Università degli Studi di Camerino
Scuola di Scienze e Tecnologie
Corso di Laurea in Informatica

Implementazione di un Honeypot e Case Study di un attacco MITM

Laureandi:
Donoval Candolfi
Andrea Perlini

Relatore:
Prof. Fausto Marcantoni

Anno accademico 2019/2020

Abstract

Giorno dopo giorno, le informazioni memorizzate nel cloud, riguardano sempre di più noi stessi, sia della sfera lavorativa sia della sfera personale.

Questi dati possono avere un valore affettivo o economico, ma comunque sia la loro perdita ci procurerebbe un danno. Perciò bisogna fare affidamento nell'evoluzione dei protocolli di sicurezza.

Prima di affidare la mia privacy o il mio business ad un sistema, è necessario dimostrarne la sua affidabilità.

Per avere una visione più ampia sulle garanzie di esso, si può utilizzare la tecnica di attacco del Man in The Middle, testandolo in un ambiente Honeypot.

Grazie a questo scenario si può imparare ad adottare le dovute contromisure nel caso di un attacco reale, cosi la mia infrastruttura sarà in grado di poterlo mitigare preventivamente senza causare incidenti di percorso.

Indice

Introduzione

Obiettivi della tesi

La tesi è redatta allo scopo di testare le funzionalità di monitoraggio degli attacchi grazie ad un Honeypot e successivamente lo abbiamo testato in un ambiente realistico adatto a ricevere un attacco da parte di terzi. Un tipico attacco che si può ricevere è conosciuto con il nome di MITM ovvero Man In The Middle.

Struttura della tesi

Il seguente documento è suddiviso in 8 capitoli:

- il primo capitolo tratta del concetto di Honeypot e di un software che implementa questa tecnica chiamato Chameleon [1]

- il secondo capitolo ci introduce un particolare tipo di attacco denominato Man In The Middle ed i software utilizzati per dimostrarlo

- all'interno del terzo capitolo enunciamo le differenze tra i due argomenti introdotti precedentemente per poi spiegarne l'utilità nel testare una rete locale

- nel quarto capitolo si introducono gli altri strumenti utilizzati per simulare e verificare dei casi reali di attacco

- il quinto capitolo descrive l'ambiente di test virtuale, ovvero GNS3 [12], dentro il quale saranno condotte le varie simulazioni

- riguardo il sesto si scelgono alcuni protocolli di rete che saranno poi sottoposti ad una serie di attacchi o eventualmente anche ad intercettazione

- mentre nel settimo capitolo analizzeremo il risultato dei test condotti nel precedente capitolo

- per finire tireremo delle conclusioni circa il raggiungimento degli obbiettivi preposti per la tesi

1. Honeypot

Il capitolo è suddiviso in due sezioni:
- presentazione del concetto teorico di Honeypot e qualche esempio di caso d'uso
- introduzione di Chameleon [1], software di diversi componenti, che implementando il concetto della sezione precedente ci permetterà di fare qualche esperimento.

1.1 Definizione di Honeypot

Un Honeypot, detto anche "barattolo di miele", è un componente hardware o software usato come esca ai fini di protezione contro attacchi di vario genere all'infrastruttura di rete aziendale o di pubblica utilità.
Il termine deriva dal personaggio dei cartoni Winnie the Pooh.
Consiste in una risorsa che sembra essere parte della rete e contenere informazioni rilevanti ma che in realtà è ben isolato e non ha contenuti sensibili.

Se l'Honeypot è ben realizzato, l'intruso non si rende conto di essere osservato nella sua attività. Una delle peculiarità del medesimo è la flessibilità nelle possibili configurazioni, infatti tra gli scopi dell'Honeypot ci sono:
prevenzione e riconoscimento degli attacchi, risposta agli attacchi, ricerca e individuazione di nuovi tipi di attacco collezionando nuove minacce, osservazione dell'attaccante e colpirlo.

In base al suo scopo e al suo utilizzo può essere classificato diversamente in Honeypot di produzione e Honeypot di ricerca.

I primi sono i più facili da utilizzare, immagazzinano un numero limitato di informazioni e sono utilizzati principalmente in ambito aziendale. Sono situati all'interno della rete aziendale assieme agli altri server di produzione, in modo tale da migliorare la sicurezza dell'azienda.

I secondi nascono per raccogliere in dettaglio le informazioni riguardanti le metodologie di attacco e le motivazioni che spingono la community di Black Hat ad eseguire attacchi informatici.

Esistono anche appositi servizi come quelli offerti da Symantec, kfsensor, honeynets, Honeytrap, che mettono a disposizione software a pagamento o free per questo genere di attività, catturando pacchetti sospetti verso l'esca. Questa attività di monitoraggio ha il vantaggio di richiedere poche risorse, aiuta a controllare i falsi positivi ed è utilizzabile con IPv6.

1.2 Chameleon

Chameleon [1] è un applicativo composto da più istanze Docker [6] di honeypot personalizzabili usato per monitorare il traffico rete, le attività di Bot, e poter registrare le credenziali ricevute da eventuali tentativi di attacchi.

Docker [6] è applicativo open-source scritto in linguaggio di programmazione Go, usato per automatizzare il processo di Deployment di applicazioni all'interno di contenitori software grazie alla virtualizzazione a livello di sistema Linux-based.
Permette l'isolamento delle risorse all'interno di container indipendenti, in grado di coesistere sulla stessa istanza di Linux, in modo da permettere ai processi di lavorare autonomamente sulla stessa macchina fisica.

1.2.1 Chameleon Node

Chameleon Node è un'integrazione del famoso Runtime NodeJS [3].
Esso permette l'avvio della WebUI, cioè il front-end di Grafana [4], scritto in AngularJS

1.2.1.1 NodeJS

Figura 1. NodeJS

NodeJs [3] è un runtime multipiattaforma orientato agli eventi per l'esecuzione di codice JavaScript, costruito sul motore V8 JavaScript.
In origine veniva utilizzato lato client.
Gli script JavaScript, generalmente sono incorporati all'interno dell'HTML di una pagina web, vengono interpretati da un motore di esecuzione incorporato direttamente all'interno di un Browser.
Node.js [3] consente invece di utilizzare Javascript anche per scrivere codice da eseguire lato server, come per esempio la produzione del contenuto delle pagine web dinamiche prima che la pagina venga inviata al Browser.
Node ha un'architettura Object Oriented che rende possibile l'I/O asincrono, cioè consente di eseguire operazioni I/O a elevato utilizzo di risorse senza bloccare il thread principale.
Questo design punta ad ottimizzare il Throghput e la scalabilità nelle applicazioni web con molte operazioni di input/output.

1.2.2 Chameleon Honeypot

Chameleon Honeypot è un'immagine Docker [6] creata appositamente per configurare ed eseguire le varie istanze dei servizi "esca".
Codesti possono essere configurati tramite il file config.json nella root di Chameleon [1].
Al suo interno posso vedere i vari protocolli abilitati, le porte esposte, e le credenziali di default.

1.2.2.1 Protocolli esposti

- DNS
  La porta usata da un'istanza DNS server è di default la 53.
- HTTP
  La porta usata da un'istanza web server è di default la 80.
- HTTPS
  La porta usata da un'istanza web server sicura è di default la 443.
- HTTP Proxy
  La porta usata da un'istanza HTTP Proxy è di default la 8080.
- SSH
  La porta usata da un'istanza SSH server sicura è di default la 8080.
- POP3
  La porta utilizzata è la 110.
- IMAP
  Nel caso in cui il client ritorni online, dato che rimane in ascolto sulla porta 143, mi notifica le nuove mail ricevute.
- SMTP
  Le porte utilizzate sono 25, 465, 587.
- RDP
  La porta usata dal protocollo è la 3389.
- VNC
  La porta usata dal protocollo è la 5900.
- SMB
  La porta utilizzata è la 445.
- SOCKS5
  Porta utilizzata 1080.
- REDIS
  Porta utilizzata 9443
- TELNET
  Porta utilizzata 23
- POSTGRES
  Porta utilizzata 9999
- MYSQL
  Porta utilizzata 3306

1.2.3 PostgresSQL

Figura 3. PostgreSQL

PostgreSQL [5] contiene un'istanza dell'omonimo DBMS utilizzato da Chameleon [1] per memorizzare tutte le informazioni delle connessioni verso l'honeypot. Questo DB è scritto in C e supporta il multipiattaforma garantendogli una certa elasticità per poter funzionare in diversi scenari.
Un altro importante punto a favore consiste nella possibilità di poter definire nuove tipologie di dato durante la creazione di un nuovo schema/tabella rendendolo ideali per la programmazione ad oggetti.

1.2.4 Grafana Web UI

Figura 4. Grafana Web UI

Grafana [4] è un applicazione web usata per la visualizzazione di infografiche e l'analisi interattiva di dati.
I dati possono essere visualizzati su pannelli formanti una dashboard, creandoli attraverso delle query builder interattivi.
Supporta nativamente gran parte delle sorgenti dati conosciute come Elasticsearch, MySQL, InfluxDB, Prometheus, PostgreSQL [5] e Microsoft SQL Server.
È anche possibile inserire componenti aggiuntivi per estendere la compatibilità con altre sorgenti, fino ad arrivare a un centinaio.

Le componenti interne su cui si basa Grafana [4] troviamo Angular, React, JQuery e D3.js. La maggior parte del front-end è scritta in TypeScript e per il back-end in Go.
Era conosciuto precedentemente con il nome di Kibana Dashboard, conferitogli dallo sviluppatore Rashid Khan nel 2013.
È distribuito come software open-source.

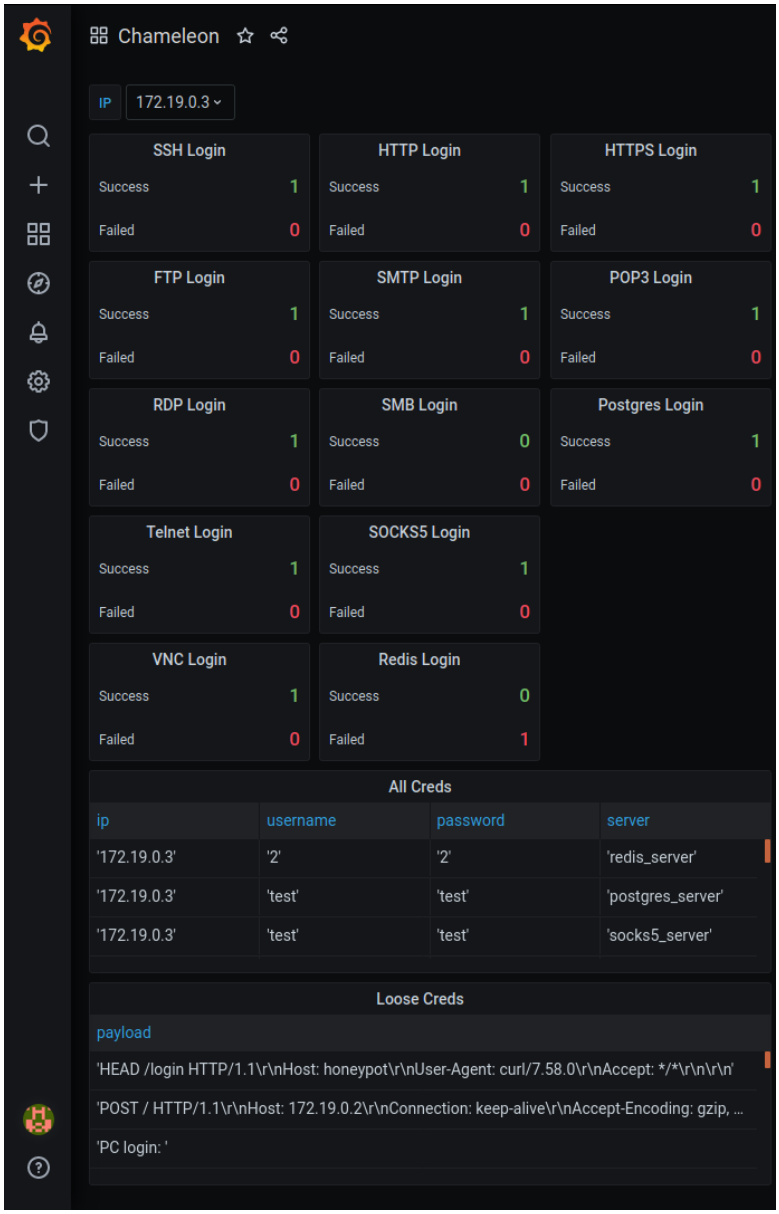


Figura 5. Grafana Web UI

Nella nostra istanza di Chameleon [1], è presente sotto il container Docker [6] chiamato "Chameleon_grafana".
Di default troviamo già preconfigurata la Dashboard per monitorare gli i tentativi di accesso ai vari protocolli con annesse le credenziali e i vari Payload degli stack TCP/IP ed UDP/IP.

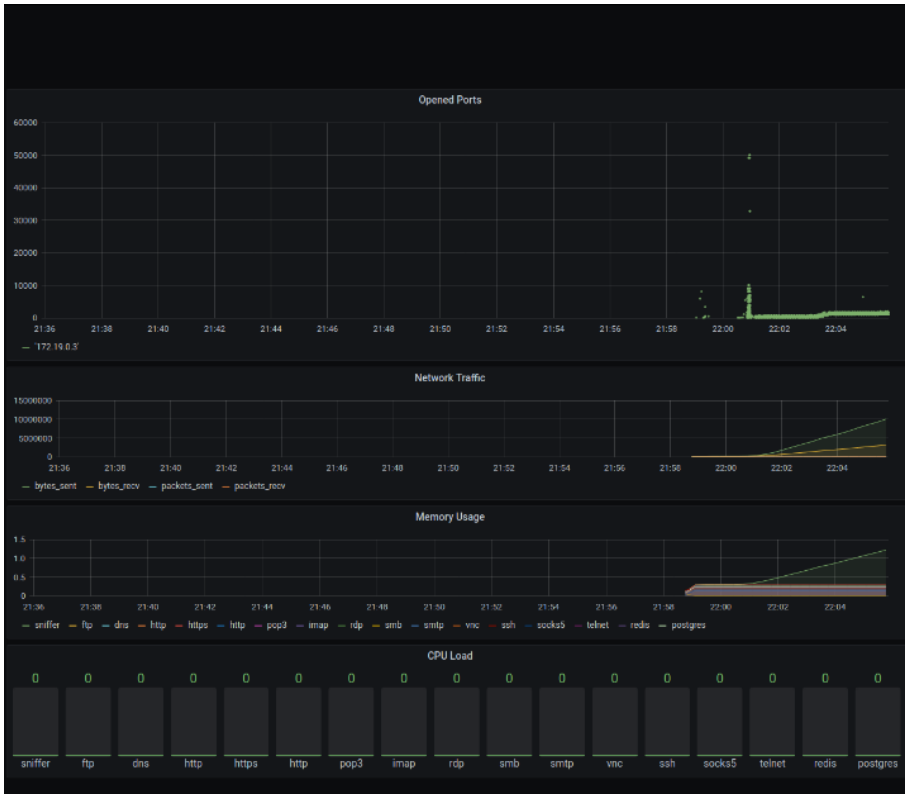


Figura 6. Grafana Web UI

Si può monitorare l'indirizzo da cui sorge un eventuale attacco all'honeypot e la porta corrispondente al servizio.
Inoltre si monitora anche il livello di utilizzo delle risorse attualmente impiegate dalla nostra macchina e il traffico di rete totale generato verso la medesima.

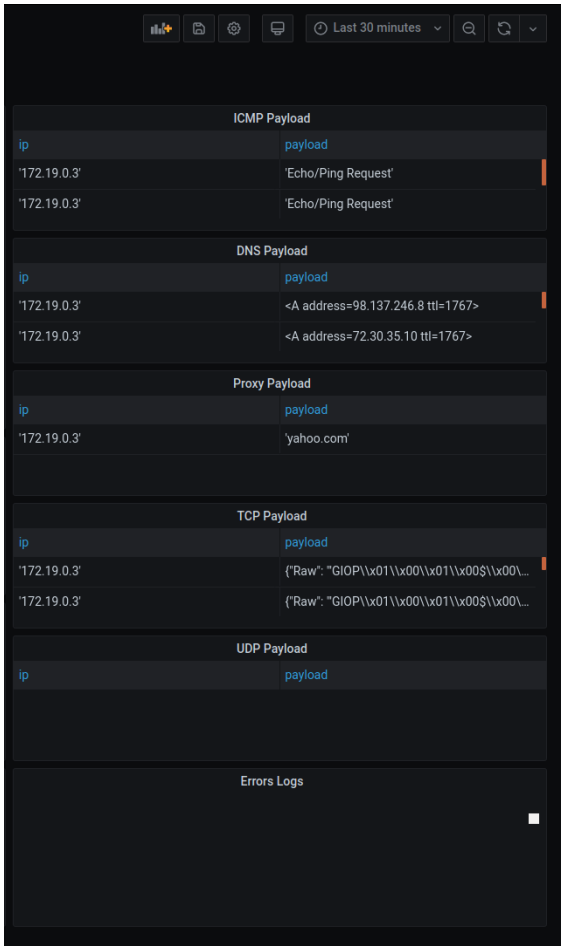


Figura 7. Grafana Web UI

In più si possono osservare i vari Payload ricevuti, corrispondenti allo stack TCP/IP.
Come si può notare in figura, ho la possibilità di visualizzare il Payload in formato Raw, quindi ancora grezzo, in modo tale da controllare un possibile inserimento di un malware al suo interno.

2. Man In The Middle

Il capitolo è suddiviso in tre sezioni:
- definiremo il concetto di Man In The Middle
- si effettuerà lo sniffing dei dati con Bettercap [9]

2.1 Definizione di Man In The Middle

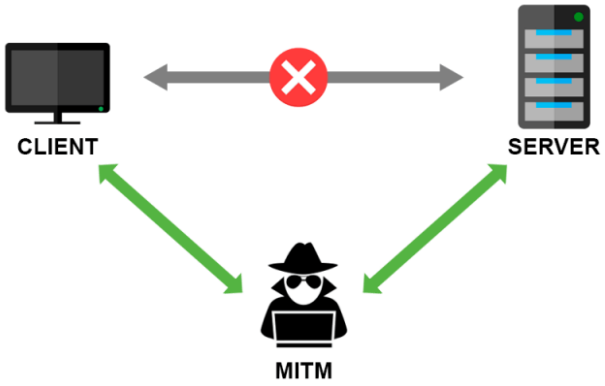


Figura 8. Man In The Middle

In italiano "Uomo di mezzo", è una terminologia usata in sicurezza informatica e nella crittografia per indicare un attacco informatico in cui qualcuno si intromette o altera la comunicazione tra due parti che credono di comunicare tra loro.

