Peer-To-Peer in Botnets

Moritz Marc Beller

Fakultät für Informatik,
Technische Universität München
{beller}@in.tum.de

Abstract. Cyber-attacks are a growing threat to the security of the world. As a result, in June of 2011, the German government founded a "national defense centre against cyber crime".[fSidIB11] This step became necessary as new forms of computer worms are no longer restricted to harming only computers and the data stored on them, as the example of "Stuxnet" demonstrates:
In 2010, a new computer worm called "Stuxnet" was discovered. It is suspected to have damaged Iran's uranium centrifuges, giving it the title "the first cyber weapon"[Ben11]. To spread and update, Stuxnet used Internet and Intranet structures to build its own *botnet*.[FMC] Stuxnet is in alignment with a series of new botnets using *peer-to-peer* communication instead of a centralized server.
This literature review paper focuses on these peer-to-peer botnets and the growing threat emerging from them. Having read this article, the reader will be well-informed about current and future techniques in peer-to-peer botnets.

1 Introduction

In this work, we describe a new means of communication in botnets, namely *peer-to-peer* communication. We motivate the research on botnets by recent real-world examples and introduce common botnet terminology. We give a classification of P2P botnets and distinguish different architectures of botnets. In the following sections, we describe the typical lifetime of a P2P botnet, the command and control structure in P2P botnets and compare conventional botnets to P2P botnets. Finally, we described possible countermeasures against P2P botnets.

2 Definitions

A computer able of executing remotely-triggered commands is called a *bot* or *zombie*. A *botnet* is a group of bots forming a common network structure.[SK07] In most recent papers on the subject ([WWAZ09], [ARZMT06]), the term botnet is defined as purely negative, i.e. a network performing destructive aims such as denial-of-service attacks attacks, sending spam or hosting a phishing website[SI07]. Other common aims include providing the aggregated CPU resources of the botnet, stealing user's credentials [Bor] or doing click fraud on affiliate networks[New11]. Click fraud is the process of generating clicks on web banners without an actual user seeing or clicking the advertisment, for the sake of making money.

In the following, We propose a bias-free definition of botnet as per our understanding technology is generally ethics-free. Additionally, there are many examples where botnets are used in a non-destructive way (e.g. [oC11]), or even to destroy existing malicious botnets.

A *botmaster* is referred to as the controller of the botnet. This doesn't necessarily have to be the founder of the botnet (cf. 4.1).

The expression *bot candidates* specifies the set of computers which are target to becoming a bot themselves.

P2P, short for *Peer-to-Peer*, being a technology buzz word of the internet in the late 1990s with the upcoming file sharing services like Napster[Inc11], has attracted less attention in recent years. P2P defines an unstructured information network amongst equals — so-called peers. Two or more peers can spontaneously exchange information without a central instance. According to [SFS05] "P2P networks promise improved scalability, lower cost of ownership, self-organized and decentralized co-ordination of previously underused or limited resources, greater fault tolerance, and better support for building ad hoc networks." These properties coupled with the fact that files exchanged in P2P networks are prone to malware, trojans and viruses make P2P networks a most-attractive base for building botnets. With so many Well-known P2P networks include the Napster[Inc11], Gnutella, Overnet and Torrent network. A *P2P bot* then is a bot that uses a P2P protocol as a means of communication with other bots.

The so-called *C&C*, command and control structure, specifies the way and protocols in which the botmaster and the bots communicate with each other. It is the central property of any botnet. Common protocols for C&C include IRC, HTTP, FTP and P2P.[Bor]

IRC — internet relay chat — is a "teleconferencing system"[irc], typically used for text chatting in channels joined by a large number of participants. While its protocol is relatively easy to implement, it provides a lot of features. It has thus become the one of the most widely-used protocols for C&C in conventional botnets.

The process of *bootstrapping* generally describes starting a more complex system outof a simple system. In regard to botnets, the term usually means loading of the bot code (often injected into the original filesharing program) and establishing a connection to other bots.[WWAZ09]

3 A brief history of botnets

The origins for the term "bot" go back to so-called IRC bots, short for IRC robots. An IRC bot is a program that handles specific tasks in the IRC automatically, so that the administrator does not need to do those routine jobs.

The first IRC bot ever to be created was named Eggdrop. Its origins go back to the year 1993. However, in April 1998 a deriviant called GT-Bot appeared and formed the first malicious botnet, using IRC's C&C structures. Four years later, in 2002, Slapper was the first botnet to make use of P2P for C&C.[LJZ]

A criminal conviction because of launching a botnet is relatively seldom, since botmaster try to hide their identities (see "steeping stones" in figure 1). Yet, in 2007, John Schiefer was sentenced to four years in prison. He built a botnet with up to 250,000 zombies, collecting passwords and bank credentials from the bots.[Hru11]

<TODO> storm botnet

4 Types of P2P botnets

4.1 Classification P2P networks

There are three types of P2P networks: "parasite", "leeching" and "bot-only".[WWAZ09]

Parasite and leeching bots infiltrate existing P2P networks, while "bot-only" networks are designed as new networks.

