MALWARE ANALYSIS 2

DISSECTING THE NGR BOT FRAMEWORK: IRC BOTNETS DIE HARD
Aditya K. Sood, Richard J. Enbody
Michigan State University, USA
Rohit Bansal
SecNiche Security, USA

IRC-based botnets [1, 2] have become the preferred choice of bot herders for remotely managing bots. IRC networks provide anonymity during communication, which makes tracking their activity more difficult. The IRC network is used for sharing files, controlling network activity and sending distributed commands to networks of infected machines. The IRC network is comprised of dedicated servers that use specific communication channels. As a result, it is possible to control a large number of infected machines through a centralized space (IRC channel) to create a complete botnet. Basically, the bot is compiled with a configuration which has a predefined IRC channel name. Once the bot is installed, it connects back to the IRC channel and the bot herder is able to send commands through that channel to operate the bot remotely. IRC-based botnets are popular for conducting Distributed Denial of Service (DDoS) attacks [3, 4]. However, the latest variants of IRC-based botnets, such as NGR, are designed to steal sensitive information by exploiting browser processes and acting as a backdoor. In this paper, we discuss the framework of the NGR bot version 1.1.0.0 which is growing in prominence in the malware world. The workings of the IRC bot are presented in Figure 1.

UNDERSTANDING THE FRAMEWORK

In this section, we present the design and analysis of the NGR bot framework. The framework consists of the bot executable with built-in modules. The design of the framework is discussed next.

RING 3 BOT

The NGR bot is a ring 3 bot that works in user space. The bot is written using the standard Visual Studio development kit. It has the characteristics of user-land rootkits [5, 6] and follows a similar process of DLL injection and hooking.

![Diagram of NGR bot framework](image)

**Figure 1: Working of IRC bot.**
to infect the running processes in the system. The DLL injection is a system-wide operation and is not restricted to the browser process. That is, the NGR bot is capable of injecting content into any process in the system. The NGR bot exhibits some properties that are found in third generation botnets such as Zeus and SpyEye. The bot is designed to infect 32-bit processes and does not support 64-bit injections at this point. However, the bot can successfully be installed in versions of Windows including XP, Vista, Windows 7 and Windows Server. From the design of the NGR bot we expect that upcoming versions will include full support for injecting into 64-bit processes. Figure 2 shows the layer model of the Windows operating system and where the bot infects it.

![Diagram of Windows layer model and bot infection](image)

**Figure 2: Ring 3 layer of OS and bot infection.**

Like all bots, the NGR bot is designed to operate in a covert manner. It can be installed in Windows using any user account without administrative privileges. The bot is capable of sending back information about the access rights of the victim’s account on the system through the IRC channel. The NGR bot’s primary installation location is the user’s application directory. The bot installs itself there under a randomly generated name so it varies from machine to machine. The bot is activated in the machine after a reboot. It ensures execution by creating a registry key with a path directly to the bot’s binary (under its random name). As the system is restarted, the NGR bot connects back to the IRC channel through a relay server and then becomes ready to receive commands, as shown in Figure 3.

The bot sends information back to the server in the following format:

```
{n([XFA])|XFKd}
```

The bot herder can decipher the information based on the string shown above. The ‘n’ parameter indicates that the bot is installed on a new machine. The ‘RU’ indicates that the victim machine is located in Russia. The ‘XP’ string indicates that the infected operating system is Windows XP. The ‘a’ parameter shows that the bot is installed using an account that has administrative rights. The string ‘kdkfkds’ is the identity of the bot generated in a pseudo random manner. The following shows how the bot sends information back to the IRC channel:

```
<new>{COUNTRY}{OPERATING SYSTEM}{USER TYPE}{random letters}
```

Figure 4 shows the NGR bot in action. The ‘-version’ command can be used to show the identity information from the NGR bot.

The bot is designed to communicate over SSL using an IRC channel. In order to set up the SSL communication, the IRC server must be configured to initiate an encrypted channel with the NGR bot. The bot binary is optimized and has a built-in module for testing the connection speed. The ‘+speed’ command on the IRC channel can be used to measure the speed for exchanging data. All the Inter Process Communication (IPC) among processes is encrypted. The NGR dropper deletes itself once the bot is successfully unpacked and installed on the victim’s machine. The bot also changes the extension of files to ‘.exe’ so that other files such as *.vbs can be executed successfully to trigger infections.

