Computer Networks 57 (2013) 436–450

Contents lists available at SciVerse ScienceDirect

Computer Networks

jou3rnal homepage: www.elsevier.com/locate/comnet

Dissecting SpyEye – Understanding the design of third generation botnets

Aditya K. Sood [a,*], Richard J. Enbody [a,1], Rohit Bansal [b,2]

[a] Department of Computer Science and Engineering, Michigan State University, East Lansing, MI, USA
[b] SecNiche Security Labs, USA

ARTICLE INFO

Article history:
Received 24 November 2011
Received in revised form 2 June 2012
Accepted 29 June 2012
Available online 3 August 2012

Keywords:
Botnets
Malware
Malicious Code
Rootkits
Cybercrime

ABSTRACT

Botnet malware is improving with the latest (3rd) generation exemplified by the SpyEye and Zeus botnets. These botnets are important to understand because they target online financial transactions, primarily with banks. In this paper, we analyze the components from multiple generations of the SpyEye botnet in order to understand both how it works and how it is evolving. SpyEye is a sophisticated piece of malware with a modular design that eases the incorporation of improvements. We will discuss in detail the complete framework of SpyEye botnet consisting of the Bot Development Kit (BDK), the plugin architecture, the backend storage server, the bot design and the web-based Command and Control (C&C) management system. In addition, we also examine the techniques used by SpyEye to steal money.

1. Introduction

Cybercriminals are making the Internet a less affable place, and the increasing monetary losses further degrade the experience. Botnets have become a popular tool for large-scale attacks on financial transactions because of their ability to spread their infections widely. As these attacks have become more lucrative and defenses have improved, the botnet frameworks have become more sophisticated. For these reasons, botnets are important to study.

Botnets are not new, but they are evolving. Botnets can be categorized using a variety of characteristics but communication protocol and motivation are two important metrics that can be used to classify them into generations. The first generation of botnets were controlled using the Internet Relay Chat (IRC) protocol, and were used primarily for conducting Distributed Denial of Service (DDoS) [3] attacks, phishing attacks [4] and stealing login credentials [5] from victims' machines. IRC-based botnets were susceptible to defensive attacks on their command-and-control infrastructure so a new generation (2nd) of botnets was developed based on Peer-2-Peer (P2P) protocols to frustrate those defensive techniques. P2P botnets use distributed client server communication architecture to form a decentralized network that is more difficult to defensively take down. P2P traffic was susceptible to blocking so designers turned to the HTTP protocol since all users expect HTTP access making broad blocking impracticable. This third generation of botnets is exemplified by the SpyEye and Zeus botnets. Third-generation botnets (sometimes called TGBs) are characterized by their economic motivation and design sophistication. Not surprisingly, the boundaries among botnet generations are not well de-

Abbreviations: MAP, Main Admin Panel; FAP, FormGrabber Admin Panel; BC, Backend Connector; BDS, Backend Database Server; BDK, Bot Development Kit; C&C, Command and Control.

* Corresponding author. Address: Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824-1226, USA. Tel.: +1 517 755 9911.

E-mail addresses: soodadit@cse.msu.edu (A.K. Sood), enbody@cse.msu.edu (R.J. Enbody), rb@secniche.org (R. Bansal).

[1] Address: Department of Computer Science and Engineering, 3115 Engineering Building, Michigan State University, East Lansing, MI 48824-1226, USA. Tel.: +1 517 353 3389.

[2] Address: New Delhi, India. Tel.: +91 95600 11494.

1389-1286/$ - see front matter Crown Copyright © 2012 Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.comnet.2012.06.021

fined and there exist hybrid botnets that combine features of multiple generations. The change in motivation over time is striking: initially botnets were used for fun and did little damage, but recent botnets are reaping large rewards by stealing money online from financial institutions.

Botnets are controlled by bot herders (a.k.a. bot masters) using a centralized server called the Command and Control (C&C) server. Bots are the agents designed to compromise the victim's machine, and after successful infection, the zombie machine becomes part of the botnet. Bots are often distributed through drive-by-downloads. [1] using Browser Exploit Packs (BEPs) [6] which are automated exploit serving frameworks. In a drive-by-download attack, a victim visits a compromised web site and that website infects the victim's machine (almost always) without the victim being aware that he/she has been infected. The mechanism is the dropper, a compressed piece of malware that bypasses defenses (such as antivirus engines) to load a bot into the victim's machine. Once on a machine, a bot can tamper with many components of the system and manipulate it for illegitimate purposes. There are multiple stages to the bot infection process, but it is quite successful because it is difficult for many users to understand and defend against it.

This paper investigates SpyEye, a third-generation botnet that has been widely deployed and targets online financial institutions. We have analyzed more than 13 variants of the SpyEye [2] botnet framework to understand its evolution over time. In addition, we have reverse engineered several variants of the SpyEye bot to understand the internals of the malicious binary. Finally, we have dissected the plugin architecture of SpyEye botnet which allows us to present a component-level design. We have formulated the paper as follows:

- Based on the behavioral, static and source code analysis, we present a complete design of the SpyEye bot framework and its components.
- We explain the techniques and tactics used by the SpyEye framework to spread malware across the World Wide Web.
- We present a history of changes that have occurred in the SpyEye botnet framework.

Our analysis of SpyEye sheds light on the design of the third generation botnets.

2. Related work

Feily et al. [7] conducted a survey on botnet detection. In their study, various botnet detection techniques were discussed which includes signature-based detection, anomaly based detection, DNS mapping and mining. Examination of the detection techniques indicated that botnets hid their control by using multitier Command & Control (C&C) based on techniques such as DNS fluxing. Paxton et al. [8] presented a prototype for identifying the communication characteristics and agents involved in the botnet transactions by introducing an analysis system they called the MasterBlaster. They characterized the botnet interactions as a social mapping because of the botnet two-way communication model: when a bot sends a command, the node has to reply. Jang et al. [9] conducted a detailed study on the Waledac botnet which is an HTTP-based botnet that also uses a P2P protocol. The Waledac botnet used a protocol switching mechanism to shift from HTTP to P2P after infection which made it hard to detect using generic detection mechanisms. Dagon et al. [10] presented taxonomy of botnet structures in which different metrics were proposed to clarify the impact of different response techniques on the botnet activity. Additionally, Dagon et al. [11] proposed a model for studying the impact of time and location on the malware infections by the botnets.

Calvet et al. [12] discussed botnet detection strategies based on modeling, analysis and experimentation on the botnets. Stone-Gross et al. [13] conducted an analytical study of the Torpig botnet that was distributed using Mebroot rootkit. It replaced the Master Boot Record (MBR) to take control of the victims' machines. Mebroot hid itself by exploiting the MBR so it was activated before the operating system was actually booted. The complete lifecycle of Torpig botnet was analyzed and various infection techniques were presented. Li et al. [14] investigated different ways to analyze malicious traffic samples to understand the relative significance of botnet probing sequences in real time. A methodology of analyzing botnet probing activity was presented. Their technique enabled them to decipher the incoming requests using local observation techniques resulting in characterization of the different scanning strategies used by the botnets.

Vo and Pieprzyk [15] presented protection measures to secure Web 2.0 services from botnet exploitation. The researchers described the sophistication of botnets using Web 2.0 services as C&C channels. To resist and circumvent botnet operations, they proposed an algorithm using an encryption routine with session keys and CAPTCHA verification. Wei et al. [16] discussed the concept of an online botnet traffic classifier in which they classified the network traffic using payload signatures and a decision tree to detect the presence of online communities participating in the botnet activities. Kok and Kurz [18] presented details of the botnet ecosystem explaining the difference between botnets and other malware. They also discussed in detail the botnet player roles and the amount of revenue generated by botnets in the underground economy. Zhu et al. [17] categorized botnet research into three different sets: understanding botnets, analyzing botnets and defending against botnets.

A significant amount of botnet research exists and is crucial for understanding and thwarting botnet attacks. We extend that work with a detailed analysis of the latest type of botnet.

