendremos que dropear el tráfico.
Aquí no haríamos DOS a las máquinas de la 10.10.102.0, si no a gran parte de la red de una organización.

Hay una variación usando otro tipo de BPDU la cuál manda solicitudes de cambio de topología.
En vez de conseguir el tráfico lo que hacemos es conseguir el root bridge, quitárnoslo y ponérnoslo.
Esto generará una cantidad de tramas que le harán una sobrecarga a los activos de red y CPUs que puede acabar en una **denegación de servicio**.

- MITM → [Man In The Middle] una persona desde su máquina conectada a un puerto de un conmutador se convierte en un nodo de las troncales de la organización de tal forma que por él pase todo el tráfico.
  Es más complicado, hay que estar conectado a dos conmutadores.



Para defender **STP** hay varios mecanismos:

Root Bridge War → configura los puertos de los conmutadores por los que no pueden llegar paquetes de un root bridge.

De manera que si por ahí llegan tramas BPDU con una prioridad baja, o del root bridge con una MAC más pequeña ponemos esa interfaz en **modo inconsistente**, de forma que escucha las tramas pero no las forwardea.

BPDU War → no nos permite hablar BPDU a las máquinas normales (nuestro PC).

Si configuro BPDU War en los puertos de los conmutadores de red y alguien intenta inyectar BPDUs en esos puertos no los va a aceptar.

BPDU Filtering → previene que ciertos puertos específicos envíen o reciban BPDUs.

De forma que el backbone no envíe tramas BPDU a las máquinas que no lo requieren. Por ejemplo, nuestros portátiles no deben recibir estas tramas.

Si stp está mal configurado y puedo enviar y recibir stp entonces puedo emitir tramas bpdu con una mac más baja que el **root bridge** actual y si cogen mis tramas puedo convertirme en **root bridge**, si yo me convierto en root bridge de una empresa todo el tráfico de esa empresa pasa por mi maquinita.

Capa 2

Si ataco en capa 2, me salto los firewalls.

arp spoofing

ip forwarding (a 1) si lo tienes activo y te llega pero no coincide tu ip con la ip destino lo reencamina. (mirar lo de DOS aquí)

- salir con q.

Mi máquina no debería hacer ip forwarding, denota que estamos haciendo algo “malo”.

si arp no funciona puedo usar:

- Mac Flooding → a nivel de conmutador, sólo funciona en algunos conmutadores.

  Si a algunos conmutadores les lleno la memoria (inundando vía MACs falsas llenamos sus tablas CAM) pasan a funcionar en **modo HUB** (medio compartido, todos los paquetes a todos los puertos, interfaz de red en modo promiscuo) tenemos todo el tráfico.

  Como si fuese un repetidor.

- ICMP redirects → si las máquinas aceptan, estos paquetes consiguen que las máquinas te configuren como default router, así el tráfico hacia fuera pasa por ti.

  Tendríamos un **half duplex mitm**.

Las máquinas no deben aceptar ICMP Redirect o ICMP Router Advertisement.

En *ettercap*:

ettercap -m icmp: se le pasa como parámetros la mac y la ip del router legítimo.

/etc/systemctl.conf reconfigura var.

Para la defensa de icmp redirects: **/pro/sys/dev/IPv4** dónde podemos ajustar si aceptan redirects.

También en **/etc/sysctl.conf**: se puede crear desde aquí, creando esas 3 variables directamente:

- Ignore ICMP broadcasts

net/ipv4/icmp_echo_ignore_broadcasts = 1

- Do not accept ICMP redirects (prevent MITM attacks)

net/ipv4/conf/all/accept_redirects = 0

- Accept ICMP redirects only for gateways listed in our default gateway list (enabled by default)

net/ipv4/conf/all/secure_redirects = 1

- Do not send ICMP redirects (we are not a router)

net/ipv4/conf/all/send_redirects = 0

soft_route : otra variable, si la tenemos activa la máquina que envía puede especificar qué camino deben seguir estos paquetes (normalmente quien los redirigía era el router).

DHCP spoofing: Ataque de ettercap.

Interceptamos la conexión de un cliente DHCP y como servidor nos ponemos como su default router.

Se le pasa un rango de IPs que será el rango de máquinas que interceptemos.

Half MITM, ya que la máquina nos mandará el tráfico pero, a la vuelta el router se lo enviará directamente a la máquina atacada.

DHCP Snooping: Defensa. Si configuramos un conmutador como DHCP Snooping True, para bloquear los paquetes DHCP en los puertos que no sean realmente DHCP.

Port stealing : si el ARP Spoofing no funciona es muy habitual hacerlo.

Los conmutadores (switch) tienen una tabla de memoria que asigna puertos-MACs, de forma dinámica (tienen una memoria asociada a los puertos con sus MACs conectadas).

Cuando un conmutador quiere conectarse a otra MAC tiene que ver a qué puerto está asociada.

Si estamos desde un puerto, podemos floodear otro puerto con MAC origen la de otra máquina y el conmutador nos lo asignará a nosotros. Con esto estamos engañando al conmutador, no a la máquina.

/etc/etter.conf → port steal delay, tasa de floodeo para robar un port.

No es MITM, porque como robas un puerto puedes robar tráfico de otras máquinas.

tree → Mostrar los directorios en forma de árbol.

NDP (protocolo) robo información en IPv6; en ipv6 no hay ni arp ni broadcast, hay icmp y direcciones multicast.

mando paquete icmp diciendo la ip de la que quiero la mac.

- M ndp → localizamos dir. MAC

El ataque intercepta esos paquetes NDP/ICMP, dirección multicast y le envenena la caché. De esta forma pasará por nuestra máquina.
Se puede usar **ettercap.**

DNS Spoofing

/etc/ettercap/etter.dns

Podemos ejecutar un ettercap con un ARP Spoofing clásico y con el plugin **-P dns_spoof**, así que en ese MITM las resoluciones DNS que nos pasen las cambiarán por las del fichero etter.dns.

SECURIZACIÓN

DOT: DNS over TLS. Protocolo de seguridad para cifrar las comunicaciones entre los clientes y los servidores usando TLS.

TLS: Extensión de SSL. HTTPs está cifrado usando mayoritariamente TLS.
Establece una clave de sesión (con un sistema simétrico) y se cifra toda la paquetería.

DNSSEC: Extensiones de seguridad para el sistema de nombres de dominio. Tratan de aumentar la seguridad

Todas las respuestas en DNSSEC son firmadas digitalmente.

Para usar DNSsec, DOT tiene que soportarlo en el sistema, con distros medianamente nuevas. La configuración normalmente es es **/etc/systemd/resolv.d**

Evilgrade

Framework que permite al usuario aprovecharse de las implementaciones de actualización deficientes inyectando fake updates.

Lo que hace es inyectar en los procesos de actualización de las máquinas, si estás en el medio [mitm, arp spoofing, TCP Hijacking] poder inyectar nuestros paquetes infectados.

- **TCP Hijacking**: robo de sesiones TCP, se intenta identificar los números de secuencia para meterte en medio y robar una sesión.

PROTECCIÓN

arp -a → Nos sale la tabla caché de MACs, todas las máquinas con las que tuvimos relación, ARP…

arp -del [MAC] → Borrar una entrada de la tabla.

Las entradas pueden ser dinámicas o estáticas.

arp flush all → borrar toda la tabla arp.

IPS Snort : Snort puede vigilar el tema del ARP Spoofing.

NAST: Herramienta sencilla

nast -m : obtiene relación IP-MACs del segmento.
nast -c : si ve una IP con una MAC distinta a la que tenía asociada, nos avisa.

ETTERCAP

ettercap -P list → lista de plugins

rand_flood - floodeo de paquetería a lo bestia.
Lo que hará es que en el **etter.conf** es cambiar el delay de floodeo de paquetería.

Muchos conmutadores tiene protección, como unicast flooding protection, si ve mucho floodeo lo corta.
O port security, que limita el número de MACs por puerto.

SNIFFING

Se puede tener un firewall de nueva generación o algo más "casero", como tener Port Spam/Port Mirroring que copiará el tráfico de esos puertos a otros.

Podríamos conectar este puerto a un PC con un IPS como **snort**, de forma que va a estar procesado en paralelo estos paquetes.

Se puede provocar un cuello de botella, para esto se usa un **port trunk** (bonding de red), una agrupación de puertos de red para incrementar los anchos de banda para atender mejor la carga, además obtenemos más tolerancia a fallos.

Si muere una tarjeta funciona todo igual, quedan más.

En **802.1Q** los trunk son los puertos que envían los tags etiquetados, es distinto.

Un bonding de red garantiza que un sistema esté siempre accesible a través de una red informática.

Tipos de Bonding:

- **Bond Mode 0** (balance-rr): se le llama 'Round-Robin'.
  Con este método, los paquetes de red enviados son rotados entre cada una de las tarjetas que forman la interfaz del bond.

