Microsoft

Microsoft Security Intelligence Report

Volume 20 | July through December, 2015

PLATINUM: Targeted attacks in South and Southeast Asia

Authors

Charlie Anthe
Cloud and Enterprise Security

Nir Ben Zvi
Enterprise and Cloud Group

Patti Chrzan
Microsoft Digital Crimes Unit

Bulent Egilmez
Office 365 - Information Protection

Elia Florio
Windows Defender Labs

Chad Foster
Bing

Roger Grimes
Microsoft IT

Paul Henry
Wadeware LLC

Beth Jester
Windows Defender

Jeff Jones
Corporate Communications

Dana Kaufman
Azure Active Directory Team

Nasos Kladakis
Azure Active Directory Team

Daniel Kondratyuk
Azure Active Directory Team

Andrea Lelli
Windows Defender Labs

Geoff McDonald
Windows Defender Labs

Michael McLaughlin
Identity Services

Nam Ng
Enterprise Cybersecurity Group

Niall O'Sullivan
Microsoft Digital Crimes Unit

Daryl Pecelj
Microsoft IT Information Security and Risk Management

Anthony Penta
Safety Platform

Ina Ragragio
Windows and Devices Group

Tim Rains
Commercial Communications

Paul Rebriy
Bing

Stefan Sellmer
Windows Defender Labs

Mark Simos
Enterprise Cybersecurity Group

Vikram Thakur
Windows Defender Labs

Alex Weinert
Azure Active Directory Team

Terry Zink
Office 365 - Information Protection

Contributors

Joey Caparas
Windows Defender Labs

Satomi Hayakawa
CSS Japan Security Response Team

Ben Hope
Corporate Communications

Yurika Kakiuchi
CSS Japan Security Response Team

Russ McRee
Windows and Devices Group

Dolcita Montemayor
Windows Defender Labs

Wendi Okun
Corporate, External, and Legal Affairs

Laura A.Robinson
Microsoft IT

Jasmine Sesso
Windows Defender Labs

Norie Tamura
CSS Japan Security Response Team

Henk van Roest
CSS Security Response Team

Pete Voss
Corporate, External, and Legal Affairs

Steve Wacker
Wadeware LLC

Iaan Whiltshire
Windows Defender Labs

Table of contents

About this report

The *Microsoft Security Intelligence Report (SIR)* focuses on software vulnerabilities, software vulnerability exploits, malware, and unwanted software. Past reports and related resources are available for download at www.microsoft.com/sir. We hope that readers find the data, insights, and guidance provided in this report useful in helping them protect their organizations, software, and users.

Reporting period

This volume of the *Microsoft Security Intelligence Report* focuses on the third and fourth quarters of 2015, with trend data for the last several quarters presented on a quarterly basis. Because vulnerability disclosures can be highly inconsistent from quarter to quarter and often occur disproportionately at certain times of the year, statistics about vulnerability disclosures are presented on a half-yearly basis.

Throughout the report, half-yearly and quarterly time periods are referenced using the *n*H*yy* or *n*Q*yy* formats, in which *yy* indicates the calendar year and *n* indicates the half or quarter. For example, 1H15 represents the first half of 2015 (January 1 through June 30), and 4Q14 represents the fourth quarter of 2014 (October 1 through December 31). To avoid confusion, please note the reporting period or periods being referenced when considering the statistics in this report.

Conventions

This report uses the Microsoft Malware Protection Center (MMPC) naming standard for families and variants of malware. For information about this standard, see "Appendix A: Theat naming conventions" on page 147 of the full report. In this report, any threat or group of threats that share a common unique base name is considered a family for the sake of presentation. This consideration includes threats that may not otherwise be considered families according to common industry practices, such as generic detections. For the purposes of this report, a threat is defined as a malware or unwanted software family or variant that is detected by the Microsoft Malware Protection Engine.

Foreword

We've been publishing threat intelligence reports for our customers, partners and the industry for 10 years now. During that time, we've published over *12,500 pages of threat intelligence*, 100+ blog posts, many videos, and delivered thousands of customer briefings all over the world. Over the years, the feedback from customers on the value of the intelligence and guidance that we've published in the Microsoft Security Intelligence Report has been nothing short of overwhelming.

In the last few years, things have changed dramatically in the threat landscape, our visibility into it, and the speed at which we can make adjustments to help protect customers. The cloud has been a security game changer and it's becoming more powerful every day.

A few of the CISOs I have talked to still aren't leveraging cloud services to help them protect their organization. Their current on-premises security strategy has them investing in SIEMs to get improved visibility into their IT environment. This doesn't provide them with the intelligence they want on the threats that other organizations have had to face, so they augment their data by procuring multiple third party threat intelligence feeds. The hope is that combining all of this data will enable the organization to better protect, detect and respond to threats.

This approach has certainly benefited many organizations. But security teams know it has challenges. Not all threat intelligence feeds are equal; some data sets are stale. It can be hard to find meaningful threats in large data sets. More data can make this even harder. Attracting and retaining security talent to analyze this data is an industry-wide challenge. If organizations can't identify meaningful threats and take action in real time, the result can be more like a history lesson than it is helpful.

This is where the Microsoft cloud can help. Informed by trillions of signals from billions of sources, Microsoft creates an intelligent security graph that helps protect endpoints, better detect attacks and accelerate our response. The intelligent security graph is powered by inputs we receive across our endpoints, consumer services, commercial services and on-premises technologies.

Every day our machine learning systems process more than 10 terabytes of data, including information on over 13 billion logins from hundreds of millions of Microsoft Account users and Azure Active Directory accounts. We've included new data in this report that provides insight into how the Microsoft cloud uses this massive data and machine learning to literally detect and prevent over a million attacks every day.

The Microsoft cloud has the scale, the threat intelligence, and the security capabilities that CISOs are looking for. If you haven't evaluated or looked at our cloud services in a while, it's time to check out some of the new security capabilities. Start with Azure Security Center, Azure Active Directory Identity Protection, and Microsoft Cloud App Security. You won't be disappointed.

In addition, you'll see from some of the data in this report that Windows 10 has been providing superior protection compared to older operating systems.

I hope you find the 20th volume of the *Microsoft Security Intelligence Report* valuable.

Tim Rains
Director, Security
Microsoft

How to use this report

The *Microsoft Security Intelligence Report* has been released twice a year since 2006. Each volume is based upon data collected from millions of computers all over the world, which not only provides valuable insights on the worldwide threat landscape, both at home and at work, but also provides detailed information about threat profiles faced by computer users in more than a hundred individual countries and regions.

To get the most out of each volume, Microsoft recommends the following:

Read

Each volume of the report consists of several parts. The primary report typically consists of a worldwide threat assessment, one or more feature articles, guidance for mitigating risk, and some supplemental information. A summary of the key findings in the report can be downloaded and reviewed separately from the full report; it highlights a number of facts and subjects that are likely to be of particular interest to readers. The regional threat assessment, available for download and in interactive form at www.microsoft.com/security/sir/threat, provides individual summaries of threat statistics and security trends for more than 100 countries and regions worldwide.

Reading the volume in its entirety will provide readers with the most benefit and context, but the report is designed to provide value in small doses as well. Take a few minutes to review the summary information to find the information that will be of most interest to you and your organization. Consult the table of contents and the index to learn more about particular topics of interest.

Share

Microsoft also encourages readers to share each released volume, or its download link, with co-workers, peers, and friends with similar interests. The *Microsoft Security Intelligence Report* is written to be useful and accessible to a wide range of audiences. Each volume contains thousands of hours of research disseminated in easy to understand language, with advanced technical jargon kept to a minimum. Each section and article is written and reviewed to provide the most value for the time it takes to read.

Assess your own risk

Reading about the threats and risks that affect different types of environments presents a good opportunity to assess your own risks. Not every computer and entity faces the same risk from all threats. Assess your own risks and determine which topics and information can help you to best defend against the most significant risks.

The volume and scope of threats facing the typical organization make it important to prioritize. The greatest risk to any computer or organization is posed by currently and recently active threats. Pay attention to the threats that have most commonly affected your region or industry, focusing particularly on the most common successful attacks in the wild that cause the most problems. Give less consideration to very rare or theoretical-only attacks, unless your computers are at particular risk for such threats.

Educate

Microsoft strives to make this report one of the most valuable sources of threat and mitigation information that you can read and share. We encourage you to use the *Microsoft Security Intelligence Report* as a guide to educate your employees, friends, and families about security-related topics.

Anyone, including a business, may link, point to, or re-use articles in the *Microsoft Security Intelligence Report* for informational purposes, provided the material is not used for publication or sale outside of your company and you comply with the following terms: You must not alter the materials in any way. You must provide a reference to the URL at which the materials were originally found. You must include the Microsoft copyright notice followed by "Used with permission from Microsoft Corporation." Please see Use of Microsoft Copyrighted Content for further information.

Ask questions

Contact your local Microsoft representative with any questions you have about the topics and facts presented in this report. We hope that each volume provides a good educational summary and helps promote dialog between people trying to best secure their computing devices. Thank you for trusting Microsoft to be your partner in the fight against malware, hackers, and other security threats.

Featured intelligence

PLATINUM: Targeted attacks in South and Southeast Asia

Microsoft proactively monitors the threat landscape for emerging threats. Part of this job involves keeping tabs on targeted activity groups, which are often the first ones to introduce new exploits and techniques that are later used widely by other attackers. The feature article "STRONTIUM: A profile of a persistent and motivated adversary," on page 3 of *Microsoft Security Intelligence Report, Volume 19 (January–June 2015)*, chronicled the activities of one such group that attracted interest because of its aggressive, persistent tactics and techniques as well as its repeated use of new zero-day exploits to attack its targets.