Per esempio, un attacco simile può invadere la privacy di una vittima, intercettando traffico proveniente dal client e l'invio al destinatario vittima di pacchetti dati alterati.

Esistono vari tipi di Man In The Middle a seconda dell'utilizzo:

- Man-in-the-wifi: è un modo più antico e comune che consiste nel compromettere una rete Wi-Fi.
Per "mettersi in mezzo" possono creare un nodo di rete Wi-Fi falso detto anche EvilTwin, gemello cattivo, che simula un punto di accesso legittimo per ingannare l'utente. In questo modo possiamo dirottare il traffico della vittima a nostro piacimento, vederne il contenuto e/o catturare cookie di sessione di un sito web.

- Man-in-the-mobile: grazie all'aumentare dei dispositivi mobili come gli smartphone, negli ultimi anni si è visto un incremento di attacchi rivolti ad essi. Uno dei principali motivi è l'intercettazioni o sniffing di SMS riguardanti le One Time Password, cioè codici monouso, che vengono usati come autenticazione a doppio a sistemi bancari o account di siti web.

L'attaccante in questo caso cerca di autenticarsi con il nodo di rete cellulare usando una Sim di un MNO (Mobile Network Operator) e un dispositivo in grado di comunicarci via radiofrequenza.

A questo punto, sfruttando SS7, ovvero il Signaling System 7, riescono ad intromettersi nel sistema di switching delle comunicazioni cellulari.

Nel nostro caso, siamo già inseriti fisicamente in una network cablata e catturiamo il traffico LAN proveniente dal computer vittima preso in esame, e reindirizziamo le richieste che effettua, verso un nostro server locale con Bettercap [9].

2.2 Bettercap [9]

Bettercap [9] è un potente, estendibile e portatile framework scritto in Go. Permette di facilitare il lavoro di ricognizione e attacco da parte dei ricercatori di sicurezza, Red-Team e chi effettua ingegneria inversa.

Permette di interagire con reti IPv4/IPv6, reti Wi-Fi, dispositivi Bluetooth, e dispositivi di input wireless.

Al suo interno troviamo gli Spoofer, cioè i componenti necessari per effettuare attacchi MITM su reti IPv4/IPv6, per i protocolli ARP, DNS [2], DHCP e NDP.

Ma anche istanze applicative di proxy a livello di pacchetto, a livello TCP, e HTTP/HTTPS pienamente personalizzabili con una semplice implementazioni di plugin in JavaScript.

In più è anche un potente sniffer di rete per la raccolta di credenziali che può essere utilizzata anche come Fuzzer del protocollo di rete, integra uno scanner di porte TCP/IP, delle API REST con un supporto asincrono alle notifiche degli eventi, e una interfaccia Web facile da usare.

Esso è distribuito come software open-source, liberamente modificabile, ed è multipiattaforma.

2.2.1 Descrizione dei moduli utilizzati

Bettercap [9] presenta un'architettura a moduli, che ne estendono le funzionalità.

Della parte Core del framework si sono usati i moduli "caplets" e "ui".

- i caplets sono Script appositamente scritti per eseguire comandi di Bettercap [9] in maniera autonoma;

- il modulo "ui" gestisce l'interfaccia grafica via Browser, come per esempio l'utilizzo del caplet "http-ui", oppure il suo aggiornamento tramite "ui.update".

Nella categoria dei moduli IPv4/IPv6 troviamo:

- net.probe che ci permette di effettuare una indicizzazione degli host presenti nella network;

Inoltre nella sotto-categoria degli Spoofers si è usati:

- arp.spoof

L'ARP spoofing è una tecnica di hacking che ci permette di andare a reindirizzare il traffico dati di una vittima. Andando ad alterare una tabella ARP possiamo far confondere l'host di destinazione sostituendo l'indirizzo MAC (Media Access Control). Infatti, chi è vittima di un attacco di questo tipo molto probabilmente si ritroverà uno stesso MAC ripetuto più volte su più dispositivi.

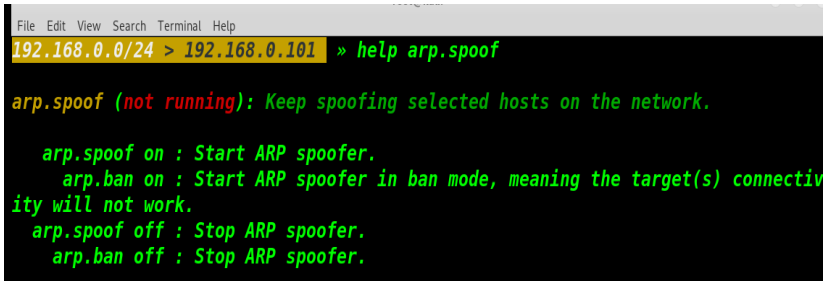


Figura 9. Arp Spoofing Bettercap

Parametri:

- `arp.spoof.fullduplex`

Il parametro fullduplex permette di andare a fare lo spoof sia del target che del gateway.

Di default è disabilitato.

- `arp.spoof.internal`

Ci permette di intercettare anche le connessioni locali oltre a quelle rivolete verso l'esterno

Di default è disabilitato.

- `arp.spoof.targets`

Tramite di esso fornisco la lista delle mie vittime. Posso utilizzare sia l'IP che il MAC.

Di default l'intera sottorete viene selezionata.

- `arp.spoof.whitelist`

Rappresenta lista dei clienti che verranno saltati durante lo spoofing.

Di default la lista è vuota.

- `dns.spoof`

Il DNS spoofing è un'altra tecnica di attacco volta a catturare e conseguentemente alterare una richiesta DNS [2]. Tramite questo modulo posso forzare la mia vittima attraverso un proxy cosi da poterlo indirizzare verso una risorsa malevola facendogli credere di comunicare con quella originale. Verificare questa manomissione è possibile confrontando il risultato delle query tra quelle che si ricevono contattando il proprio default DNS [2] e quelle ricevute da un altro server verificato, inoltre l'IP dell'attaccante è presente nell'intestazione dei pacchetti IP che contengono i pacchetti UDP con le risposte contraffatte.

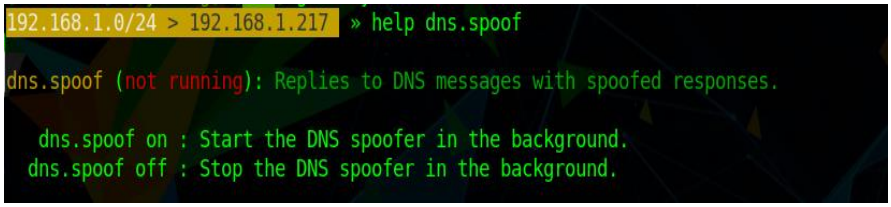


Figura 10. DNS Spoofing Bettercap

Parametri:

- `dns.spoof.address`

Serve per specificare indirizzo al quale verranno mappati i domini.

Di default è impostato sull'indirizzo della macchina in uso dalla quale si sta lanciando l'attacco.

- `dns.spoof.all`

Tramite questo parametro possiamo abilitare la risposta a qualsiasi richiesta altrimenti il programma risponderà solamente a quelle indirizzate al pc locale.

Di default il parametro è disabilitato.

- `dns.spoof.domains`

Mi permette di impostare i nomi dei domini, separati da virgole, sul quale effettuare lo spoof.

Per impostazione predefinita la lista è vuota.

- `dns.spoof.hosts`

Serve per specificare il file dove scrivere la mappatura tra domini e indirizzi.

Il file predefinito è quello degli host della macchina locale.

- `dns.spoof.ttl`

Ci permette di impostare il Time To Live in millisecondi ovvero il tempo in cui il risultato di una query DNS [2].

Di default è di 1 secondo ovvero 1024 millisecondi.

Nella categoria dei Proxies:

- `http.proxy`

Questo modulo è in grado di creare un http proxy trasparente volto allo spoofing dell'intero traffico sul medesimo protocollo.

L'utente vittima non sarà in grado di accorgersi dell'attacco e crederà di comunicare direttamente con il sito richiesto. Avremo la possibilità ad esempio di poter clonare dei cookie di sessione o direttamente di recuperare delle credenziali di accesso.

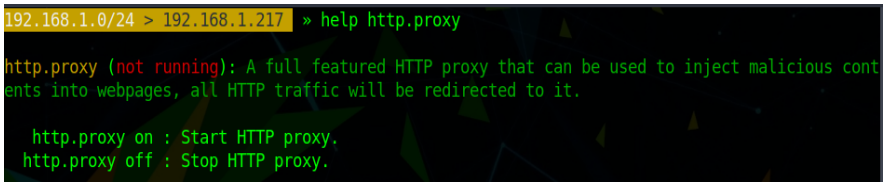


Figura 11. HTTP Proxy Bettercap

Parametri:

- `http.port`

Il seguente parametro permette di specificare la porta sulla quale il traffico verrà reindirizzato una volta che il proxy sarà attivo.

La porta predefinita è la 80.

- `http.proxy.address`

Possiamo specificare l'indirizzo o l'interfaccia sul quale verrà esposto il server.

Di default sarà in ascolto su tutte

- `http.proxy.sslstrip`

Ci consente di abilitare la funzionalità di SSL Stripping ovvero effettuare il downgrade di una sessione protetta in modo da poter "leggere" l'intero traffico.

Se non specificato questo parametro è disabilitato.

- `http.proxy.port`

Specifica la porta sulla quale saremo in ascolto finché il server sarà attivo.

La porta di default è la 8080.

- `http.proxy.script`

Questo parametro specifica il percorso di uno script da applicare nel modulo del proxy.

Di default il percorso è vuoto.

- `http.proxy.injectjs`

Tramite questa opzione possiamo andare ad iniettare, per ogni pagina visualizzata, un java script a nostro piacimento.

Se non impostato il percorso è vuoto.

- `http.proxy.blacklist`

Possiamo creare una lista degli host che verranno scartati durante una sessione di proxying. C'è anche il supporto alle wildcard.

Di default la lista è vuota.

- `http.proxy.whitelist`

Funziona esattamente all'inverso della blacklist sopra indicata.

Anche qua se il parametro non è impostato la lista è vuota.

- `https.proxy`

Modulo con lo stesso funzionamento del precedente ma si applica al protocollo HTTPS. Ovviamente visto la sessione criptata da un certificato SSL sarà necessario configurare ulteriori parametri in modo da far credere alla vittima di essere in una sessione sicura.

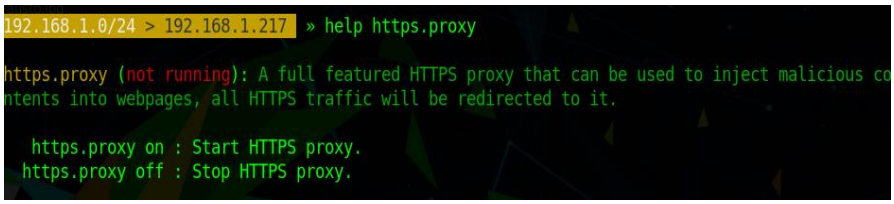


Figura 12. HTTPS Proxy Bettercap

Parametri aggiuntivi:

- `https.proxy.certificate`

Possiamo specificare il percorso di un Certification Authority con estensione “cert.pem”.

Di default ne verrà utilizzato uno autogenerato cioè “~/.bettercap-ca.cert.pem”.

- `https.proxy.key`

Serve ad indicare la chiave associata al certificato sopra fornito e ha esenzione “key.pem”

Se non indicato ne viene autogenerata una “~/.bettercap-ca.key.pem”.

- `https.proxy.certificate.bits`

Ci permette di indicare il numero di bit per la chiave privata riguardante il certification authority.

Il numero di bit impostato di default se non specificato è di 4096.

- Riguardo la configurazione delle proprietà aggiuntive del certificato possiamo utilizzare anche i seguenti parametri:

- `https.proxy.certificate.commonname`
- `https.proxy.certificate.country`
- `https.proxy.certificate.locality`
- `https.proxy.certificate.organization`
- `https.proxy.certificate.organizationalunit`

Nella categoria dei Servers:

- `http.server`

È incluso in Bettercap [9] anche un server http nel quale ospitare file, script o anche semplici pagine web fruibili da tutta la rete.

Il caso d'uso tipico è ospitare uno script JS, contenete codice malevolo, da poter iniettare poi alle vittime tramite il modulo dell'http proxy precedentemente descritto.

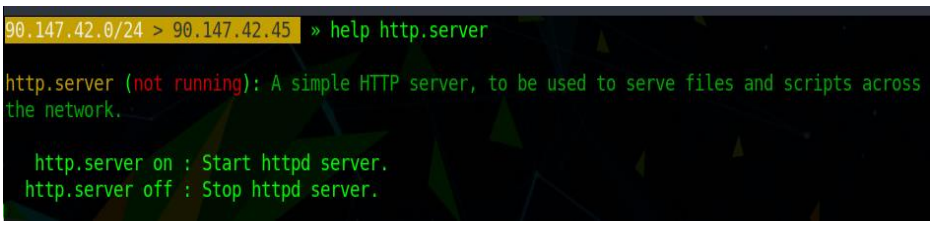


Figura 13. HTTP Server Bettercap

Parametri:

- `http.server.path`

  Possiamo la cartella di destinazione per i file da esporre sulla rete.

  La posizione predefinita è l'attuale cartella di lavoro.

- `http.server.address`

  Indico l'interfaccia o l'indirizzo di ascolto sul quale sarà attivo il server.

  Di default sono tutti gli IP della macchina.

- `http.server.port`

  Specifico la porta sul quale sarà in ascolto il server.

  Nel caso in cui non specifico questo parametro la porta sarà la 80.

- `https.server`

  Simile al modulo precedentemente decritto ma ci permette di ricevere traffico in HTTPS e relativo supporto ai certificati.

  Abbinato tipicamente all' HTTPS proxy ed espone la porta 443.

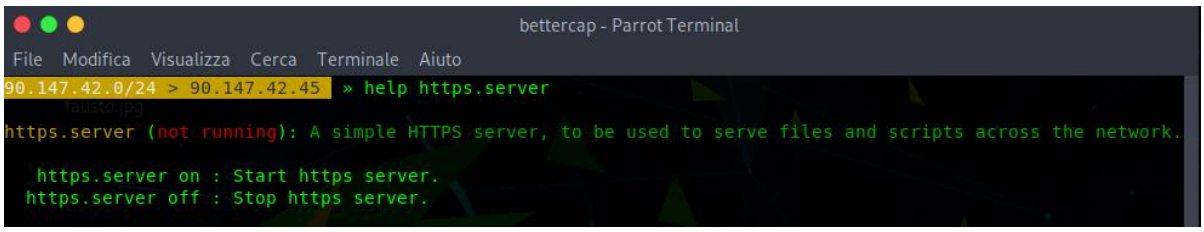


Figura 14. HTTPS Server Bettercap

Parametri in aggiunta rispetto a quelli esposti nel modulo precedente:

- `https.server.certificate`

Indico il percorso del certificate authority che andrà ad usare il web server.

Nel caso in cui non sia stato specificato un percorso ne verrà autogenerato uno.

- `https.server.key`

Fornisco la chiave relativa del certificato.

Se non indicata viene autogenerata.

- `https.server.certificate.bits`

Indico in bit della chiave privata relativa al certification authority.

Valore predefinito è di 4096 bit.

Parametri per le proprietà aggiuntive riguardo il certificato:

- `https.server.certificate.commonname`
- `https.server.certificate.country`
- `https.server.certificate.locality`
- `https.server.certificate.organization`
- `https.server.certificate.organizationalunit`

3. Differenze tra i due sistemi

Precedentemente abbiamo introdotto due concetti ovvero Honeypot e MITM che sotto certi aspetti potrebbero anche somigliarsi ma hanno profonde differenze. Basta pensare al loro fine:

- il primo ha uno scopo preventivo e serve a monitorare la rete per riuscire a scoprire se è presente qualche agente malevolo il quale sta tentando di forzare risorse della propria infrastruttura.

- il secondo permette di intercettare traffico sia per uno scopo di attacco sia per scopo puramente difensivo ovvero riuscire ad indentificare ed escludere un possibile agente malevolo all'interno della rete.

3.1 Perché utilizzare Man In The Middle

I fini di un attacco Man In The Middle possono essere svariati e tato vale per i dati sensibili che possiamo estrapolare. Di seguito andremo ad analizzare 2 possibili casistiche in cui questa particolare tecnica ci viene in aiuto:

- **Clonare la sessione tramite acquisizione dei cookie**

Durante l'utilizzo di un portale web ad accesso ristretto verremo identificati tramite dei cookie che vengono memorizzati nel nostro browser.

Tra di essi sarà presente un token in grado di poter far riconoscere al server la nostra identità ed ovviamente questo identificativo sarà valido per un determinato periodo di tempo.

Figura 15. Header HTML Message Body

Ogni volata che effettuiamo una richiesta al portale il browser scambia questo token senza alcun tipo di offuscamento ed è qui che si può andare ad "infilarsi nel mezzo". Effettuando lo sniffing del traffico in uscita dalla vittima possiamo ricavare l'header delle richieste in http nel quale è presente il cookie interessato.

Una volta estratto il token, l'attaccante, potrà poi andare ad inviare nuove richieste al portale usando l'identità clonata dalla vittima.

Ovviamente scaduto il tempo di validità della nostra chiave di autorizzazione dovremo rieffettuare l'attacco.

- **Alterare richieste/risposte tra la vittima e il server**

Accade sempre più spesso di utilizzare piattaforme di messaggistica online sia per lavoro che per interessi personali.

Immaginiamo di voler andare a "mettere le mani" sul contenuto di un messaggio. Acquisendo i pacchetti in entrata e in uscita potremo leggere il contenuto del messaggio appena inviato o ricevuto, salvo eventuali sistemi di cifratura del testo.

