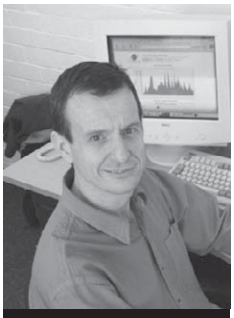
Botnet command and control techniques

Simon Heron, director, Network Box (UK)
Defence Systems



Simon Heron

"Incredibly, around 50 per cent of finance directors tested blithely inserted an anonymously sent USB key into their company computers."

For today's generation, the latest gizmos and gadgets present few challenges. Sophisticated online communities are just part of modern life. The problem with this inherent confidence and familiarity is that it can lead to carelessness and a disregard for danger.

Complacency while surfing the internet can be every bit as dangerous as older generations' lack of knowledge around security issues. Criminals are using this care-free attitude and are adopting increasingly sophisticated tactics to dupe users into installing malware onto their systems.

Infecting a machine

The most common methods of malware distribution are widely known. Emails that use social engineering techniques to persuade users to download executables are a common attack technique. Instant messenger attachments and other P2P file sharing technologies are also used to get that initial piece of malware onto users' machines.

In 2007, we can also expect the usual exploitation of vulnerabilities in operating systems. Last year Microsoft announced 140 vulnerabilities, which was more than double the combined total for 2004 and 2005. With the burgeoning popularity of video file sharing, it is inevitable that criminals will exploit this trend and use it as a distribution channel for their malicious code.

Script injection vulnerabilities are also an effective way of downloading malware. Adware and spyware brokers are increasingly turning to affiliate advertising programs to hide their code, leaving users exposed to unseen threats. Today's users can unwittingly visit infected websites that use a complex HTML 'GET' command to exploit the vulnerability by running local commands, operating a number of commands like 'wget', 'curl' or 'fetch', for example, to try and download the initial malware code.

In the past, there have been applications on the web that have been exploited directly. The most obvious is the SQL Slammer worm that attacked versions of MS SQL server exposed on the internet. Within 15 minutes, the worm spread from its source in Korea in the far east to the west coast of America via Europe. Such vulnerabilities are rare, but when they arise they are a gold mine for bot herders trying to install a network of botnets.

Much of the time, however, hackers rely on human curiosity. Recently, a consulting firm sent 500 USB sticks to finance directors at 500 companies in the UK. The devices were sent anonymously and just gave an invitation to a party. Incredibly, around 50% of the finance directors contacted, blithely inserted the stick into their company computers. While companies involved in media were the main violators, technology companies ared no better than retail and transportation companies. These devices could have quite easily been used to plant malware on the host PC while entertaining the finance director with some application or other.

