

# Chapter 2

## Botnets Threat Analysis and Detection

Anoop Chowdary Atluri and Vinh Tran

### 2.1 Introduction

A botnet is a collection of Internet computers that have been set up to execute unintended operations. The owners of these machines often are not aware of the status of their devices, which is due to a lack of protection on the computers (e.g., no antivirus or firewall). When a computer without basic protection is used to browse the Internet, the user may click on a number of different links as well as download many types of files. If the files are Trojan/malware, they can automatically create a backdoor to communicate to the command center and hide their processes from the end user.

This chapter gives a walkthrough of the botnet phenomenon by centering the discussion on some famous examples, which are also representative of some of the main bot families available.

The chapter starts with a brief historical review and a discussion of botnets architectures. This is followed by a review of famous botnets examples, a discussion of techniques used by botnet to evade detection, and finally, a review of protection techniques and strategies.

### 2.2 Evolution of Botnets: History and Topologies

Botnet evolution started with Sub7 (a trojan) and Pretty Park (a worm) in 1999; both introduced the concept of a victim machine connecting to an IRC channel to listen for malicious commands (Ferguson 2015a, b). Then it comes to the Global Threat

---

A.C. Atluri • V. Tran (✉)  
New York Institute of Technology,  
701 West Georgia Street, 17th Floor, Vancouver, BC, Canada, V7Y 1K8  
e-mail: [ranlucvinh@gmail.com](mailto:ranlucvinh@gmail.com); [atlurianoop.4@gmail.com](mailto:atlurianoop.4@gmail.com)

Bot (Gtbot) in 2000; this botnet is based on the mIRC client which makes it possible to run custom script depending on the IRC commands. One of the most famous Gtbot attacks is to scan for host infected with Sub7 and update it to Gtbot.

In 2002, two new botnets were introduced, called SDBot and Agobot. SDBot was a single binary file, written in C++. The corresponding code was commercialized, and as a result, many new botnets were born inspired from it. Agobot, on the other hand, was considered a more advanced botnet, which suggested the principle of modular, staged attacks as payloads. Agobot infection comprises of three stages: first stage consists of installing a backdoor, then trying to disable the host antivirus, and lastly blocking access to websites of known security vendors.

In 2003, Spybot was created, as a transformation of SDBot. This new botnet introduced some new functionality such as keylogging, data mining, and SPIM (instant messaging spam). Rbot was also surfaced in the same year. This bot introduced the SOCKS proxy and included DDOS feature and information stealing tools. Moreover, the bot was also the first one to use compression and encryption to avoid detection. The year 2004 saw the rise of Bagle and Bobax, the first spam botnets. In 2006, ZeuS or Zbot was introduced and is still now one of the world most famous botnets. The year 2007 saw the birth of Storm, Cutwail, and Srizbi botnets.

The history of botnets closely correlates with the evolution of botnets topologies and architectures. Botnets are implemented using different topologies, including the following four main architectures (Ollman 2009):

- **Star:** This hierarchy (see Fig. 2.1) allows the bot to communicate directly with its master. This approach helps the simplest one; it facilitates bot management and makes sure the communication between both the parties are fast and accurate. However, it suffers from single point of failure and system administrators can easily block the connection to the master.
- **Multi-server:** This topology (see Fig. 2.2) is a more advanced form of the Star architecture. It tackles the problem of single point of failure and also makes sure that the bots can reach its closest geographical master (assuming the C&C servers are set up in multiple countries). Nevertheless, this hierarchy requires more effort to set up and plan from the master.
- **Hierarchical:** This topology (see Fig. 2.3) allows a bot to act as a supervisor for a group of other bots. The supervisor bot can directly connect to the master and update instructions/code base. This approach hides the presence of the master and makes tracing back to the master more difficult. Also, botmaster can easily share/lease/sell a portion of the botnet to other botmaster. Nonetheless, this architecture adds a level of latency to the update between bots, because the lower-level bot needs to wait for instructions sent from the supervisor bot, making real-time attack harder than the previous topology.
- **Random (Peer to Peer):** The last design (see Fig. 2.4) is called random or peer to peer (P2P). This is by far the most advanced topology in botnet. Any bot agent can send/forward commands to the next one; these instructions are often designed

