



Università
di Catania

RELAZIONE PROGETTO

MitB (Difesa)

Petronio Petronio Davide

Matricola: 1000044165

Indice

1. Introduzione.....	3
1.1 Obiettivo del progetto	3
1.2 Rilevanza degli attacchi MitB	3
2. Man in the Browser.....	4
2.1 Definizione	4
2.2 Funzionamento.....	4
2.3 Infezione	5
2.4 Differenza rispetto al Man-in-the-Middle.....	5
2.5 Impatti	6
3. Strategie Difensive	7
3.1 Strategie preventive	7
3.2 Verifica Out-of-Band	8
3.3 Content Security Policy	9
3.4 Monitoraggio del DOM.....	9
3.5 Analisi del traffico di rete.....	10
4. Demo.....	11
4.1 Progettazione della demo.....	11
4.2 Componenti della demo.....	11
4.2.1 Estensione Totally Benign	11
4.2.2 Server Web.....	12
4.2.3 Bot Telegram	13
4.2.4 Server Proxy	13
4.3 Risultati della demo	13
5. Conclusioni	15
5.1 Efficacia delle contromisure adottate	15
5.2 Limitazioni emerse.....	16

1. Introduzione

1.1 Obiettivo del progetto

L'obiettivo di questo progetto è dare una descrizione di cosa sia un attacco ***Man in the browser*** e fornire delle dimostrazioni di come è possibile **difendersi in caso di attacchi** di questo genere. Al fine di dare una dimostrazione reale di ciò, si procederà ad allegare una demo che implementa le difese di cui discuteremo.

1.2 Rilevanza degli attacchi MitB

Gli attacchi *Man-in-the-Browser* rappresentano una delle minacce più insidiose nell'ambito della sicurezza informatica moderna. A differenza di altri attacchi molto più visibili e rumorosi, un **MitB agisce in modo silenzioso** e trasparente per l'utente, intercettando e modificando le comunicazioni tra browser e applicazioni web **senza che il traffico venga intercettato a livello di rete**. Questo rende il rilevamento particolarmente difficile, poiché anche connessioni cifrate (HTTPS) possono essere compromesse una volta che il malware è attivo all'interno del browser stesso.

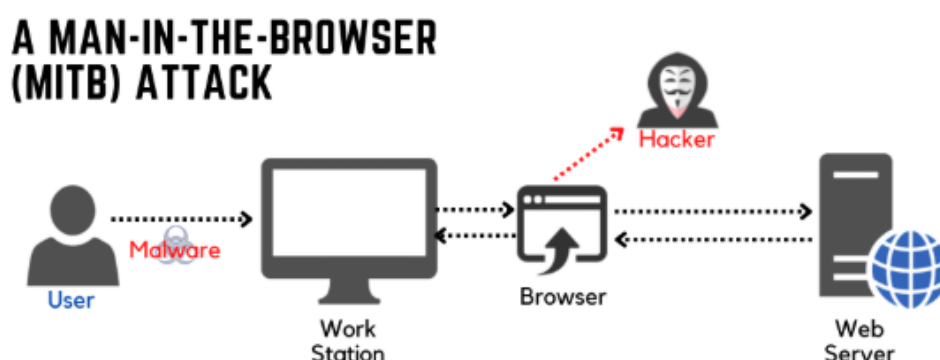
La rilevanza di questi attacchi è particolarmente alta nei contesti in cui vengono effettuate **operazioni sensibili**, come ad esempio l'home banking e l'autenticazione ai servizi online. Il fatto che un attaccante possa **alterare i dati inseriti dall'utente o visualizzati sullo schermo** permette di realizzare frodi finanziarie sofisticate, spesso senza che né l'utente né il server se ne rendano conto.

Inoltre, il crescente utilizzo di **estensioni del browser** renderà ancora più facile l'attuazione di questi tipi di attacchi ampliandone di molto la portata e non limitandosi ad un singolo individuo.

2. Man in the Browser

2.1 Definizione

L'attacco man in the browser è un attacco che, tramite un trojan, **infetta il browser** sfruttando dei privilegi che gli permettono di **accedere ai contenuti del browser** e, quindi, di **intercettare e modificare** tutto ciò che fa l'utente nella pagina senza che questo si accorga di nulla di anomalo.



