

# Crxbot Malware Analysis

Melissa E.

This is an analysis of the executable with md5 59a95f668e1bd00f30fe8c99af675691 (after unzipping) as found on <http://www.malwarechallenge.info/challenge.html>.

- Describe your malware lab.

I own several computers running Windows, Linux, and OSX, but none of them are quite powerful enough to run a virtual machine very well (as they're all tiny), so I very bravely did this work on a production machine. The computer runs Windows XP SP3, with standard Microsoft development and debugging software (including Sysinternals), as well as an assortment of free debugging tools. Specifically, I used Process Explorer, Process Monitor, CFF Explorer, IDA 4.9, PeID, UPX, XVI32, and Wireshark.

- What information can you gather about the malware without executing it? / What is its intended purpose? / Is the malware packed?

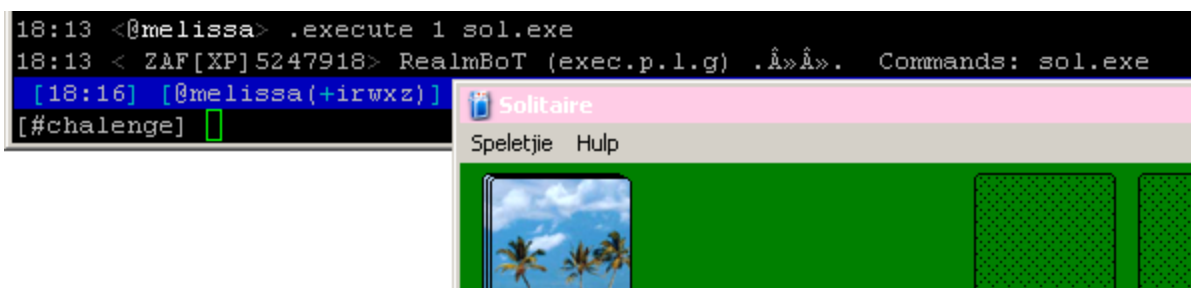
Looking at the strings inside (using a hex editor such as XVI32 or the Unix strings program), one can see IRC commands (PRIVMSG, JOIN, etc) as well as things that allude to DDoSing (-updflood), implying that this is a run-of-the-mill IRC bot that wants to donate the victim's bandwidth to very unworthy causes. As expected, the import table lists the Winsock DLL, though the table appears to be corrupt and not correctly displaying all imported functions. CFF Explorer lists the executable type as UPX 3.0, and PeID claims the version as a little lower- but the standard UPX unpacker cannot restore it, throwing an exception about the file being subsequently modified. A disassembly with IDA shows a small segment of sensible code at the entry point, which ends by jumping into quite a bit of gibberish- the packed malicious code. It could be manually unpacked, but it'd be a lot faster to let the malware do that for us.

- Describe the malware's behavior. / Describe its C&C. / Describe its commands. / Is it possible to take it over?

Watching it in Process Explorer, you can see as it copies itself to C:\WINDOWS\WinSec32.exe (same md5) and replaces its own process with this new one, passing the path of the old instance as an argument. (This is presumably to stop it from copying itself again.) Process Monitor keeps an eye on it as it installs itself for autostart under HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run with an

innocent name like "Svchost local services." I started up Wireshark to watch exactly what data it transmits over the Internet. It attempts to connect to testirc1.sh1xy2bg.net, which is now down, so I set up my hosts file (at %system%\drivers\etc\hosts) to redirect that to a private IRC server. The software connects to the IRC server with a pseudo-random username like ZAF[XP]8699904, creates channel #challenge (sic), and sets the topic to ".asc vnc 100 0 0 -r -b". It sets the requirement of mode "+v" (voice) to speak to it in the channel, which only an operator can give. (After discovering the source code much later in the process, I learned that the [XP] in its nick indeed refers to the operating system. The program also supports encoding in the victim's country, but that seems to be turned off.)

At this point, we can easily see into its unpacked memory (using, for example, the Strings tab provided by Process Explorer), and among other things it appears to have an entire password-cracking dictionary contained inside. As revealed by the command strings and response strings in memory, it supports complex functionality such as a homemade FTP server, as well as many little commands such as showing IP and host info, executing arbitrary programs, and of course DDoS. The name and version banner of the FTP server it starts is "NzmxFtpd/TxmxFtpd 0wns j0". It contains the URLs "http://nivdav.net/Winsec32.exe" and "http://www.W32-gen.us" in memory, as well as a lot of very rude status messages for when a botmaster fails to authenticate successfully- in fact, it swore at me very violently when I figured out that the login sequence is ".login gemp123", because my host did not match the host it was expecting, which is "legalize.it". (At first, I wasn't sure what exactly was the correct method of talking to the bots, but I figured it out eventually by trial and error.) Fortunately, there are ways of getting IRC to lie about your hostname to other users (with the help of administrators), so soon I was "melissa@legalize.it" and it was happy to talk to me. A moment later...



The 1 is a boolean that means "visibly", as it assumes you don't want the user noticing anything by default.

The author signed his work "Crxbot Alias REalmbot -by Lindem-". He's not very consistent, as everywhere else in the binary it is called RealmBoT. One can only wonder why it has the same name as the Realmbot used to cheat at World of Warcraft.

- Is it possible to find the sourcecode of this malware?

[http://darksun.ws/download/Bots/Crx-realmBot.VNC.exploit.and.RFI-\(rfi.not.tested\).rar](http://darksun.ws/download/Bots/Crx-realmBot.VNC.exploit.and.RFI-(rfi.not.tested).rar)

I found it by appropriate application of Google and shady friends, the same way you find anything.

```
char prefix = '.';
char botid[] = "1";
char version[] = "Crxbot Alias REalmbot -by Lindem-";
char password[] = "pass";
char server[] = "nonna.sytes.net";
char serverpass[] = "";
char channel[] = "#vml";
char chanpass[] = "pass";
char filename[] = "nodkrn23.exe"; // copy in Windirectory
char keylogfile[] = "lol.dat";
char valuenam[] = "Microsoft Svchost local services";
char nickconst[] = "[w32]gen-";
char szLocalPayloadFile[]="lol.dll";
char modeonconn[] = "+i-x+s";
char exploitchan[] = "#vml";
char keylogchan[] = "#vml";
char topics[] = ".asc vnc 75 0 0 -r -b"; //bot on join change topic
Fixxed
```

Clearly, this is the origin of the malware on hand. After finding the source code, I could now see that the references to VNC were in relation to an exploit against it, and the fact that the values above are stored in the binary in order makes it really easy to find the specific customized ones in memory.

- How would you write a custom detection and removal tool for this program?

It has a very large amount of unique strings (author's thoughtful signature, etc) that would readily identify this program and subtle variations thereof. Since the packing used wasn't very tight (one could easily pick out entire strings or large fragments thereof in the compressed binary), you probably wouldn't even need a custom unpacker. The program takes no steps to protect itself; you can simply kill the process, delete its executable and its registry key, and be done with it.