Parasite botnets recruit new bots only from the set of existing P2P participants; they try to infect system inside the P2P network and make them become bots. Due to the often illegal content distributed in file sharing networks, they are a perfect culture medium of viruses, malware and worms. It is thus convenient for an attacker to spread a highly-demanded file (e.g. some cracked computer game, software or porn) containing the injection code sequences for his bot. This code is then injected into the file sharing client. Vulnerable hosts in the network are infected this way. On the downside, this means that the spread of the bot is limited to the size of the P2P network.

In contrast, leeching bots not only try to infiltrate systems which are already part of the P2P network, but also systems outside of the P2P network. Natuarally, they are bigger in size as they have to deliver the P2P client, too. This might be more difficult to achieve as it means that systems must unwillingly take part in the network. Often, firewalls and port-forwarding are not configured for use with the P2P network on these systems, reducing the performance of the botnet: Since the owner of the computer usually doesn't even know he is participating in a P2P network, he has no intention of opening the ports in his firewall. Leeching bots can spread through any possible measure: File sharing, downloads on websites, email attachments and instant messanging.

There are good reasons for either strategy: Using an existing P2P network as a base — like parasite and leeching bots do — unburdens the botmaster from setting up and building a botnet infrastructure. It profits from the established P2P network, making use of filtering, error-correction and encryption as far as the chosen network has support for it. On the other hand, features are limited to the existing P2P protocol. A specifically-built P2P bot-only network is natuarally more tailored towards its purpose. Due to the bot-exclusive memberships, it might be easier to shutdown the botnet for an attacker as all participants can be considered bots and there is no risk of accidentally shutting down an innocent member. Bot-only networks are described in detail in 5.2.

4.2 Lifetime of P2P botnets

Wang et al.[WWAZ09] differentiate three stages of P2P botnets:

- Recruiting bot members (infecting others)
- Forming the botnet (construction phase)
- Standing by for instruction
  This is the actual "operational" phase of the botnet. Bots are awaiting instructions from their master. Instructions can either be actual commands or performing updates. In this phase, the chosen C&C structure is essential.

It should be noted that these phases are not strictly exclusive, e.g. during the third phase (standing by for instruction), building of the botnet may well continue. In fact, this is an inherent property of any P2P network: Constant transformation of the network. It is only until a critical mass of bots has proceeded past phase one and two, that the botnet can be called operational.

5 Architectures of Botnets

In reality, there exist two principally different architectures of botnets. A new architecture has been proposed by scientists, which combines the advantages of both. The following nomenclature is extracted from [SI07]:

5.1 Centralized Architecture

Historically the oldest form of botnets, centralized architectures are built up in such a way that there is one central spot which broadcasts messages between the

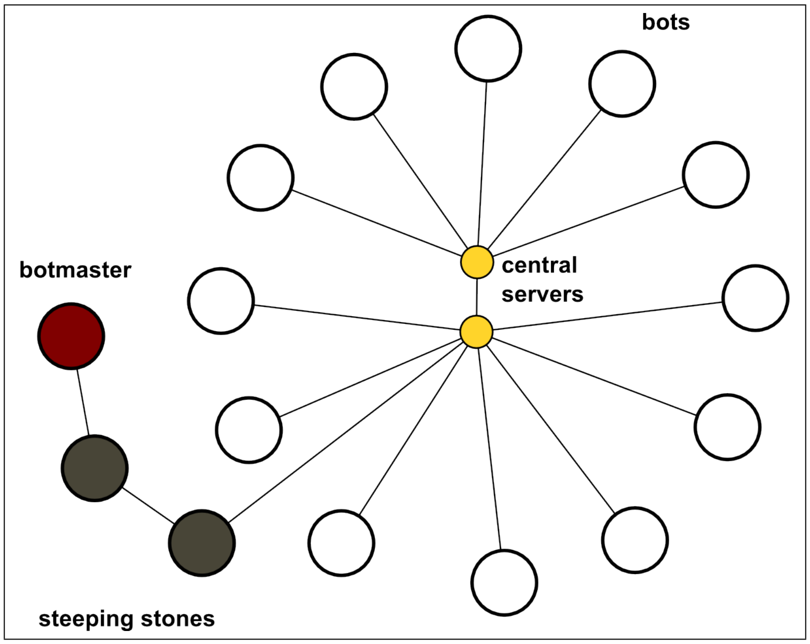

Fig. 1. Graph depicting connections in a centralized network. Note how all bots only have a connection to the central point. (Source: Own work)

