P2P as botnet command and control: a deeper insight

David Dittrich
Applied Physics Laboratory
University of Washington
dittrich@u.washington.edu

Sven Dietrich
Computer Science Department
Stevens Institute of Technology
spock@cs.stevens.edu

Abstract

The research community is now focusing on the integration of peer-to-peer (P2P) concepts as incremental improvements to distributed malicious software networks (now generically referred to as botnets*). While much research exists in the field of P2P in terms of protocols, scalability, and availability of content in P2P file sharing networks, less exists (until this last year) in terms of the shift in C&C from* central *C&C using clear-text protocols, such as IRC and HTTP, to* distributed *mechanisms for C&C where the botnet* becomes *the C&C, and is resilient to attempts to mitigate it.*

In this paper we review some of the recent work in understanding the newest botnets that employ P2P technology to increase their survivability, and to conceal the identities of their operators. We extend work done to date in explaining some of the features of the Nugache P2P botnet, and compare how current proposals for dealing with P2P botnets would or would not affect a pure-P2P botnet like Nugache. Our findings are based on a comprehensive 2-year study of this botnet.

1 Introduction

There has been tremendous interest paid in the past year to *Peer-to-peer botnets*, principally Storm, a hybrid bot that uses the Overnet peer-to-peer (P2P) protocol for bots to find their concealed central command and control (C&C) servers.

Most papers now refer to any distributed malware network as a *botnet* (regardless of whether it has anything to do with IRC, where the term *bot*, short for *robot* was first used.) There is a strong bias in many botnet papers, be they IRC or P2P based, on the use of worm-like methods of propagation, and to focus on central *command and control* (C&C) concepts as the means to observe botnet activity, count bots, and to mitigate the botnet by taking control of the central C&C channel away from the botnet operator.

The research community has been warning about the advent of P2P malware since at least 2005, [2, 8, 4] but not until after 2007 when the Storm trojan gained wide media attention had there been many deep studies into botnets moving to the use of P2P concepts for C&C. One of the biggest hurdles posed by P2P malware, as opposed to the more widely seen central IRC-based command and control botnets, is the difficulty in estimating the size of the P2P botnet and means of rendering the botnet unusable. In this paper, we review some of the research to date into P2P malware networks, and give further and complementary insights into the most successful pure-P2P botnet network seen to date, Nugache[1] [20], and show the challenges in dealing with this type of botnet.

The paper is structured as follows: we provide an extensive comparison, as well as clarification and correction, to existing work in Section 2, describe the data collection mechanism for our study in Section 3, and show some results of the traffic analysis in Section 4.

2 Related Work

The existence of Nugache was documented widely in the anti-virus/malware community in late April, early May 2006, [14] but variants ceased to be directly detected by anti-virus (AV) engines in the latter part of 2006. Since Nugache was initially easily identified through some invariant aspects within the host file system and through connections on port *8/tcp*, it may be that the investment in time to do more detailed and time intensive analysis was not deemed to be worthwhile. Once the Nugache program was altered to use

[1] Documents [27, 12] disclosing the author of Nugache were released in late June 2008.

random high-numbered service ports for each bot, the malware dropped off of nearly everyone's radar. Another round of analyses were done by AV vendors in December 2006, as the operators of the Nugache network began propagating the malware by way of blog posts and spam that directed users to a dropper identified as TROJ_DLOADER.IBZ. [24, 25, 26, 15]

In January 2008, Nugache received a burst of attention in relation to two other botnets, Storm and Rizo (a classic IRC-based central C&C botnet.) This resulted from a talk in November 2007 and article that followed in December 2007 [6] that discussed sharing of malware code, concepts, and propagation techniques. Some articles [11] claimed that Nugache was overtaking Storm in size and was responsible for a recent drop in bot prices on the black market. This is not supported by our analysis, which shows no observed commands or activity in the Nugache network since September 2007, and the population of Nugache nodes had shown a downward trend in size that continues to this day. [3] This drop-off could be explained by the arrest of the Nugache author in early September 2007. [27, 12]

Trend Micro [23] discuss P2P for C&C, but discount it due to latency issues. Nugache, on the other hand, supports *adding latency* through the use of random duration `Sleep()` calls (see Section 4.1.1). In their list of rallying mechanisms, they include hard-coding and two ways of using DNS, but do not foresee P2P as a rallying mechanism, nor do they take into consideration a two-phase method of rallying that involves pre-seeding a list on a host with up-to-date contact points, rather than the more naive method of using *only* the hard-coded addresses in the binary, which necessitates updating the binary to update the seed list. Nugache solves these problems in a very elegant way not observed in other published research listed here.