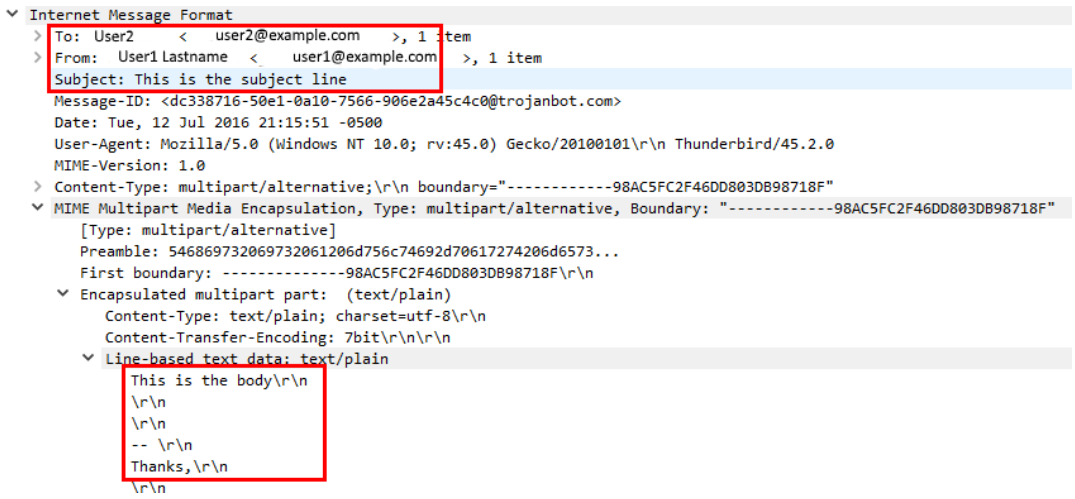


Figura 16. Frame TCP Wireshark

Tramite il web proxy, nel quale forziamo il passaggio dei dati della vittima, possiamo cambiare il contenuto del body del pacchetto e di conseguenza il copro del testo.

Così facendo potremo far perdere un importante appuntamento di lavoro semplicemente alterando l'orario che il collega della vittima ha scritto via messaggio.

4. Altri strumenti utilizzati

In questo capitolo introdurremo gli altri strumenti che verranno utilizzati durante i test riguardanti:

- Tentativi di attacco verso l'honeypot
- Verificare e prevenire un'intrusione MITM

4.1 Testing Honeypot

Per poter testare questo ambiente utilizziamo attacchi di tipo brute-force verso i vari protocolli che espone Chameleon [1].

Per il brute-force utilizzeremo dei dizionari ovvero una raccolta di termini comuni che riguardanti diverse tematiche come:

- Nomi di sport e relativa terminologia
- Personaggi pubblici
- Videogiochi
- Località geografiche note
- Date di eventi storici importanti
- …

Lo strumento che introdurremo successivamente utilizzerà tutti i termini presenti in questi file per ottenere l'accesso ad un sistema. Gli svantaggi maggiori di questo tipo di attacco sono:

- Impiego risorse

L'invio massivo di tentativi di accesso impegnano completamente o quasi completamente le risorse dell'host nel quale è in esecuzione.

- Tempo

Tentando tutti i termini presenti in maniera "stupida" occupiamo quantità di tempo importanti soprattutto in presenza di dizionari "corposi" e per questo spesso si è scoraggiati dall'uso di questo tipo di attacco.

Un dizionario può essere composto da più raccolte unite tra di loro oppure si può autogenerare tramite determinati framework basati sull'intelligenza artificiale.

4.1.1 THC-Hydra [8]

Trattiamo di uno strumento di attacco ottimizzato per il multithreading ovvero capace di effettuare più tentativi di accesso allo stesso tempo. Inoltre risulta molto flessibile l'aggiunta di moduli aggiuntivi.

Viene spesso utilizzato da ricercatori e consulenti di networking security per verificare quanto sia complicato forzare un l'accesso ad un sistema.

Protocolli supportati nativamente (in ordine alfabetico):

- Cisco AAA
- Cisco auth
- Cisco enable
- CVS
- FTP
- HTTP(S)-FORM-GET
- HTTP(S)-FORM-POST
- HTTP(S)-GET
- HTTP(S)-HEAD
- HTTP-Proxy
- ICQ
- IMAP
- IRC
- LDAP
- MS-SQL
- MySQL
- NNTP
- Oracle Listener
- Oracle SID
- PC-Anywhere
- PC-NFS
- POP3
- PostgreSQL [5]
- RDP
- Rexec
- Rlogin
- Rsh
- SIP
- SMB(NT)
- SMTP
- SNMP v1+v2+v3
- SOCKS5
- SSH (v1 and v2)
- SSHKEY
- Subversion
- Teamspeak (TS2)
- Telnet
- VMware-Auth
- VNC
- XMPP

Possiamo utilizzarlo in due modalità:

- Tool di attacco
- Tool di verifica password

Nel nostro caso sfrutteremo solo la prima.

Parametri per configurare ed avviare il brute-force:

- `-R`

Ripristina la sessione precedente magari dopo un eventuale crash dell'applicativo.

- `-S`

Effettua una connessione SSL.

- `-s PORT`

Specifica la porta nel caso in cui essa sia differente dalla predefinita del protocollo.

- `-l LOGIN / -L FILE`

In caso di carattere minuscolo specifico il nome utente con il quale effettuare il login mentre se maiuscolo specifico il file contenete la lista dei nomi da tentare.

- `-p PASS/ -P FILE`

Con il carattere minuscolo tento una sola password mentre nell'altro caso passo il percorso di un dizionario.

- `server`

Specifico il target sul qualche verrà effettuato l'attacco brute-force

- `service`

Indico il servizio che sarà oggetto dell'attacco.

4.2 Verifica e prevenzione attacco MITM

Prima di correre ai ripari per la perdita di informazioni personali o per la compromissione di risorse critiche della propria infrastruttura, risulta molto conveniente mettere in piedi determinati meccanismi, onde evitare brutte sorprese.

Ogni connessione tra client e server dovrebbe essere messa al sicuro con la criptazione del contenuto, in modo da rendere illeggibile il medesimo nell'immediato e rendere impossibile la manomissione.

Di certo determinate informazioni saranno sempre leggibili, come la macchina che sto contattando o il protocollo che sto utilizzando.

Capire se siamo sotto attacco è possibile tramite determinati strumenti integrati nel sistema, che ci illustrano il percorso dei nostri pacchetti all'interno della rete.

4.2.1 Net-tools [10] (unix)

Il pacchetto net-tools [10] disponibile per sistemi Unix-like comprende una suite di strumenti volti a monitorare la nostra configurazione. È pre-integrato nelle distribuzioni basate su Debian.

Di seguito un elenco con i Tool usufruiti, utili per le nostre verifiche:

- arp

Viene usato per manipolare la cache della tabella ARP nel kernel.

La tabella ARP contiene le associazioni tra gli indirizzi e i relativi indirizzi MAC di appartenenza.

Possiamo inoltre aggiungere o rimuovere singoli record dalla tabella oppure effettuare il dump cioè svuotare l'intera tabella.

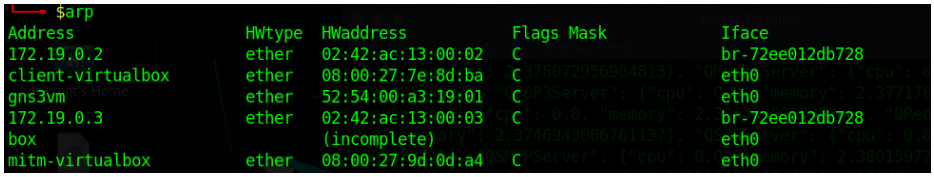


Figura 17. Arp

- ifconfig

Ci consente di visualizzare tutte le informazioni relativi agli adattatori di rete fisici e virtuali. Possiamo trovare informazioni relative sia all'indirizzo IP sia al MAC, la maschera di sottorete, il nostro default gateway, i nostri DNS server, la quantità di pacchetti scambiati e i relativi errori.

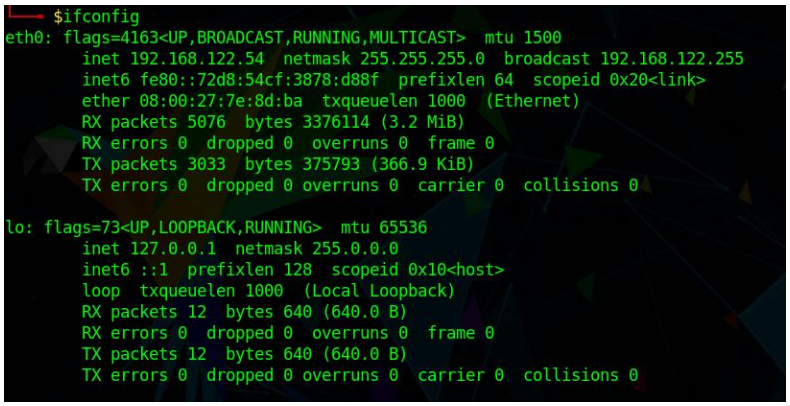


Figura 18. ifconfig

- netstat

Vediamo tutte le connessioni remote o locali instaurate dalla nostra macchina. Abbiamo a nostra disposizione:

- il tipo di pacchetto (TCP o UDP)
- numero dei pacchetti ricevuti
- numero dei pacchetti inviati
- l'indirizzo locale da relativo alla connessione
- l'indirizzo remoto relativo alla connessione
- lo stato della connessione

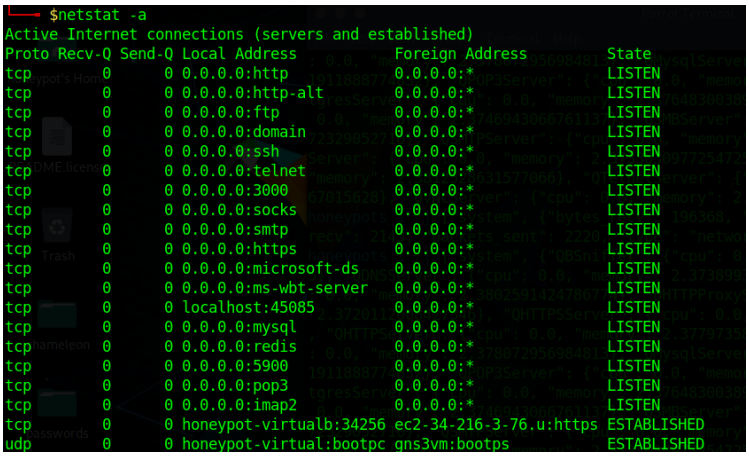


Figura 19. Netstat

5. Configurazione dell'ambiente di test

Nel seguente capitolo andremo ad illustrare come abbiamo organizzato il nostro ambiente di testing con il relativo strumento di simulazione.
Tratteremo inoltre della configurazione che è stata realizzata per poter simulare un'infrastruttura aziendale.

5.1 Predisposizione di un ambiente virtuale

La scelta è ricaduta su di un ambiente virtuale non solo per un fatto di praticità, ma per poter disporre di accessori non facilmente reperibili.

Ovviamente questo ha anche svariati svantaggi come quello di dover disporre di un host in grado di emulare dispositivi multipli come switch, router e pc.

Come tutti i dispositivi anche i vari software trattati finora saranno tutti presenti all'interno di questo ambiente.

5.1.2 GNS3 [12]

GNS3 [12] ovvero Graphic Network Simulator è un simulatore di rete virtuale che consente la comunicazione tra dispositivi reali e virtuali.

Ad oggi, 13 anni dopo la sua prima versione, è comunemente utilizzato negli esami di certificazioni professionali e per testare la struttura di reti molto complesse.

Le componenti chiavi di questo software sono:

- GNS3 UI
- GNS3 Server
- GNS3 VM

GNS3 UI

Rappresenta l'ambiente nel quale andiamo a "disegnare" la nostra rete virtuale.

Possiamo aggiungere vari tipi di oggetti come router, switch, dispositivi "finali", dispositivi di sicurezza ed infine i cavi ovvero oggetti virtuali che collegano tra di loro i vari dispositivi.

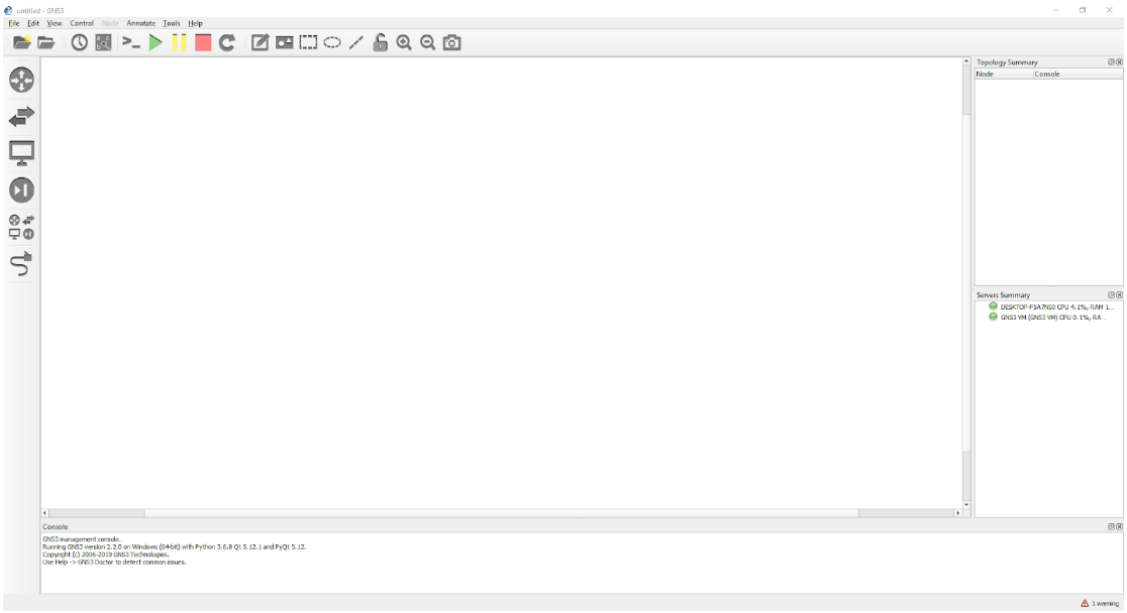


Figura 20. GNS3 UI

Per ogni oggetto si può scegliere la periferica di collegamento e monitorarne lo stato.

Oltre ai device già precaricati, questo software ci dà la possibilità di aggiungere nuovi template, ovviamente fornendone un'immagine detta Appliance.

Un Appliance può essere un Virtual PC, in grado di emulare un computer a riga di terminale, oppure inserire un Browser come Firefox, o un DHCP Server.

L'inserimento dei dispositivi è possibile tramite un Drag and Drop, che ugualmente vale per il loro collegamento.

Per avviare tutto il nostro schema virtuale utilizziamo il bottone Play che esegue il boot dei nostri device, così poi da poterli configurare ed eseguire vari test.

Potremo anche analizzare il traffico posizionandoci sopra un collegamento ed ispezionarlo pacchetto per pacchetto.

GNS3 server

Istanza del server di GNS3 [12] che può essere ospitata in locale nella propria macchina e fa da engine per il modulo della UI.

Si prende carico di istanziare la rete virtuale ed effettuare i vari collegamenti tra i vari device e al suo interno contiene anche diversi emulatori come:

- Dynamips

In grado di emulare vecchio hardware Cisco come il 3725 anche se potremo andare a caricare anche nuove versioni del sistema operativo Cisco ovvero IOS.

- Qemu

Utilizzato anche lui per virtualizzare immagini Cisco ma anche per altri tipi di device di rete come un router con firmware OpenWrt.

Riguardo la virtualizzazione dei dispositivi “finali” in questo caso GNS3 [12] si appoggia ad Hypervisor esterni:

- VMware [14]
- VirtualBox [13]
- Hyper-V

Successivamente possiamo importare le vari macchine e gestirne la configurazione, graficamente all’interno della UI.

Altri moduli aggiuntivi di virtualizzazione:

- VPCS

Semplice virtualizzatore delle funzionalità base di un pc usato tipicamente per testare la connettività.

Docker [6]

Lo possiamo utilizzare per virtualizzare singoli applicativi da collegare direttamente alla nostra rete. Ad esempio, possiamo inserire un container per un server FTP senza dover creare una macchina virtuale dedicata consumando molte meno risorse.

GNS3 VM

La macchina virtuale di GNS3 [12], basata su sistema Linux, può essere eseguita in un Hypervisor locale o su di un server remoto.

Consiste in un contenitore in grado di eseguire al suo interno tutti i vari device della rete virtuale basati sugli emulatori interni. È possibile agganciarsi direttamente dalla UI selezionando un server remoto.

In questo scenario si deve disporre sempre di VMware [14] o di VirtualBox [13] necessari per la virtualizzazione dei dispositivi “finali”.

Per il suo corretto funzionamento una volta avviata dobbiamo tassativamente configurare le interfacce affinché le network virtuali possano comunicare con l’esterno.

5.2 Simulazione di un'architettura di rete

Per la nostra simulazione abbiamo ricreato una rete basica che potesse assomigliare ad un contesto aziendale. La rete, contenuta completamente all'interno di GNS3 [12], avrà al suo interno un dispositivo chiamato NAT cioè una sorta di default gateway dietro al quale saranno mascherati i vari device virtuali ed ha 2 interfacce di rete. La prima ha un IP pubblico, mentre alla seconda è assegnato un IP privato 192.168.122.1.

Ovviamente tutti saranno collegati tramite uno switch unmanaged da 8 porte.

Si è scelto di usare come sistema operativo ParrotOs [7], basato su Debian, per la sua immediatezza d'uso, la sua completezza di strumenti preintegrati, e la sua compatibilità con qualunque macchina lo si usi.