- **Bond Mode 1** (active-backup): una tarjeta funciona y otra está ociosa, si una se estropea pasa a funcionar la otra. Nos da tolerancia a fallos.
- **Bond Mode 2** (balance-xor): decide porque interfaz de red transmite haciendo una operación XOR entre MAC Origen, MAC Destino y módulo con el número de interfaces de red.
  Nos da tolerancia a fallos y aumenta el ancho de banda.
- **Bond Mode 3** (broadcast): todos los paquete se transmiten por todas las tarjetas.
- **Bond Mode 4** (802.3ad): es un estándar, agregación de enlace, o **"port trunking"**.
  Nos da más velocidad y alta disponibilidad.
- **Bond Mode 5** (balance-tlb): sólo hace balanceo en base a la carga de las interfaces de red.
- **Bond Mode 6** (balance-alb): balanceo tanto en envío como en recepción.

TAP (tests access ports)

Dispositivos de sniffing no detectables, muchas veces se encuentran entre el router exterior a internet y el firewall de acceso a tú empresa.

En otras ocasiones se ubica entre la red de mi empresa y el firewall.

Suelen tener interconexión con los firewalls para tomar decisiones.

sidejacking [secuestro de sesión o secuestro de cookies] → uso de credenciales de identificación no autorizadas para secuestrar una sesión web válida de forma remota con el fin de hacerse cargo de un servidor web específico.

Firewalls de nueva generación trabajan en todas las capas, vigilan mucho la integridad de los **hipervisores** ya que si lo tiran caería todo el sistema que gestiona todas las máquinas virtuales.

SIEM [Big Data]

Gestiona toda la información.

- **Splunk**: software para buscar, monitorizar y analizar macrodatos generados por máquinas de aplicaciones, sistemas e infraestructura IT a través de una interfaz web.
- Prelude: capaz de recuperar cualquier tipo de log.
- Logrhythm
- SIEM en CESGA: Volúmenes de información enormes. Utilizan el Framework apache hadoop [código abierto que permite programación distribuida].
  Tú programas, él se ocupa de la distribución.

Usa su propio sistema de ficheros.

Feeds de reputación: índice de reputación de determinadas sides o IPs.

Lo podemos aplicar en nuestros proxys, nuestros firewalls…

https://zeltser.com/malicious-ip-blocklists/ → lista de redes que hayan sido expuestas a ataques.

http://www.sorbs.net/ → mayoría de las estafetas de mail de las máquinas.

https://www.virustotal.com/gui/ → testea un fichero ante antivirus y antimalwares.

MITM en HTTPS

Si hacemos un **MITM** tendremos tráfico cifrado, no podremos leer nada.
Ettercap tiene modos para acceder a los datos, usa el más antiguo, **inyección de certificado**.

Cuando el servidor envía el certificado digital lo cogemos y somos nosotros los que abrimos una conexión HTTPs con el servidor.
Entre nuestra máquina y el cliente (víctima) inyectamos un certificado falso, tendremos 2 conexiones HTTPs.

Nuestro certificado seguramente no esté autorizado por una autoridad certificadora, por tanto el navegador de la víctima puede avisar de que hay un **falso certificado**.

Si cambiamos el certificado por uno que parezca “bonito” con el nombre de la empresa, el usuario puede ser engañado.

- SSL Strip → tipo de ciberataque que trata de hacerse con los datos de un usuario cuando accede a una dirección web protegida mediante un certificado SSL/TLS, es decir, cuando estamos utilizando el protocolo de la capa de aplicación **HTTPS.**

  Mete HTTP en claro pero de forma que el cliente vea que se está usando HTTPs, y no puede haber problema de certificado, *ettercap* tiene un módulo para eso.

  Protocolo HSTS: Obliga a que todas las comunicaciones siempre funcionen sobre HTTPS

  cuando nos conectamos al servidor por primera vez nos manda un token para el repositorio de HSTS, y nos valdrá para protegernos de herramientas de suplantación como SSL Strip.

- SSL Strip 2

Sólo los routers deben ejecutar protocolos de routing, si ves una máquina haciéndolo mala cosa.

HERRAMIENTAS

- SNIFFING → DSniff, SSHMITM…
    - **tcpkill** : para matar conexiones.
    - **tcpnice** : ralentiza conexiones TCP en una LAN.
- GREP → ngrep <interface> -x cadena: busca una cadena en una interfaz.

Hijacking → robo de sesión por cookies.

secuestrar la cookie para robar user/passwords. Se puede obtener de muchas maneras.

Cookies convencionales: se han usado para procesos de autenticación.
El servidor te manda una cookie y tú la almacenas en el navegador para que en futuras visitas no tener que volver a conectarte con usuario/password.

No dejan de ser un String que se almacena en nuestro navegador que nos identifican y nos tracean.
Visualizar y modificar nuestro repositorio de cookies:

- cookie editor
- cookie cooler

HTML5 → nacen alternativas al repositorio de cookies: **webstorage**, permite almacenar en memoria del navegador elementos del tipo clave-valor.

Los principales tipos de almacenamiento aquí son el **sessionstorage** y **localstorage** donde los datos son persistentes, se almacenará información y esta seguirá en las siguientes visitas, funciona como una cookie.

Una ventaja es el tamaño, que en localstorage es mucho mayor, y además no tienen fecha de expiración como las cookies.

HTML 5 incluye un API de JavaScript para interactuar con el repositorio de webstorage.

Supercookies o **cookies persistentes**, usan otro tipo de repositorios. (flash o silverlight)

Hay tecnologías para seguir traceando aunque se borren las cookies.

Cookies 4K tamaño máximo

Framework evercookie

Incluye más de una decena de tecnologías para poder soportar cookies persistentes.

- **cookies locales**
- **cookies de terceros**

cooking con HSTS

Se emplean **flags** de la especificación HSTS (HTTP Strict Transport Security) para guardar un identificador que le permita al sitio reconocer al usuario desde cualquier web, incluso si navega desde un navegador en modo anónimo o privado.

Tema 6: D[D]oS

1ª clasificación a **nivel lógico** o **flooding**

Ataques lógicos: una determinada vulnerabilidad que explotándola provoca una denegación de servicio.

Solución → parcheo.

Ataques por inundación: ataques con mucho tráfico para parar o tirar el servicio.

FA (Factor Amplificación): define cuántos paquetes van a "floodear" a la máquina objetivo.

> Pej: Un FA=1 sería un ataque directo. Yo mando un SYN a la máquina → 1:1.
> Si envio un paquete a broadcast de una red de 200 máquinas y cada una responde a la máquina que queremos atacar, ahí tendremos un FA=200

- **Floodeo directo**: ip origen → ip destino
  Inyectan directamente, no tiene por qué ser desde tú máquina.
  Puedes atacar mediante **ip spoofing**.
- **Inundación Reflectivos**: inyectan paquetería. La ip origen es la máquina que queremos floodear, el destino son las demás máquinas.

  Las máquinas destino van a responder a la máquina víctima, y los FA van a incrementarse secuencialmente.
  Muy importante el **ip spoofing** ya que es un problema muy grave facilitando los ataques DoS.

  Si empleamos un **firewall de control de estado** podría ayudarnos ya que dropea todo [la víctima no ha iniciado el SYN de las conexiones]

¿Y con tráfico UDP? UDP no tiene control de estado, pero también lo implementan (muy diferente a TCP), algunos firewalls si implementan cierto control de estado.

Firewall en **modo cluster** → Trabajan 2 en modo paralelo.

Ataques controlados de manera remota

- **BotNets** → suelen ser redes de máquinas infectadas con algún agente.

  Hay muchísimas: Mariposa en el 2010, **Morait** en 2016, la cual infectó IoT.

```
$:(){:|:&};:
```

Este conjunto de carácteres tirará nuestra máquina.
Los "**:**" equivalen a una función, los "**()**" son los parámetros de esta y los "**{}**" son el contenido.
En la función llamamos a la misma (**:**) y le pasamos la salida con **|** a la función y al "**&**" lo ejecuta como background.

En una función que se llama recursivamente (**fork bomb**) al cabo de muy poco se tienen miles de procesos.

Muchas veces en las máquinas metemos cuotas de disco, pero no de procesos, por lo cual cualquier usuario podría tirar la máquina con un simple comando.
Si hubiese cuota de procesador se crearían procesos que consuman mucho más recursos para tirarla.
Estas cuotas básicamente se encargan de definir la cantidad de espacio para un archivo en un usuario específico. Esto evita que un usuario en particular pueda abusar del espacio global del disco en cosas que no son vitales.

Ataques clásicos

- **Smurf attack**: ICMP Echo Request con IP Origen la máquina que quiero floodear y destino broadcast.