3. Experiment setup

The key to our approach was to create an experimental setup so we could analyze the SpyEye botnet in action. We were particularly interested in the communication pat-

terns between the bot and the C&C server. In order to create such a setup, we needed to obtain and then install a bot in a controlled environment, and we required access to a C&C server. Botnets can be found in malware repositories, but access to the C&C server was more challenging. We performed penetration testing on the C&C servers to detect vulnerabilities in the web application interface used by SpyEye. We detected SQL injection vulnerabilities in the SpyEye C&C web interface ([24] that allowed remote access to the server. Once the SpyEye bot was installed in the controlled environment and we had access to the C&C server, we could dissect the network activity.

Other researchers have examined botnets' secondary effects such as phishing emails, analyzing DNS queries [19] and mapping C&C servers based on DNS network data for performing analysis. These techniques are good for deciphering the botnet behavior and assisting in detection but they do not provide much insight into the core design of botnets. Our setup gave us access to hidden details of SpyEye.

It took 6 months to collect a sufficiently large selection of SpyEye versions for this study. To get samples, we monitored various P2P networks, storage repositories, and rapidshare storage version systems. We also monitored several underground forums to find any new development taking place in SpyEye. It was actually a hard process to collect the SpyEye binaries for reverse engineering. SpyEye uses a builder (bot generator) to generate a bot. Several versions of SpyEye builder were protected with a Hardware Identifier (HWID). It uses VMProtect which is a lock mechanism used to install software to a specific physical device or system. Both HWID and VMProtect exist to prevent the access that we needed, so they created a daunting challenge, but surmountable with effort and time. For analyzing and generating the bot, the builder has to be reverse engineered against these. More details will be presented in the bot builder section. We used multiple sets of virtual machines running the Windows OS to emulate the environment so we could map and monitor changes in the systems in real time.

We used tools such as IDA Pro Disassembler, Olly Debugger, Exescope, PEID, Anubis [20] and BitBlaze [21] to trace the properties of the SpyEye binaries. In addition, we created custom designed scripts for further analysis of the bots. For traffic analysis, we used Wireshark, tcpdump, xplico, and network miner tools. We used the following to organize our study of the impact of SpyEye.

- The compromised system could seriously affect the safety of users surfing the World Wide Web.
- The executable modifies the file system so it can have a dramatic impact on the system state.
- The executable reads, modifies, creates and monitors registry values.
- The executable registers processes to be active at system start resulting in unwanted actions to be performed automatically.

The overall results of our experiment are modeled into a framework as presented in the next section.

4. Inside SpyEye framework

The SpyEye framework is composed of three layers that include client-side components, server-side components, and intermediate-interface components. The framework has a built-in Bot Development Kit (BDK). The complete modular design of SpyEye framework is presented in Fig. 1. In the following sections, we will discuss the design.

4.1. C&C server – Main Admin Panel

The Main Admin Panel (MAP) on the C&C server is the heart of SpyEye. It is designed for executing administrative operations and maintaining the records of bots that are installed on the compromised machines. MAP is a PHP-based interactive web management panel that is installed on the web server and utilizes Web 2.0 technologies to communicate over HTTP. The MAP functionality can be summarized as follows:

- MAP sends commands to the SpyEye bot on the compromised client. If a bot goes astray, MAP has the capability to ignore or uninstall it.
- MAP logs all changes in the installed bots.
- MAP can create tasks for SpyEye bots remotely. The main three types of tasks (jobs) are: updating the bot executable, updating the bot configuration and loading third party executable.
- MAP has an anti-virus testing (virtest) module so it is possible to find the detection rate of SpyEye.
- MAP has a module for managing and controlling the nature of plugins that are configured in the bot during build time. However, MAP can also activate any plugin for a particular bot later on. This capability shows that SpyEye supports a plugin architecture that is portable and flexible.
- MAP has a configuration component that can configure various modules in the framework.
- MAP has a statistics module that manages the traffic statistics and successful infections by the SpyEye bot based on geographical preferences.
- MAP has a variety of interfaces such as FTP, SOCKS and RDP that work directly with the plugins. The variety of servers provides flexibility for compromised machines to bypass some host-based protection software such as firewalls. A particularly useful task is to enable an infected machine within a Network Address Translation (NAT) environment to connect back to the C&C server.

MAP is the information hub and guardian of the SpyEye botnet. MAP executes global commands using HTTP to provide robust control over the operational bots.

There is a Backend Database Server (BDS), but it is not hosted on the same domain as the MAP. Bot herders use different domains to host MAP and BDS in order to reduce complexity and loss: if one server is compromised, the other can be preserved.

Fig. 2 shows the MAP interface control panel of the latest version of SpyEye. Table 1 shows the set of configuration parameters used in MAP:

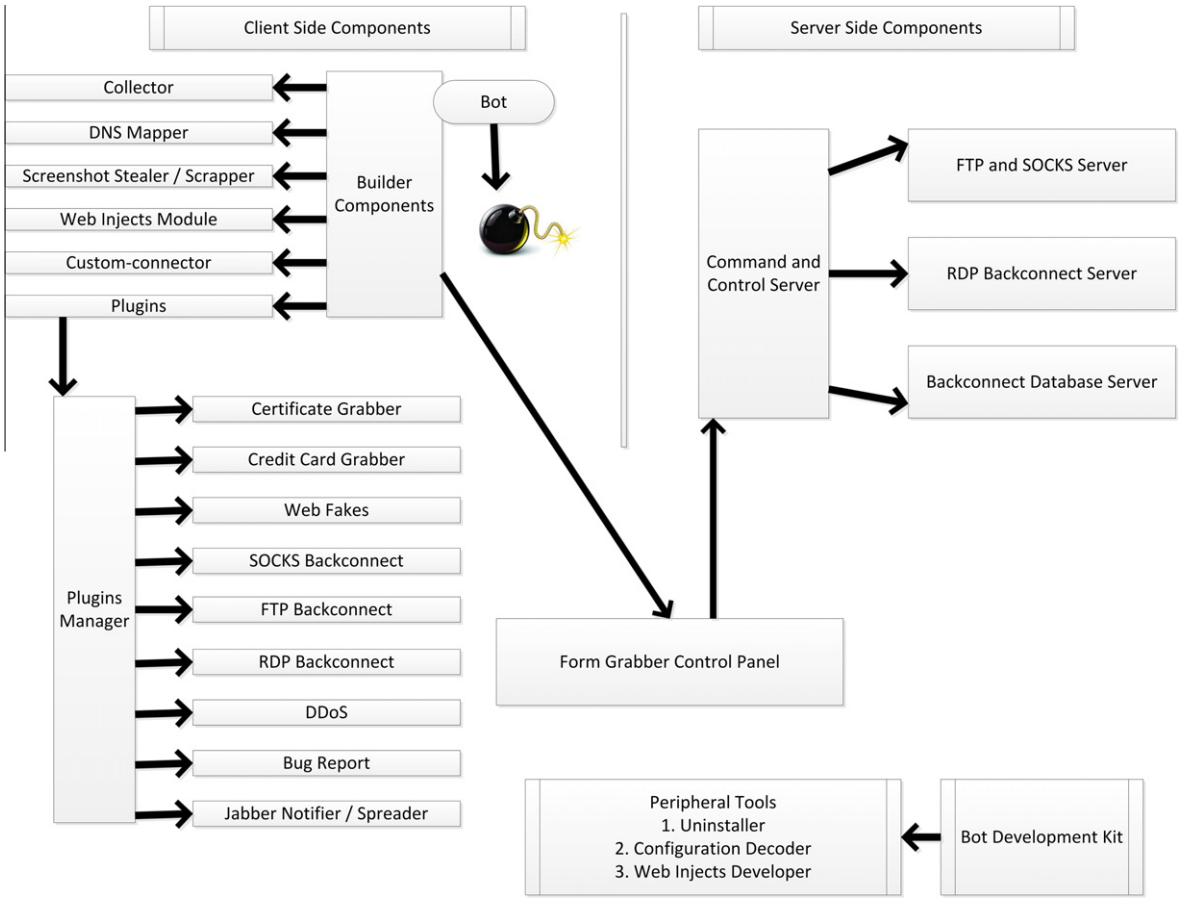

Fig. 1. SpyEye framework in action.