Fig. 2.1 Star formation

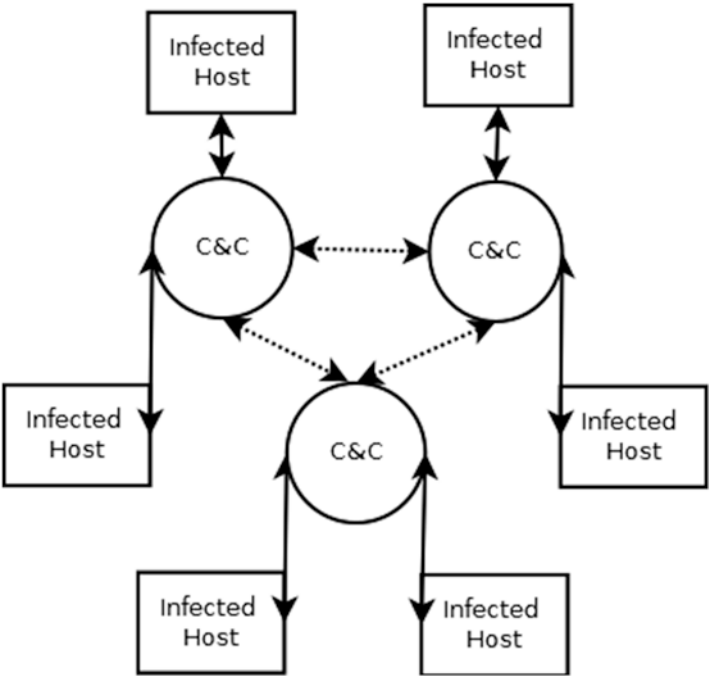
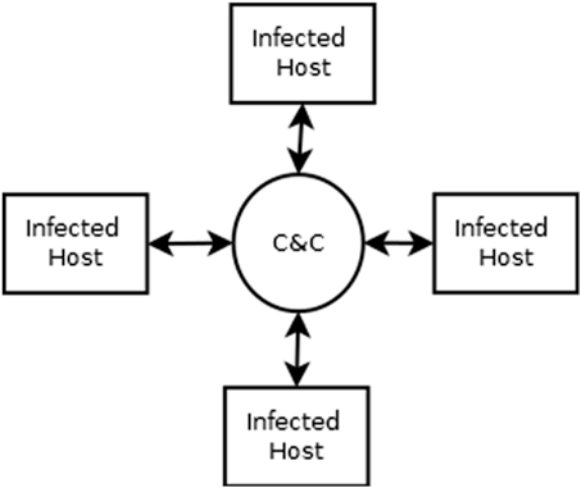
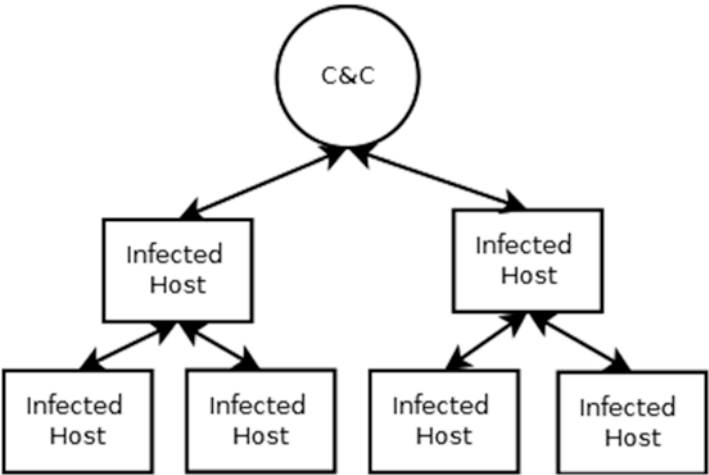
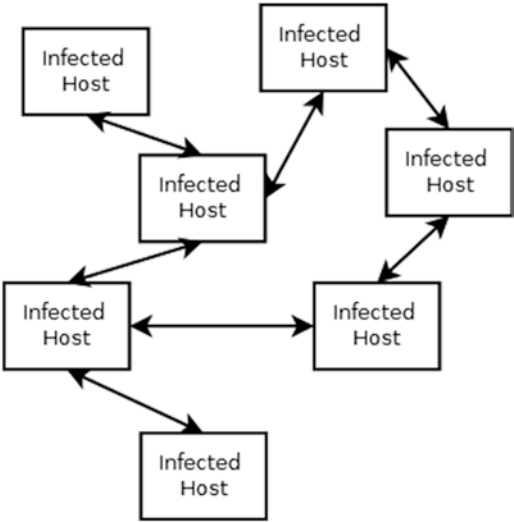


Fig. 2.2 Multi-server formation



**Fig. 2.3** Hierarchical formation

**Fig. 2.4** Peer to peer formation



in a way that it can pass on to the next available node in the net. This method allows the botmaster to avoid detection/shutdown, as it would take a considerable amount of time to trace the communication between bots. However, the design helps researchers to track down the infected hosts easily, since monitoring one bot can reveal information about its communication with others.