This section describes the history, behavior, and tactics of a newly discovered targeted activity group, which Microsoft has code-named PLATINUM. Microsoft is sharing some of the information it has gathered on this group in the hope that it will raise awareness of the group's activities and help organizations take immediate advantage of available mitigations that can significantly reduce the risks they face from this and similar groups.

Adversary profile

PLATINUM has been targeting its victims since at least as early as 2009, and may have been active for several years prior. Its activities are distinctly different not only from those typically seen in untargeted attacks, but from many targeted attacks as well. A large share of targeted attacks can be characterized as opportunistic: the activity group changes its target profiles and attack geographies based on geopolitical seasons, and may attack institutions all over the world. Like many such groups, PLATINUM seeks to steal sensitive intellectual property related to government interests, but its range of preferred targets is consistently limited to specific governmental organizations, defense institutes, intelligence agencies, diplomatic institutions, and telecommunication providers in South and Southeast Asia. The group's persistent use of *spear phishing* tactics (phishing attempts aimed at specific individuals) and access to previously undiscovered zero-day exploits have made it a highly resilient threat.

After researching PLATINUM, Microsoft has identified the following key characteristics of the group and its activities:

- PLATINUM has conducted several cyber espionage campaigns since at least 2009.
- PLATINUM focuses on a small number of campaigns per year, which reduces the risk of detection and helps the group stay unnoticed and focused for a longer period of time.
- PLATINUM has focused on targets associated with governments and related organizations in South and Southeast Asia.

PLATINUM has been targeting its victims since at least as early as 2009.

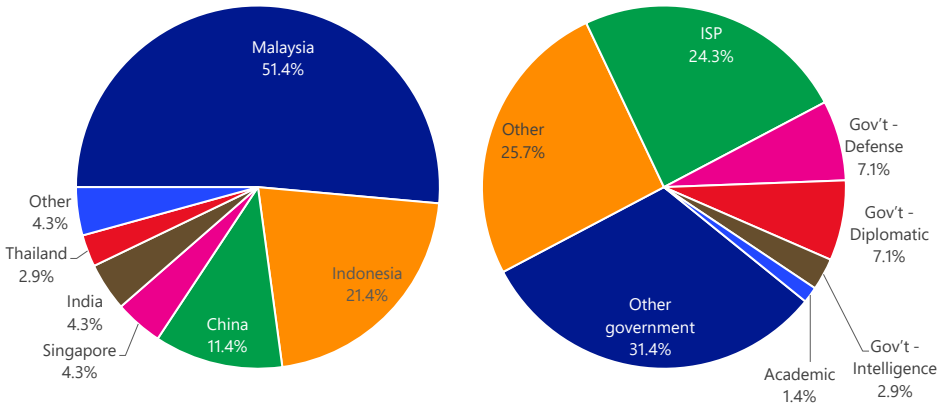
- PLATINUM has used multiple unpatched vulnerabilities in zero-day exploits against its victims.
- Spear phishing is the group's main method of infecting targeted users' computers.
- PLATINUM makes a concerted effort to hide their infection tracks, by self-deleting malicious components, or by using server side logic in "one shot mode" where remotely hosted malicious components are only allowed to load once
- PLATINUM often spear phishes its targets at their non-official or private email accounts, to gain access to the intended organization's network.
- PLATINUM uses custom-developed malicious tools and has the resources to update these tools often to avoid being detected.
- PLATINUM configures its backdoor malware to restrict its activities to victims' working hours, in an attempt to disguise post-infection network activity within normal user traffic.
- PLATINUM does not conduct its espionage activity to engage in direct financial gain, but instead uses stolen information for indirect economic advantages.
- In some cases, the combination of these mechanisms—use of undisclosed zero-day exploits, custom malware that is not used elsewhere, PLATINUM's skill in covering its tracks, and others—has enabled the group to compromise targets for several years without being detected.

Targeted activity groups are skilled at covering their tracks and evading detection, and it can be very difficult to definitively associate an activity group with a specific nation-state or group of individuals. Attackers could be patriotic groups, opportunistic cyber units, state-sponsored hackers, or intelligence agents. Although PLATINUM could belong to any one of the aforementioned

categories, the group shows traits of being well funded, organized, and focused on information that would be of most use to government bodies.

Methods of attack

Figure 1. Known victims attacked by PLATINUM since 2009, by country/region (left) and type of institution (right)



Although PLATINUM primarily targets its intended victims using spear phishing, some data indicates the group's usage of drive-by attacks against vulnerable browser-plugins. The group's methods for performing reconnaissance to determine whom to pursue remains unknown, and the number of victims targeted at each affected institution is consistently very small. In some cases, the victims were targeted at their non-official email addresses, demonstrating that the scope of PLATINUM's research capabilities is fairly extensive. For the initial infection, PLATINUM typically lures its victims by sending malicious documents that contain exploits for vulnerabilities in various software programs, with links or remotely loaded components (images or scripts or templates) that are delivered to targets only once. The group has made concerted efforts toward designing their initial spear-phishes in a manner that only delivers the final payload to the intended victim. The group is known to have used a number of zero-day exploits, for which no security update was available at the time of transmission, in these attempts. (All have subsequently been addressed by security updates from the affected vendors.)

Figure 2. A typical lure document sent by PLATINUM to a prospective victim

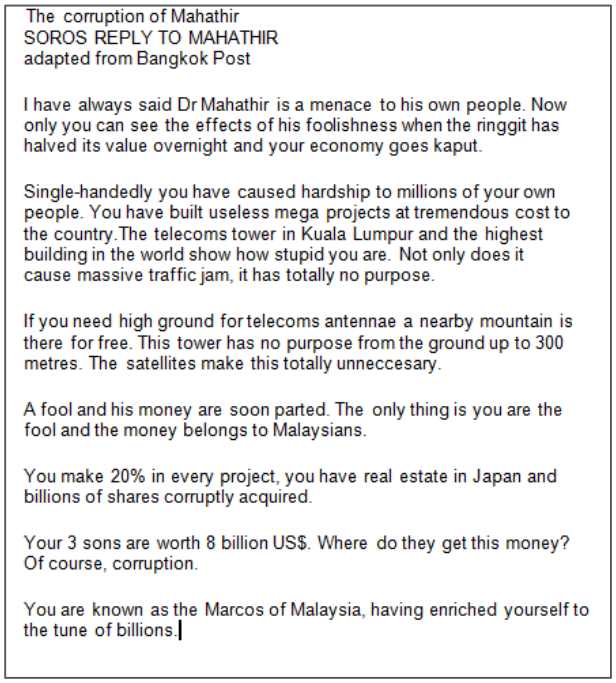
Lure documents are typically given topical names that may be of interest to the recipient. Such lures often address controversial subjects or offer provocative opinions, in an effort to incite the reader into opening them. Figure 3 shows a sample of such titles.

Figure 3. Example document titles used by PLATINUM to deliver exploits

SHA1	Filename
e9f900b5d01320ccd4990fd322a459d709d43e4b	Gambar gambar Rumah Gay Didiet Prabowo di Sentul Bogor.doc
9a4e82ba371cd2fedea0b889c879daee7a01e1b1	The real reason Prabowo wants to be President.doc
92a3ece981bb5e0a3ee4277f08236c1d38b54053	Malaysia a victim of American irregular warfare ops.doc
0bc08dca86bd95f43ccc78ef4b27d81f28b4b769	Tu Vi Nam Tan Mao 2011.doc
f4af574124e9020ef3d0a7be9f1e42c2261e97e6	Indians having fun.doc

These documents were sent to intended victims in Vietnam, Indonesia, India, and Malaysia, and the filenames contain references to cities, politicians, and current events in those locations. The oldest confirmed PLATINUM exploit was named "The corruption of Mahathir," a document that was transmitted in 2009 referencing the former prime minister of Malaysia, Mahathir Mohamad.

Figure 4. The oldest confirmed lure document sent by PLATINUM, in 2009



The corruption of Mahathir
SOROS REPLY TO MAHATHIR
adapted from Bangkok Post

I have always said Dr Mahathir is a menace to his own people. Now only you can see the effects of his foolishness when the ringgit has halved its value overnight and your economy goes kaput.

Single-handedly you have caused hardship to millions of your own people. You have built useless mega projects at tremendous cost to the country.The telecoms tower in Kuala Lumpur and the highest building in the world show how stupid you are. Not only does it cause massive traffic jam, it has totally no purpose.

If you need high ground for telecoms antennae a nearby mountain is there for free. This tower has no purpose from the ground up to 300 metres. The satellites make this totally unneccesary.

A fool and his money are soon parted. The only thing is you are the fool and the money belongs to Malaysians.

