Botnets — Secret Puppetry with Computers

Balaji Prasad T.K (bpt@email.arizona.edu)
Nupur Maheshwari (nupurm@email.arizona.edu)
Department of Computer Science,
University of Arizona

Abstract

Botnets have traditionally been and are still considered one of the top threats to Internet security. The scope of botnets transcends the boundaries of Internet security, leveraging a variety of technologies and strategies. They are capable of launching attacks at a massive scale that are difficult to defend. In this paper we describe anatomy of botnets and malware that recruits new bots, variants of botnets and strategies of defense against botnets.

1 Introduction

Malware is a computer program that compramises the security of a computing system. These malware include *viruses* that replicates and mutates itself to avoid detection in host, *worms* that spreads itself across the network, *Trojan Horses* that pretend to serve a useful purpose but rather undermining the system, or a *Rootkit* that enjoys unauthorized root access in the system to evade detection and continue its malicious activities. A botnet is a network of compromised hosts called"Zombies" which have malware installed in them - called the "bot". The bot exploits vulnerability in the users system and opens a back door, which enables the attacker (BotMaster or BotHerder) to remotely control the zombie [2]. The compromised host listens to commands from the remote master and can report confidential details to the botmaster. Over a period the botmaster and the zombies actively scan for newer victims or set traps so that the zombie army grows in size and strength.
Botnets are the single biggest menace to Internet security. A recent report suggest that 83% of global spam in 2011 was sent by botnets, marking a 6% increase from the previous year. Reports claim that close to 80% of all email messages are spam. Recent statistics suggest that the botnet market is growing and consolidating. 2010 reported a 600% growth in the number of victims and 2011 maintains the trend but only 3 of the top ten botnets reported in 2010 are now active, indicating that newer botnets keep emerging as threat. With the increase in the number of smart phone users, the number of mobile botnets is also on the rise.

2 The Botnet Threat Landscape

Botnets threats are multifaceted and can launch a denial of service attack and several other attacks on a critical government website or other crucial systems and are posing a threat to the security of a nation. The hackers managed to temporarily bring down sites such as cia.gov (the US Central Intelligence Agency) and www.soca.gov.uk (the British Serious Organized Crime Agency.)

2.1 What Can a “Zombie-Army” Do?

Botnet creation begins with the spreading of the malware called “Bot” along with an tembedded payload, The portion of malware that actually exploits a vulnerability, which spreads by email attachments or clicking links that cause drive-by download of the malware. Once the malware manages to penetrate the victim’s machine, it creates a backdoor. Now the infected machine further recruits new machines, building the “Bot-army” and when the size of the army is sufficiently large, it can used for a variety of attacks described below.

Distributed denial of service attack : With thousands of zombies under control, the botmaster can launch a denial of service attack on any high profile web site, governmental service, DNS servers thus making them unavailable for legitimate users. Attack usually floods them with fake requests, usually TCP SYN packets, or brute forcing passwords.

Spyware and Malware : The zombies can monitor the activities of the user and constantly report it to the master. It can sniff into the data of the user, log keystrokes, survey vulnerabilities of the port it to a third party.

Identity theft : Botnets are capable of stealing personal information relating to a person’s identity, credit card details, SSN, passwords, spending patterns etc.

Adware : Zombies may automatically download, install, and display ad pop-ups and make the user’s browser to periodically visit certain websites.

E-mail spam : The master can organize zombie computers to send large scale emails to thousands of users. More than 80% of the spam mails today are sent by zombies.

Click fraud : A malware installed in a zombie can have the capability of imitating a legitimate browser increasing the click counts on pay-per-click advertising networks, thus generating illegitimate revenue [3].

Phishing : The zombies can identify servers with vulnerabilities that can be exploited to host fake websites that resemble legitimate ones so as to trick the users into giving out his personal information.

Cyber Extortion : The botnets have been reported to extort money from famous companies by launching a DDOS attack and demanding money to withdraw the attack. Companies are left with little means of saving their image and stopping the launched attack.

Anonymous Internet connections : The attacker can use one of his zombies for nefarious attacks on other systems and uses it as a means of concealing his identity.

Illegal file transfers : botnets can be used as a site for hosting illegal material for monetary gain. The host’s identity is again concealed as he hides behind the zombie army that he had built.