4.2. Backend Collector

The Backend Collector (BC) is a Backend Database Server (BDS) that is the primary component of the SpyEye framework for storing stolen data. The BC is activated as a daemon on the same domain or a different domain where the MAP is hosted. The BC is queried through a well-defined web interface to make management easy. Bots installed on the compromised machines send data directly back to the BC that stores the extracted information such as screenshots and credentials. The BC is structured in a modular fashion and configured with a predefined database having a specific set of tables created for storing information. Mostly, SpyEye uses MySQL for database operations. To make the data transfer process faster, the bot has a built-in capability to use LZO compression for efficiently sending data in a compressed format. LZO is probably used because it works fine on all operating systems, as it is platform independent by design. In SpyEye terminology, the TCP based packets for transferring data used by bots to send logs to C&C is called "sausages". By using encryption and compression, the C&C server collects data efficiently and securely.

The BC consists of several tables that read input data from several files stored on the server which are discussed as follows:

- "*table_screens.sql*" file holds the information about various screenshots take from victims' machines.
- "*table_reports.sql*" file holds the information about statistics and reports of the infected machines.
- "*table_register.sql*" file holds the information about the bots that are registered with C&C.
- "*table_hostban.sql*" file is updated with the information about the number of banned hosts.
- "*table_exceptions.sql*" file manages information about various exceptions.
- "*table_creditcards.sql*" file holds the information about stolen credit card numbers.
- "*table_certifications.sql*" file is updated with information about certificates.

We will provide detailed information in later sections about the APIs that are used to implement data transfer routines from infected machines to the C&C.

4.3. FormGrabber Admin Panel

The Form Grabber Admin Panel (FAP) is an intermediate graphical interface used to manage and explore the data stored in the BC. The FAP interface is designed in PHP with AJAX to communicate with the BC in an asynchronous manner. Basically, information stolen by all the plugins

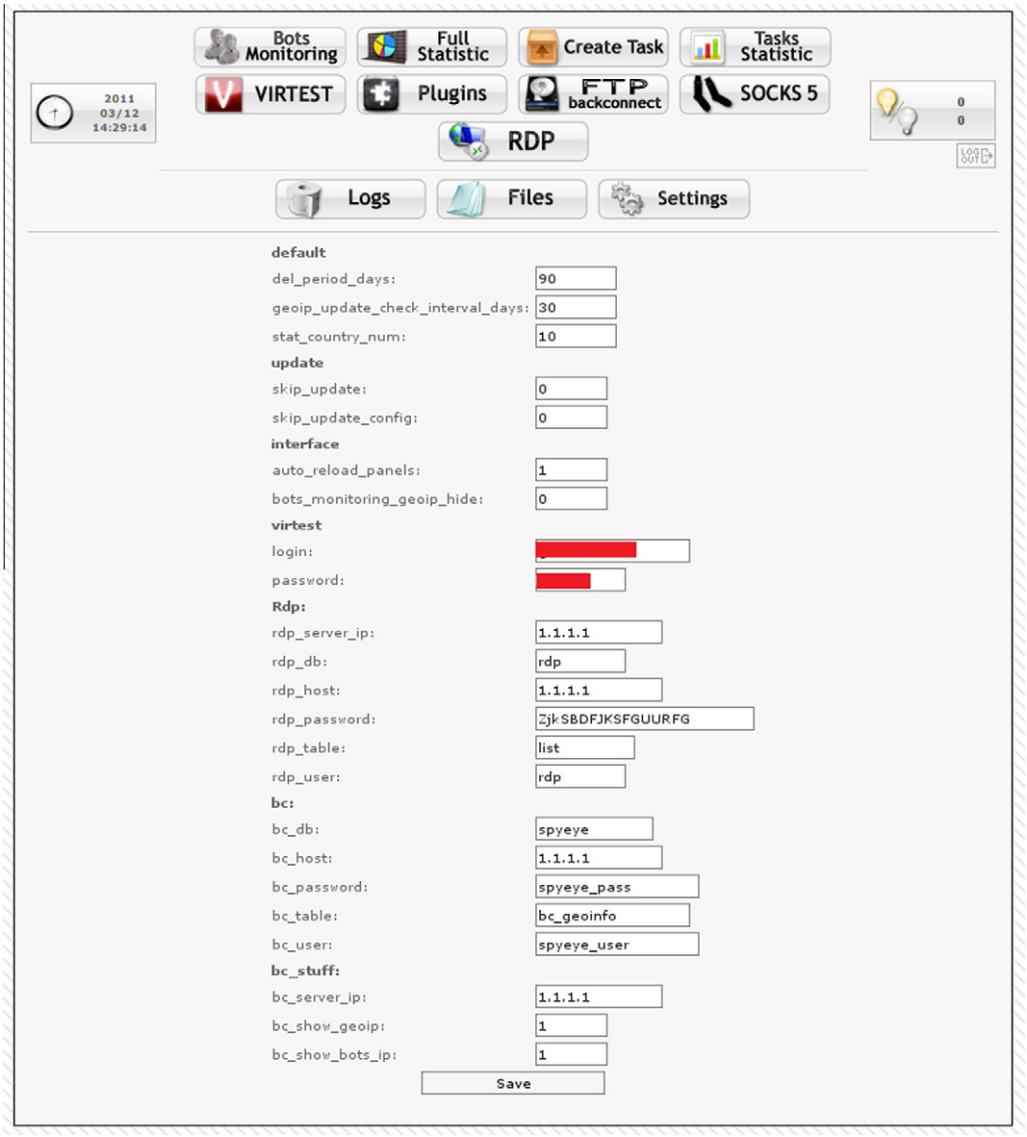

Fig. 2. SpyEye's Main Admin Panel (MAP).

and bots can be easily queried using FAP. It is possible for the FAP and MAP to not be hosted on the same servers. The FAP has number of modules:

- The "*Find Info*" module queries the BC to display information about the bots.
- The "*Statistic*" module queries the BC for details of ongoing infections across the world.
- The "*FTP Accounts*" module queries the stolen FTP credentials present in the BC.
- The "*Settings*" module is used to configure the different options in FAP for querying data.
- The "*Screen Shots*" module queries the BC to provide screen shots of websites taken by bots.
- The "*BOA Grabber*" module looks for the presence of credentials of Bank of America user accounts.
- The "*CC Grabber*" module queries the BC for stolen credit card information.
- The "*Certificate Grabber*" module queries the BC for stolen certificates from compromised machines.

Fig. 3 shows the FAP of SpyEye botnet.

4.4. BackConnect Servers – FTP, SOCKS and RDP

Currently SpyEye has three additional servers, FTP, SOCKS and RDP, for supporting its activities. These servers are called BackConnect servers because the C&C initiates a connection back to these servers for communication when the network is not reachable by the C&C due to the presence of NAT. That is, the host machines in the internal network do not have dedicated IP addresses. In those

Table 1
Configuration parameters used in SpyEye's MAP.

MAP configuration parameters	Details
bc_db	BDS database name to be used for storing stolen data
bc_host	Host IP address where the BDS is to be hosted
bc_password	Password for the configured database
bc_table	Tables having information about installed bots
bc_user	User configured for BDS for accessing data
bc_server_ip	Host IP address where the BDS daemon runs
bc_show_geoip	To display the information about bots GeoIP
bc_show_bots_ip	To display information about the bots IP
login [virtest]	Login account username for virtest services
password [virtest]	Password of the account configured for virtest services
auto_reload_panels	Updating statistic panel automatically after a few seconds
bots_monitoring_geoip_hide	Hiding the GeoIP information of bots
use build-in pe loader	Configuring the execution mode of the bot
replace	Replacing the bot executable if required
skip_update	Skipping or configuring the bot with latest updates
skip_update_config	Skipping or configuring the bot with latest configuration
del_period_days	Specifies the number of days after which the bot is to be deleted
geoip_update_check_interval_days	Specifies the number of days after which GeoIP info is updated
stat_country_num	Displaying the number of countries to be displayed in statistics
rdp_host	Host IP where the RDP database is hosted
rdp_db	Name of the configured RDP database
rdp_server_ip	Server IP address where the RDP daemon runs
rdp_user	RDP database user
rdp_password	RDP database password
rdp_table	Table having information about bots using RDP back connect

environments, the plugins initiate connections back to the BackConnect servers for dumping stolen information. This technique works quite well for exfiltration of data from compromised machines. Table 2 shows the details of configuration parameters used by the SpyEye for different servers.

