An Introduction to Botnet Communication

Marlou Pors (4008847 - M.L.Pors@student.tudelft.nl)

ET4397IN - Network Security
Delft University of Technology

***Abstract*—Botnets are networks formed by "enslaving" host computers, called bots, that are controlled by one or more attackers, called botmasters, with the intention of performing malicious activities. The bots and botmaster(s) communicate via a command and control channel (C&C), where the actual communication is mostly accomplished by well-known protocols such as IRC, HTTP and P2P. This survey covers used C&C infrastructures, used protocols as IRC, HTTP and P2P as well as newly communication strategies which follow the trend of covert communication. We will also discuss to what extend proposed communication protocols are seen in current botnets.**

I. Introduction

Networks formed by malware-compromised machines, also known as botnets, have become a serious threat to the Internet. In 2012, there was stated that around 16-25% of the computers connected to the Internet were members of botnets [1]. Botnet activity is responsible for huge financial losses to Internet Service Providers, private companies, governments and home users. For example, the global cost of spam in 2007 has been estimated to have been US$ 100 billion, with US$ 35 billion allocated to the US alone.

In botnets,"enslaved" host computers are called bots. The one(s) which control these bots with the intention of performing malicious activities are called botmasters [1]. The bots and botmaster(s) communicate via a command and control channel, also called the botnet C&C. Namely, communication is needed to follow the life cycle of a bot, which is depicted in Fig. 1. For example, after the infection of a computer, this computers needs more data to complete its evolution to a function able bot. The bot retrieves this data in form of a secondary injection (phase 2). The C&C is also used to distribute the commands to perform malicious activities and to maintain the existing botnet (phase 3 to 5).

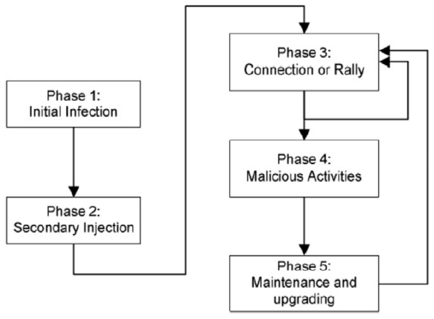


Figure 1: The life cycle of a botnet [1]

The actual communication between bots is mostly accomplished by well-known protocols, such as IRC, HTTP and P2P protocols. But, both botnet creators as researchers are constantly looking for new communication techniques which can be applied in botnets. Botnet creators want to avoid detection mechanisms researchers have come up with and researchers want to get insight into possible new strategies malware creators can come up with next, avoiding their detection mechanisms. In this 'battle', new trends that are seen are the construction of complex botnet infrastructures and the usage of protocols which can be exploited to create covert communication.

This survey covers both the well-known protocols as well as new proposed communication strategies for botnets. The remainder of this paper is organized as follows. In Section II we discuss C&C infrastructures and the history of botnets. In Section III to Section VI, we discuss the different communication protocols and strategies. We end this survey with a discussion to what extend analyzed and in literature proposed protocols are (going to be) used in real botnets. Section VIII stretches out how the interested reader can advise the literature used for this survey to gain more information.

II. Background on Botnet Command and Control (C&C)

The C&C infrastructure typically serves as the only way to control bots within the botnet and is necessary for maintaining a stable connection within the botnet to operate efficiently. C&C infrastructures can be centralized, decentralized or have a random topology [2]. One or more communication protocols are used by the botmaster(s) to command the bots and coordinate their actions. In some infrastructures bots can also communicate with each other in order to exchange commands or synchronization data.

