Detecting peer-to-peer botnets

Reinier Schoof & **Ralph Koning**
System and Network Engineering
University of Amsterdam
mail: reinier.schoof@os3.nl, ralph.koning@os3.nl

February 4, 2007

1 Introduction

Spam, DDoS and phishing are common problems on the Internet nowadays. In the past, attackers tended to use centralized high bandwidth connections to accomplish their tasks. Now that home users have high bandwidth internet connections, attackers have started infecting and using these home computers instead to for their attacks. Attacking from distributed locations, attackers are harder to catch or stop and often have more bandwidth to abuse. New methods are required to detect the forming of these widespread networks of infected hosts, especially now that it seems attackers have discovered the peer-to-peer (P2P) technology.

1.1 Context

As part of our study *System and Network Engineering*, we did a four week research project at SURFnet [1], the dutch internet service provider for educational and scientific institutes.

1.2 Problem definition

Fighting botnets is often a matter of finding their weak spot: their central point of command, or *command-and-control server*. This is usually an IRC, *Internet Relay Chat*, network where all compromised computers connect to, but with the use of P2P technology, this central point of command is nowhere to find: the hosts connect to each other and the attacker only has to become one of the peers to broadcast his commands over the network. A new detection and fighting method is required to prevent or stop such hazardous networks.

2 Background

To understand the meaning of this document, one should know something about both botnets and peer-to-peer networks.

2.1 Botnets

A computer waiting for its commander to give it orders is called a *bot* (or sometimes a *zombie*). A collection of these bots connected to a network is called a *botnet*, but usually we talk of a botnet when we mean a network of compromised computers which can be controlled by an attacker to e.g. distribute spam mail or start DDoS attacks. This way the original attacker remains anonymous. These computers are usually compromised by malicious

software, *malware*, like viruses or trojans and wait for their attacker to give them commands what to attack and when. Bots often connect to an IRC network. On this network they join a channel which is operated by the attacker, which gives them their instructions via the channel. Botnets can consist of thousands or millions of hosts and are therefore able to attack in a very distributed and powerful way. An attack is difficult to stop because of its large number of sources. Therefore it is essential to prevent the forming of these networks for example by preventing computers from getting infected or by taking down the central command point the bots are contacting.

2.2 P2P

Differing from the classic client-server model, on which most internet services are based, in a P2P network the connected hosts or *peers*, connect to each other, being both server and client (*ad hoc*), requiring no dedicated servers for the communication between the peers. In P2P-networks every peer is equal, however, some protocols use centralized servers for search operations or define certain 'superpeers': peers with high bandwidth and/or computing power available to perform such tasks in order to make the network more efficient. P2P-networks these days are widely used for file sharing and video streaming and are responsible for most of the total internet traffic [2].

3 P2P botnets

P2P botnets are bots that use P2P technology to accomplish certain tasks. The most common ways for a bot to use P2P are spreading and control, each described below.

3.1 Spreading

Using P2P networks for spreading copies of the bot itself is a well known way for a bot to make use of P2P technology. The reason for this is sharing files on a P2P network is very popular among users and the chances of a user downloading and installing malware by mistake are big. Some bots rename themselves to match popular files on the network by which the chance finding a bot instead of a regular file increases.

Most of the P2P bots found are of this type, therefore some antivirus companies prefix the names of these bots with *P2P* e.g. *P2P-Worm.Win32.SpyBot.fb*. Although the spreading factor of these bots is high, most of them connect to IRC servers and can be detected using techniques as packet inspection, flow analysis and monitoring the activity on the IRC server.

3.2 Control

The other kind of P2P bot uses P2P technology to send commands to each other instead of waiting for commands via an IRC channel. To control the botnet, the only thing the commander has to do is join the network as another peer and send the commands to other peers which pass them along. Although there are not many bots using this technology right now, they form a much greater threat. This is because ideally P2P uses no central host which can be detected and shut down disabling the whole botnet. However there are disadvantages: like normal P2P protocols one must solve problems like peer discovery and network responsiveness. Nevertheless the huge advantage of a decentralized network will increase the popularity of the use of P2P technology for bot writers and owners.