connected bots and the botmaster. It's common to have more than one central server[WSZ10], but even with several servers, the architecture still stays centralized. For if you shutdown this central point, the network is inoperable. This resembles the biggest weakness of centralized architectures: As soon as you are able to cut the server off the net, the network is dead. On the other hand, latency becomes minimal, as the routing distance for one package needed to reach each node in the network is minimal (only one transition is needed). Bandwidth, however, is generally limited by the server's resources, making it hard to receive or transmit big chunks of data. Furthermore, it holds that all the routes in the network have the same length. This is something which is fundamentally more complex in decentralized networks and can be a great simplification of command distribution and monitoring the network.

Due to their nature, centralized architectures are usually implemented with an IRC C&C or similar[CJM05]. The central server is normally not owned by the botmaster. This would make detecting his identity easy. In many countries, launching an "evil-minded botnet" is a serious crime. Instead, hacked or public IRC servers are used as the central C&C node. A connection from the attacker's computer to the central server is often obfuscated by many in-between relays, tunnels and encryption. In figure 1 this is shown as the "steeping stones" which shall hide an attacker's idenity.

5.2 Decentralized Architecture

Dezentralized architectures do not rely on the special role of one central server. Instead, they are built upon the principal of equality, namely that the "peer nodes (both client and server) are all equal"[SI07]. The topology of the network is far more complex than in centralized architectures, forming a mesh as shown in figure 2. It is thus more difficult for a bot to join the botnet. Extensive bootstrapping is required, as the bot has to figure out an already-participating peer to connect to in the beginning. Once inside the net, information about other peers is exchanged between nodes. There are two approaches for bootstrapping[WWAZ09]:

- A list of peers likely to be online is hardcoded into the client. This list can later be updated
- A shared web cache on the internet stores information about peers. The address of which is hardcoded.

As can be seen, Bootstrapping is a critical and vulnerable point in any P2P botnet. Considerable efforts by botmaster have been made to circumvent the need to bootstrap[WSZ10]. This is further discussed in 7.

Once inside the net, information about other peers is exchanged between nodes.

Distributing commands and data in such a network is complicated, as it has to be assured that the message reaches all clients. As a general rule of thumb, the better inter-connected the nodes are, the higher the probability for a message to reach all recipients.

A decentralized network has the advantage of having the accumulated resources and bandwidth of all the peers in the network available. However, latency might be bad, as routing through the network is not trivial (cf. figure 2). P2P networks are generally considerd to be harder to disable (cf. section 7 on page 10).

It is to be discussed whether P2P networks with a centralized server architecture for certain services like file-indexing — we refer to them as “Napster-like” botnets — fall into the “decentralized” category. Principally, the connection graph differs a lot from centralized networks, but centralized and Napster-like networks share the same weaknesses, as could be seen when Napster was shut down in 2001[Wik11].[1] Dittrich et al. [DD07] would consider Napster-like botnets a hybrid architecture, whereas Steggink et al. [SI07] classify it as decentralized. In the following, we stick with Steggink’s definition, as there seems to be a broader consensus in the literature towards their nomenclature (cf. [WSZ10]).

5.3 Hybrid Architecture

Hybrid architectures are botnet architectures driven by scientific development and up to this date only exist in theory. They have been described as “the advanced P2P botnet”[WSZ10] and the “super botnet”[VAJ07]. The approach is to study current botnets, analyze their weaknesses and propose a better solution. This anticipates how botmasters could improve their botnets in the future. This way, even today, we know what future botnets could look like and how to better defend against them.

The proposed new hybrid P2P botnets do not have a pre-set communication architecture, following the strict P2P definiton. Their network connectivity is solely determined by the *peer list* in each bot.

Only machines with static IPs appear as bots in the peer list, so-called *servents*.[WSZ10] This way, it is guranteed that the distributed peer lists are maximally deadlink-free. This is a specialization of the “all peers are equal” contract in P2P: Some bots — the servents — have special obligations described in the following. The clients are then typically bots behind firewalls, machines with private or dynamic IPs.

Network construction phase Infection is done no different than in conventional botnets (cf. 4.1). The basic construction procedure has two mechanisms:

[1] This was performed as an act of cofirming with the decision made in the intellectual property case of US AM Records, Inc. v. Napster, Inc., 239 F.3d 1004 (2001) and not an explicit attack against the Napster network, but the fact that the Napster network could so easily stop the network by just disconnecting its central server shows the inherent weakness of centralized architectures.