Esistono diversi esempi reali di malware man-in-the-browser, come **SpyEye**, **Zeus**, **Torpig**, **URLZone** e **Silentbanker**. La maggior parte di questi erano dei malware che prendevano di mira specifiche banche per intercettare credenziali di login o effettuare bonifici falsificati.

2.2 Funzionamento

Il primo esempio noto di attacco *Man-in-the-Browser* risale al **2005**, quando **Augusto Paes de Barros** dimostrò come fosse possibile compromettere **Internet Explorer** sfruttando i **Browser Helper Objects (BHO)**, un meccanismo nativamente supportato dal browser per consentire l'estensione delle sue funzionalità.

L'attacco si basava su un **trojan** mascherato da strumento legittimo (ad esempio un lettore di file PDF), che una volta installato come BHO, era in grado di **intercettare, manipolare o registrare le attività dell'utente all'interno del browser**, agendo quindi come un agente invisibile e persistente.

Con l'evoluzione dei browser e dei modelli di sicurezza, i **BHO sono stati deprecati** e, ad oggi, non sono più supportati nei browser moderni.

Tuttavia, il concetto di estendibilità del browser non è scomparso ma è stato sostituito da un sistema più moderno e standardizzato basato su **estensioni** supportate da tutti i principali browser. Queste estensioni utilizzano **HTML, CSS e JavaScript** per aggiungere funzionalità all'esperienza di navigazione e possono avere accesso ai contenuti delle pagine web tramite apposite **API**.

2.3 Infezione

Nel panorama attuale, gli attacchi *Man-in-the-Browser* vengono spesso realizzati attraverso **estensioni del browser trojanizzate**. Queste estensioni, apparentemente legittime e innocue, **nascondono funzionalità malevole** che compromettono la privacy dell'utente una volta installate.

Sebbene i principali store di estensioni prevedano **meccanismi di controllo e revisione**, è importante sottolineare che le estensioni sono **in continuo aggiornamento** e non è garantito che ogni aggiornamento venga sottoposto a un'analisi approfondita, aprendo la possibilità a **modifiche dannose introdotte successivamente alla pubblicazione iniziale**.

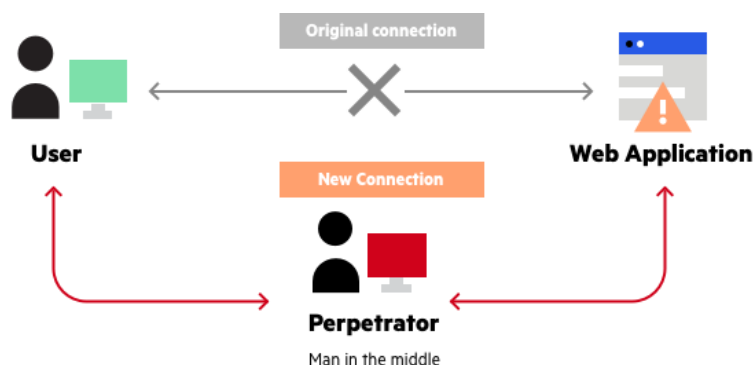
Le estensioni dispongono di un **livello di accesso molto elevato** all'interno del browser: possono **leggere, modificare e riscrivere** i contenuti delle pagine visitate, **intercettare input** utente e **accedere ai dati** delle sessioni. Gli elevati permessi di cui sono dotate consentono loro di implementare **funzionalità utili** come il blocco della pubblicità, la modalità scura o il salvataggio automatico delle credenziali.

Tuttavia, gli stessi privilegi possono essere sfruttati per **intercettare informazioni sensibili, manipolare dati** in tempo reale o **esfiltrare credenziali**.

2.4 Differenza rispetto al Man-in-the-Middle

L'attacco *Man-in-the-Browser* (MitB) condivide alcune caratteristiche concettuali con il più noto attacco *Man-in-the-Middle* (MitM), ma si distingue in modo significativo per quanto riguarda il **livello di esecuzione**.

Nel classico scenario *Man-in-the-Middle*, due entità legittime (ad esempio, A e B) comunicano tra loro attraverso una rete, mentre un terzo attore malevolo (C) si interpone tra i due, **intercettando, monitorando o modificando il traffico di rete** senza che le parti coinvolte se ne rendano conto. Questo tipo di attacco agisce quindi a **livello di rete** ed è, in genere, ostacolato da protocolli di cifratura come **HTTPS/TLS**, i quali garantiscono l'integrità e la riservatezza del canale di comunicazione.