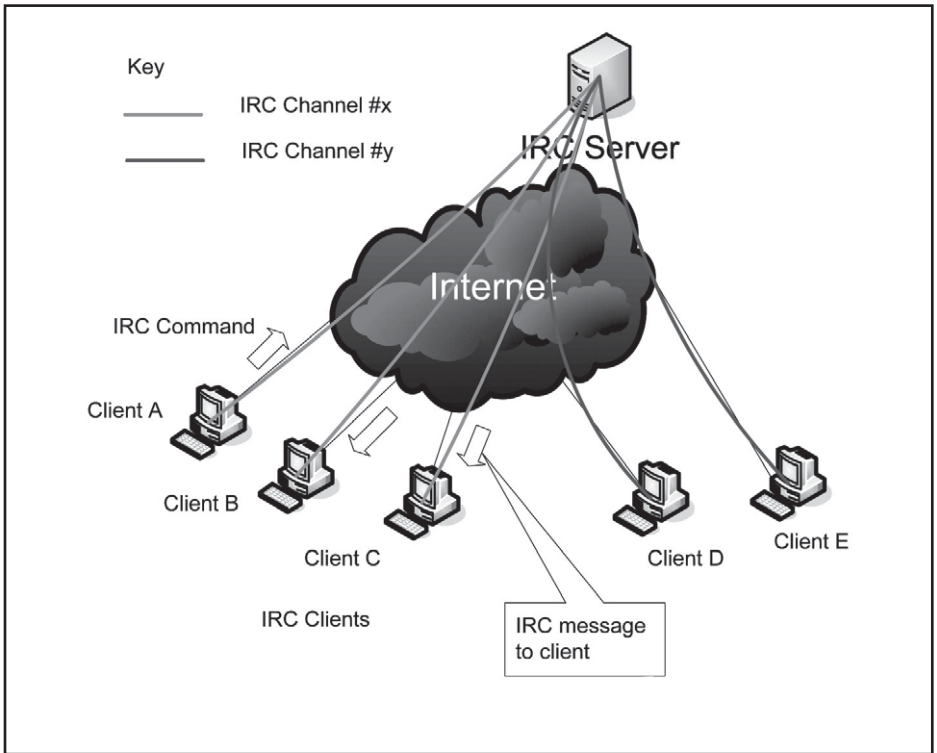


Figure 1: Message distribution via IRC

Command and control

As outlined above, there are a myriad of techniques a computer criminal can use to plant malware on a machine. Once installed, the malware can be used to execute any manner of pernicious tasks.

Many Trojans try to tap into a command and control channel which will enable the bot herder to tell the Trojan what to do. Commonly, the Trojan joins an IRC channel and waits for commands via that channel. To understand fully how this works, it is necessary to understand the IRC protocol as defined in RFC 1459[1] (although very few clients and servers rely strictly on the above RFC as a reference).

> "Bits of code are being digitally signed to prevent corruption, which means that only the bot herder with the correct key can control the botnet."

In IRC, clients make a connection to a server on tcp/194, but many use tcp/6667 or ports in that vicinity. Once the socket is connected, the client joins a channel using the IRC protocol. IRC is a text-based protocol allowing messages to be via a server that then identifies the recipient. In figure 1, client A joins channel #x and sends messages to clients B and C who are on the same channel. Clients D and E, who have connected to channel #y, do not receive these messages.

So, when a Trojan is installed, it needs an IRC server address, port, and channel. It will connect to the server on the port and the channel indicated and waits for commands. The bot herder then communicates via IRC with instructions for the bot. A simplified command sequence is shown below. Obviously, other messages can be sent and received depending on the functionality of the bot.

IRC command sequence

The bot herder sets the channel topic, which tells the bot what to do. The format is as follows:

> nickname!username command parameters
>
> So here 'nick' the bot herder, tells his bots to attack a site
>
> **:nick!joebloggs@acme.com**
> **TOPIC ddos www.victim.com**
>
> Then, a bot on a compromised net (compromised.com) reports that it is DDoS-ing the victim:
>
> **:bot1!bot1@compromised.com**
> **PRIVMSG I am ddosing www.victim.com**

IRC makes it possible for the bot herder to send out a command telling the bot to update itself at a particular website or ftp server. This allows the botnet to be kept up to date and new versions to be downloaded. It also means that if the update site is removed, the bot herder can send out a new update source.

Further developments resulted in bots being instructed to contact and infect other hosts. One interesting technique was to target 'Google dorks' – inept administrators whose poorly configured servers had been indexed by Google. Carefully-tailored search queries can identify servers that are vulnerable or at least promising targets (ever tried searching for "'Welcome to phpMyAdmin' AND 'Create new database'"?). These targets could be phpMyAdmin web frontends that had not been secured or CGI directories with scripts that might be useful to the hacker.

The problem with using IRC is that the bots need to know the IRC server, the port and the channel. This allows anti-malware organisations to disable the channel or the server, and hence stop the botnet in its tracks. The clients are still infected but they have no way of receiving instructions, rendering them dormant and likely never to be used again.