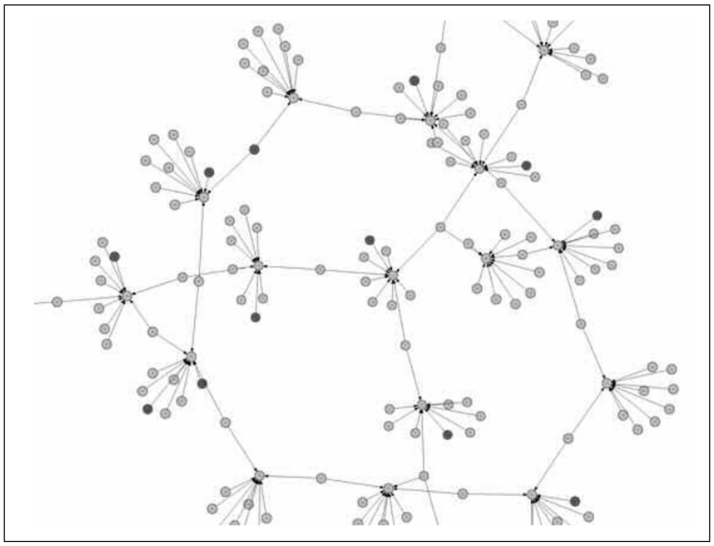

Fig. 2. P2P network (Source: [DD07])

– **New Infection:** "Bot A passes its peer list to a vulnerable host B when compromising it. If A is a servent bot, B adds A into its peer list (by randomly replacing one entry if its peer list is full). If A knows that B is a servent bot (A may not be aware of B's identity, for example, when B is compromised by an e-mail virus sent from A), A adds B into its peer list in the same way." [WSZ10]
– **Reinfection:** "If bot A reinfects bot B, bot B will then replace [a series of] randomly selected bots in its peer list with [...] bots from the peer list provided by A. Again, bots A and B will add each other into their respective peer lists if the other one is a servent bot."[WSZ10]

Both the advanced P2P botnet and the super botnet have their own P2P protocols for C&C. They implement push and pull C&C.[WWAZ09] When a bot receives a command it forwards it to all the peers in its list (push). If a bot cannot accept incoming connections (due to network misconfiguration, or a firewall), it actively polls other peers in its connection list from time to time to receive new commands (pull, cf. 6.1 on page 10).

Lifetime Comparison against superbot net In the following, we will concentrate on the "the advanced P2P botnet" as described by [WSZ10]. It was shown by Wang et al. [WSZ10] that the super botnet suggested by Vogt et al. is not likely to become successful in a real world scenario:

– In contrast to what Vogt et al. assume, most of the compromised computers cannot act as servents (due to a firewall, NAT[2] or dynamic IP address) in the real world.
– Even though Vogt et al. demonstrated the robustness of their constructed botnet (cf. [VAJ07]), they rely on the assumption that enough reinfections will occcrr during the early lifetime of the botnet, namely the buildup-phase. According to [WSZ10] this is a false assumption as heavy reinfection-seeking during buildup will lead to easy detection of the botnet and a lot of wasted resources. Following this approach, Wang et al. showed that the super botnet algorithms for propagation — namely only "new infection" and "reinfection" — require over 200,000

[2] Network Address Translation, occurs behind a router

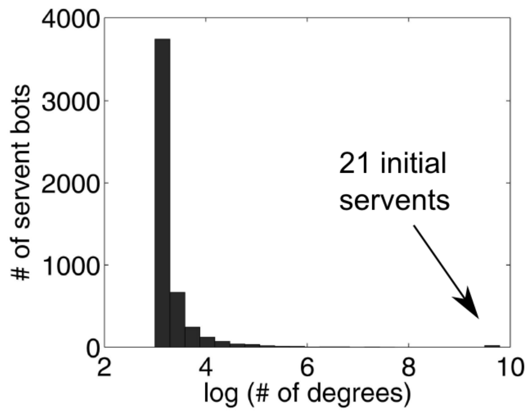

Fig. 3. Degree-distribution graph of servent bots assuming only new-infections and reinfections (derived work, original source: [WSZ10]). Simulation constraints: possible vulnerable hosts $n = 500,000$, stop of growth of botnet after $n = 20,000$, peer list size $M = 20$, 21 initial servent bots.