En los ficheros de configuración **/proc/sys/net/ipv4** hay un fichero **icmp_eco_ignore_broadcast** que por defecto está a 1, por tanto, si a las máquinas les llega un ping a la dirección de broadcast lo dropean y no responden, evitando un floodeo smurf.

icmp_eco_ignore_all : por defecto a 0. Si lo ponemos a 1 nuestra máquina ya no responderá al ping.

reverse_path_filtering (rp_filter) : por defecto viene a 0. Método que ayuda a prevenir ataques por IP Spoofing.

1 → Strict Mode (definido en RFC 3064) : cada paquete que llega al router es testeado contra la tabla de enrutamiento, si la interfaz en la que el paquete es recibido no es el mejor camino de vuelta, este es dropeado.
2 → Loose Mode (definido en RFC 3064) : routing asimétrico (llega un paquete por una iface de red y sale por otra).
Cada paquete que llega al router es testeado contra la tabla de enrutamiento, si la dirección de origen no es enrutable por ninguna interfaz dropeo el paquete.

Derivaciones del smurf:

- **Fraggle attack**: igual pero a nivel UDP, por tanto si hay un firewall de control de estado tiene muchas más posibilidades de saltarlo.

DEFENSA

Lo mínimo es que el router bloquee paquetes broadcast de fuera de la red.
Podrías atacar a un sitio spoofeando su IP como origen contra direcciones de broadcast de redes ajenas a nosotros, si nuestro router permite rootearlo liaremos una enorme.

HERRAMIENTAS

Hping3 -> Generador y analizador de paquetes de código abierto para TCP/IP.

Packit -> Herramienta de análisis e inyección de paquetes.

Scapy -> Herramienta de manipulación de paquetes.

SYN flood

Con SYN llegamos a las máquinas DMZ, donde las organizaciones tienen las máquinas con comunicación con internet.

Nuestras máquinas tienen un **TCB**: Transmission Control Block. Pila dónde se almacena la info de las conexiones abiertas de nuestra máquina.

Acceso secuencial.
Con un TCB muy grande puede afectar al rendimiento.

En /proc/sys/net/ipv4/tcp… que por defecto viene a 128, determina el tamaño máximo de esa estructura de memoria → Podemos mantener hasta 128 entradas.

/proc/sys/net/ipv4/tcp_synack_retries = 5s → con un timeout muy pequeño puede hacer una denegación de servicio.

SYN flood llena las entradas antes de que salten los timeouts (SYN - SYN/ACK y aquí empieza el timeout, si se acaba antes del ACK se cierra la conexión).
Si se llena la pila está muerta a nivel de red, no acepta nuevas conexiones.

DEFENSA

SYN COOKIES

/proc/sys/net/ipv4/tcp/tcp/syn_cookies = 1 activamos las syn_cookies

Así, una vez se llene la estructura TCB en un SYN Flood, genera un **número de secuencia** en el que va codificado/hasheado la IP Origen, IP Destino y el puerto, de tal forma que cuando me responda con el ACK a ese número de secuencia puede reconstruir ese IP Origen/destino/puerto.

Por tanto la información mínima y necesaria para hacer el handshake ya no la sacamos del TCB, sino del número de secuencia de los paquetes.

SYN PROXIES

Un firewall puede actuar como proxy, captando los SYNs y realizando el handshake. Proxy a nivel de SYN, para esto se suele hacer **Ip Spoofing** a la IP a la que se va a conectar, luego el SYN Proxy le pasa la conexión a la IP Real.

SYN CACHÉ

Cada entrada en el TCB mantiene:

- IP Local, IP y puerto remoto
- Iface de red para la conexión
- Estado y tamaño de ventana (local y remota)
- Nº de secuencia de paquetes que han recibido su ACK y los que no
- ...

Se usa una tabla hash (acceso mucho más rápido), en está caché se almacena la información mínima e indispensable para gestionar el **handshake** (puerto ip origen y destino, pej.).

UDP Flood

Muchas veces los firewalls de las organizaciones no gestionan control de estado UDP, por tanto se podría hacer un reflectivo contra determinados puertos.

udpflood → Herramienta simple por línea de comandos que puede crear y envíar paquetes UDP.

DEFENSA

Filtrado.

TTL

Cada vez que pasa por un router se va decrementando, cuando llega a 0 se reenvía a la máquina origen a través de un paquete ICMP.

Si inyectamos paquetes con TTL pequeño expirará de forma que le mande muchos paquetes ICMP a la máquina origen.

Aislar máquinas → plugin **isolate** de Ettercap, isolar una máquina es aislarla.
Si se lo hacemos a la 10.10.102.5 nadie podrá comunicarse con el router.

Router advertisement

SLAAC: [configuración automática de dirección sin estado] Cuando la máquina se enciende se le configura una interfaz Link Local.

Los dispositivos dependen de los mensajes de anuncio de router (RA) de **ICMPv6** del router local para obtener la información necesaria.

Con **RA** podemos configurar las direcciones de enlace local de una red, con FakeRouter6 podemos hacer esto.
Si miramos con ifconfig -a veremos que nuestra interfaz de red por la que llega esa paquetería está almacenando muchas link local.
De esta forma la CPU de las máquinas **llegan al 100%** hasta que se caen.

D[D]oS

Muchas veces cuando llega mucho tráfico de distintas IP las dropeamos, pero a veces no sabremos identificar cual es legítimo o no legítimo.
Muchas veces jugamos a nivel ancho de banda, le bajamos el AB a la máquina para que esa máquina pueda comunicarse contigo pero sin pasarse.

IPTables

```
# iptables -A INPUT -p tcp --dport 80 -m hashlimit
--hashlimit-upto 50/min --hashlimit-burst 400
--hashlimit-mode srcip --hashlimit-name http -j ACCEPT
```

```
# iptables -A INPUT -p tcp --dport 22 -m hashlimit
--hashlimit 1/min --hashlimit-mode srcip
--hashlimit-name ssh -m conntrack --cstate NEW -j
ACCEPT
```

Esto va asignado al **QoS** (Calidad de servicio) ya que va vinculado al AB y a la infraestructura.

Normalmente tenemos **balanceadores de carga** (en ataque DoS), esto transforma el ataque DoS en una "lucha" paraa ver si el DoS lo da tirado o la infraestructura lo aguanta.

Tunning de máquinas: quitamos servicios y ficheros que no hacen falta. Ajustamos parámetros…

Esta máquina responderá mucho mejor a ataques DoS y demás.

El disco es la parte más lenta, hay que cuidar los **RAID**.
Los sistemas de ficheros también, cuando nos dan a elegir los sistemas de ficheros en linux hay que elegir los más recientes, aunque depende del uso que le vayas a dar.

Por ejemplo, al montar una estafeta de correo electrónico, se puede elegir entre **Mail box** o **Maildir.**

Mail Box → por cada usuario tendremos un fichero con todo su correo electrónico.
Maildir → para cada usuario un fichero por cada correo electrónico que tenga.

Las implicaciones son **muy distintas**, una con pocos ficheros pero grandes, y la otra al revés, el acceso a disco es totalmente distinto.

STRESS TOOLS

Jmeter: propia de apache, para pruebas.

Soapui: open source, para pruebas de carga en servicios web basados en soap y rest…

ApacheBench: Testear servidores web

APACHE

Directorios apache:

- **mods_enable**: los que están cargados, normalmente enlaces a mods_available.
- **mods_available**: conjunto de módulos instalados o preinstalados.
- **sides_available**: Los sides son páginas web, aquí están el conjunto al que podemos proporcionar servicios.
- **sides_enable**: A los que le estamos prestando servicio.

Módulos

```
# nano /etc/apache2/mods-enabled/evasive.conf
<IfModule mod_evasive20.c>
DOSHashTableSize 3097
# máximo peticiones de una misma página por tiempo definido
 DOSPageCount 2
 DOSPageInterval 1
# máximo peticiones al site por tiempo definido
 DOSSiteCount 100
 DOSSiteInterval 1
# Periodo de tiempo de baneo en segundos
 DOSBlockingPeriod 500
```

Vienc por defecto

mod_evasive

```
# Periodo de tiempo de baneo en segundos
 DOSBlockingPeriod 500
# DOSWhitelist 127.0.0.1
DOSEmailNotify lsi@localhost
DOSLogDir "/var/log/apache2/mod_evasive"
# tb. el comando podría ser un iptables para dropear
# /sbin/iptables -I INPUT -p TCP --dport 80 -s %s -j DROP
DOSSystemCommand "/bin/echo %s >>
/var/log/apache2/mod_evasive/dos_evasive.log && /bin/date >>
/var/log/apache2/mod_evasive/dos_evasive.log"
```

Comandos

Verificamos que **todo esté OK** a nivel sintáctico.

sudo apache2ctl -t
Syntax OK
systemctl restart apache2

Podemos probar con un script en Perl de mod-evasive.:

```
# perl /usr/share/doc/libapache2-mod-evasive/examples/test.pl
```

O con el ApacheBench.:

```
# ab -n100 -c5 http://127.0.0.1/index.html:80
```

Mod security → web application firewall (**WAF**)

Tiene un conjunto de reglas, se pone delante del servidor para protegernos de distintos ataques.

