J Comput Virol (2007) 3:103–111
DOI 10.1007/s11416-007-0043-3

EICAR 2007 BEST ACADEMIC PAPERS

Bot countermeasures

Vinoo Thomas · Nitin Jyoti

Received: 12 January 2007 / Accepted: 12 March 2007 / Published online: 5 April 2007

Abstract Administrators must have faith in the security products installed today at the desktop and gateway levels of their networks. They have faith that these technologies provide a reasonable protection against most worms from infecting and spreading within the internal network. However an overdependence on the very security products installed leaves many standing potentially exposed when the network is hit with an undetected piece of malware. For any organization, internal bot infections cause serious repercussions, including loss of man hours and downtime. The average cost[1] of such disasters runs into the tens of thousands of dollars. The most recent cases are the W32/Mocbot,[2] W32/Mytob,[3] and W32/Zotob[4] outbreaks, which caused widespread havoc within several large corporate networks. Having an early warning system in place that proactively alerts and captures bot-like activity on an internal network goes a long way in the containment and isolation of the source of infection or attack. Furthermore, no organization should rely solely on a security vendor's information or solution. Organizations must also have in place their own information gathering methods, techniques, and defences. This paper describes setting up an IRC honeypot on a network, using minimal resources and requiring little maintenance. The honeypot serves, as an early warning system to proactively alert on bot-like activity. We also discuss using the internal IRC honeypot to disrupt the flow between bots and their command and control (C&C) server. This can allow the network administrator to gain control over infected machines and assist in removing bots from infected machines.

Vinoo Thomas and Nitin Jyoti are Virus Researchers with McAfee Avert Labs, based in Bangalore, India.

V. Thomas (✉) · N. Jyoti
McAfee Avert Labs, c/o McAfee Software (India) Pvt. Ltd.,
Embassy Golf Links Business Park, Pine Valley,
Bangalore 560071, India
e-mail: vinoo@avertlabs.com

N. Jyoti
e-mail: nitin@avertlabs.com

1 Background

Microsoft's Patch Tuesday has become synonymous with IT administrators having to schedule downtime and ensure that every workstation and server on the network is fully patched against newly discovered Microsoft vulnerabilities. These vulnerabilities have sometimes taken as long as 200 days to be fixed and for a patch to be rolled out.[5] Now patches for reported vulnerabilities are released regularly on the second Tuesday of every month.

Virus authors, on the other hand, have been at the cutting edge for including exploit code in their creations for every vulnerability reported. The chart shows the time between a vulnerability being reported and how long it took for virus authors to incorporate it into a worm candidate. Over the years, the window between exploit discoveries to its incorporation into a worm candidate has shrunk from months, to weeks, to 0-day. This leaves administrators with very little time to schedule and deploy patches to all servers and workstations on their network. And during this vulnerable time frame if the network is hit with a bot that uses a 0-day vulnerability, an organization would be faced with a potential outbreak scenario.

[1] http://www.pwc.com/uk/eng/ins-sol/publ/pwc_dti-fullsurveyresults06.pdf.

[2] http://vil.nai.com/vil/content/v_136637.htm.

[3] http://vil.nai.com/vil/content/v_132158.htm.

[4] http://vil.nai.com/vil/content/v_135433.htm.

[5] http://news.com.com/2100-1002_3-5158625.html.

Patch	Malware	Patch availability	Worm attack Date	Number of days for worm to appear
MS01-020	Nimda	Oct 17, 2000	Sept 18, 2001	335 Days
MS02-061	Slammer	July 24, 2002	Jan 25, 2003	185 Days
MS03-026	Blaster	July 16, 2003	Aug 11, 2003	26 Days
MS04-011	Sasser	Apr 13, 2004	Apr 30, 2004	17 Days
MS05-039	Zotob	Aug 09, 2005	Aug 14, 2005	5 Days
MS06-040	Mocbot	Aug 08, 2006	Aug 12, 2006	4 Days

Most administrators do not have the flexibility to deploy patches immediately to the network for policy reasons. For example, the organization could be using legacy software, which could break if a new service pack was applied; keeping these legacy applications running takes precedence over applying the latest Windows hot fixes. System administrators, especially those who work in hospitals and other mission critical jobs, don't have the luxury of doing a Windows update!