Al contrario, l'attacco *Man-in-the-Browser* opera a un livello più alto, ovvero a **livello applicativo**. In questo scenario, è il **browser stesso ad essere compromesso** da un malware che intercetta o altera i dati **prima** che vengano cifrati e inviati sulla rete. In questo modo, il malware è in grado di **bypassare completamente i meccanismi di cifratura** in quanto agisce **prima che la protezione venga applicata**.

Questo conferisce all'attacco MitB un vantaggio significativo: una volta che il browser è stato compromesso, **non sono necessarie ulteriori operazioni per decifrare o intercettare la comunicazione** poiché, il malware, ha accesso diretto alle informazioni nella loro forma originaria comprese credenziali, transazioni bancarie e altri dati sensibili.

2.5 Impatti

Non è difficile comprendere la pericolosità degli attacchi *Man-in-the-Browser*, considerando il livello di controllo che il malware acquisisce sul browser della vittima. Una volta compromesso il browser, l'attaccante può **monitorare in tempo reale tutto ciò che l'utente visualizza o digita**, agendo di fatto come uno

spyware avanzato in grado di violare completamente la **riservatezza** delle informazioni scambiate online.

Uno degli aspetti più critici di questo attacco è la sua capacità di **modificare dinamicamente i dati inseriti o ricevuti dal browser**, compromettendo così anche l'**integrità** delle transazioni. Ad esempio, in un contesto bancario, il malware può alterare l'IBAN del destinatario di un bonifico o modificare l'importo di una transazione, senza che l'utente se ne accorga, presentando a schermo dati falsificati ma apparentemente coerenti.

Oltre alla violazione della **confidenzialità** e dell'**integrità**, l'attacco MitB può anche avere impatti diretti sulla **disponibilità dei servizi**, nel caso in cui venga utilizzato per bloccare o manipolare l'accesso a determinate risorse, inserire contenuti ingannevoli o forzare il download di ulteriori componenti malevoli.

3. Strategie Difensive

3.1 Strategie preventive

Poiché gli attacchi *Man-in-the-Browser* si basano quasi sempre sull'installazione di un **trojan all'interno del browser**, la prevenzione è strettamente legata alla capacità di **bloccare il malware prima che venga installato**.

Una prima linea di difesa efficace è rappresentata dall'uso di **software antivirus e anti-malware aggiornati** in grado di rilevare comportamenti anomali o firme note associate a estensioni malevole o trojan. Allo stesso tempo, è fondamentale **mantenere aggiornato il browser** e tutti i componenti del sistema operativo, così da ridurre al minimo la presenza di vulnerabilità note che potrebbero essere sfruttate per facilitare l'infezione.

Un'altra misura preventiva cruciale è quella di **limitare l'installazione di estensioni** esclusivamente agli **store ufficiali**, i quali applicano (almeno in linea teorica) controlli minimi di sicurezza e policy di revisione. L'installazione di estensioni da fonti esterne o repository non verificati **comporta un rischio significativo**, poiché queste non sono sottoposte ad alcun controllo e potrebbero contenere codice malevolo nascosto.

In contesti aziendali può essere opportuno **disabilitare completamente l'installazione di estensioni** tramite policy di gruppo o strumenti di gestione centralizzata come *Google Workspace*.

È importante sottolineare che, **una volta che il browser è stato infettato**, l'attacco *Man-in-the-Browser* risulta **estremamente difficile da rilevare**. L'utente non percepisce alcuna anomalia anche perché l'estensione potrebbe effettivamente fornire **funzionalità utili o desiderate**, mascherando così la sua vera natura.

3.2 Verifica Out-of-Band

Uno degli attacchi più comuni eseguiti tramite un malware *Man-in-the-Browser* consiste nella **manipolazione dei dati inseriti nei form web**, in particolare durante operazioni finanziarie come i bonifici bancari. In questo scenario, l'utente compila correttamente il modulo di trasferimento fondi, specificando l'IBAN del destinatario legittimo. Tuttavia, **prima che la richiesta venga inviata alla banca**, il trojan agisce a livello del browser, **modificando in tempo reale i dati inseriti** e sostituendo l'IBAN con quello dell'attaccante o di un conto a lui riconducibile.

Dal punto di vista dell'utente, tutto appare normale: la conferma visualizzata riflette i dati originali, mentre l'operazione inviata alla banca contiene informazioni alterate, portando a una **transazione fraudolenta invisibile**.