P.ej: de fingerprinting devuelve un 403 a un **whatweb**, un **nmap** al puerto 80, etc.

Mod_antiloris → Específico contra el slowloris y sus variantes evitando demasiadas conexiones desde una dirección IP.

El ataque consiste en enviar cabeceras sin acabar llenando el servidor.

Mod_reqtimeout → Proporciona una forma conveniente de establecer tiempos de espera y velocidades de datos mínimas para recibir requests.

PORT ISOLATION

La DMZ es nuestro primer frontend, es la parte más vulnerable.
Si la configuramos de manera que haya servidores que no estén conectados entre ellos [configurando el conmutador] (**no estén conectados en capa 2**), si obtienen permisos de una no pueden ir a las otras.

FLOODS

Direct Flooding attack

```
# packit -c 0 -b 0 -sR -d 10.10.102.100 -F S [-S 1000
-D 80]
```

Relfective Flooding attack

```
# packit -c 0 -b 0 -s 10.10.102.100 -d R -F S [-S 1000
-D 80]
```

```
# hping3 -S -p 80 --flood --rand-source 10.10.102.50
```

LAN ATTACK

Actualmente los kernels los detectan, pero los antiguos pueden ser vulnerables ante esto.

DDoS a nivel mundial

Lo más habitual es **atacar los servidores web**. Normalmente montamos balanceadores de carga, aceleradores…

Varnish: Es un acelerador de caché inversa, para que no se sobrecarguen los servidores.

SNDP (IPv6): Es el NDP pero securizado, usando clave pública y privada. Securiza mucho la obtención de las MACs, dificultando más los MITM.

thc-ipv6: FakeRouter6 tira *Debian*, *Ubuntu* y *Windows* si no han sido mínimamente tuneados.

Con unas variables se puede securizar.

VARS

/proc/sys/net/ipv6

autoconf

accept_ra

A 0 desactivamos el SLAAC, la pila no va a configurar la Link Local y no aceptaremos los paquetes RA.

[*Nino*] - Un alumno de LSI estaba administrando una serie de sides de un servidor apache con cierto patrón de periodicidad. Al parecer desde varios sitios apuntaban a su servidor mediante iframes.
En las hojas de estilo podemos tener ciertos códigos HTML con cierto iframe con cierto servidor web.

Framekiller → Actualmente muchos navegadores bloquean los **iframes**.
marcos dentro del marco de la página que permiten incrustar docs,videos..

DENEGACIÓN EN CAPA 7

Slowhttptest: Herramienta que **simula** algunos ataques de denegación de servicio en la capa de aplicación.

slowloris → herramienta que permite que una sola máquina elimine el servidor web de otra máquina con un ancho de banda mínimo.
Abre un montón de conexiones con un servidor web y le pasa lentamente los campos de la cabecera, sin llegar a terminar el envío.
Las cabeceras se finalizan con dos líneas en blanco, pero aquí no se ponen.
Con esto el servidor se viene abajo.
[Ataque Slow Headers]

slow http post → Abre muchas conexiones y envía cabeceras, luego empieza a enviar los datos muy despacito, de manera que no salten los timeouts para producir el máximo daño posible al servidor.
"Content-Length" menor a la cantidad que se envía (mensajes HTTP sin acabar) Queda abierta conexión esperando.

[Ataque Slow Post]

slow http read → Peticiones legítimas pero se ralentiza el proceso de lectura de las respuestas.
[Ataque Slow Read]

Apache Range Header → causa un uso muy significativo de memoria y CPU en el servidor.

PROTECCIÓN

Variables de Apache 2. El más típico es el **timeout**, por defecto viene 300 segundos, hay que bajarlo a 40-50.

- **mod_antiloris**: para parar los ataques slowloris.
- **ossec**: detecta algunos de estos ataques con su conjunto de reglas.

A nivel de firewall, jugando con el ancho de banda:

- **mod_evasive**: pone límites por IP a las páginas y al side concreto.

Doona → Herramienta de fuzzing de red.
Es una bifurcación de Bruteforce Exploit Detector Tool (BED).

BED es un programa diseñado para comprobar daemons en busca de posibles buffer overflows, errores de formato de cadena, etc.

Fuzzing: probar el comportamiento de una aplicación frente a unos datos generados específicamente para hacer que un programa falle.

(ya sea enviando cadenas largas o probando a desbordar valores numéricos)

Thc TLS DoS → Herramienta de DoS en SSL o TLS.

Estos “procesos” tienen un coste para el servidor mucho mayor (handshaking, cifrado...).
Con muchas menos conexiones en un DoS se puede hacer mucho más daño al servidor.

Torshammer → Ataque en capa 7 tipo post **TORIFICADO**, estamos por detrás de la red tor de manera que si alguien nos intenta tracear va a tener 3 nodos de la red tor de por medio.

DoS en WiFi

802.11W: cifra y autentica tramas, evitando X tipos de DoS en WiFi.

Sistema de prevención de intrusiones específico para Wifis.

- Uno de cisco
- OpenWIPS-ng
- Acrylic Wi-Fi → análisis de cobertura y seguridad WiFi, mapear disp al alcance..

DIFERENCIAS	2.4GHZ	5GHZ
CANALES	14 canales no superpuestos	25 canales no superpuestos
INTERFERENCIAS	Más interferencias	Menos interferencias
VELOCIDAD MÁXIMA	Menos velocidad de conexión	Más velocidad de conexión
RANGO DE RED	Mayor rango	Menor rango
ESTÁNDAR	IEEE 802.11b, 802.11g, 802.11n (B, G y N)	IEEE 802.11a, 802.11n, 802.11ac (A, N, AC)

Bandas wifi

- 2.4GHz

  Tenemos 13 canales, se suelen usar 12 y muchos de ellos solapados. Si pones muchos puntos de acceso en uno o en los solapados puedes tener problemas.

  Están los puntos de acceso dinámicos, que al encender decidirán el canal ideal donde habría menos colisión o conflicto.

- 5GHz
  No hay solapamiento de ningún canal.

  Lleva peor el paso de obstáculos físicos, la cobertura en distancia es menor en general.

Owisam

Define un conjunto de controles y que hay que mirar para evaluar la seguridad en redes wifi, comprueba de todo.

WiFiJammer

Pones un WifiJammer en un edificio de Coruña y vas a deautenticar todos los clientes wifis y AP. Dejando a los usuarios sin conectividad.

DEFENSA D[D]oS

Se pueden usar whitelist, balanceadores, clusters…

Cloud traffic scrubbing center

Puedes redirigir el tráfico cuando recibes un ataque para que esa organización se encargue del filtrado. Mandando el tráfico legítimo.

Hardening

Si nuestros sistemas están por defecto las máquinas van a responder peor contra los ataques.

Para bloquear los ataques podemos acelerar comiendo todas las conexiones rápidamente sin que haya mucha implicación en el tráfico legítimo.
O bien realentizar, cediendo el ancho de banda y controlando la situación.

FireQOS New User → Paquete que te ayuda a configurar el traffic shaping en Linux.

Bases de datos y "listas negras"

Chequea en una IP o un conjunto de IPs contra unas 50 BD de blacklist, que son de todo ámbito, si sale ahí es que han estado en algo poco seguro.

SpamHaus

SBL: **Spamhaus Blacklist**, conjunto de unas IPs de las que no se recomienda aceptar email, se puede integrar en las estafetas de correo electrónico.

Hay que tener cuidado, pues por algún error en nuestras configuraciones nuestra organización podría acabar por error en esta lista.

También tenemos otras listas como **exploits Black List** → lista de IPs comprometidas por exploits o hijacking.

- **DBL**: Domain Block List.
- **ZEN**: Se combinan las tres listas anteriores.

CRACKEO DE PASSWORDS

No es lo mismo que el password guessing, es cual es muy lento y fácil de bloquear. Es Off-line.

John the Ripper -> de las más típicas para romper passwords hasheados.

Hashcat -> Herramienta para la recuperación de contraseñas.

/etc/shadow: aquí tenemos passwords hasheados de nuestras máquinas.
Veremos:

- una línea por cada usuario con su nombre.
- un $número$ que representa el tipo de hash (1, 2...).
- caracteres en base 64 para codificar ciertos caracteres, esto es el **salt de hasheado**. (bits aleatorios que se usan como una de las entradas en una función derivadora de claves)
- varios números con ciertos significados, como fecha de creación, tiempo para que cambie...

Rainbow tables: tablas precalculadas de hashes que se pueden utilizar en este tipo de hashes.

Si nuestro password es hola tendrá un hash, pero para que no sea igual siempre se usa el salt de hasheado. Básicamente servirá para complicar el crackeado, y se podría aplicar más veces el salt.

