Mitmproxy

Introducció

Mitmproxy (https://mitmproxy.org) és el nom d'una suite formada per tres programes independents:

* *mitmproxy* és un proxy HTTP/S interactiu dissenyat per funcionar com a "man-in-the-middle" entre clients i servidors web. Permet interceptar, examinar, modificar i reenviar trànsit HTTP/S de forma interactiva. Amb ell es pot modificar el contingut HTML+CSS+Javascript provinent de servidors, es poden injectar nous elements dins d'aquest contingut, es poden redireccionar les peticions dels clients o directament contestar-les sense reenviar-les al destí legítim, es poden filtrar les dades enviades per les peticions i/o les respostes corresponents, es poden obtenir i/o modificar els valors de les capçaleres de client i servidor, etc. Tot això des d'una interfície basada en Ncurses.

* *mitmweb* és una interfície web per a *mitmproxy*. Permet també examinar i modificar interactivament el trànsit HTTP/S però fent servir el navegador com a marc de treball en comptes del terminal. Per accedir a l'entorn ofert per *mitmweb* s'ha d'escriure a la barra de direccions la URL http://127.0.0.1:8081 (que és per on *mitmweb* escolta per defecte), però tant la IP com el port d'escolta es poden modificar amb *--web-iface x.x.x.x* i *–web-port nº*, respectivament.

* *mitmdump* és la versió no interactiva de *mitmproxy*. Permet visualitzar, gravar i transformar programàticament el trànsit HTTP/S que el travessa. És el programa que farem servir quan volguem executar scripts amb l'API de Mitproxy

En resum, les principals característiques de les eines anteriors (les quals s'expliquen més a l'article https://docs.mitmproxy.org/stable/overview-features) són :

-Intercepten i mostren les peticions i respostes HTTP/S, permetent a més filtrar la sortida obtinguda
-Generen certificats TLS "al vol" per aconseguir el desxifrat del trànsit HTTPS en temps real
-Editen el trànsit HTTP/S sobre la marxa (interactivament) o via scripts Python (programàticament), ja siguin les capçaleres o el cos de peticions o de respostes convenientment indicades
-Conserven converses HTTP completes per a la reproducció i anàlisi posteriors

A més de la paritat de característiques entre aquestes tres eines; totes comparteixen un mecanisme de configuració comú i la majoria de les opcions de línia de comandes són iguals també.

NOTA: Els protocols suportats per Mitmproxy són HTTP/1.1, HTTP/2, HTTP/3, WebSockets i, a més, també DNS i tràfic TCP i UDP genèric. A https://docs.mitmproxy.org/stable/concepts-protocols podeu trobar més informació. D'altra banda, Mitmproxy té diversos "modes de funcionament: com a proxy transparent, com a proxy invers, com a proxy SOCKS, com a proxy Wireguard, només funcionant com a resolver DNS, etc, etc La documentació específica de cadascun d'aquests modes es pot consultar a https://docs.mitmproxy.org/stable/concepts-modes però nosaltres en principi només veurem el mode "transparent HTTP/S".

Instal.lació de Mitmproxy i configuració com a proxy transparent HTTP/S

Per posar en marxa qualsevol dels programes que formen part de la suite Mitmproxy no cal instal.lar res: només descarregar-se (per exemple amb la comanda *wget* o directament des del navegador) l'arxiu "tar.gz" amb la versió més nova disponible de https://mitmproxy.org/downloads i descomprimir-lo al nostre sistema (recordeu que això es pot en el terminal així: *tar -xzf mitmproxy-XXX.tar.gz*). Fent això ens trobarem directament amb tres executables (*mitmproxy*, *mitmweb* i *mitmdump*) que ja estaran disponibles per usar-se.

NOTA: Per veure com fer altres instal.lacions alternatives, mireu: https://docs.mitmproxy.org/stable/overview-installation

El mode més habitual de funcionament de Mitmproxy és el de "proxy HTTP/S transparent", així que a continuació mostrarem els passos per posar en marxa Mitmproxy en aquest mode (les instruccions oficials es troben a https://docs.mitmproxy.org/stable/howto-transparent) i aconseguir que rebi el tràfic desitjat:

1.-Per a què el proxy on s'executarà Mitmproxy funcioni de forma transparent, primer cal que l'establim com a porta d'enllaç per defecte del/s client/s del/s qual/s volguem inspeccionar el su tràfic HTTP/S. Això es podria aconseguir, si gestinem el servidor DHCP de la LAN, indicant-ho a la seva configuració. No obstant, si no podem administrar cap servidor DHCP, un alternativa que tenim és realitzar un atac "ARP Spoofing". Ja hem vist anteriorment eines (com Nping o Scapy) que ens permeten fer això gràcies a fabricar "a mà" respostes ARP adients, però ara farem servir, per veure una altra possibilitat diferent, la comanda especialitzada *arpspoof* (pertanyent al paquet "**dsniff**"). Per tant, un cop instal·lat aquest nou paquet a la màquina atacant, en un terminal mantindrem executant-se tota l'estona lacomanda següent (que enganyarà a la víctima fent creure que nosaltres som la porta d'enllaç)...:

sudo arpspoof -i enp0s3 -t ip.vic.ti.ma ip.gat.ew.ay

...i en un altre terminal diferent (si estem en un servidor podem pulsar ALT+CTRL+Fnº per obrir-ne un de nou, si aquest servidor està funcionant en VirtualBox, llavors la combinació serà CTRLdret+Fnº) haurem de mantenir executant-se tota l'estona aquesta altra comanda (que enganyarà a la porta d'enllaç fent creure que nosaltres som la víctima):

sudo arpspoof -i enp0s3 -t ip.gat.ew.ay ip.vic.ti.ma

> NOTA: "enp0s3" representa la tarja de xarxa de la nostra màquina atacant
> NOTA: Aquest pas no seria necessari (i el següent tampoc!) en el cas de voler executar Mitmproxy com un "proxy" estàndard (és a dir, no transparent). Això sí, llavors hauríem de configurar el/s navegador/s dels clients dels quals volguéssim inspeccionar les seves peticions adientment per tal de què fessin servir explícitament la màquina on estarà executant-se Mitmproxy com a proxy HTTP/S. Això a Firefox es pot fer anant a a *about:preferences* i clicant sobre el botó "Paràmetres" de l'apartat "Paràmetres de xarxa" del final: apareixerà un quadre on, després de triar l'opció " Configuració manual del servidor intermediari" podrem indicar la IP i port (p. def. el 8080) del "servidor" Mitmproxy on el navegador reenviarà totes les peticions generades per l'usuari.

2.-No oblidem habilitar l'"IP-forwarding" (si no, com ja sabem, la víctima perdrà la connectivitat i l'atac es convertiria en un de tipus DoS en impedir-se als clients l'accés a Internet). Recordem que això es pot fer de forma temporal executant (en un altre terminal), la comanda següent:

sudo sysctl net.ipv4.ip_forward=1

> NOTA: És recomanable executar també aquesta altra ordre a la màquina de l'atacant: *sudo sysctl net.ipv4.conf.all.send_redirects = 0* (o editar el fitxer de configuració de sysctl associat) per tal de no informar a la màquina víctima de què hi ha una ruta més curta cap a l'encaminador real (saltant-se el servidor intermediari)

3.-Les peticions HTTP per defecte van dirigides al port 80 del seu destí mentre que les peticions HTTPS van dirigides al port 443. Mitmproxy, per la seva banda, escolta per defecte al port 8080. Suposant que Mitmproxy estigui funcionant en mode transparent, si volem que Mitmproxy rebi en aquest port 8080 les peticions dirigides al port 80 i al port 443, haurem de configurar prèviament el tallafocs del sistema on Mitmproxy es vagi a executar (que és qui rebrà les peticions als ports 80 i 44) per tal de què aquest les redireccioni convenientment al port 8080. La manera més fàcil de fer-ho és executant la següent comanda en un terminal disponible:

sudo nft -f redir80443.txt

..on "redir80443.txt" hauria de tenir un contingut semblant a aquest (a més de les eventuals regles que hi puguin haver a les eventuals cadenes "input", "output" i/o "forward" de la taula "filter"):

```
#!/usr/sbin/nft -f
flush ruleset
table ip nat {
        chain prerouting {
                type nat hook prerouting priority 100 ;
                iifname enp0s3 tcp dport 80 redirect to 8080
                iifname enp0s3 tcp dport 443 redirect to 8080
        }
}
```

NOTA: "enp0s3" representa la tarja de xarxa per on Mitmproxy estarà escoltant. Per defecte Mitmproxy escolta a totes les tarjes de xarxa que trobi al sistema. Si es vol restringir la seva escolta a una tarja en particular, es pot afegir el paràmetre *--listen-host x.x.x.x* (on "x.x.x.x" representa la direcció IP -de les que puguin haver a les diferents tarjes presents al sistema- a la què Mitmproxy es vincularà)
NOTA: L'efecte de la comanda anterior només dura mentre el sistema estigui encés. Si es vol que aquestes regles siguin permanents, cal modificar el servei Nftables per a què llegís el fitxer en qüestió. Això es pot fer editant el seu arxiu "*.service" corresponent (per més informació, consulta el document sobre el tallafocs de Linux).
NOTA: Podem canviar el port d'escolta de Mitmproxy amb el seu paràmetre *-p nº* però això no és solució perquè hem de poder rebre les peticions dirigides a dos ports: si fem *-p 443* ens estarem perdent les peticions HTTP i si fem *-p 80* ens estarem perdent les peticions HTTPS.