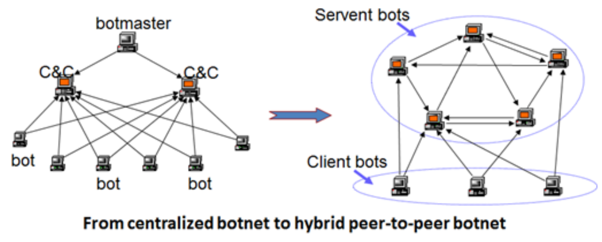


Figure 2: Different C&C infrastructures [3]

A. C&C Infrastructures

Naturally, a centralized topology is characterized by a central point that forwards messages between bots, this central point mostly being a server on which the botmaster places his commands. Messages sent in such a system tend to have low latency as they only need to transit a few well-known hops. Weakness of such systems is that they are easier to detect since many bots connect to the same point. The discovery of the central location can compromise the whole system; in other words, most centralized botnets suffer from a single point of failure.

Peer-to-Peer (P2P) botnet communication, which is an example of a decentralized (or distributed) infrastructure, has the advantage over centralized topologies that it is much harder to be disrupted. Such botnets make use of existing of modified P2P communication protocols, on which we elaborate in Section IV.

A third model for command and control is a random topology. Here, the communication system is based on the principle that no single bot knows about any more than one other bot. In this topology a bot or controller that wanted to send a message would encrypt it and then randomly scan the Internet and pass along the message when it detected another bot [2]. Although the design of such a system is relatively simple, the detectability and message latency is high.

From the summary on the history on botnets below it becomes clear that throughout the years the focus has shifted from centralized to decentralized botnets. Almost no real botnet makes use of a random topology. This is why we will not elaborate on that any further.

B. Botnet History

Historically, botnets originated from the Internet Relay Chat (IRC), a text-based chat-system that organizes communication in channels [1]. There, the main idea behind the botnets was to control interaction in the chat rooms and for example provide administration support. The first known IRC bot was Eggdrop (1993), which used the IRC communication channel to distribute commands to the bots. Some literature state that Eggdrop was a non-malicious botnet [4].

IRC botnets, which have a centralized architecture, suffered from the exposure to single point-of-failure, making them easy to be shut down when detected. New bots evolved using new available protocols and architectures to cope with previous known vulnerabilities. Newer centralized botnets started to use different protocols, such as HTTP. Besides this, bots started to spread through networks such as file-sharing networks and peer-to-peer networks. A game-changing event was the appearance of the *Nugache* Worm, first detected in 2005, considered to be responsible for the creation of one of the first botnets with a successfully distributed C&C infrastructure, based on a P2P protocol [5].