In February 2007, Schoof and Koning analyzed the April 2006 version of Nugache (the version that uses port *8/tcp*.) [18] While they noted obvious aspects, such as the use of Windows Registry keys, they did not fully comprehend the role of the Windows Registry keys for maintaining a list of likely servent peers in relation to the hard-coded list of 22 default hosts.[2]

A year later, Steggink and Idziejcak [19] claim, "Only one bot, Peacomm is build [sic] from the ground with a complete decentralized P2P structure," which seems to contradict earlier work [18] and is inconsistent with the observations of other researchers who found Storm to be a more conventional trojan – albeit multipartite, very versatile and flexible – that used Overnet only to conceal its central C&C servers, not as the C&C mechanism itself. [16, 20, 9]

In February 2007, Vogt, et al, [28] discuss the feasibility of creating a hypothetical *super botnet*. Their model assumes a single worm is used for propagation, with an algorithm that requires the adversary make some complicated choices in order to create the super botnet, and poses a problem to the attacker of propagating routing information throughout the super botnet without exposing the complete routing information to defenders or rivals. They claim that C&C in and of itself is a weakness that would expose the super botnet. They postulate how their concept of a super botnet would be constructed using many C&C servers, each provided with routing information using a complex algorithm. They also propose another complex algorithm for the attacker concealing their location using encrypted *time bombs*.

Our study of Nugache shows that there are much simpler ways to achieve the same effects proposed by Vogt et al. [28], and in some cases with fewer drawbacks than suggested for their super botnet. We believe that the problem is not C&C itself – a botnet would be useless if the adversary could not control it! – but rather the weakness lies in the use of *central* C&C, especially in clear text. [3] By creating a super botnet that relies on multiple, relatively easily detectible central C&C points, there is little additional survivability beyond what some attackers gain today by using multiple small botnets instead of one large one. Their main contribution is suggesting how attackers might do more elaborate encryption of commands than is done even by Nugache, which is more sophisticated in its use of encryption than other malware discussed to date, even Storm. [20]

The continued focus on central C&C on the defensive side can also affect detection of pure-P2P malware. If the detection mechanism relies on closely-timed connections to a central point, pure-P2P malware would not be detected. At the same time, pure-P2P malware containing obsolete seed lists can trigger false-positive "scanning" events. [7]

Also in April 2007, Wang, et al, [29] describe a theoretical advanced hybrid peer-to-peer botnet, based in part on a cursory analysis of a version of the Nugache P2P bot from May/June of 2006 (mentioned earlier in this section.) Many features of their proposed bot are shared by the more recent version of Nugache that was still active at the time, however their work is more speculative than factually descriptive of Nugache's actual design [5]. They claimed weaknesses in Nugache, includ-

[2] They did not know, for example, that if the Registry is already filled with a peer list, there is no need for the bot to use the default list at all. In fact, this is the technique used by the trojan horse *dropper* that infects hosts with Nugache.

ing the same mistake regarding the seed list [18]. Those weaknesses and some of the advantages of their theoretic hybrid do not hold true for the current state of development of Nugache. These authors, like others, have a strong affinity towards worm-like rapid propagation and central C&C concepts. Their proposed advanced hybrid, and their models for its robustness, assume this, and extend the central C&C model to include multiple C&C servers that are announced, or retrieved, in order to maintain the advanced botnet. While they propose an interesting propagation model and peer-list updating mechanism, including a manual *update* command that the operator must execute at critical points in the botnet growth, this is not how Nugache works in practice.

