

# New Directions in Peer-to-Peer Malware

David Dittrich  
University of Washington  
dittrich@u.washington.edu

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

## 1 Introduction

Intruders are constantly changing, improving, and extending the capabilities of their malicious software (malware). Not only have their tools evolved from single-purpose programs run manually to attack from one host to one other host, but have now evolved into feature-rich distributed peer-to-peer networks of bots (botnets).

In this paper, we will examine in general the multiple generations of malware leading up to the latter category of botnets, and in specific the failures and successes in the category of hybrid/P2P botnets. We will include some thoughts on the implications for telecommunications providers in an age where anti-virus, anti-spam, intrusion prevention systems, and firewalls individually may to prevent the compromise of internal enterprise or customer systems, and how reaction capabilities must adapt to the emerging threat environment. The convergence of features in today's "smart phones" will bring this threat to hand-held devices in the same manner as historically has been seen in desktop and portable computers.

Over the last 5 years, P2P botnets have emerged and reached the maturity to be used for their malicious purpose over larger scale in size and time. This article extends the work in [3] to focus more on the P2P aspects of command and control channels in malware.

## 2 Generational Comparison of Denial of Service Attack Tools

The word *attack* can be interpreted many ways, and is often the source of much confusion when trying to determine a defensive response. For example, the response to a theft of intellectual property or financial data is far different than the response to a denial of service attack. Likewise, there are two different

preventative strategies, and two different detection strategies.

To simplify the analysis of the generational differences in malware, we will focus only on those malware artifacts that implement Denial of Service (DoS) attacks [11] intended to cause one or more targetted systems or services to crash, lock up, or otherwise be rendered unresponsive to normal, legitimate requests. While it is obvious that some of the examples included here are not intended solely for DoS attacks, they all include DoS capability.

There are at least six generations of DoS-related attack tools that can be seen over time. These generations are depicted in the timeline in Figure 1.

They break down into two distinct sets: Single-threaded tools and distributed tools. Within these sets, there is a progression towards more and more powerful tools, with the next generation being a transition to a higher-level of efficiency and power than the previous generation.<sup>1</sup>

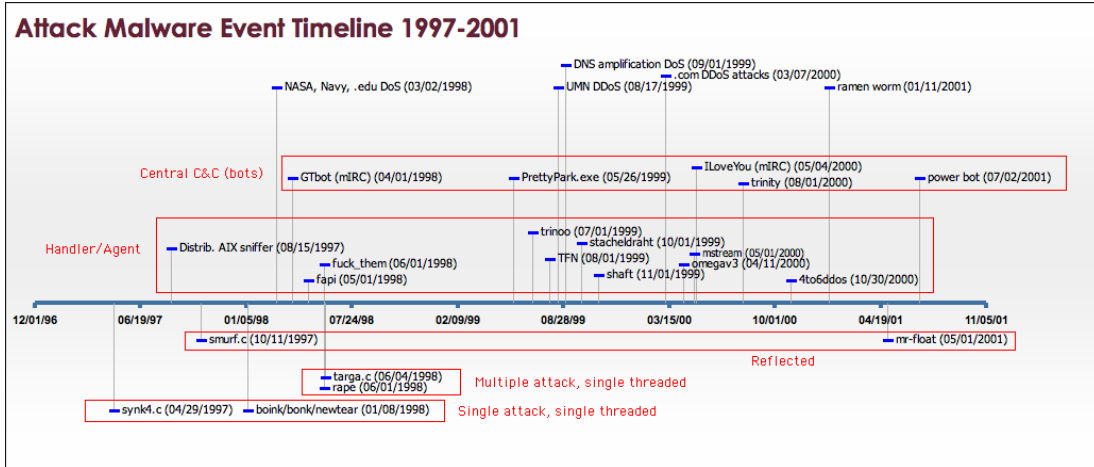
In ascending order, the six generations seen to date follow in the next subsections.