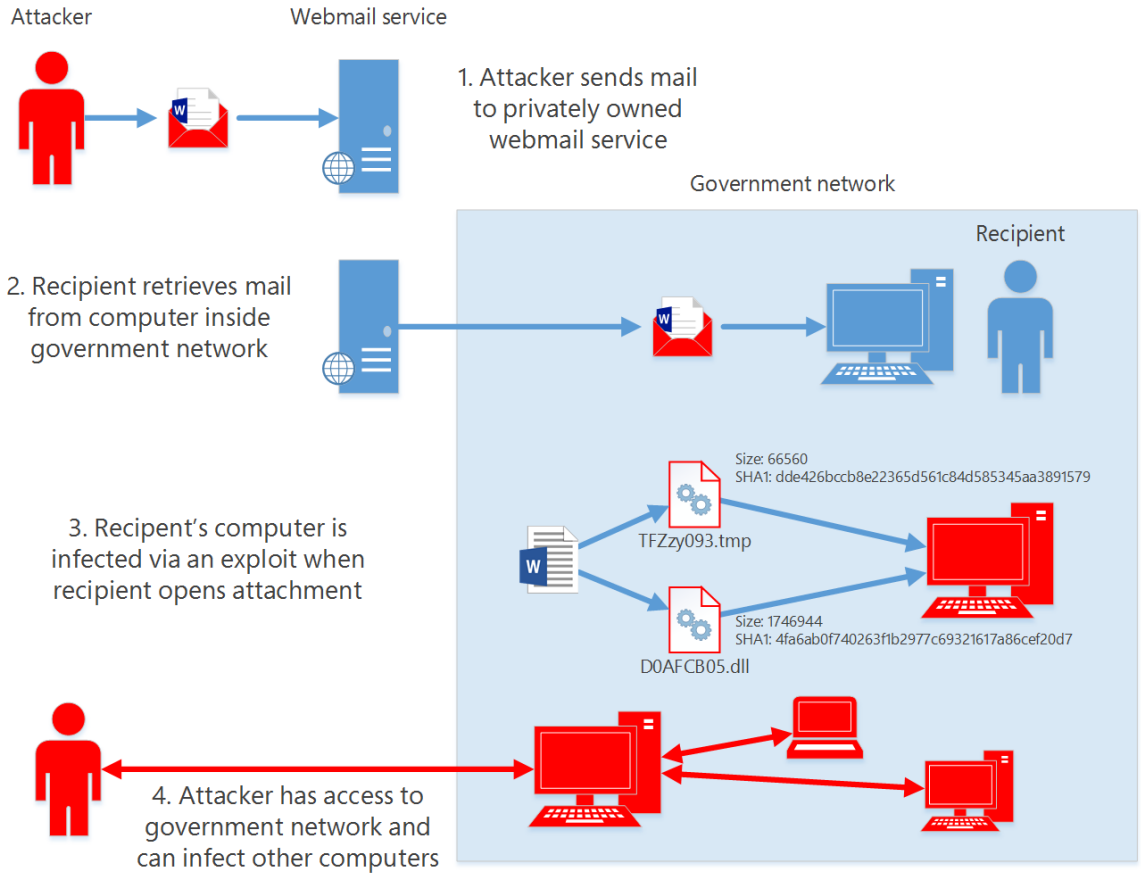
You make 20% in every project, you have real estate in Japan and billions of shares corruptly acquired.

Your 3 sons are worth 8 billion US$. Where do they get this money? Of course, corruption.

You are known as the Marcos of Malaysia, having enriched yourself to the tune of billions.

PLATINUM's recent activities remain focused on tactics such as these. In February 2016, PLATINUM was observed using a legitimate website dedicated to news about the Indian government as an infection vector. This site, which is not associated with the Indian government itself, also provides a free email service for its users, giving them email addresses with the site's own domain name. PLATINUM sent spear phishing messages to users of the service, which included some Indian government officials. After infecting an unsuspecting user this way, the attackers had complete control of the user's computer and used it as a stepping stone into the official network to which the user belonged.

Figure 5. PLATINUM used a private webmail service to infect a government network



PLATINUM's approach toward exploiting vulnerabilities varies between campaigns. In one case from 2013, the target was sent a malicious document through a spear phishing email message.[1] The document, when opened, used an embedded ActiveX control to download a JavaScript file from a remote site that used a previously unknown vulnerability in some versions of Windows (later designated CVE-2013-7331) to read information about the browser's installed components.[2]

[1] Microsoft thanks Google for identifying and reporting this attack.
[2] Microsoft issued Security Bulletin MS14-052 in September 2014 to address the issue. CVE-2013-7331 has never affected Windows 10.

Figure 6. Malicious Word 2003 files used by PLATINUM to deliver CVE-2013-7331

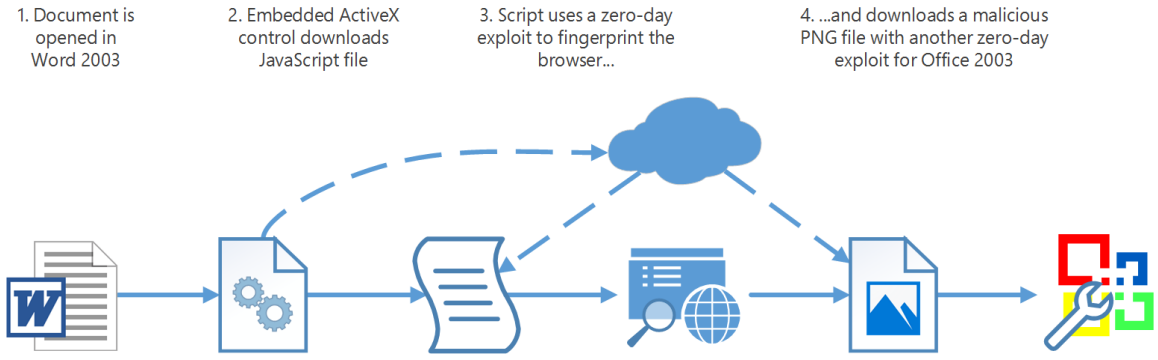
Filename	SHA1	URL for PNG Exploit
Gerakan Anti SBY II.doc	1bdc1a0bc995c1beb363b11b71c14324be8577c9	mister.nofrillspace.com/users/web8_dice/4226/space.gif
Tu_Vi_Nam_Tan_Mao_2011.doc	2a33542038a85db4911d7b846573f6b251e16b2d	intent.nofrillspace.com/users/web11_focus/3807/space.gif
Wikileaks Indonesia.doc	d6a795e839f51c1a5aeabf5c10664936ebbef8ea	mister.nofrillspace.com/users/web8_dice/3791/space.gif
Top 11 Aerial Surveillance Devices.doc	f362feedc046899a78c4480c32dda4ea82a3e8c0	intent.nofrillspace.com/users/web11_focus/4307/space.gif
SEMBOYAN_1.doc	f751cdfaef99c6184f45a563f3d81ff1ada25565	www.police28122011.0fees.net/pages/013/space.gif

Figure 7. Malicious JavaScript used by PLATINUM to perform fingerprinting on a victim's browser

```
header = "res://";
footer = '/';
0;
jav_arr = new Array();
arr[0] = "C:\\progra~1\\Java\\jre7\\bin\\npjpi170_05.dll";
arr[1] = "C:\\progra~1\\Java\\jre7\\bin\\npjpi170_06.dll";
```

While fingerprinting the versions of the browser plug-ins, the script loads a remotely hosted malicious PNG file that exploited another previously unknown vulnerability (designated CVE-2013-1331), which affected Microsoft Office 2003 SP3.[3] Exploiting the vulnerability resulted in memory corruption, which allowed the attacker to execute remote code on the computer.

Figure 8. An exploit mechanism used by PLATINUM

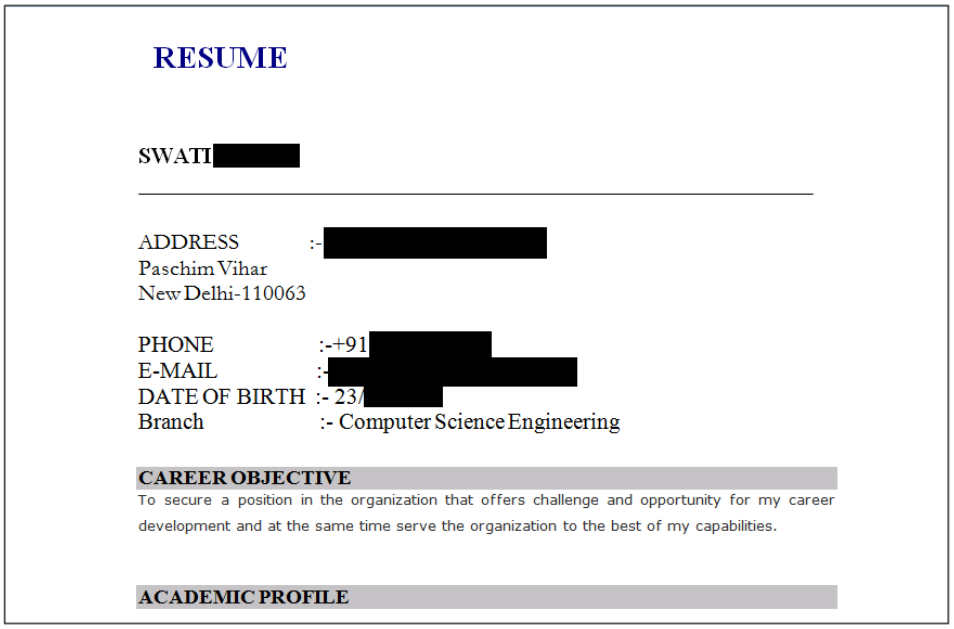


Another combination of lure documents with the aforementioned embedded ActiveX control was seen along with a Dipsind executable named as pp4x322.dll during a different attack. The unique name of this executable indicated a possible DLL side-loading vulnerability also being used by PLATINUM against PowerPoint 2007.

[3] Microsoft issued Security Bulletin MS13-051 in June 2013 to address the issue.