Nugache has a much simpler mechanism that is automatic, independent of propagation activity, and does not require peers to poll for routing information. Neither is there a requirement for bulk-transfers of routing information beyond when new peers first join the network. [3] While both other works [29, 28] assume worm-like propagation, with reinfection having to be factored in via a routing information distribution requirement, we agree with the authors that this assumption is not a real-world consideration. Our observations of Nugache bear this out (see Section 4.2). Finally, the requirement of an *update* command and sensors itself introduces a fatal weakness in the design of the advanced hybrid botnet, which itself could be used to enumerate and identify peers as they point out in their proposed defenses. Nugache does not suffer this weakness, nor might it readily succumb to the other two strategies for mitigation suggested by Wang, et al, namely: (a) taking out the peer list updating servents or high-degree of connectivity nodes, neither of which exist in the Nugache network, or; (b) randomly removing even a large percentage of servents, which in Nugache are somewhat independent.[3] [3]

3 Data Collection

In order to get a much more detailed and comprehensive picture of the attacker's activity, network-level data is required. This includes full-packet captures that allow detailed analysis of individual network flows, mechanisms of attacking vulnerable target systems, Domain Name System lookups, and C&C sessions.

It also includes higher-level meta-data about connections, such as that obtained from IP packet headers. Source and destination host addresses, times, durations of flows, all of these things are not easily determined through just analysis of host-level data.

The actual content of flows, however, is only easily observed if there is no use of encryption. This is one of the principle problems faced in the analysis of Nugache, since its most recent incarnation does *all* C&C, as well as binary updates, over its heavily encrypted P2P channel. Without visibility on individual flows, analysis must focus instead on the remaining available meta-data listed above and how it relates to available host-level data, including reverse engineering of the malware itself, or attacking the cryptography using less direct methods. Knowledge can still be gained from the meta-data level, including learning about active peers, how the attacker is using the network, when the malware updates itself, etc.

The network traffic analyzed in this paper was collected using two early development releases (versions *roo-1.0.194* and *roo-1.1-RC-2*; the current release is *roo-1.4*) of the Honeynet Project's *Honeywall* bootable CD-ROM. [22] The Honeywall captures full-packet data and stores it, and flow related information, for later analysis, as per the Honeynet Project's definition of *Data Capture.* [21] The collected information comprises header data, Snort intrusion detection signatures, passive operating system fingerprints, as well as full packet captures that illustrate the exchanges between the hosts.

The Honeywall provided a good foundation for the capture of network traffic, as it has basic alerting capabilities, and the ability to throttle some outbound scanning traffic, and a mechanism for remotely blocking access to the honeypot if/when an emergency occurs [22]. The Honeywall was not, however, designed to analyze a P2P malware artifact and and we had to overcome some limitations through programming new functionality on top of the basic Honeywall.

4 Network Traffic Analysis

Due the use of strong cryptography, there is very little information that can be directly obtained from monitoring network traffic. Early versions of Nugache still had an IRC based C&C mechanism, but after C&C was moved to the encrypted P2P channel, only indirect observations could be made from network traffic analysis alone.

The three principle foci for network traffic analysis were: (1) observations related to P2P connections;

[3] It is the assumption that the advanced hybrid botnet uses a modified central C&C mechanism that provides the strongest proposed defense, which attacks this assumed weakness. Some have suggested that multiple defensive counter-attack strategies at once may be more effective than naive defenses when dealing with complex distributed attack networks. [13]

(2) observations about scanning activity associated with propagation through exploitation of remote vulnerabilities in LSASS and DCOM services; and (3) observations of DDoS attacks performed using the P2P network.

4.1 Peer-to-Peer Connections

4.1.1 Peers over time

During an approximately 2 month period in mid-2006, a total of 1974 IP addresses were witnessed engaging in port *8/tcp* connections, either inbound or outbound to our honeypot. Some of the outbound connections failed, due to the peers being no longer active, being behind a firewall, or blocked by an IPS.[4]

Comparison of the list of *8/tcp* IP addresses – either successful or failed connections – with two lists of peers seen in IRC traffic shows that there is limited overlap between these sets. The two largest sets of peers seen in IRC contain 1428 and 1206 peers each, respectively. (The first flow was obtained in mid-May 2006 and lasted 34 minutes, and the second one was obtained late May 2006 and lasted 2 minutes.) The overlap between them is 284 IP addresses. Comparing each of these IP address lists with the list of 1974 IP addresses seen in port *8/tcp* flows during this six-week period finds 195 IP addresses in common with the second set and 228 IP addresses in common with the first set. There are 84 IP addresses in common to all three sets.