Un hash no es lo mismo que un generador de password hasheados. Ya que aquí se puede hashear de distintas formas, etc.

-a = modo de ataque.
-m = tipo de hash.

Extracción de los ficheros **/etc/passwd** y **/etc/shadow**
unshadow: Combinar los ficheros passwd y shadow
unshadow passwd.txt shadow.txt > **passwords**

```
hashcat -m 1800 -a 0 passwords diccionario.txt -r
```

A John podemos pasarle un diccionario, incluso con reglas, por ejemplo que nos pase el primer carácter a mayúscula, y distinto tipo de tunnings en los diccionarios.

RainbowCrack -> crackea hashes con rainbow tables.

CeWL -> Spidering en webs.
Para crackear hashes, retorna una lista de palabras que pueden ser usadas por John The Ripper, por ejemplo.

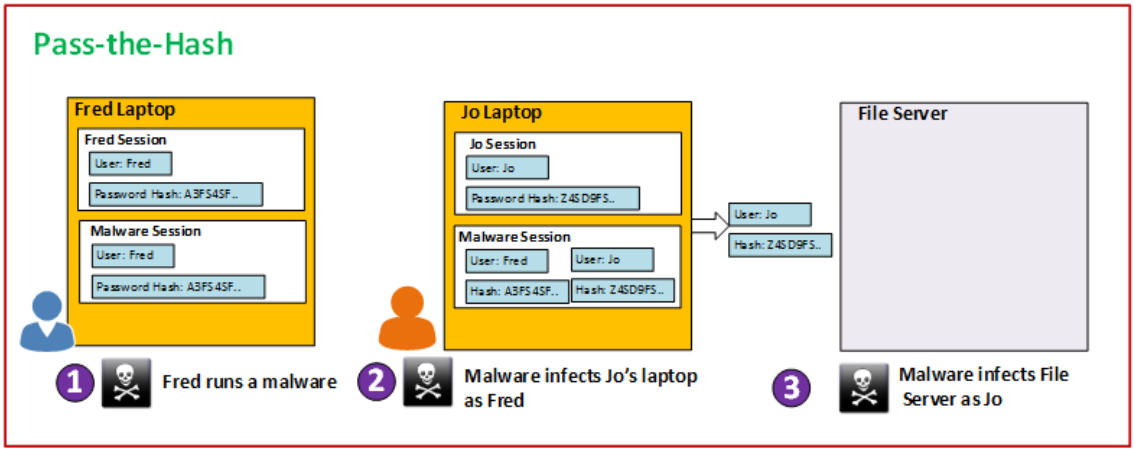
Mutator -> Un script que creará combinaciones de una palabra dada.

Se puede acotar mucho el espacio sabiendo las recomendaciones de la empresa para las contraseñas, por la longitud o si les mandan usar mayúsculas que seguramente sea el primer carácter.

The Harvester -> Programa en python al que le podemos poner un dominio y un buscador o varios y hará uso de estos para localizar correos electrónicos de ese dominio web.

Pass the hash -> Reutilizando el hash podemos hacernos pasar por otra persona, autenticandonos por ella.

Password guessing -> On-line. Se puede usar para DoS ip spoofeando una IP para que la bloqueen.
Herramientas para realizar ataques de fuerza bruta a servicios activos cliente/servidor:



- **Hydra** : este tipo de ataques son activos.
- **Medusa** : alternativa a hydra.
- **Ncrack**

Fail2Ban -> escanea los ficheros de log (/var/log/apache/error_log) y banea las IPs que muestran malas intenciones.

OSSEC -> analiza logs y aplica reglas de firewalling.

GRUB

Si no tenemos grub securizado, con acceso físico a la máquina se puede añadir un **/bin/sh** para acceder como root sin saber la contraseña.

Proteger el Grub:

- **/boot/grub/grub.cfg** -- Al principio del archivo, debemos indicar qué usuarios dispondrán de permisos para ejecutar y editar las entradas de GRUB.

1. grub-mkpasswd-pbkdf2
2. El **hash PBKDF2** de su contraseña. la metemos en **/boot/grub/grub.cfg**
3. Editamos el archivo **/etc/grub.d/00_header** y le añadimos la lista de usuarios y claves:

   cat << EOF
   set superusers=»root»
   password_pbkdf2 root <clave>
   password_pbkdf2 usuario1 <clave2>
4. ejecutar grub-mkconfig, nos generará por defecto las entradas que ya teníamos y estarán protegidas

DISCOS DUROS

Hay **sonidos** típicos reconocidos en los laboratorios de recuperación de datos, si nuestro disco duro hace algún de estos sonidos y aún tenemos acceso a los datos debemos hacer un backup lo antes posible.

Tema 7: Firewalling e IPtables.

Iptables se integra a nivel de kernel y procesa toda la paquetería de red de nuestra máquina.

Se centra en las capas 3-4 [IPv4], pero también **Ip6tables** (filtrado IPv6), **MTables** (filtrado capa 2,a nivel de MAC), **arptables** para filtrado de arp e **ipset**, que nos permite definir conjuntos de máquinas y direcciones para bloquear además de decidir reglas contra ese conjunto de direcciones.

Reglas

Dependiendo del origen y destino de los paquetes se aplicarán unas reglas u otras.

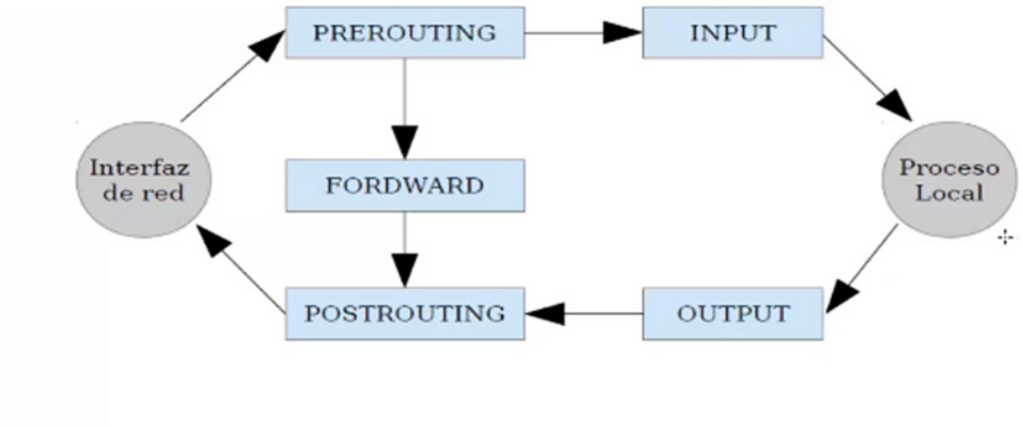
> P.ej. A los paquetes que entran directamente por un interface de red a nuestra máquina se aplicará las **reglas input**, a los que salen las **reglas output** y los que nos llegan y vamos a mandar a una máquina distinta (**forwarding**) [sería el típico firewall que hace routing, aplicando las reglas forward].

También hay **reglas NAT** (redirección de puertos, cambiar origen y destino) y antes hay reglas mangle.

Cadena

Son grupos de reglas que se van a aplicar en determinados momentos del procesado de los paquetes.
Hay **5 cadenas** → input, output, forward y otras dos a mayores:

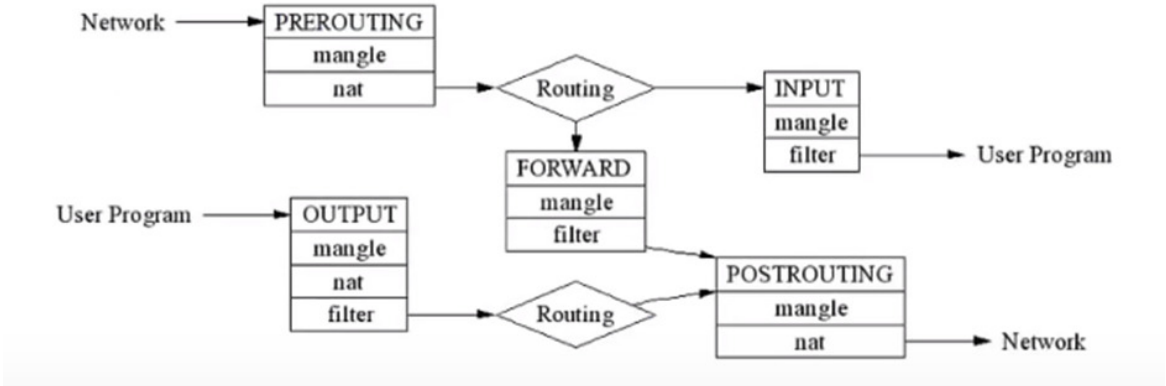
- **prerouting**: acciones que aplicamos a los paquetes antes de que se decida el routing.
- **postrouting**: antes de que se hagan las tareas de routing una vez decidido el destino.



Proceso local: recepciona los paquetes.