4.5. SpyEye Bot Development Kit

SpyEye's Bot Development Kit (BDK) is a type of Software Development Kit (SDK). Instead of SDK, we call it a BDK because the framework completely revolves around the building of a bot for nefarious purposes. SpyEye's BDK is designed in VC++ which supports modular design, in particular, their plug-in architecture. The idea behind the BDK is to make the infection framework portable so that third party malware authors can easily incorporate custom code—increasing the sophistication of spinoffs. The plugin architecture is structured as a Dynamic Link Library (DLL) having a configuration file with extension .CFG. By default, the plugin name is the name of the DLL. To design a custom plugin, the BDK expects three functions to be defined in the exported DLL table named INIT, START and STOP. The INIT routine defines the initialization of the plugin; the START and STOP routines are required for managing the plugin from the SpyEye MAP (described above). The BDK provides two ways of executing code using plugins:

- *Local Infections*: When the plugin is designed for local scope, then the plugin code executes in the main process infected by the bot, e.g. explorer.exe
- *Global Infections*: When the plugin is designed for global scope, then the plugin code executes in all the processes available in the system.

We analyzed the SpyEye's BDK and found that number of functions are used as presented in Table 3.

4.6. SpyEye Bot Builder

Builder is used to build the bot with a specific configuration. The bot herder applies the required settings in the builder which includes plugin information, MAP configuration and BC configuration, etc. After successful compilation of the code with the applied configuration, the malicious bot is generated. The builder usually has configuration parameters that require paths for including resources that can be present locally or remotely. The plugins and other peripheral programs reside in the MAP and are compiled dynamically during build time.

The builder program is usually protected with some packer or system specific protection module. For example, the SpyEye builder is protected with VMprotect and HWID protection. VMprotect is a system-specific obfuscation technique in which machine instructions are transformed into pseudo code that is randomized. VMProtect is also called a virtualization obfuscator because it changes the x86 instructions into custom code that is interpreted and executed during run time. For that reason, VMProtect contains its own interpreter. It is possible to reverse engineer the VMprotect [22], but it is time consuming. HWID stands for Hardware Identification and is a technique to provide software licenses for individual machines. HWID uses VMProtect to install the SpyEye builder to a specific physical device. One goal of using HWID protection is to prevent others, including analysts, from debugging and reverse engineering the builder itself. The builder also has other options to thwart reverse engineering of the bot: SpyEye uses the UPX packer to randomize the Object

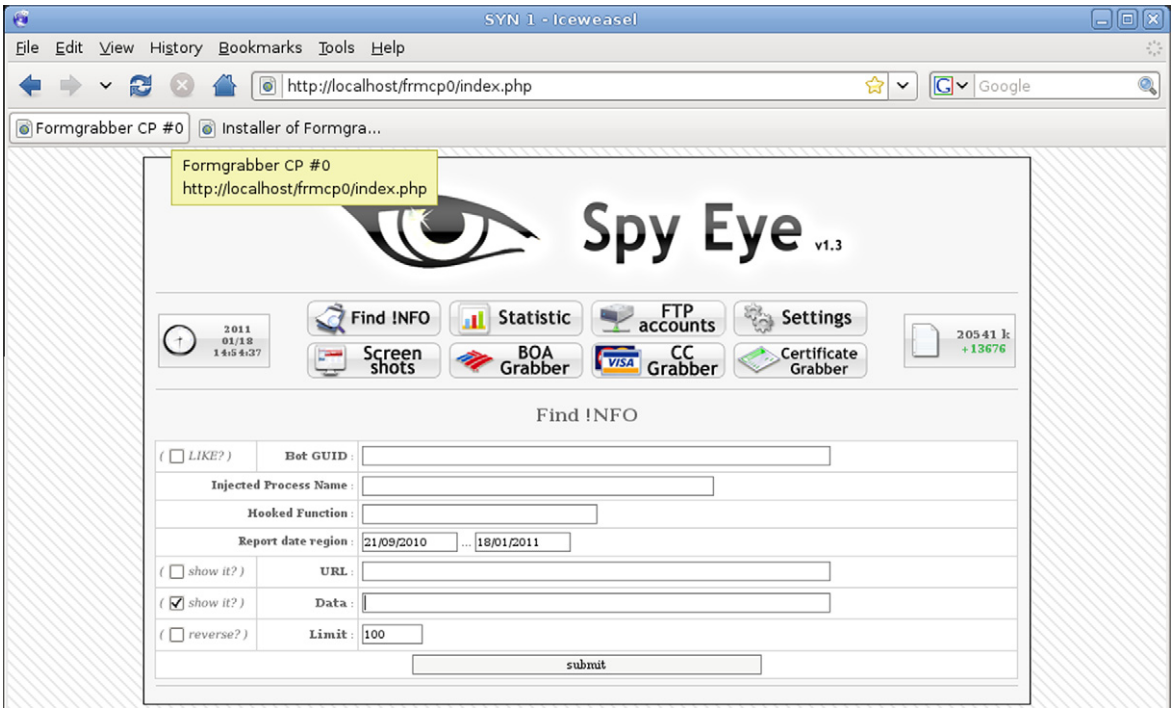

Fig. 3. SpyEye's FormGrabber Admin Panel (FAP) in action.

Table 2
Backconnect server-configuration parameters.

Server configuration parameters	Details
socks_port	Port used for listening connections from socks-plugin
ftp_port	Port used for listening connections from ftp-plugin
ping_timeout	Wait time for getting a reply from a bot. Bot is removed in 5 s if no reply received
threads_number	Thread count to process network activity
ftp_limit	Maximum number of sockets for connections by ftp-plugin
socks_limit	Maximum number of sockets for connections by socks-plugin
Login	FTP server authentication – username used by ftp-plugin
password	FTP server authentication – password used by the ftp-plugin
geoip_path	Path to the GeoIP DB file (GeoIPCity.dat by default)
mysql_host	Server IP address for running MySQL daemon
mysql_user	MySQL database user
mysql_pass	MySQL database password
mysql_db	Database for writing information about bots
mysql_table	Tables for writing information about bots
mysql_table_rdp	Tables used for RDP connections
mysql_table_logs	Tables used for RDP logs
cfg_file_log_enabled	Configuration file for storing debug information with flags
cfg_file_log	Dumping log information to a specified path
cfg_file_log_maxsize	Maximum size of the log file
cfg_file_blacklist	Blacklisting bots for sending wrong magic codes
cfg_ip_address	RDP server IP address for listening connections
cfg_rdp_port_in	Port number used for incoming connections
cfg_rdp_port_out	Port number used for triggering back connects
magic_code	String required for authentication of clients

Entry Points (OEP) in the bot. In addition, SpyEye's decentralized design increases the difficulty of detecting all of its components. To install SpyEye builder, the builder itself had to be compromised by us. Steven [23] has blogged about the details of reverse engineering SpyEye builder so the bot could be installed.

The installation and configuration files are the part of MAP and included during the build process. To manage and protect the build, there is a symmetric encryption key shared between the MAP and builder. Once the bot is generated, the encryption key is hard coded into the bot itself. Configuration parameters are also encrypted. Since the

Table 3
Listed function in SpyEye's BDK.