Some of this difference could be accounted for through use of dynamic addresses (e.g., high-speed dialup lines or peers moving between DHCP assigned addresses.) Another cause could be attrition of peers through identification and cleanup. These are some of the same issues identified by Bhagwan et al. [1], and make it inaccurate to call these lists of *peers* and not just IP addresses.

More worrisome was the possibility that there were many more compromised hosts than was evident from network traffic captured at the single point of observation initially set up for this analysis. The P2P mechanism for primary C&C could be allowing the attacker to control a significantly larger number of hosts and only bring a small percentage (say 10%) of them at any one time into service for scanning or propagation activity. Later on, as we were able to decrypt some of the flows with the aid of host-based tools[5], we were able to verify the Nugache operator was using probabilistic delegation, which could account for some of the small set overlap we had observed, as seen here:

```
if(Rand(0,99)==0)) {
  Sleep(Rand(0, 16000000));
  Logs.Send("208.77.188.166", 31337);
}
```

Enumeration of this botnet in related work [3] revealed that at its peak there must have been at least 6,000 *active* IP:port pairs at any given time, with a total infected footprint of close to 11,000 IP:port pairs. While some regularly scheduled snapshots revealed IP:port pair sets of about 1,500 in June 2007, the intersection of them would only be about 700-800 IP:port pairs due to fluctuations in host reachability and IP address reassignment. Recent events [12] explain why the botnet has been decaying since summer 2007.

4.1.2 Peer Key Exchange

The malware uses 256-bit Rijndael to encrypt the P2P based command and control communication [20]. The session key is exchanged RSA-style with ephemeral 512-bit (64 byte) to 1024-bit (128 byte) keys.

For a 512-bit key exchange, it functions as follows: Peer A sends `00 02` to Peer B to announce a peer key exchange, followed by 64 bytes. Peer B, in turn, replies with 64 bytes, followed by a series of acknowledgment and keep-alive messages. More precisely:

1. Peer A generates two large prime numbers p and q, and computes the 512-bit (or 64 byte) modulus n = pq and $\phi = (p-1)(q-1)$. Peers A and B share the *public exponent* e = 65537 ($2^{16}+1$). Peer A also generates the decryption exponent d such that $ed = 1 \bmod \phi$. Peer A sends n over to M.

2. Peer B receives n and creates a PKCS#1 v1.5 [10] RSA encryption message M as outlined in Table 1, and encrypts it to a message $C = M^e \bmod n$.

3. Peer B decrypts the message as $M' = C^d \bmod n$, and extracts the 32-byte (or 256-bit) session key from the message.

4. Both peers are encrypting the connection using Rijndael-256-OFB.

[4] We discovered that one particular IPS appears to have been tuned by an operator with a sense of humor, who configured their IPS to respond to incoming SYN packets on port *8/tcp* with ACK—RST packets containing as the data payload the string, *Go away, we're not home.*

[5] Without going into details, reverse engineering allowed us to identify runtime in-memory data structures from which the Rijndael encryption keys could be reconstructed. Given a memory dump, and full-packet data captures, any/all flows initiated in one direction can be manually decrypted.

00 02	nonce (21 bytes)	check (4 bytes)	session key (32 bytes)	TestN (4 bytes)

Table 1. One example of a RFC2313-inspired key exchange message M

4.1.3 Traffic on Peer-to-Peer Port

The malware receives several inbound connections per day, and makes (or attempts to make) several outbound connections per day. Overall there are about a dozen connections active at a given time, and comprise regular peer list exchanges, software comparisons and upgrades, and simple commands via a set of numeric commands [20].