Per contrastare questi attacchi, molte banche adottano meccanismi di sicurezza avanzati, tra cui la **verifica out-of-band**. Questa tecnica consiste nell'inviare una richiesta di conferma tramite un **canale di comunicazione separato e considerato più sicuro**, tipicamente un'applicazione mobile dedicata. Quando l'utente avvia una transazione dal browser, riceve una notifica sul proprio smartphone, dove viene richiesto di **verificare i dati principali dell'operazione (importo, beneficiario, IBAN)** e autorizzarla in modo esplicito.



Questo approccio si basa sul principio dello **Zero Trust** secondo cui **nessuna richiesta viene considerata affidabile per default**, nemmeno se questa proviene da un dispositivo o un'applicazione apparentemente legittimi. L'utilizzo di canali separati per l'autenticazione e l'autorizzazione rappresenta una **barriera efficace** contro attacchi MitB, in quanto **isola il processo di conferma da eventuali compromissioni del browser**

3.3 Content Security Policy

Un'ulteriore misura difensiva contro i possibili attacchi utilizzati nei *Man-in-the-Browser* è rappresentata dall'implementazione di una **Content Security Policy (CSP)**. Questa è una direttiva che può essere configurata **lato server** per limitare le fonti da cui un'applicazione web è autorizzata a caricare contenuti dinamici, in particolare **script JavaScript**.

In molti casi, i malware MitB operano attraverso **l'iniezione di codice malevolo all'interno delle pagine web**, trasformando l'attacco in una forma di **Cross-Site Scripting (XSS)** eseguita localmente. Tali iniezioni permettono al codice iniettato di intercettare dati sensibili, alterare il comportamento della pagina o reindirizzare l'utente verso risorse esterne controllate dall'attaccante.

Una CSP ben configurata consente di **limitare in modo rigido le fonti da cui gli script possono essere caricati**, bloccando così l'esecuzione di codice **non autorizzato o iniettato dinamicamente**. Ad esempio, è possibile specificare che solo gli script provenienti da domini precisi (come il dominio principale dell'applicazione) siano considerati attendibili, e impedire al browser di interpretare codice inline o caricato da URL esterni.

Questa politica di sicurezza non previene l'infezione del browser né blocca direttamente l'installazione del malware, ma **riduce significativamente la superficie d'attacco** per quanto riguarda la **manipolazione e l'esecuzione di codice all'interno delle pagine web legittime**.

3.4 Monitoraggio del DOM

Un meccanismo difensivo per contrastare gli attacchi *Man-in-the-Browser* consiste nell'**analisi e monitoraggio delle modifiche al Document Object**

Model (DOM) della pagina web durante la sua esecuzione. Questo approccio si basa sull'idea che un malware attivo nel browser, per manipolare dati o iniettare codice, **debba necessariamente interagire con il DOM**, inserendo, rimuovendo o modificando elementi della struttura della pagina.

Attraverso l'uso di **strumenti di integrity checking del DOM**, è possibile rilevare **modifiche sospette o non previste** rispetto allo stato atteso del documento. Tuttavia, è importante sottolineare che si tratta di un meccanismo **reattivo, non preventivo**. Il rilevamento della compromissione arriva solo dopo che questa è già avvenuta anche se risulta comunque utile per mandare all'utente degli alert in quei casi in cui il **malware cerca di sostituire il contenuto della pagina con uno fittizio**.

3.5 Analisi del traffico di rete

L'analisi del traffico di rete non rappresenta una vera e propria misura preventiva contro gli attacchi *Man-in-the-Browser*, ma può costituire un valido strumento di **rilevamento post-infezione**, soprattutto nei casi in cui il malware tenti di **comunicare con l'esterno** per esfiltrare informazioni raccolte.

Anche se questi attacchi sono progettati per operare in maniera invisibile all'utente, **qualsiasi tentativo di invio dati verso un server remoto genera inevitabilmente traffico di rete**, che può essere intercettato e analizzato. L'utilizzo di strumenti di **network packet inspection** come *Wireshark*, o di soluzioni avanzate di **Endpoint Detection and Response (EDR)**, consente di identificare **pattern di comunicazione anomali**, richieste verso **domini sconosciuti** o traffico sospetto proveniente dal browser stesso.