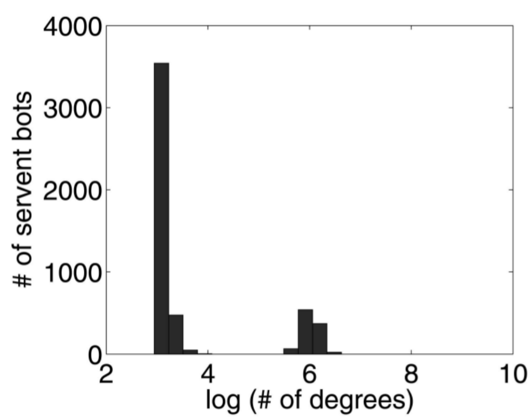

Fig. 4. Degree-distribution graph of servent bots simulating peer list update propagation, also (Source: [WSZ10]). Most notably, the 21 initial servent bots cannot be spotted. Instead, a robust backbone for the P2P network has formed out of 1,000 servent bots v (around x-axis 6), $deg(v) \in [300; 500]$.

infections events to create an evenly balanced botnet of only 20,000 vulnerable bots. As a result, when not enough reinfections occur, a scenario as depicted in figure 3 arises: Because the botnet stops growth after having infected 20,000 hosts, but there is such a huge amount of vulnerable hosts (500,000), a reinfection event rarely ever happens. This means that servent bots only seldom exchange parts of their peer lists. As a consequence, the connection to servent bots is extremely unbalanced: In the network graph, 80% of servent bots have a degree less than 30, while the initial 21 servents have degrees between 14,000 – 17,000: Most of the newly introduced servents have very few peers (both clients and servents) connected to them, essentially degrading the P2P botnet to a central network with 21 main servers. This is by no means an ideal P2P botnet. When it is not possible to have enough reinfections, the "new infection" and "reinfection" propagation measures are obviously not enough.

Peer list updating Because of this problem, Wang et al. propose a new, third propagation method: Peer list updating. The idea behind this is that bots update their peer lists frequently. However, this imposes a severe security problem: An attacker capturing only one bot could soon reveal the identity of many servents in the network. Thus, a new command is introduced: Enforced peer list updating. As described in [WSZ10], it is possible for a botmaster to monitor his botnet, i.e. determine how many servents exist. After a sufficiently large time after construction phase, he can enforce a peer list update: All bots obtain a new peer list from a specified *sensor host*. This sensor host is equipped with the knowledge of all the servents in the network by the monitor-command issued by the botmaster. Upon query, the sensor host creates a peer list in the following way: It randomly chooses servent bots, composes an updated peer list out of them and sends it back to the querying bot. After each peer list update command, all bots will have "uniform and balanced connections."[WSZ10]

The network graph obtained is then similar to the one shown in figure 2 on page 6.

It is to be discussed at which point in time it makes sense for the botmaster to enforce a peer list update, for every update command bears the risk of discovery of parts of the network. This is further discussed in [WSZ10]. Simulations with an update after the first 1,000 infections show that this update strategy will result in a degree distribution depicted in figure 4: The first 1,000 servents have many balanced connections ($deg(v) \in [300; 500]$), forming the robust backbone and connecting the hybrid P2P network tightly together. However, the remaining 4,000 servents (connected to the network after the update-command was run) have the known symptom of having degrees between 20-30, a situation well-known from the simulations of the superbot net (cf. figure 3). Thus, a forced peer list update from time to time seems necessary to make for a good P2P infrastructure.

Further improvements The proposed hybrid network has several other advantages over common, existing P2P networks: It doesn't need bootstrapping, removing a single point of failure. Bootstrapping is avoided due to the three propagation measures (new infections, reinfections and forced peer list updates). An initial peer list is simply passed on to the newly infected zombie by the machine infecting it. Due to its fixed size peer list ($M = 20$ in the simulations for 3, 4), when an attacker gets access to a bot, it doesn't reveal whole (sub-)nets. Only machines with static IPs appear as peer bots, so-called *servents*. This way, it is guranteed that the distributed peer lists are maximally deadlink-free. Data encryption in the hybrid P2P botnet has two functions: First, it makes it hard to sniff for patterns in internet traffic to detect a possible botnet. Second, the authenticity of the issued commands

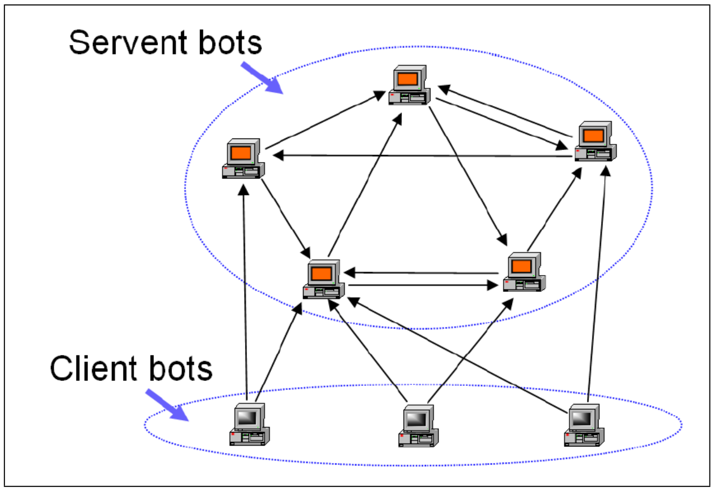

Fig. 5. Hybrid network (Source: [WSZ10])

can be verified so that only commands signed by the botmaster are executed. The bot software can be shipped with a hardcoded public key from the botmaster.