Brute forcing : botnets can be used to send series of request and brute force passwords to gain illegitimate access to bank accounts. This can be easily defended these days by restricting the number of password attempts.

2.2 Some Famous Botnets

Some botnets took the world by storm. A few notorious ones are:

- *The Storm bot* is a P2P botnet which peaked during 2007. Number of victims was estimated between 50,000 and 10 million. Storm bot was famous for its "fighting-back" capabilities—it was designed to launch a DDoS attack on any system that tried to scan and delete the malware that created Storm bot. Malware behind this bot is called *"Storm-Worm"* which exploits vulnerabilities in Windows. It began infecting computers using an e-mail message with a subject line about a recent weather disaster, "230 dead as storm batters Europe."
- *Conflicker* is another popular p2pbotnet built with the malware also called *Conflicker*. The attacker sends a carefully crafted RPC request, which causes buffer overflow in the target machine. The target machine is forced to execute a shellcode, which in turn communicates with the attackers machine to download a Win32 DLL. The DLL attaches itself with svchost.exe and hence starts every time the computer is rebooted. It managed to penetrate the French Navy, United Kingdom Ministry of Defense, Manchester City council's system and police network, and the German army systems.
- *Zbots* is a notorious botnet that is still active and it is built with a Trojan called Zeus. Zeus mainly spreads through drive-by-downloads and Phishing scams where the unsuspecting user is tricked into clicking a link. A 2009 finding suggests that Zeus had compromised over 74,000 FTP accounts on websites of companies such as the Bank of America, NASA, Monster.com, ABC, Oracle, Play.com, Cisco, Amazon, and Business Week. Upon execution the Trojan automatically gathers any Internet Explorer, FTP, or POP3 passwords that are contained within Protected Storage (PStore).

3 Technical Overview and Botnet Architecture

The four main components to building and sustaining a botnet is to create it, propagate the malware to recruit newer victims, obfuscate itself from detection and take-over, and finally establish a means of controlling the victims.

3.1 Building a Botnet

The botmaster exploits the vulnerabilities in a victim's machine to open a backdoor through which he can take control over the machine. A simple Google search can yield a number of botnet source codes and sophisticated tools which are used for various criminal activities based on owning a botnet. Operating system and browser vulnerabilities are also available for sale. The ones that are new and still unpatched by the OS vendor (also called zero-day exploits) are the most wanted by attackers. This attack procedure can be described in a five step process [4]:

1. *Probe* - As a first step the master probes a range of IP addresses and network services and waits for a response. Once it receives a response from one of the machines, a victim has been identified for exploitation
2. *Exploit* - Now the master can attempt a well known exploit in the vulnerable machine. For instance it can try a buffer overflow attack on port 135 (Windows RPC). Now a back door channel has been established to breach the victim's machine (In this case, through Windows RPC port 135)
3. *Force Binary Download* - Now the attacker forces the victim to execute script or *shellcode* that downloads the entire binaries from the master site and attaches itself to a well known service at the victim's machine (Like the svchost, which starts up every time the machine is rebooted).
4. *Communicate With Dictator* - The infected victim now establishes itself as a zombie by establishing connection to a botnet C&C server established over a variety of channels like IRC or HTTP (explained in more detail below.)

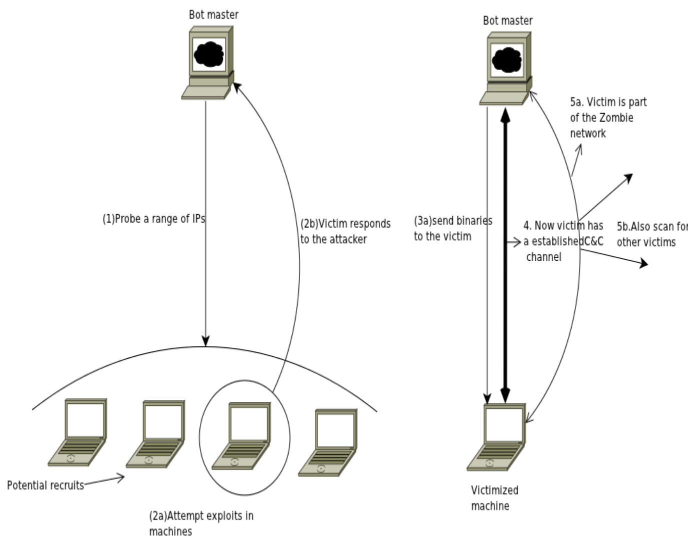
Bot master
(1)Probe a range of IPs
(2b)Victim responds to the attacker
Potential recruits
(2a)Attempt exploits in machines
Bot master
5a. Victim is part of the Zombie network
(3a)send binaries to the victim
4. Now victim has a establishedC&C channel
5b.Also scan for other victims
Victimized machine