It is also possible to join these IRC channels and to work out the command structure and hence start controlling the botnet. It is then possible to talk, be it ever so briefly, with the bot herder. This doesn't tend to be a comfortable conversation for either party.

Upping the ante

Bot herders tried to counter this by encrypting this information in their scripts which stops others taking control. However, sooner or later, the server is identified and the botnet is decapitated. Bot herders then created a series of back-up IRC channels or servers. However, they soon met with the same fate; it simply took a little longer for anti-malware organisations to clean up.

This prompted bot herders to turn to P2P distribution. New versions of the bot can be injected into the P2P network at any point and eventually will get copied across the botnet. These bits of code are being digitally signed to prevent corruption, which means that only the bot herder with the correct key can control the botnet. However, most peer-to-peer networks have a server or set of servers that are contacted to join the network; this again makes the botnet vulnerable to being decapitated.

The solution for bot herders was to create a new P2P network with its own rules, as demonstrated by the Sinit family of trojans. The trojan installs itself via any of the methods discussed previously, but the clever part comes in the command and control phase.

Using UDP/53, the trojan starts probing for other infected machines at random IP addresses. UDP/53 is DNS and is usually open on most firewalls and, even though the packet is malformed, few systems detect or report this. The packet contains a random high number port for reasons described below and the latest version of itself (usually indicated as a timestamp). It will also include any information it has about other compromised machines.

Sooner or later a corrupted machine will be contacted and that machine will respond. One machine, 'Remote B', is shown responding in figure 2. The response packet includes the IP address of the original machine ('Infected PC'), as seen by Remote B along with the high value port number which was in the discovery packet. This enables the original querying machine to know if it is behind a NATed firewall or on a multi-homed host. If it can connect to this address on the high numbered port, it sends out this new IP address in all future discovery request packets.

The protocol has a few other commands. For instance, an infected machine that sees that a peer has a more recent timestamp (and therefore newer software) will request a file transfer from that peer.

Sinit also implements a simple web server on TCP port 53 and on a random, high numbered port that can serve up one file named kx.exe.. It waits to be contacted and then downloads itself to uninfected hosts. This requires some vulnerability on the uninfected host which will allow the code to be installed like the script injection exploit described above.

The advantage of this P2P technique to the bot herder is that there is no 'head' to cut off. All infected hosts speak to each other and the only option is to try and clean up all the hosts. This is not as difficult as it might appear for this bot. By making the appropriate query to an infected host – GET http://IP_address_of_infected_machine:53/kx.exe – a list of infected hosts along with their version numbers will be returned.

A more recent version of this type of bot is the Pecoan family. Like Sinit, the Pecoan trojan attempts to establish communications with other systems using its own P2P network. It connects to certain IP addresses through local UDP port 4000 (Win32/Pecoan.G uses local UDP port 7871) in order to download and execute arbitrary files onto the compromised system.