Si se va a hacer forwarding no llega a los procesos locales de nuestra máquina.

Tablas



- **mangle**: Tienen las 5 cadenas.
- **nat**: Tiene 3 cadenas: prerouting, output y postrouting.
- **filter**: Suele ser la tabla por defecto, con 3 cadenas: input,output y forward.

En la imagen vemos la *red*, Network es lo que nos llega.
Aplicamos la cadena prerouting, ejecutando las reglas de la tabla mangle y después de la tabla nat, a continuación pasa a Routing dónde se decide que se hace con el paquete.

- Si no es para nuestra máquina se forwardea siguiendo el mismo proceso hasta Network otra vez.
- Si no, a la cadena input aplicando las dos tablas, primero mangle y después filter para mandarlo a User Program (Proceso Local).
  De User Program saldrá a output donde se aplicarán las 3 tablas en el orden marcado, será ruteado hasta postrouting y con el mismo proceso llega finalmente a Network.

```
iptables [-t table] COMANDO CADENA condición acción [opciones]
    1       2       3       4       5       6       7
```

```
iptables -t filter -A INPUT -p tcp --dport 23 -j DROP
    1       2      3  4          5          6
```

Regla en IPtables.
Si no se especifica una tabla irá a la default -> filter.

En este ejemplo ponemos una regla que a los paquetes tcp con destino el puerto 23 de nuestra máquina los dropearemos.
Hay que tener en cuenta que las reglas dentro de cada una de las tablas se ejecutan **secuencialmente** hasta encontrar una que cumpla las condiciones.

Opciones: -j [VALOR]

- **DROP**: eliminamos el paquete y no mandamos mensaje de respuesta al equipo.
- **ACCEPT**: aceptamos el paquete.
- **REJECT**: eliminamos el paquete y mandamos un paquete ICMP al equipo que hizo la petición para indicarle que no está permitida
- **LOG:** "logueamos con un mensaje de log" y seguimos procesando.

¿Qué diferencia hay si ponemos las reglas en distintas tablas de la misma cadena?

Se ejecutarán en **distinto orden**, y esto tiene implicaciones.
Por ejemplo, las mangle son para modificar paquetes, por tanto si las ejecutamos antes o después de otras los resultados serán diferentes.

Configuración firewall

Normalmente los configuramos **persistentes**, de manera que cuando encendemos la máquina arranca con ella.
Si no es persistente, al hacer reboot se restablecerán las reglas.

Firewall básico.:

```
#!/bin/sh
## Borrado de reglas
/sbin/iptables -F     # Borra reglas
/sbin/iptables -X     # Borra cadenas
## Borra reglas de la tabla nat
/sbin/iptables -t nat -F
### Política por defecto aceptar (más sencillo, pero
## peor configuración de seguridad)
## Muchos howtos, demasiados, utilizan como
# política por defecto de OUTPUT ACCEPT
/sbin/iptables -P INPUT DROP
```

Borramos reglas al iniciar el firewall.

Para borrar reglas:

- iptables -D INPUT 8 → Borra regla 8 del INPUT
-F borra todas las reglas una por una

```
### Política por defecto aceptar (más sencillo, pero
## peor configuración de seguridad)
## Muchos howtos, demasiados, utilizan como
# política por defecto de OUTPUT ACCEPT
/sbin/iptables -P INPUT DROP
/sbin/iptables -P OUTPUT DROP
#No forwardea rute paquetes ajenos
/sbin/iptables -P FORWARD DROP
# Aceptar tráfico localhost, interface lo
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -i lo -j ACCEPT
```

No hacemos forward, **dropeamos**.

¿Qué es la política por defecto?

Por ejemplo, en el caso de **input DROP**, cuando nos llega un paquete procesamos todas las reglas, y si llegamos a la última y no ha aplicado ninguna lo dropeamos. [Es como los wrappers con host.deny con ALL:ALL].

En muchos howto ponen **output ACCEPT** por sencillez del firewall, no es aconsejable desde el punto de vista de la seguridad.
Lo mejor es dropear todo y mediante reglas hacer que funcione todo bien. Nada de políticas por defecto de accept.

```
# Aceptar tráfico localhost, interface lo
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OUTPUT -i lo -j ACCEPT
## Acepto acceso UDP a NTP y TCP a syslog
/sbin/iptables -A INPUT -s 10.10.150.cliente -d
10.10.150.mi-equipo -p udp --dport 123 -j ACCEPT
/sbin/iptables -A INPUT -s 10.10.150.cliente -d
10.10.150.mi-equipo -p tcp --dport 514 -j ACCEPT
## Acepto ssh
/sbin/iptables -a INPUT -s 10.10.150.cliente -d
10.10.150.mi-equipo -p tcp --dport 22 -j ACCEPT
```

Empezamos a meter reglas. Con -i especificamos un interfaz, aquí por ejemplo el localhost trabaja libremente.

La regla del NTP es para el servidor, como se puede apreciar (puerto destino).
Por esta regla le dejamos entrar por UDP al puerto 123 a mi compañero de prácticas, pero no a los demás.

El ssh permite que nuestro compañero de prácticas se pueda conectar a nuestra máquina.

¿Funcionaría?

No, en el **output** está todo para que se dropee en su política por defecto.
Por tanto nuestro compañero podría conectarse por input, pero después por output enviaremos un paquete ssh con distintos ip/puerto de origen/destino (estarán intercambiados).

```
/sbin/iptables -a OUTPUT -s 10.10.150.mi-equipo -d
10.10.150.cliente -p tcp --sport 22 -j ACCEPT
# Floodeo de paquetes IP spoofing no los mete
# por no tener el origen especificado
## Lo mismo para ntp y syslog

## Desde un servidor nadie debería navegar.
## No permitir abrir conexiones hacia fuera web
## En la de prácticas entre las máquinas del grupo
## de prácticas para acceso web http y https al
## compañero de la práctica 2 y 3
```

Antes se ponía el dport (destino), pero ahora podría cambiar el puerto destino que utilice el compañero, pero el **sport** (origen) va a ser sí o sí el 22, por donde entró.

Podemos especificar el **interfaz**, **IPs** y **puertos** de origen/destino.

Importante porque podemos evitar el floodeo por paquetes IP spoofing, ya que el firewall dropearía esta paquetería.

En un servidor no se puede navegar. Un rsyslog solo da su servicio de logs, nadie debería poder navegar por internet.

¿En qué fichero está la información de las IPs y Puertos que usa apt-get?

/etc/apt/sources.list

FIREWALL DE CONTROL DE ESTADO

Controla el handshake, los paquetes que no lo cumplan se dropean.
Estados handshake:

- New: el paquete ha comenzado una nueva conexión, o de lo contrario asociado con una conexión que no ha visto paquetes en ambas direcciones.
- Established: significa que el paquete está asociado con una conexión que ha visto paquetes en ambas direcciones. [conexión aceptada]

- Related: significa que el paquete está **iniciando una nueva conexión**, pero está asociado con una conexión existente, como una transferencia de datos **FTP** o un error **ICMP** [conexión genera otra conexión].

Los paquetes **related** son por ejemplo las FTP pasivas.
Tienes el puerto de control 20, pero la transferencia se hace por otro, (en este caso generalmente por encima del 1024) así que se genera otra conexión (related).

Pero no siempre es así. Por ejemplo, en una conexión ya establecida si se produce un icmp de un error en la conexión.

Y el control de estados????

```
I .....
/sbin/iptables -P INPUT DROP
/sbin/iptables -P FORWARD DROP
/sbin/iptables -P OUTPUT DROP
/sbin/iptables -A INPUT -i lo -j ACCEPT
/sbin/iptables -A OPUTPUT -i lo -j ACCEPT
#Aceptar tráfico de sesiones establecidas o tráfico
#relacionado con las sesiones establecidas
/sbin/iptables -A INPUT -m conntrack --ctstate
                        ESTABLISHED, RELATED -j
                        ACCEPT
```

por defecto DROP y aceptando localhost

```
#Aceptar tráfico de sesiones establecidas o tráfico
#relacionado con las sesiones establecidas
/sbin/iptables -A INPUT -m conntrack --ctstate
                        ESTABLISHED, RELATED -j
                        ACCEPT
/sbin/iptables -A OUTPUT -m conntrack --ctstate
                        ESTABLISHED, RELATED -j
                        ACCEPT
/sbin/iptables -A INPUT -p tcp --dport 22 -s
                        10.10.30.0/24 -d
                        10.10.102.100 -m conntrack
                        --ctstate NEW -j ACCEPT
## etc. de entrada a rsyslog, etc., etc.
/sbin/iptables -A OUTPUT -p tcp --dport 22 -s
```

-m -> carga los módulos
--ctstate -> define los estados, lo que esté ahí lo aceptas.

En la imagen no aceptaremos nada, ya que no permitiría ningún estado new.