Questa tecnica è particolarmente utile nei contesti aziendali dove i flussi di rete sono costantemente monitorati e possono essere correlati con l'attività degli utenti o delle applicazioni per evidenziare comportamenti insoliti. In presenza di traffico cifrato, sebbene non sia possibile accedere direttamente ai contenuti trasmessi, **la frequenza, la destinazione e il volume dei pacchetti** possono comunque costituire indicatori affidabili di compromissione.

4. Demo

4.1 Progettazione della demo

L'obiettivo della demo è quello di valutare l'efficacia di alcune delle misure di sicurezza trattate precedentemente, simulando un attacco *Man-in-the-Browser*. A tal fine, si è deciso di progettare e utilizzare, all'interno di un ambiente **sandbox isolato**, una **estensione malevola** sviluppata ad hoc per condurre varie tipologie di attacco contro un'applicazione web eseguita su un **server locale**.

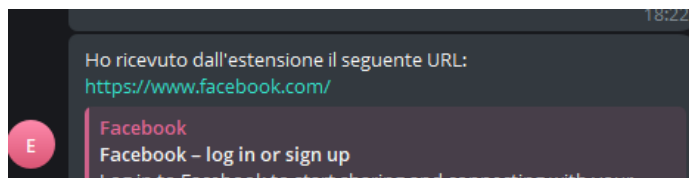
Il comportamento dell'estensione include operazioni tipiche di un attacco MitB, come la **manipolazione dei contenuti del DOM**, **raccolta di credenziali** e **invio di dati verso un server controllato dall'attaccante**. Il server locale, a sua volta, è configurato per implementare alcune delle **strategie difensive discusse in precedenza**, come la **Content Security Policy**, il **monitoraggio delle modifiche al DOM** e **verifiche out-of-band**.

4.2 Componenti della demo

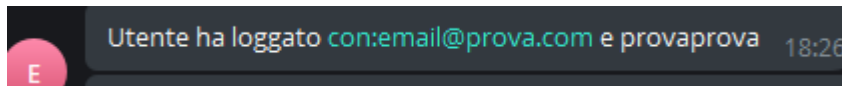
4.2.1 Estensione Totally Benign

Come primo componente di questa demo troviamo l'estensione "*Totally Benign*". Questa estensione malevola è scritta in Javascript e segue lo standard delle estensioni di Chrome *Manifest v3*. L'estensione potrà effettuare i seguenti attacchi:

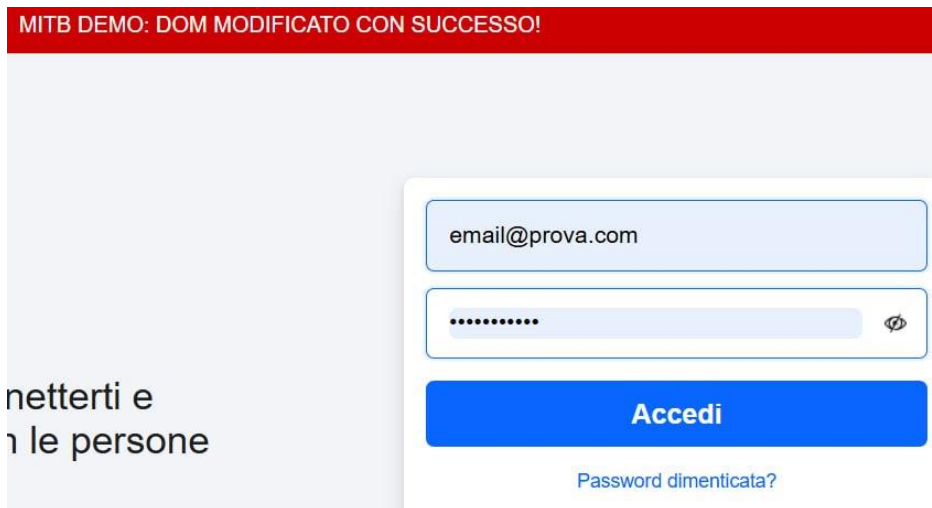
1. **Spyware URL:** Ogni volta che l'utente accederà ad un nuovo sito mediante il browser l'estensione si occuperà di inviare l'URL ad un bot Telegram che è controllato dall'attaccante.



2. **Intercettazione di e-mail e password:** Ad ogni invio di form che comprendono campi di tipo e-mail e password, i valori di questi vengono inviati al bot Telegram controllato dall'attaccante.