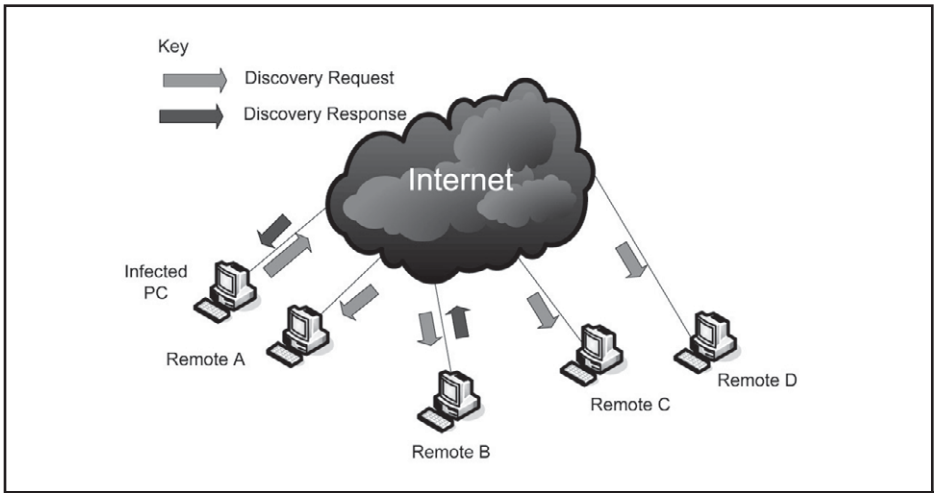


Figure 2: Sinit discovery phase

Using rootkit technology, Peocan hides what it downloads. It still refers to a command and control server, but if this server is disabled, the P2P nature of the software allows the bot herder to send out a new server address and retain the botnet.

The disadvantage of this trojan is that if a company correctly ties down outgoing traffic to only that which is expected, the trojan cannot get out on these ports. However, that leaves a vast number of incorrectly configured firewalls and an army of home PCs without firewalls.

Life cycle

Criminals start off by creating a trojan to support the operations they perceive as profitable (see figure 3, step 1). These might include denial of service attacks and spam relaying. They can be done by taking existing code and modifying it to suit their requirements. A well known bot herder, Californian Jeanson James Ancheta, did this using a trojan called RxBot.

The trojan (or bot) then has to be distributed (step 2). The bot herder also needs to create a command and

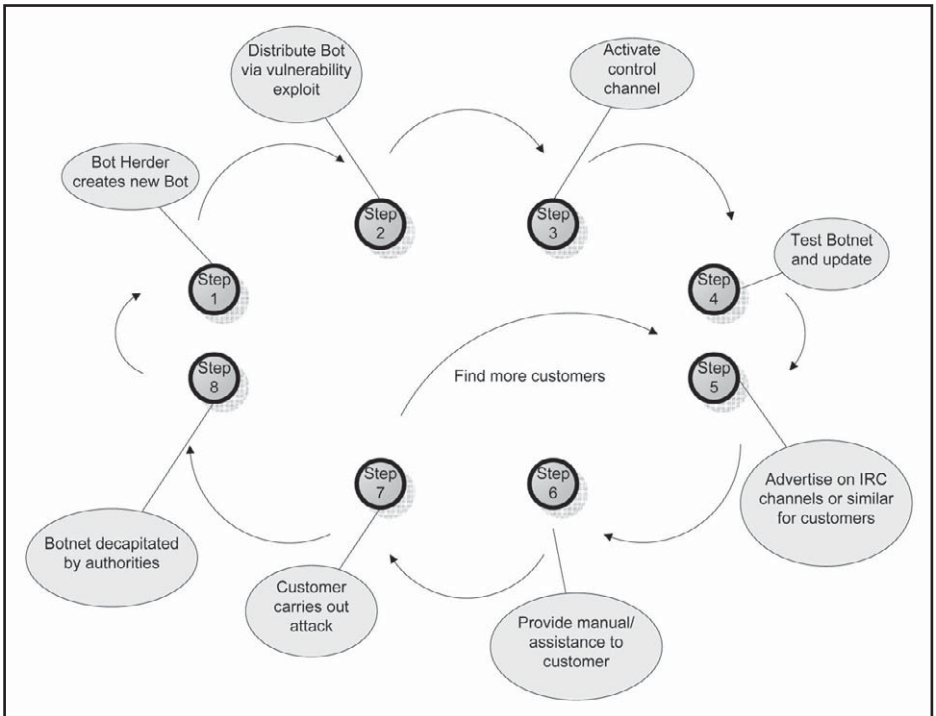


Figure 3: Botnet life cycle

control channel (step 3). Ancheta used MSN and file share vulnerabilities to spread his bot far and wide. Using this method, he built a botnet of around 400,000 compromised machines using the IRC method of command and control.

Like all developers, hackers test and update their code to ensure that it is effective; this can result in some curious and seemingly pointless spam arriving in people's inboxes (step 4). The hacker then has to find some customers (step 5). Ancheta used IRC, but other hackers may already be connected with organised crime organisations or unethical, opportunistic business partners.