BDK functions	Details
TakeGateToCollector	Executed to trigger a separate thread for sending data back to BC using gate function
TakeGateToCollector2	Executed for sending data back to BC using gate function without instantiating a separate thread
TakeBotGuid	A unique identifier for the bot
TakeBotPath	A path where bot is installed in the machine
TakeBotVersion	The version of the installed bot
GetState	Used to decipher the state of plugins [on/off]
KeepAlive	A flag used for verifying the state of the plugins
IsGlobal	A flag used to determine whether to inject code in all active processes
Callback_OnBeforeProcessUrl	Callback function to get the details of URL before it is processed
Callback_OnBeforeLoadPage	Callback function to get the content of webpage before it is loaded in the browser
Callback_OnAfterLoadingPage	Callback function to get the contents of webpage after it is loaded in the browser
Callback_ChangePostRequest	Callback function to get the contents of POST request
TakeGetPage	Function used for downloading the resources over HTTP/HTTPS
TakeGetPage2	Function used for downloading the resources over HTTP/HTTPS with enhanced functionality
TakeFreeMem	Function used by SpyEye memory manager to free the dynamic memory
Callback__WS2_32__send	Callback function used for sending and manipulating the output of the data
TakeConfigCrc32Callback	Callback function to determine the identification of the bot configuration file
TakeBotExeMd5Callback	Callback function to determine the MD5 hash of the bot
TakePluginsListCallback	Callback function used for collecting data about the installed plugins
TakeMainCpGateOutputCallback	Callback function used to transfer the information from MAP to the bot
MainCpGateInput	Function used by the SpyEye bot to transfer data back to MAP
TakeUpdateBotExe	Function used for updating the bot executable
TakeUpdateConfig	Function used for updating the bot configuration file
TakeStartExe	Function executed by plugins to trigger the execution of third-party executable
TakeGatePipeSendMsg	Function used to manage a custom connector plugin

configuration file is included from a remote location, the builder actively manages communication to avoid disruption in the transfer of configuration parameters.

The builder also has the ability to build a lightweight bot close to 40 KB. One trick is to keep objects such as the configuration file (config.bin) in the MAP rather than in the bot. During the build, the bot downloads the configuration file directly from MAP. The builder also supports ZLIB compression used for optimizing data. However, this desirable feature can also cause a problem. Under certain scenarios, ZLIB compression impacts the state of bot in performing Web Injects—the injection of malicious data into a web session (more later). The problem can occur because some web servers send data in compressed format such as "gzip/deflate" by specifying a Content-Encoding header in the HTTP request. As a rule of thumb, the browser has to send the Accept-Encoding header to inform the web server about the type of data encoding recognized. So ZLIB is required when the web server sends data back in compressed format. If the ZLIB compression is not enabled and data is compressed, the bot fails to perform Web Injects.

Other tidbits: the generated bot has a unique name that is required for system wide control for executing operations. Also, during the build process, the bot is usually configured with an option to delete the cookies regularly.

Fig. 4 shows the SpyEye builder in action.

4.6.1. Understanding the Bot

The bot has similarities to rootkits in that it digs deep to hide in the system. Rootkits sit in between operating system and the applications and this positioning makes them hard to detect. In the ring model, the hardware and OS divide control among four basic rings: Ring 0 to Ring 3. Ring 0 is for the kernel, Ring 3 is for the user, and most OS's make little use of the other rings. Rootkits prefer Ring 0, but the SpyEye bot primarily works in Ring 3. Code executed in the Ring 3 has fewer capabilities than in Ring 0, but Ring 3 rootkits are easier to write and easier to install yet retain significant control over the applications' address space. Ring 3 rootkits can use Import Address Table (IAT) hooking, inline hooking and DLL Injections using *CreateRemoteThread, Registry AppInit_DLLs* setting or using *SetWindowsHookEx* APIs to hook system and application routines.

Fig. 5 shows the working nature of a Ring 3 bot.

As discussed earlier, the generated bot has an interface with plugin architecture. Depending on the plugin, it can inject code in a specific application process or all processes running in the system. The primary processes subject to injection are *firefox.exe*, *iexplore.exe* and *explorer.exe* since the browser is the window to the Internet. The bot works by injecting DLLs into the legitimate browser process, and then it hooks critical functions such as those that provide information about the data flow or the data itself. With the Ring 3 modifications mentioned above, the bot can use the native windows API to execute the code.

In a sense, the bot self-destructs after it executes, in that it "melts" into the system to hide its presence. If a dropper is executed to install the code, it then deletes itself from the system. Once installed the bot takes control of the browser process, so it can commandeer all the data from browser and transfer it back to the BC. The capability of bot depends on the plugins that can be developed to handle specific tasks to extract specific data related to specific targets. For example, the builder can be configured with Anti-Rapport, a module that neutralizes anti-rootkits such as Trusteer Rapport. Basically the bot kills the anti-rootkit

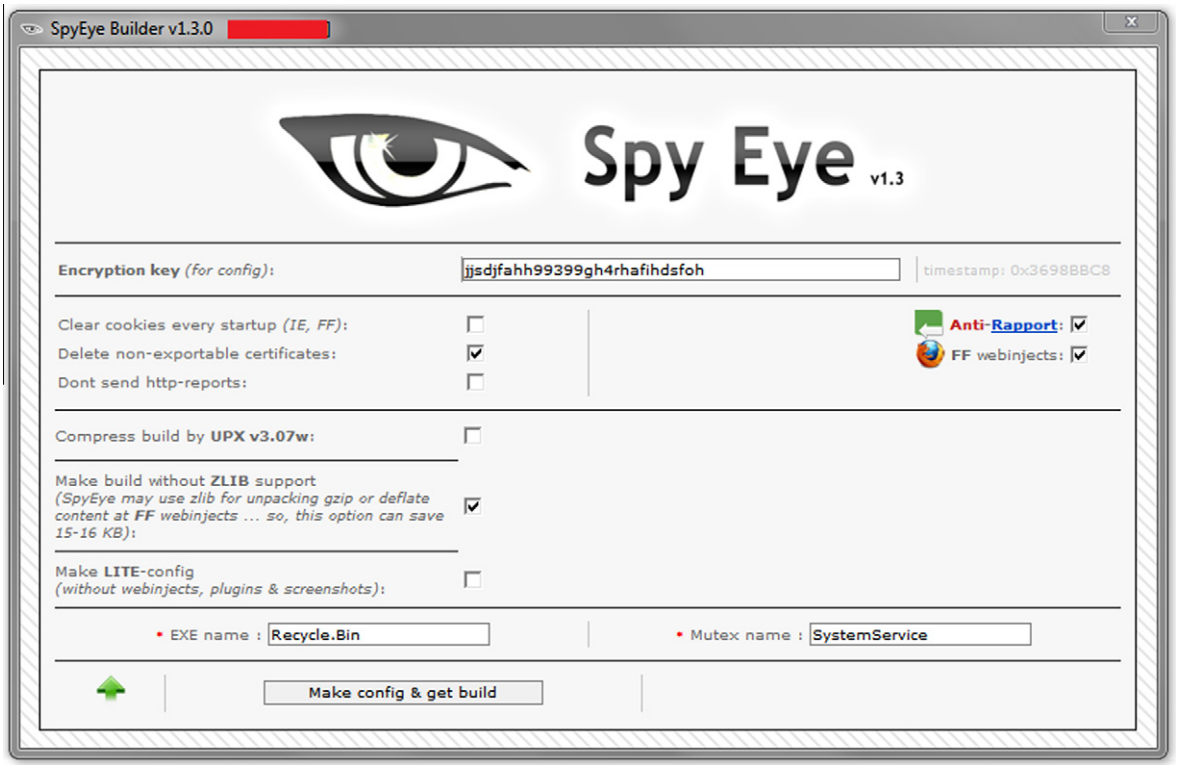

Fig. 4. SpyEye builder in action.

threads and blocks its ability to write messages back to the reporting engine.