In another case from August 2015, Microsoft investigated a malicious document (named Resume.docx) that had been uploaded to the VirusTotal malware analysis service.[4] The person who submitted the file did so through an IP address based in India, suggesting that the person or their organization had been targeted by the spear phish document.

Figure 9. A malicious Word document used by PLATINUM to target a victim


RESUME

SWATI[redacted]

ADDRESS :-[redacted]
Paschim Vihar
New Delhi-110063

PHONE :-+91[redacted]
E-MAIL :-[redacted]
DATE OF BIRTH :- 23/[redacted]
Branch :- Computer Science Engineering

CAREER OBJECTIVE

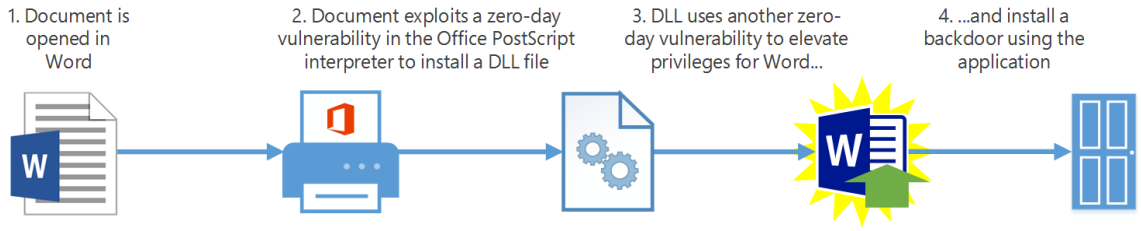
To secure a position in the organization that offers challenge and opportunity for my career development and at the same time serve the organization to the best of my capabilities.

ACADEMIC PROFILE

When the document was opened in Word, it exploited a previously unknown vulnerability in the Microsoft Office PostScript interpreter (designated CVE-2015-2545) that enabled it to execute the attacker's code and drop an attacker-generated malicious DLL onto the computer.[5] The DLL exploited another previously unknown vulnerability (designated CVE-2015-2546) in the Windows kernel, which enabled it to elevate privileges for the Word executable and subsequently install a backdoor through the application.[6] Researching this attack and the malware used therein led Microsoft to discover other instances of PLATINUM attacking users in India around August 2015.

[4] Microsoft thanks FireEye for identifying and reporting this attack.
[5] Microsoft issued Security Bulletin MS15-099 in September 2015 to address the issue. Windows 10 is not affected by the exploit used in this case due to built-in mitigations.
[6] Microsoft issued Security Bulletin MS15-097 in September 2015 to address the issue.

Figure 10. Another exploit mechanism used by PLATINUM



In total, PLATINUM made use of four zero-day exploits during these two attack campaigns (two remote code execution bugs, one privilege escalation, and one information disclosure), showing an ability to spend a non-trivial amount of resources to either acquire professionally written zero-day exploits from unknown markets or research and utilize the zero-day exploits themselves. In both these campaigns, the activity group included remote triggers to deactivate exploitation, with an attempt to conceal the vulnerability and prevent analysis of the attack. The resources required to research and deploy multiple zero-day exploits within the same attack campaign are considerable. Such activity requires a significant amount of investment in research and development, along with the discipline to ensure that the exploits are not used until the appropriate time, and that no one involved with the project leaks them to other parties.

> PLATINUM used four zero-day exploits during these two campaigns.

Technical details

After gaining access to a victim's computer, PLATINUM installs its own custom-built malware to communicate with the compromised computer, issue commands, and move laterally through the network. The broad collection of backdoors and tools, and the differences between them, suggest the involvement of multiple teams or vendors in the development process. This section describes some of the tools used by the group.

Dipsind

PLATINUM uses a number of different custom-developed backdoors to communicate with infected computers. The lack of any significant evidence of shared code between any of these backdoor families is another clue as to the scope of the resources on which the activity group is able to draw, and the precautions the group is willing and able to take to avoid losing its ability to conduct its espionage operations.

The group's most frequently used backdoors belong to a malware family that Microsoft has designated Dipsind, although some variants are detected under different names. Multiple Dipsind variants have been identified, all of which are believed to be used exclusively by PLATINUM.

The first variant, Win32/Dipsind.A!dha, is a lightweight application providing backdoor access to remote attackers. It can be customized for every victim to ensure that it remains undetected in targeted networks. It supports a small set of instructions that allow the attacker to perform basic functions, such as uploading or downloading files and spawning remote shells.

Figure 11. Sample configuration file for Win32/Dipsind.A

```
RunTimeFolderUser =   […….]
RunTimeFolderAdmin = [%commonfiles%]\System\Network
Provision\
RunTimeFileNameDll = xmlprv.dll
ServiceKeyName = xmlprv
ServiceKeyNameDll = xmlprv

cmdpathAdmin = [%commonfiles%]\System\Network
Provision\Library\
cmdname = wscntfy.exe

slpSite1 = scienceweek.scieron.com
pollSlpTime = 10

pollcommandsite1 = [ AES encrypted ]
pollcommandTime = 10

officeStart = 00:00, officeEnd = 23:59
sat = 1, sun = 1

checkurl1 = http://www.google.com/
```

Each Dipsind file contains an embedded encrypted configuration file that acts as a control for the backdoor. This configuration file also includes the initial command and control (C&C) location the Dipsind backdoor uses in addition to the *pollcommandsite* variable, which references a URL where additional backup C&Cs can be polled. Configurable parameters include instructions on where Dipsind should install a copy of cmd.exe for spawning a remote shell, depending on the user's privileges, and the hours during which the backdoor should function and exfiltrate information. This capability allows the backdoor to confine its activities to normal working hours, making its communications harder to distinguish from normal network traffic.

Dipsind has been observed using a combination of IP addresses and domains for its C&C infrastructure. The domains are a mix of registered domains and free subdomains obtained through dynamic DNS providers. Collected data showed that most victim networks allowed unfiltered access to the dynamic DNS hosts. The hosts and domains are hosted on compromised infrastructure based in several different countries, some within academic institutions. In some cases, the backdoors are configured to connect to IP addresses instead of domain names. These factors make it challenging to locate the activity group's infrastructure.

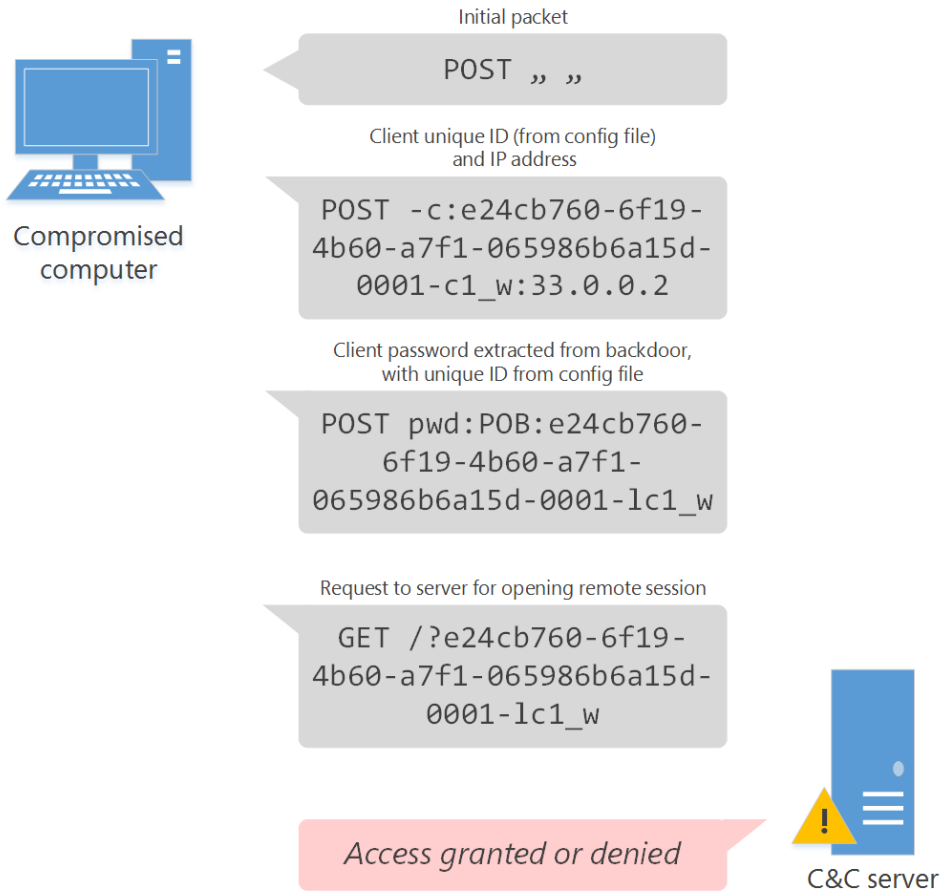
Figure 12 shows a sampling of C&C infrastructure used by PLATINUM between 2009 and 2015.

Figure 12. Some of the domains and addresses used by PLATINUM

Registered domains	Dynamic DNS	Hardcoded IPs
• box62.a-inet.net	• scienceweek.scieron.com	• 200.61.248.8
• eclipse.a-inet.net	• mobileworld.darktech.org	• 209.45.65.163
• joomlastats.a-inet.net	• geocities.efnet.at	• 190.96.47.9
• updates.joomlastats.co.cc	• bpl.blogsite.org	• 192.192.114.1
• server.joomlastats.co.cc	• wiki.servebbs.net	• 61.31.203.98

After Dipsind.A is installed on the victim's computer, it connects to its C&C server for authentication. All network traffic is over HTTP, base64 encoded, with the underlying data encrypted using AES256 in ECB mode. Authentication is a five-step process, as shown in the following figure:

Figure 13. Win32/Dipsind.A initial communication protocol (as decrypted)



Analysis of several samples of this variant show exactly the same AES key (AOPSH03SK09POKSID7FF674PSLI91965) in use since 2009. The initial HTTP POST made by this backdoor appears as "*ud7LDjtsTHe2tWeC8DYo8A***", which translates to a simple whitespace. This sequence makes a simple network indicator usable by defenders.