4.-Finalment, executarem el Mitmproxy pròpiament dit, en mode transparent (sense ser "root"!), així:

./mitmproxy -m transparent --showhost

NOTA: El paràmetre *--showhost* serveix per fer que Mitmproxy mostri els noms dels servidors als quals els clients fan les seves peticions (a partir del valor de les seves respectives capçaleres "Host:" capturades). Si no s'indica, Mitmproxy mostrarà simplement direccions IP.
NOTA: En el cas de no voler executar Mitmproxy com un proxy transparent sinó com un proxy estàndard (perquè els navegadors ja s'hagin configurat adientment, veure darrera nota del punt següent, el paràmetre *-m transparent* no s'haurà d'indicar.

5.-A partir d'aquí, quan Mitmproxy rebi una petició HTTPS, estigui funcionant ja sigui en mode "estàndard" o transparent, el que fa és aturar aquesta petició per fer-la ell mateix al servidor final, (obtenint d'aquesta manera el certificat TLS original del servidor legítim, com l'obtendria un client qualsevol); amb la informació recollida d'aquest certificat (concretament, el "CommonName" i el "subjectAltName"), Mitmproxy genera "al vol" un altre certificat amb la mateixa informació però signat per ell mateix, que és el que reenviarà al client que ha fet la petició (en comptes del certificat del servidor "legítim"; noteu en aquest sentit que Mitmproxy crearà un certificat "impostat" diferent per cada servidor "legítim" diferent al que s'hi vulgui accedir). A partir d'aquí, Mitmproxy reprèn la connexió entre aquest client i el servidor remot concret "com si no hagués passat res" però ja estarà posicionat com a"man in the middle": tot el que rebi del client ho veurà perquè el client farà servir precisament el certificat "impostat" de Mitmproxy per xifrar la seva comunicació...amb el propi Mitmproxy! i tot el que rebrà del servidor ho veurà igualment perquè Mitmproxy hi actua contra ell com un client estàndard.

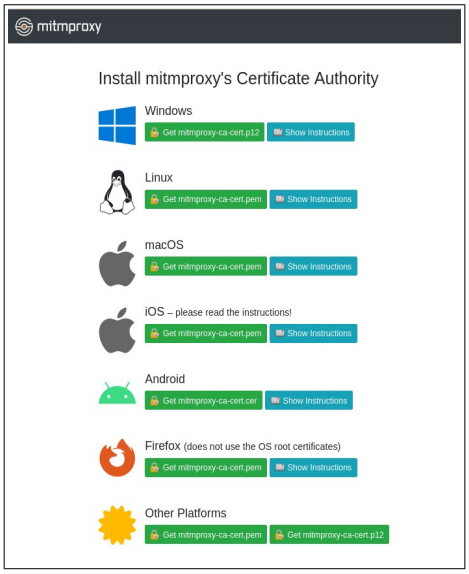
No obstant, actualment els clients no "s'empassen" tan fàcilment un certificat que no hagi sigut signat per una CA legítima "confiable" (és a dir, per una CA el certificat arrel de la qual no estigui integrat a la seva configuració). És per això que, si ens disposem ja a navegar a través de Mitmproxy, els navegadors emetran un missatge d'error TLS de tipus "SEC_ERROR_BAD_SIGNATURE" per cada domini HTTPS al que volguem accedir. Per tant, cal fer uns passos extra més.

NOTA: Depenent del lloc web que visitem, en comptes de mostrar-se el missatge d'error "SEC_ERROR_BAD_SIGNATURE", és possible que aparegui un missatge similar però que permeti afegir una "excepció" per tal de deixar que el navegador "confii" en aquell certificat concret (per aquell domini en concret) per sempre més sense que torni a aparèixer cap advertència (és a dir, per tal d'integrar la CA de Mitproxy com una CA vàlida més, per aquell domini en concret). En aquest cas, ja estaríem, però cal dir que cada cop són menys els servidors TLS que permeten aquesta possibilitat d'establir excepcions degut a la popularització de la tecnologia HSTS, que és precisament la que la prohibeix.
NOTA: Per a què Mitmproxy pugui actuar com a CA, la primera vegada que alguna de les comandes de la suite Mitmproxy s'executa, es creen els següents fitxers automàticament a la carpeta "**~/.mitmproxy**" (també creada en aquell moment):

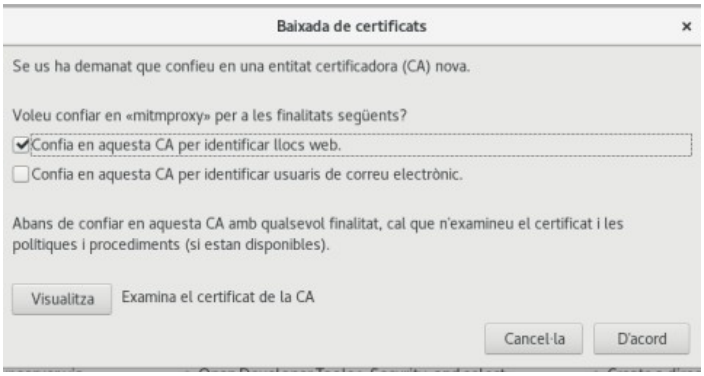
"**mitmproxy-ca.pem**" : El certificat "arrel" de la CA Mitmproxy en format PEM. Inclou a dins la clau privada també, així que és molt recomanable mantenir aquest fitxer protegit!
"**mitmproxy-ca-cert.pem**" : El certificat "arrel" de la CA Mitmproxy en format PEM. Aquest és el fitxer que caldrà distribuïr als diferents clients per tal de què confiïn en aquesta CA (i no aparegui el missatge d'avís als navegadors, per exemple). Més sobre això en els següents paràgrafs. Existeix un altre fitxer anomenat "**mitmproxy-ca-cert.cer**" que és el mateix, només que certs dispositius Android esperen l'extensió "cer" en comptes de "pem"
"**mitmproxy-ca.p12**" : El certificat "arrel" de la CA Mitmproxy en format PKC12. Per usar en sistemes Windows. També està l'arxiu "**mitmproxy-ca-cert.p12**", que és el mateix però en format PKCS12 (també per usar en sistemes Windows)

NOTA: Per raons de seguretat, els fitxers anteriors es generen únicament una vegada (la primera que s'executa algun binari de la suite Mitmproxy i no són comuns amb altres instal.lacions de Mitmproxy que es puguin realitzar en el mateix o altres dispositius.
NOTA: Es poden tenir els certificats en una altra carpeta diferent de "~/.mitmproxy" sempre i quan en executar *mitmproxy* s'indiqui la seva ruta amb el paràmetre *--set confdir=/ruta/carpeta/certs* Si la carpeta en qüestió no contingués els certificats necessaris, Mitmproxy els crearà allà automàticament.

Concretament, hem d'aconseguir que el fitxer "mitmproxy-ca-cert.pem" (mencionat a la NOTA anterior) sigui acceptat als clients com a certificat "arrel" de la CA Mitmproxy. Per fer això, si no tenim el control de les màquines clients a la pràctica és difícil: haurem de recórrer o bé a l'"enginyeria social" (és a dir, enganyar a algú per a què faci el que volem sense saber què està fent) o bé a un exploit que pugui ser executat en remot o bé també per algú que no en sigui conscient. Si tenim el control de les màquines clients llavors aquest pas és un tràmit (per exemple, per integrar el certificat "arrel" al Firefox només cal obrir la direcció *about:preferences#privacy* i clicar sobre el botó "Mostra el certificats; allà hem d'anar a la pestanya "Entitats" i clicar sobre el botó "Importa" per triar el fitxer "mitmproxy-ca-cert.pem" i ja està). De totes formes, per fer aquest procediment més senzill i ràpid (sobre tot si tenim molts clients que volem que confiin en la CA Mitmproxy), Mitmproxy ofereix una manera més còmoda de fer aquest tràmit, que és obrint el navegador en la màquina client a la qual volem integrar el certificat "arrel" i escriure a la seva barra de direccions la URL http://mitm.it S'hauria de veure (si tenim el *mitmproxy* en marxa) la següent pàgina:



A partir d'aquí només cal clicar sobre la icona adient per seguir les instruccions. Concretament, en el cas de voler integrar el certificat dins del navegador Firefox funcionant sobre Linux (que és el que farem) haurem de descarregar el fitxer "mitmproxy-ca-cert.pem" disponible sota l'opció "Firefox" i, seguidament, tal com hem dit, escriure a la barra de direccions l'adreça *about:preferences#privacy* i clicar sobre el botó "**Mostra el certificats**" que allà apareixerà per tal d'anar a la pestanya "**Entitats**" i clicar sobre el botó "**Importa**"; es mostrarà llavors una finestra emergent com la següent...:



...on s'haurà de xequejar com a mínim la primera opció (tal com es mostra a la imatge) i acceptar.