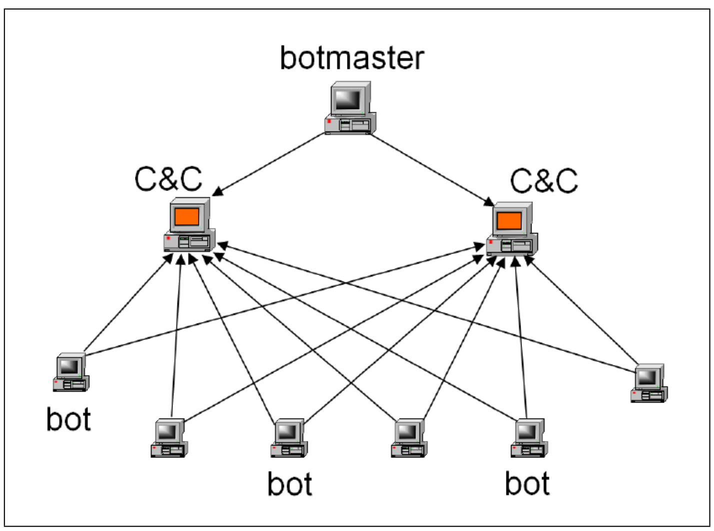

Fig. 6. Another graphical representation of a central network (Source: [WSZ10])

Compared to a central-strucutre botnet (see 6), the hybrid P2P net (see 5) is only an extension of the orginial network: It is essentially equivalent to a central architecture P2P net. However, the amount of servers (in the form of servents) is greatly increased, as is the number of interconnections between them. The great number of servents is the primary reason why the hybrid P2P botnet is supposedly very hard to shut down.[WSZ10]

As tempting as such an advanced new botnet protocol may sound, it has to be acknowedlged that the hybrid P2P botnet is a purely academical, theoretical idea that has never been tested in practice: "The network may not be as stable and robust as expected due to complex network conditions and defenses."[WWAZ09]

6 Command and Control structures in P2P botnets

Central server, hybrid, completely decentralized

Dezentralized: Peacom p.7 in 10.1.1.112.3561.pdf see p. 3 in 10.1.1.153.8296 authentication of commands

6.1 Push/Pull mechanism

Commands can be distributed in two ways: Either via a push or a pull mechanism. "Pulling", being the more trivial of the two approaches, is the process of a client actively asking a server whether there's new instructions for it. To be up-to-date this has to be perforemd periodically, increasing network load. Push on the other hand is technically more advanced, as commands from the server will be automatically "forwarded" to the clients — the difference between push and pull in a bot is much like IMAP IDLE compared to POP3 for mail: With POP3, the user client has to periodically ask the server whether there are new messages available (pull), whereas with IMAP, the client gets a new message pushed by the server. This has the advantage of reducing the network traffic, but it requires that the server can open a connection to the clients (cf. 5.3).

7 Detection of and Counter measure against evil P2P botnets

What can an attacker do to de-arm a botnet at all? The ideal solution would be to shutdown all participants in the botnet (and only those). Realistically, this is almost never possible. It is also illegal in most countries to start a botnet, as it implies gaining access to another computer. This falls into the category of hacking. On the other hand, paradoxically, shutting down a botnet is likewise illegal, for it normally requires the attacker — be he benign or not — to do the same: Hacking into computers.

Common sub-tasks in fighting botnets are therefore, from an attacker's perspective:

- Detecting the botnet at all, which can be very difficult for P2P parasite and leeching networks, as a separation between "bot" P2P and "real" P2P datastreams has to be made.
- Analyzing the botnet: Finding the servers (if there are any) and participating zombies. It is also helpful to be able to estimate the size of the botnet. When using a technique like sybil attacks, this gives an estimation of how many resources are needed, i.e. how much money will be needed to take down the botnet.
- Preventing further spread of the botnet: Fixing the security exploit of the bot's injection vector. Most often, when a bot installs itself unnoticed from the owner (*injection*), it does so by using a security problem (*exploit*) in the software. If the manufactorer fixes and updates this security hole, spread of the bot cannot continue on upgraded systems.
- Disabling the bot(sub-)nets: Making clients loose inter-connection (in P2P networks), shutting down central servers (in central networks).
- Infiltrating botnet to do non-malicious tasks, i.e. not to forward and not to execute incoming commands.