3.2.1 Closed P2P networks

Most bots use a P2P network that is dedicated to botnet communication. This means that the bot uses a home made, or borrowed P2P protocol to communicate. For example *Phatbot* [3] uses code from the *WASTE* project to implement P2P. But other bots like *nugache* [4] and *SpamThru* [5] have apparently integrated their own protocol to communicate.

3.2.2 Open P2P networks

Some bots use an existing P2P network, which is normally used for filesharing, for their communicating. This can be done by either transferring files or using its embedded chat system. At the time of writing there are no known bots of this type. However it is possible that they will emerge because of the huge popularity of P2P file sharing and the fact that their traffic is completely mixed with regular P2P traffic, which makes them much harder to detect.

4 Bot Analysis

To know how to detect P2P bots, one first needs to understand P2P bots. A controlled environment is set up to analyse the behavior of some bots. In this environment bots can safely be deployed, without running the risk of participating in a true botnet. In our research, two binaries of interesting bots that use P2P technology, Nugache [4] and Sinit [6], were available.

4.1 Environment

The test environment consists of four computers: three machines running Microsoft Windows XP SP2 and one machine running FreeBSD 6.2. The FreeBSD machine functions as a software router and connects the other three computers. The router runs softflowd [7] to collect netflow data and send this to a netflow aggregator, which in our case is the same machine, running nfcapd [8] with nfsen [9]. All traffic from the three machines can be analyzed this way.

The first bot tested is Sinit, a trojan which uses P2P technology to spread itself. When started Sinit tries to reach other Sinit infected hosts by sending special *discovery* packets to port 53 of random IP addresses on the Internet. When Sinit receives a *discovery response* packet, a connection between Sinit and the hosts sending the response is established. The two hosts exchange lists of peers they successfully connected to and Sinit will try to connect to these hosts. Once part of the P2P network, Sinit will spread additional trojans through the network.

The other bot tested on our environment is Nugache [4], a trojan that uses P2P technology for communication. Nugache opens TCP port 8 and has a static list of IP addresses to which it will try to connect on TCP port 8. When connected to one of the 22 initial peers, it will exchange the list of successfully connected peers. When told so by the commander, it will start DoS attacks. The communication is encrypted, or at least obfuscated. Nugache spreads over AIM, America Online Instant Messenger.

While the two bots were active in the controlled environment, softflowd collected the netflow data, for later analysis.

4.2 Results

Sinit started a web server on port 53 and also listens on UDP port 53 and a random high numbered UDP port. It started sending data to UDP port 53 of seemingly random IP ad-

dresses. As Sinit was started, *tcpdump* analyzed all its traffic and the analysis showed it immediately started probing IPs, without any evidence of retrieving a list of peers from the Internet. The binary was ran through several sandboxes [10] [11] and as a result, the IP addresses resulting from these runs did not match ours. Previous analysis by LURHQ [6] shows that assumption is true: Sinit tries to reach peers by trying random IPs. This way, Sinit does not require a central point for peer list exchange. The netflow statistics in *nfsen* displayed high ICMP traffic returning ICMP 3 (*host unreachable*). The web server embedded in Sinit only serves a single file, `/kx.exe`, which is the Sinit binary. Running *Microsoft Malicious Software Removal Tool* showed that Sinit had in fact infected the computer with other malware.

We also exposed the sandboxes to Nugache and found out that Nugache installs the list with hosts into Windows' registry, ironically enough under the key \\`HKCU\Software\GNU\`. Altering this list made Nugache trying to reach the new hosts, but we could not get another PC infected with a version of Nugache containing the altered list. Nugache keeps track of successful connections in the register and how many times each IP in the list was tried. We installed Nugache on all three of our test computers and on each of them altered the lists so that they would try to connect to each other. This obviously resulted in more successful connections, but the exchanged data was not readable, although it showed certain patterns which might mean that it is only obfuscated since true encryption would be likely to garble the data much more.