NOTA: El pas descrit al punt anterior és només per si volem integrar el certificat arrel de la CA Mitproxy a la llista de certificats arrel gestionada pel navegador Firefox en particular. Si el volguéssim integrar a la llista de certificats arrel gestionada pel sistema en general (que és diferent de la particular del Firefox) i, així, possibilitar que aquesta CA del Mitmproxy la pugui fer servir altres programes més enllà d'aquest navegador concret, les instruccions per fer-ho també estan senyalades a la pàgina mitm.it. Concretament, a Ubuntu consisteixen en executar aquestes dues comandes: *mv mitmproxy-ca-cert.pem /usr/local/share/ca-certificates/mitmproxy.crt && sudo update-ca-certificates*
NOTA: En el cas d'usar un altre sistema operatiu (Windows, MacOS, etc), les instruccions concretes es troben descrites a https://docs.mitmproxy.org/stable/concepts-certificates. Per sistemes Android (i els seus simuladors) tenim una pàgina específica: https://docs.mitmproxy.org/stable/howto-install-system-trusted-ca-android

Primeres proves

Un cop realitzats els cinc punts anteriors, podrem veure com a la pantalla oberta del *mitmproxy* han d'aparèixer llistades totes les peticions HTTP/S que aquest proxy està rebent del/s client/s (o bé atacats via "ARP Spoofing" o bé configurats per fer servir el proxy explícitament) i que està reenviant als servidors finals corresponents per a què ningú noti res.

NOTA: En realitat, cada línia que mostra *mitmproxy* a la seva pantalla inicial és un "flow", que consisteix tant en una petició com la seva corresponent resposta (és a dir, una parella petició<->resposta).

La informació que mostra aquesta pantalla inicial de Mitmproxy és força evident: l'hora quan s'ha detectat la petició, el protocol de la petició (HTTP, HTTPS, WSS -WebSockets-, etc), el mètode HTTP usat en la petició, l'URL demanada (separada entre nom DNS del servidor remot i el "path" de la petició) , el codi de resposta obtingut, el "Content-Type" del recurs obtingut, el seu tamany i el temps que s'ha trigat en obtenir-se del servidor. A mesura que el/s navegador/s "inspeccionats" vagin navegant, aquesta pantalla inicial s'anirà omplint de més i més "flows".

Un cop comprovat que la intercepció funciona, el primer que hem de saber és com moure'ns dins de la pantalla inicial de Mitmproxy (i dins d'altres que ara mateix no veiem!) per tal d'obtenir tota la informació que necessitem de tot el trànsit capturat i, si fos el cas, per tal de poder-lo modificar.

Referència de tecles

Les tecles per treballar dins la consola general de *mitmproxy* més habituals són:

*__Cursors amunt/avall__: Es mou amunt i avall (un a un) pels "flows" que surten a la pantalla inicial
*__Av Pag/Re pag__: Es mou amunt i avall (de pàgina en pàgina) pels "flows" que surten a la pantalla. Inicial. La tecla **SPACE** en aquesta pantalla és equivalent a Av Pag.
*__F__: Segueix els "flows" en temps real a mesura que apareixen en pantalla
*__G__ : Va al darrer "flow"
*__g:__ Va al primer "flow"
*__z__ : Esborra (de la pantalla i de la RAM) tots els "flows"
*__d__ : Esborra el "flow" actualment sel.leccionat
*__q__ (o **ESC**): Surt del programa
*__Q__ : Surt del programa inmediatament
*__?__ : Mostra l'ajuda (bàsicament "keybindings" generals i els concrets de la pantalla que estigui activa en aquell moment, a més dels filtres disponibles). Per sortir-ne, "q" D'altra banda, "**C**" mostra la llista de totes les comandes interactives (aquesta llista es pot obtenir també amb el paràmetre *--commands* de *mitmproxy*; per més informació sobre com fer servir les comandes en general, però, podeu consultar https://docs.mitmproxy.org/stable/concepts-commands).

NOTA: Les comandes interactives mostrades per "**C**" es poden executar si prèviament pulsem els dos punts ("**:**"). Més endavant les estudiarem però, de totes formes, moltes d'elles (les més habituals) tenen associat un "keybinding" (que també veurem) i, per tant, no se solen escriure explícitament. L'associació entre keybinding i comanda es pot veure amb "**K**". Per desfer una comanda sense arribar a executar-la, cal pulsar **ESC**
NOTA: A https://docs.mitmproxy.org/stable/addons-commands s'indiquen les instruccions de com crear els nostres propis comandes

*__ENTER__ (o __P__): Selecciona un "flow" per veure'n els detalls de la petició i resposta corresponents, (entrant d'aquesta forma en una pantalla diferent, que anomenarem "pantalla de detalls"). Entre aquests detalls podem destacar, a més de la data i hora de la petició, les capçaleres i el cos de la petició (si n'hi hagués) per una banda, les capçaleres i el cos de la resposta (si n´hi hagués) per una altra, i finalment les dades sobre la connexió TLS. Per moure's entre les tres columnes principals ("Request", "Response" i "(TLS) Details" es pot usar **TAB** o bé els **cursors dreta/esquerra**. Per sortir de la pantalla de detalls i tornar a la pantalla inicial, "**q**" o "**ESC**"

***SPACE**: En la pantalla de detalls aquesta tecla serveix per anar al següent "flow" (en ordre cronològic); per anar a l'anterior caldria polsar en canvi la tecla "**p**".

*"**/**" : En la pantalla de detalls, busca una cadena (case-sensitive) dins el contingut. Amb "**n**" es va a la següent instància trobada. Amb "**N**" a l'anterior.

*__m__ : En la pantalla de detalls, canvia el mode de visualització del cos de la resposta (hex, JSON...). Per defecte és "auto"

*__E__ : Mostra la pantalla d'events. Per sortir-ne, "q" (o "ESC").

*__f__ : Estableix un filtre de visualització de "flows". Els filtres possibles es mostren al següent apartat Per cancel·lar, pulsar "ESC"

*__i__ : Estableix un filtre d'intercepció de "flows". La llista de filtres possibles és la mateixa que la dels filtres de visualització però en aquest cas serveixen per parar els "flows" que concordin (ja siguin peticions o respostes, segons el filtre indicat), alterar-los si s'escau de la forma que volguem i llavors enviar-les al seu destí. Els "flows" parats es veuen vermells a la pantalla inicial de *mitmproxy.*

*__I__ : Activa/desactiva la intercepció en temps real (pulsant alternativament aquesta tecla)

*__A__ : Allibera tots els "flows" (és a dir, peticions i respostes) interceptats en aquest moment

*__X__ : Elimina tots els "flows" interceptats (per a què no arribin mai al seu destí)

Un cop tinguem algun flow interceptat, podrem sel.leccionar-lo (entrant, com sempre, a la seva pantalla de detalls). En aquesta pantalla podrem fer, amb les següents tecles, diverses accions en aquest "flow" aturat:

__e__ : Permet editar qualsevol aspecte de la petició/resposta actualment en pantalla (capçaleres, contingut del cos, URL sencera, cookies, codi de resposta, mètode, parts concretes del "path"...). Un cop triat l'aspecte en concret, apareixerà l'editor per defecte del sistema per tal de procedir a realitzar (i guardar) les modificacions.

__a__ : "Allibera" el "flow" actualment en pantalla per tal de què arribi al seu destí

__V__ : Desfà els canvis realitzats al "flow" actualment en pantalla

*__o__ : Ordena la llista de "flows" segons un criteri dels que apareixen al quadre emergent: tamany, data, URL o mètode

*__m__ : Des/Marca el "flow" actualment sel.leccionat. Amb "**M**" es mostraran alternativament només els "flows" marcats o tots. Amb "**U**" s'eliminen totes les marques.

*__w__: Guarda la llista sencera de "flows" mostrats a la pantalla inicial en un fitxer (el nom del qual s'ha d'indicar a la comanda que apareix a la barra inferior). Per cancel.lar, "ESC".

*__W__: Guarda la llista sencera de "flows" mostrats a la pantalla inicial en un fitxer (el nom del qual s'ha d'indicar a la comanda que apareix a la barra inferior) en temps real. Per cancel.lar, "ESC".

*__L__ : Carrega un conjunt de "flows" guardat prèviament en un fitxer (el nom del qual s'ha d'indicar a la comanda que apareix a la barra inferior). Per cancel.lar, "ESC".

*__x__ : Guarda la petició seleccionada (en format curl o el que s'esculli) en un fitxer (el nom del qual s'ha d'indicar a la comanda que apareix a la barra inferior). Si es vol guardar el "flow" sencer (és a dir, la parella petició + resposta), la manera més senzilla és entrar a la pantalla de detalls i allà pulsar "**w".** Per cancel.lar, "ESC".

*__b__ : Guarda el cos de la resposta seleccionada en un fitxer (el nom del qual s'ha d'indicar a la comanda que apareix a la barra inferior). Per cancel.lar, "ESC".

*__r__ : Reenvia la petició corresponent al "flow" seleccionat al mateix destí d'aquest "flow"

*__n__ : Crea una nova petició (apareix una comanda a la barra inferior que deixa indicar el mètode i la URL destí). Per cancel.lar, "ESC".

*__D__ : Duplica el "flow" actualment seleccionat

* **|** : Executa un determinat script sobre el "flow" sel.leccionat (apareix una comanda a la barra inferior que deixa indicar el nom de l'script desitjat) Sobre els scripts en parlarem més endavant. Per cancel.lar, "ESC".

***O** : Mostra la llista d'opcions de configuració de Mitmproxy amb el seu valor actual (similar a com ho fa el paràmetre *--options* de *mitmproxy*) però amb la possibilitat de canviar "al vol" qualsevol dels valors allà indicats, reflectint-se a l'instant el canvi indicat. Per poder editar un valor s'ha de pulsar ENTER primer...i per finalitzar-la cal pulsar ENTER un altre cop. Per sortir d'aquesta pantalla, "q".

NOTA: Als exercicis estudiarem amb més profunditat moltes de les comandes subjacents a les tecles llistades als paràgrafs anteriors, a més d'altres comandes que no tenen associada una tecla determinada.