To add to these woes, every once in a while a hot fix from Microsoft breaks something in the operating system[6] or adversely affects other applications.[7] Once a patch is rolled out via WSUS (Windows Server Update Service[8]), it cannot be rolled back centrally; a faulty patch from Microsoft can prove costly for the organization. Administrators need more time to deploy these hot fixes in a test environment and QA them properly before deploying them to the enterprise.

In the scenario we just described, until all machines are fully patched, the administrator is dependent on the antivirus capability to detect the latest threats and their SLA (Service Level Agreement) with their antivirus vendor to help them in dire situations. Antivirus is still a reactive solution and overdependence on signature-based detection can be risky when faced with an undetected piece of malware spreading fast on the internal network.

2 Bots pangs

What differentiates a bot from a worm? A bot spreads precisely under the control of the attacker also known as the "botnet herder" who commands his army of bots. In stark contrast, a worm spreads mindlessly out of control. Historically, ever since the appearance of the first Internet worm dubbed, the "Morris Worm",[9] we have seen worms spreading much faster than what their authors intended.

Why do bot infections pose the greatest security threat to organisations today? A workstation infected with a worm could be a nuisance, but a workstation infected with a bot means that the hacker has direct access to the internal network from behind the firewall. And this insider access gives the hacker the potential for committing far more serious damage.

Bots are a constantly evolving family of threats; they have been seen to exploit vulnerabilities quicker than patches can be deployed. Fuelled by financial incentives and readily available modular source code, bots have become multi blended and have developed instant messaging (IM), mass-mailing (MM), and peer-to-peer sharing (P2P) capabilities. They also drop rootkits to conceal their presence on infected systems. Once a network is infected, cleaning can be difficult for these reasons:

- If machines are unpatched, a cleaning tool or an antivirus program is not going to be of much help. Reinfection would occur almost immediately as long as there are other infected machines on the network.
- Bots autosecure the system by disabling administrative shares, remote registry, telnet, and DCOM functionality on infected machines in order to make the machines inaccessible via the network for remote investigation or cleanup.
- The volume of network traffic created by bots makes it impossible for an administrator to do a Windows update on affected machines.
- Bots tend to kill antivirus and firewall processes and this makes cleaning a system difficult even with updated signatures, as the antivirus is killed upon launch.
- Bots modify registry entries so they remain active even when the infected machine is booted in Windows safe mode.

These scenarios can be dealt with in the shortest time possible if an Internet Relay Chat (IRC) server is already set up internally. This IRC server could be tweaked to act as a command and control (C&C) centre to combat the rogue bots; an administrator could issue centralized commands to stop or uninstall these bots on the network [9].

[6] http://news.com.com/Critical+Microsoft+fix+breaks+some+Net+connections/2100-1002-6086130.html?part=dht&tag=nl.e703.

[7] http://news.com.com/Microsoft+patch+can+cause+IE+trouble/2100-1002_3-6106039.html?tag=nefd.top.

[8] http://www.microsoft.com/windowsserversystem/updateservices/default.mspx.

[9] http://en.wikipedia.org/wiki/Morris_worm.

3 IRC for command and control

The potential for profit is the biggest reason for the exponential growth of bots and botnets. The underground community is always willing to share its wares and expertise with anyone for the right price [8]. Because of the open-source nature of bots, information is readily available on the Internet for download, and explains how to incorporate exploit code, compromise systems, and evade antivirus detection.

Setting up a botnet requires minimal technical skills, and IRC is the preferred medium used by botnet herders to control a botnet. An attacker can either use a public IRC server for C&C purposes or build their own. IRC allows an attacker to control infected machines sitting behind NAT, and the bot can be configured to connect back to the C&C server listening on any configured port.

To maintain high availability of the C&C server, hackers often configure the bot to connect via a fully qualified domain name (FQDN). If a hard-coded IP address is used and that IP address is disconnected from the Internet (by the ISP or authorities), the botnet herder would lose control of the botnet. Thus herders use a FQDN, and both IP address and the FQDN would need to be removed before the botnet herder would lose total control of the botnet. Hackers nowadays use FQDNs provided by dynamic DNS providers on the Internet; this abuse of dynamic DNS has given hackers the flexibility to avoid a complete shutdown of their infrastructure.