4.3 Other bots

Besides the bots mentioned before, we also looked at two more bots we were not able to retrieve, *Phatbot* and *Spamthru*.

4.3.1 Phatbot

Although we did manage to get the source code of Phatbot, which is a fork of *Agobot*, we were not able to compile it within our time frame. We did do some research on it and it turned out that Phatbot communicates using the technology of *WASTE*, an encrypted open-source P2P network [12], where Agobot still relies on IRC to communicate. Phatbot infected hosts find other peers by using cache servers on the Gnutella P2P network, which Gnutella clients use to find near peers. Phatbot looks for clients identified by *GNUT*, a Gnutella client, but on a different port than usual. It also has a list of processes to kill when it runs, consisting of both antivirus software and competing malware.

4.3.2 Spamthru

SpamThru is an interesting piece of malware. It is controlled from a single control server, but when it is taken down, commanding a single peer in the custom P2P network allows the attacker to define a new point of command and thus regaining control of the network. Just like Phatbot, SpamThru tries to wipe out any other malware but SpamThru downloads a complete pirated version of Kaspersky which scans the system, except SpamThru's files, for other viruses.

5 Detection

By analyzing bots in our controlled environment, the following characteristics were discovered which make detection possible.

5.1 Open ports

As you can see in our bot analysis a common requirement among P2P bots, and P2P technology in general, is that a specific port, or range of ports, must be opened to be able to connect to each other. This makes detection of P2P bots easier, since they all have certain specific ports to listen on. By monitoring data traffic to and from these ports or by actively scanning for them one can map infected hosts. Of course this method comes with false positives, and especially when bots start to use ports used by other applications you need additional methods to rule them out.

5.2 Connection failures

Analysis of netflow data showed a high rate of failed connections just after initialisation of the bot. This is a common problem for a lot of P2P networks and is caused by active firewalls, NAT (*Network Address Translation*) and hosts where the P2P application is not running anymore or hosts that are shut down. This results in a lot of ICMP "destination unreachable" packets and TCP reset packets which can be picked up in flow analysis. P2P file sharing networks use special hosts like *supernodes* or *trackers* to mitigate this problem, but since this makes botnets more vulnerable, it is not an attractive way to deploy the network. Taking out a supernode causes multiple peers to be disconnected from the network. Traffic to and from these special hosts is much denser compared to regular peers and therefore makes them easier to detect.

5.3 Peer discovery

Finding which host to connect to also seems to be a problem. Analysis of Nugache showed it used a static list of IP addresses to connect to upon initialisation. This isn't a smart way for a bot to implement P2P and maybe one can even call it a flaw because you can take the IP's in that list under surveillance and monitor which hosts connect to them. Then one can map them which can give some insights in the network. Although this method of peer discovery is very efficient, it's not likely future bots will implement this method. However this bot nicely illustrates the problem of efficient but cloaked peer discovery and Nugache's approach does not seem to solve this problem.

5.4 Obfuscation

Bots running on public P2P networks are a bit harder to detect because they use an existing protocol and infrastructure and have the characteristics of a regular user. Deep packet inspection may be required to distinguish this traffic from the normal P2P traffic as long as no encryption is used. Botnet control requires a lot less bandwidth than file sharing but this can of course be easily obfuscated by downloading random binary data.

There are also some regularity's in the current generation of P2P bots and even some signatures for its traffic [5]. With this information IDS, *Intrusion Detection System*, rules can be set up to detect this kind of botnet activity. This can be easilly obfuscated, but scanning for a lot of suspicious looking data on a well known port like Sinit does on port 53 will certainly help detecting P2P botnets.