One possibility that was contemplated in the early stages of analysis was that the P2P protocol exists primarily to prevent anyone who finds a peer from being able to see more than a small percentage (maybe 10% or less) of the entire P2P network, until such time as the attacker wants to use the P2P network and instructs them all to go join a specific IRC channel, listen for commands and execute them, then disconnect until further notice. This would mean a defender finding a single peer would have only a very small window of time in which to catch an IRC connection and determine the complete extent of the P2P network. (The delay between IRC connections showing up appeared to be on the order of a week, and the connections appeared to last only a short period of time, on the order of an hour or so.) After the upgrade in June 2006 when the C&C port changed to randomly chosen port numbers above 1024, the bot authors shifted their primary C&C from IRC channels to the P2P channel.

4.1.4 Traffic on IRC

In the early versions of Nugache, we witnessed several IRC sessions (in clear text), with as many as 1043 peers in one of the channels. There are at least two IRC sessions that involved scanning. Here are unique commands[6] sent to the channel:

```
#ch4nn3l :notify:this
#ch4nn3l :quit:ch4nn3l
#ch4nn3l :scan:exploit,lsass
#ch4nn3l :scan:payload,HTTPEXEC,http://
  www.[deleted].com/files/a.exe
#ch4nn3l :scan:start,3000
#ch4nn3l :scan:target,add,192.168.0.*
#ch4nn3l :scan:target,add,192.168.1.*
#ch4nn3l :scan:target,add,192.168.100.*
#ch4nn3l :scan:target,add,R5000
#ch4nn3l :scan:target,current
#ch4nn3l :spaim:10
```

This excerpt shows examples of C&C traffic instructing the peers to use the LSASS exploit, forcing downloading of the malware from the specified URL, and targeting three non-routable address blocks (as defined by RFC 1918[17]) and 5000 randomly selected addresses.[7] The RFC1918 address blocks are the ones typically found behind firewalls, broadband routers, and other private networks, which would be otherwise inaccessible to the attacker. Not only that, but these three specific non-routable blocks being scanned by the Nugache operator were the defaults used by three of the most common commodity network devices, implying the attacker was purposefully scanning *behind* any network defenses on home or small-office/home-office (SOHO) networks.

Subsequent scanning, as seen in Table 2show targeting of the only hosts on the local routable network, or exclusively random addresses.[8]

4.2 Propagation

The controllers of Nugache have employed multiple methods of propagation, from direct attack against remotely accessible services (and attacking hosts *behind* commodity NAT and WiFi devices), to completely indirect attacks using some clever social engineering methods.

The malware itself is capable of direct propagation via three mechanisms: (1) Exploitation of remotely accessible vulnerabilities in the LSASS service (139/tcp) and RPC-DCOM (445/tcp); (2) Emailing copies of itself to potential targets obtained from the Windows Address Book (WAB) that *do not contain* certain keywords.[9]; and (3) Via instant messenger (AIM and MSN), sending the potential victim a message created from randomly chosen sentence fragments accompanied by a link for the victim to click on.

Another mechanism for indirect propagation was observed, involving a complex form of a trojan horse, or *dropper*, attack. The attack works like this:

[6] Only the channel and domain name are obfuscated. The rest of the commands are verbatim.

[7] "Random" in this context is defined to be 1.*.*.* through 223.*.*.*. These are the originally defined IPv4 "Class A" through "Class C" network address ranges.

[8] Because of differences in the way the honeywall "data control" mechanisms were set, some scanning times are longer than others. This does not represent the maximum scan rate that could be obtained by this malware.

[9] This would prevent mail from going to system administrators or support personnel.