Upon infecting a host, the bot would home in on the attackers' IRC server and attempt to join a specified channel. Once it successfully joins the channel, an attacker can pass commands to the bot. Usually channel topics are preset so that once a bot joins the channel, it immediately executes whatever command is set. This is the reason why bots disconnected from the Internet do not replicate unless self-spreading was hard coded while compiling the bot.

To disrupt any communication between bots and their C&C server, most organizations have implemented firewall rules blocking standard IRC ports 6666-6669. Botnet herders have countered this by making their bots connect back on commonly used TCP ports 21, 80, or 443, which most corporate firewalls allow.

To tip off administrators about any IRC connection initiated from the local area network irrespective of the destination port, one would need software or an appliance that inspects traffic at the gateway level. IRC connections are usually transmitted in clear text and have distinct commands that are passed between the client and server for communication.

A suggested method is running a packet sniffer on the monitor port of the switch and setting a rule to trigger an alert for IRC traffic. Following is a sample sniffer capture when an IRC bot connects to an IRC server.

```
NOTICE AUTH :*** Looking up your hostname
NOTICE AUTH :*** Found your hostname, cached
NOTICE AUTH :*** Checking Ident
NOTICE AUTH :*** No ident response

NICK [P00|USA|89252416]
USER 2K-2833 * 0 :VINOOVM2000
```

The bot attempts a connection to the IRC server announcing the operating system version and hostname of the infected machine as identification strings for the UserID and Name fields.

```
:irc.botspot.com 001 [P00|USA| :Welcome to the BotSpot IRC Network
  [P00|USA|
:irc.botspot.com 002 [P00|USA| :Your host is irc.botspot.com, run-
  ning version beware1.5.7
:irc.botspot.com 003 [P00|USA| :This server was created Tue Jul 13
  2006 at 20:36:07 GMT
:irc.botspot.com 004 [P00|USA| irc.botspot.com beware1.5.7 dgikoswx
  biklmnoprstv

:irc.botspot.com 005 [P00|USA| MAP SILENCE=15 WHOX WALLCHOPS
  WALLVOICES USERIP CPRIVMSG CNOTICE MODES=6 MAXCHANNELS=10
  MAXBANS=45 :are supported by this server
:irc.botspot.com 005 [P00|USA| NICKLEN=9 TOPICLEN=160 AWAYLEN=160
  KICKLEN=160 CHANTYPES=#& PREFIX=(ov)@+ CHANMODES=b,k,l,rimnpst
  CASEMAPPING=rfc1459 :are supported by this server
:irc.botspot.com 251 [P00|USA| :There are 1 users and 1 invisible
  on 1 servers
:irc.botspot.com 254 [P00|USA| 1 :channels formed
:irc.botspot.com 255 [P00|USA| :I have 2 clients and 0 servers
:irc.botspot.com NOTICE [P00|USA| :Highest connection count: 2
  (2 clients)

:irc.botspot.com 375 [P00|USA| :- irc.botspot.com Message of the day
:irc.botspot.com 372 [P00|USA| :- Welcome to BotSpot -- McAfee Avert
  IRC Honeypot
:irc.botspot.com 376 [P00|USA| :End of /MOTD command.
```

The IRC server displays a welcome banner text and conveys information back to the client regarding features supported by the server, numbers of channels, and connected clients and message of the day if any.

```
MODE [P00|USA|89252416]
JOIN #101 0bfu$cation
:[P00|USA|!~2k-2833@192.168.1.59 JOIN :#101
:irc.botspot.com 353 [P00|USA| = #101 :[P00|USA| @Avert
:irc.botspot.com 366 [P00|USA| #101 :End of /NAMES list.
```

The bot attempts to try and join the attacker's channel using a hard-coded password "0bfu$cation". It also retrieves information about the channel operator and names of other bots in the channel. Once successfully connected to the channel, it receives the topic of the channel and interprets it as a command.