It must be noted that these tasks are of generic nature and might not be applicable to the botnet at hand. However, they represent a solid way of taking down a botnet.

While it is — in theory — realtivley easy to shutdown a centralized architecture by determining the central servers and disabling those (e.g. through DoS attacks[3]), P2P networks are arguably harder to disable, given they are properly protected. The legal implcations of shutting down a botnet are also to be considered: When proven to be malicious, it is possible for the police to get a warrant to cut-off the servers physically from the net. This cannot pratically be done for non-centralized P2P networks, since the bots are usually private property, with their owners unwillingly taking part in the botnet. Additionally they exist in so great masses that a physical take-down would be impossible to coordinate.

There are three well-known techniques to fight P2P botnets in particular.

7.1 Index poisoning

Almost all P2P file sharing systems have so-called *indices*, in which they map a desired content to a location where it can be found (IP address and port). An index can thus be considered as a "lookup table" for finding the information where a user can retreive a certain content he searches for. Therefore, indices are a vital part of any such P2P system emplyoing them. Depending on the architecture of the P2P net (cf. 5), the index might be stored in a central position (Napster-like filesharing systems), or be distributed over a larger number of nodes. [LNR06]

The principal idea of index-poisoning is given by [WWAZ09]: "Originally, index poisoning attack was introduced to prevent illegal distribution of copyrighted content in P2P networks. The main idea is to insert [a] massive number of bogus records into the index. If a peer receives [a] bogus record, it could end up not being able to locate the file (nonexistent location), or downloading the wrong file."

It is easy to transfer this idea to the C&C of botnets. You only have to think that bots are not usually looking for files, but for commands. Once you know under which keys the botnet commands are stored in the index records, an attacker trying to shutdown the network can insert false commands under the same keys. If there is enough false information, chances to hit the real command issued by the botmaster are slim. The index gets "flooded" or poisoned by wrong commands. These can either be NOPs (no operations, meaning a command that does nothing), or may even help to disguise the botnet. Because of its principal of inserting faulty records into a system, index poisoning is also referred to as a "pollution attack"[LNR06].

Index-poising has been reported to have succeeded in fighting two recent P2P bots, namely Trojan.Peacomm and Stormnet. ([GSN+07], [LNR06])

7.2 Sybil attack

A *sybil attack* on a botnet is a form of attack where large quantities of fake nodes (the so-referred to sybils) are inserted into the botnet. By infiltrating the botnet's peer list, the sybils try to interrupt the C&C between real bots. In contrast to the more statical pollution attack, a sybil attack requires the attacker to have active hardware acting as the sybils and also to take part in the P2P network. Note however, that it is not neccessary and even contra-indicative for the sybils to execute or distribute the commands sent over the P2P network's C&C structures. The idea of the sybil attack lies in the fact that there exist so many sybils in the network that issued commands get neither distributed nor executed, really. [DFNM]

Generally speaking, it is more expensive to run a sybil attack than an index-poisoning attack, as the former requires far more infrastructure at the disposal of the attacker.

[3] Denial of Service attacks, i.e. generating so much traffic the target cannot function normally any more

Botmasters could utilize known P2P technique to fight against the weakness of sybil attacks: According to [Dou02], it is common practice for a P2P system to distribute computational or stoarge tasks among remote peers to protect against the threat of hostile peers. For if a complete task is given to a hostile peer only, chances are the task is never executed and completely lost. However, this risk is mitigated when the task is replicated to several peers, as it is unlikely for all peers to be hostile. In this way, a botnet could at least slightly compensate for an ever-growing mass of sybils in that its commands get executed anyway with a high probability (determined by the level of redundancy introduced).

This does not solve the problem of sybils interrupting communication in the first place. Countermeasures against this are presented in 5.3 on page 5.

Sybil attacks have been simulated to be a valid countermeasure against the Storm botnet.[DFNM]

7.3 Compromising Bootstrapping

Bootstrapping is a vulnerable point in any P2P botnet. When a hardcoded peer list is used (cf. 5.2 for details), it is sufficient to take down all the peers in the bootstrapping table for the network to eventually shutdown: New bots simply can't find an initial peer to connect to. Botmasters have reacted to this by providing a Gnutella-like web-cache or updateable bootstrapping tables. As interrupting the bootstrapping process can so easily be done, it has been completely removed in the hybrid botnets (cf. 5.3 on page 8).

8 Conclusions and Outlook

This paper gave an overview of the state-of-the-art in P2P botnets. While [WWAZ09] conclude that a shutdown of a P2P botnet is principally no harder than that of a central network when employing the index poisoning technique, our understanding and findings, as shown in this paper, are substantially different and backed-up by both reason and literature ([ZC06], [BH09]). Additionally, new P2P protcols can be made resistent to index poisoning (shown in the same paper, [WWAZ09]), or file indices can be completely ommited, whereas the central nature of the servers remains a single point of failure in centralized architectures.