Figure 1: How a Victim is Recruited

5. *Establish and Scan* - As a final step in consolidating itself to the bot network, the victim finds a way to establish a dedicated port to communicate with C&C server and receive regular updates and modules to add to its malicious capabilities and then starts its outbound scan.These sequence of steps are shown in *Figure 1*.

3.2 Propagation and Obfuscation

The master and zombies have to actively scan for victims. One common way of spreading is to send spam emails with intriguing links which lead to websites where the Trojans are embedded. The attackers generally trick people into installing malware by attaching them to a crack tool for games etc. Most common way to is to exploit vulnerabilities in operating systems, instant messaging systems, browsers, and email clients. The members of a botnet generally use encryption and obfuscation while propagating or while communicating with the botmaster. The malware generally consist of an encryption routine and a key that encrypts the Trojan to defend detection through signatures. It also comes with an associated mutation engine that mutates the virus to produce different versions of same virus. The bots also use encoding techniques for maintaining their resources and also to communicate with the master. Storm worm, that we discussed earlier, maintains a list of infected P2P hosts that are encoded in hexadecimal. Other bots generally use various means of obfuscation in their communication - use of base64 encoding in GET requests is a common example.

3.3 Command and Control

Once the victim is exploited to be a part of the botnet, a command and control channel is established between the zombie and the master. The C&C architecture generally falls in the following categories;

IRC-Based C&C: One of the obvious means of communication is to directly establish a connection with the host, but that would compromise the identity of the botmaster. So a public message drop point can be used but it is involves a lot of latency. IRC channels turn out be the obvious choice which enables instant exchange point that delivers messages instantly [4].IRC's protocol is widely deployed in the Internet, has a text based command syntax and there are a large number of publicly available IRC channels. The IRC network consist of a number of IRC servers. Master and each of the bots connect to one of the IRC servers

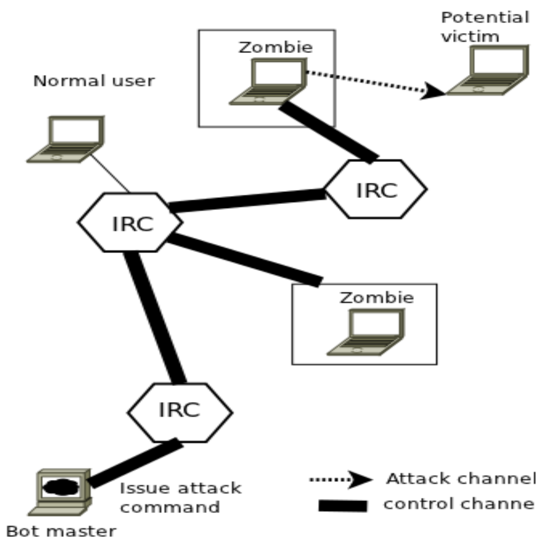

Figure 2: IRC C&C

as shown in *Figure 2*.

P2P C&C: IRC-based botnets described above have centralized master which is single point of failure for the entire botnet. Recent botnets have moved to P2P based architecture for C&C in which the botmaster can use any of the nodes to pass commands or collect information from other nodes in the botnet [9].This makes a sturdy architecture as even if a large portion of the botnet has been taken down, still the remaining portion of the network survives and manages to remain in communication with the master, but they are more difficult to modify and manage than the IRC based ones.

Web-based C&C: Botnets have now evolved to use HTTP and HTTPS protocols for C&C. The bots can now talk to a web server acting as their master. Using Web based protocols gives distinct advantage to the adversary as HTTP ports are always enabled and this C&C merges well with the normal traffic to provide obscurity. To avoid a single point of failure (which is the web server here) the adversary usually uses more than one server to act as master or use random domains which are updated from time to time.