3. **Iniezione di codice Javascript e contenuti nel DOM:** In ogni pagina visitata si cercherà di iniettare del codice Javascript e HTML che contengono semplicemente un console.log e un banner all’inizio della pagina.



4. **Modifica dei parametri di bonifici:** Quando l’utente cercherà di effettuare un bonifico per mezzo del sito web creato, i campi del form verranno modificati per far sì che il destinatario del bonifico sia sempre l’attaccante.

4.2.2 Server Web

Questo server web realizzato tramite spring boot simulerà un sito bancario in cui sarà possibile visualizzare la lista degli utenti e il relativo saldo, la lista di tutte le transazioni effettuate e metterà a disposizione una pagina dedicata all’invio dei bonifici.

Implementerà inoltre le seguenti difese:

1. **CSP:** Il CSP cercherà di bloccare qualunque tentativo di esecuzione di codice Javascript esterno a quello del server stesso.
2. **Verifica del DOM:** Tramite uno script Javascript verificherà se sono state apportate modifiche al DOM.
3. **Verifica Out-of-band:** Una volta inviata la richiesta di bonifico, lo stato di quest’ultimo sarà impostato a “In attesa” e, nel mentre, sarà inviata una richiesta di conferma all’utente mediante il bot ufficiale di Telegram della

banca. L'utente potrà quindi verificare importo e destinatario del bonifico e potrà confermarlo mediante l'apposito comando.

4.2.3 Bot Telegram

I due bot Telegram **simuleranno la comunicazione dall'estensione verso l'attaccante e dalla banca verso i suoi clienti** secondo le modalità descritte precedentemente. Entrambi i bot saranno collegati ad un servizio chiamato ngrok, questo permetterà di **collegare i bot Telegram ad un servizio HTTPS online** che reindirizzerà il traffico al server locale che hosta i due bot.

4.2.4 Server Proxy

Al fine di **contrastare le funzionalità da Spyware** dell'estensione malevola, è stato implementato uno script in Python che simuli il comportamento di un firewall Proxy o di un EDR. In particolare, è stata utilizzata la libreria **mitmproxy per creare un proxy locale** che intercetta tutto il traffico HTTP/HTTPS generato dal browser.

Questa soluzione si è rivelata parecchio utile poiché **elimina la complessità dell'analisi di pacchetti a livello di rete** che richiederebbe la risoluzione DNS per ogni indirizzo IP coinvolto. Il proxy, una volta rilevata un'elevata frequenza di richieste verso domini non presenti in una whitelist, blocca automaticamente il traffico verso quest'ultimi simulando una reazione realistica dei sistemi di difesa.

Questa rappresenta una misura di sicurezza realistica **applicata** generalmente **in contesti aziendali** per cui il **traffico verrà gestito attraverso un proxy** gestito da un firewall o da un EDR con funzionalità di monitoraggio e blocco del traffico anomalo.

4.3 Risultati della demo

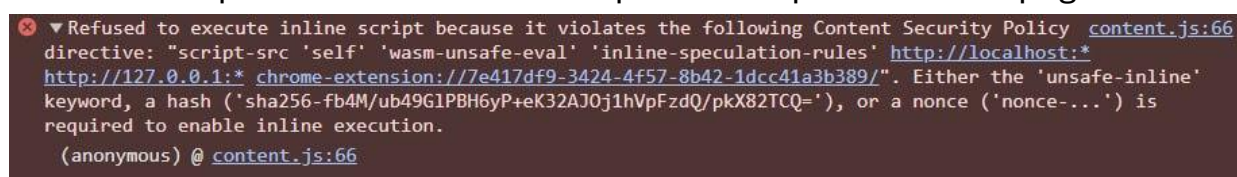
La fase di test ha confermato **l'efficacia delle contromisure implementate contro i diversi attacchi** effettuati dall'estensione *Totally Benign*. Le funzionalità dell'estensione sono state testate e si osservati i seguenti risultati:

1. Iniezione di codice Javascript

L'estensione tentava di inserire un banner e di eseguire codice JavaScript arbitrario all'interno delle pagine web visitate dall'utente. Tuttavia, la **Content Security Policy (CSP)** configurata sul server ha efficacemente

bloccato l'esecuzione di script non autorizzati, impedendo l'iniezione attiva e visibile del codice malevolo.