Referència de filtres

Els filtres (de visualització i d'intercepció) més habituals són:

~m "*regex*" : El mètode de la petició concorda amb l'exp. reg. indicada
~d "*regex*" : El domini de la petició concorda amb l'exp. reg. Indicada
~u "*regex*" : La URL de la petició concorda amb l'exp. reg. indicada
~src "*regex*" : La direcció IP d'origen concorda amb l'exp. reg. Indicada
~dst "*regex*" : La direcció IP de destí concorda amb l'exp. reg. Indicada
~hq *"regex"* : Una capçalera de client conté cadena del tipus "nom:valor" concordant amb l'exp reg
~hs "*regex*" : Una capçalera de servidor conté cadena del tipus "nom:valor" concordant amb l'" "
~tq "*regex*" : El valor de la capçalera de client Content-Type concorda amb l'exp. reg. indicada
~ts "*regex*" : El valor de la capçalera de servidor Content-Type concorda amb l'exp. reg. indicada
~bq "*regex*" : El cos de la petició conté cadenes concordants amb l'exp. reg. indicada (útil en les peticions de tipus POST)
~bs "*regex*" : El cos de la resposta conté cadenes concordants amb l'exp. reg. indicada
~c *nº* : El codi de la resposta HTTP coincideix amb l'indicat
~a : La resposta inclou un "asset" (CSS, Javascript, imatges, etc)
~q : Petició que no té resposta associada

NOTA: Altres filtres més genèrics són ~e (mostra només els "flows" amb errors), ~marked (mostra només els "flows" marcats), ~http (mostra només els "flows" HTTP/S), ~dns (mostra només els "flows DNS), ~tcp (mostra només els "flows" TCP genèrics no associats a HTTP), ~udp (mostra només els "flows" UDP genèrics no associats a HTTP) o ~websocket (mostra només els "flows" de tipus websocket). En tot cas, per veure la llista sencera, consulteu https://docs.mitmproxy.org/stable/concepts-filters

Per exemple:

*Peticions a "google.com" : *~d google\.com*
*Peticions que envïin "cookies": *~hq Cookie*
*Peticions de tipus POST: *~m post*
*Respostes contenint imatges: *~ts image/.**
*Respostes el cos de les quals contingui la paraula "test": *~bs test*

Els filtres anteriors es poden combinar mitjançant parèntesis i els operadors **!** (not), **|** (o) i **&** (i, per defecte). Per exemple, l'expressió *!(~tq "text/html") & !(~u "lala")* farà que Mitmproxy mostri tots els "flows" excepte els que contenen peticions amb un "Content-Type" igual a "text/html" i fetes a una URL que tingui la cadena "lala")

Un breu repàs a l'ús d'exp. reg a Python es pot trobar a https://docs.python.org/3/howto/regex.html

Referència de paràmetres

A més dels paràmetres dels executables de la suite Mitmproxy ja vistos en paràgrafs anteriors (concretament, *--listen-host*, *-p*, *--comands*, *--options*, *--showhost* i *-m transparent*, i a més dels particulars *--web-iface* i *--web-port* de *mitmweb*), altres paràmetres comuns són:

-h : Mostra la llista completa de paràmetres
--version : Mostra la versió de Mitmproxy (i les de Python i OpenSSL utilitzades)
-v : Mode verbós (útil sobre tot a *mitmdump*)
-q : Mode silenciós (útil sobre tot a *mitmdump*; només es mostraran els missatges d'error)

-w /ruta/fitxercaptura : Guarda en un fitxer tots els "flows" detectats. Equivalent a la tecla "**W**". Si no es vol sobreescriure el fitxer actual sinó afegir-ne contingut nou, s'ha de precedir la ruta amb un símbol "+", així: *-w +/ruta/fitxercaptura*
-r /ruta/fitxer/capturaAnterior: Llegeix una captura de "flows" prèviament guardada. Se sol usar juntament amb el paràmetre *-n*, que serveix per no obrir el port d'escolta del Mitmproxy. Equivalent a la tecla "**L**".
-C /ruta/fitxer/capturaAnterior: Reenvia les peticions presents en una captura de "flows" prèviament guardada. Se sol usar juntament amb el paràmetre *-n* també

--view-filter "filtre visualització" Estableix un filtre de visualització. Equivalent a la tecla "**f**"
--intercept "filtre intercepció" Estableix un filtre d'intercepció. Equivalent a la tecla "**i**".

-s /ruta/script.py : Executa l'script indicat (de tipus Python però fent servir l'API específica de Mitmproxy) sobre cadascun dels "flows" reconeguts per Mitmproxy. Aquests scripts (per què aquest paràmetre es pot indicar més d'una vegada) poden realitzar qualsevol tasca automatitzada en relació a la manipulació tant de les peticions com de les respostes. Per tant, permeten no haver d'usar Mitmproxy de forma interactiva. Els estudiarem més endavant

-m mode : A més del mode "transparent", Mitmproxy també pot funcionar de forma estàndard (*-m regular*), com a proxy SOCKS (*-m socks5*), proxy invers (*-m reverse:http[s]://x.x.x.x[:nº]*) o proxy "upstream" (*-m upstream:http[s]://x.x.x.x[:nº]*)

--anticache : Elimina les capçaleres de petició que podrien causar una resposta amb codi 304. És a dir, força a què el servidor reenvïi el contingut original encara que estigui catxejat pel navegador.
--anticomp : Intenta convèncer al servidors per a què envïin les respostes sense comprimir. Tant aquesta opció com l'anterior són recomanables a l'hora d'observar un comportament més "net" de les transaccions HTTP

--ignore-host "regex" : Ignora (i, per tant, reenvia sense inspeccionar) tot el tràfic dirigit al/s destí/ns que concordi/n amb l'expressió regular donada (la qual serà ha de ser comparable amb una cadena del tipus "host:nºport", com ara "*^example\.com:443$*"). Aquesta opció pot ser indicada varis cops. Teniu més informació sobre aquesta opció a https://docs.mitmproxy.org/stable/howto-ignoredomains

> NOTA: Altres exemples de possibles expr. reg. a usar amb *--ignore-host* són, per exemple: "*^(.+\.)?apple\.com:443$*" (per ignorar tant "apple.com" com qualsevol dels seus subdominis) o "*^example.com*" (per ignorar "example.com" però no els seus subdominis) . En el cas de fer servir el mode transparent, la documentació oficial recomana usar IPs de destí en comptes de noms; així, ens podríem trobar que per ignorar una determinada IP hauríem d'escriure "*17\.178\.96\.59:443*" i per ignorar un rang, "*17\.178\.\d+\.\d+:443*"

-B "patró" : Reemplaça automàticament un determinat valor textual per un altre en algun lloc dins del cos de les peticions i/o respostes inspeccionades. Aquesta opció pot ser indicada varis cops. El "patró" pot tenir les següents formes:

"*|filter|regex|replacement*" "*|filter|regex|@ruta/fitxer*" "*|regex|replacement*" "*|regex|@ruta/fitxer*"

on "filter" pot ser qualsevol filtre dels ja coneguts per tal d'indicar la característica del/s flux/s on s'aplicarà el reemplaçament (és a dir, en quines peticions, respostes,...és un valor opcional: si no

s'indica el procés es realitzarà a tots les peticions i respostes); "regex" és una expressió regular que defineix els possibles textos que seran reemplaçats en concordar; "replacement" és el text concret que els reemplaçarà. Si el text reemplaçador comença amb un "@", significarà que aquest text és en realitat el contingut del fitxer la ruta del qual s'indica a continuació de l'arroba. El símbol "|" pot ser un altre qualsevol, no cal que sigui "|". Per exemple, l'expressió *-B "|~q|foo|bar"* modifica qualsevol ocurrència de la paraula "foo" per la paraula "bar" en totes les peticions (però no en les respostes).

-H "patró" : Afegeix (si no existeix) o modifica/esborra (si sí existeix) la capçalera indicada, la qual pot ser de client o de resposta. Aquesta opció pot ser indicada varis cops. El "patró" pot tenir les següents formes:

"|filter|header|value" *"|filter|header|@ruta/fitxer"* *"|header|value"* *"|header|@ruta/fitxer"*

on "filter" pot ser qualsevol filtre dels ja coneguts per tal d'indicar la característica del/s flux/s on s'afegirà/modificarà/eliminarà la capçalera; "header" és el nom d'aquesta capçalera en qüestió i "value" el seu valor. Si el valor comença amb un "@", significarà que aquest valor és en realitat el contingut del fitxer la ruta del qual s'indica a continuació de l'arroba. Si no s'indica cap valor (valor buit) s'entendrà que es vol eliminar la capçalera en qüestió. El símbol "|" pot ser un altre qualsevol, no cal que sigui "|". Per exemple, l'expressió *-H "|~q|Host|mitmproxy.org"* modifica la capçalera (de client) "Host" existent a cada petició pel valor indicat: Per afegir, d'altra banda, la capçalera "Host":"mitmproxy.org" però només a les peticions que no en portin cap capçalera "Host" per defecte, es podria fer això altre: *-H "|~q & !~h Host:|Host|mitmproxy.org"*

--map-local "patró" : Defineix quines peticions HTTP seran redireccionades a quins arxius locals: aquests arxius seran enviats transparentment al client en lloc del recurs inicialment demanat. Aquesta opció pot ser indicada varis cops. El "patró" pot tenir les següents formes:

"|filter|regex|/ruta/fitxer" *"|regex|/ruta/fitxer"*

on "filter" pot ser qualsevol filtre dels ja coneguts per tal d'indicar la característica de la petició a la què s'aplicarà la redirecció (és un valor opcional: si no s'indica el procés es realitzarà a tots les peticions); "regex" és una expressió regular que s'avaluarà contra la URL completa de la petició; "/ruta/fitxer" és el fitxer que serà servit al client si l'expressió regular ha concordat. El símbol "|" pot ser un altre qualsevol, no cal que sigui "|".