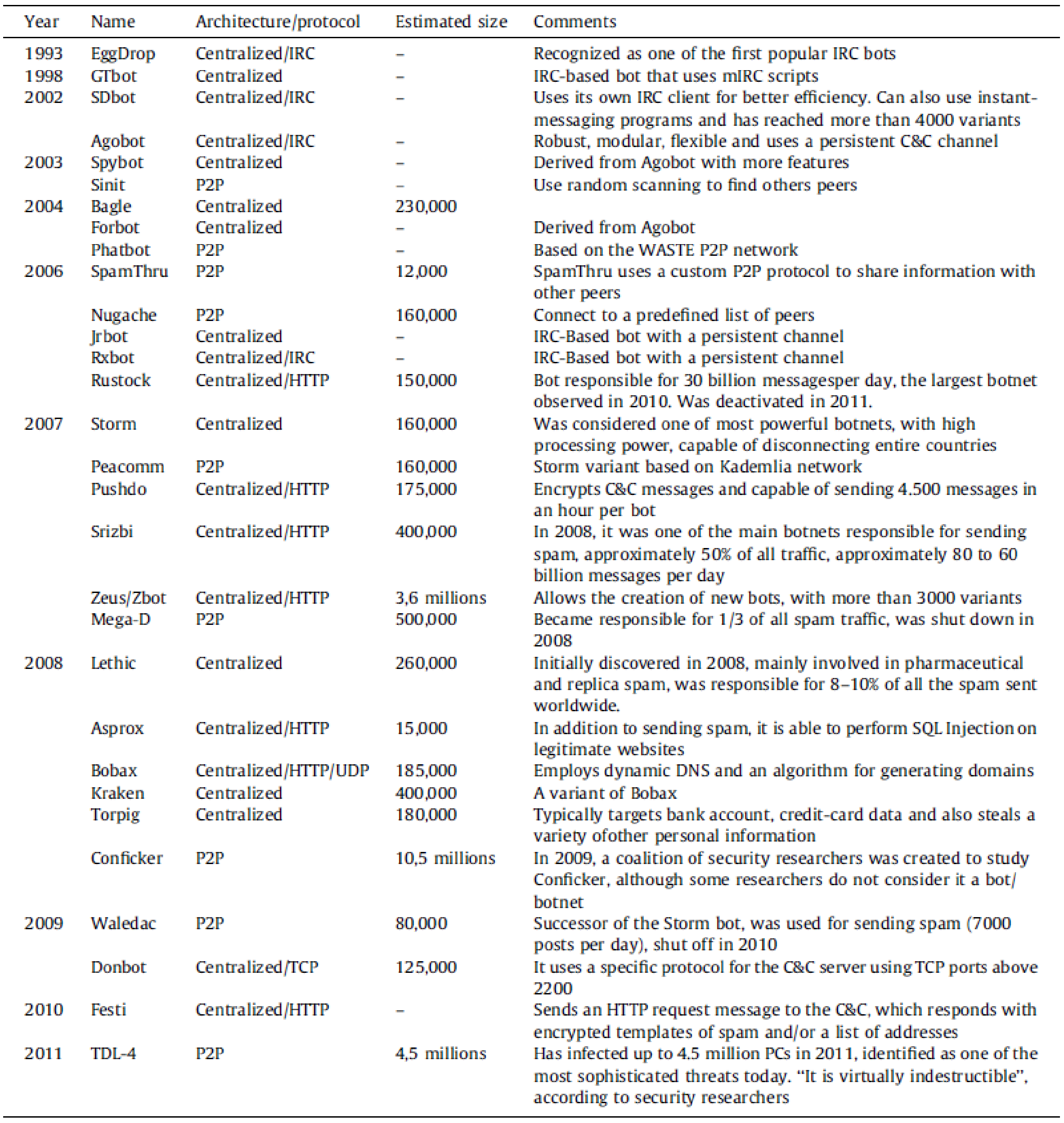
Fig. 3 presents a botnet timeline containing important real bots and their main features. From this, we can clearly see how the usage of different protocols has shifted and that the biggest botnets use P2P. Newer botnets used modified and complex infrastructures and start to use the HTTP protocol. Also, the usage of DNS is seen as an improvement of the usage of P2P. A known botnet that uses DNS is *Feederbot* (2011) [6]. Besides this, researchers find new protocols that can be exploited for malicious use, such as the Session Initiation Protocol (SIP), which appears in studies since 2009.

III. IRC-Based Botnets

From Section II it follows that the first botnets had a centralized infrastructure and used the IRC protocol. The Internet Relay Chat (IRC) protocol is a protocol which enables virtually instant communication with a public exchange point [2]. It provides a common protocol that is widely deployed across the Internet and has a simple text-based command syntax. There are a large number of existing IRC networks that can be used as public exchange points. Most of these networks lack any strong authentication and a number of tools to provide anonymity on IRC networks are available. Thus, IRC provides a simple, low-latency, widely available and anonymous command and control channel for botnet communication where each bot can log into an IRC channel to seek for the commands from the botmaster.

Well known real botnets which are IRC-based are *Agobot*, *SDbot*, *Spybot* and *GTbot* [1], [7]. From Fig. 3, we can see that these botnets are relatively old and originate from the beginning of the botnet era.

IRC-based botnets are brittle, because firewall can avoid them by filtering related ports [7]. And again, because IRC-based botnets are centralized, they suffer from single point of failure; if the C&C server(s) is blocked by law enforcement, the bot network is useless. Nowadays a lot of detection techniques exist to identify malicious behavior over IRC.