Targets	*Port*	*Month*	*Duration*
192.168.0.*, 192.168.1.*, 192.168.100.*, Random	445/tcp	May 2006	38 min
192.168.0.*, 192.168.1.*, 192.168.100.*	445/tcp	Jun 2006	51 min
192.168.0.*, 192.168.1.*, 10.1.*.*	445/tcp	Jun 2006	2.25 days
192.168.0.*	445/tcp	Jul 2006	6 min
Local net	445/tcp	Sep 2006	11 min
Local net	445/tcp	Sep 2006	2 h 8 min
Local net	139/tcp	Sep 2006	13 min
Local net	445/tcp	Sep 2006	12 min
Local net	139/tcp	Sep 2006	2 h 44 min
Random	139/tcp	Oct 2006	1 h 3 min
Random	445/tcp	Oct 2006	24 min
Random	445/tcp	Oct 2006	34 h 24 min
Random	139/tcp	Oct 2006	2.33 days
Random	139/tcp	Oct 2006	6.75 days
Random	139/tcp	Oct 2006	3.25 days
Random	139/tcp	Oct 2006	19 h 8 min
Random	139/tcp	Oct 2006	23 h 33 min
Random	139/tcp	Nov 2006	2 h 10 min
Random	139/tcp	Nov 2006	8 h 7 min
Random	139/tcp	Dec 2006	7 h 6 min
Random	139/tcp	Dec 2006	26 min
Random	139/tcp	June 2007	54 min

Table 2. Observed Scanning/Exploit Behavior

- The attacker selects a freeware application (in this case, a video editing application). A copy of the distribution archive is downloaded and the `SETUP.EXE` program is altered to include a list of 300 active peers and bootstrap code that connects to one of these peers, downloads the malware, and installs/runs it.

- The attacker sets up a web server on the Internet and registers a domain name, in this case a variation on a popular web service. The default web page on this server is a duplicate of the popular web service, looking and acting exactly like the real site. Elsewhere on the server is a special application (PHP script) that serves up a copy of the malware, as well as a copy of the altered freeware application distribution archive file.

- The attackers then place links on popular freeware download aggregator sites where users go to search for freeware/shareware applications. Two popular sites have been used to date. The download link points to the trojan horse archive. The aggregator sites claim the download file is "guaranteed 100% free of viruses or spyware" (a claim that, in this case, is false, however AV engines have never seen this malware downloader, so they would report it virus-free anyway.)

- The attacker then uses an active Nugache P2P network to "download" the archive thousands of times in a short period of time, rocketing the trojaned archive to the top of the popularity charts (in fact, to the #1 most download file on one site.) Anyone going to the home page of this site, and looking at the Top Ten list, would see this trojaned archive as the most popular program on the site. The peers do not in fact download the file, but simply make the proper HTTP `POST` command to trigger the download counter. Enough of a delay is introduced between `POST`s so as to not effect a denial of service attack, only to boost the download counter to a sufficient degree to raise the popularity rating of the trojaned archive.

4.3 DDoS Related Activity

During the period of observation, there were only a handful of instances in which DDoS activity was observed: once involving an HTTP `GET` attack against a commercial victim, another instance of what may have been extortion[10] related attacks against three commercial sites, and one involving a UDP flooding attack. There was also some low-level HTTP `GET` activity, which at first appeared to be a DDoS attack, but closer analysis showed that these requests were used to increase the popularity of a trojan horse installer described in the previous section.

4.3.1 HTTP GET flooding attack

HTTP `GET` flooding was observed in October 2006, over a two-day period, all directed towards a single target served by a DDoS defense service.

In the HTTP `GET` flooding attacks, we observed that the malware was coded to randomize the target path in the URI of the `GET` request in ways designed to defeat the filtering mechanisms employed. The frequency of

[10] One series of attacks over period of days, two weeks prior to Valentine's Day 2007, targeted three online jewelry stores. One of the three targets was hit much harder than the others, while one attack failed altogether, due it is believed to a bug in the code; The two commands that generated outbound packets used complete URLs including prototype, DNS name, and path, while the command that was not accompanied by outbound traffic specified only a DNS name.)

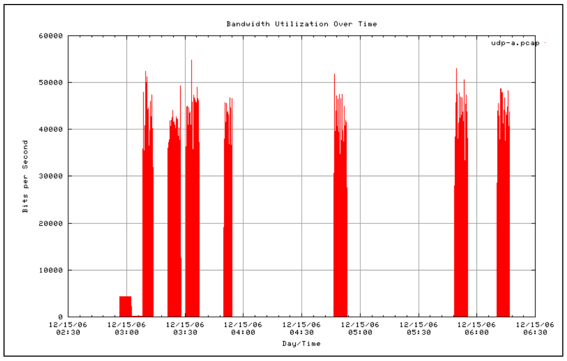

Figure 1. Observed UDP flooding activity

initiating GET requests was varied (as is common in DDoS attacks) in order to use the minimal attack resources necessary to achieve the objective of disrupting the target service.