In aggiunta a ciò, si è scoperto che il browser **Google Chrome, per impostazione predefinita, limita l'uso di script inline**. Questo, unito alla CSP, rappresenta una **protezione ulteriore** contro l'iniezione di codice, riducendo la probabilità che l'attacco possa compromettere la pagina.

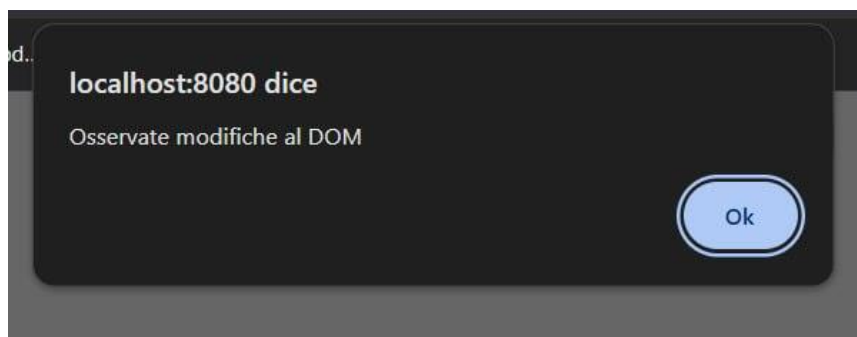


```

❌ ▼ Refused to execute inline script because it violates the following Content Security Policy content.js:66
directive: "script-src 'self' 'wasm-unsafe-eval' 'inline-speculation-rules' http://localhost:*
http://127.0.0.1:* chrome-extension://7e417df9-3424-4f57-8b42-1dcc41a3b389/'. Either the 'unsafe-inline'
keyword, a hash ('sha256-fb4M/ub49G1PBH6yP+eK32AJ0j1hVpFzdQ/pkX82TCQ='), or a nonce ('nonce-...') is
required to enable inline execution.
(anonymous) @ content.js:66
    
```

2. Modifiche al DOM

Il **sistema di rilevamento delle modifiche al DOM** ha funzionato correttamente, segnalando la presenza di elementi non previsti all'interno della pagina. In seguito all'iniezione di contenuti da parte dell'estensione, lo script di monitoraggio ha generato un **avviso (alert)** per notificare all'utente che la struttura della pagina potrebbe essere stata compromessa, suggerendo quindi di effettuare delle verifiche prima di proseguire con l'interazione.

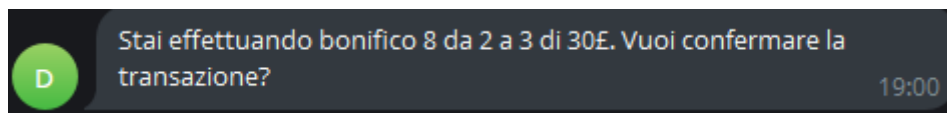


3. Modifica dei parametri di bonifici

Nel tentativo di simulare una tipica azione fraudolenta, l'estensione ha alterato i dati inseriti nel **form per l'esecuzione di bonifici**, sostituendo l'IBAN originale del destinatario con uno controllato dall'attaccante. Tuttavia, la presenza della **verifica out-of-band** ha reso inefficace l'attacco: una volta inviata la richiesta, il server ha richiesto una **conferma esplicita tramite l'app Telegram**, indicando i dettagli reali della transazione.

Questo ha permesso all'utente di **notare immediatamente la**

discrepanza tra i dati originariamente inseriti e quelli presenti nella richiesta finale, bloccando così l'esecuzione dell'operazione fraudolenta. Si conferma quindi l'efficacia del meccanismo **out-of-band** come **barriera affidabile contro la compromissione dell'integrità dei dati**, anche nel caso di un browser già infetto.



4. Spyware URL

Per quanto riguarda le funzionalità da spyware, queste sono state correttamente **rilevate dal server proxy**, che le ha **bloccate classificandole come traffico sospetto**, dopo aver intercettato numerose richieste verso un dominio non presente nella whitelist nell'arco di un solo minuto.