> NOTA: Es pot indicar la ruta d'una carpeta enlloc de la d'un fitxer. Si és així, totes els arxius indicats al final de la URL original es redireccionaran a un arxiu homònim sota la carpeta en qüestió. Per exemple, fent "|*example.com/static/foo|~/static"* es redireccionarà la petició "https://example.com/static/foo/bar.css" a "~/static/bar.css"

-M "patró" : Defineix quines URLs de les peticions seran modificades per tal de redirigir-les al nou destí: el nou recurs serà enviat transparentment al client en lloc del recurs inicialment demanat. Aquesta opció pot ser indicada varis cops. El "patró" pot tenir les següents formes:

"|filter|regex|replacement" *"|regex|replacement"*

on "filter" pot ser qualsevol filtre dels ja coneguts per tal d'indicar la característica de la petició a la què s'aplicarà la redirecció (és un valor opcional: si no s'indica el procés es realitzarà a tots les peticions); "regex" és una expressió regular que s'avaluarà contra la URL completa de la petició; "replacement" és el text concret que reemplaçarà la part de la URL concordant. El símbol "|" pot ser un altre qualsevol, no cal que sigui "|". Per exemple, per reenviar totes les peticions de "example.org" a "mitmproxy.org" només caldria fer *-M "|~m GET|//example.org|//mitmproxy.org"* . Per reenviar, en canvi, totes les peticions que acabin en ".jpg" a "https://placedog.net/640/480?random " caldria fer això: *-M "|.*\.jpg$|https://placedog.net/640/480?random"*

Referència d'opcions

L'arxiu de configuració de la suite Mitmproxy (compartit pels tres binaris) que determina el seu comportament és un fitxer YAML anomenat "**~/.mitmproxy/config.yaml**". Els valors de les opcions allà guardades es poden visualitzar, tal com ja sabem, amb el paràmetre *--options* La llista completa de totes les opcions existents també la podem consultar a https://docs.mitmproxy.org/stable/concepts-options

Si es vol canviar algun valor concret d'alguna opció només per una determinada execució de Mitmproxy sense haver de modficar el fitxer de configuració es pot fer servir el paràmetre *--set optionName=optionValue* No obstant, les opcions més habituals ja tenen un paràmetre específicament per elles (és el cas, per exemple, entre d'altres, dels paràmetres: *-r, -w, -s, -m, -p, --view-filter, --intercept, --web-iface, --web-port,...*). Moltes d'aquestes opcions també tenen una tecla adient associada a la pantalla interactiva de *mitmproxy* per tal de configurar-se "al vol" (són els casos on es pot veure que la comanda que apareix a la barra inferior en pulsar una determinada tecla és "set", com per exemple "f" o "i", entre altres).

Opcions que no estan en el grup d'"opcions comunes" però que són interessants són, per exemple: **console_palette** (per especificar el tema estètic; els possibles valors apareixen per escollir d'una llista emergent); **console_focus_follow** (valor booleà que per defecte és *false* però que si és *true* provoca el mateix efecte que si s'hagués pulsat la tecla "F" en la pantalla interactiva); **stream_large_bodies** (valor de tipus *nº{k|m|g}* que estableix el tamany del cos de les respostes a partir del qual Mitmproxy el deixarà de "buffejar" (i, per tant, el passarà sense inspeccionar), etc, etc.

> NOTA: By default, Mitmproxy will read an entire request/response, perform any indicated manipulations on it, and then send the message on to the other party. This can be problematic when downloading or uploading large files. When streaming is enabled, message bodies are not buffered on the proxy but instead sent directly to the server/client. HTTP headers are still fully buffered before being sent. Request/response streaming is enabled by specifying a size cutoff in the stream_large_bodies option.

D'altra banda, també sabem que dins de la consola interactiva de *mitmproxy* podem pulsar la tecla "**O**" per obtenir igualment la llista d'opcions amb els seus valors actuals i, a més, tenir la possibilitat d'actualitzar-los "al vol". Si volem que aquestes modificacions es guardin en l'arxiu de configuració per properes execucions, cal pulsar la tecla "**S**" a la mateixa pantalla que llista les opcions. Altres tecles interessants en aquesta pantalla són "**D**" (per reinicialitzar totes les opcions al seu valor per defecte) i "**d**" (per reinicialitzar l'opció sel.leccionada al seu valor per defecte).

Finalment, dir que, amb una mica de desenvolupament en Python, nosaltres mateixos podríem afegir opcions pròpies a Mitmproxy més enllà de les que aquest ofereix de forma estàndard, tal com s'explica aquí: https://docs.mitmproxy.org/stable/addons-options

Scripts: objectes "HTTPx"

Mitmproxy internament reconeix cada "flow" HTTP/S com un objecte Python de tipus "HTTPFlow", (que anomenarem per exemple "*flux*"), del qual deriven altres dos objectes importants (que s'han d'anomenar, ara sí obligatòriament, "*flux.request*" i "*flux.response*") de tipus "HTTPRequest" i "HTTPResponse", respectivament. Les propietats i mètodes genèrics d'un objecte de tipus "HTTPFlow" (com el que anomenem "*flux*") són heretats, per tant, tant pels objectes de tipus "HTTPRequest" (com el que anomenem "*flux.request*") com pels de tipus "HTTPResponse" (com el que anomenem "*flux.response*"), afegint-hi cadascun els seus propis mètodes i propietats.

> NOTA: Els objectes "HTTPFlow" deriven al seu torn del tipus d'objecte "Flow", més genèric (del qual deriven també els objectes "TCPFlow" o "UDPFlow", que no veurem). La seva definició, propietats i mètodes es poden consultar a https://docs.mitmproxy.org/stable/api/mitmproxy/flow.html

Tot seguit es llisten les propietats i mètodes més importants dels objectes de tipus "HTTPRequest" i "HTTPResponse", una informació que ens serà molt valuosa quan desenvolupem scripts Mitmproxy (veure més avall). Cal tenir en compte que el valor de les propietats tant de "*flux.request*" com de "*flux.response*" es poden modificar, aconseguint així que Mitmproxy "falsifiqui" els valors originals de la petició o resposta.

> NOTA: A https://docs.mitmproxy.org/stable/api/mitmproxy/http.html es pot consultar la llista completa

*Propietats *objecte "flux.request":*

flux.request.scheme : Protocol usat a la petició (pot valer la cadena "http://" o "https://")
flux.request.http_version : Versió del protocol de la petició (és una cadena tipus "HTTP/2")
flux.request.method : Mètode de la petició (com ara la cadena "GET", "POST", etc)
flux.request.pretty_host : Nom DNS del servidor al qual s'envia la petició (obtingut del valor de la capçalera "Host" de la petició).
flux.request.port : Port del servidor on s'enviarà la petició
flux.request.path : "Path" de la petició (per exemple: "/index.html").
flux.request.query : Diccionari que conté la "querystring" de la petició (com ara: "*?a=1&b=2*") on les claus son els noms de les variables i els valors els seus valors respectius
flux.request.pretty_url : URL de la petició (formada a partir de les parts "scheme", "pretty_host", "port", "path" i "querystring")
flux.request.headers : Diccionari on les claus son els noms de les capçaleres enviades pel client en la petició i els valors són els seus valors respectius
flux.request.cookies : Diccionari on les claus són els noms de les diferents cookies enviades pel client en la petició i els valors són tuples de dos elements: el valor de la cookie en qüestió i un altre diccionari contenint els noms i -si en tenen, valors- dels seus atributs)
flux.request.content : El cos de la petició -normalment POST- (en hexadecimal)
flux.request.text : El cos de la petició -normalment POST- (és una cadena)

NOTA: Es pot assignar a *flux.request.content* un valor cadena si aquesta s'inclou com a primer paràmetre de la funció *bytes()* de Python (la qual transforma la cadena passada en bytes seguint la codificació indicada com a segon paràmetre, codificació que farem que sigui sempre "UTF-8"), així: *flux.request.content=bytes("cadena","UTF-8")*. No obstant, tenint la possibilitat d'usar *flux.request.text* no caldrà

*Mètode d'un objecte "http.Request" nou:

http.Request.make("metode","URL", "", {"capça1":"valor1","capça2":"valor2"}) : Crea al vol una petició del "no res" amb el mètode indicat com a primer paràmetre, la URL demanada com a segon paràmetre i, opcionalment, el contingut de la petició (si per exemple és de tipus POST) en binari (llegir nota següent) com a tercer paràmetre i les capçaleres de petició com a quart paràmetre.

*Propietats *objecte "flux.response":*

flux.response.http_version : Versió del protocol de resposta (és una cadena tipus "HTTP/2")
flux.response.status_code : Codi de la resposta (200, 404,...)
flux.response.headers : Diccionari on les claus son els noms de les capçaleres enviades pel servidor (case-insensitive) i els valors són els seus valors respectius
flux.response.cookies : Diccionari on les claus són els noms de les diferents cookies enviades pel servidor i els valors són tuples de dos elements: el valor de la cookie en qüestió i un altre diccionari contenint els noms i -si en tenen, valors- dels seus atributs)
flux.response.content : El cos de la resposta (en hexadecimal)
flux.response.text : El cos de la resposta (és una cadena)

*Mètode d'un objecte "http.Response" nou:

http.Response.make(nº,b"cadena", {"capça1":"valor1","capça2":"valor2"}) : Crea al vol una resposta del "no res" amb el codi de resposta indicat com a primer paràmetre, el contingut indicat com a segon paràmetre i, opcionalment, les capçaleres de resposta com a tercer paràmetre.