Di seguito un elenco dei dispositivi "finali", virtualizzati tramite VMware [14], presenti all'interno della rete:

- parrot-mitm

Si tratta di una macchina Linux con distribuzione Parrot [7] in grado con al suo interno installato Bettercap [9], strumento necessario per il Man In The Middle.
Il suo indirizzo IP è 192.168.122.245

- parrot-honeypot

Troviamo sempre lo stesso sistema operativo ma abbiamo installato un'istanza di Chameleon Honeypot ed i relativi contenitori di Docker [6].
Il suo indirizzo IP è 192.168.122.131

- parrot-client

Una macchina sempre basata su Parrot [7] sulla quale verrà effettuato l'attacco Man In The Middle ed è anche la sorgente del brute-force.
Il suo indirizzo IP è 192.168.122.54

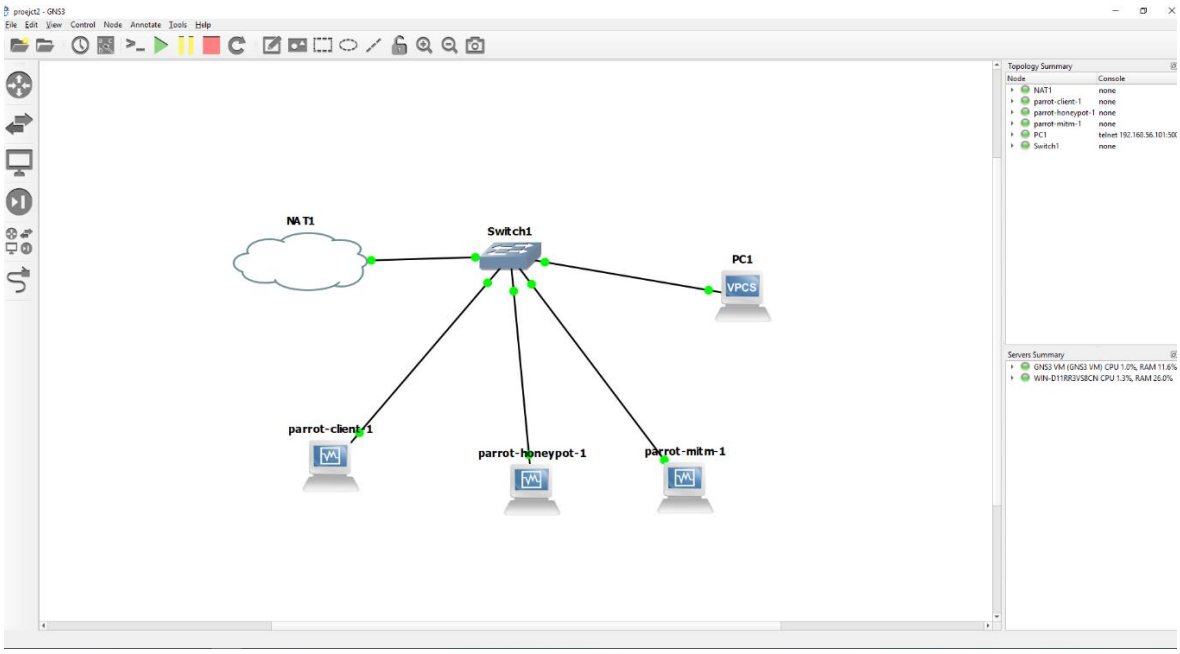


Figura 21. Schema rete virtuale

6. Svolgimento del test

In questo nuovo capitolo tratteremo lo svolgimento pratico dei test sull'honeypot e il case study degli attacchi Man In The Middle.

Prima di tutto però introdurremo quali protocolli saranno presi in analisi durante i test.

6.1 Scelta protocolli da analizzare

Prima dello svolgimento effettivo del test abbiamo dovuto scegliere quali protocolli andare ad attaccare durante il brute-force su Chameleon [1] e quali sniffare con Bettercap [9].

Riguardo agli attacchi verso l'Honeypot abbiamo scelto:

- HTTP
- HTTPS
- FTP
- SSH
- TELNET
- SOCKS5

Mentre durante gli attacchi MITM analizzeremo:

- ARP
- DNS [2]
- HTTP
- HTTPS

6.2 Attacco Bruteforce verso l'Honeypot

Durante l'attacco abbiamo usato due macchine ovvero "parrot-client" e "parrot-Honeypot".

La prima con un'istanza di THC-Hydra **[8]** e la seconda con Chameleon [1] avviato.

Test HTTP

Effettueremo delle richieste GET usando il protocollo HTTP verso l'Honeypot.

Stringa configurazione THC-Hydra [8]:

- `hydra -l user -P passwords.txt http-get://192.168.122.131`

```
┌─[client@client-virtualbox]─[~/Desktop]
└──╼ $hydra -l user -P passwords.txt http-get://192.168.122.131
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in mi
litary or secret service organizations, or for illegal purposes (this is non-bin
ding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-27 23:15:
08
[WARNING] You must supply the web page as an additional option or via -m, defaul
t path set to /
[DATA] max 16 tasks per 1 server, overall 16 tasks, 999998 login tries (l:1/p:99
9998), ~62500 tries per task
[DATA] attacking http-get://192.168.122.131:80/
[80][http-get] host: 192.168.122.131   login: user   password: 123456
[80][http-get] host: 192.168.122.131   login: user   password: password
[80][http-get] host: 192.168.122.131   login: user   password: qwerty
[80][http-get] host: 192.168.122.131   login: user   password: 123456789
[80][http-get] host: 192.168.122.131   login: user   password: 12345678
[80][http-get] host: 192.168.122.131   login: user   password: 12345
[80][http-get] host: 192.168.122.131   login: user   password: 1234567
[80][http-get] host: 192.168.122.131   login: user   password: 1234
```

Figura 22. Esecuzione di TCH-Hydra su HTTP

Report generato da Chameleon [1] consultato tramite Grafana [4]:

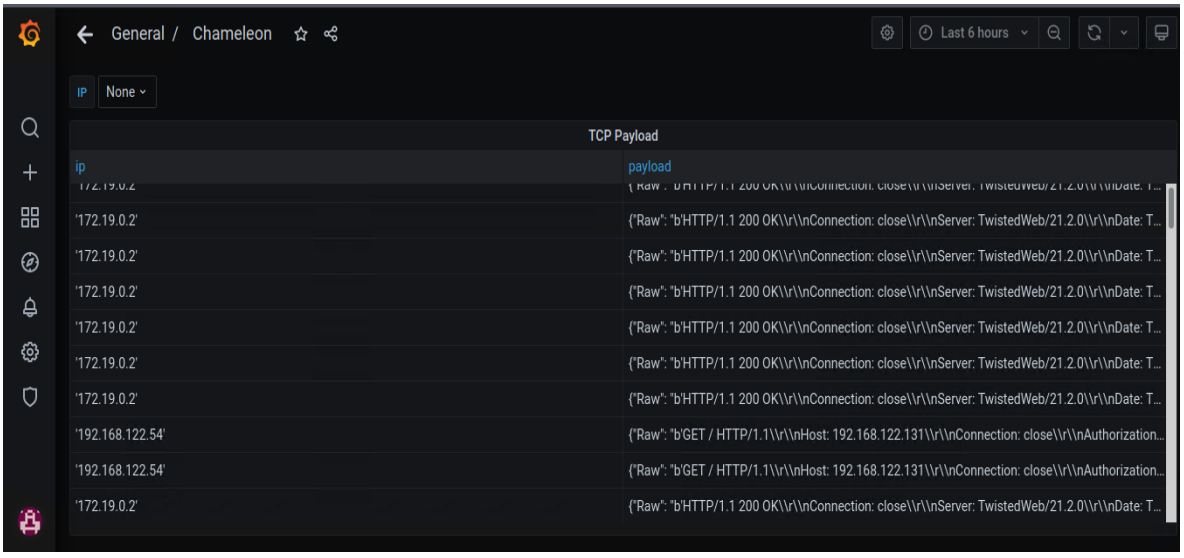


Figura 23. Report attacco su HTTP

Otterremo lo stesso risultato con il protocollo HTTPS.

Test FTP

Effettueremo delle richieste di autenticazione usando il protocollo FTP verso l'Honeypot.

Stringa configurazione THC-Hydra [8]:

- `hydra -l user -P passwords.txt ftp://192.168.122.131`

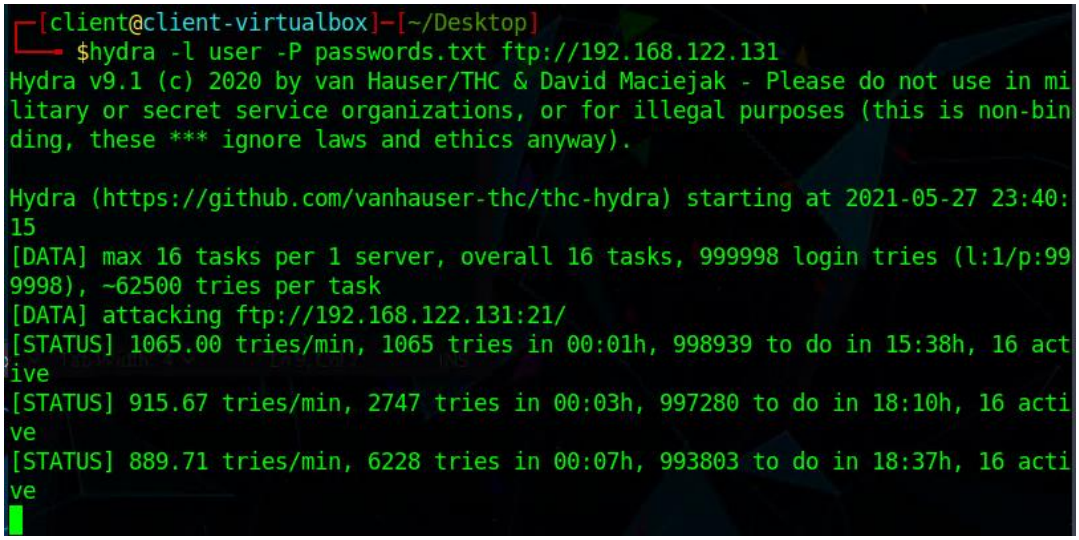


Figura 24. Esecuzione di THC-Hydra su FTP

Report generato da Chameleon [1] consultato tramite Grafana [4]:

Figura 25. Report attacco su FTP

Test SSH

Effettueremo delle richieste di autenticazione usando il protocollo SSH verso l'Honeypot.

Stringa configurazione THC-Hydra [8]:

- `hydra -l user -P passwords.txt ssh://192.168.122.131`

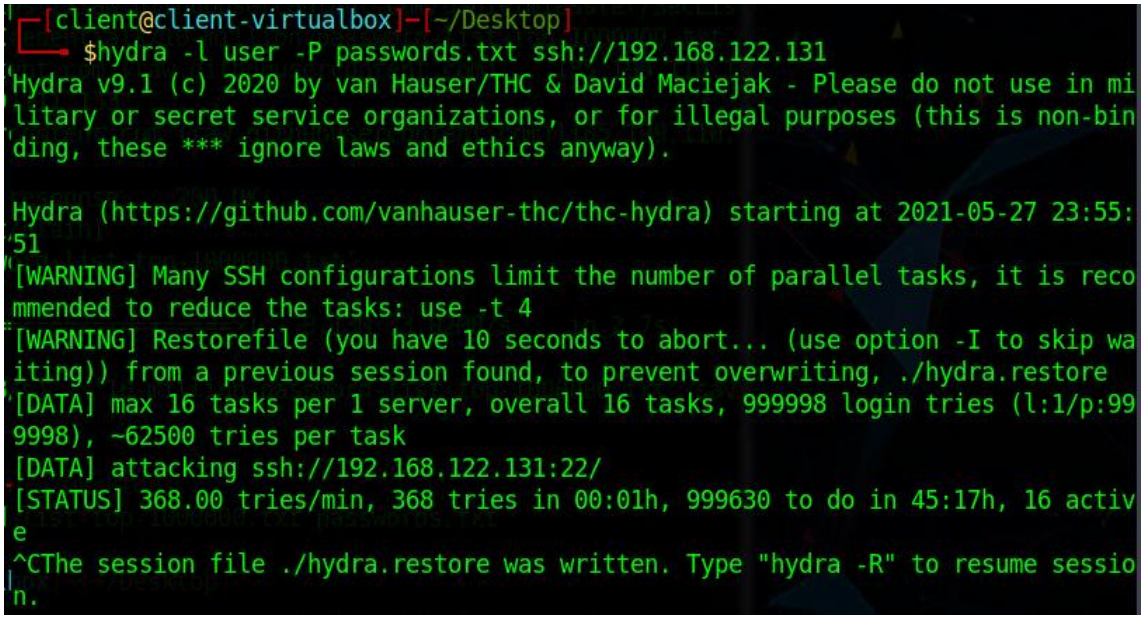


Figura 26. Esecuzione di THC-Hydra su SSH

Report delle credenziali visibili da Grafana [4]:

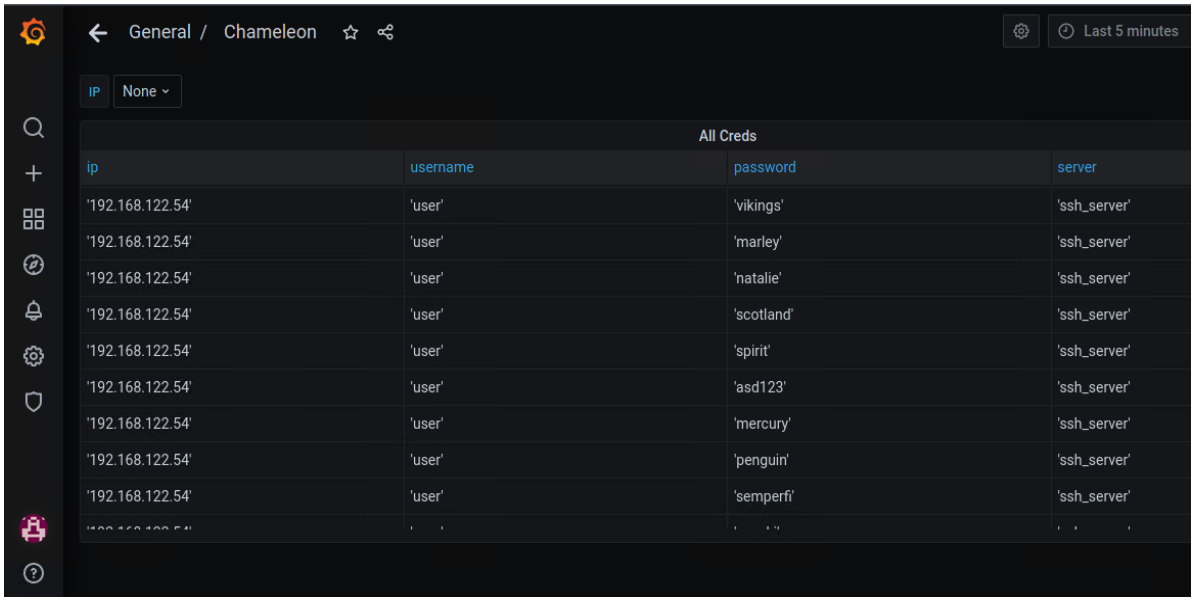


Figura 27. Report attacco su SSH

Test TELNET

Effettueremo delle richieste di autenticazione usando il protocollo TELNET verso l'Honeypot.

Stringa configurazione THC-Hydra [8]:

- `hydra -l user -P passwords.txt telnet://192.168.122.131`

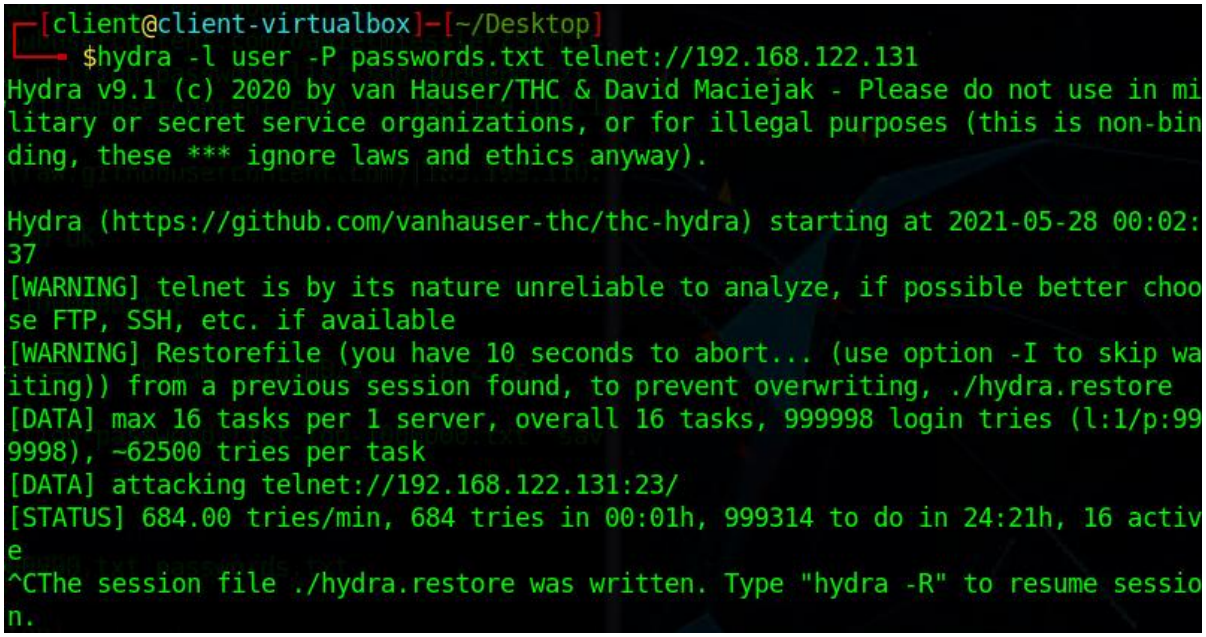


Figura 28. Esecuzione di THC-Hydra su TELNET

Rappresentazione log su Chameleon [1]:

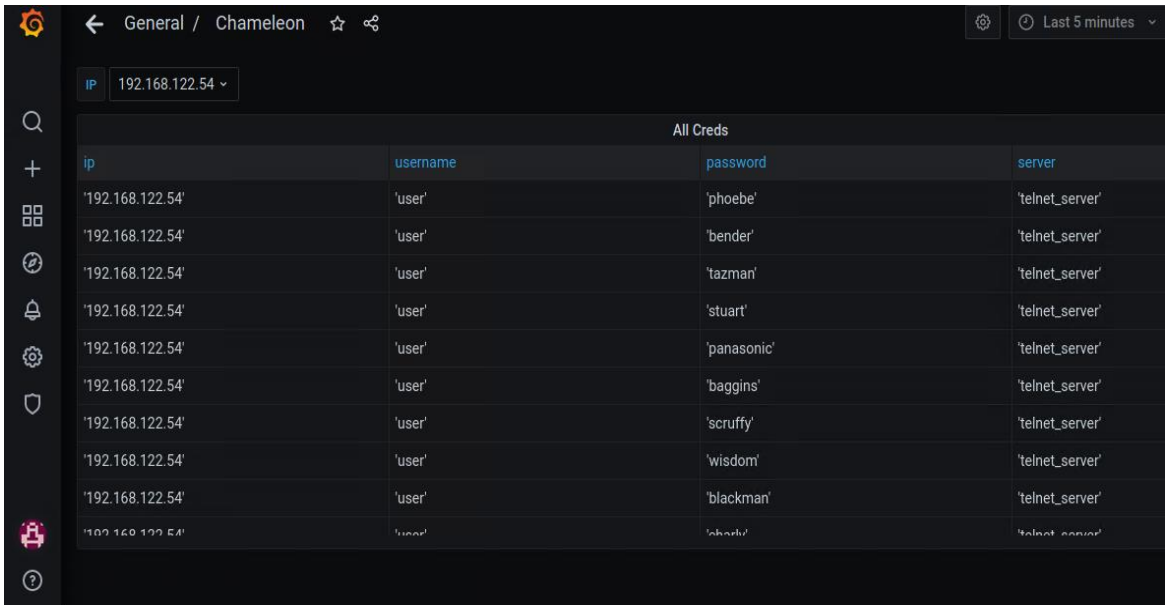


Figura 29. Report attacco su TELNET

Test SOCKS5

Effettueremo delle richieste usando il protocollo SOCKS5 verso l'Honeypot.