Por eso se pone la tercera regla, para dejar establecer una **conexión SSH** (hacer el handshake) desde la VPN (10.10.30.XXX) a la máquina destino.

¿Y si nos conectamos hacía fuera?

Por ejemplo para nuestro compañero de prácticas.

```
                                ACCEPT
/sbin/iptables -A INPUT -p tcp --dport 22 -s
                                10.10.30.0/24 -d
                                10.10.102.100 -m conntrack
                                --ctstate NEW -j ACCEPT
## etc. de entrada a rsyslog, etc., etc.
/sbin/iptables -A OUTPUT -p tcp --dport 22 -s
                                10.10.30.mia -d
                                10.10.102.compi -m conntrack
                                --ctstate NEW -j ACCEPT
```

LOGS

Reglas logeo y luego dropeo.:

```
/sbin/iptables -A input -p tcp --dport 80 -m string --
string "/etc/passwd" -j LOG --log-ip-options --log-
tcp-options --log-prefix "password access"
```

```
/sbin/iptables -A input -p tcp --dport 80 -m string --
string "/etc/passwd" -j DROP
```

O accesos no autorizados al ssh.:

```
# /sbin/iptables -A INPUT -p tcp -m tcp --dport 22 -j
LOG --log-prefix 'Intento fallido acceso ssh.: ' --
log-level 4
```

módulo string → permite hacer cosas con el **payload** del paquete, por ejemplo, buscar determinado contenido.

--string "/etc/passwd" → busca (en el contenido del paquete de las páginas web en este caso) en el payload del paquete si aparece en algún momento el término "/etc/passwd".

Si es así entonces haremos un log, con el mensaje de log "password access" y después lo dropea.

También podríamos registrar accesos no autorizados a SSH.
Como se le han aplicado todas las reglas al paquete y ninguna ha matcheado, salta esta al final (previa al dropeo) y loguea el acceso no autorizado.

Control de estado UDP

Se registra cuando se abre una conexión nueva desde el conntrack dentro de Iptables.
Aunque no comprueba las etiquetas, Iptables permite una conexión que no esté relacionada con una petición previa.

Puertos múltiples

```
-m multiport --dport 22,80,XX
```

Aplicar la misma regla a varios puertos sin tener que repetirla para cada uno.

Ocultar el firewall

Por defecto, una máquina no dropea paquetes.

Si es por TCP, en vez de dropear responde con un **TCP RST** (flag RST activo) y si es por UDP respondería con un **ICMP Port Unreachable**.

Podemos implementar esto mismo de forma que no se note que estamos usando un firewall. Con las reglas Iptables se usaría un **reject**.

```
iptables -A INPUT -p TCP -j REJECT --rejectwith tcp-reset
iptables -A INPUT -p UDP -j reject --rejectwith icmp-port-unreachable
```

`iptables -L` → Nos lista todas las reglas que tenemos dadas de alta en el firewall.
`iptables -n` → No resuelve DNS.
`iptables -L --line -number` → Pone números de cómo están numeradas las reglas.

Para hacer el firewall persistente se usaría el paquete **Iptables Persistent**.

No es necesario. Se gestionaría con el systemctl.

Se podría hacer que el script se ejecutase en booteado o en /etc/network/interfaces.

Atención al orden → se ejecuta todo el script secuencialmente.

Es mejor poner las políticas por defecto al final, si no ya empezaría a dropear todo y quizás nos tiramos el ssh.

> Hay gente que pone un pequeño **sleep** de unos minutos al final del script para así probar el firewall y que después se desactive solo.
> Poniendo las políticas por defecto a accept y borrando las reglas.

IPv6

Ip6tables

Tener en cuenta las Link-Local de ens33 y ens34 y las direcciones tun6to4 (2002), así podremos poner reglas en IPv6.

FwSnort → Firewall Snort. Capa de aplicación IDS/IPS con iptables.

Permite integrar reglas del sistema de detección de intrusiones **snort** como reglas iptables.

LOGS (rsyslog)

/etc/rsyslog.conf → dónde montamos cliente/servidor.

Podemos poner: :msg, contains, “intento fallido ssh” /log/fichero

Cada vez que Iptables detecta un acceso no autorizado genera un track de log de rsyslog, **nivel 4 warning**.
Se almacenará en el fichero mencionado.

Netfilter

Framework que permite actuar con paquetes de nuestra máquina en distintas etapas.Hay varios proyectos, dos principales:

- **iptables**
- **nftables**

Nftables → Diseñado para solventar algunas deficiencias de iptables.

Es muy fácil pasar de [reglas iptables → reglas nftables].

También hay utilidades para pasarlas de una a otra: **iptables translate** o **ip6table translate**.

Con el translate iríamos regla a regla, pero se podría hacer todo de golpe:

- **save fichero** → para que guarde todas las reglas iptables en un fichero
- **restore translate [fichero]**→ haciendo la traducción de todas las reglas de este.

Se integra todo en la misma herramienta (filtrado capa 2,3, ip 4 y 6...).
Añade las familias de direcciones de forma que se aplica una regla a una familia (si no se especifica se usa la **default = IPv4**).

Familias:

IPv4	IPv6	Inet (IPv4 e IPv6)	Arp	Bridge (Capa 2)

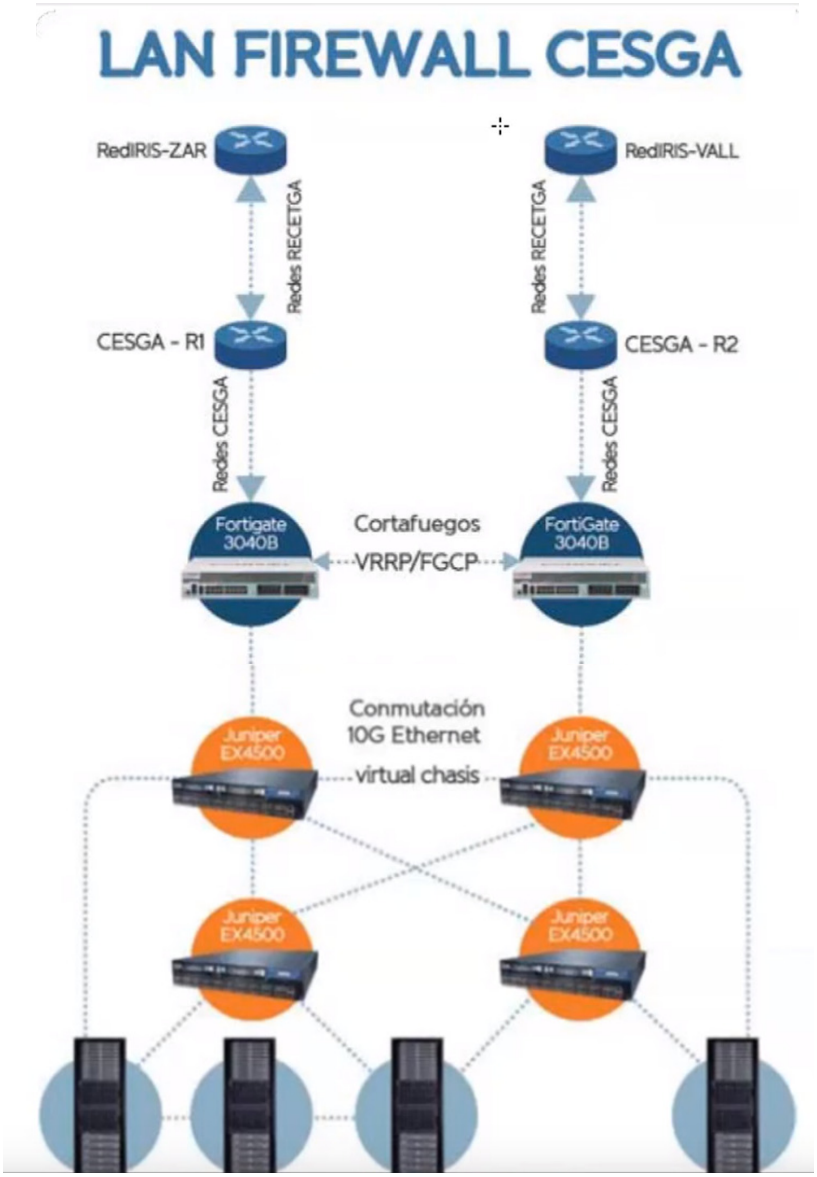
Otros proyectos:

- Psad → Conjunto de daemons que analiza los logs que crea iptables.
  Incorpora un conjunto de reglas (muchas vienen de snort) para detectar todo tipo de cosas (routing, backdoors, fingerprinting...) al igual que hace OSSEC.

- FirewallBuilder → Configurar y administrar firewalls mucho más fácil, puede usarse para configurar Iptables.
- Easyfwgen → Le especificas interfaces, servicios y más datos y la página genera un script con las sentencias para proteger nuestra máquina.
- ufw - Uncomplicated Firewall → Realmente no es un firewall, es una API.