Year	Name	Architecture/protocol	Estimated size	Comments
1993	EggDrop	Centralized/IRC	–	Recognized as one of the first popular IRC bots
1998	GTbot	Centralized	–	IRC-based bot that uses mIRC scripts
2002	SDbot	Centralized/IRC	–	Uses its own IRC client for better efficiency. Can also use instant-messaging programs and has reached more than 4000 variants
	Agobot	Centralized/IRC	–	Robust, modular, flexible and uses a persistent C&C channel
2003	Spybot	Centralized	–	Derived from Agobot with more features
	Sinit	P2P	–	Use random scanning to find others peers
2004	Bagle	Centralized	230,000	
	Forbot	Centralized	–	Derived from Agobot
	Phatbot	P2P	–	Based on the WASTE P2P network
2006	SpamThru	P2P	12,000	SpamThru uses a custom P2P protocol to share information with other peers
	Nugache	P2P	160,000	Connect to a predefined list of peers
	Jrbot	Centralized	–	IRC-Based bot with a persistent channel
	Rxbot	Centralized/IRC	–	IRC-Based bot with a persistent channel
	Rustock	Centralized/HTTP	150,000	Bot responsible for 30 billion messagesper day, the largest botnet observed in 2010. Was deactivated in 2011.
2007	Storm	Centralized	160,000	Was considered one of most powerful botnets, with high processing power, capable of disconnecting entire countries
	Peacomm	P2P	160,000	Storm variant based on Kademlia network
	Pushdo	Centralized/HTTP	175,000	Encrypts C&C messages and capable of sending 4.500 messages in an hour per bot
	Srizbi	Centralized/HTTP	400,000	In 2008, it was one of the main botnets responsible for sending spam, approximately 50% of all traffic, approximately 80 to 60 billion messages per day
	Zeus/Zbot	Centralized/HTTP	3,6 millions	Allows the creation of new bots, with more than 3000 variants
	Mega-D	P2P	500,000	Became responsible for 1/3 of all spam traffic, was shut down in 2008
2008	Lethic	Centralized	260,000	Initially discovered in 2008, mainly involved in pharmaceutical and replica spam, was responsible for 8–10% of all the spam sent worldwide.
	Asprox	Centralized/HTTP	15,000	In addition to sending spam, it is able to perform SQL Injection on legitimate websites
	Bobax	Centralized/HTTP/UDP	185,000	Employs dynamic DNS and an algorithm for generating domains
	Kraken	Centralized	400,000	A variant of Bobax
	Torpig	Centralized	180,000	Typically targets bank account, credit-card data and also steals a variety ofother personal information
	Conficker	P2P	10,5 millions	In 2009, a coalition of security researchers was created to study Conficker, although some researchers do not consider it a bot/botnet
2009	Waledac	P2P	80,000	Successor of the Storm bot, was used for sending spam (7000 posts per day), shut off in 2010
	Donbot	Centralized/TCP	125,000	It uses a specific protocol for the C&C server using TCP ports above 2200
2010	Festi	Centralized/HTTP	–	Sends an HTTP request message to the C&C, which responds with encrypted templates of spam and/or a list of addresses
2011	TDL-4	P2P	4,5 millions	Has infected up to 4.5 million PCs in 2011, identified as one of the most sophisticated threats today. "It is virtually indestructible", according to security researchers

Figure 3: Botnets Timeline [1]

IV. Peer-to-Peer botnets

When it turned out that IRC-based protocols were easy to take down, botnet creators focused on P2P-based botnets. The P2P (decentralized) architecture is seen as a true replacement for the centralized C&C communication [7]. In this model, bots communicate with each other, transferring received commands and new updates from the botmaster among each other. Fig. 2 shows the evolution from centralized to peer-to-peer botnets. It is directly clear that this architecture does not suffer from a single point of failure.