Stringa configurazione THC-Hydra [8]:

- `hydra -l user -P passwords.txt socks5://192.168.122.131`

```
┌─[✗]─[client@client-virtualbox]─[~/Desktop]
└──╼ $hydra -l user -P passwords.txt socks5://192.168.122.131
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in mi
litary or secret service organizations, or for illegal purposes (this is non-bin
ding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-05-28 00:24:
57
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip wa
iting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 999998 login tries (l:1/p:99
9998), ~62500 tries per task
[DATA] attacking socks5://192.168.122.131:1080/
[STATUS] 16.00 tries/min, 16 tries in 00:01h, 999982 to do in 1041:39h, 16 activ
e
[STATUS] 5.33 tries/min, 16 tries in 00:03h, 999982 to do in 3124:57h, 16 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume sessio
n.
```

Figura 30. Esecuzione di THC-Hydra su SOCKS5

Log dell'attacco su SOCKS5 visto da Grafana [4]:

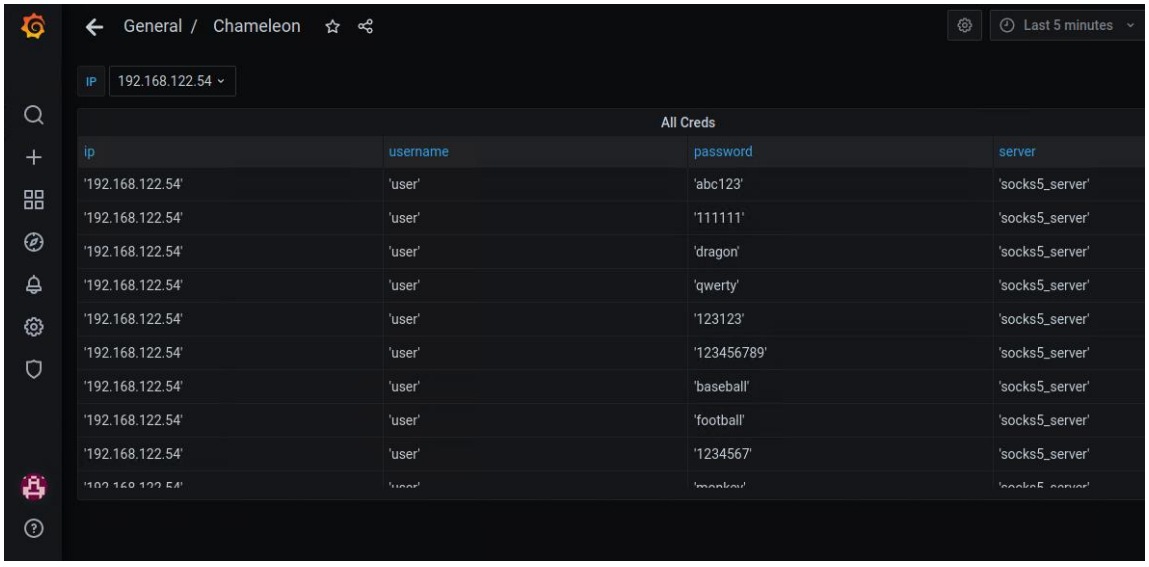


Figura 31. Report attacco su SOCKS5

6.3 Intercettazione traffico con tecnica MITM

In questa sezione andremo, tramite Bettercap [9], a identificare un host vittima per poi intercettarne il traffico sia in HTTP sia in HTTPS.

Identificazione target

Prima di tutto dobbiamo rintracciare l'indirizzo della vittima e per questo effettueremo una scansione di rete.

Dobbiamo avviare il Daemon di Discovery, per poi terminarlo dopo qualche secondo.
Inizieremo con lo sniffing del traffico in HTTP.

I comandi per la scansione sono:

- net.probe on
- net.probe off

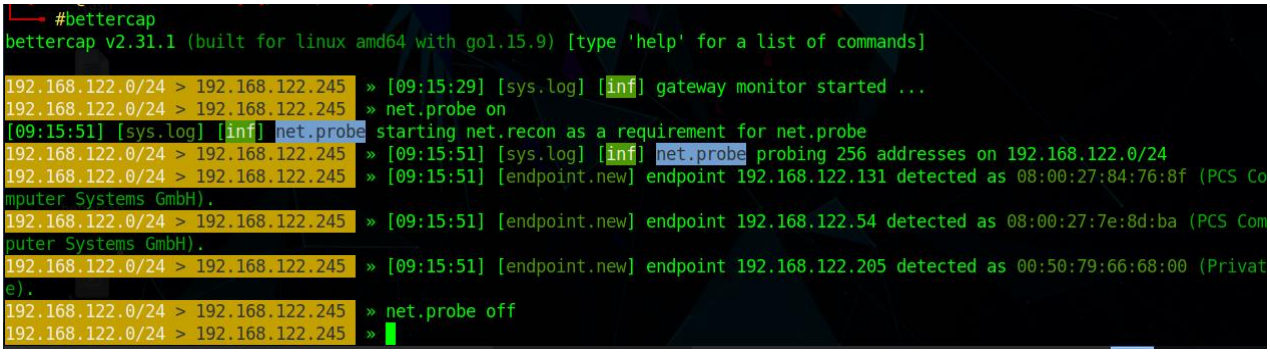


Figura 32. Bettercap net.probe

Per consultare i risultati della nostra scansione sarà necessaria la seguente stringa:

- net.show

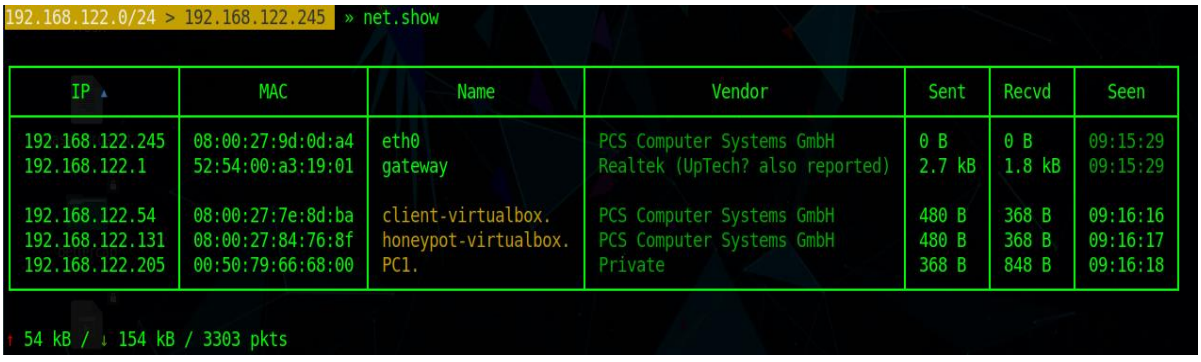


Figura 33. Bettercap net.show

Sniffing traffico HTTP

Prima di tutto, proveremo ad effettuare uno sniffing verso la macchina denominata “parrot-client”, considerata la macchina “vittima”, utilizzando come vettore il protocollo HTTP con la seguente sequenza di comandi :

- `set http.proxy.sslstrip true`
- `set net.sniff.verbose false`
- `set arp.spoof.targets 192.168.122.54`
- `net.sniff on`
- `http.proxy on`
- `arp.spoof on`

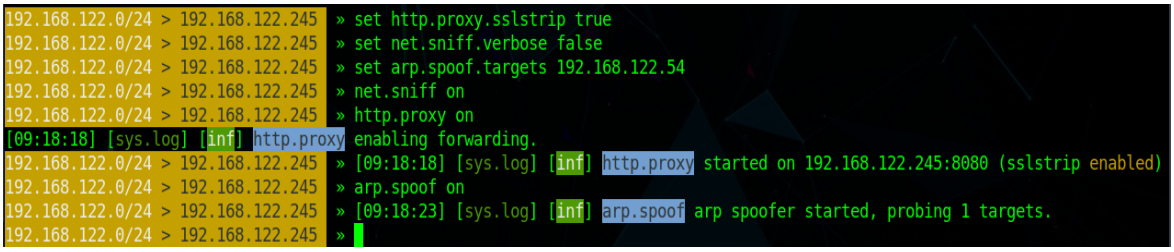


Figura 34. Bettercap sniffing HTTP

Una volta avviata la sessione, vedremo scorrere nella nostra console il log del traffico HTTP della vittima.

Abbiamo preso come esempio la login di una agenzia di marketing che presenta un Form non protetto, ovvero non in HTTPS:

- http://login.ebiquiti.com

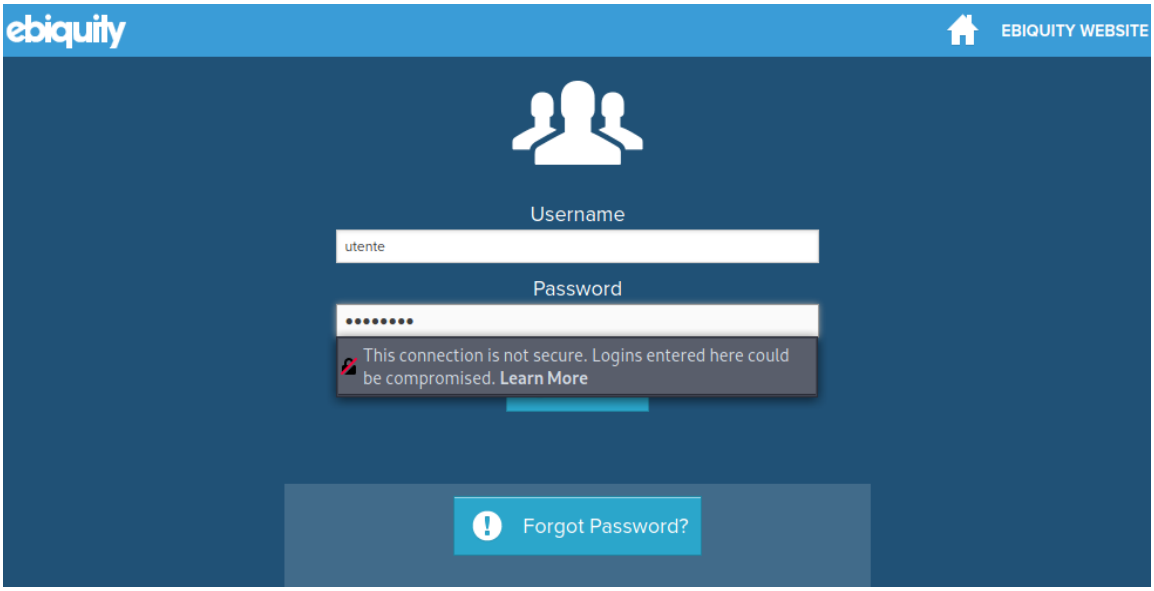


Figura 35. Sito di esempio in HTTP

Tenteremo di inserire delle credenziali fittizie che ci aspettiamo di osservare nei log di Bettercap [9].

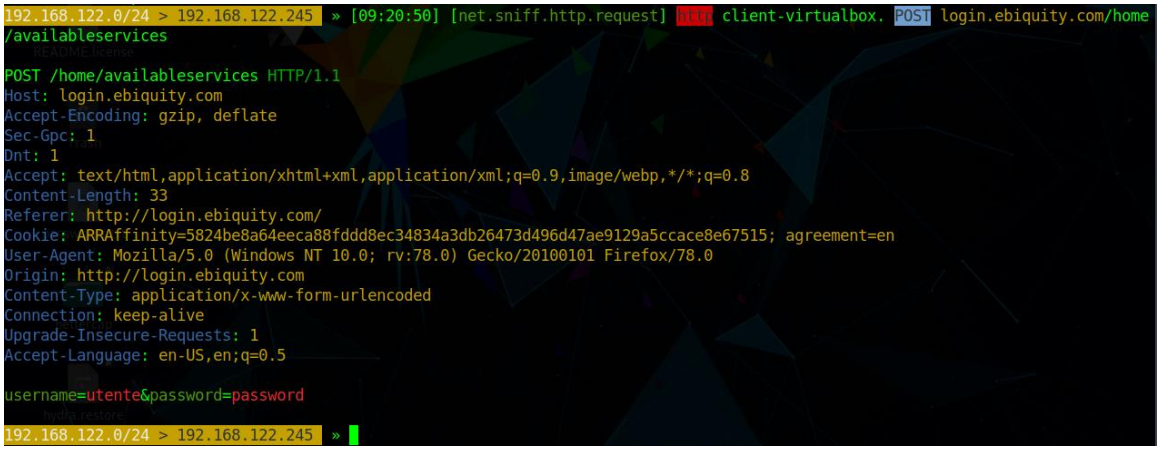


Figura 36. Esempio di frame HTTP intercettato

Come si può notare dalla figura, Bettercap [9] ci presenta un log con l'indicazione dell'URL catturato, il protocollo e l'Header del pacchetto HTTP.

Quindi si può dedurre che è stato fatto un caricamento di dati, tramite la dicitura POST, che identifica un metodo di richiesta HTTP per inviare un'entità verso una specifica risorsa, in questo caso il PATH della login del sito precedente.

In basso, troviamo i dati catturati dalla Form, cioè l'username e la password immessi.

Sniffing traffico HTTPS

Tratteremo poi l'intercettazione del protocollo HTTPS.

La procedura per identificare la vittima si effettua sempre con il medesimo comando di Discovery trattato precedentemente.

Proveremo ad effettuare uno sniffing verso la macchina denominata "parrot-client", considerata la macchina "vittima", utilizzando come vettore il protocollo HTTPS con la seguente sequenza di comandi :

- `set https.proxy.sslstrip true`
- `set net.sniff.verbose false`
- `set arp.spoof.targets 192.168.122.54`
- `net.sniff on`
- `https.proxy on`
- `arp.spoof on`

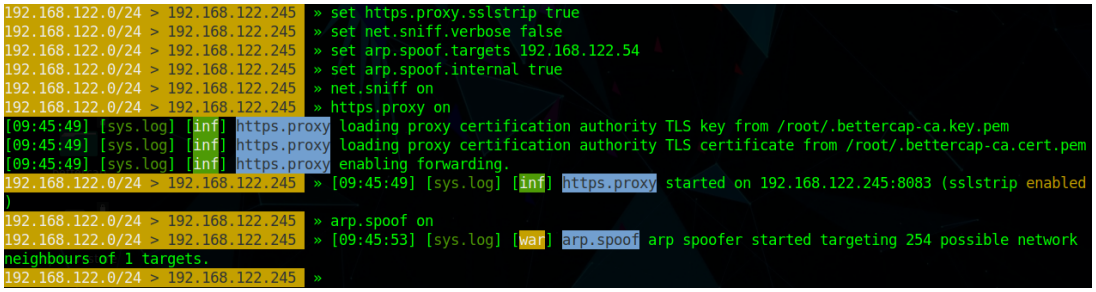


Figura 37. Avvio sniffing HTTPS

Una volta avviata la sessione, vedremo scorrere nella nostra console il log del traffico HTTPS della vittima.

Abbiamo preso come esempio delle pagine protette per poter vedere se si riusciva a catturare il traffico:

- https://www.libero.it
- https://www.vimeo.com
- https://www.twitter.com

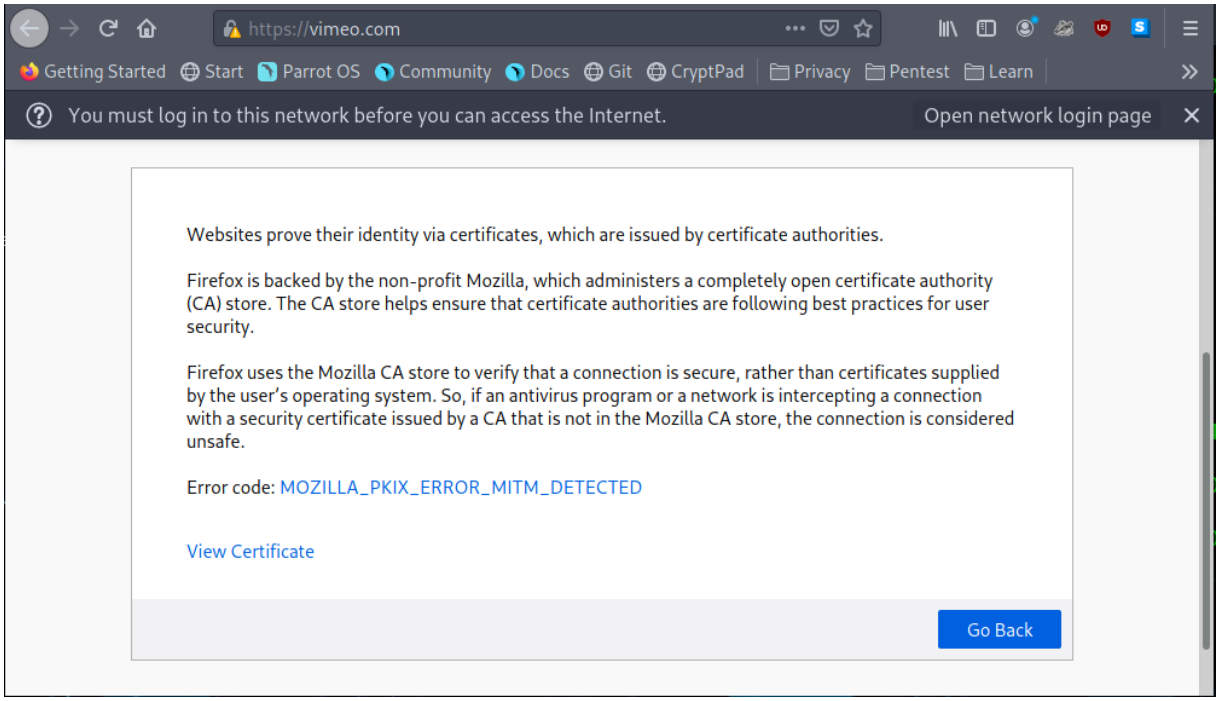


Figura 38. Sito esempio HTTPS Vimeo.com

Naturalmente, con il procedere degli aggiornamenti dei vari browser, sono state implementate delle policy di sicurezza per poter rilevare attacchi alle connessioni protette, verificando la validità del certificato immesso.

Questo ha avuto la conseguenza di non poter portare a termine l'attacco.