4.6.2. Custom-connector module

Custom-connector is a module that acts as an interface between the bot and MAP. This functionality was introduced in the SpyEye version 1.3.x. After some time interval, the bot sends identity information back to Custom-connector using an HTTP GET request. Basically, the information maps the environment of the victim's machine including operating system details and installed browser. Based on the identity information of the bot, MAP sends command

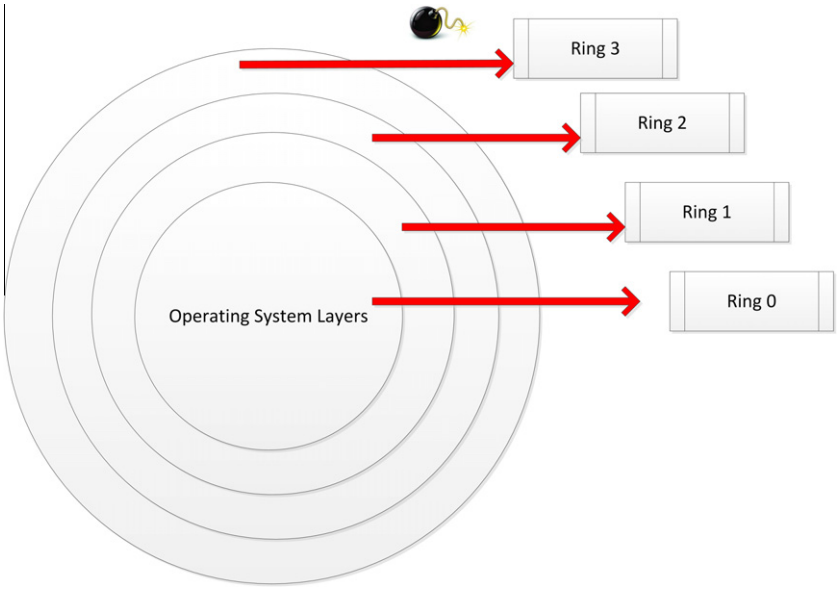

Fig. 5. Abstract layout of working of ring 3 bot.

back to the bot that include configuration and executable updates. With that information in hand, it is easy for the bot herder to encrypt the traffic between bot and MAP. The Custom-connector plugin sends information in a specific format back to MAP as presented in Scheme 1.

The string presented in Scheme 1 contains information that is extracted using various functions defined in the BDK. In this example, "**guid**" is an outcome of a *TakeBotGuid* call that contains the unique identifier for the installed bot. The "**ver**" holds the value extracted using *TakeBotVersion*. Similarly, "**ccrc**", "**md5**", "**plg**" contains the values extracted from the infected machine using *TakeConfigCrc32Callback* , *TakeBotExeMd5Callback* and *TakePluginsListCallback,* respectively. The remaining parameters are optional but cover information related to installed plugins.

4.6.3. Web Injects

Web Injects is a technique of injecting illegitimate content into the HTTP responses coming from legitimate websites. This infection strategy is very effective because it allows the bot to exploit the browser process (hooked already) and to inject content that is displayed as inline content in the browser. This practice is quite malicious because of its ability to modify the web pages on the client side. For example, consider a user visiting a bank's website. The incoming HTTP response is modified to inject fake "username" and "password" input boxes that look legitimate to the user but in-fact it is rogue content. This process exploits an unsuspecting user to fool them into providing their login credentials. The installed bot captures that information and sends it back to the BC. Scheme 2 shows an example of Web Injects code present in a webinjects.txt file.

An injection is divided into four tags that are specified in a webinjects.txt file:

- *data_before/data_end*: This tag designates a portion of the webpage after which the injection is placed, i.e. in the final page it will be "before" the injected code.
- *data_inject/data_end*: This tag defines the actual code to be injected into the incoming HTTP response. The injection can be a single line of code or a complete form as there is no specific restriction placed on the length of the injected payload.
- *data_after/data_end*: This tag defines a portion of the webpage before which the injection is performed, i.e. in the final page it will be "after" the injected code.
- *set_url*: This tag defines the URL of the targeted website against which the Web Injects is to be performed.

Additionally, flags G (GET) and P (POST) are used in the *set_url* tag to force the bot to perform Web Injects when a GET (or POST) request is initiated. Additional flags include L and H which are used for grabbing injects in webpages. The main difference between the L and H flags is that the L flag specifically extracts the data present in the data_before, data_inject and data_after tags whereas the H flag captures the rest of the webpage. In SpyEye terminology, the extracted webpage content is termed as ripped content and is passed back to the backend database. The ripped content can be queried through FAP by specifying the search pattern string "Grabbed Data" in the hooked function module. The Web Injects technique is complex and illustrates the sophistication of third-generation botnets.

4.6.4. Screen Scraper

The Screen Scraper is a module implemented in SpyEye to scrape the content of a screen in a pre-defined interval of time. During build, the rules for a screen scraper are specified in the *screenshots.txt* file. Once the bot is built, it has the capability of taking screen shots whenever a user clicks the mouse. By default, the center of the screenshot is the mouse cursor. This module can be customized for a specific target website so the bot will only take screen shots for that domain only. The screen scraper configuration requires information in the format as shown in Scheme 3.

The parameters in Scheme 3 cover the layout of a screenshot to be taken by the bot. The %URL_MASK% is used to define the URL against which the screen scraper plugin is activated. The %WIDTH% and %HEIGHT% defines the width and height of the screenshot respectively. The %MINIMUM_CLICKS% and %MINIMUM_SECONDS% are used to turn off the rules after a specific number of clicks and seconds respectively.

4.6.5. DNS mapper and multiple collector

SpyEye also supports two additional functionalities in order to make the build process easier. These are:

- DNS Mapping is a technique that can be specified by SpyEye during the build process to specify the names of DNS servers (in the *dns.txt* file) so the bot will use the listed DNS servers to resolve the address of the C&C server. This capability is important because a number of networks restrict the DNS servers by blacklisting them.
- SpyEye also has the capability to specify multiple connectors during build time so that the bot is able to send the information back to all the servers listed in the *connectors.txt* file. This is a failsafe option so if one server fails, the information can be sent to another server.

```
gate.php? guid=!USER-
5C377A2CCF!046502F4& ver=11435&stat=ONLINE&ie=6.0.2900.2180&os=5.1.2600&ut=Admin&
ccrc=13A7F1B3& md5=b9c3cb2cdc66b1f4465fe56cc34040b2& plg=customconnector
```

Scheme 1. Information transferred to gate component.

```
set_url */my.ebay.com/*CurrentPage=MyeBayPersonalInfo* GL
data_before
 Registered email address</td>*<img*>
data_end
data_inject
e-mail:
data_end
data_after
</td>
data_end

set_url *.ebay.com/*eBayISAPI.dll?* GL
data_before
(<a href="http://feedback.ebay.com/ws/eBayISAPI.dll?ViewFeedback&*">
data_end
data_inject
Feedback:
data_end
data_after
</a>
data_end
```

Scheme 2. Web Injects in action.

```
%U RL_MASK% %WIDTH% %HEIGHT% %MINIMUM_CLICKS%
%MINIMUM_SECONDS%
```

Scheme 3. Configuration parameters used in screen scraper module.

4.7. Plugin architecture

Plugins are designed to extend the capability of a bot and are a characteristic of third-generation botnets. As discussed in the BDK section, plugins play a critical role in the success of data exfiltration from compromised machines. To execute plugins, these bots do not use the standard PE loader provided by the OS to download code from memory because that process is detectable by antivirus engines. Instead, they use their own custom loader for running plugin code from memory without detection. The plugins are actually stored in the bot configuration file. This feature results in smaller size of and faster execution. A BDK also has built-in, shellcode-specific plugin such as *ConfigShellcode* and *InstallShellcode* to trigger the configuration and installation of the bots on the victim machine. Plugin functionality is discussed next.

4.7.1. Certificate Grabber

The Certificate Grabber plugin is designed to steal certificates from the Windows built-in cryptographic storage. The certificate grabber works differently for Internet Explorer (IE) and Firefox (FF) so they each have a plugin. This is because IE is a built-in component of the Windows OS whereas Firefox uses its own repository and storage mechanism to handle certificates. This plugin has only one configuration parameter: the minimum time the bot waits before sending the certificates back to the BC. The stolen certificates are marked with "FF" or "IE" so that data harvesting is easier in the FAP.

4.7.2. Credit Card Grabber

Credit-Card Grabber plugin is designed in a sophisticated manner for extracting Credit Card (CC) numbers from POST requests sent by the browser. This plugin has a built-in implementation of the LUHN algorithm for validating the CC. If the CC number is validated, the plugin will send that information back to the BC. The ripped CC numbers can be found easily by querying an interface through the FAP.