4 A Closer Look at The Anatomy of Well Known Bots

In this section we describe the anatomy and mechanisms employed by well known malwares to open a back door for C&C, mode of control over the bots and also to obscure their presence to escape detection [6].

4.1 Architecture

- *AgoBot*: AgoBot also known as Phatbot is one of the oldest known bots. It has close to 20,000 lines of code in C/C++. It is an IRC based bot with a huge arsenal of exploits with the ability to launch DDoS attacks and harvest passwords through key logging and traffic sniffing.
- *SDBot*: SDBot is also an old bot known since 2002.Today this bot has hundreds of variants providing a wide range of capabilities but the core code is very compact when compared to AgoBot with just 2000 lines of C code. The design is in such a way that extension of code to add a newer capability is very straightforward and also diffuses accountability of the creator.

4.2 Botnet Control Mechanism

- *AgoBot*: AgoBot uses IRC for C&C. It uses set of IRC commands as wells as specific commands developed for this bot. The bot command set includes commands that make the bot to perform specific tasks like some of the example task commands shown in the following table:

Command	**Function**
bot.execute	Makes the bot execute a specific .exe
bot.sysinfo	Echo the bots system information
bot.status	Echo bot status information
bot.nick	Changes the nickname of the bot
bot.open	Opens a specified file
bot.remove	Removes the bot from the host

  In addition to commands, AgoBot also uses a lot of control variables to decide on the capabilities of the bot. For instance DDoS_max_threads variable, if set, will direct the bot to flood the specified host with SYN packets.

- *SDBot*: SDBot uses a relatively straight forward set of IRC commands to exercise control. The steps that a bot follows to contact a server are *(i)* Send the nick name and user name to server *(ii)* Respond to a PING command from master with a PONG and expect for a return code from the master *(iii)* Send a JOIN request to the host and when the connection is established, expect for commands from the server.

4.3 Host Control Mechanism

- *AgoBot:* The set of host control mechanism possessed by AgoBot is quite comprehensive. It can be widely categorized into *(i)* Secure the system, i.e. take countermeasures to prevent other attacks like patching the vulnerabilities.*(ii)* A set of *harvest* commands to harvest sensitive information *(iii)* A list of *pctrl* commands that lists and kills processes running on victim machine *(iv)* A list of *inst* commands to add or delete auto start entries

Type of command	**Functions**
harvest	capable of returning a list of CD keys, emails, AOL specific information, windows registry information etc.
pctrl	capable of returning set of all processes, kill a process, info about running services, stop specific services etc.
inst	capable of adding and deleting an auto start entry

- *SDBot:* The host control capabilities provided SDBot is somewhat limited to basic remote execution of commands and gathering some information from the victim site. Some of the commands include *download*- download a file from specified URL and execute it, *killthread*- kill a specified thread, *sysinfo*- list host system information, *execute*-run a specified program, *Update*- If the version of bot is different, download and upgrade the bot to the latest version.

4.4 Attack Mechanism

- *AgoBot:* AgoBot contains an elaborate set of exploits and attacks and they are well maintained and updated with every variant of AgoBot. It includes a number of scanners for worm backdoors. It also includes brute forcing capabilities for opening NetBIOS shares and brute forcing the passwords of open SQL servers. It also comes with capabilities of performing DDoS attacks on specified host.

- *SDBot :* SDBot's core capabilities are relatively benign (to give its creator a chance to disown any responsibilities) but it was easy to add newer modules to increase its capabilities. SDBot includes modules for sending UDP and ICMP packets to specified host which can be controlled using a simple command of the following type:
  udp/ping <host to attack> <port no. of packets> <packet size>

5 Detection and Prevention

Detection and prevention of Botnets is a challenging task. Let's walk through some of the techniques.

5.1 Anomaly Based Detection

The Botnet C&C master generally performs active scanning of the network to evaluate vulnerabilities and recruit potential victims. This scanning involves sending TCP SYN and other control packets to find open ports. This fact can used to identify anomalous behavior in network by forming what is called a *TCP work weight*.

$$w = (\mathrm{SYN_n} + \mathrm{ACK_n} + \mathrm{FIN_n})/\mathrm{TCP_n}$$

Where,
w = TCP work weight, $\mathrm{SYN_n}$ = number of SYN packets, $\mathrm{ACK_n}$ = number of ACK packets, $\mathrm{FIN_n}$ = number of FIN packets, $\mathrm{TCP_n}$ = Total number of TCP packets in the sample period.