A typical channel topic could be set as follows so that without human intervention, this command is automatically passed to the bot when it joins.

```
.advscan netapi 200 5 0 -r --s

[SCAN]: Random Scanner started : 192.168.x.x:445 connect
        timeout 5 seconds 0 with 200 threads.

[SCAN]: IP: 192.168.164.xxx:445, Scan thread: 0, Sub-thread: 1.
[SCAN]: IP: 192.168.160.xxx:445, Scan thread: 0, Sub-thread: 2.
[SCAN]: IP: 192.168.122.xxx:445, Scan thread: 0, Sub-thread: 3.
[SCAN]: IP: 192.168.143.xxx:445, Scan thread: 0, Sub-thread: 4.
[SCAN]: IP: 192.168.121.xxx:445, Scan thread: 0, Sub-thread: 5.
```

The preceding channel topic directs the bot to perform the following functions: The second example of a possible topic instructs the bot to download a binary from a remote Web server and execute it (parameter 1). This could be used to dynamically update the bot with an enhanced version upon connecting or to download and execute further malware.

```
.down http://webserver/update.exe c:\a.exe 1
```

If the channel topic does not contain any command for the bot, it sits idle in the channel, awaiting a command (Table 1).

Experienced botnet herders prune their IRC C&C servers to meet their exact needs. Some of the commonly observed modifications include stripping down functionality like informative commands, server performance optimizations, reporting low number of connected users, even though thousands may exist and triggering alerts whenever unauthorized users or commands are observed [5].

In the packet capture example just described, we observe certain unique keywords specific to IRC. The first thing that happens in IRC is that the client sends the commands "NICK" and "USER" in either order.

By examining packets from the mirror port of the switch, one can generate alerts for IRC traffic originating from the internal network. To implement this using a Windows machine, the packet sniffer CommView[10] is connected to the mirror port of the switch. With CommView, one can use Boolean logic to create custom rules to trigger an alert on a specified packet occurrence. Figure 1 shows how a combination of the keywords "NICK" and "USER" is used to trigger an alert every time IRC-like traffic is observed. This rule set is effective as it triggers irrespective of whichever port a bot would use to connect to an IRC server. Once a packet is identified by the rule set, the sniffer can be configured to alert an administrator and capture all traffic for that session and dump it to a file, as shown in Fig. 2. The IRC session dump comes in handy during network forensics to reconstruct the sequence of events, typically when one has to replay captured network traffic.

From a captured IRC session, one can learn about the FQDN or IP address of the C&C server, the channel name, password to control the bot, and if any commands were passed back to the bot. With this information, we could approach the local CERT[11] authorities, or volunteer security groups like the ISOTF[12] or Shadow Server[13] that specialize in taking down botnets.

4 Analysing a captured bot sample

Bot herders use password authentication mechanisms in their bots to keep unauthorized users from controlling them. In bot binaries, the passwords are sometimes stored in clear text, but increasingly, newer methods have been devised to prevent analysis and disclosure to those in the IT security community.

In this paper we analyse an open-source bot popularly known as the Reptile Bot. A detailed analysis of a previous variant (covered by [7]) makes an interesting read. A commonly used method is for bot samples to be packed with

[10] http://www.tamosoft.com.

[11] http://www.cert.org.

[12] http://isotf.org.

[13] http://www.shadowserver.org.

Table 1 Table describing the .advscan command syntax

.advscan	Bot command to scan for vulnerable systems
netapi	Attempt to exploit vulnerable hosts using the MS06-040[a] exploit
200	The number of concurrent threads
5	The number of seconds to delay between scans
0	Perform the scanning activity for an unlimited time
−r	The IP addresses it attempts to scan would be generated randomly
−s	The scan would be silent and not report its findings back in the channel