We have described the NGR bot functionality above. In the next sections, we will present the rest of the NGR bot framework.

**GRABBERS – BROWSERS, FTP & POP3**

The NGR bot uses a form-grabbing module to extract sensitive information from the victim’s machine. The latest version of the NGR bot is robust enough to execute hooking in both Internet Explorer and Mozilla Firefox. IE uses wininet.dll, whereas Firefox uses nspr4.dll for HTTP communication. The NGR bot hooks various functions in these libraries and captures the GET/POST requests to extract credentials in the forms. Since it uses form grabbing, the NGR bot does not have a keylogging module. We presented details of the form-grabbing technique in [7]. Additionally, the NGR bot has a built-in FTP grabber module that hooks the ws2_32.dll functions to extract the credentials for various FTP servers. Finally, a POP3 grabber module works in a similar way to the FTP grabber module.

Figure 5 shows how the NGR bot sends credentials back to the IRC server.

On the IRC channel, the ‘-logins’ and ‘-stats’ commands show the number of fetched credentials and related stats.

**SPREADERS**

Spreader modules are used to spread the botnet across a variety of interfaces of the victim machine. These interfaces include USB devices and Instant Messengers (IMs) such
as MSN. The NGR bot has the following built-in spreaders.

**USB spreader**: The NGR bot infects USB devices and replicates itself on them. Once the victim machine is infected with the bot, the built-in USB spreader module waits for USB devices to be inserted and tries to infect them. The USB spreader module uses a linking technique in which .lnk files are inserted into the USB drive with a path to the NGR bot. A desktop.ini file is also created to hide the folder in which the bot resides. In addition, the NGR bot is able to infect USB drives using an obfuscated autorun.inf method. This method can be activated using the ‘-mod usb’ command on the IRC channel. This module works on all versions of Windows.

**MSN spreader**: The NGR bot also has a built-in MSN spreader module that hooks the ws2_32! send function to detect MSN messages being sent. The spreader module monitors the MSN communication channel and waits for a certain set of messages so that it can start injecting illegitimate messages. The spreader can successfully inject processes such as msnmigr.exe, wcomm.exe, pidgin.exe and mmsigs.exe using protocols msn10 and msn21. The ‘~msn.in’ and ‘~msn.set’ commands are defined in the NGR bot for this purpose.

**DNS MODIFIER**

DNS entry modifications are an important part of the NGR bot. Generally, DNS modification can be achieved in two ways:

- **Updating the hosts file** - The bot can update the entries
in the hosts file in order to manipulate the DNS resolution.

- **Hooking dnsapi.dll** – The bot can hook the required DNS DLL file and manipulate the entry present in the rule file.

The NGR bot hooks the dnsapi.dll file to modify the DNS entries on the victim’s machine. This module is incorporated in the NGR bot so that virus detection websites such as VirusTotal, Kaspersky, and so on can be blocked on the host. The DNS modifier is also capable of setting a DNS redirection so that a legitimate website’s address is mapped to an illegitimate one. This feature is used to serve malware. The ‘-chdns’ command is used to perform those actions. Figure 6 shows the DNS modifier in action.

**PROACTIVE DEFENCE (PDEF+) AND RUSKILL**

Bot wars are on the rise as large botnets compete to infect the same computers. The NGR bot has a built-in module that kills other installed IRC bots in the system. PDEF+ is an active threat detection module that monitors and scrutinizes the various APIs and the file system to detect and remove infections. This module can detect and block malware that has been distributed using USB drives, IRC bots and browser exploits. The NGR bot has modified this module to detect and kill the butterfly bot, butterfly flooder, GBOT and all other IRC-based bots.

The built-in Ruskill module is designed to stealthily execute files. The NGR bot has the built-in command ‘-baja’ which is used to download malicious executables from a remote website. The downloaded binary executes automatically and triggers infection. The Ruskill module monitors the downloaded binary and flags it, then deletes the binary on system reboot. This functionality is widely used by IRC bots to remove downloaded files after execution.

**DENIAL OF SERVICE (DoS)**

Denial of Service is a primary functionality of IRC bots so the NGR bot is well equipped with DoS modules:

- **SYN flooder**: The bot sends a continuous flow of TCP packets with the SYN flag. The SYN flood can take down web servers that other flooders fail to.

- **UDP flooder**: The bot sends a continuous flow of UDP packets to take down the target. This module is designed to target small networks.