## 2.3 Famous Botnets

There are a great number of botnets worldwide; however, most botnets have similar functionality and often are variants of some previous botnet. This chapter only focuses on ZeuS, Koobface, and Windigo as examples of popular botnets. The reason for picking these is that ZeuS is one of the early botnets that still remains famous until now, and it has gone through multiple waves of revolution. As for Koobface, this botnet represents a new form of malware that spreads through online social network, and using user's friend list as a means of propagation. Lastly, Windigo represents one of the few famous botnets targeting primarily Linux platforms.

### 2.3.1 ZeuS or Zbot

*Overview:* ZeuS is a family of credentials-stealing trojans which first surfaced around 2007 (Andriesse and Bos 2014). Since then, ZeuS has grown to be one of the world's most famous botnets. Older versions of ZeuS, which relied on IRC Command Center, have been studied by scientists and security professionals (Falliere and Chien 2009). In 2011, a more advanced version of ZeuS was introduced, called GameoverZeuS. This variant uses P2P with encryption instead of IRC channel. The modern Zeus versions with advanced features such as encryption and communication pattern not only harden detection process but also prevent the network from being infiltrated by "outsiders."

*Encryption:* Early versions of ZeuS use a simple mechanism for encryption, known as "visual encryption," which basically encrypts each byte by XORing with the preceding byte (Andriesse et al. 2013). Later versions introduce RC4 encryption. "Outsider" bots, which are used by researchers and security personnel, to penetrate the network, becomes counterproductive, since the fake bot needs to know under what identifier it is known to other bots in the network in order to decrypt the message.

*Communication pattern:* (Andriesse et al. 2013) Zeus maintains a passive and an active thread. The passive thread acts like a server, listening for incoming request. The sender's information of any successful handled request is stored in a bot's peer list. On one hand, if the receiving bot already has more than 50 peers in its list, the sender bot data will be saved in a queue for future peer list update. However, the sender bot will be automatically added if the peer list is 50 or less. On the other hand, if the sender identifier is already on the list, all information (such as IP and ports) is updated, to keep a fresh connection with its peer.

The active thread runs in a cycle and automatically repeats after a specified amount of time. In each iteration, the bot attempts to connect every peer in its list, asking for updated version of binary and configuration file. Each peer has five chances to reply to the request; if there is no response after five times, the bot will

first check if it actually made the request to the recipient by checking for Internet connection; then depending on the Internet status, it will drop the unresponsive peer and update the list. Moreover, if the bot has less than 25 peers, it will try to connect to all its neighbors asking for the neighbor's peer list. This mechanism assures the botnet network always stays fresh and long-period-disconnected bot can recover quickly even with a minimal number of peers.

### 2.3.2 *Koobface*

*Overview:* Koobface is one of the first malwares to target online social networks (OSN) (Baltazar et al. 2009; Thomas and Nicol 2010; Sophos Press 2007). The botnet first appeared around early 2009 and has caused severe damage to social networks users. The koobface malware, unlike others, has its binary split into multiple modules, each of which has a separate functionality that handles different type of OSN. Additionally, instead of spreading through spam email, the malware uses OSN messaging service to propagate. This is a very effective way to escalate the infection, as people often have no doubt about their friend's messages (Fortinet White Paper 2013). Once clicked on the link in the message, user will be redirected to a fake page, created by social engineering toolkit (usually fake YouTube page). Here, users will be asked to install a fake plugin in order to view the content. The fake plugin is the koobface downloader, which will attempt to find out the OSN the user is using and then download the necessary components accordingly. As of 2009, the malware was able to identify a significant amount of various OSN such as Facebook, Twitter, MySpace, Friendster, Hi5, Netlog, Bebo, and so on.