4.7.3. Web Fakes

The Web Fakes plugin is designed to spoof the content of HTTP/HTTPS resources without showing the legitimate content to the user. Generally, this plugin redirects the legitimate URL to the URL resource defined by the bot herder in the configuration file for Web Fakes. The Web Fakes plugin is configured as presented in Scheme 4.

The %URL_MASK% is used to configure the target URL which is to be masked and %URL_REDIRECT% is used to redirect the browser to the specified URL. Additionally, the %POST_BLACK_MASK% and %POST_WHITE_MASK% parameters are defined for deactivating and activating the rules after POST requests respectively. HTTP POST, GET and other requests are defined in the %FLAGS% parameter. To block and unblock the rules %BLOCK_URL% and %UNBLOCK_URL% parameters are used.

4.7.4. Distributed Denial of Service (DDoS)

This plugin is designed to launch DDoS attacks against anti SpyEye servers or detection websites such as *abuse.ch*. Currently, this plugin is not powerful enough to conduct a

```
entry "WebFakes"
%URL_MASK% %URL_REDIRECT% %FLAGS% %POST_BLACK_MASK%
%POST_W HITE_MASK% %BLOCK_URL% %WEBFAKE_NAME% %UNBLOCK_URL%
end
```

Scheme 4. Web Fakes configuration parameters.

DDoS for a long time. This plugin was introduced in the latest versions of the SpyEye version starting from 1.3.x. It can also support multiple flooding at the same time. The configuration file has a set of parameters as presented in Scheme 5.

The flood parameter is used for specifying the flood type which can be Slowloris, SYN or UDP. The target parameter specifies the destination to be flooded. The port parameter is the used for declaring the target port that is to be flooded. The time parameter is used for defining the time interval that the flood remains active. The DDoS plugin is started from the MAP to begin the flooding attack.

4.7.5. Backconnect Plugins (FTP/SOCKS/RDP)

The Backconnect plugins are designed for flexibility of communication. SpyEye supports Backconnect plugins for FTP, SOCKS and RDP. The functionality is as follows:

- The FTP Backconnect plugin starts an FTP server interface on the bot in the compromised machine and provides access to the bot herder through Backconnect FTP server. As a result, the bot herder is able to access the victim's machine's directory structure from the C&C panel. It is also possible for the bot herder to use a custom FTP server to initiate connections with the compromised machine.
- The SOCKS Backconnect plugin starts a SOCKS server interface on the bot in the compromised machine. Basically, the SOCKS protocol is implemented to bypass host-based intrusion detection systems and firewalls. With this plugin installed, a bot installed on the machines inside a NAT can be connected back to the C&C.

Scheme 6 shows the list of parameters that are used to configure FTP and SOCKS Backconnect plugins. Here %BOTNAME% specifies the name of the bot. The %IP% and %PORT% parameters define the IP address and port number for connections. Additionally, %RECONNECT_INTERVAL_MSEC% defines the time interval after which the plugin starts connecting again. To define some autorun commands, the %AUTORUN_FLAG% parameter is used.

- The RDP Backconnect plugin starts an RDP server on the compromised machine and forwards the connection to the Backconnect server on the C&C panel. This plugin also creates a hidden user on the system so that the RDP can be accessed remotely. This plugin is quite useful for the bot herder to initiate financial transactions from the victim's machine using RDP connections that look legitimate. The plugin has the following configuration parameters as presented in Scheme 7:

In this plugin, %IP_OF_BC_SERVER% and %PORT_OF_BC_SERVER%" hold the IP address and port number of the Backconnect server respectively. For accessing the Windows RDP, %WINDOWS_LOGIN% and %WINDOWS_PASSWORD% parameters are used to define the credentials. To authenticate the clients, %MAGIC_CODE% is used and to execute any commands the %URL_TO_PORTABLE_TCMD parameter is used.

Other plugins include Bug Report which is used to communicate the errors in the system when the bot fails. Finally, the Jabber Notifier is a notification call system used by MAP to send commands for starting plugins in the bot installed in the compromised machine.

4.7.6. FlashCamControl

Latest builds of SpyEye has shown one more improvement in plugin architecture with the addition of flash cam control plugin. This plugin is used to hijack inbuilt webcam and microphone on user's computer by manipulating the integrity of flash software. This is primarily done by tampering the configuration settings of flash software. As a result, this plugin allows the permanent grant permissions in the settings file for targeted websites. To perform this attack, a malicious flash file is required to be included in user's browser which is possible by using previously discussed plugins i.e. Web Fakes and Web Injects. Hence, this malicious file interacts with the browser and is used for sending recorded data to C&C server. However, this development is still very new but can be used at large scale with passage of time.

5. Chronology of SpyEye framework

By collecting the multiple versions of SpyEye, we were able to observe and study the evolution of their capabilities. By studying SpyEye over time, our analysis provided us with a better understanding of the software. Additionally, we hope that the extracted knowledge can be used to determine the future trends of botnets. The chronology of the SpyEye TGB is presented in Fig. 6.

SpyEye has improved significantly over time. In version 1.0.0, the framework supported only Form Grabber, POP3 Grabber, FTP Grabber and Auto Fill. These modules are basic information-stealing plugins. In version 1.0.65, the

```
flood_type target port time
```

Scheme 5. DDoS plugin configuration parameters.

```
%BOTNAME%;%IP%;%PORT%;%RECONNECT_INTERVAL_MSEC%;%AUTORUN_FLA
G%
```

Scheme 6. Configuration parameters for FTP and SOCKS BackConnect.

```
%IP_OF_BC_SERVER%  :%PORT_OF_BC_SERVER%;%MAG  IC_CODE%;%WINDO
WS_LOGIN%;%WINDOWS_PASSWORD%;%URL_TO_PORTABLE_TCMD%
```

Scheme 7. Configuration parameters for RDP plugin.

framework added a loader to execute commands and load malicious executable from the C&C server. This version also implemented a geo-location module for improved tracking of bot infections around the world. In versions, 1.0.70 and 1.0.72, Zeus bot killer and anti-detection techniques were added respectively. In addition, several bugs were fixed. The Backend Collector and LZO compression were included in version 1.0.75. LZO compression speeds up data transfer from the infected machine. In SpyEye version 1.0.80, support for a dropper was removed and a Web Injects module for Internet Explorer was added. After this version, the bot existed without a dropper and was lighter and more compressed. In version 1.1.0, the author added Web Injects support for Firefox and Netscape, and the builder was protected with HWID. Additionally, a screen scraper and the Bank of America information stealing plugin were added. In version 1.2.0, several Backconnect plugins were added (FTP, SOCKS and RDP) to bypass Network Address Translation (NAT). Also, a Credit Card Grabber plugin was included. In version 1.2.22, the BackConnect database was made more independent so it could be deployed on a different domain. Also, several optimization features were added in this version. In version, 1.2.60, the configuration module design was lightened, cookie support was made more robust and a certificate handling module was added. Currently, version 1.3.x scheme is running which

Versions	Chronological changes and advancements (Module specialities)
1.0.0	FormGrabber Module | POP3 Grabber | FTP Grabber | AutoFill
1.0.65	Loader | Bot Analytics | Location based Graph Generation | GEO based Infection Loader
1.0.70	Zeus Bot Killer | Basic Authentication Credential Stealer
1.0.72	Undetectable Bot (AV Bypass) | Bug Fixes
1.0.75	Backend Collector | LZO Compression | Bot Size Compression| Log Parsing
1.0.80	Dropper support removed | Web Injects (IE) supported
1.1.0	Web Injects Support for FireFox, Netscape | VM Protected Builder (Hardware Lock) |Screenshots Capturing Module | Bank of America (BOA) Grabber
1.2.0	Custom Plugins Framework | BackConnect Plugins (FTP/SOCKS) | Web Injects Optimization | Enhancement in SpyEye Bot Development Kit (BDT) | Credit Card Grabber Module
1.2.22	Unlimited Paths to BC Database | Plugin Support | Robust Storage for Web Injects | Web Fakes Module added | HTTP Traffic Optimization | Virus Test Module
1.2.4	Enhanced Data Stealing and Collection Module | "TakeGateToCollector" SDK API added
1.2.60	LITE Config Module added | Robust Cookies Handling support | Non Exportable Certificate Removal Module
1.3.x (1-4)	Email Grabber support | DDoS Plugin added | Advancements in Web Injects | Anti Rapport | RDP BackConnect Module added | Custom-connector Plugin | FlashCamControl

Fig. 6. Chronology of SpyEye's developments.

now includes a built-in DDoS plugin, an RDP BackConnect plugin, optimized Web Injects, a Custom-connector plugin for the interface, and Email Grabber support. Another advanced feature has been introduced in latest version of SpyEye termed as FlashCamControl that is used for hijacking cameras and microphones on victim machine. If the development of SpyEye stays active, we can expect more interesting tactics in the near future.