P2P botnets can be divided in three different classes: *parasite*, *leeching* and *bot-only* P2P botnets. These classes represent in what form the P2P network is used as part of the botnet. In a parasite P2P botnet, all the bots are selected from vulnerable hosts within an existing P2P network and it uses this available P2P network for command and control. In such a botnet, it is not necessary to perform any further action to form the botnet, because bots can find and communicate with each other by simply using the current P2P protocol. A leeching P2P botnet refers to one whose members join and depend on an existing P2P network, but the bots could be vulnerable hosts that were either inside or outside this existing P2P network. The procedure of a bot becoming part of a P2P botnet is called the *bootstrap* process. A bot-only P2P botnets builds its own network, in which all the members are bots.

In bot-only botnets, most of the time a self-defined or adopted P2P communication protocol is used. Leeching and parasite P2P botnets can leverage existing P2P protocols. This has several advantages. First, these protocols have been tested in P2P file-sharing applications for a long time, so they tend to be less error-prone than newly designed ones. Second, they have nice properties to improve performance of P2P systems and mitigate network problems, such as link failure or churn.

One kind of protocol that has been found in early P2P networks are protocols that are based on the *Kademlia* algorithm, a distributed hash table which can be used to implement so-call structured P2P overlays [8]. Examples which use this are *Bittorrent*, *Gnutella* and *Overnet*.

A simple P2P botnet

A well known real P2P botnet is *Trojan.Peacomm* [4], which uses the Overnet network and thus leverages the Kademlia protocol. Peacomm is a P2P variant of *Storm*. As Fig. 3 states, Storm was considered one of the most powerful botnets.

[4] states that the communication protocol for the Trojan.Peacomm bot can be divided into five important steps. First, the bot publishes itself on the Overnet network and connects to a hard coded list of peers. The other steps are part of the download and execution of the secondary injection (phase 2 in Fig. 1). The content of this secondary injection determines the functionality of the bot (for example the injection can contain a SMTP email spamming component). When the botmasters want to give the bot other functionality, this is done with a new injection distributed over the network. In this way, the total functionality of the botnet can change over time.

One weakness of Peacomm is the hard coded list of peers for the bootstrap procedure. Namely, if these peers stop responding to requests to join the Overnet network, then the Peacomm binary will fail to bootstrap and download the secondary injections.

Complex P2P botnets

As stated above, some P2P botnets use self-defined or adopted P2P communication protocols. An advantage of this is that a botnet can be more flexible if it uses a new protocol designed by its botmaster.

A first kind of complex botnets is a *hybrid* P2P botnet, which is shown to be much harder to be shut down [3]. Important properties of this botnet is that the bootstrap procedure is avoided and that the servent bots take the role of the command and control servers as shown at the right of Fig. 2. In this way, a botmaster can inject her commands through any bot(s) in the botnet. An extra property is that each bot in the network has a different peer list containing only a part of the complete network. The connectivity (but also the difficulty of bringing the botnet down) depends on these peer lists.

The other complex P2P botnet is a so-called *super botnet* [9], where the botnet is composed of a number of small centralized botnets. Here, commands are pushed from one small botnet to another and within a small centralized botnet, bots can pull the command from their C&C servers.

Both the hybrid botnet as the super botnet are proposed infrastructures. The researches do not elaborate on the precise communication rules that could be used inside the infrastructures. These could be from complete new protocols, supporting the new infrastructures, but one could also use parts of existing protocols such as HTTP to complete the communication between the bots (and botmaster).

V. HTTP-Based Botnets

A kind of botnet which started as one that resides between IRC-based botnets and current P2P botnets is HTTP-based botnets. Although this protocol can be used in P2P networks, most of the HTTP-based botnets use a centralized C&C infrastructure. From Fig. 3, one can see the popularity of using HTTP. In fact, currently HTTP-based botnets are seen as the most dangerous ones [10]. This is because botmasters can use the HTTP protocol to hide their activities among the normal web flows and easily avoid current detection methods like firewalls and is thus a form of covert communication.