A second Dipsind variant registers as a Winlogon Event Notify DLL. This backdoor contains a minimized feature list from the original Dipsind variant, and supports a more limited number of commands. It sets the following registry keys in the HKEY_LOCAL_MACHINE hive for persistence and functionality:

- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\Cscdll32\Asynchronous
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\Cscdll32\DllName

- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\Cscdll32\Impersonate
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\Cscdll32\Startup
- SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Notify\Cscdll32\shutdown
- SOFTWARE\Microsoft\Windows\CurrentVersion\Run\cscdll32

At least two additional minor versions of this variant exist, each of which show improvements in command implementation.

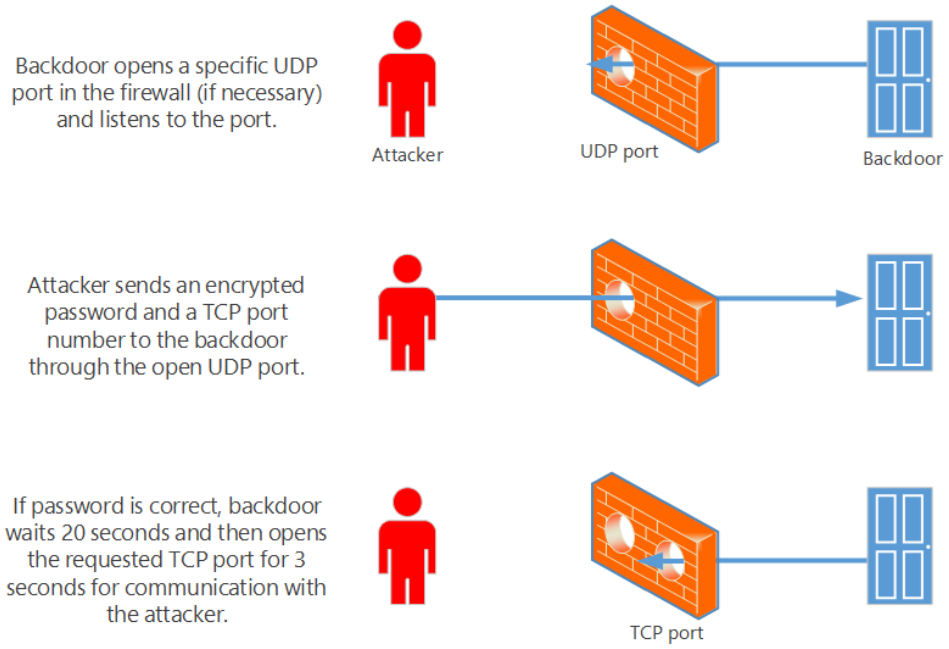
One interesting feature of this variant is the way it implements a mechanism similar to port knocking to allow remote attackers to connect to a compromised computer without leaving any connection open for too long. The sequence of events is as follows:

1. The backdoor is installed via an exploit.
2. The backdoor sets a registry key to open a specific UDP port through the local firewall, if any, and listens to the port for incoming traffic.
3. At a remote location, the attacker executes a tool (called PK2 here, although the actual name of the tool is unknown) using the following parameters:

   Pk2.exe <IP> <UDP Port> <TCP Port> <Password>

   where the IP address is that of the computer with the backdoor, the UDP port is the one specified by the backdoor, and the password is a string encrypted by the tool before being sent.
4. The backdoor receives the UDP packets, and then checks to see if the password is valid.
5. If the password is indeed valid, the backdoor will wait for exactly 20 seconds and only then open the PK2 specified TCP port for a window of 3 seconds.

Figure 14. How the Dipsind knocker component communicates with an attacker



PK2 is also designed to connect to such open TCP ports and act as a console client for issuing commands to the backdoor. When running PK2 as a console client, the attacker needs to re-enter the password to authenticate a second time against the backdoor, and issue commands such as #sz to upload a file and #rz to download a file. During this research, one such collection of tools was obtained that had the password set to "t@ng0p@ss". All communication used by this backdoor and PK2 is encrypted. If a connection from PK2 is not received within the 3-second window, the TCP port is shut and PK2 would need to reinitialize the port-knocking process.

JPIN

In addition to Dipsind and its variants, PLATINUM uses a few other families of custom-built backdoors within its attack toolset. These families of backdoors are significantly different in their capabilities and have completely different code bases. While one family relies on a small number of supported commands and simple shells, the other delves into more convoluted methods of injections, checks, and supported feature sets.

Microsoft researchers refer to one such set of backdoor variants collectively as JPIN, which is the name of a service it uses when installed. JPIN is a comprehensive tool for executing and extracting information from the

compromised computer. There is strong evidence to suggest that the developers of the JPIN and Dipsind code bases were in some way related.

JPIN has its own installer and uninstaller component, which deletes itself when it encounters a version of Windows earlier than Windows XP or finds any of these security-related processes running:

Figure 15. Security-related processes avoided by the JPIN installer

Process	Security product
360tray.exe	360 Safeguard
bdagent.exe	BitDefender
proguard.exe	Process Guard
blackd.exe	BlackICE
blackice.exe	BlackICE
savservice.exe	Sophos Anti-Virus
avp.exe	Kaspersky Anti-Virus
rstray.exe	Rising Anti-virus
cmccore.exe	CMC Antivirus
cmctrayicon.exe	CMC Antivirus
zhudongfangyu.exe	360 Safeguard

After installing the backdoor, the installer deletes itself from the compromised computer.

PLATINUM uses at least three distinct JPIN variants. One variant typically runs with a mutex named hMSVmm and installs itself in the folders %appdata%\Comm\Jpin and %userprofile%\AppData\Resource\Jpin. After it is installed and started, the JPIN service can perform the following tasks, among others:

- Obtain information about the computer, such as operating system version, user name, privileges, disk space, and so on.
- List running services, processes, job IDs, and task IDs.
- Enumerate drives and their types.
- Enumerate registry keys.
- Load a custom keylogger.

- Download files.
- Download and upgrade itself.
- Acquire network information such as DNS, IP, proxies, and so on.
- Exfiltrate information over HTTP GET and POST requests, with the data stored either within the HTTP body or within the URL parameters.
- Lower security settings by tampering with registry keys.
- Inject content into the lsass.exe process, in order to load the keylogger module into lsass and call its exported function.
- Communicate via FTP.
- Send email via SMTP.
- Change permissions on files using the cacls.exe command-line utility.

JPIN can also target mobile suite applications and extract data from them. The backdoor contains code that looks for installed instances of Symbian, BlackBerry, and Windows Phone management applications. If any are found, the backdoor logs sync dates, IMEI data, phone manufacturer and model information, software version date, memory, location, and capacity, among other information.

JPIN can target mobile suite applications and extract data from them.

The second JPIN variant is very similar to the first one. It downloads the backdoor payload from remote locations via the BITS service, using the COM object for BITS. This variant also has its own installer and uninstaller component, which deletes itself when it encounters a version of Windows earlier than Windows XP or finds any of the processes listed in Figure 15 running.

The third known variant does not check for the processes listed in Figure 15. It uses an installer component that includes the backdoor as payload disguised as a bitmap within its resource section. The payload is in an encrypted and compressed form, disguised to avoid any suspicion from security solutions. This variant has been seen installing itself into the following file system paths:

- %appdata%\Java\support
- %appdata%\support
- %userprofile%\AppData\Local\Java\Support

- %userprofile%\AppData\Local\Support

adbupd

Another backdoor used by PLATINUM is very similar to the Dipsind family. It is informally referred to internally at Microsoft as adbupd, which is the name of the service under which it is installed. Salient features of this backdoor include the following:

- It tries to install itself under several different names within the Program Files directory.
- It has the ability to support plug-ins to modularize functionality.
- It contains a copy of the OpenSSL library to support encryption when sending or receiving data.
- It contains functionality to run a copy of cmd.exe.
- The configuration file is very similar to the original Dipsind family.
- This backdoor class uses multiple methods of achieving persistence, one of which is using WMI /MOF compiled scripts, such as the one shown in Figure 16.

Figure 16. WMI script used by the adpupd backdoor to achieve persistence

```
#pragma namespace("\\\\.\\ROOT\\cimv2")
instance of __Win32Provider as $P
{
    Name = "adbupdConsumer";
    ClsId = "{74ba9ce4-fbf1-4097-32b8-34f446f037d8}";
    HostingModel = "LocalSystemHost";
};
instance of __EventConsumerProviderRegistration
{
    Provider = $P;
    ConsumerClassNames = {"adbupdConsumer"};
};
class adbupdConsumer : __EventConsumer
{
    [key] string Mode;
};
instance of adbupdConsumer as $CONSMR
{
    Mode = "persistent";
};
```

```
instance of __EventFilter as $FLT
{
    Name = "adbupdFilter";
    Query = "SELECT * FROM __InstanceCreationEvent WHERE
TargetInstance ISA \"Win32_NTLogEvent\"";
    QueryLanguage = "WQL";
};
instance of __FilterToConsumerBinding as $B
{
    Consumer = $CONSMR;
    Filter = $FLT;
};
```

Keyloggers

The PLATINUM group has written a few different versions of keyloggers that perform their functions in different ways, most likely to take advantage of different weaknesses in victims' computing environments. The keyloggers can be broadly classified into two groups: those that log keystrokes through raw device input, and user mode keyloggers that use Windows hook interfaces to gather information. In particular, this second group also has the capability of dumping users' credentials using the same technique employed by Mimikatz. Both groups can set permissions on specific files to Everyone, and work in tandem with the PLATINUM backdoors.