A large value of TCP work weight means the packets are dominated by control packets which are typical to Botnet master scanning for victims. If such scan packets can be traced to a IRC host, most probably a Botmaster has been identified. But recent Botnets use what is called **"Idle scanning"**. Imagine a master M wants to scan potential victim P, he spoofs his IP address to B - a bot (who will eventually take the blame of scanning) and sends a SYN request to P, the potential. P thinks B has sent a SYN request to it and either responds with a SYN+ACK or RST depending on whether the port was open or not repectively. Now if B gets a SYN+ACK, it'll respond with a RST as it had no idea about the connection. Now the master can easily find the IPID of the reset packet of the bot B(Note that IPID is incremented with every packet sent out the system) which will show what P responded to B and hence whether P's port was open. Now the detection becomes tricky.
In such situations, instead of tracking the host with anomalous packet behavior, we can form a *Host Exposure Map* which captures the host-port combinations of the connections in which the host generally involves. This data should be obtained by initially training the system and capturing the pattern. Now any activity on the host which doesn't fall in the Exposure Map can be reported and attacker tracked down.

5.2 Detection by Dialogue Co-relation

The victimized host goes into specific states during attack and initial establishment of C&C channel. So a correlation engine which models these specific states can be used in the network to detect malicious communications [4]. To be more specific, the packet scanner will look for initial inbound scanning, using an exploit, binary download, coordination with the master and outbound attack propagation. The model is described in *Figure 3*

The dialogue co-relation engine sits at the perimeter of the network and make use of the services of *Intrusion Detection Systems (IDS)* to analyse the contents of payload against malicious signatures and also to determine inbound and outbound scan patterns. It keeps track of the various stages in the infection trail, and reports an alert when the interaction model fits into malicious pattern.

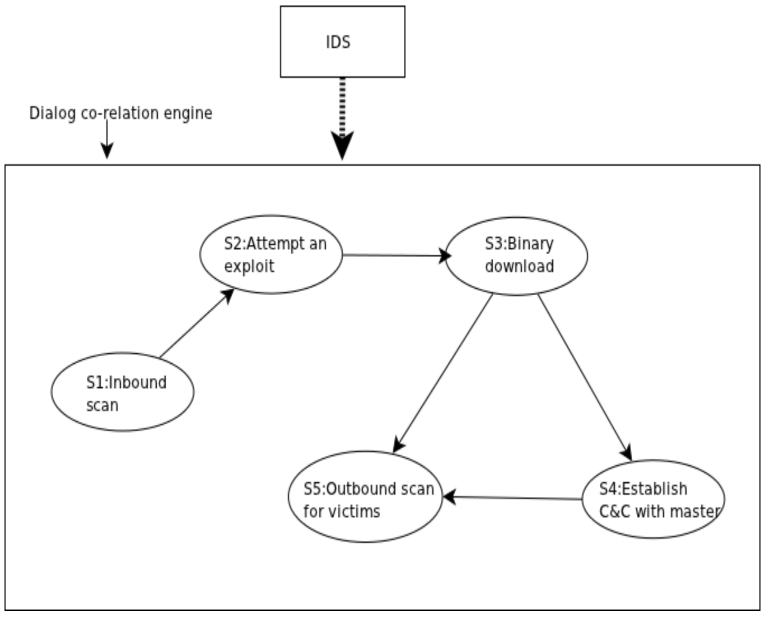
IDS
Dialog co-relation engine
S1:Inbound scan
S2:Attempt an exploit
S3:Binary download
S4:Establish C&C with master
S5:Outbound scan for victims

Figure 3: Dialog co-relation for Botnet detection

5.3 P2P Botnet Detection

We have seen how P2P botnets differ from regular botnets in earlier section. One of the effective ways of P2P botnet detection is to monitor the target P2P network and form statistical fingerprints which profiles the different P2P applications including the legitimate ones (like Skype) and cluster based analysis which profiles the hosts. This can be devised as a two step process-

1. First process involves detection of hosts in the network that involve in P2P communication. This can be done by filtering the packets of each host and extracting a number of statistical values which will isolate the P2P hosts.