  Se usa la línea de comandos, pero tiene una sintaxis muy sencilla. Especificas lo que quieres y él interacciona con IPTables.

CESGA



Hay redundancia, dos líneas distintas que llegan al cesga y enlazan dos firewalls (redundancia de firewalls, por si falla uno).

Después del doble firewalling engancha con los switches Juniper.

Esta arquitectura es la habitual.

Protocolos entre los firewalls:

- VRRP: protocolo que permite gestionar routing redundante con tolerancia de fallos, para que a todos los efectos los dos routers sean vistos como si fuesen el mismo.

- FGCP: protocolo propietario de los propios firewalls para permitir la alta disponibilidad. Mediante él se comunican todo el rato por si cae uno.

Tema 8: Monitorización y Accounting

acct

Genera BDs donde se van a ir registrando toda la monitorización del sistema. Añade determinadas entradas a los sistemas de CRON para hacer determinados procesos como rotar BDs diaria y semanalmente.

Comandos:

- ad *user* -> total de horas de ese usuario a mi máquina.
- ad -d -> listado de horas por cada día.
- lastcomm -> toda la relación de todos los comandos que se han ejecutado en la máquina (ettercap, ls...), con su tiempo de CPU, fecha y hora de ejecución.
  Si aparece:
  S : ejecutado por root.
  F : proceso generado mediante un fork.
  D : proceso que ha fallado, generando un fichero core (volcado de memoria del proceso por si se le quiere hacer por ejemplo un debug)
  X : proceso finalizado por una señal.
- sa -> → resumen de todo lo que he ejecutado en la máquina (distinto al lastcomm).
  Por ejemplo:
  Daría sólo una línea para el *ettercap* con:
  El número de veces que se ha ejecutado, el tiempo real consumido, número medio de operaciones de E/S y **k** [unidad de cpu storage integral en kilocore/segundos, kbytes por segundo].
- last *user* -> fechas y horas de conexiones a la máquina, terminal, ip de conexión… lo saca de **/var/log/wtmp**.

sysstat

Proporciona información:

- **Instantánea** (parámetros de la red en un periodo indicado)
- Mantener un **histórico** de toda la información de la máquina.
  Para todo tipo de análisis de rendimiento, búsqueda de cuellos de botellas, problemas de seguridad, implicaciones de un ataque DoS...

- iostat → Monitoriza la carga del dispositivo de E/S del sistema al observar el tiempo que los dispositivos están activos en relación a las tasas de transferencia.
  steal time (% de tiempo que una cpu virtual espera por una cpu real según la propia gestión del hipervisor).

También da información de los sistemas de almacenamiento, como el número de kbytes leídos por segundo, escritos por segundo…
Tps, número de transferencias por segundo en ese dispositivo.

iostat 3 10 -> 10 muestras con toda la información de la máquina cada 3 segundos.

SAR

Todo tipo de información de CPU, red, kernel…

sar [periodo] [nº muestras] **/var/log/sa/saXX**

Se le puede indicar qué información se quiere mostrar.

P.ej : **sar -r 5 8** : muestra información de memoria (memoria utilizada sin contar el kernel)

Otro parámetro → **%commit**: indica el porcentaje de memoria necesario para la carga de trabajo actual en relación con la cantidad de total física de memoria RAM y swapping que tenemos.

Puede estar al 100%, esto es posible dado a que en la gestión de procesos que usa el kernel se puede usar overcommit y si se queda sin memoria tiene que cerrar procesos [asignar memoria virtual sin garantía de que exista almacenamiento físico para la misma.]

Si un parámetro está en rojo significa que hay un problema.

Información del swapping → **sar -S 2 4** : 4 muestras cada 2 segundos sobre el swap del sistema.

También se puede almacenar un histórico para procesarlo, etc.

/etc/default/systat → enable true: va a activar todas las BD y va a empezar a recolectar información.

sa2 → resumen diario.

Se puede monitorizar por ventanas, pidiendo información entre determinadas horas

satf → permite coger cualquier BD de monitorización y exportarlo a otro formato, como pasarlo a ASCII separados con ;

Se puede generar XML también.

isag

Herramienta para la representación gráfica de toda está información.
Seleccionas una BD y puedes hacer gráficas según horas, uso de CPU..

Comandos y herramientas del sistema

- ls cpu → información de la cpu, su arquitectura, cachés, cores, modelos, modo de operación (32/64 bits), little/big endian, id del fabricante, modelo exacto, MHz, MIPS..
- /proc/cpuinfo
- uname -a → nivel versión kernel.
- uptime → tiempo encendido desde el último booteado, nº de users y parámetros de carga media del sistema en los últimos 5/10/15 minutos, **tiene que estar por debajo de 3 s**.
- dmesg → toda la información de booteado del sistema, /var/log/dmesg.

- lsmod → muestra los módulos cargados por el sistema.
- lspci → información de los buses pci y dispositivos conectados.
- time [programa/comando] → información del programa/comando.
- ps → procesos.
- df → proporciona información sobre ficheros y discos duros, como puntos de montaje, etc. info almacenamiento.
- du → ocupación de ficheros y directorios.
- who → conjunto de usuarios conectados y + info sobre ellos.
- gcc → -g incluye información para poder debuggear.
  -pg conjunto de información para gprof.
- lld → librerías compartidas que necesita cada programa.
- fsck → para analizar un fichero de ficheros, y también puede intentar repararlo.
- vmstat → informar las estadísticas de memoria virtual y proporcionar información sobre eventos del sistema.
- apt install isag -> app gráfica.
- **gprof** → gprof produce un perfil de ejecución de programas C, Pascal o Fortran77.
- gdb → ensamblar y desemblandar cosas, lo tengo en código ensamblador.
- top → monitoriza todo tipo información de cpu, memoria, procesos…
- atop → evolución del top, con muchísima información.
- mpstat → Nos da otro tipo de información, más o menos en la línea del iostat.

Herramientas de establecimiento de cuotas

¿Qué es asignar cuota de disco?

Cuantos i-nodes, tamaño en disco y tamaño por fichero le dejamos a un usuario.

Se pueden configurar con unos ficheros de configuración y comandos. Es **necesario**, no podemos dejar que cualquier usuario nos llene el disco duro como quiere.

Con esto podemos **bloquear un fork bomb**.

Es recomendable establecer cuotas a todos los niveles.

- ulimit → ver cuotas.
- ulimit -a → ver cuales pones.

Cacti

Zabbix : Sistema de Monitorización de Redes.. Está diseñado para monitorizar y registrar el estado de varios servicios de red, servidores y hardware de red.

Munin

Nagios : Clásico de monitorización distribuida a todos los niveles.

Ntop : Monitorización de red y ancho de banda consumidos.

MRTG : Un clásico. muy vinculado a las troncales de red, el cesga lo usa.

Ossim : Conjunto de herramientas open para la administración de eventos de seguridad.

SIEM (Security Information and Event Management)

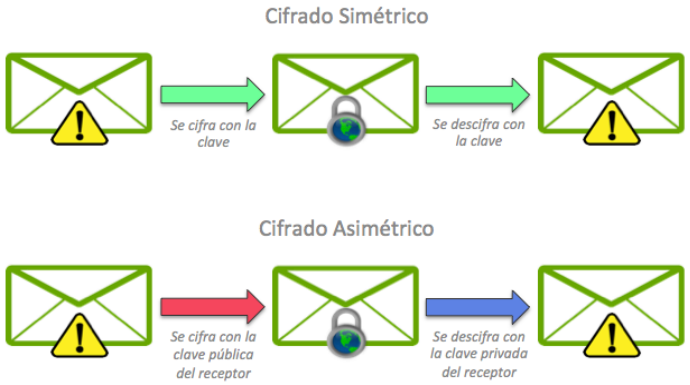
Splunk
Prelude
Microfocus
Logrhythm

Anexo

¿Cómo configuro con iptables tantas conexiones? **[Examen]**
hashlimit

- Cifrado Simétrico: misma clave para cifrar y descifrar, es necesario compartir la clave con el destinatario.
  Cualquiera que tenga la clave puede descifrar el mensaje.
  Esta técnica es usada para encriptar la conexión.

- Cifrado Asimétrico: clave pública para cifrar los datos y una clave privada que se usa para descifrarlos.
  Un mensaje cifrado por la clave pública de una máquina, sólo puede ser descifrado por la misma clave privada de la máquina.



Los algoritmos simétricos son muchos más rápidos que los asimétricos.

Algoritmo Diffie Hellman

Usado principalmente para el intercambio de claves.
Permite acordar una clave secreta entre dos máquinas, a través de un canal inseguro, con la que posteriormente cifrar las comunicaciones entre dos máquinas.

La firma digital de un certificado se hace siempre con la **clave privada**. Si se quiere enviar un cifrado asimétrico, se hará con la **pública** del otro.