*Features:* This botnet not only breaks captcha by forcing other infected machine's user to solve it but also creates fake OSN accounts in order to befriend with potential victims. Research has shown that a normal user has 41 % probability to accept a friend request from strangers on Facebook (Irani et al. 2011); this is why KoobFace has become so successful and led the way for a new form of malware that spread through OSN.

### 2.3.3 *Windigo*

*Overview:* The botnet has a long history (Bilodeau et al. 2015), starting from 2011; it comprises of a few different malwares which take care of different tasks. Most of the modules (e.g., Linux/Ebury, Linux/Cdorked, Linux/Onimiki), however, are specialized in compromising linux servers (e.g., web, dns servers). There are also two other malwares (Win32/Boaxxe.G and Win32/Glubteta.M) targeting Windows computers' end users. Like any other modern botnet, Windigo also carries out a number of tasks ranging from sending spams, drive-by downloads, advertisement

fraud, and credentials stealing; however, one important point to notice is the main victims are Linux servers, which mean they have more resources, bandwidth, and also have more potential to reach end users via web servers. The main Linux components are summarized in the following.

*Linux/Ebury* (Bilodeau et al. 2015): main functions are creating backdoor shell and credentials stealing. One of the outstanding attributes of this malware is its ability to run in a very stealthy way, because maintaining an SSH backdoor shell is a difficult task. In order to do this, the creator has applied many different techniques, and some of them are as follows:

- Utilize linux pipes as much as possible
- Leave no information in log files
- Alter OpenSSH binaries code at runtime instead of modifying the current files on disk
- Use a centralized backdoor in a library

*Linux/Cdorked* (Bilodeau et al. 2015): It is used to redirect traffic from infected servers to malicious sites; some of the most common web servers (apache, nginx) have been infected with variants of this malware. In order to deploy this malware, the botnet uses previously installed Linux/Ebury to download a complete source code of the web server; it also gets another patch from an infected server. Then the patch is applied on to the new source code and a new binary is compiled, after that, the original web server binary is replaced by the new malicious binary. When making a redirection, the malware tries to guess if the current user is a system admin by checking a number of url keywords and cookies; this mechanism allows the malware to act under the radar and thus avoid detection.

*Linux/Onimiki*: It is a domain name service component which acts together with Linux/Cdorked. Whenever a redirection is made from a Cdorked infected machine, Onimiki will try to resolve the domain name in the url. It is also noted that Onimiki uses BIND name server and this offers a number of advantages, such as the following:

- It is stateless and requires no configuration when Onimiki is installed, thus allowing the malware to act alone without any further interaction with the operators.
- It allows fast rotation of subdomains and legitimate domains.
- Its reputation of the legitimate domains helps Onimiki avoid blacklisting.

Table 2.1 summarizes the main features of the three botnets examples considered above.

## 2.4 Botnet Detection Evasion Techniques

Botnet uses many different methods to avoid detection; some popular techniques are as follows:

**Table 2.1** Comparison of Zeus, Koobface, and Windigo

|                               | Zeus/Zbot                                                                                                                                                                                                                    | Koobface                                                                                                                                                                                  | Windigo                                                                                                                                                                  |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Infection vectors             | Infection vectors vary widely; some main mechanism are spam, drive-by download                                                                                                                                               | Mainly through online social network                                                                                                                                                      | Spread through linux servers                                                                                                                                             |
| Features                      | A DIY bot that is features-rich, easy to use. Underground criminals can easily purchase a copy of Zeus and build a version of this malware. Maintaining a sophisticated P2P network which makes taking down operation harder | Can detect multiple online social network and has various components that can act differently depending on the detected online social network. Break captcha by forcing users to solve it | Although this botnet mainly targets linux servers, it has the ability to take control of the windows machines which established connection to the infected linux servers |
| Availability and distribution | Freely available with purchasability makes Zeus the most popular botnet                                                                                                                                                      | Not available for purchase                                                                                                                                                                | Not available for purchase                                                                                                                                               |