2. Now the second phase involves separation of legitimate P2P hosts from the malicious ones [11].

This can be done by the following parameters:

- *persistence pattern :* There is usually a difference in the amount of time the legitimate P2P user and a botnet host remain active in the P2P network. For instance a file sharing application may be closed as soon as the job is done, but a bot remains active as long as the system is up.

- *interaction pattern :* The botnet peers also follow a pattern in interaction including using the same protocol, and the large overlap in the number of peers contacted by two different bots.

The P2P detection system is intuitively described in *Figure 4*.

6 Evolution of Botnets

The botnet architecture has evolved over time with the attacker's modus operandi being well known to the security community and also with the sophistication of tools and computing resources available to the attacker community. Initially the attacker relied preliminarily on security through obscurity as the existence

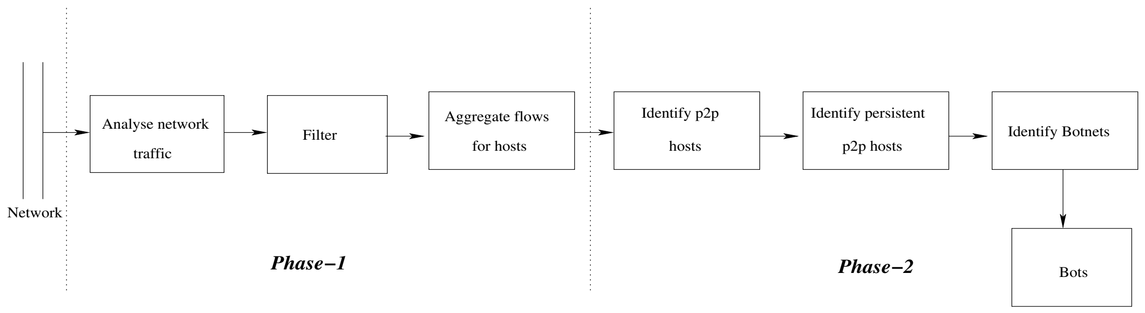

Figure 4: P2P botnet detection

of botnets was not widely known. Attackers used *IRC channels* to command and control the zombies. This was very insecure as it was possible to bust them easily and also for other adversaries from the attacker community to take control of their botnets [1]. Despite its weaknesses, the classic architecture has its strength in ease of setup and maintenance, widely and freely available infrastructure for botnet setup, support from the expert hacker community, and the ability to fully control and track his bot army.

6.1 From Chaotic to Professional

Today's botnets have moved from public IRC networks to *privately hosted servers*. To prevent adversaries from take-over [8], the bots are designed to ignore any commands that do not contain a secret pass-phrase as the prefix of the commands. Now the botmaster can control the victims with *HTTP* rather than IRC. The victim now loses the option of shutting down the IRC port to prevent contact with command and control center as shutting the HTTP port down is practically infeasible. They also have fancy GUIs with mining and querying capabilities that the botmaster can choose to send spam mails from zombies in a particular country. It also has ability to send pre-formatted SQL queries, sending data and receiving commands over HTTP. Most of them are so sophisticated and work at such low levels that they could record the HTTP payloads before they are actually encrypted.

A notorious hack kit called *Mpack* provides a classic example of second generation botnets [1] [5]. *The Mpack kit* has an arsenal of scripts that could detect the recruit's OS, browser information, location, other software installed in the system and from the library of exploits available with the kit, could then decide the best means to bring down the recruit. Thus the newer generation bots don't actively scan for recruits but instead they lure recruits and turn them into honeypots for further multiplication. Also with the explosive growth of smart phone users, mobiles and smart phones are turning out to be one of the major platforms for botnets. Such devices are almost always online and have more and more functionalities and processing power built in them. Such devices are connected to computers to synchronize contents and this could be exploited to spread and update malware [10] [7]. The Figure 5 depicts the evolution in the architecture of the modern botnets.