6. Conclusion

In this paper, we presented an example of third-generation botnets based on analysis of the most widely used botnet in this category—SpyEye. We have learned a number of lessons from analyzing this sophisticated malware. Our analysis was based on infiltration combined with static and behavioral analysis topped off with penetration testing on binaries and malware domains. This approach provided us with a unique opportunity to understand the characteristics and exploitation techniques used by the SpyEye bot in executing the attacks for stealing critical information from victims' machines. First, we analyzed the design of the SpyEye bot itself, which helped us to understand its various components. We analyzed some of the source code to understand the basic layout and how the many modules are cross referenced. Next, we analyzed the exploitation tactics used by the SpyEye framework in performing various malicious operations on the web browsers and operating systems. After that, we performed behavioral and static analysis on the generated binaries to understand the infection in victim systems and the way browsers are exploited. We generated a complete chronology of SpyEye advancements that took place over time. Finally, we collected all the information to put together a picture of how third-generation botnets worked. It is our hope that knowledge gained during this analysis can be transformed into better defensive mechanisms.

References

[1] B. Stone-Gross, M. Cova, C. Kruegel, G. Vigna, Peering through the iframe, INFOCOM, 2011 Proceedings IEEE, 10–15 April, 2011, pp. 411–415.

[2] A. Sood, R. Enbody, Browser Exploit packs – death by bundled exploits, in: Proceedings of 21st Annual Virus Bulletin Conference, Barcelona, Spain, 2011.

[3] D. Moore, G. Voelker, S. Savage, Inferring internet denial of service activity, in: Usenix Security Symposium, 2001.

[4] A. Ramachandran, N. Feamster. Understanding the network-level behavior of spammers, in: ACM SIGCOMM, 2006.

[5] S. Saroiu, S. Gribble, H. Levy, Measurement and Analysis of Spyware in a University Environment. In Networked Systems Design and Implementation (NSDI), 2004.

[6] A. Sood, R. Enbody, Spying on SpyEye, in: Proceedings of Hack in the Box (HITB) Security Conference, Amsterdam, Netherlands, 2011.

[7] M. Feily, A. Shahrestani, S. Ramadass, A survey of Botnet and Botnet detection, in: Third International Conference on Emerging Security Information, Systems and Technologies, SECURWARE '09, 18–23 June, 2009, pp. 268–273.

[8] N.C. Paxton, G. Ahn, M. Shehab, MasterBlaster: identifying influential players in Botnet transactions, in: IEEE 35th Annual Computer Software and Applications Conference (COMPSAC), 18–22 July, 2011, pp. 413–419.

[9] Dae-il Jang, Minsoo Kim, Hyun-chul Jung, Bong-Nam Noh, Analysis of HTTP2P botnet: case study waledac, in: IEEE 9th Malaysia International Conference on Communications (MICC), 15–17 December, 2009, pp. 409–412.

[10] D. Dagon, Guofei Gu, C.P. Lee, Wenke Lee, A taxonomy of Botnet structures, in: Twenty-Third Annual Computer Security Applications Conference, ACSAC 2007, 10–14 December, 2007, pp. 325–339.

[11] D. Dagon, C. Zou, W. Lee, Modeling botnet propagation using time zones, in: Proceedings of the 13th Annual Network and Distributed System Security Symposium (NDSS 2006), San Diego, CA, 2006.

[12] Joan Calvet, Carlton R. Davis, José M. Fernandez, Jean-Yves Marion, Pier-Luc St-Onge, Wadie Guizani, Pierre-Marc Bureau, Anil Somayaji, The case for in-the-lab botnet experimentation: creating and taking down a 3000-node botnet, in: Proceedings of the 26th Annual Computer Security Applications Conference (ACSAC '10), ACM, New York, NY, USA, 2010.

[13] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, Giovanni Vigna, Your botnet is my botnet: analysis of a botnet takeover, in: Proceedings of the 16th ACM conference on Computer and Communications Security, (CCS '09), ACM, New York, NY, USA, 2009, pp. 635–647.

[14] Zhichun Li, Anup Goyal, Yan Chen, Vern Paxson, Automating analysis of large-scale botnet probing events, in: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security (ASIACCS '09), ACM, New York, NY, USA, 2009, pp. 11–22.

[15] N.H. Vo, J. Pieprzyk, Protecting web 2.0 Services from botnet exploitations, 2010 Second Cybercrime and Trustworthy Computing Workshop (CTC), 19–20 July, 2010, pp. 18–28.

[16] Lu Wei, M. Tavallaee, G. Rammidi, A.A. Ghorbani, BotCop: An Online Botnet Traffic Classifier, in: Communication Networks and Services Research Conference, 2009. CNSR '09. Seventh Annual, 11–13 May 2009, pp. 70–77.

[17] Zhaosheng Zhu, Guohan Lu, Yan Chen, Z.J. Fu, P. Roberts, Keesook Han, Botnet research survey, in: 32nd Annual IEEE, International Computer Software and Applications, COMPSAC '08, July 28 2008–August 1 2008, pp. 967–972.

[18] Jan Kok, Bernhard Kurz, Analysis of the BotNet ecosystem, in: 10th Conference of Telecommunication, Media and Internet Techno-Economics (CTTE), 16–18 May, 2011, pp. 1–10.

[19] M. Rajab, J. Zarfoss, F. Monrose, A. Terzis. My Botnet is Bigger than yours (maybe, better than yours): why size estimates remain challenging, USENIX Workshop on Hot Topics in Understanding Botnet, 2007.

[20] U. Bayer, C. Kruegel, E. Kirda, TTAnalyze: A tool for analyzing malware, in: 15th European Institute for Computer Antivirus Research (EICAR 2006) Annual Conference, April 2006.

[21] Dawn Song, David Brumley, Heng Yin, Juan Caballero, Ivan Jager, Min Gyung Kang, Zhenkai Liang, James Newsome, Pongsin Poosankam, Prateek Saxena, BitBlaze: a new approach to computer security via binary analysis, in: Proceedings of the 4th International Conference on Information Systems Security (ICISS), December, 2008.

[22] Rolf Rolles, Unpacking virtualization obfuscators, in: Proceedings of Workshop on Offensive Technologies, August, 2009.

[23] K. Steven, Cracking SpyEye 1.3.x, 2011. <http://xylibox.blogspot.com/2011/08/cracking-spyeye-13x.html>.

[24] Malware at Stake, Blasting SpyEye C&C – SQL Injection Wins, 2011. <http://secniche.blogspot.com/2011/08/blasting-spyeye-c-sql-injection-wins.html>.

Aditya K. Sood is a senior security researcher/consultant and Ph.D. candidate at Michigan State University. He has already worked in the security domain for Armorize, COSEINC and KPMG. He is also a founder of SecNiche Security Labs, an independent arena for cutting edge computer security research. He has presented his research at leading security conferences and is a regular contributor to various security journals and magazines.

Richard J. Enbody, Ph.D., is associate professor in the Department of Computer Science and Engineering at Michigan State University (USA) where he joined the faculty in 1987. Enbody has served as acting and associate chair of the department and as director of the computer engineering undergraduate program. His research interests include computer security; computer architecture; web-based distance education; and parallel processing.

Rohit Bansal is a senior security researcher at SecNiche Security Labs, an independent security research arena. On the professional front, he has worked with large consulting firms including KPMG, PWC, L&T. He has also coauthored papers on hacking and malware.