Hot patcher

One of PLATINUM's most recent and interesting tools is meant to inject code into processes using a variety of injection techniques. In addition to using several publicly known injection methods to perform this task, it also takes advantage of an obscure operating system feature known as *hot patching*.

Hot patching is an operating system-supported feature for installing updates without having to reboot or restart a process. At a high level, hot patching can transparently apply patches to executables and DLLs in actively running processes, which does not happen with traditional methods of code injection such as CreateRemoteThread or WriteProcessMemory. Instead, the kernel is instructed to perform the injection by invoking NtSetSystemInformation (with an appropriate SystemInformationClass) to apply the patch. The information about the patch is delivered via a specially crafted DLL that is loaded into the target process.

The hot patching feature originally shipped with Windows Server 2003 and was used to ship 10 patches to Windows Server 2003. It was removed in Windows 8 and has not been included in subsequent releases of Windows. PLATINUM appears to believe that enough of their targeted users continue to run the earlier versions of Windows to make the technique a useful tool, at least until early 2017 (see page 22).

The technique PLATINUM uses to inject code via hot patching was first documented by security researchers in 2013.[7] Administrator permissions are required for hot patching, and the technique used by PLATINUM does not attempt to evade this requirement through exploitation. Rather, the component's use of the hot patching feature appears to be a way to avoid being detected, because many antivirus solutions monitor non-system processes for the regular injection methods such as CreateRemoteThread. If the tool fails to inject code using hot patching, it reverts to attempting the other more common code injection techniques into common Windows processes, primarily targeting winlogon.exe, lsass.exe, and svchost.exe:

- CreateRemoteThread
- NtQueueApcThread
- RtlCreateUserThread
- NtCreateThreadEx

The hot patching component performs the following steps:

1. It patches the loader with a proper hot patch to treat injected DLLs with execute page permissions. This step is required for DLLs loaded from memory (in an attempt to further conceal the malicious code).

2. The backdoor is injected into svchost using the hot patch API. Patching the loader is done by creating a section named \knowndlls\mstbl.dll. This DLL does not reside on disk, but is rather treated as a cached DLL by the session manager. It then proceeds to write a PE file within that section.

3. The PE file will have one section (.hotp1) with the hot patch header structure. This structure contains all the information necessary to perform the patching

[7] Alex Ionescu, "Hotpatching the Hotpatcher: Stealth File-less DLL Injection," SyScan 2013, https://www.yumpu.com/en/document/view/14255220/alexsyscan13/23.

of function ntdll!LdrpMapViewOfSection, which will cause the loader to treat created sections as PAGE_EXECUTE_READWRITE instead of PAGE_READWRITE. The patch is successfully applied by invoking NtSetSystemInformation.

4. After the memory permission issue is solved, the injector proceeds to inject the malicious DLL into svchost. Again, it creates a (now executable) section named knowndlls\fgrps.dll and invokes NtSetSystemInformation, which causes the final payload to be loaded and executed within the target process (svchost).

5. The malicious hot patching component appears to have an expiration date of January 15, 2017. After that date, the DLL will no longer perform the injection, but rather execute another PLATINUM implant (C:\Program Files\Windows Journal\Templates\Cpl\jnwmon.exe –ua), which may be related to an uninstall routine. (The component has not been observed in use since March 9, 2016, which may indicate that PLATINUM has chosen to stop using it earlier than the configured expiration date.)

Miscellaneous

Finally, the PLATINUM group also uses small single-purpose applications that duplicate some of the functionality of the backdoors. A couple of examples are:

- A stand-alone persistence tool that takes other files as input and ensures persistence across reboots.
- A stand-alone loader that runs another executable. It has some exported functions whose names can be used in DLL files installed as LSA password filters, but such functions are basically empty and there is no known evidence that this tool was ever used in this way. On the whole, this DLL looks like a test, suggesting that the attackers may have researched and possibly implemented variants of their malware that can be installed as LSA password filters.

Exploit (CVE-2015-2545)

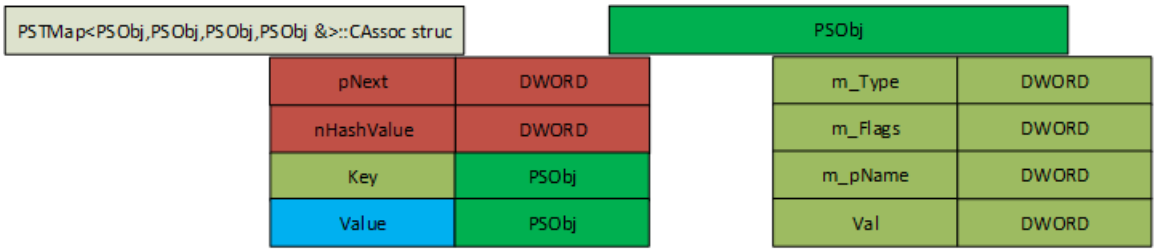
CVE-2015-2545 is a use-after-free vulnerability in the embedded PostScript filter of Microsoft Office.[8] The exploit was crafted in PostScript and is able to bypass

[8] Microsoft issued Security Bulletin MS15-099 in September 2015 to address the issue.

Address Space Layout Randomization (ASLR) and Data Execution Prevention (DEP).

This vulnerability allows the attacker to forge a CAssoc structure, shown in Figure 17, and so also indirectly the PSObjs in the structure. The PostScript interpreter deciphers the value field (Val) based on the type field (m_type), which are under complete control of the attacker. Having developed this technique, the attacker will craft and use a combination of file, string, and integer objects to gain a reliable arbitrary code execution.

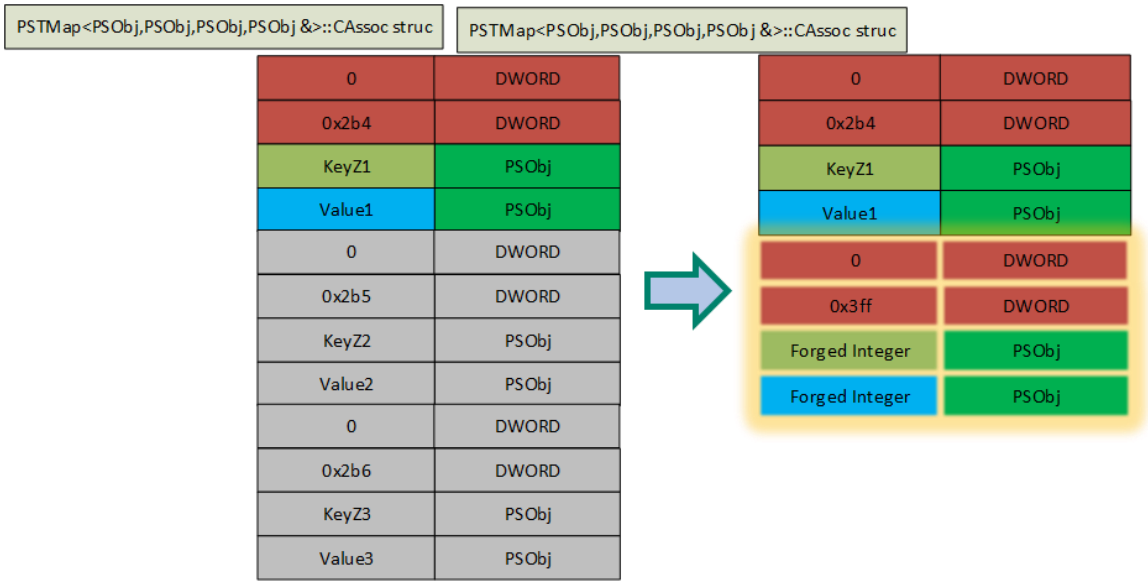
Figure 17. Memory layout of CSssoc structure and its embedded PSObjs



Root cause: The attacker defined in PostScript a dictionary with three elements, which leads to an allocation of three CAssoc structures in PSTMap.

Within a Forall loop, the last two elements are undefined and a string is initialized. The PostScript statement results in a deallocation of the last two CAssoc structures and the string gets allocated in the previously freed memory address. The PostScript-put operand is used to fill the string with data to mimic a CAssoc structure. By setting the hash table index to 0x3ff, the loop will exit because the hash table at that time has a max-size of 0x400. Upon exiting the loop, a reference will be returned to the secondary element, which is the forged structure.

Figure 18. Reusage of deallocated memory by a forged CAssoc structure



Acquire full memory RW access: The described method is used to craft a PSString object in which the length of the string is set to a maximum value. As a result, the exploit can use PostScript methods to search for ROP gadgets to dynamically assemble a ROP shellcode.

Figure 19. Getinterval method of PSString is used to find ROP gadgets

```
bind def / colortone25 {
    colortonee ImageCurve3 458752 getinterval < 94 C3 > search { //xchg eax,esp / ret
        length / offset exch def pop pop
    } {
        pop
    }
```

The purpose of this approach is to call VirtualProtect to set the pages of the second-stage shellcode as executable. As a result, DEP and ASLR are bypassed.