PROBLEMI	TERMINALE	OUTPUT	CONSOLE DI DEBUG	PORTE			
>>21:48:31	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fhypixel.net%2Fdownloads%2F	200	[no content]	399ms	
21:48:35	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fwww.facebook.com%2F	200	[no content]	720ms	
21:48:38	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FCiao	200	[no content]	256ms	
21:48:41	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FLingua_	200	[no content]	224ms	
21:48:43	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FLingua_	200	[no content]	231ms	
21:48:47	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FLingua_	403	text/plain	39b	7ms
21:48:51	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FRepubbl	403	text/plain	44b	7ms
21:48:56	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FRepubbl	403	text/plain	44b	8ms
21:48:57	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FRepubbl	403	text/plain	44b	16ms
21:48:57	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2Findex.php%2F	403	text/plain	44b	18ms
21:49:00	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2Findex.php%2F	403	text/plain	44b	8ms
21:49:02	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FPagina_	403	text/plain	44b	8ms
21:49:07	HTTPS GET	...ngrok-free.app	/evilbot?urlPage=https%3A%2F%2Fit.wikipedia.org%2Fwiki%2FTricolo	403	text/plain	44b	5ms

5. Conclusioni

5.1 Efficacia delle contromisure adottate

Come emerso dall'analisi e dalla demo realizzata, **le contromisure implementate si sono dimostrate efficaci** nel contrastare gli attacchi condotti dall'estensione malevola. Queste contromisure sono in realtà **soluzioni di sicurezza generali** che trovano impiego in vari scenari reali.

In particolare, la **verifica out-of-band** (o, più in generale, l'autenticazione a due fattori) rappresenta oggi uno **standard di sicurezza largamente adottato**, specialmente nei contesti ad alta sensibilità come l'online banking o l'accesso a servizi online. Il principio è quello di **non fidarsi mai completamente del canale primario di comunicazione** e richiedere una **conferma tramite un canale considerato sicuro**.

Allo stesso modo, l'adozione della **Content Security Policy (CSP)** è ormai prassi comune nella maggior parte dei siti web moderni. Questo meccanismo fornisce una **barriera robusta contro l'esecuzione di codice non autorizzato**, riducendo significativamente la superficie d'attacco disponibile per estensioni o script malevoli.

Per quanto riguarda il server proxy, questo si è dimostrato **efficace nell'identificare le richieste dirette verso domini non presenti nella whitelist**, in particolare verso ngrok-free.app, che è il dominio utilizzato per l'esfiltrazione dei dati. Una volta **rilevato un numero elevato di richieste** verso tale dominio, lo script **interviene intercettando e bloccando la comunicazione**, restituendo una risposta HTTP 403 al client.

È importante sottolineare che il **meccanismo di blocco non si attiva immediatamente**, ma solo al **superamento di una soglia configurata** (nella demo, 5 richieste in 60 secondi), il che **rende il sistema potenzialmente vulnerabile a tecniche di evasione**. Un'estensione malevola, ad esempio, potrebbe **aggirare il blocco inviando i dati esfiltrati a intervalli regolari** e diluiti nel tempo, **riducendo il traffico** e impedendone il rilevamento.

Tuttavia, in contesti reali, è plausibile che un sistema di difesa più rigoroso adotti un approccio **whitelist-based**, consentendo le **connessioni esclusivamente verso domini considerati attendibili** e bloccando tutto il traffico verso destinazioni non autorizzate.

5.2 Limitazioni emerse

Bloccare questo tipo di malware si è rivelato particolarmente difficile. Una volta installata, l'estensione malevola **agisce in modo silenzioso**, senza manifestare comportamenti evidenti che possano insospettire l'utente, soprattutto quando include **funzionalità legittime** che mascherano quelle dannose.

In particolare, l'individuazione automatica delle attività di tipo *spyware* come l'invio di URL o dati di input verso server esterni risulta difficile, poiché **il traffico generato può sembrare lecito o indistinguibile da quello prodotto da funzionalità legittime**. L'adozione di strumenti avanzati per l'**analisi del traffico di rete**, come sistemi EDR o IDS/IPS, può aiutare nella rilevazione, ma questi non

sono esenti dal generare **falsi positivi e falsi negativi**, rendendo il blocco parecchio complesso.

Fortunatamente, ad oggi, questi attacchi sono **relativamente rari** grazie ai **rigorosi controlli** applicati dai principali browser sugli store ufficiali e a una **maggiore consapevolezza da parte degli utenti**. In questo contesto, la **formazione e l'educazione dell'utente finale** restano una delle contromisure più efficaci: evitare di installare estensioni da fonti non attendibili o sconosciute è una pratica semplice ma fondamentale per ridurre il rischio.