7 Conclusion

Having evolved with technology and time, the botnets still remain the greatest threat to the security of networked computing resources. The botnet kits of today use every conceivable form of attack with a score of malwares, worms, Trojans, root kits, spam engines that analyze the vulnerabilities of computers before

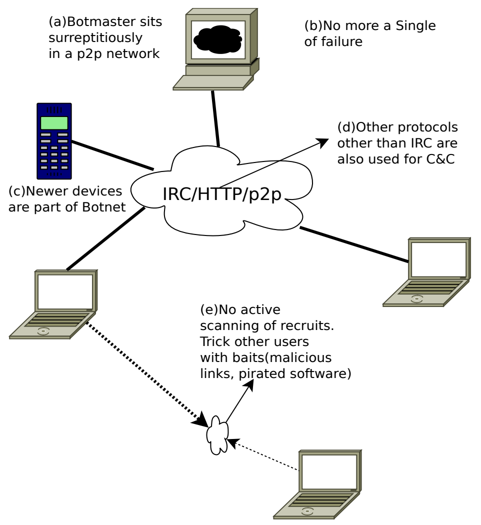
(a)Botmaster sits surreptitiously in a p2p network
(b)No more a Single of failure
(d)Other protocols other than IRC are also used for C&C
(c)Newer devices are part of Botnet
IRC/HTTP/p2p
(e)No active scanning of recruits. Trick other users with baits(malicious links, pirated software)

Figure 5: Botnet evolution

attacking them. The fight against botnets starts from a very rudimentary level at the user's general awareness of threats to his system posed by Malware and botnets. Majority of the attacks are due to ignorance and indifference to the actual threat. The following can be considered as general guideline for basic safety against malware:

- Stay aware and updated about the threats and spread the *awareness*
- Install *security* updates for OS, browser etc. promptly
- Don't visit untrusted links or visit unknown websites and get infected.
- Avoid using peer-to-peer software as much as possible.
- *Block JavaScript*-Change your settings to enable browser prompt before executing any script.
- *Watch your ports* for unexpected inbound and outbound traffic. Disable all unused, unnecessary ports.

With these general personal awareness and discipline the spread of botnets can be controlled to a large extent. A sufficiently large botnet can pose serious threat to the security of an entire nation. In the era where computers run everything from businesses to governments, it's time to take computer security against botnets seriously and carry out sufficient research in this area and not letting the attacker dictate terms.

References

[1] Zheng Bu, Pedro Bueno, and Rahul Kashyap. The new era of botnets. *White paper from McAfee*.

[2] Evan Cooke, Farnam Jahanian, and Danny McPherson. The zombie roundup:understanding, detecting, and disrupting botnets. *SRUTI 05, USENIX association*, 2005.

[3] N. Daswani, M. Stoppelman, the Google Click Quality, and Security Teams. The anatomy of clickbot. *USENIX Hotbots07*, 2007.

[4] Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. Bothunter: Detecting malware infection through ids-driven dialog correlation. *In Proceedings of the 16th USENIX Security Symposium (Security'07)*, Aug 2007.

[5] Corey Nachreiner and Scott Pinzon. Understanding and blocking the new botnets. *white paper from LiveSecurity*, April 2008.

[6] P.Barford and V.Yegneswaran. An inside look at botnets. *Special Workshop on Malware Detection,Advances in Information Security, Springer Verlag*, 2006.

[7] Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating bluetooth as a medium for botnet command and control. *Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, July 2010.

[8] Brett Stone-Gross, Marco Cova, Lorenzo Cavallaro, Bob Gilbert, Martin Szydlowski, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. Your botnet is my botnet: Analysis of a botnet takeover. *CCS '09 Proceedings of the 16th ACM conference on Computer and communications security*, 2009.

[9] Ping Wang, Sherri Sparks, and Cliff C. Zou. An advanced hybrid peer-to-peer botnet. *In Proceedings of the First Workshop on Hot Topics in Understanding Botnets*, 2007.

[10] Cui Xiang, Fang Binxing, Yin Lihua Liu, Xiaoyi, and Zang Tianning. Andbot: Towards advanced mobile botnets. *In Proceedings of the 4th USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, March 2011.

[11] Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Detecting stealthy p2p botnets using statistical traffic fingerprints. *IEEE/IFIP 41st International Conference on Dependable Systems and Networks*, page June, 2011.