Arbitrary code execution: To redirect code execution to the ROP chain, the exploit crafts a PSFile Object in which the vtable is controlled by the attacker. By calling the bytesavailable method within the PostScript code, arbitrary code execution is achieved.

Identity

Although the exact identity of PLATINUM remains unknown, the technical indicators observed so far can help create a profile of the attacker.

- Usage of multiple backdoors. The different backdoors written by or for the group indicate a considerable investment over time. Research indicates that PLATINUM has used multiple backdoors concurrently at times, which could represent either multiple teams within the activity group performing different campaigns or different versions of the tools being used against varying victim networks.
- Zero-day exploits. PLATINUM has used several zero-day exploits against its victims. Regardless of whether PLATINUM researched these exploits themselves or purchased them from independent researchers, the monetary investment required to collect and deploy zero-day exploits at this level is considerable.
- Victim geography. More often than not, research into targeted attacks shows activity groups becoming opportunistic and attacking topical targets; that is, targets considered valuable based on the geopolitical events of the year. PLATINUM has consistently targeted victims within a small set of countries in South and Southeast Asia. In addition, the victims are consistently associated with a small set of entities that are directly or indirectly connected to governments.

The monetary investment required to collect and deploy zero-day exploits at this level is considerable.

- Tools. Some of the tools used by PLATINUM, such as the port-knocking backdoor, show signs of organized thinking. PLATINUM has developed or commissioned a number of custom tools to provide the group with access to victim resources. This behavior exhibits PLATINUM's ability to adapt to victim networks, which is further evidence of the group's considerable resources for development and maintenance.

  Any of these traits by themselves could be the work of a single resourceful attacker or a small group of like-minded individuals, but the presence of all of them is a clear indication of a well-resourced, focused, and disciplined group of attackers vying for information from government-related entities.

Guidance

PLATINUM is an extremely difficult adversary for targeted organizations to defend against. It possesses a wide range of technical exploitation capabilities, significant resources for researching or purchasing complicated zero-day exploits, the ability to sustain persistence across victim networks for years, and

the manpower to develop and maintain a large number of tools to use within unique victim networks. Their ability to research their victims prior to targeting them, along with the capability to architect exploits that only work once or for a short period of time, makes it very difficult to investigate or track their activities. That said, there are steps that organizations can take to reduce the likelihood of PLATINUM conducting successful attacks against their employees and networks.

- Take advantage of native mitigations built into Windows 10. Newer versions of Windows include critical mitigations that render some of PLATINUM's exploits ineffective when deployed. For example, the summer 2015 attack that used the unusual "resumé" would not have been successful on Windows 10 as-is because of the presence of the Supervisor Mode Execution Prevention (SMEP) mitigation, even without the latest security updates installed. Even if CVE-2015-2546 affected Windows 10, the exploitation would have required much more technical prowess to succeed; ultimately, SMEP makes it more difficult for attackers. The hooking and in-memory patching techniques used by the malicious hot patcher component are also not effective against newer versions of Windows.
- Apply all security updates as soon as they become available. Microsoft deeply researches each security issue, proactively addresses the flaw, and mitigates the attack surface around the affected component(s). For example, one zero-day exploit (CVE-2015-2545) used by PLATINUM was addressed immediately in September 2015. Subsequently, in November, Microsoft also released a proactive security update for the same component that ended up mitigating other exploits surfacing in-the-wild after the first attack. Customers who applied the security updates in November without delay would have been protected against the second wave of exploits. Such measures of hardening the underlying application happen often. MS09-017 is yet another example, in which installation of newly available security updates significantly reduced the attack surface.
- Consider disabling features, such as EPS or macros, in powerful products like Microsoft Office by using Group Policy. Not all organizations find the need to enable all features. For example, in the PLATINUM attack campaign that used CVE-2015-2545, a network in which Office EPS was disabled would not have been affected.
- Enterprise networks should segregate high business impact (HBI) data-holding segments from Internet-connected networks. Sharing of removable

media between these air-gapped networks should be strictly enforced. In the case of PLATINUM, such a network architecture would prevent targeted users from accessing third-party email services and thereby granting attackers access to sensitive segments of the organizational network.

- Conduct enterprise software security awareness training, and build awareness of malware prevention. PLATINUM may have used zero-day flaws to compromise victim computers, but doing so required action by the user, who either clicked a link in an email or opened an attachment to allow the attacker to take control of their computer. Security training can raise awareness and reduce the risk associated with this attack vector.

> Apply all security updates as soon as they become available.

- Institute a strong network firewall and proxy. Many tools used by attackers are not compatible with network proxies. In the case of PLATINUM's version of port knocking, the opening of a UDP port would have been rendered moot if a network firewall was blocking access for inbound packets to the host's open port.
- Enterprise networks should consider blocking certain types of websites that don't serve the interest of the business. PLATINUM makes extensive use of C&Cs that use dynamic DNS hosts. Although such free services can be very useful at a personal level, blocking access to such hosts at a local DNS server can minimize post-compromise activity.
- Prepare your network to be forensically ready, so that you can achieve containment and recovery if a compromise occurs. A forensically ready network that records authentications, password changes, and other significant network events can help identify affected systems quickly.
- Make sure that your organization's Internet-facing assets are always running up-to-date applications and security updates, and that they are regularly audited for suspicious files and activity. A number of researched PLATINUM victims had their public-facing infrastructure compromised through previously unknown flaws.

Detection indicators

Figure 20 consists of detection rules for a number of PLATINUM malware samples to be used with YARA (https://plusvic.github.io/yara/), an open source pattern matching tool for malware detection.