[a] http://www.microsoft.com/technet/security/bulletin/MS06-040.mspx

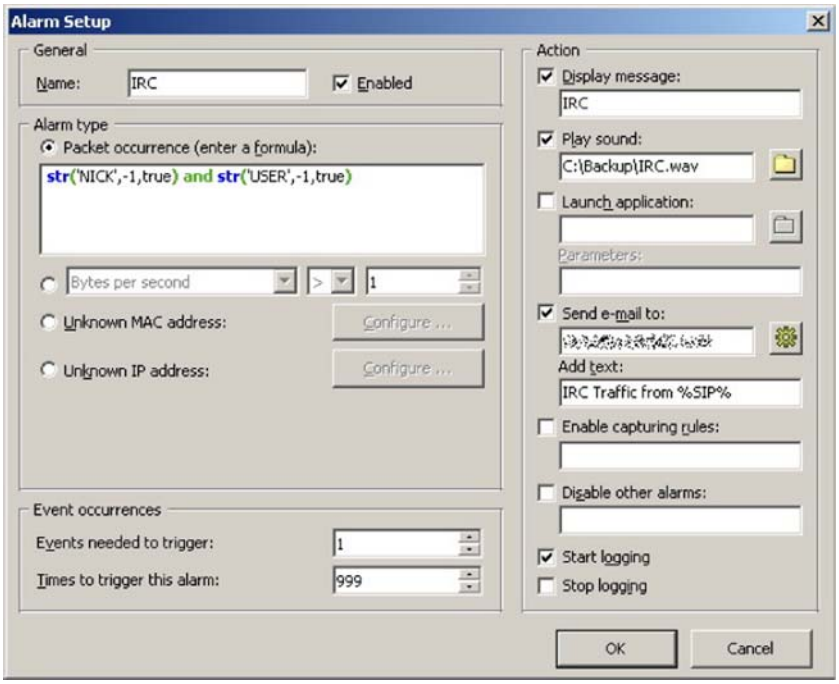

Fig. 1 Configuring CommView to trigger an alert

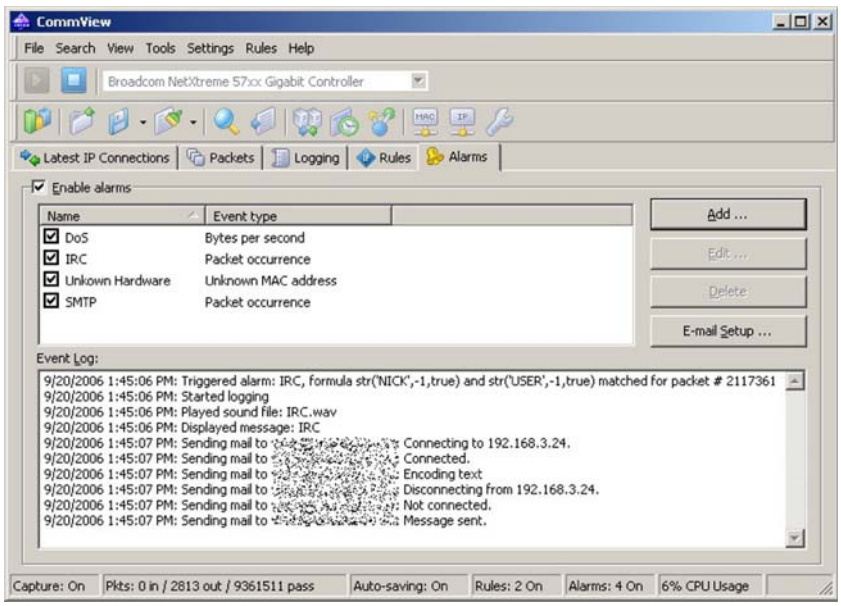

Fig. 2 Alert being sent to administrator on a rule match

the latest packers and encryption obfuscating code [4]. This makes it difficult to reverse-engineer or to pull valuable string data from a captured bot binary (Fig. 3).

A quick-and-dirty way to view interesting strings of a packed sample is to execute it and take a memory dump of a running process. By searching the memory dump of the bot

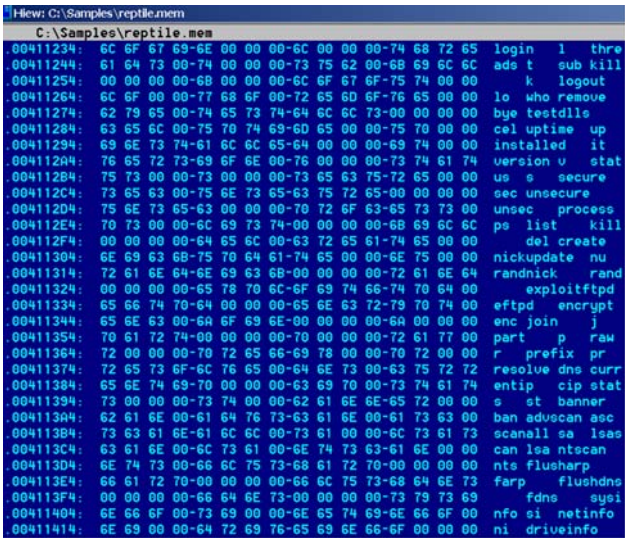

Fig. 3 Hex editor view of a captured bot sample

program for interesting strings, one can chance upon commands supported by the bot. The IRC server and channel it connects to is always hard coded within the bot.