### 2.1 Single-threaded, single-attack

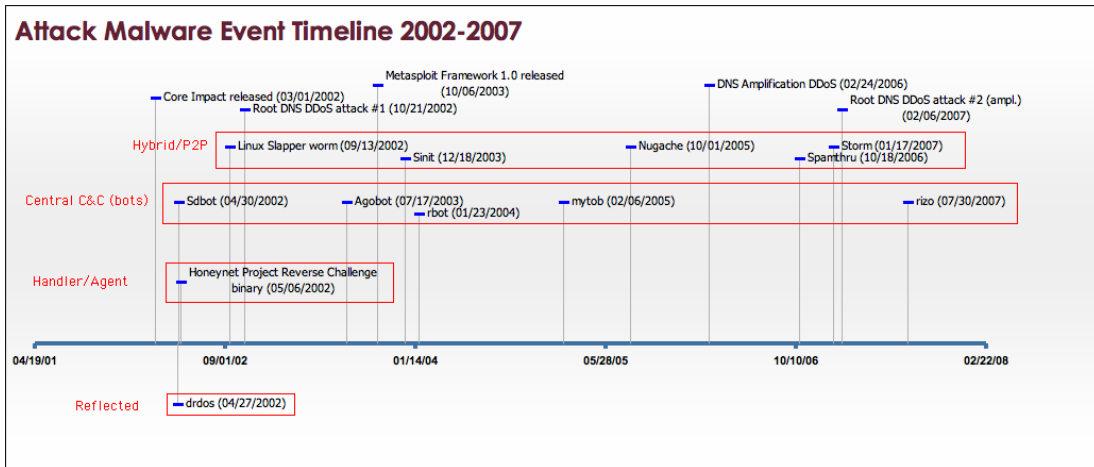
These programs are generally small, simple, single-purpose, and run in a single process or thread. Many were written as *proof-of-concept* exploits that illustrate a vulnerability, and in many cases serve as a means of verifying whether a vulnerability is present or not, and even allowing testing of fixes and patches. In that sense, they serve a positive purpose. In the hands of someone with malicious intent, however, they can be used to cause damage.<sup>2</sup> SYN flooding (to prevent systems from accepting any more legitimate connections), fragmented packet attacks that crash

<sup>1</sup>There are other attributes of the tools listed in each of the generations that cause an overlap. These overlaps will be pointed out in the descriptions of the generations and tools.

<sup>2</sup>For an analysis of positive vs. negative aspects of exploits, the concept of *full disclosure*, etc., see Section 6.7.2 of [11]



(a) Timeline to 2001



(b) Timeline from 2002

Figure 1: Malware timelines

Windows TCP/IP stacks, and malformed packet attacks that crash routers, were very popular in the late 1990s.

## 2.2 Single-threaded, multiple-attack

Trying to choose the right exploit to perform a DoS attack against a victim who may use a system that is patched for the chosen exploit, or may have greater available bandwidth, can be difficult. To increase the chances of successfully denying service to the intended target, some attackers took to using scripts or C programs that run a large number of single-threaded/single-attack tools in turn. Two popular programs that bundled a dozen or more exploits were *targa.c* and *rape*. Since these programs only increase

the possibility of finding a workable exploit, but do nothing to increase the available bandwidth for a flooding attack, someone with a fully patched system on a fast network connection would always win the battle. More firepower was needed, and so other means of involving a larger number of hosts in an attack were required.

## 2.3 Single-threaded, reflected/amplified

A vulnerability was found involving ICMP Echo Request packets with forged source addresses (of the intended victim) that were directed at the broadcast address of subnets with a large number of hosts that would respond to the request. (This is known as a *directed incoming broadcast* request flood.) The most

popular form of this attack was known as a *Smurf* attack<sup>3</sup> For each inbound ICMP Echo Request would come a multiple number of replies, resulting in an amplification effect, as well as being reflected off the hosts in the vulnerable network (making traceback harder.) Paxson described a large number of such attacks that would reflect and/or amplify attack packets. [16] Another related form of reflected attack was described in the analysis of the *Power bot*, called an *unwitting agent* attack because it does not require any malicious software, but rather exploits a remotely accessible vulnerability to run a standard command on the vulnerable host that causes it to attack the intended target. (E.g., The script *mrfloat* used the Unicode directory traversal hole in Microsoft's IIS web server to run PING.EXE, sending a series of ICMP Echo Request packets to the victim site. It was used on May 4, 2001, to take down the [www.whitehouse.gov](http://www.whitehouse.gov) web site. The Power bot, two months later, used this same vulnerability for both propagation of the bot itself and for Distributed Denial of Service (DDoS) attacks. Power would scan for vulnerable hosts, add them to a list it kept locally, and either use the list for unwitting agent DDoS attacks or to upload and install a copy of the Power bot.)

## 2.4 Distributed, handler-agent

A series of client/server applications, all using custom protocols and source code bases, were created for one purpose only: to perform DDoS attacks. The motivation for these programs (per an email exchange with one of the authors) was for a small group who controlled a very large number of compromised hosts to fight back against a much larger group who were using single-threaded, single-attack programs in a coordinated (yet manual) fashion. The larger groups would coordinate the attack using IRC, informing the group which IP addresses to attack. The smaller group, capable of controlling hundreds of computers per person (in an automated fashion) were able to out-gun their opponents by one to perhaps two orders of magnitude. The primary limitations of this generation of attack tools were the necessity to create a unique command and control protocol, limitations in the ability to scale the attack network due to finite open file handle tables in the Unix systems used for these attacks, and weaknesses in the programs them-

<sup>3</sup>Name for the blue creatures created by Belgian cartoonist Peyo, see <http://www.smurf.com/>

selves that allowed a defender (or rival) to scan for them on the network.