- Domain generation algorithm (DGA or Domain Flux): According to Khattak et al. (2014), DGA is an approach to dynamically generate the C&C address. The botmaster builds a specific mechanism to randomly create the server address and sets up the DNS record to point the address to the C&C. An example using this technique is ZeuSGameover malware (Andriess et al. 2013); the algorithm is triggered when all peers are unresponsive or the bot fails to update for more than a week.
- IP flux (Shin and Gu 2010): this technique is similar to DGA, but instead of associating multiple domains with one IP, it attempts to alter DNS records to have various IP addresses linked to one domain. The method is aided with the help of Dynamic DNS. IP flux has two different types:
  - Single FLUX: the idea is to have intermediaries between the bot clients and bot master, providing a layer of anonymity for the bot master. These middle layer machines are often called “proxy bots,” which are also infected machines chosen by the master.
  - Double flux: is an advanced version of single flux, which abstracts the domain name and IP address of the proxy bots. When bot agents try to connect to proxy bots, they will be redirected to name servers controlled by the master. These name servers will handle the domain name matching and generating, and make sure that name and IP pairs change frequently so the connection will not be blacklisted or blocked.
- Binary obfuscation (Shin and Gu 2010): the bot client uses various techniques to defeat host-based security application, one of which is polymorphism. It is an attempt to reconstruct the bot into different forms but still maintain the same

**Table 2.2** Botnet detection evasion techniques summary

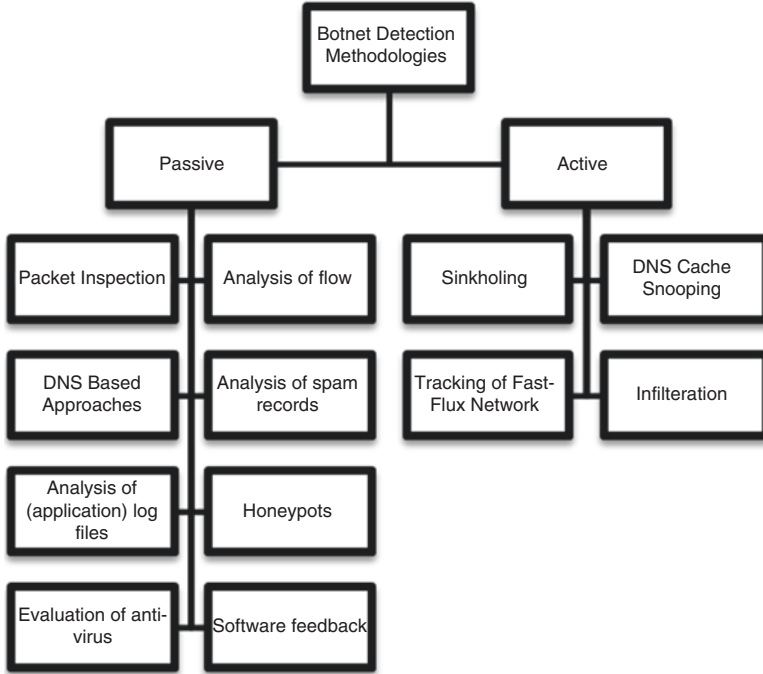
|                             |                                                                                                                                                                                                                                                    |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Domain generation algorithm | A mechanism which allows bot agents to connect with master through a variety of different domain names. The bot master takes care of generating the domain name and sets up the servers; the agents will have a list of names to try connecting to |
| Single flux                 | Bot master chooses some infected machines to become proxy bots or fake master. This technique helps the master to become harder to track down and thus stay alive longer                                                                           |
| Double flux                 | An advanced version of single flux, which takes the connection to another level by adding the complexity of domain name generation                                                                                                                 |
| Binary obfuscation          | To defeat the host-based defense, bot agents can be built into different binary form, but still maintaining the functionality. This technique is carried out often by encryption and packing                                                       |
| Security suppression        | Certain types of botnets have the ability to disable local security service and also block users from finding a security solution                                                                                                                  |
| Anti-analysis               | Some early day botnets have the ability to change its behaviors based on the environment it's running on                                                                                                                                           |

- functionality, by using encryption or packing. Some advanced packers can build a completely different binary for every packed request. Despite success in hiding its identity, the bot binary can still be detected while executing due to memory-based detection approach. To work around this problem, the bot agent uses another practice called metamorphism, which gives the bot the ability to be rebuilt into different, but semantically equivalent code.
- Security suppression: When infecting a weak machine, the malware attempts to disable all or several security services on the host. For example, (Bilodeau et al. 2015) the malware Conficker will attempt to disable some security service in Windows when infected; it also sets up a blacklist which prevents users to access certain security site.
  - Anti-analysis: Certain botnets have the ability to scan the environment which they are running on, and depending on the results, they can disable/change their behaviors to appear harmless or mislead the researchers. This technique was quite popular in the early days of botnet, but after the explosion of virtual technology, this method is being forgotten as criminals also want to target virtual users.