Because a lot of bot code is reused, the commands and authentication mechanisms are becoming widely known. Some attackers modify the C&C language used by their bots so even if a competitor acquired the passwords, they would still need detailed knowledge or packet captures of command sequences to control the botnet.

As there is no standardisation of botnet commands, most attackers change their commands very often. In many cases, command-replies are even translated to their native language [1].

For example, attackers commonly change the login command. Instead of using the default "login," an attacker could easily change this to "nigol" or something more obscure, raising the bar for control over the bot. Other command names can be modified just as easily.

Additionally, intrusion detection systems (IDS) and intrusion prevention systems (IPS) use known bot commands as signatures for detecting bot-like activity on the network. These events have led to botnet-herders modifying the C&C language used by their bots.

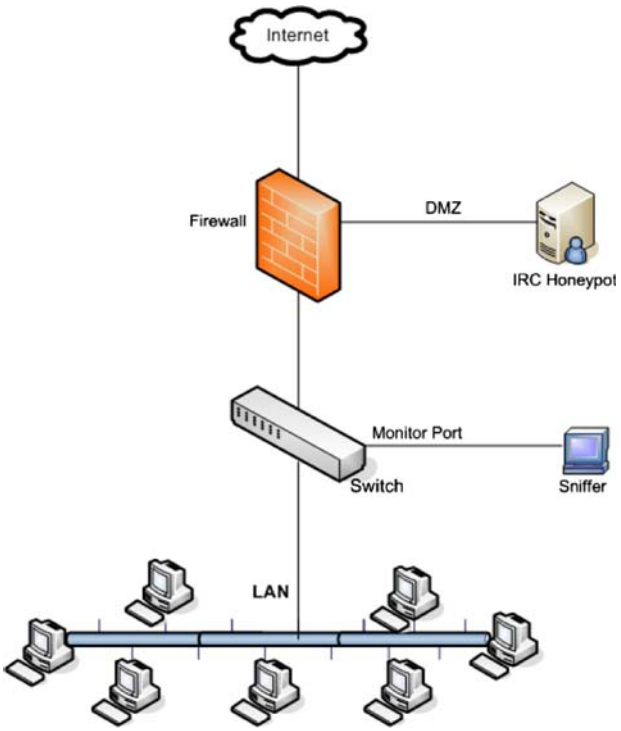

Fig. 4 Network topology placing the IRC honeypot in the DMZ

for Windows. It can be used with any existing IRC client and is easy to set up. Just extract the ∼200 KB zip file and it runs out of box. The server options are configured via config files and it runs entirely from the command line. Beware ircd is fully compatible with IRC protocol RFC1459.[15]

From our analysis so far, we already know which server and channel the bot would connect to. And the sniffer captures tells us which port the bot uses to connect. The IRC honeypot is configured to listen on the port the bot uses to connect and is placed in a DMZ.[16] network as shown in Fig. 4

At the firewall we create a rule to redirect IRC traffic to our in-house honeypot and ensure that we are logged in to this channel first, before the bot connects. This way, we will be the channel operator and will be able to pass commands to the bot.

An example iptable rule on the firewall could be:

```
iptables -t nat -A PREROUTING -i eth0 -s <affected network address> -p tcp
  --dport <port used by bot> -j DNAT --to <ip address of honeypot>
```

5 Using the IRC Honeypot to disrupt a Botnet

To set up an IRC honeypot, we can use any of the freely available IRC servers. In this paper, we use Beware ircd[14]

Alternately, for bots that use a FQDN to connect back to the attacker, we can create an "A" record[17] (an address record that maps a hostname to a 32-bit IP address) accordingly on

[14] http://ircd.bircd.org.

[15] http://www.faqs.org/rfcs/rfc1459.html.

[16] http://en.wikipedia.org/wiki/Demilitarized_zone_(computing).