6 Countermeasures

Fighting botnets using P2P technology is not going to be easy. Once the creators find a way to distribute host lists in an other way

then downloading them from a central point or hardcode them, detecting suspicious traffic between regular P2P traffic will be the only way to find infected hosts. Apart from certain flaws in the design of the bots we've discussed in this report, there is no known way of discovering them. Thorough bot analysis is then necessary to discover flaws in the design:

6.1 Nugache

As said before, Nugache uses a list of 22 IPs which the infected host tries to connect to. These hosts then send an updated list of infected hosts and the host connects to a certain amount of them. The list of IPs consists always at least of the initial 22 and only gets appended with other infected hosts. The list in the binary, however, remains the same: after having collected new list of peers, the hardcoded list is not updated. By taking these 22 hosts down, or blocking traffic to them, no new peer can connect to the P2P network. But still, this only stops the botnet from growing and does not affect hosts that were already connected to the network. Nugache listens on port 8, a normally unassigned port, so traffic on this port can be easily monitored.

6.2 Phatbot

To discover other peers, Phatbot infected hosts look in the Gnutella cache servers for clients with clientid *GNUT*, a common Gnutella client. Phatbot looks for particular GNUT clients listening on other ports than usual. With the help of the administrators of these servers, these invalid clientids can be banned from their servers and growth of the Phatbot network can be stopped.

6.3 Sinit

Sinit is the only trojan we found that uses no central point for peer discovery. Instead of retrieving a list of peers, Sinit just starts probing randomly chosen IPs. This is not a very effective way, since that includes over 4 billion possible IP addresses and our analysis also showed that its approach had only little effect. Another characteristic of Sinit is that it communicates with other peers on port 53, the standard DNS port, and its packets are always the same format. Intrusion detection systems can learn how to detect such traffic and block it, since this does not concern ordinary DNS traffic. Again, this approach only fragments the botnet and stops it from growing.

6.4 SpamThru

SpamThru only uses P2P technology as a backup communication protocol, to use when the central command point is taken out. The original central point of command is known. Apart from taking this host down, all traffic to this server can be monitored to map infected hosts.

7 Conclusion

Although a truly decentralized P2P botnet can be a huge threat, there are some problems with the implementation. As the bots we have discussed show us, peer discovery is a common problem and no matter how these bots try to solve it, it opens up a possible approach for detection: by keeping an initial list of hosts the botnet is easier to shut down and the randomly search for infected hosts, the large number of failed connections that could reveal an infection. Also announcing hosts to a central point like Phatbot does make the botnet more

exposed on hosts under a third party's command. Also the P2P network relies on opened ports to allow incoming connections which can be easily detected.

As long as people can make big money out of botnets and the activities they are used for, botnet owners will keep trying to create more sophisticated bots to keep a low profile. But unless there is a good solution for earlier mentioned problems, P2P botnets remain detectable and can be fought with countermeasures.

References

[1] SURFnet. http://www.surfnet.nl/.

[2] CacheLogic. The impact of p2p. *CacheLogic*, 2006. http://www.cachelogic.com/home/pages/isp/p2ptoday.php.

[3] LURHQ Threat Intelligence Group. Phatbot p2p trojan analysis. *LURHQ*, 2004. http://www.lurhq.com/phatbot.html.

[4] Robert Lemos. Bot software looks to improve peerage. *SecurityFocus*, 2006. http://www.securityfocus.com/news/11390/.

[5] Joe Stewart. Spamthru trojan analysis. *SecureWorks*, 2004. http://www.secureworks.com/research/threats/spamthru/.

[6] LURHQ Threat Intelligence Group. Sinit p2p trojan analysis. *LURHQ*, 2004. http://www.lurhq.com/sinit.html.

[7] SoftFlowd. http://www.mindrot.org/projects/softflowd/.

[8] NFDUMP. http://nfdump.sourceforge.net/.

[9] NfSen. http://nfsen.sourceforge.net/.

[10] Norman sandbox. http://sandbox.norman.no/.

[11] CWSandbox. http://www.cwsandbox.org/.

[12] WASTE. http://waste.sourceforge.net/.