Figure 20. Detection indicators for PLATINUM malware

```
rule Trojan_Win32_PlaSrv : Platinum
{
  meta:
    author = "Microsoft"
    description = "Hotpatching Injector"
    original_sample_sha1 =
"ff7f949da665ba8ce9fb01da357b51415634eaad"
    unpacked_sample_sha1 =
"dff2fee984ba9f5a8f5d97582c83fca4fa1fe131"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $Section_name = ".hotp1"
    $offset_x59 = { C7 80 64 01 00 00 00 00 01 00 }

  condition:
    $Section_name and $offset_x59
}

rule Trojan_Win32_Platual : Platinum
{
  meta:
    author = "Microsoft"
    description = "Installer component"
    original_sample_sha1 =
"e0ac2ae221328313a7eee33e9be0924c46e2beb9"
    unpacked_sample_sha1 =
"ccaf36c2d02c3c5ca24eeeb7b1eae7742a23a86a"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $class_name = "AVCObfuscation"
    $scrambled_dir = { A8 8B B8 E3 B1 D7 FE 85 51 32 3E C0 F1 B7
73 99 }

  condition:
    $class_name and $scrambled_dir
}

rule Trojan_Win32_Plaplex : Platinum
{
  meta:
```

```
    author = "Microsoft"
    description = "Variant of the JPin backdoor"
    original_sample_sha1 =
"ca3bda30a3cdc15afb78e54fa1bbb9300d268d66"
    unpacked_sample_sha1 =
"2fe3c80e98bbb0cf5a0c4da286cd48ec78130a24"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $class_name1 = "AVCObfuscation"
    $class_name2 = "AVCSetiriControl"

  condition:
    $class_name1 and $class_name2
}

rule Trojan_Win32_Dipsind_B : Platinum
{
  meta:
    author = "Microsoft"
    description = "Dipsind Family"
    sample_sha1 = "09e0dfbb5543c708c0dd6a89fd22bbb96dc4ca1c"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $frg1 = {8D 90 04 01 00 00 33 C0 F2 AE F7 D1 2B F9 8B C1 8B F7
8B FA C1 E9 02 F3 A5 8B C8 83 E1 03 F3 A4 8B 4D EC 8B 15 ?? ?? ??
?? 89 91 ?? 07 00 00 }
    $frg2 = {68 A1 86 01 00 C1 E9 02 F3 AB 8B CA 83 E1 03 F3 AA}
    $frg3 = {C0 E8 07 D0 E1 0A C1 8A C8 32 D0 C0 E9 07 D0 E0 0A C8
32 CA 80 F1 63}

  condition:
    $frg1 and $frg2 and $frg3
}
rule Trojan_Win32_PlaKeylog_B : Platinum
{
  meta:
    author = "Microsoft"
    description = "Keylogger component"
    original_sample_sha1 =
"0096a3e0c97b85ca75164f48230ae530c94a2b77"
```

```
    unpacked_sample_sha1 =
"6a1412daaa9bdc553689537df0a004d44f8a45fd"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $hook = {C6 06 FF 46 C6 06 25}
    $dasm_engine = {80 C9 10 88 0E 8A CA 80 E1 07 43 88 56 03 80
F9 05}

  condition:
    $hook and $dasm_engine
}
rule Trojan_Win32_Adupib : Platinum
{
  meta:
    author = "Microsoft"
    description = "Adupib SSL Backdoor"
    original_sample_sha1 =
"d3ad0933e1b114b14c2b3a2c59d7f8a95ea0bcbd"
    unpacked_sample_sha1 =
"a80051d5ae124fd9e5cc03e699dd91c2b373978b"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = "POLL_RATE"
    $str2 = "OP_TIME(end hour)"
    $str3 = "%d:TCP:*:Enabled"
    $str4 = "%s[PwFF_cfg%d]"
    $str5 = "Fake_GetDlgItemTextW: ***value***="

  condition:
    $str1 and $str2 and $str3 and $str4 and $str5
}
rule Trojan_Win32_PlaLsaLog : Platinum
{
  meta:
    author = "Microsoft"
    description = "Loader / possible incomplete LSA Password
Filter"
    original_sample_sha1 =
"fa087986697e4117c394c9a58cb9f316b2d9f7d8"
    unpacked_sample_sha1 =
"29cb81dbe491143b2f8b67beaeae6557d8944ab4"
```

```
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {8A 1C 01 32 DA 88 1C 01 8B 74 24 0C 41 3B CE 7C EF 5B
5F C6 04 01 00 5E 81 C4 04 01 00 00 C3}
    $str2 = "PasswordChangeNotify"

  condition:
    $str1 and $str2
}
rule Trojan_Win32_Plagon : Platinum
{
  meta:
    author = "Microsoft"
    description = "Dipsind variant"
    original_sample_sha1 =
"48b89f61d58b57dba6a0ca857bce97bab636af65"
    unpacked_sample_sha1 =
"6dccf88d89ad7b8611b1bc2e9fb8baea41bdb65a"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"

  strings:
    $str1 = "VPLRXZHTU"
    $str2 = {64 6F 67 32 6A 7E 6C}
    $str3 = "Dqpqftk(Wou\"Isztk)"
    $str4 = "StartThreadAtWinLogon"


  condition:
    $str1 and $str2 and $str3 and $str4
}
rule Trojan_Win32_Plakelog : Platinum
{
  meta:
    author = "Microsoft"
    description = "Raw-input based keylogger"
    original_sample_sha1 =
"3907a9e41df805f912f821a47031164b6636bd04"
    unpacked_sample_sha1 =
"960feeb15a0939ec0b53dcb6815adbf7ac1e7bb2"
    activity_group = "Platinum"
    version = "1.0"
```

```
    last_modified = "2016-04-12"

  strings:
    $str1 = "<0x02>" wide
    $str2 = "[CTR-BRK]" wide
    $str3 = "[/WIN]" wide
    $str4 = {8A 16 8A 18 32 DA 46 88 18 8B 15 08 E6 42 00 40 41 3B
CA 72 EB 5E 5B}

  condition:
    $str1 and $str2 and $str3 and $str4
}
rule Trojan_Win32_Plainst : Platinum
{
  meta:
    author = "Microsoft"
    description = "Installer component"
    original_sample_sha1 =
"99c08d31af211a0e17f92dd312ec7ca2b9469ecb"
    unpacked_sample_sha1 =
"dcb6cf7cf7c8fdfc89656a042f81136bda354ba6"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {66 8B 14 4D 18 50 01 10 8B 45 08 66 33 14 70 46 66 89
54 77 FE 66 83 7C 77 FE 00 75 B7 8B 4D FC 89 41 08 8D 04 36 89 41
0C 89 79 04}
    $str2 = {4b D3 91 49 A1 80 91 42 83 B6 33 28 36 6B 90 97}

  condition:
     $str1 and $str2
}
rule Trojan_Win32_Plagicom : Platinum
{
  meta:
    author = "Microsoft"
    description = "Installer component"
    original_sample_sha1 =
"99dcb148b053f4cef6df5fa1ec5d33971a58bd1e"
    unpacked_sample_sha1 =
"c1c950bc6a2ad67488e675da4dfc8916831239a7"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
```

```
  strings:
    $str1 = {C6 44 24 ?? 68 C6 44 24 ?? 4D C6 44 24 ?? 53 C6 44 24
?? 56 C6 44 24 ?? 00}
    $str2 = "OUEMM/EMM"
    $str3 = {85 C9 7E 08 FE 0C 10 40 3B C1 7C F8 C3}

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Plaklog : Platinum
{
  meta:
    author = "Microsoft"
    description = "Hook-based keylogger"
    original_sample_sha1 =
"831a5a29d47ab85ee3216d4e75f18d93641a9819"
    unpacked_sample_sha1 =
"e18750207ddbd939975466a0e01bd84e75327dda"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"

  strings:
    $str1 = "++[%s^^unknown^^%s]++"
    $str2 = "vtfs43/emm"
    $str3 = {33 C9 39 4C 24 08 7E 10 8B 44 24 04 03 C1 80 00 08 41
3B 4C 24 08 7C F0 C3}

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Plapiio : Platinum
{
  meta:
    author = "Microsoft"
    description = "JPin backdoor"
    original_sample_sha1 =
"3119de80088c52bd8097394092847cd984606c88"
    unpacked_sample_sha1 =
"3acb8fe2a5eb3478b4553907a571b6614eb5455c"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = "ServiceMain"
```

```
    $str2 = "Startup"
    $str3 = {C6 45 ?? 68 C6 45 ?? 4D C6 45 ?? 53 C6 45 ?? 56 C6 45
?? 6D C6 45 ?? 6D}

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Plabit : Platinum
{
  meta:
    author = "Microsoft"
    description = "Installer component"
    sample_sha1 = "6d1169775a552230302131f9385135d385efd166"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {4b D3 91 49 A1 80 91 42 83 B6 33 28 36 6B 90 97}
    $str2 = "GetInstanceW"
    $str3 = {8B D0 83 E2 1F 8A 14 0A 30 14 30 40 3B 44 24 04 72
EE}

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Placisc2 : Platinum
{
  meta:
    author = "Microsoft"
    description = "Dipsind variant"
    original_sample_sha1 =
"bf944eb70a382bd77ee5b47548ea9a4969de0527"
    unpacked_sample_sha1 =
"d807648ddecc4572c7b04405f496d25700e0be6e"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {76 16 8B D0 83 E2 07 8A 4C 14 24 8A 14 18 32 D1 88 14
18 40 3B C7 72 EA }
    $str2 = "VPLRXZHTU"
    $str3 = "%d) Command:%s"
    $str4 = {0D 0A 2D 2D 2D 2D 2D 09 2D 2D 2D 2D 2D 2D 0D 0A}

  condition:
```

```
    $str1 and $str2 and $str3 and $str4
}
rule Trojan_Win32_Placisc3 : Platinum
{
  meta:
    author = "Microsoft"
    description = "Dipsind variant"
    original_sample_sha1 =
"1b542dd0dacfcd4200879221709f5fa9683cdcda"
    unpacked_sample_sha1 =
"bbd4992ee3f3a3267732151636359cf94fb4575d"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {BA 6E 00 00 00 66 89 95 ?? ?? FF FF B8 73 00 00 00 66
89 85 ?? ?? FF FF B9 64 00 00 00 66 89 8D ?? ?? FF FF BA 65 00 00
00 66 89 95 ?? ?? FF FF B8 6C 00 00 00}
    $str2 = "VPLRXZHTU"
    $str3 = {8B 44 24 ?? 8A 04 01 41 32 C2 3B CF 7C F2 88 03}

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Placisc4 : Platinum
{
  meta:
    author = "Microsoft"
    description = "Installer for Dipsind variant"
    original_sample_sha1 =
"3d17828632e8ff1560f6094703ece5433bc69586"
    unpacked_sample_sha1 =
"2abb8e1e9cac24be474e4955c63108ff86d1a034"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = {8D 71 01 8B C6 99 BB 0A 00 00 00 F7 FB 0F BE D2 0F BE
04 39 2B C2 88 04 39 84 C0 74 0A}
    $str2 = {6A 04 68 00 20 00 00 68 00 00 40 00 6A 00 FF D5}
    $str3 = {C6 44 24 ?? 64 C6 44 24 ?? 6F C6 44 24 ?? 67 C6 44 24
?? 32 C6 44 24 ?? 6A}

  condition:
    $str1 and $str2 and $str3
```

```
}
rule Trojan_Win32_Plakpers : Platinum
{
  meta:
    author = "Microsoft"
    description = "Injector / loader component"
    original_sample_sha1 =
"fa083d744d278c6f4865f095cfd2feabee558056"
    unpacked_sample_sha1 =
"3a678b5c9c46b5b87bfcb18306ed50fadfc6372e"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = "MyFileMappingObject"
    $str2 = "[%.3u]  %s  %s  %s [%s:" wide
    $str3 = "%s\\{%s}\\%s" wide

  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Plainst2 : Platinum
{
  meta:
    author = "Microsoft"
    description = "Zc tool"
    original_sample_sha1 =
"3f2ce812c38ff5ac3d813394291a5867e2cddcf2"
    unpacked_sample_sha1 =
"88ff852b1b8077ad5a19cc438afb2402462fbd1a"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = "Connected [%s:%d]..."
    $str2 = "reuse possible: %c"
    $str3 = "] => %d%%\x0a"


  condition:
    $str1 and $str2 and $str3
}
rule Trojan_Win32_Plakpeer : Platinum
{
  meta:
```

```
    author = "Microsoft"
    description = "Zc tool v2"
    original_sample_sha1 =
"2155c20483528377b5e3fde004bb604198463d29"
    unpacked_sample_sha1 =
"dc991ef598825daabd9e70bac92c79154363bab2"
    activity_group = "Platinum"
    version = "1.0"
    last_modified = "2016-04-12"
  strings:
    $str1 = "@@E0020(%d)" wide
    $str2 =
/exit.{0,3}@exit.{0,3}new.{0,3}query.{0,3}rcz.{0,3}scz/ wide
    $str3 = "---###---" wide
    $str4 = "---@@@---" wide


  condition:
    $str1 and $str2 and $str3 and $str4
}
```

One Microsoft Way
Redmond, WA 98052-6399
microsoft.com/security