Tratteremo poi nelle analisi queste casistiche.

```
[09:45:49] [sys.log] [inf] https.proxy loading proxy certification authority TLS key from /root/.bettercap-ca.key.pem
[09:45:49] [sys.log] [inf] https.proxy loading proxy certification authority TLS certificate from /root/.bettercap-ca.cert.pem
[09:45:49] [sys.log] [inf] https.proxy enabling forwarding.
192.168.122.0/24 > 192.168.122.245 » [09:45:49] [sys.log] [inf] https.proxy started on 192.168.122.245:8083 (sslstrip enabled
)
192.168.122.0/24 > 192.168.122.245 » arp.spoof on
192.168.122.0/24 > 192.168.122.245 » [09:45:53] [sys.log] [war] arp.spoof arp spoofer started targeting 254 possible network
neighbours of 1 targets.
192.168.122.0/24 > 192.168.122.245 » [09:47:12] [sys.log] [inf] https.proxy creating spoofed certificate for push.services.mo
zilla.com:443
192.168.122.0/24 > 192.168.122.245 » [09:47:13] [net.sniff.https] sni client-virtualbox. > https://push.services.mozilla.com
```

Figura 39. Esempio di sniffing HTTPS non funzionante

Sniffing traffico HTTPS tramite Hijacking dell'HSTS

Per poter riuscire a catturare le richieste e le risposte ricevute tramite il protocollo HTTPS, non ci è bastato effettuare un semplice proxy, ma è stato necessario bypassare una procedura che implementa una politica di sicurezza per le comunicazioni web, chiamato HSTS.

L'HSTS, cioè HTTP Strict Transport Security, è emesso dal server nell'intestazione di un messaggio di una comunicazione HTTPS.

Con questa tecnica è stato possibile scongiurare il Man in The Middle con la tecnica di SSL-stripping, cioè permette di convertire una connessione HTTP sicura, appoggiata a SSL o TLS, in una connessione HTTP in chiaro.

Si è proceduti quindi nell'eseguire un caplet messo a disposizione da Bettercap [9] che permette di fare ciò.

I comandi che sono stati effettuati sono i seguenti:

- `bettercap -eval "caplets.update; ui.update; q"`

  Per poter aggiornare i caplets e la UI.

- `bettercap -caplet hstshijack/hstshijack`

  Per poter avviare il caplet di Hijack dell'HSTS, con preimpostate le configurazioni necessarie, come i certificati da bypassare e i domini da attaccare.

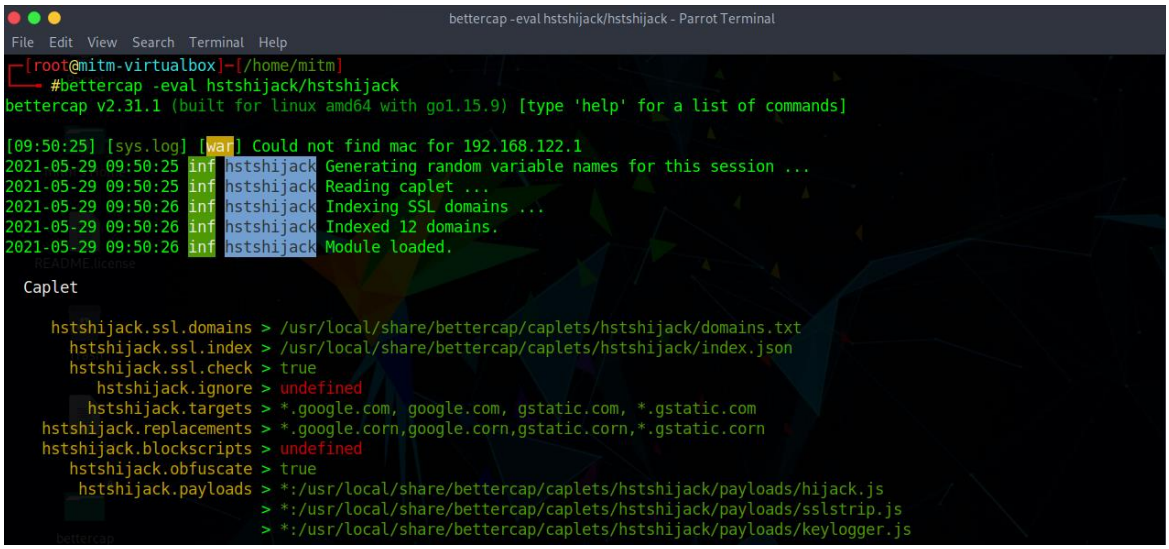


Figura 40. Hijack HSTS bettercap parte 1

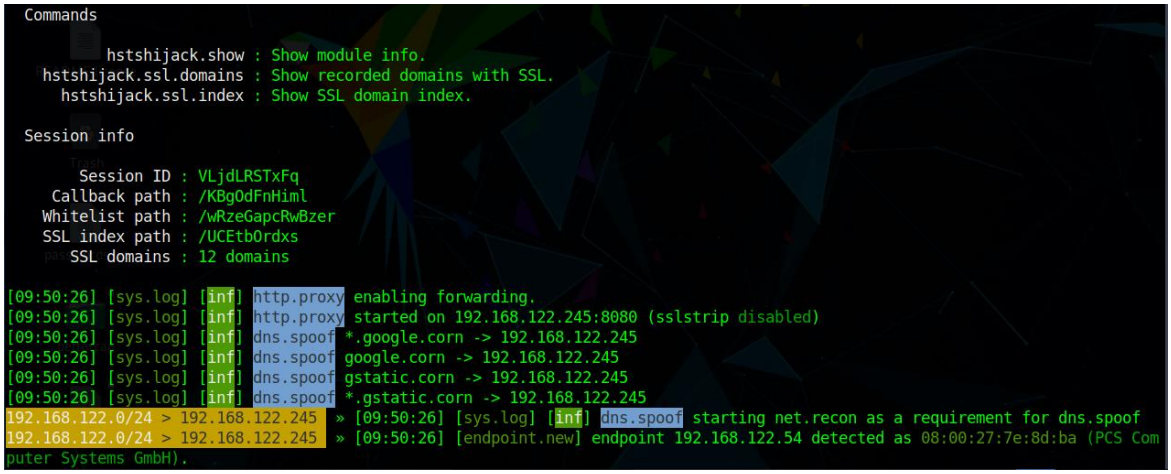


Figura 41. Hijack HSTS bettercap parte 2

Avvieremo poi il Daemon di Discovery dei dispositivi connessi alla rete e setteremo il target che in questo caso è la macchina “vittima” dell’attacco:

- net.probe on
- net.probe off
- net.show
- set http.proxy.sslstrip true
- set net.sniff.verbose false
- set arp.spoof.targets 192.168.122.54
- arp.spoof.fullduplex true
- arp.spoof on
- http.proxy on
- net.sniff on

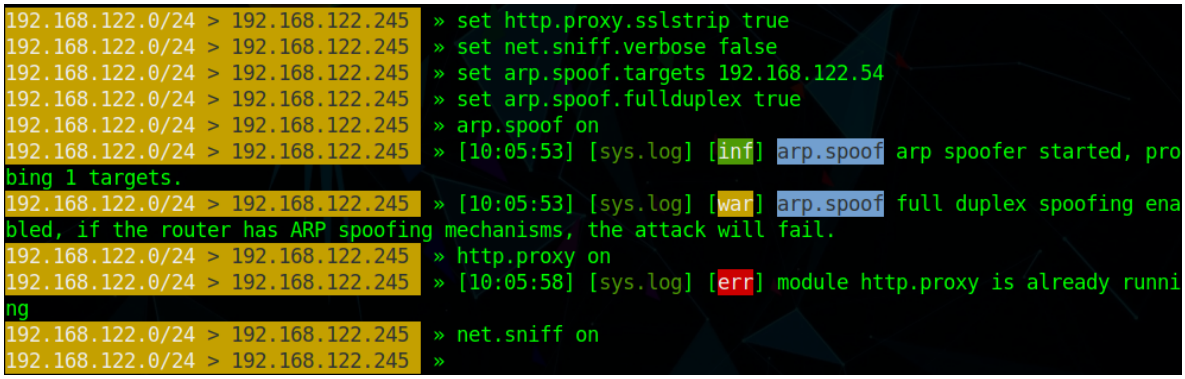


Figura 42. Abilitazione sniffing HSTS con HSTS

Come controprova abbiamo navigato su Google tramite la macchina “parrot-client”, e si è visto cosa riusciva a catturare:

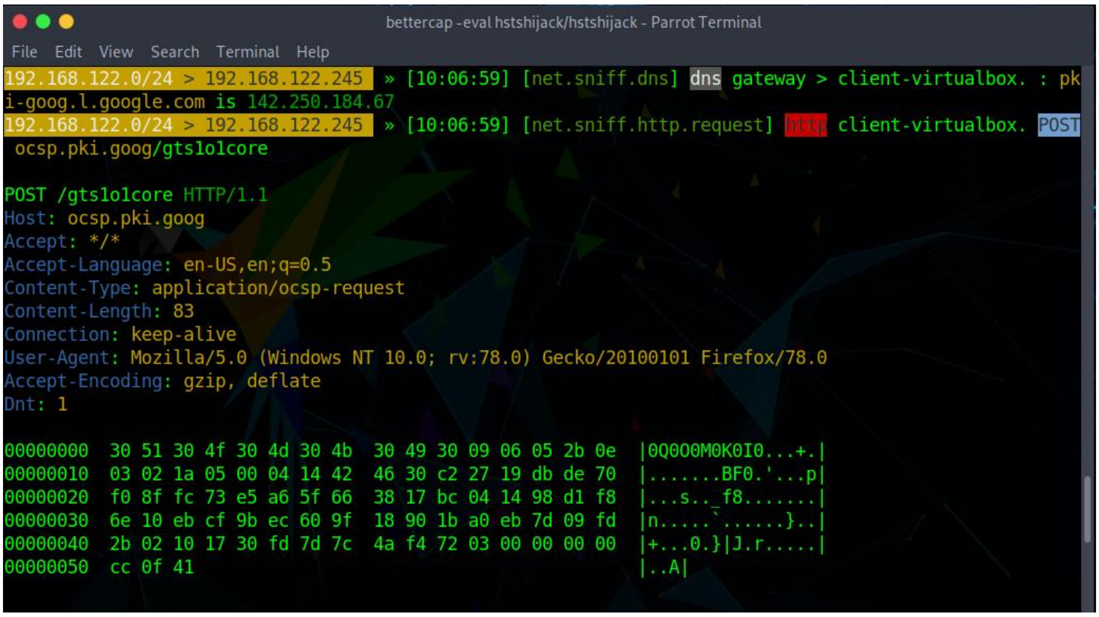


Figura 43. Esempio sniffing parte 1

È stata effettuata un'altra prova sul sito pccomponents.com :

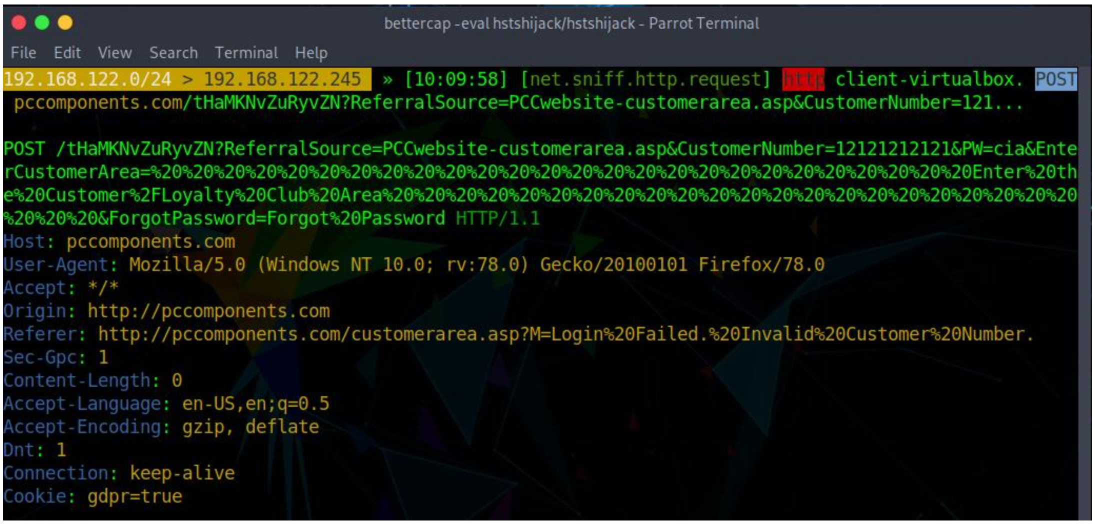


Figura 44. Esempio sniffing parte 2

Inserendo anche dei dati di accesso fittizi come prova:

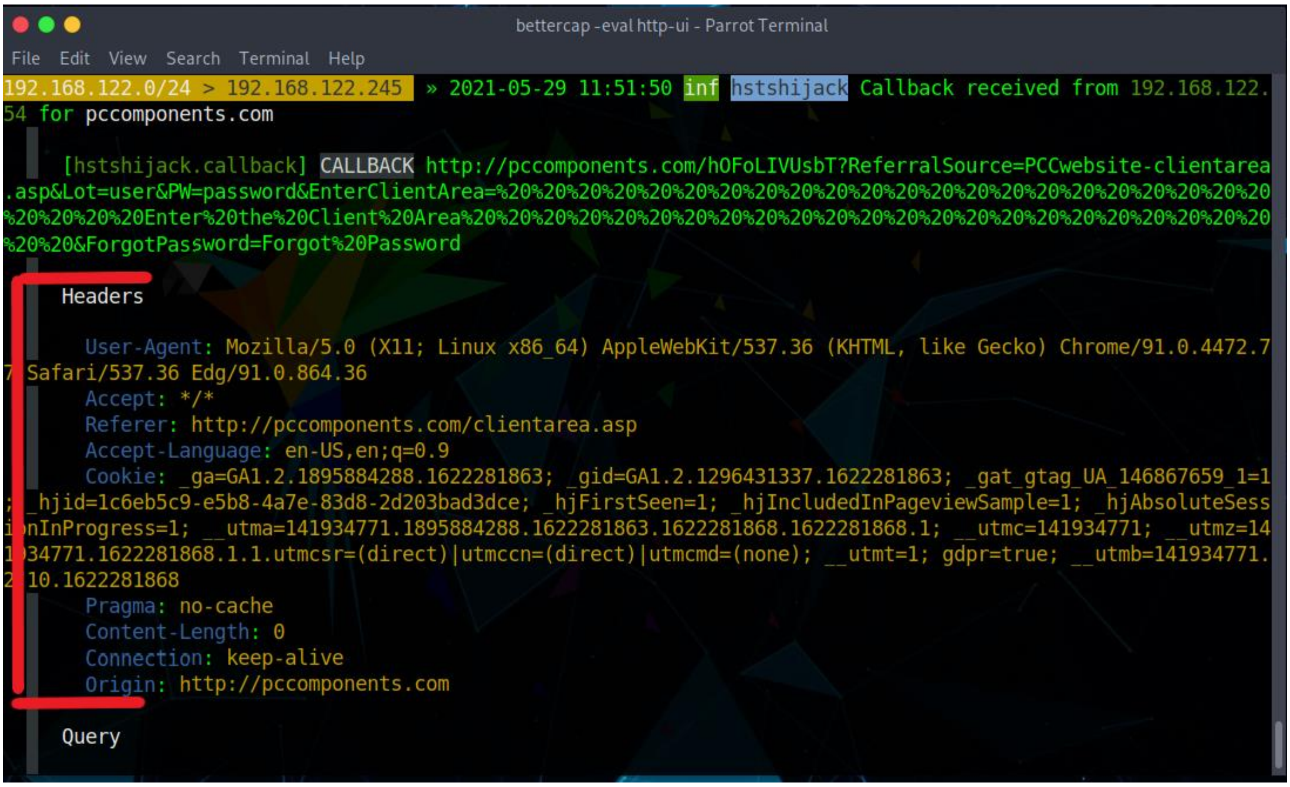


Figura 45. Esempio sniffing parte 3

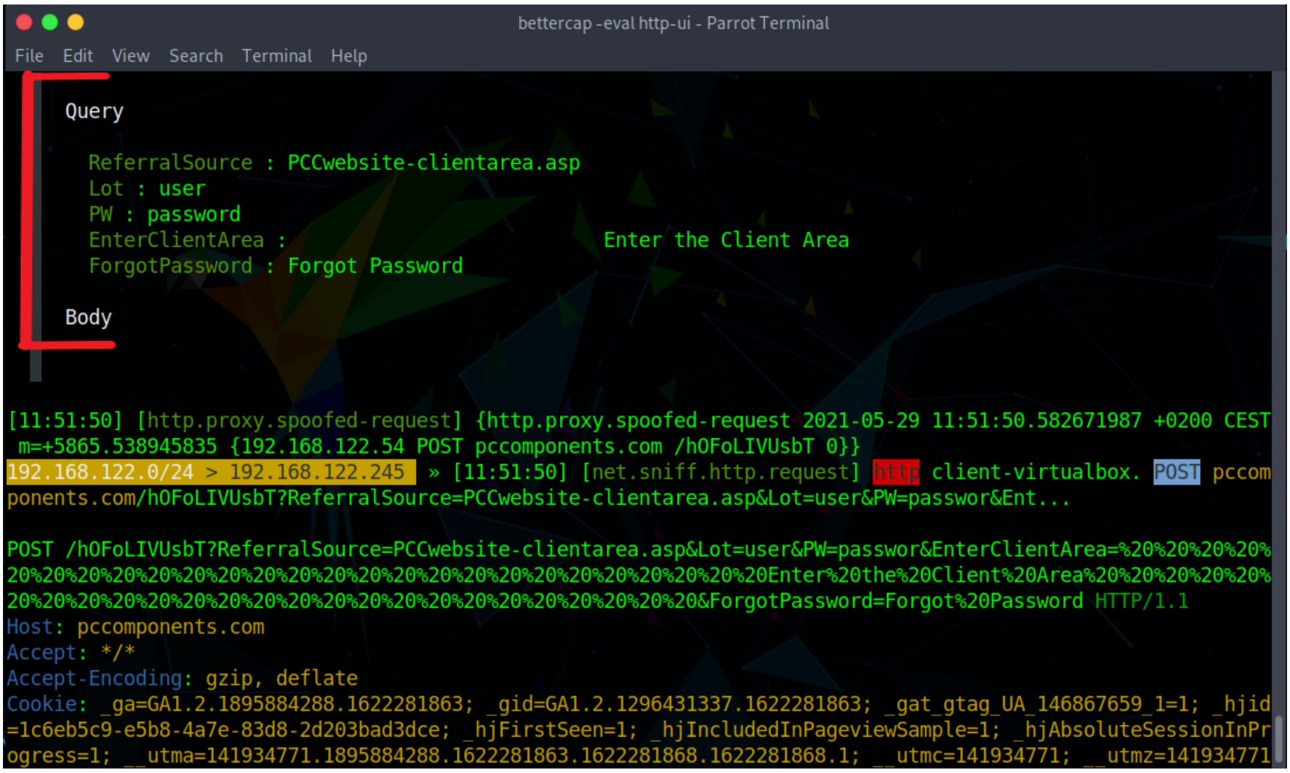


Figura 46. Esempio sniffing parte 4

Si può notare in figura come Bettercap [9] riesca a restituirmi l'Header del pacchetto e il suo payload contenente la query generata dalla pagina e il suo body.