## 2.5 Distributed, central command and control

Fights between individuals and groups communicating on IRC was the primary motivation for creating the earlier generations of attack programs. Because of this, and the fact that bots were already being used by these attackers for other purposes, it made sense to create DDoS bots that used standard IRC for command and control, allowing the attacker to use one front end – their favorite IRC client – for everything. IRC networks were also already tuned and resourced to handle a huge number of users world-wide.

*Remote Access Trojans*, and even the *PrettyPark* trojan, were made to use IRC for remote control, as were bots that served up pirated audio, video, game, and commercial software files (known as *XDCC bots*.) Single-purpose programs like *kaiten.c* and *knight.c* performed primarily DDoS functions, and later programs such as Agobot/Phatbot integrated dozens of functions in a single program.

## 2.6 Distributed, hybrid

The first public attack involving the use of malware with P2P capabilities was the *Linux Slapper Worm* in 2002 [7]. While the program claimed to handle over 16 million simultaneous connections, it never got to anywhere near that size (due mostly to the noise of worm propagation activity.) IRC bots of the previous generation, such as *Agobot* (with a custom P2P protocol that appears to have not been successful, and was abandoned) and *Phatbot* (which used the open source *WASTE* P2P source code, but was likewise abandoned), were early attempts to move to P2P. More recent examples of P2P include *Sinit* in 2003, [4] *Zindos* in 2004, [19] *Nugache*<sup>4</sup> in 2005, *Spamthru* in 2006, [18] and the mis-named *Storm worm*<sup>5</sup> in 2007.

<sup>4</sup>While *Nugache* was discussed in private circles as early as October 2005, there is no public discussion of *Nugache* until April 2006.

<sup>5</sup>*Storm*, or *Peacomm* as it is alternately known, is actually a trojan, not a worm.

## 2.7 Specific Distributed System Network Structures

The evolution of command and control channels over the last few years [3] has taken us from simple hierarchical structures (handler/agent) to more structured and centralized command and control structures (such as IRC channels and botnets), to the concept of a decentralized structure such as *P2P* networks.

### 2.7.1 Peer to Peer Networks

The use of Peer-to-peer mechanisms for command and control introduce an entirely new topology and communication flow than previous models. [3]

In the P2P model, all attack agents form a randomly connected network, where no single host or network of hosts are responsible for central communication. Unlike either of the two previous models, the P2P model does not need any central host or hosts responsible for command and control, or even for joining the P2P network.<sup>6</sup>

Commands are retransmitted through the P2P network a limited number of times, enough for all peers to see and act on the command. The attacker is able to connect to any of the peers using a special client program and initiate commands, which are then relayed throughout the P2P network. Responses are similarly routed through the network until they reach the intended recipient (or are dropped because they have exceeded a “time-to-live” threshold.)

Use of P2P command and control in malware was envisioned as early as 2000 [23]. As shown in Section 2.6, attempts to implement P2P mechanisms achieved only limited success until 2006. The most successful of the P2P-capable malware networks were Nugache and Storm. While Storm used P2P (and Fast Flux DNS[9]) for rallying bots to its central command and control servers, only Nugache used its P2P channel as the primary means of command and control. [20]

After the first publications to mention the advent of P2P-capable malware in 2005, [1, 6] attention within the security industry and research community began to shift. [12, 10] The Congressional Research Service has even looked at the subject in detail, with an eye towards policy implications for Congress. [22]

<sup>6</sup>The SpamThru Trojan [18], which also uses a P2P model for some command and control, also employs a central server for its spam templates. This is a hybrid of the IRC and P2P models.

There are many different models for P2P networks that result in different network graph topologies, [2] as well as different strategies for attacking the resulting networks. [13]

## 3 Conclusion

Starting around the time of the first DDoS attacks on the root DNS servers in 2002, central command and control IRC-based botnets became the principle subject of research and investigation in academic, commercial, and legal circles. The principle means of fighting botnets is to focus on identifying and removing command and control channels on IRC servers, or take down the *rogue* IRC servers themselves. [17, 1, 5, 8] These efforts haven't proven effective against the typical clear-text bots using standard IRC protocols, and law enforcement activities have resulted in many successful prosecutions. [21, 14, 15])

Command and control traffic that *does not* utilize a single IRC channel and is heavily encrypted, significantly increases the difficulty and time it takes for the entire attack network can be identified and mitigated. It is clear that attackers have been trying for years to move away from clear text IRC-based command and control, and are achieving some success using pure-P2P malware networks over the last 5 or so years. This trend is finally being recognized in the research community, and it will only be a matter of time before a mass-migration away from the easy to mitigate IRC-based botnets occurs. The long-running Storm spam deluge is further evidence of what will face us in the near future.