[17] http://en.wikipedia.org/wiki/A_record\#Types_of_DNS_records.

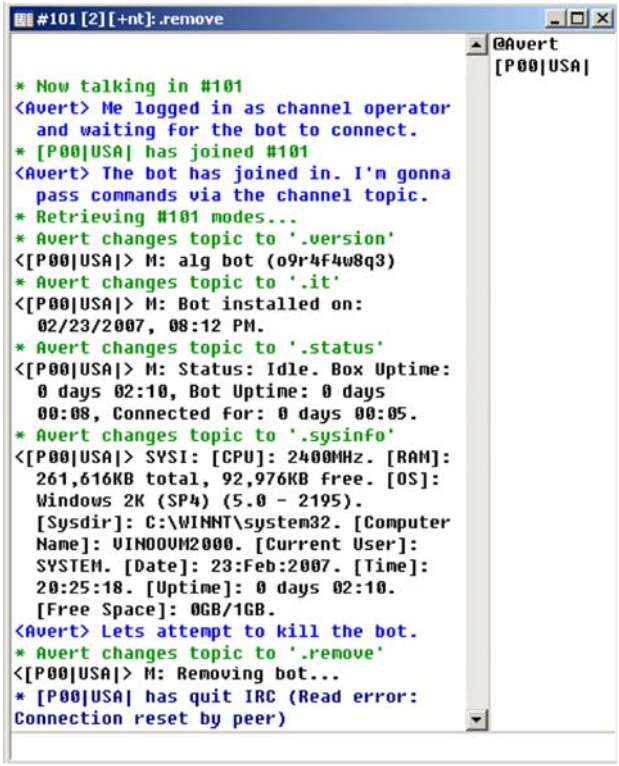

Fig. 5 Controlling a bot connected to the internal IRC honeypot

the internal DNS server. This record will direct any name resolution from machines on the internal network to the attacker's server name and point it to the IRC honeypot.

Upon execution, when the bot attempts to resolve and home in on the attacker's IRC server, it will instead get redirected to our in-house honeypot. Once the bot connects, we can send commands through the channel topic, private messages or simple chat messages. It's noteworthy that sending commands through the channel topic doesn't require you to log in to the bot first. Apparently people who can change topics are automatically trusted.

In this example we pass the desired commands to the bot using the channel topic. Every time the bot is kicked out of the channel, it would immediately try to reconnect. Upon reconnecting it would execute whatever command that is set as the current channel topic.

It is important to note that if no command is set on the channel topic, the bots on the infected network would connect to the channel and be idle. Once we get the hang of passing commands to the bot, if supported, we can issue an uninstall command, as shown in Fig. 5, and every bot that connects to this channel hereafter would uninstall itself from the infected machine. Earlier work by [2] goes deeper into syntactic details of issuing commands to various botnet families (Table 2).

6 The last laugh

Bot authors appear to have had the last laugh by ensuring that bots which are commanded to remove themselves from infected hosts leave the system in an unsecured state and open to infections. The following module found in a bot source code is an indication that this was done to ensure future infections even after the bot has been removed.

```
#ifdef WIN32

// should unsecure system as remove bot to allow recycling \newline
// Set EnableDCOM to "Y"

HKEY hkey=NULL;
DWORD dwSize=128;
char szDataBuf[128];
strcpy(szDataBuf, "Y");
dwSize=strlen(szDataBuf);

lRet=RegOpenKeyEx(HKEY_LOCAL_MACHINE, "Software\\
  \Microsoft\\OLE", 0, KEY{\_}READ, hkey);
RegSetValueEx(hkey, "EnableDCOM", NULL, REG_SZ, (unsigned
  char*)szDataBuf, dwSize);
RegCloseKey(hkey);

// UnSecure Shares

Execute("net.exe", "net share c$=c:\\ ");
Execute("net.exe", "net share ipc$");
Execute("net.exe", "net share admin$");
```

Table 2 Table describing sample commands for the Agobot, Sdbot, and Spybot families