A simple centralized HTTP botnet

An example of a simple centralized botnet which uses HTTP is *BlackEnergy* (2007) [11]. BlackEnergy is a web-based distributed denail of service (DDoS) botnet, which was mostly used by Russian hacker underground. The HTTP C&C was built on PHP and MySQL; the botnet configuration was stored in a simple MySQL database. This database stored three different types of commands: DDoS attack commands, download functionality commands and commands to stop the bot. The bot retrieves data by sending a POST message to the server and waiting for a response. The botmaster is given a simple PHP-based web UI to configure their botnet.

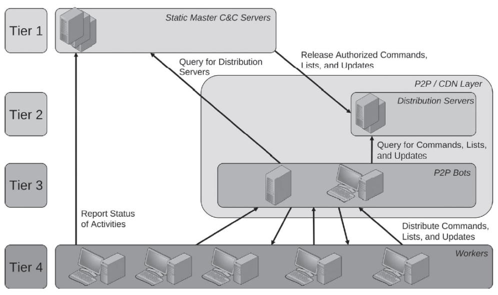

Tier 1
Tier 2
Tier 3
Tier 4
Static Master C&C Servers
Query for Distribution Servers
Release Authorized Commands, Lists, and Updates
P2P / CDN Layer
Distribution Servers
Query for Commands, Lists, and Updates
P2P Bots
Report Status of Activities
Distribute Commands, Lists, and Updates
Workers

Figure 4: Miner Botnet infrastructure [5]

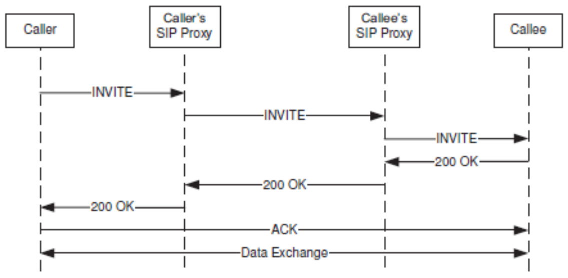


Figure 5: Basic SIP session establishment [13]

Complex HTTP botnets

HTTP can also be used to create complex botnets. An example of a more complex centralized HTTP-based is a modified version of the TDL-4 botnet (2012). This botnet uses a domain flux technique in HTTP to periodically change the C&C server domains to avoid detection methods. An example of a complex P2P botnet which uses HTTP is the P2P variant of Zeus (2012) [12], which is a significant evolution of earlier Zeus variants (such as the one from 2007 (Fig. 3)). Another example is the Miner botnet (2012) [5].

What is seen in both these botnets is a more complex infrastructure using more layers. AS an example, Fig. 4 shows the C&C infrastructure of the Miner Botnet. Here, the system is divided into four tiers, where the upper tiers are of more relevance from the botmaster's perspective. Tiers 2 and 3 form the P2P network of the Miner Botnet. The complex infrastructure does not complicate the communication protocol, as the structure of the P2P communication protocol which used HTTP is shared by all tiers.

VI. Covert Communication

As seen in HTTP-based botnets, a new trend in botnet communication is the exploitation of well-known protocols to create covert communication which can be used for botnet communication. Below we will shortly discuss possible exploits for SIP and DNS.

A. *SIP-based botnets*

As mobile handsets are getting more powerful in terms of processing and networking capabilities, they represent additional attractive targets for future botnets [13]. Because of this, research is done towards the possible exploitation of the Session Initiation Protocol (SIP), which is used as the prevalent signaling protocol for handling multimedia sessions over the Internet and 3GPP realms. [13], [14] show that SIP can be used as a covert channel for building a botnet C&C, where information is hidden in plain sight.

A basic SIP session establishment, depicted in Fig. 5, is one of the moments at which the protocol can be exploited. Another way to exploit the SIP protocol is via the Session Description Protocol (SDP) information residing inside SIP [14]. Namely, data inside the SIP messages is relayed unmodified by the SIP proxies, as it is considered to be application specific. Thus, configuration data and commands can easily be send as the message (data) inside the INVITE messages or SDP information.