NOTA: La "b" que apareix just davant de la cadena escrita entre cometes no és cap error: és el símbol utilitzat per Python per indicar que la cadena que el segueix s'ha d'interpretar binàriament i no pas com a text. És equivalent a fer servir la funció *bytes()* amb la codificació ASCII

Scripts: referència d'events HTTP/S (i més)

Mitmproxy porta definit "de fàbrica" l'esquelet de certes funcions que ens són oferides per tal de què nosaltres, com a programadors, omplim el seu cos amb el codi que volguem, codi que s'executarà quan es detecti cert event, també predefinit en Mitmproxy. És a dir, cada event reconegut per Mitmproxy està associat internament i de forma unívoca a un d'aquests esquelets de funcions predefinits, de manera que sempre que Mitmproxy detecti cert event, executarà el codi corresponent a la funció particular que estigui associada, codi omplert per nosaltres.. Moltes d'aquestes funcions tenen com a (únic) paràmetre obligatori un objecte de tipus "HTTPFlow" (que anomenarem "flux"), el qual podrà ser inspeccionat i/o manipulat en el cos de la funció "on the fly".

A continuació es mostra una llista dels esquelets de funcions associades a events més importants relacionats amb el tràfic de tipus HTTP/S. La llista completa (on també apareixen events relacionats amb altres tipus de tràfic, com ara WebSockets, tràfic DNS o "raw" TCP però també amb possibles estats del propi programa Mitmproxy) es pot consultar a https://docs.mitmproxy.org/stable/api/events.html .

def requestheaders(flux): Mitmproxy ha llegit les capçaleres de la petició de "flux" (el cos encara no)
def request(flux): Mitmproxy ha llegit tota la petició pertanyent a "flux"
def responseheaders(flux): Mitmproxy ha llegit les capçaleres de la resposta de "flux" (el cos " ")
def response(flux): Mitmproxy ha llegit tota la resposta pertanyent a "flux"
def error(flux): Mitmproxy ha detectat un error (una resposta invàlida, una connexió interrompuda, un timeout...). Aquí no es contemplen les respostes HTTP vàlides d'error que són simples respostes amb un codi concret (503, 201, etc)

> NOTA: Per poder fer servir qualsevol de les funcions anteriors en un script propi de Mitmproxy (com veurem al proper apartat) és imprescindible que abans de res importem el mòdul "mitmproxy.http", que és on es troben declarades aquestes funcions (i també l'objecte "mitmproxy.http.HTTPFlow" que moltes d'elles prenen com a paràmetre). Això es fa afegint al principi de l'script la línia *from mitmproxy import http*

D'altra banda, altres events més genèrics que també poden ser interessants de saber són, per exemple:

def client_connected(layer): Mitmproxy detecta un nova connexió d'un client. Cal tenir en compte que cada connexió pot correspondre's a múltiples peticions HTTP. El paràmetre "layer" indicat a l'esquelet és de tipus *mitmproxy.proxy.protocol.Layer* (veure nota inferior)
def client_disconnected(layer): Mitmproxy detecta la desconnexió d'un client.

> NOTA: Per poder fer servir qualsevol de les funcions anteriors en un script propi de Mitmproxy (com veurem al proper apartat) és imprescindible que abans de res importem el mòdul "mitmproxy.proxy.protocol", que és on es troben declarades aquestes funcions (i també l'objecte "mitmproxy.proxy.protocol.Layer" que prenen com a paràmetre). Això es fa afegint al principi de l'script la línia *from mitmproxy import proxy.protocol*

def server_connected(conn): Mitmproxy es connecta a un servidor. Cal tenir en compte que cada connexió pot correspondre's a múltiples peticions HTTP. El paràmetre "conn" indicat a l'esquelet és de tipus *mitmproxy.connections.ServerConnection* (veure nota inferior)
def server_disconnected(conn): Mitmproxy es desconnecta d'un servidor.

> NOTA: Per poder fer servir qualsevol de les funcions anteriors en un script propi de Mitmproxy (com veurem al proper apartat) és imprescindible que abans de res importem el mòdul "mitmproxy.connections", que és on es troben declarades aquestes funcions (i també l'objecte "mitmproxy.connections.ServerConnection" que prenen com a paràmetre). Això es fa afegint al principi de l'script la línia *from mitmproxy import connections*

Convé saber, finalment, que Mitproxy pot utilitzar el mòdul oficial "logging", un mòdul que ve incorporat de sèrie amb l'intèrpret Python (el qual, això sí, haurem d'importar prèviament en els nostres scripts amb la línia *import logging* si el volem fer servir) i que conté tot un conjunt de funcions que serveixen per registrar al "log" de l'entorn on s'executa l'script (en aquest cas, l'entorn és el propi Mitmproxy i, per tant, el "log" estarà accessible, recordem, a través de la tecla "**E**" en la pantalla inicial o també directament a través de la sortida estàndard de *mitmdump*) missatges personalitzats per nosaltres, els quals ens podran ser útils a l'hora de depurar els nostres scripts. Concretament, algunes d'aquestes funcions són:

logging.debug("missatge" + algunavariable): Mostra un missatge informatiu de poca prioritat
logging.info("missatge" + algunavariable): Mostra un missatge informatiu de prioritat mitjana
logging.warning("missatge" + algunavariable): Mostra un missatge d'avís
logging.error("missatge" + algunavariable): Mostra un missatge d'error

NOTA: Les comandes de sortida estàndard de Python com *print()* també es poden usar si l'script s'executa amb *mitmdump*

Scripts: creació i exemples

A continuació es presenta un exemple d'script (que anomenarem "elmeuscript.py") que respon a totes les peticions, siguin com siguin, amb una mateixa frase en text pla:

```
import logging
from mitmproxy import http
def response(flux):
    flux.response.status_code=200  #Si no s'indica, es mantindrà el codi de la resposta original
    flux.response.content=bytes("THE POWER BELONGS TO US","UTF-8")
    logging.info("RESPOSTA ENVIADA:"  + flux.request.pretty_url)
```

La forma d'executar aquest script és mitjançant el paràmetre *-s /ruta/elmeuscript.py* de *mitmproxy* o *mitmdump*, per exemple així: **./mitmdump -m transparent -s /ruta/elmeuscript.py**

Un altre exemple d'script (l'anomenarem "**A**") que modifica una mica el comportament de l'anterior és el següent, el qual depenent de la URL demanada pels clients (o més en concret, si es detecta la cadena "elpuig" en qualsevol lloc d'ella... podeu provar tant "https://www.google.com/search?q=elpuig" com "https://elpuig.xeill.net", etc) els hi retorna una resposta o una altra:

```
from mitmproxy import http
def response(flux):
   if "elpuig" in flux.request.pretty_url:
        flux.response.status_code=200
        flux.response.content=bytes("HEY!","UTF-8")
   else:
        flux.response.status_code=404
        flux.response.content=bytes("Personalized 404","UTF-8")
```

Un altre exemple d'script sutilment diferent a l'anterior és el següent (l'anomenarem "**B**"), el qual igualment intercepta les peticions dirigides a una URL particular (en aquest cas és una URL exacta) però ara envia com a resposta un contingut generat al vol pel propi Mitmproxy: la gràcia aquí és que no s'envia cap data al servidor remot i per tant, no és que s'estigui "falsificant" el contingut de la resposta legítima, sinó que directament s'està creant una resposta nova sense que el servidor legítim ni tan sols rebi la petició:

```
import logging
from mitmproxy import http
def request(flux):
   if flux.request.pretty_url == "https://example.com/path":
      flux.response = http.Response.make(200, b"Hello World",
                                         {"Content-Type": "text/html", "Connection":"close"} )
      logging.info(str(flux.response))
```

Un altre exemple d'script és el següent (l'anomenarem "**C**"), el qual afegeix a cada petició una capçalera concreta (de client) amb un valor determinat fixe i afegeix a cada resposta una altra capçalera (de servidor) amb un valor comptador que va augmentant a cada resposta (i mostra el resultat al "log" per estar-ne segurs):

```
import logging
from mitmproxy import http
num=0
def request(flux):
   flux.request.headers["unacapclient"] = "unvalor"
def response(flux):
   global num
   num = num +1
   flux.response.headers["unacapservidor"] = str(num)
   logging.info(flux.request.headers["unacapclient"] + " " + flux.response.headers["unacapservidor"])
```

NOTA: Si la capçalera indicada no existís (ja sigui de client o de servidor), es crearà una de nova; si sí existís, se sobreescriurà el seu valor. Això podria servir, per exemple, per falsificar el valor de l'"user-agent", entre moltes altres possibilitats.