Also observed was an amplification effect caused by the mitigation mechanism itself. For every ~100 byte HTTP request, a ~400 byte reply was sent back indicating the request had been refused due to suspected DDoS activity. While this is a relatively small ($4x$) amplification effect, it occurred *within the target network infrastructure*, maximizing the DoS effect on the victim. The attacker would thus need $1/4$ the number of agents to saturate the victim's network resources. Reacting to this issue, changes to the system used to protect the target were implemented, reducing the effectiveness of possible future attacks (requiring more than $4x$ the number of attacking hosts to get the same effect.) Unfortunately, even with an effective reaction to this type of attack, the attacker can always result to a traffic consumption attack involving non-TCP based floods. This brings us to the next attack method observed, a UDP packet flood.

4.3.2 UDP Flooding

DDoS attack traffic involving UDP flooding was observed in the middle of December 2006, with seven targets in common between the two agents we controlled at the time. The targets were a handful of non-commercially related hosts on broadband/DSL lines. Both nodes being observed started and ended their attacks within seconds of each other, and sent similar traffic to the same targets. Figure 1 shows the bandwidth utilization graph for one of the two nodes.

Analysis of the network traffic showed a distinct large packet (~1100 bytes) coming from one peer to peer A. Peer A immediately sends out a similarly sized packet to 6 peers. Another similar sized packet comes in from a second peer, but this time there is no packet from peer A. Peer A then begins flooding the target. Within less than a second, the same behavior – incoming large packet, relayed to a handful of peers, second incoming large packet with no relay, followed by initiation of flooding – involving peer B. This observation is consistent with a TTL-based request transmission mechanism through a P2P network as used by some P2P protocols.

5 Conclusion

We have seen a quick survey of the research done recently in the analysis and possible mitigation of advanced botnets using P2P technology. We have corrected some misconceptions based on a real-world example of a peer-to-peer malware artifact designed for resilience against disruption or take over, creating a strong network for conduct of activities questionable in nature. The difficulty of observing the C&C traffic represents a hurdle that a competitor or a network defender must undertake.

The constant, evolutionary steps over the past six years towards integrating P2P as the primary C&C mechanism [4] have possibly reached a new plateau in distributed malware. The current detection and mitigation approaches, such as DNS-based or text-based signature detection, will be impeded by this new class of distributed malware.

The success of Nugache at being active for so long, with little general attention, shows that low-volume activity and indirect means of propagation can get past some of today's technical defenses. We suggest that while there is some hope of improving technical defenses, including correlation techniques, another promising avenue may be to increase the non-technical defense capacity in terms of coordinated and collaborative incident response and investigation.

Acknowledgements

The authors would like to thank Brian Eckman, Matt Wilson, Joe Stewart, John Hernandez, Adam Turoff, Michael Collins, Phil Groce, Ross Kinder, and others at CERT, the University of Washington, and elsewhere for their support, assistance, and valuable discussions.

References

[1] R. Bhagwan, S. Savage, and G. Voelker. Understanding availability. In *Proc. of the 2nd International Workshop*

on Peer-to-Peer Systems, LNCS 2735. Springer Verlag, February 2003.

[2] E. Cooke, F. Jahanian, and D. McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *SRUTI05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005.

[3] D. Dittrich and S. Dietrich. Discovery techniques for P2P botnets. Technical Report CS 2008-4, Stevens Institute of Technology, September 2008.

[4] D. Dittrich and S. Dietrich. New directions in P2P malware. In *Proceedings of the 2008 IEEE Sarnoff Symposium*, page 5, Princeton, New Jersey, USA, April 2008.

[5] D. Dittrich and S. Dietrich. Technical Report CS 2008-3. Stevens Institute of Technology, June 2008.

[6] D. Fischer. Storm, Nugache lead dangerous new botnet barrage. *SearchSecurity.com*, December 2007. `http://searchsecurity.techtarget.com/news/article/0,289142,sid14_gci1286808,00.html`.