Unlike in the IRC service, in SIP there is no rather easily detectable broadcasting of information to all connected hosts. On the other hand, the cost to establish and maintain a dedicated botnet topology is heavily reduced, as the SIP infrastructure was designed to be a stable and reliable signaling platform for arbitrary services [13], making it more favorable than P2P protocols. An extra advantage is that SIP bots can select both direct and indirect communication, such as it fits to the botmaster's strategy for the botnet.

[13] implements a botnet having the functionality of the well-analyzed *Storm* bot, where the entire botnet communication is tunneled over SIP. This is a good example of a research that is done to gain insight into new strategies hackers can develop to introduce new botnets. But, up to today, no real botnet using SIP as their communication protocol is analyzed. This could mean that no real botnet of today does use the SIP exploitation or that researchers have not found / detected one.

B. *A DNS-based botnet*

Another kind of botnet C&C channel which aims to hide their messages in common application layer protocols are botnets which exploit the Domain Name System (DNS). *Feederbot* [6] is a botnet which uses DNS tunneling to evade detection. In this approach, data is transmitted withing (resource record) fields of a DNS message.

Advantages of using DNS as the covert channel is that in environments with heavily restricted Internet access (e.g. by use of firewalls and proxies) DNS is usually one of the few protocols that is allowed to pass without further ado. Also, DNS was designed as a distributed system and as such provides advantages in terms of resilience, such as we have with P2P networks. An example of how Feederbot uses this distributed system is as follows: A bot receives instructions from several DNS C&C servers of which a very small subset are the bootstrapping C&C servers which are contacted when the bot is launched.

VII. Discussion & Conclusion

In this survey we introduced the basics of botnet communication and elaborated on different communication protocols such as IRC, HTTP and P2P. Following the botnet timeline (Fig. 3), we see that the popularity of centralized IRC-based botnet has decreased and most botnets (up to 2011) use P2P protocols or HTTP.

Currently, HTTP-based botnets are seen as the most dangerous ones [10]. Both botnet creators and researchers keep investigating new communication methods to respectively avoid detection techniques or come up with new communication techniques malware creator can also come op with (in order to create new detection techniques). HTTP-based botnets follow the trend that focus shifts towards covert channels for botnet communication. Namely, botmasters can use HTTP to hide their activities among the normal web flows.

Question arises to what extend communication strategies proposed by researchers, such as covert channels, are indeed strategies malware creators will use. In newer real botnets, such as Zeus and Miner (both 2012), we see that the botmasters stick to P2P but make the infrastructure more complex, using multiple layers. Besides this complex infrastructure, both botnets use HTTP. There are not yet found a lot of botnets which use a covert channel due to exploits in the SIP or DNS protocol; this survey treats the DNS-exploiting Feederbot (2011) shortly.

The fact that literature does not (yet) cover a lot of analyzed botnets which do implement covert communication can mean different things. First, although it seems a good opportunity for botnet communication, it could not (yet) be picked up by malware creators. Second, it could be picked up, but the created botnets are not yet found by malware researchers (which means a win for botnet creators). A disadvantage of malware research is that botnets (and other malware) can only be analyzed when it is found, and thus this kind of research is always lagging behind what happens at this very moment.

Our expectation is that current botnet creators keep focusing on HTTP, rather than trying other protocol exploits. For example, researchers elaborate on the SIP exploit in literature from 2009, but no botnet is analyzed which really does use this protocol. This while the literature also does cover some detection methods for these kind of botnets - so if they existed, they should have been found by now. Besides this, HTTP is a good example protocol which can be used in both a centralized and decentralized botnet topology. Another advantage for malware creators is that the protocol is not a complicated one to learn, while one can achieve various things with it.

Combining HTTP and P2P has proven to create powerful botnets, such as Zeus, which is one of the most dangerous HTTP-based botnets [10]. HTTP is relatively easy to use and can be used to successfully hide information. on the other hand, P2P structures provides a resilient network which is hard to break down even if it is detected. Why change a winning team?