The command and control is not straightforward. The bot herder will either assist the customer in carrying out the attacks or provide documentation to allow them to control portions of the botnet (step 6). The customer then carries out the attack (step 7) and the bot herder goes in search of his next customer (back to step 5).

Sooner or later someone will notice the botnet (step 8). In the case of the Sinit trojan, network managers began to identify malformed DNS packets which made them suspicious. The botnet is then either dismantled or decapitated, forcing the hacker, if not already caught, to devise a new ploy.

Conclusion

Botnets are becoming ever-more sophisticated in order to circumvent network defences. Given the infighting that goes on between bot herders, and the ongoing battle with anti-virus organisations, it is interesting to see digitally-signed software being used by the criminal community to protect their investments. So techniques supposed to protect legitimate software are now being taken up to protect illegitimate software.

It seems that herding bots is becoming a business. The infamous bot herder, Ancheta, had a botnet of 400,000 machines and a manual that would explain to his customers how to distribute spam or execute denial of service attacks. He had to find customers and support them; this is the process of legitimate industry.

The case of Ancheta also illustrates that the life of a bot herder can be profitable – he accumulated 60,000 USD in six months. This lends weight to the argument that money, rather than notoriety, is now the driving force behind cyber-crime. And while Ancheta was eventually caught, there are many others like him around the world willing and able to provide a service to those who want to use the internet for illegitimate purposes.

References

1. J. Oikarinen, D. Reed, "RFC 1459." Internet EFC.STD.FYI/ BCP archives. May 1993. March 2007 < http://www.faqs.org/rfcs/rfc1459.html>.

About the author

Simon Heron has over 16 years experience in the IT industry, including eight years experience in internet security. During this time he has developed and designed technologies ranging from firewalls, anti-virus, LANs and WANs.

Prior to Network Box, Heron co-founded and was technical director of Cresco Technologies, a network design and simulation solution company with customers in the USA, Europe and China. Before that he worked for Microsystems Engineering as a project manager, where he implemented network security for the company. He has an MSc in microprocessor technology and applications, and a BSc in naval architecture and shipbuilding.

Data breach activity is getting worse

March was a landmark month for data breaches. TJX, the retail group which has been slowly revealing the scope of the data intrusion that it originally discovered in December, finally unveiled the full extent of the damage. In its annual report filing to the SEC, the company revealed that 45.6m credit card numbers from transactions in 2003 were appropriated by thieves, who infiltrated the system in 2005 or 2006. The majority of the card data was stored in clear text.

The report said that investigating and fixing the problem has already cost the company $5m in the fourth quarter of fiscal 2007, and it is expecting lawsuits to follow. "We do not have enough information to reasonably estimate losses we may incur arising from the computer intrusion," says the report.

Data security firm Protegrity has a good idea, though. CEO Gordon Rapkin says that using the firm's risk model, it estimates the cost at around $1.6bn. The company factored in communication costs, downtime, auditing and remediation costs, along with ongoing investigations and collaboration with credit card company queries. Brand impact also played a part.

"I looked at the stock exchange over the last three month period," said Rapkin. "They used to trade at $30 and now they're $27. It's already cost their shareholders a number similar to what we projected." Share prices hit $30 just after the firm announced the data breach in January.

The data breaches just keep coming. This month, the Georgia department of Community Health reported that it lost a CD with the names, birth dates and social security numbers of 2.9m citizens. The Department of Energy's Counterintelligence Directorate has also lost 20 PCs. 14 of them were known to hold classified information at the Secret level, and the remaining six could have held classified information, said the report, which added: "Considering the sensitivity of the data regularly processed in CN, the shortcomings identified during our review were of major concern." The Department of Energy and its sub-branches are responsible for activities including nuclear weapons stewardship, counterproliferation, and research and development.