E di conseguenza anche su Libero.it:

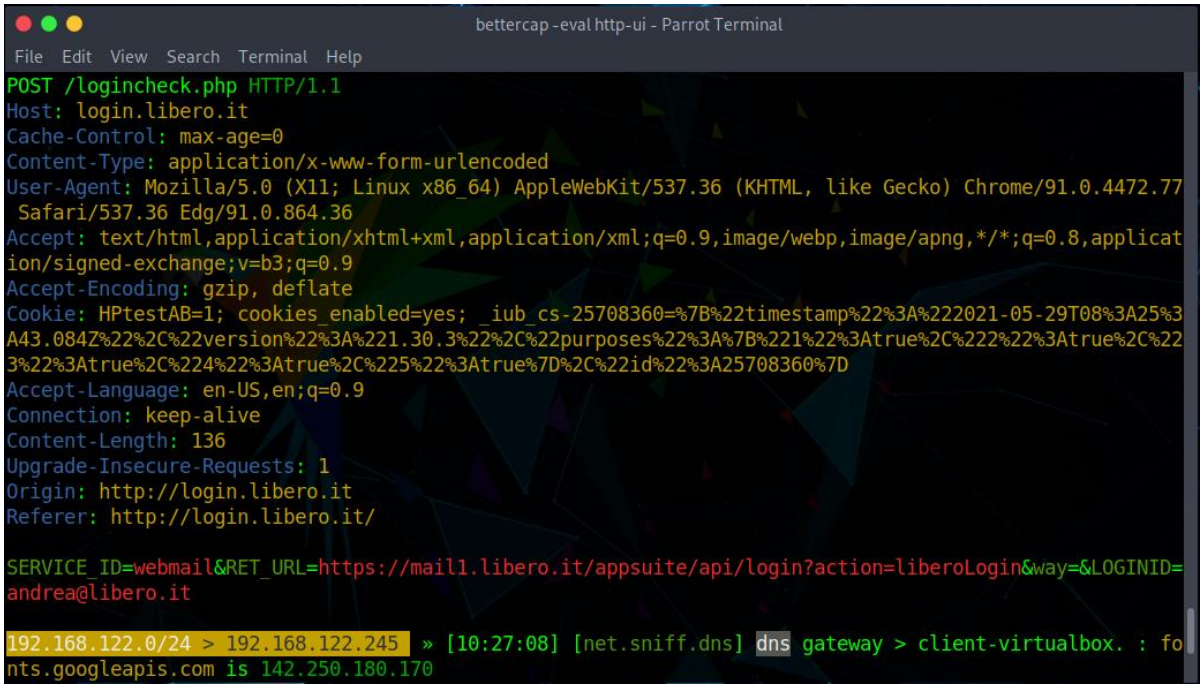


Figura 47. Esempio sniffing parte 5

6.4 Alterazione traffico con tecnica MITM

Ora dimostreremo come sia possibile alterare il traffico web passante su protocollo HTTP e HTTPS, riuscendo così ad eseguire codice potenzialmente malevolo all'interno della macchina vittima.

Per poter realizzare questa tecnica, andremo ad usare il modulo "injectjs" contenuti nei Transparent proxy relativi ad entrambi i protocolli nominati in precedenza.

Inoltre abbiamo anche realizzato uno script in JavaScript in grado di far apparire un popup in ogni pagina visualizzata con un semplice alert.

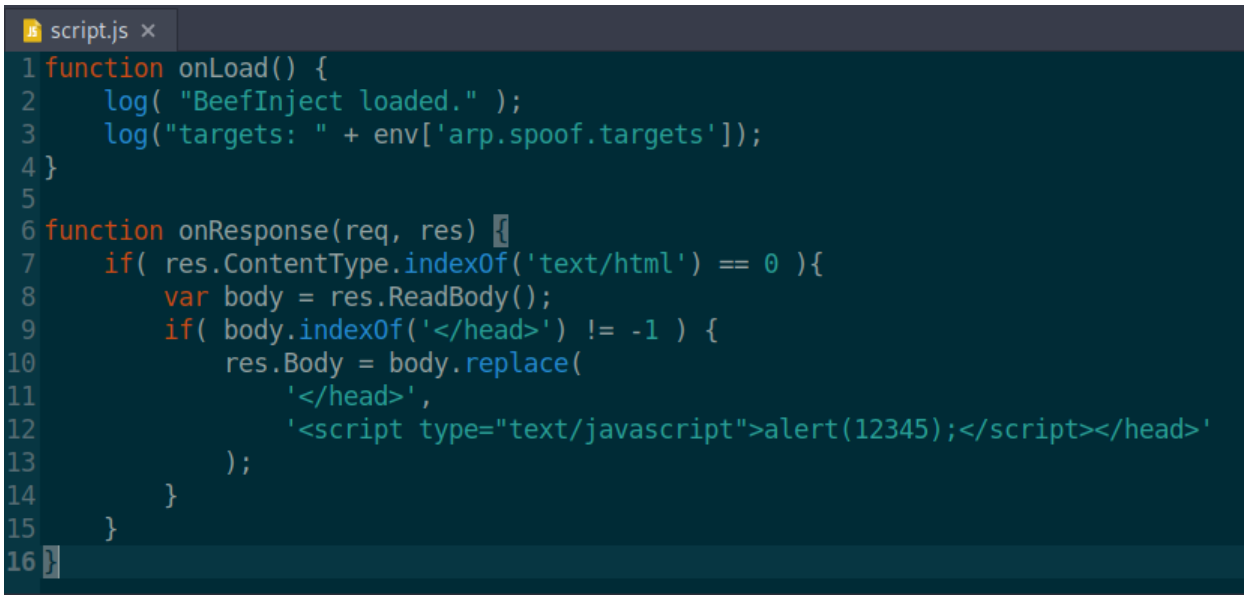


Figura 48. Script JS alterazione traffico

Per poter avviare l'attacco verso la macchina "parrot-client", dovremo configurare Bettercap [9] con i seguenti parametri:

- `set http.proxy.script script.js`
- `set http.proxy.sslstrip true`
- `set net.sniff.verbose false`
- `set arp.spoof.targets 192.168.122.54`
- `http.proxy on`
- `arp.spoof on`
- `set https.proxy.script script.js`
- `https.proxy on`

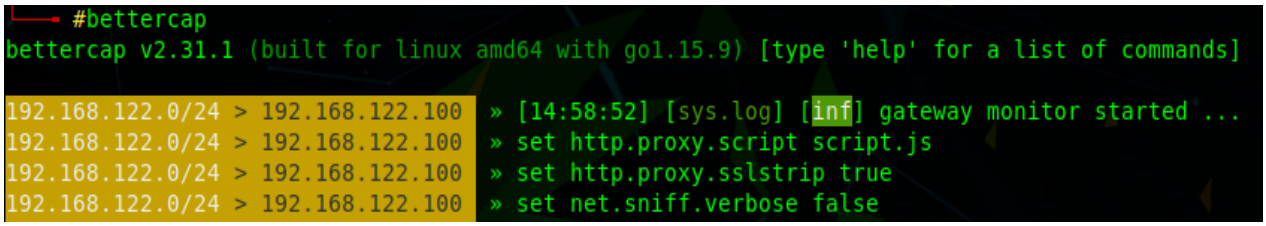


Figura 49. Comandi Bettercap avvio alterazione traffico parte 1

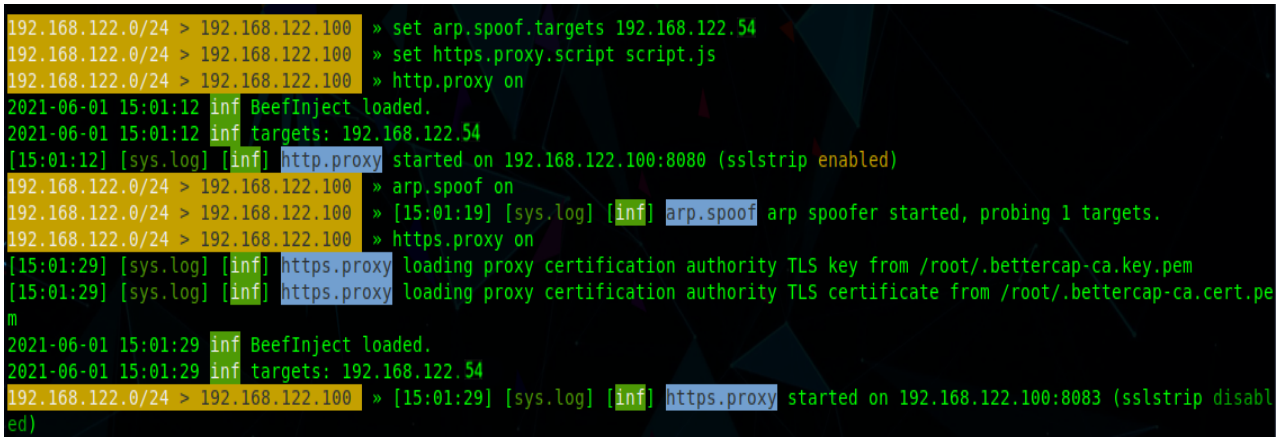


Figura 50. Comandi Bettercap avvio alterazione traffico parte 2

Successivamente si è visitato un qualsiasi sito web non protetto per verificare se fosse andato tutto a buon fine.

- login.ebiquity.com

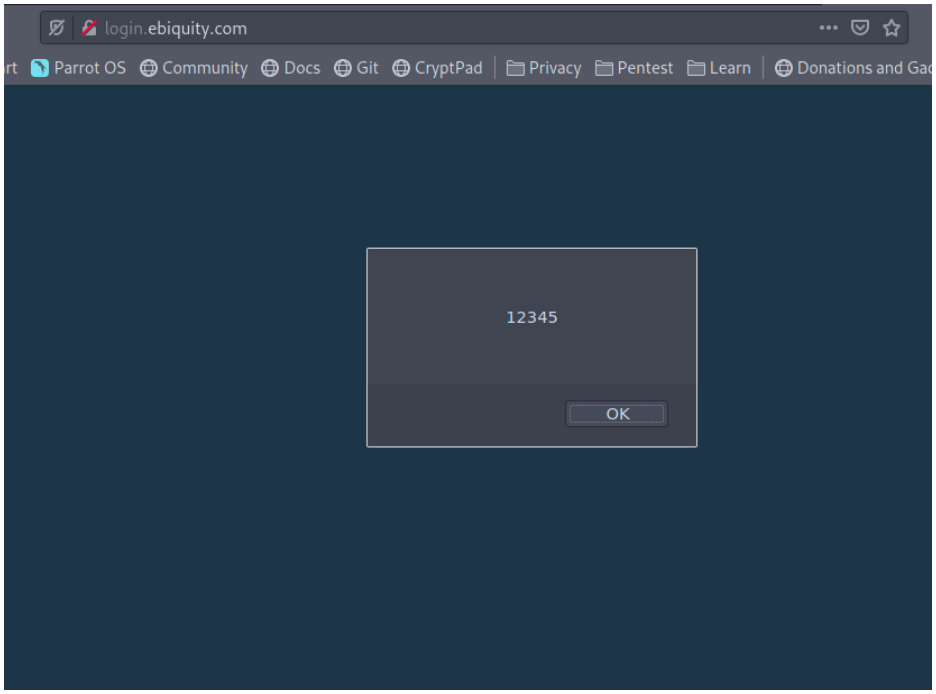


Figura 51. Popup sito ebiquity.com

- motodepocafacile.it

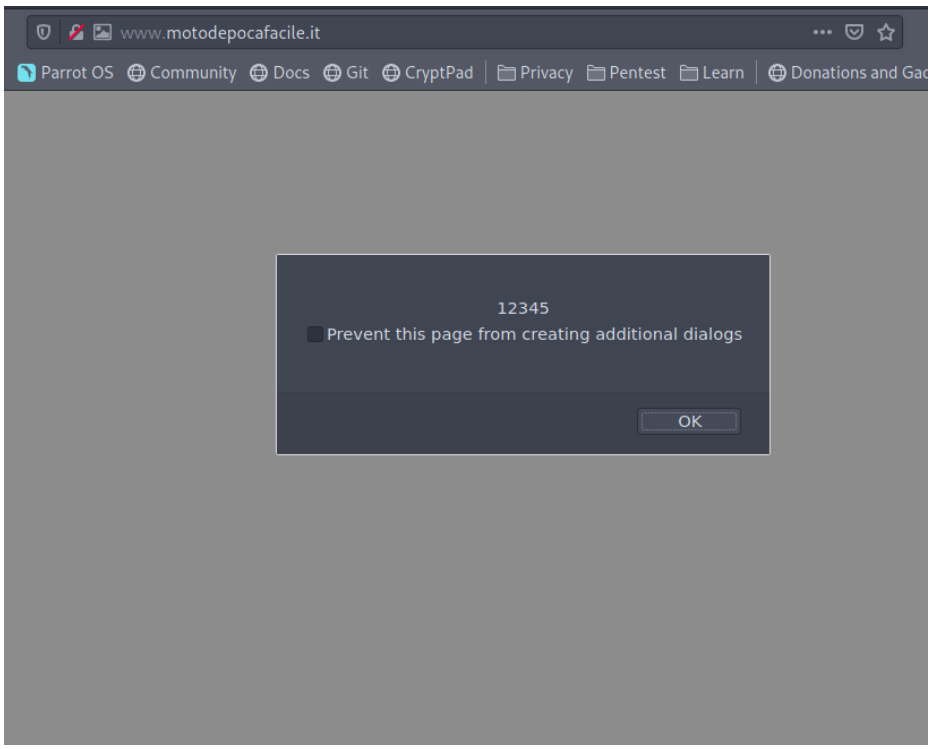


Figura 52. Popup sito motodepocafile.it

Come si può notare in figura, il testing ha avuto successo ed è stato iniettato nel body del DOM HTML lo script precedentemente mostrato.

6.5 Intercettazione di un attacco su Honeypot con tecnica MITM

Proveremo anche ad effettuare lo sniffing durante un attacco brute-force sempre condotto tramite THC-Hydra **[8]** su di un honeypot Chameleon [1].

Quindi cattureremo il traffico generato dalla macchina “parrot-mitm” in uscita verso “parrot-honeypot.”

In questo modo potremo vedere cosa è sotto attacco e da chi proviene l’attacco.

Risulta utile come mossa difensiva se si vuole “spiare” chi compie l’attacco, e poter bloccare in tempo la comunicazione da lui instaurata.

Riguardo la configurazione di Bettercap [9] è la medesima descritta nei sotto capitoli precedenti.

Sniffing bruteforce HTTP

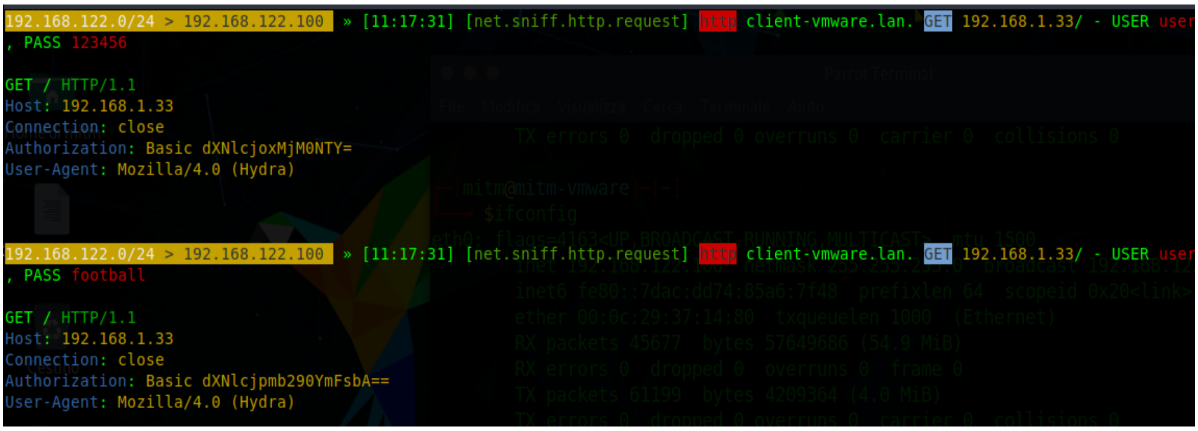


Figura 53. Sniffing bruteforce HTTP

Sniffing bruteforce FTP

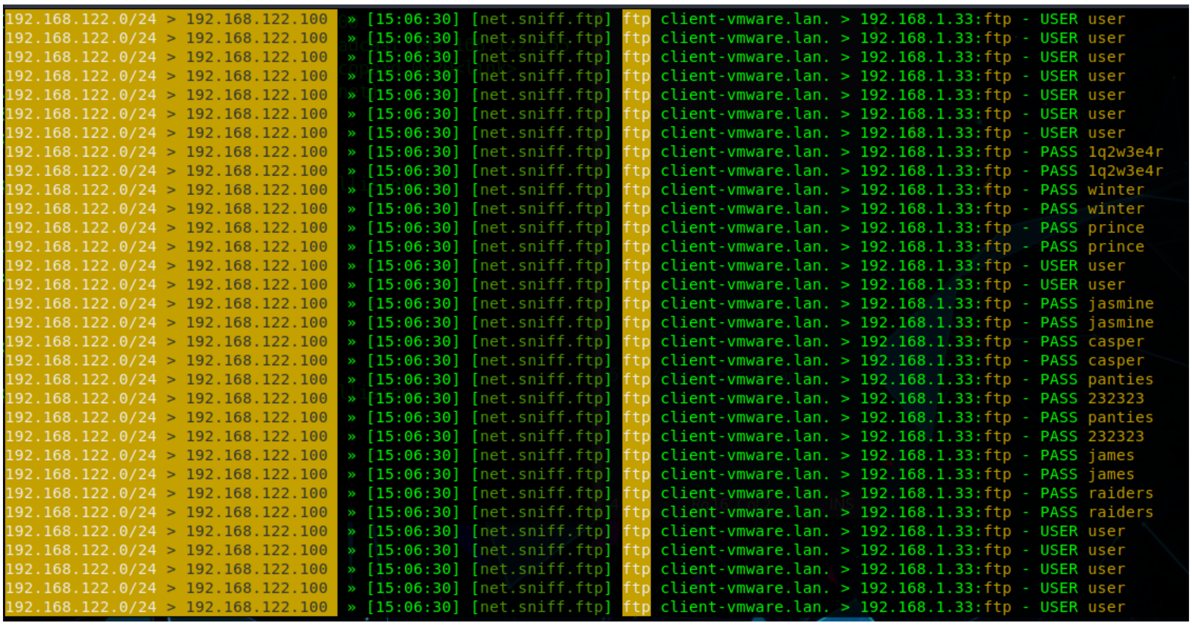


Figura 53. Sniffing bruteforce FTP

7. Analisi dei risultati

In questa parte andremo a commentare le dimostrazioni contenute nel capitolo 6, sia per quanto riguarda l'honeypot sia il MITM.