[7] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium (Security'08)*, July 2008.

[8] T. Holz. A short visit to the bot zoo. *IEEE Security & Privacy Magazine*, 3(3):76–79, May-June 2005.

[9] T. Holz, M. Steiner, F. Dahl, E. W. Biersack, and F. Freiling. Measurements and mitigation of peer-to-peer-based botnets: a case study on storm worm. In *LEET'08: First USENIX Workshop on Large-Scale Exploits and Emergent Threats, April 15, 2008, San Francisco, CA, USA*, April 2008.

[10] B. Kaliski. PKCS #1: RSA Encryption version 1.5. RFC 2313, Internet Engineering Task Force, 1998.

[11] E. Messmer. Nugache worm kicking up a Storm, January 2008. `http://www.networkworld.com/news/2008/010708-nugache-worm.html`.

[12] J. Miller. Cheyenne teen pleads guilty to computer crime. *Star-Tribune*, August 2008. `http://www.casperstartribune.net/articles/2008/08/16/news/casper/2c6fb0ecfe2ddf6c872574a700057d26.txt`.

[13] S. Nagaraja and R. Anderson. The topology of covert conflict. Technical Report UCAM-CL-TR-637, University of Cambridge, July 2005.

[14] J. Nazario. Nugache: TCP port 8 Bot, May 2006. `http://asert.arbornetworks.com/2006/05/nugache-tcp-port-8-bot/`.

[15] J. Paz. Fake Media Player Movie - TROJ_DLOADER.IBZ, December 2006. `http://blog.trendmicro.com/fake-media-player-movie-troj-dloaderibz/`.

[16] P. Porras, H. Saidi, and V. Yegneswaran. A Multi-perspective Analysis of the Storm (Peacomm) Worm. Technical report, Computer Science Laboratory, SRI International, 2007.

[17] Y. Rekhter, B. Moskowitz, D. Karrenberg, G. J. de Groot, and E. Lear. Address allocation for private internets. RFC 1918, Internet Engineering Task Force, 1996.

[18] R. Schoof and R. Koning. Detecting peer-to-peer botnets, February 2007. `http://staff.science.uva.nl/~delaat/sne-2006-2007/p17/report.pdf`.

[19] M. Steggink and I. Idziejcak. Detection of peer-to-peer botnets, February 2008. `http://staff.science.uva.nl/~delaat/sne-2007-2008/p22/presentation.pdf`.

[20] S. Stover, D. Dittrich, J. Hernandez, and S. Dietrich. Analysis of the Storm and Nugache Trojans: P2P is here. In *USENIX ;login: vol. 32, no. 6*, December 2007.

[21] The Honeynet Project. Honeynet Definitions, Requirements, and Standards version 1.6.0, 2004. `http://www.honeynet.org/alliance/requirements.html`.

[22] The Honeynet Project. Honeywall, 2005. `http://www.honeynet.org/papers/cdrom/`.

[23] Trend Micro. Taxonomy of Botnet Threats, November 2006.

[24] Trend Micro. TROJ_DLOADER.IBZ, December 2006. `http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=TROJ\_DLOADER.IBZ`.

[25] Trend Micro. WORM_NUGACHE.G. `http://www.trendmicro.com/vinfo/virusencyclo/default5.asp?VName=WORM\_NUGACHE.G`, December 2006.

[26] Trend Micro. WORM_NUGACHE.G (Japanese web site), December 2006. `http://blog.trendmicro.co.jp/archives/1053`.

[27] United States Department of Justice. Wyoming man charged with infecting thousands of computers with 'trojan' that he used to commit fraud. Release No. 08-090, June 2008.

[28] R. Vogt, J. Aycock, and J. Michael J. Jacobson. Army of botnets. In *Proceedings of the 2007 Network and Distributed System Security Symposium (NDSS 2007)*, pages 111–123, February 2007.

[29] P. Wang, S. Sparks, and C. C. Zou. An advanced hybrid peer-to-peer botnet. In *HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 2–2, Berkeley, CA, USA, 2007. USENIX Association.