In fact, because they are harder to disable, more agile and dynamic in their lives, we believe that P2P botnets will play an ever-growing role in the future. It can also be cheaper to setup a P2P botnet, since there should be no costs for central servers. Centralized architectures on the other hand have been seen to decline and will probably continue to do so. Also, some signifcant research for improvement in the exisiting P2P protocols has been conducted, removing many of the current weaknesses of P2P architectures.

References

ARZMT06. M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52. ACM, 2006.

Ben11. A. Benzin. The first cyberweapon? 2011.

BH09. A. Berger and M. Hefeeda. Exploiting sip for botnet communication. In *Secure Network Protocols, 2009. NPSec 2009. 5th IEEE Workshop on*, pages 31–36. IEEE, 2009.

Bor. R. Borgaonkar. An analysis of the asprox botnet. In *Emerging Security Information Systems and Technologies (SECURWARE), 2010 Fourth International Conference on*, pages 148–153. IEEE.

CJM05. E. Cooke, F. Jahanian, and D. McPherson. The zombie roundup: Understanding, detecting, and disrupting botnets. In *Proceedings of the USENIX SRUTI Workshop*, pages 39–44, 2005.

DD07. D. Dittrich and S. Dietrich. Command and control structures in malware. *Usenix magazine*, 32(6), 2007.

DFNM. C.R. Davis, J.M. Fernandez, S. Neville, and J. McHugh. Sybil attacks as a mitigation strategy against the storm botnet. In *Malicious and Unwanted Software, 2008. MALWARE 2008. 3rd International Conference on*, pages 32–40. IEEE.

Dou02. J. Douceur. The sybil attack. *Peer-to-peer Systems*, pages 251–260, 2002.

FMC. N. Falliere, L.O. Murchu, and E. Chien. W32. stuxnet dossier. *Symantec Security Response*.

fSidIB11. Bundesamt für Sicherheit in der Informationstechnik (BSI). Eröffnung des nationalen cyber-abwehrzentrums. `http://www.bmi.bund.de/SharedDocs/Termine/DE/2011/cyber.html?nn=373114`, June 2011.

GSN+07. J.B. Grizzard, V. Sharma, C. Nunnery, B.B.H. Kang, and D. Dagon. Peer-to-peer botnets: Overview and case study. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, pages 1–1. USENIX Association, 2007.

Hru11. Joel Hruska. Security admin, botmaster sentenced to four years in prison. `http://arstechnica.com/security/news/2009/03/security-admin-botmaster-sentenced-to-four-years-in-prison.ars`, June 2011.

Inc11. Napster Inc. Napster. `http://www.napster.com`, June 2011.

irc. Irc - protocol defintion. `http://tools.ietf.org/html/rfc1459section-1`.

LJZ. C. Li, W. Jiang, and X. Zou. Botnet: Survey and case study. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 1184–1187. IEEE.

LNR06. J. Liang, N. Naoumov, and K.W. Ross. The index poisoning attack in p2p file sharing systems. In *IEEE INFOCOM*, volume 6. Citeseer, 2006.

New11. C. T. News. Expert: Botnets no. 1 emerging internet threat. `http://www.cnn.com/2006/TECH/internet/01/31/furst/`, June 2011.

oC11. University of California. Seti@home. `http://setiathome.berkeley.edu/`, June 2011.

SFS05. D. Schoder, K. Fischbach, and C. Schmitt. Core concepts in peer-to-peer. *Peer-to-peer computing: the evolution of a disruptive technology*, page 1, 2005.

SI07. M. Steggink and I. Idziejczak. Detection of peer-to-peer botnets. *University of Amsterdam, Netherlands*, 2007.

SK07. R. Schoof and R. Koning. Detecting peer-to-peer botnets. *University of Amsterdam*, 2007.

VAJ07. R. Vogt, J. Aycock, and M. Jacobson. Army of botnets. In *Network and Distributed System Security Symposium (NDSS)*. Citeseer, 2007.

Wik11. Wikipedia. Napster. `http://en.wikipedia.org/wiki/Napster`, June 2011.

WSZ10. Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing*, 7:113–127, 2010.

WWAZ09. P. Wang, L. Wu, B. Aslam, and C.C. Zou. A systematic study on peer-to-peer botnets. In *Computer Communications and Networks, 2009. ICCCN 2009. Proceedings of 18th Internatonal Conference on*, pages 1–8. IEEE, 2009.

ZC06. C.C. Zou and R. Cunningham. Honeypot-aware advanced botnet construction and maintenance. 2006.