Un altre exemple d'script és el següent, el qual afegeix una parella clau<->valor a la "querystring" de totes les peticions:

```
from mitmproxy import http
def request(flux):
   flux.request.query["Pepe"] = "Es guapo"
```

Un altre exemple d'script és el següent, el qual comprova si es fan peticions que contenen la cadena ".exe" (és a dir, descàrregues d'executables); si és així, falsifica la resposta per tal d'oferir al client un altre ".exe" diferent de l'original (que se suposa que és "malware").

```
import logging
from mitmproxy import http
#La següent línia també es podria haver escrit en dos: with open("/root/ransomware.exe","rb") as fi:
                                                          badExe=fi.read()
badExe=open("/root/ransomware.exe","rb").read()
def response(flux):
   #El condicional es podria haver escrit: if flux.request.path.endswith(".exe"): , on "endswith()" és un mètode Python
   if ".exe" in flux.request.pretty_url:
      logging.info("EXE detected!")
      flux.response.content=badExe
```

Un altre exemple d'script similar a l'anterior és el següent (l'anomenarem "**D**"), el qual, substitueix totes les imatges PNG descarregades de les pàgines obtingudes d'un determinat servidor web ("www.hola.com") per una imatge concreta oferida directament per Mitmproxy (la ruta relativa de la qual és respecte on s'hagi executat la comanda *mitmdump*, compte). A més, força a fer sempre les peticions contra el servidor remot indicat tot i que s'hagi realitzat abans i el seu resultat estigui a la catxé del navegador (per a què l'efecte es produeixi sempre):

```
import logging
from mitmproxy import http
foto=open("foto.jpg","rb").read()
def request(flux):
   if (flux.request.pretty_host == "www.hola.com"):
      flux.request.anticache()
      flux.request.anticomp()
def response(flux):
   if (flux.request.pretty_host == "www.hola.com") and (flux.response.headers["Content-Type"] == "image/jpeg"):
      flux.response.content=foto
      #flux.response.headers["Content-Length"] = str(len(foto))
      logging.info("S'han canviat les fotos de " + flux.request.pretty_url)
```

Un altre exemple d'script que manipula les respostes originals obtingudes de servidors remots és el següent (l'anomenarem "**E**"): aquí el que es fa és alterar el codi font de la pàgina HTML obtinguda per tal d'afegir una línia de codi CSS (que, en aquest cas concret, altera l'orientació vertical de tot el contingut visible de la pàgina). Per això es fa servir el mètode *replace()* propi de Python (més mètodes útils de manipulació de contingut, com el ja vist *endsWith()* i altres, es poden veure , entre altres llocs, a https://www.w3schools.com/python/python_ref_string.asp) i el truc està en substituir un contingut que sabem que sempre hi serà a qualsevol pàgina HTML (l'etiqueta </head>, final del bloc on se sol trobar els diferents codis CSS i Javascript emprats per la pàgina en qüestió), per un contingut que hi afegeix el codi "intrús" que volem:

```
from mitmproxy import http
def response(flux):
    codiIntrus = b"<style>body {transform: scaleX(-1);}</style></head>"
    flux.response.content = flux.response.content.replace(b"</head>", codiIntrus)
```

El següent exemple (que anomenarem "**F**") va més enllà: directament redirecciona les peticions contra un servidor a un altre de diferent ("pretty_host" representa el valor obtingut de la capçalera "Host:" de la petició i "host" representa el nom DNS del servidor on es vol (re)enviar aquesta petició):

```
from mitmproxy import http
def request(flux):
  if flux.request.pretty_host == "example.org":
      flux.request.host = "mitmproxy.org"
```

D'altra banda, en el cas de què la petició interceptada sigui la corresponent a una petició POST feta en un formulari, es podrien afegir més ítems a la petició original amb aquest codi:

```
from mitmproxy import http
def request(flux):
  #Si la petició és de tipus POST "urlencoded" (la típica de formularis)...
  if flux.request.urlencoded_form:
    #...s'afegeix un element més al diccionari
    flux.request.urlencoded_form["nomNouElement"] = "valorNouElement"
  else:  #Si no, igualment es pot enviar al destí noves dades com si fossin provinents d'un formulari hipotètic
    flow.request.urlencoded_form = [ ("element1", "valor1"),  ("element2","valor2")  ]
```

Finalment, el següent codi (que anomenarem "**G**") pot ser molt irritant...¿veieu per què?:

```
import random
import logging
from mitmproxy import http
percentage=25
def request(flux):
  if random.randint(1,100) < percentage:
    logging.warn('Down ' + flux.request.url)
    flux.response = http.Response.make(503, '', {})
```

NOTA: A la carpeta https://github.com/mitmproxy/mitmproxy/tree/main/examples/addons es poden trobar varis exemples oficials d'scripts (així com també a https://docs.mitmproxy.org/stable/addons-examples) i a la carpeta https://github.com/mitmproxy/mitmproxy/tree/main/examples/contrib n'hi han més (de tercers). També es poden trobar d'altres molt pràctics a https://github.com/KevCui/mitm-scripts En tot cas, es recomana molt llegir el seu codi per aprendre'n més possibilitats.

EXERCICIS:

0.-a) Arrenca una màquina virtual Server (Ubuntu o Fedora, és igual) amb la seva tarja en mode "adaptador pont". Aquesta màquina representarà el sistema de l'atacant. Escriu-hi i executa (com a "root"!) el següent script per tal de realitzar l'atac "ARPSpoofing" (mitjançant la comanda *arpspoof* del paquet "dsniff", que hauràs d'haver instal·lat prèviament), activar l'"IP-Forward" i carregar les regles del tallafocs adients indicades a la teoria (i així no caldrà que escriguis cada cop les mateixes comandes). Òbviament, hauràs de canviar la IP indicada en lila per la IP de la teva màquina real (que farà de víctima) i és probable que també hagis de canviar la IP de la porta d'enllaç del Puig per la què correspongui a la teva aula:

```
#!/bin/bash
arpspoof -t 192.168.12.10 192.168.12.123 2> /dev/null &
arpspoof -t 192.168.12.123 192.168.12.10 2> /dev/null &
sysctl net.ipv4.ip_forward=1
nft -f redir80443.txt
```

NOTA: Executant l'script, a més, tindràs l'avantatge que tindràs alliberat el terminal i no hauràs de tenir varis terminals oberts a la vegada cadascun executant una comanda diferent

b) Descarrega el paquet "Mitmproxy" en aquesta màquina virtual, descomprimeix-lo i executa el binari *mitmproxy* en mode transparent, tal com diu la teoria

c) Obre el navegador Firefox de la teva màquina real i vés a http://mitm.it per descarregar el certificat CA de Mitmproxy i integra'l en aquest navegador seguint les instruccions que allà s'hi descriuen (i tal com diu la teoria)

d) Comprova, finalment, que ja pots veure a la consola de Mitmproxy el tràfic HTTP/S generat i rebut per aquest navegador. Confirma (clicant sobre la icona del cadenat que apareix a l'esquerra de la barra de direccions del navegador->fletxeta-> botó "Més informació"->botó "Mostra el certificat") que el certificat obtingut d'allà on hagis anat ve signat per la CA "mitproxy".

e) Prova un filtre. Per exemple, implementa el filtre adient per mostrar a la pantalla del Mitmproxy només les peticions dirigides al domini "xeill.net" i les respostes que no siguin "assets". Un cop comprovat, esborra'l

1.-a) Amb la màquina atacant executant l'atac "ARP Spoofing" i tenint el Mitmproxy obert, obre el navegador a la màquina víctima i vés a https://moodle.elpuig.xeill.net per intentar iniciar sessió amb una contrasenya incorrecta (pots fer servir algun filtre com *~d elpuig.xeill.net & ~m POST* per tenir la pantalla més "neta"). Localitza a la llista de peticions+respostes que apareix a la pantalla de Mitmproxy la petició POST corresponent a l'enviament del nom d'usuari i contrasenya i observa a la pantalla de detalls de la petició quins valors són. ¿Quina és la resposta d'aquesta petició?

NOTA: Desactiva, si la vas activar prèviament, l'opció de "following" del Mitmproxy (tecla "F"); si no ho fas, a la pantalla de detalls aniran apareixent automàticament els dels nous fluxes que vagin apareixent (és a dir, perdràs de vista els detalls del flux inicialment sel·leccionat)

b) Després de pulsar "**z**" per netejar la pantalla del Mitmproxy, inicia sessió a https://moodle.elpuig.xeill.net de nou però ara amb la contrasenya correcta i troba de nou a la llista de peticions+respostes la petició POST corresponent a l'enviament del nou nom d'usuari i contrasenya (observa a la pantalla de detalls de la petició quins valors són). ¿Quina és la resposta d'aquesta petició ara? ¿Com podem saber que l'inici de sessió s'ha realitzat amb èxit?

c) Repeteix el primer apartat però ara loguejant-te a Facebook o qualsevol altre lloc web que autentiqui amb contrasenyes (i canviant el filtre de domini adientment però mantenint el filtre per només veure les peticions de tipus POST). Localitza a la llista de peticions+respostes la petició POST corresponent a l'enviament del nom d'usuari i contrasenya ¿Què veus a la pantalla de detalls de la petició?