## References

- [1] Evan Cooke, Farnam Jahanian, and Danny McPherson. The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets. In *SRUTI05: Proceedings of the Steps to Reducing Unwanted Traffic on the Internet Workshop*, 2005. <http://www.eecs.umich.edu/~emcooke/pubs/botnets-sruti05.pdf>.
- [2] David Dagon, Guofei Gu, Chris Lee, and Wenke Lee. A taxonomy of botnet structures. In *Proceedings of the 23 Annual Computer Security Applications Conference (ACSAC'07)*, December 2007.
- [3] Dave Dittrich and Sven Dietrich. Command and control structures in malware: From Handler/Agent to P2P. In *USENIX ;login: vol. 32, no. 6*, December 2007. <http://www.usenix.org/publications/login/2007-12/pdfs/dittrich.pdf>.

- [4] Threat Intelligence Group. Sinit p2p trojan analysis. <http://www.lurhq.com/sinit.html>.
- [5] Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*, February 2008.
- [6] Thorsten Holz. A short visit to the bot zoo. *IEEE Security & Privacy Magazine*, 3(3):76–79, May-June 2005. <http://lufgi4.informatik.rwth-aachen.de/publications-pdfs/13.pdf>.
- [7] HoneyNet Project. Scan of the Month 25: Slapper Worm .unlock.c source file, 2002. <http://www.honeynet.org/scans/scan25/.unlock.nl.c>.
- [8] The HoneyNet Project. Tracking Botnets, 2005. <http://www.honeynet.org/papers/bots/>.
- [9] The HoneyNet Project. Fast-Flux Service Networks, 2007. <http://www.honeynet.org/papers/ff/fast-flux.html>.
- [10] Trend Micro. Taxonomy of Botnet Threats, November 2006. <http://us.trendmicro.com/imperia/md/content/us/pdf/threats/securitylibrary/botnettaxonomywhitepapernovember2006.pdf>.
- [11] Jelena Mirković, Sven Dietrich, David Dittrich, and Peter Reiher. *Internet Denial of Service: Attack and Defense Mechanisms*. Prentice Hall PTR, 2004.
- [12] Lysa Myers. Aim for bot coordination. [http://www.mcafee.com/us/local\\_content/white\\_papers/threat\\_center/wp\\_vb2006\\_myers.pdf](http://www.mcafee.com/us/local_content/white_papers/threat_center/wp_vb2006_myers.pdf).
- [13] Shishir Nagaraja and Ross Anderson. The topology of covert conflict. Technical Report UCAM-CL-TR-637, University of Cambridge, July 2005. <http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-637.pdf>.
- [14] Department of Justice. Criminal Complaint: United States of America v. Paul G. Ashley, Jonathan David Hall, Joshua James Schichtel, Richard Roby and Lee Graham Walker, 2004. <http://www.reverse.net/operationcyberslam.pdf>.
- [15] Department of Justice. Over One Million Potential Victims of Botnet Cyber Crime, 2007. <http://www.ic3.gov/media/initiatives/BotRoast.pdf>.
- [16] Vern Paxson. An Analysis of Using Reflectors for Distributed Denial-of-Service Attacks. <http://www.icir.org/vern/papers/reflectors.CCR.01/reflectors.html>.
- [17] Ryan Naraine. Botnet Hunters Search for 'Command and Control' Servers. *eWeek.com*, 2005. <http://www.eweek.com/article2/0,1759,1829347,00.asp>.
- [18] SecureWorks. SpamThru trojan analysis, October 2006. <http://www.secureworks.com/analysis/spamthru/>.
- [19] Joe Stewart. Zindos worm analysis, July 2004. <http://www.secureworks.com/research/threats/zindos>.
- [20] Sam Stover, Dave Dittrich, John Hernandez, and Sven Dietrich. Analysis of the Storm and Nugache Trojans: P2P is here. In *USENIX ;login: vol. 32, no. 6*, December 2007. <http://www.usenix.org/publications/login/2007-12/pdfs/stover.pdf>.
- [21] United States Department of Justice. U.S. v. James Jeanson Ancheta. <http://news.findlaw.com/hdocs/docs/cyberlaw/usanchetaind.pdf>.
- [22] Clay Wilson. Botnets, Cybercrime, and Cyberterrorism: Vulnerabilities and Policy Issues for Congress, November 2007. <http://fas.org/sgp/crs/terror/RL32114.pdf>.
- [23] Michal Zalewski. “i don’t think i really love you,” or writting [sic] internet worms for fun and profit. <http://seclists.org/lists/vuln-dev/2000/May/0159.html>.