Command	Description	Example
bot.die	Terminates the bot	<Agobot> .bot.die <– Agobot has quit (Read error: Connection reset by peer)
bot.quit	Causes the bot to quit IRC and terminate itself	<Agobot> .bot.quit <– Agobot has quit (Read error: Connection reset by peer)
bot.remove	Completely removes the bot from the system	<Agobot> .bot.remove removing bot... <– Agobot has quit (Read error: Connection reset by peer)
die	Causes the bot to immediately close on the host's computer. This will also kill all threads, so be careful. Try to use 'quit' if possible instead of die, because die performs no cleanup.	<[sdbot]> .die * sdbot has quit (connection reset by peer)
quit	Causes the bot to quit the current server (with the specified message, if any). This command will also close the bot and kill all threads, so be careful.	<[sdbot]> .quit * sdbot has quit (goodbye.)
remove	Causes the bot to totally remove itself from the host computer (including autostart)	<[sdbot]> .remove removing bot... * sdbot has quit (Connection reset by peer)
quit	Bot quits running	<[spybot]> .quit * spybot has quit IRC (Client exited)
uninstall	It doesn't delete the server, only removes the startup keys	<[spybot]> .uninstall uninstalling... * spybot has quit IRC (Client exited)

In this example, DCOM for Windows, which allows a program on one machine to run code on another machine, is re-enabled. Also the administrative shares on the given machine are shared, leaving the machine in a vulnerable state and open to infection by network file infectors and open-share worms.

A compromised system can't be trusted. One can never guarantee or be sure that we've found all the back doors the attacker left to get back in. The fact that one can't find any more may mean only that we don't know where to look, or that the system is so compromised that what we are seeing is not actually what is there. Time to rebuild from scratch.

7 Conclusion

"Bot technology is rapidly evolving, often aided and abetted, unfortunately, by the open-source movement" [3, p. 4]. The bad guys of today test their malicious code against popular antivirus products to ensure their creations are undetectable before releasing them into the wild.

Open communication channels between antivirus firms, the authorities and ISPs, and implementing processes to deal with rogue servers hosting botnet channels could significantly cut down the lifespan of botnets. Most IRC bots variants include uninstall commands that will successfully

remove the bot from an infected system. The antivirus industry and ISPs must join forces and work together on sharing this information so that illegal botnets are quickly removed and their C&C servers shut down, once the binary has been analysed.

On the flip side, as more and more ISPs and IRC operators clamp down on illegal botnets, malware authors will look towards alternate C&C mechanisms such as IM and P2P [6]. As malware keeps pace with technology and evolves, the security community needs to invest more into research and stopping these next-generation threats.

References

1. Bächer, P., Holz, T., Kötter, M., Wicherski, G.: Know your enemy: tracking Botnets, from http://www.honeynet.org/papers/bots/ (2005)
2. Barford, P., Yegneswaran, V.: An inside look at Botnets, special workshop on malware detection. In: Advances in Information Security. Springer, Berlin from http://www.cs.wisc.edu/~pb/botnets_final.pdf (2006)
3. Baylor, K., Brown, C.: Killing Botnets: a view from the trenches. McAfee Whitepaper, from http://www.mcafee.com/us/local_content/white_papers/wp_botnet.pdf (2006)
4. Canavan, J.: The evolution of Malicious IRC Bots. In: Proceedings from Virus Bulletin 2005 Conference, Dublin, Ireland, from http://www.symantec.com/avcenter/reference/the.evolution.of.malicious.irc.bots.pdf (2005)
5. Ianelli, N., Hackworth, A.: Botnets as a vehicle for online crime CERT Coordination Center, from http://www.cert.org/archive/pdf/Botnets.pdf (2005)
6. Myers, L.: AIM for Bot co-ordination. In: Proceedings from Virus Bulletin 2006 Conference, Montreal, Canada, from http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_vb2006_myers.pdf (2006)
7. Porst, S.: Public malware contest Luxembourgish Computer Security Research & Response Team (CSRRT-LU), from http://www.the-interweb.com/serendipity/index.php?/archives/2006/05.html (2006)
8. Thomas, R., Martin, J.: The underground economy: priceless. The USENIX Magazine, December 2006, from http://www.usenix.org/publications/login/2006-12/openpdfs/cymru.pdf (2006)
9. Thomas, V., Jyoti, N.: Defeating IRC Bots on the internal network. Virus Bulletin, February, 2007, from http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_vb_defeating_irc_bots.pdf (2007)