VIII. Used Literature & Further Reading

This paper served as an introduction to botnets and their communication protocols (used and proposed ones). The interested reader is advised to start reading [1], which is a comprehensive survey on botnets in general. This survey also contains an elaborate summary on botnet detection, an other focus topic within botnet research. [1] contains the presented botnets timeline (Fig. 3) including references to all botnets.

One who is interested to look further into the used literature should note that the advised papers can be sorted in three categories. Papers which contain surveys on botnets are [1], [2], [7], [15], [16]. Here, [15] examines and compares the attacking capabilities of different families of today's active botnets. [2] focuses on IRC-based botnets and their detection and [16] focuses on P2P-based botnets and possible countermeasures.

Papers which contain proposed communication protocols are [3], [9], [13], [14], [17]. Modified P2P protocols are proposed in [3], [9], [17]. The SIP protocol elaborated on in [13], [14] is more known as a proposed communication protocol then as an actively used one.

The other articles analyze real botnets which are shortly discussed in this paper. Peacomm in [4], Miner in [5], Feederbot in [6], BlackEnergy in [11] and Zeus is discussed in [12].

References

[1] S. Silva *et al.*, "Botnets: A Survey," *Computer Networks 57*, pp. 378–403, 2012.

[2] E. Cooke, F. Jahamian, and D. McPherson, "The Zombie Roundup: Understanding, Detecting, and Disrupting Botnets," *Steps to Reducing Unwanted Traffic on the Internet Workshop (SRUTI)*, 2005.

[3] P. Wang, S. Sparks, and C. Zou, "An Advanced Hybrid Peer-to-Peer Botnet," *Transactions on Dependable and Secure Computing*, no. 2, pp. 113–127, 2010.

[4] J. Grizzard and othersl, "Peer-to-Peer Botnets: Overview and Case Study," *Proceeding HotBots'07*, 2007.

[5] D. Plohmann and E. Gerhards-Padilla, "Case Study of the Miner Botnet," *4th International Conference on Cyber Conflict*, 2012.

[6] C. Dietrick, H. Bos *et al.*, "On Botnets that use DNS for Command and Control," *Seventh European Conference on Computer network Defense*, 2011.

[7] Soltani *et al.*, "A Survey on Real World Botnets and Detection Mechanisms," *International Journal of Information & Network Security*, no. 2, pp. 116–127, 2014.

[8] P. Maymounkov and D. Mazieres, "A peer-to-peer information system based on the XOR metrix," *International Workshop on Peer-to-Peer Systems*, 2002.

[9] R. Vogt, J. Aycock, and M. Jacobson, "Army of Botnets," *Proceedings of the 14th Network and Distributed Systems Security Symposium*, 2007.

[10] "HTTP-Botnets: The Dark Side of an Standard Protocol!" http://securityaffairs.co/wordpress/13747/cyber-crime/http-botnets-the-dark-side-of-an-standard-protocol.html, accessed:2015-03-27.

[11] J. Nazario, "BlackEnergy DDoS Bot Analysis," *Arbor*, 2007.

[12] D. Andriesse and H. Bos, "An Analysis of the Zeus Peer-to-Peer Protocol," *Technical Report IR-CS-74*, 2014.

[13] A. Berger and M. Hefeeda, "Exploiting SIP for Botnet Communication," *5th Workshop on Secure Network Protocols*, pp. 31–36, 2009.

[14] Z. Tsiatsikas *et al.*, "Hidden in Plain Sight. SDP-Based Covert Channel for Botnet Communication," *TrustBus 2015, LNCS 9264*, pp. 48–59, 2015.

[15] W. Chang *et al.*, "Measuring Botnets in the Wild: Some new trends," *ASIA CCS'15*, 2015.

[16] P. Wang, L. Wu, B. Aslam, and C. Zou, "A Systematic Study on Peer-to-Peer Botnets," *Computer Communications and Networks*, pp. 1–8, 2009.

[17] G. Starnberger, C. Kruegel, and E. Kirda, "Overbot - A botnet protocol based on Kademlia," *SecureComm*, 2008.