NOTA: Hauràs pogut comprobar que, a diferència de Moodle, Facebook no envia la contrasenya "tal qual" al servidor sinó que l'envia "hasheada". Per tant, un cop obtingut el "hash" amb Mitmproxy encara caldria "crackejar-lo" (amb Hashcat o una eina similar). Per guardar el "hash" de Mitmproxy en un fitxer i així poder treballar millor el més fàcil és tornar a la pantalla principal que llista els fluxes i, un cop sel.leccionat la petició concreta on s'envia l'usuari i contrasenya, pulsar "**x**" per tal d'exportar aquesta petició (apareixerà un quadre que preguntarà què es vol exportar: si només la petició, si només la resposta, si totes dues...amb només la petició ja ens serà prou), indicant a la barra de comandes el nom del fitxer on es guardarà allò exportat. En tot cas, caldria saber, no obstant, l'algoritme fets servir per calcular el "hash", el qual, actualment, utilitza una sal derivada del temps i l'algoritme base64 per codificar el resultat; per confirmar-ho, caldria estudiar el codi font del formulari (CTRL+SHIFT+C a Firefox) i veure quines funcions Javascript es criden en pulsar el botó del formulari de login. En qualsevol cas, és una tasca feixuga i existeixen tècniques alternatives (que veurem més endavant) que són més directes per aconseguir obtenir accés no permès a aquest tipus de webs.
NOTA: Si la "víctima" no es logueja explícitament sinó que ja entra en una sessió prèviament guardada mitjançant "cookies", la captura de credencials encara és més difícil ja que s'han emprar mecanismes de "Session hijacking" (també anomenat "HTTP Spoofing"), en els quals intervé la manipulació adient de la "cookie" en qüestió. Aquestes tècniques, de fet, també poden servir per modificar altres dades en les peticions (com ara el preu d'un article comprat en una botiga per un determinat usuari loguejat, etc) però que força més complexos.

2.-a) Tanca l'execució de Mitmproxy i executa al terminal la següent comanda: *export MIMTPROXY_EDITOR=$(which nano)* . Això permetrà al Mitmproxy obrir l'editor *nano* quan haguem de modificar a mà algun contingut pertanyent a algun flux (per defecte s'usa l'editor *vi*). Un cop fet això, torna a executar de nou Mitmproxy.

b) Implementa el filtre adient per mostrar només els fluxes les peticions dels quals vagin dirigides al domini "facebook.com", i, d'altra banda, intercepta totes les respostes provinents d'aquest domini que tinguin un "Content-Type" de "text/html". Tot seguit, obre el navegador de la "víctima" i vés a https://www.facebook.com ¿Què veus a la pàgina mostrada pel navegador (que hauria de mostrar el formulari d'inici de sessió de Facebook)?

c) Tria la petició principal (és a dir, la realitzada a la pàgina inicial del formulari de login de Facebook) i, a la seva pantalla de detalls, pulsa "e"; al quadre que apareix, pulsa "b" (per escollir modificar el cos de la resposta). A l'editor que apareix, busca la frase "Inicia la sessió" mostrada pel botó de "submit" i substitueix-la per qualsevol altra frase. Guarda el canvi i torna a la pantalla principal de fluxes

d) Reestableix tots els fluxes interromputs (pulsant "A"). ¿Què veus ara a la pàgina del navegador que vas sol·licitar a l'apartat b)?

3.-a) Prova l'script anomenat "A" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria. ¿Què passa quan vols navegar a qualsevol pàgina des del navegador de la víctima?

b) Prova l'script anomenat "B" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria però utilitzant com a URL de prova "https://www.mysql.com ¿Què passa llavors si escrius aquesta URL a la barra de direccions del navegador de la víctima per intentar anar-hi?

c) Prova l'script anomenat "C" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria. Obre el navegador de la víctima i pulsa "F12" per a mostrar el quadre "Desenvolupador web"->"Xarxa". Prova de navegar per qualsevol pàgina i fixa't (després de filtrar només el contingut de tipus "HTML", per tenir-ho tot més net) en les capçaleres de servidor de la resposta qualsevol que hagis sel·leccionat en aquest moment: ¿veus la capçalera "unacapservidor"?, ¿què val?. ¿Per què creus, en canvi, que no es veu la capçalera "unacapclient" (però aquesta sí que es veu, en canvi, a la sortida de la comanda *mitmdump*)?

d) Descarrega una foto qualsevol (que anomenarem "foto.jpg") de tamany mitjà (uns 640x480 píxels, per exemple) a la mateixa carpeta on es trobi la comanda *mitmdump* i tot seguit prova l'script anomenat "D" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria (afegint el paràmetre *–anticache* a la comanda *mitmdump*). ¿Què passa quan accedeixes a la web https://www.hola.com des del navegador de la víctima?

e) Prova l'script anomenat "E" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria. ¿Què passa quan vols navegar a qualsevol pàgina des del navegador de la víctima?

eII) Modifica el codi de l'script anterior per a què faci un altre canvi CSS; concretament, per a què canvïi el color del text de totes les pàgines a vermell (el codi CSS necessari per fer això és *body {color:red;}*)

> NOTA: A https://github.com/tlrobinson/evil.css/blob/master/evil.css existeixen molts altres exemples de codis CSS més o menys maliciosos que es podrien incrustar també en les pàgines interceptades
> NOTA: Si volguéssim incrustar algun codi Javascript (en lloc de CSS) el procediment seria similar, tal com mostra la següent plantilla (les tres cometes és una notació de Python per indicar que el que hi ha a dins és text amb salts de línia):
>
> ```
> from mitmproxy import http
> def response(flux):
>   codiIntrus=b"""<script>
>   //Tot el codi Javascript que es vulgui...
>   </script></body>"""
>   flux.response.content = flux.response.content.replace(b"</body>", codiIntrus)
> ```

eIII) En el sentit que indica la NOTA anterior, digues per a què podries fer servir el codi Javascript (més el PHP auxiliar que l'acompanya) mostrat en aquest article: https://wiremask.eu/articles/xss-keylogger-tutorial?

> NOTA: A https://book.hacktricks.xyz/pentesting-web/dangling-markup-html-scriptless-injection teniu més exemples d'injecció de codi HTML/Javascript maliciós. D'altra banda, un exemple concret i real (a més de senzill) d'injecció Javascript maliciosa és el conjunt d'scripts disponibles a https://github.com/arnaucode/coffeeMiner , molt recomanable de consultar

f) Prova l'script anomenat "F" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria. ¿Què passa quan vols navegar a la web "example.org" des del navegador de la víctima?

g) Prova l'script anomenat "G" mostrat i explicat a l'apartat "Scripts: creació i exemples" de la teoria. ¿Què passa quan vols navegar a qualsevol pàgina des del navegador de la víctima varis cops seguits? Pots comprovar què està passant si obres amb "F12" el quadre "Desenvolupador web"->"Xarxa" i observes les respostes rebudes a la petició principal ("/"). ¿Què passa si augmentes el valor de la variable *percentage* de l'script (per exemple, a 75) i tornes a provar-ho?

h) ¿Quina és la intenció del codi https://github.com/mitmproxy/mitmproxy/blob/main/examples/contrib/block_dns_over_https.py (el punt clau és la presència del mètode *flow.kill()*)?

4.-a) Executa *mitmdump -m transparent -M "|//.*\.example.org|//mitmproxy.org" --anticache* i tot seguit obre el navegador de la víctima per anar a https://www.example.org : ¿què veus i per què?

aII) Executa *mitmdump -m transparent -M "|\^.+\.jpg|https://placedog.net/640/480?random" --anticache* i tot seguit obre el navegador de la víctima per anar a un lloc web qualsevol amb imatges : ¿què veus i per què?

aIII) ¿Quina diferència hi ha entre l'opció *-M* (utilitzada als apartats anteriors) i l'opció *–map-local*?

b) Executa *mitmdump -m transparent -B "|~s|BBC|Dog" --anticache* i tot seguit obre el navegador de la víctima per anar a https://www.bbc.com : ¿què veus i per què?

c) Executa *mitmdump -m transparent -H "|~q|user-agent|wget v(1.1.1)" --anticache* i tot seguit obre el navegador de la víctima i vés a qualsevol pàgina web : ¿què veus i per què?

> Pista: Pots comprovar què està passant si obres amb "F12" el quadre "Desenvolupador web"->"Xarxa" i observes les capçaleres de les peticions enviades

d) Executa *mitmdump -m transparent --set block_list="|~d ^.*twitter.com$|500"* i tot seguit obre el navegador de la víctima per anar a https://www.twitter.com : ¿què veus i per què?

5.- Executa *mitmweb -m transparent --web-port 9191 --web-host 0.0.0.0* i tot seguit obre el navegador de la víctima i vés a https://ip.mitm.pro.xy:9191 ¿Què veus? ¿Pots, per exemple, interceptar tràfic? ¿I repetir peticions ja fetes?

NOTA: En pot iniciar *mitmweb* en segon pla com si fos un servei més de l'usuari (opció molt útil, per exemple, en sistemes "headless") seguint els següents passos:

1.-Escriu en un arxiu (que anomenarem "start_mitmweb.sh"), i sota la línia "#!/bin/bash", la comanda anterior (i dona-li permisos d'execució)
2.-Crea l'arxiu "/home/usuari/.config/systemd/user/mitmweb.service" amb el següent contingut:

```
[Unit]
Description=mitmweb service
After=network.target
[Service]
Type=simple
ExecStart=/home/usuari/start_mitmweb.sh
Restart=on-failure
RestartSec=5
[Install]
WantedBy=multi-user.target
```

3.-Habilita el servei per a què s'executi automàticament els propers reinicis executant les següents comandes:
systemctl --user daemon-reload && systemctl --user enable mitmweb

Desfés l'atac "ARPSpoofing", desactiva l'"IP-Forward" i descarrega les regles del tallafocs emprades als exercicis anteriors per tal de què la víctima no perdi la connexió a Internet i pugui continuar navegant sense problemes (ara sí, però, sense cap intermediari). Això ho pots fer d'una sola tacada sense haver d'estar escrivint cada cop les mateixes comandes amb un script com el següent (l'hauràs d'executar com a "root"):

```
#!/bin/bash
killall -s 9 arpspoof
sysctl net.ipv4.ip_forward=0
nft flush ruleset
```