7.1 Report degli attacchi su Honeypot

Ora andremo a visionare il report complessivo generato da Grafana [4] dopo i vari tentativi di accesso fatti con THC-Hydra **[8]**.

Il report è suddiviso in tre sezioni:

- sommario dei protocolli

In questa sezione possiamo visionare i protocolli appena attaccati ed il numero dei tentativi, in attacco brute-force, su ciascuno.

Inoltre, in basso abbiamo la possibilità di leggere le credenziali con le quali abbiamo forzato il sistema, suddivise per protocollo.

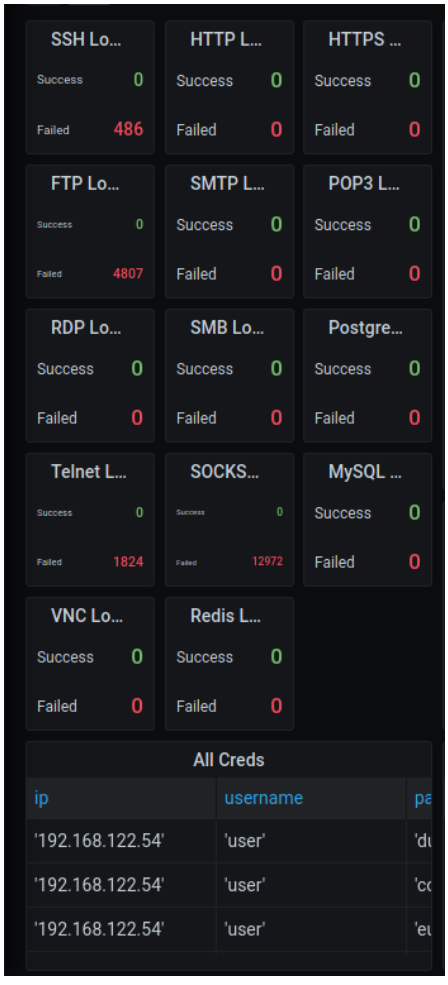


Figura 54. Analisi Grafana sommario protocolli

- dettagli dei pacchetti in entrata

Oltre al sommario dei protocolli possiamo ottenere informazioni dettagliate su tutti i pacchetti ricevuti da Chameleon [1].
Infatti, avremo a disposizione i dettagli sui loro body, ben suddivisi per tipologia.

Così facendo, potremo conoscerne il contenuto e la sorgente.

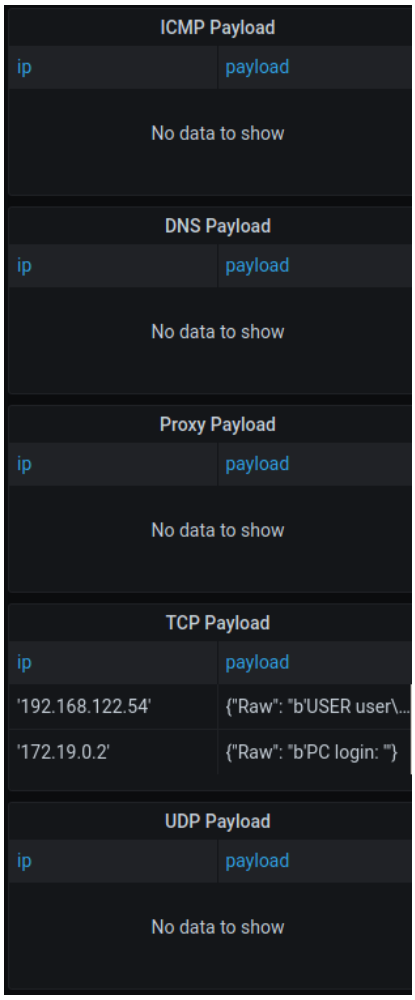


Figura 55. Analisi Grafana pacchetti in entrata

In questo caso l'unica sezione popolata da dati è quella del TCP, considerato che i protocolli provati durante i test sfruttavano l'omonimo tipo di pacchetto.

Per esempio, nel caso in cui avessi effettuato un "ping", esso sarebbe stato registrato nella sezione ICMP.

- riepilogo complessivo

Nella sezione centrale troviamo un riepilogo complessivo del traffico ricevuto dall'honeypot.

In alto troviamo un grafico a punti rappresentate le sessioni aperte ed al centro il numero di byte e pacchetti in entrata e uscita.

Si può vedere chiaramente dal grafico il momento in cui sono partiti i 5 attacchi svolti nel capitolo precedente.

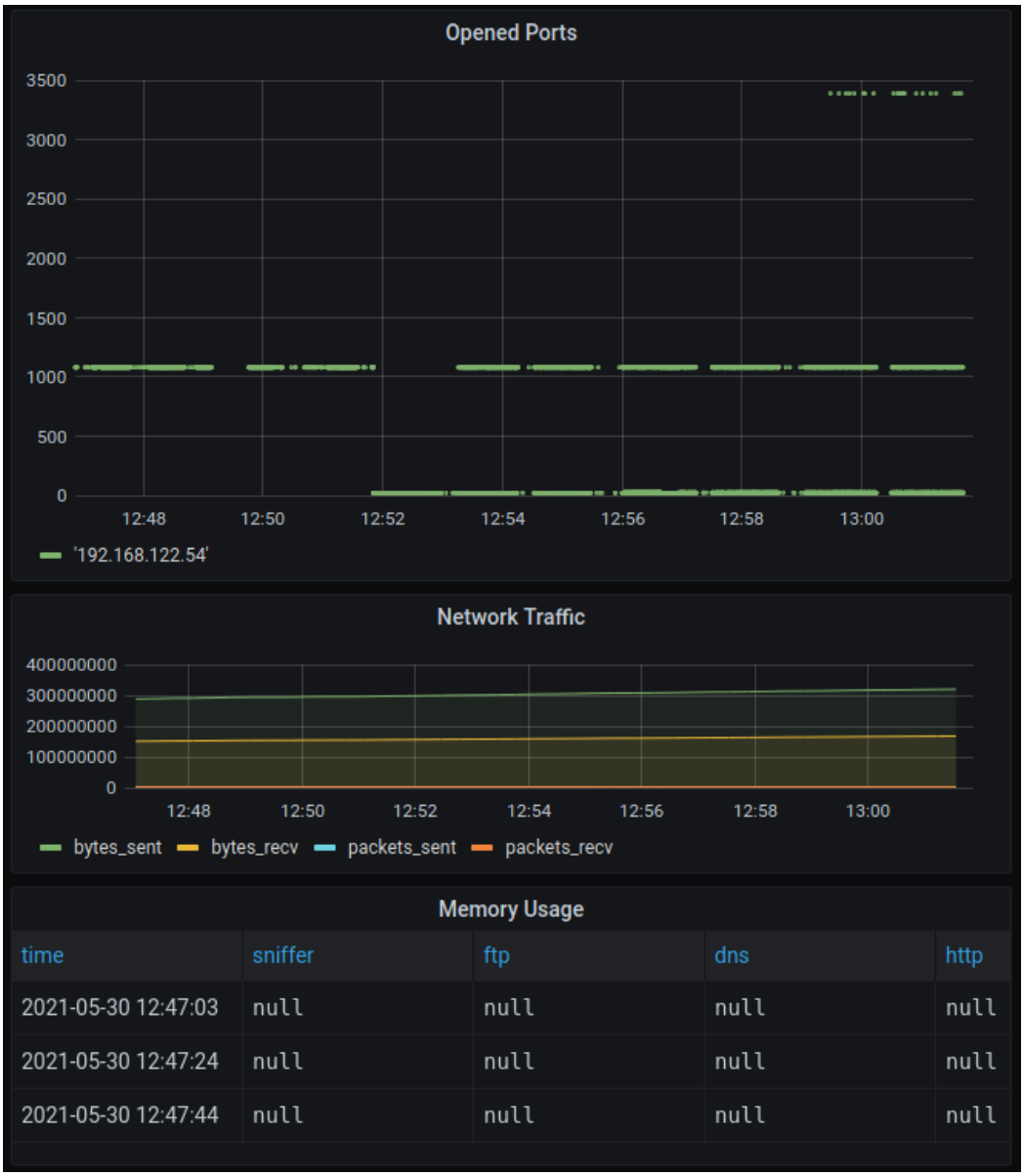


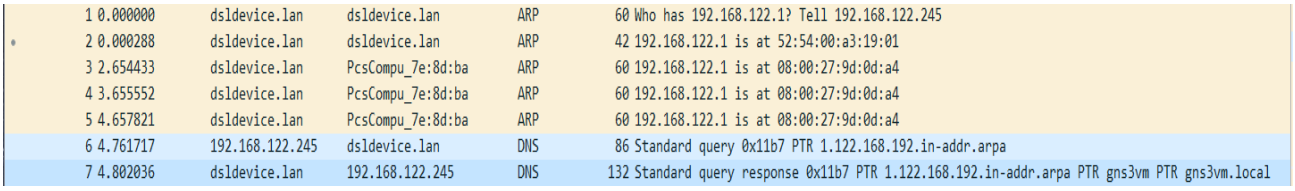
Figura 56. Analisi Grafana riepilogo complessivo

7.2 Verifica dello spoofing tramite MITM

Durante la fase di intercettazione, tramite il software GNS3 [12], possiamo osservare lo scambio di pacchetti tra il nostro attaccante e la nostra vittima.

I pacchetti che invia Bettercap [9] alla nostra vittima la induce a cambiare rotta verso un host a nostro piacimento che in questo caso sarà la macchina parrot-mitm.

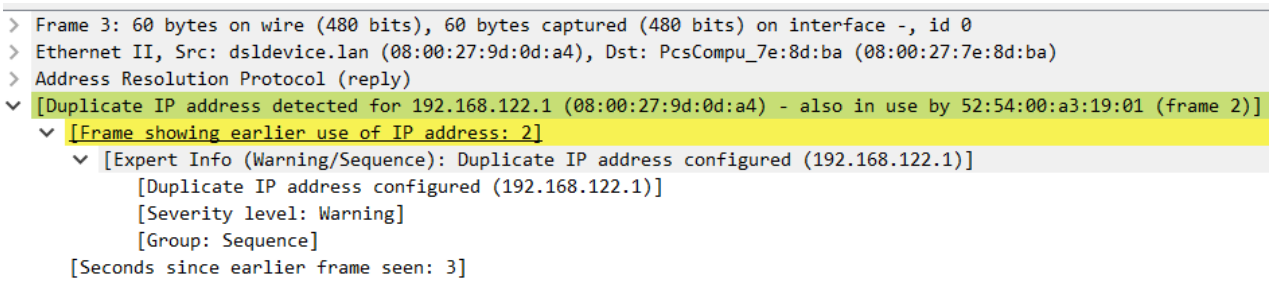
Utilizzeremo Wireshark [11] per mostrare i pacchetti trasferiti.



1	0.000000	dsldevice.lan	dsldevice.lan	ARP	60 Who has 192.168.122.1? Tell 192.168.122.245
2	0.000288	dsldevice.lan	dsldevice.lan	ARP	42 192.168.122.1 is at 52:54:00:a3:19:01
3	2.654433	dsldevice.lan	PcsCompu_7e:8d:ba	ARP	60 192.168.122.1 is at 08:00:27:9d:0d:a4
4	3.655552	dsldevice.lan	PcsCompu_7e:8d:ba	ARP	60 192.168.122.1 is at 08:00:27:9d:0d:a4
5	4.657821	dsldevice.lan	PcsCompu_7e:8d:ba	ARP	60 192.168.122.1 is at 08:00:27:9d:0d:a4
6	4.761717	192.168.122.245	dsldevice.lan	DNS	86 Standard query 0x11b7 PTR 1.122.168.192.in-addr.arpa
7	4.802036	dsldevice.lan	192.168.122.245	DNS	132 Standard query response 0x11b7 PTR 1.122.168.192.in-addr.arpa PTR gns3vm PTR gns3vm.local

Figura 57. Wireshark ARP Spoofing parte 1

La vera e propria alterazione avviene nel terzo frame dove la macchina che esegue Bettercap [9] si presenta con il suo indirizzo MAC, oltre a quello della macchina attaccata .



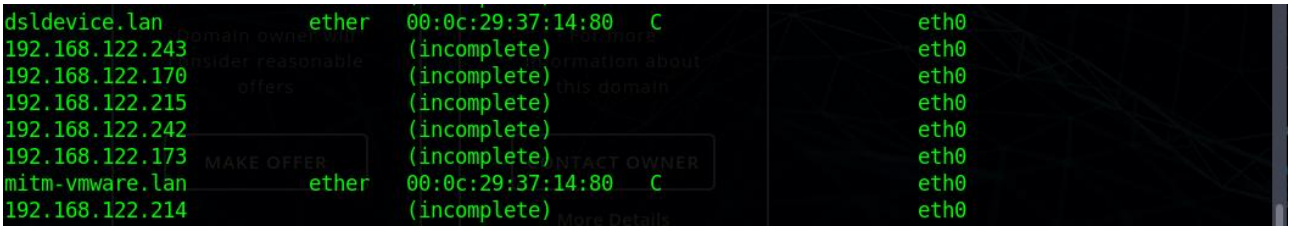
```
> Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface -, id 0
> Ethernet II, Src: dsldevice.lan (08:00:27:9d:0d:a4), Dst: PcsCompu_7e:8d:ba (08:00:27:7e:8d:ba)
> Address Resolution Protocol (reply)
v [Duplicate IP address detected for 192.168.122.1 (08:00:27:9d:0d:a4) - also in use by 52:54:00:a3:19:01 (frame 2)]
  v [Frame showing earlier use of IP address: 2]
    v [Expert Info (Warning/Sequence): Duplicate IP address configured (192.168.122.1)]
        [Duplicate IP address configured (192.168.122.1)]
        [Severity level: Warning]
        [Group: Sequence]
    [Seconds since earlier frame seen: 3]
```

Figura 58. Wireshark ARP Spoofing parte 2

Nella figura sopra espongo il contenuto del frame numero 3 nella quale l'attaccante si presenta alla vittima come il default gateway.

In questo modo ho 2 indirizzi MAC associati allo stesso IP. Questo fa sì che, senza generare ulteriori pacchetti con altrettanti numeri di sequenza e acknowledgment, abbia lo stesso pacchetto che riceverà la vittima senza alterare la comunicazione esistente.

Posso verificare il successo di questa alterazione della tabella ARP anche dalla macchina vittima, utilizzando l'utility arp inclusa nei net-tools **[10]**.



```
dsldevice.lan            ether   00:0c:29:37:14:80   C                     eth0
192.168.122.243                  (incomplete)                              eth0
192.168.122.170                  (incomplete)                              eth0
192.168.122.215                  (incomplete)                              eth0
192.168.122.242                  (incomplete)                              eth0
192.168.122.173                  (incomplete)                              eth0
mitm-vmware.lan          ether   00:0c:29:37:14:80   C                     eth0
192.168.122.214                  (incomplete)                              eth0
```

Figura 59. Tabella ARP vittima

Possiamo vedere come la macchina attaccante abbia lo stesso indirizzo MAC del default gateway.

8. Conclusioni

In conclusione volevamo esprimere qualche giudizio personale sugli strumenti utilizzati.

Riguardo Chameleon [1] abbiamo apprezzato la sua suddivisione in “contenitori” dentro al quale troviamo incapsulati i vari servizi. Un altro elemento che abbiamo apprezzato particolarmente è stata la dashboard ovvero Grafana [4], nella quale posso vedere una timeline degli attacchi in maniera davvero molto chiara.

In merito invece la configurazione di PostreSQL non abbiamo apprezzato la mancanza di persistenza che non ci permette di andare ad analizzare i dati dopo un riavvio dell’applicazione.

In merito a THC-Hydra **[8]** ci è sembrato da subito di semplice utilizzo e non troppo energivoro dal punto di visita delle risorse.

Invece nei confronti di Bettercap [9], non lo abbiamo visto come un progetto molto maturo, il che significa che è instabile e molto spesso lo si è visto dai sistemi di prevenzione.

9. Sviluppi futuri

Il passo successivo della nostra dimostrazione riguarda l’implementazione della rete realizzata su GNS3 [12] in maniere fisica.

L’honeypot potrebbe essere implementato con un altro software magari con il supporto alla persistenza dei dati e un sistema di notifiche in grado di avvertimi di un attacco.

Riguardo l’attacco verso l’honeypot pensavamo di sostituire la tipologia di attacco.

Ovvero un sistema più intelligente che non si limiti semplicemente a leggere un file ed inviare tutte le password presenti al suo interno, ma che riconosca il tipo più adeguato di password/hash da inviare al server.

Un’ altro cosa da migliorare è Bettercap [9], in modo che sia in grado di eseguire lo sniffing su di un maggior numero di protocolli.

In fine vorremmo consigliare di utilizzare hardware più moderno di quello impiegato durante i test, per agevolare la procedura di creazione delle macchine virtuali e la loro esecuzione su GNS3 [12].

Bibliografia

[1] Chameleon: https://github.com/qeeqbox/chameleon

[2] DNS: https://www.iana.org/

[3] NodeJS: https://nodejs.org/it/

[4] Grafana: https://grafana.com/

[5] PostgreSQL: https://www.postgresql.org/

[6] Docker: https://www.docker.com/

[7] ParrotOS: https://www.parrotsec.org/

[8] THC-Hydra: https://github.com/vanhauser-thc/thc-hydra

[9] Bettercap: https://www.bettercap.org/

[10] Net-tools: https://sourceforge.net/projects/net-tools/

[11] Wireshark: https://www.wireshark.org/

[12] GNS3: https://www.gns3.com/

[13] VirtualBox: https://www.virtualbox.org/

[14] VMware: https://www.vmware.com/

Elenco delle figure