- **Slowloris**: The bot has a built-in Slowloris module [8]. This conducts DoS attacks against Apache web servers in which the module opens many connections to the web server and holds them open for a long period of time. As a result, the target website’s connection pool becomes exhausted because the connection remains open and no new connections can be served by the web server.

Figure 7 shows the working of a Slowloris module.

**CONCLUSION**

In this paper, we have presented a detailed framework of the NGR bot and the different types of modules it supports. The NGR bot has been widely used to trigger infections and compromise machines. Our analysis has revealed that the bot is very effective and capable of running in a concealed manner. Looking at the development of the NGR bot, we can expect further advancement such as support for hooking 64-bit processes, Facebook IM spreaders and so on in the
near future. The sophisticated framework of the NGR bot indicates that IRC-based botnets will continue to be a hard nut to crack.

REFERENCES


Appendix – NGR bot commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Options</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>~baja</td>
<td>&lt;url&gt;,&lt;md5&gt;,&lt;r&gt;,&lt;n&gt;</td>
<td>Bot downloads and executes a file from the specified URL.</td>
</tr>
<tr>
<td>~updlt</td>
<td>&lt;url&gt;,&lt;md5&gt;,&lt;r&gt;</td>
<td>Bot updates its file, but the update does not take effect until the system is restarted.</td>
</tr>
<tr>
<td>~mata</td>
<td></td>
<td>Bot disconnects from the IRC server.</td>
</tr>
<tr>
<td>~11mpia</td>
<td></td>
<td>Bot removes itself from the system.</td>
</tr>
<tr>
<td>~mudo</td>
<td>[state]</td>
<td>Enables/disables all output to IRC relating to commands and features.</td>
</tr>
<tr>
<td>~version</td>
<td></td>
<td>Bot displays its version, customer name, the MD5 hash of its file, and its installed file path.</td>
</tr>
<tr>
<td>~visit</td>
<td>[url][state]</td>
<td>Bot creates a browser instance and visits the specified link.</td>
</tr>
<tr>
<td>~rc</td>
<td>&lt;n&gt;,&gt;g&gt;</td>
<td>Bot disconnects from the IRC server and waits 15 seconds before reconnecting.</td>
</tr>
<tr>
<td>~move</td>
<td>&lt;rule&gt;,&lt;options&gt;&lt;channel&gt;,&lt;key&gt;</td>
<td>Bot joins the specified channel.</td>
</tr>
<tr>
<td>~p4rt</td>
<td>&lt;rule&gt;,&lt;options&gt;&lt;channel&gt;</td>
<td>Bot leaves the specified channel.</td>
</tr>
<tr>
<td>~pais</td>
<td>&lt;rule&gt;</td>
<td>Bot joins the channel for its country.</td>
</tr>
<tr>
<td>~mix</td>
<td>&lt;rule&gt;</td>
<td>Bot leaves the channel for its country.</td>
</tr>
<tr>
<td>~speed</td>
<td></td>
<td>Bot determines the average upload speed.</td>
</tr>
<tr>
<td>~mod</td>
<td>[module],[state]</td>
<td>Enables/disables modules that use hooks.</td>
</tr>
<tr>
<td>~stats</td>
<td>&lt;l&gt;,&lt;l&gt;,&lt;s&gt;</td>
<td>Retrieves statistics for spreading and/or login grabbing.</td>
</tr>
<tr>
<td>~logins</td>
<td>&lt;site,-c&gt;</td>
<td>Retrieves all grabbed and cached logins.</td>
</tr>
<tr>
<td>~stop</td>
<td></td>
<td>Bot ends all running flood tasks.</td>
</tr>
<tr>
<td>~ssyn</td>
<td>[host],[port],[seconds]</td>
<td>SYN flooder.</td>
</tr>
<tr>
<td>~udp</td>
<td>[host],[port],[seconds]</td>
<td>UDP flooder.</td>
</tr>
<tr>
<td>~msn.int</td>
<td>[interval]</td>
<td>Sets the number of MSN messages in a conversation before one is changed with the spreading message.</td>
</tr>
<tr>
<td>~msn.set</td>
<td>[message]</td>
<td>Sets the message that will be used for MSN spreading.</td>
</tr>
<tr>
<td>~chdns</td>
<td>[url][domain1 &lt;domain2lip2&gt;l]</td>
<td>Bot blocks access to or redirects the specified domain/IP address.</td>
</tr>
</tbody>
</table>