Table 2.2 summarizes the evasion techniques outlined above.

## 2.5 Botnet Detection Methodologies

Botnet Detection techniques can be grouped in various categories. Figure 2.5 depicts those different categories, which include both active and passive techniques (Plohmann et al. 2015; SANS Institute InfoSec Reading Room 2015).



**Fig. 2.5** Botnet detection methodologies tree

### 2.5.1 *Passive Techniques*

Passive measurement techniques are a group of few methodologies where data is gathered through monitoring and observation alone. Using passive measurement techniques, we can track activities without interfering the production network or making changes in any kind of evidence. There will always be a limited amount of data which can be collected from passive methods and this data can be used for analysis (Plohmann et al. [2015](#)).

#### 2.5.1.1 **Packet Inspection**

The most common methodology under passive botnet detection system is Packet Inspection of local network data. The main objective of this technique is to ensure various parameters of packets are matched like protocol field, identification, flags, and content with huge database of predefined abnormal and suspicious behavior which allows identifying bots by analysis of data only.

For instance, there might be a data packet consisting of shell script code which is being used to inject malware in network and that particular malware is communicating

with the public address which can host the data. The predefined patterns are also referred to as detection signatures.

The main characteristic of packet inspection approach is that it can be incorporated in typical Intrusion Detection Systems (IDS) where attacks are identified based on predefined signature database. The inspection service runs in two deployment modes, which include as a single appliance for entire network that is known as network-based detection system (NIDS), and Host-Based Network Detection service (HIPS) where every node of the network runs a separate instance of the detector.

Intrusion detection system is primarily used to just detect the malicious activities and they are reported to network administrator, and network administrators are responsible to take action on infected areas.

Some commonly identified drawbacks of intrusion detection or prevention techniques include the fact that when the network traffic flow is very high it is difficult to perform complete inspection of the packets. If we make use of techniques like packet sampling or packet filtering prior to analysis, chances of missing malicious packets are too high.

Furthermore, intrusion detection systems are known for their high false alarm rates, which is a serious limiting factor.

### **2.5.1.2 Analysis of Flow Records**

Analysis of flow records can be considered as a technique for tracing network traffic at a nonrepresentational level. In the packet inspection approach, the packet is described to some level of details; each and every packet should be inspected in an aggregated form. In the flow record approach, when a data stream is considered for analysis it goes under a process where several parameters are matched. These parameters include addresses of the source and destination, port numbers and the protocol which is used in the packets, how many packets are transmitted, and size and duration of the session.

Net flow can be considered as one of prominent examples for the analysis of flow record format. Like with packet inspection, the main aim of flow record analysis is to differentiate and identify the traffic patterns by creating a scheme to detect malicious traffic.

### **2.5.1.3 DNS-Based Approaches**

A connection should be initiated and established with infected hosts or commanding server by considering botnet infrastructure whenever hosts have been infected by a botnet. This can usually be achieved by integrating a communication protocol with the malware. This can be done in two ways as follows.

An IP address can be integrated into the bot, which will be executable upon distribution, but the IP address should be fixed. A predefined domain name should be used, which will be contacted if the host system is compromised. To avoid downtime

by providing redundancy, multiple IP addresses can be associated or mapped with a single domain name. On demand, these IP addresses can be changed to dynamic whereby they are not configured for static use (Plohmann et al. 2015).

#### **2.5.1.4 Analysis of Spam Records**

Spam emails are irrelevant messages sent to a large number of users. Spam represents a common drive of botnets. The analysis of spam records provides a method of identifying and anticipating botnet infection attempts. Unlike DNS-based approaches, which target primarily the C&C phase, spam analysis aims at detecting botnets at the infection phase, and this technique will eventually detect botnets that essentially do spamming. Spam analysis involves identifying regular emails communications and distinguishing illegitimate message contents.

Distinguished patterns of spam mails are produced by the bot eventually forming the foundation or base for botnet detection. The content of message offers a good initiation point for matching and characterization of messages related to the email protocol header and content.

The correct placement of spam traps will be helpful summation to this schema. Usually spam traps are mailing addresses with no prolific functionality other than to accept unrecognized and unwanted mails and can be distinguished as a distinct variety of honey tokens or honeypots.

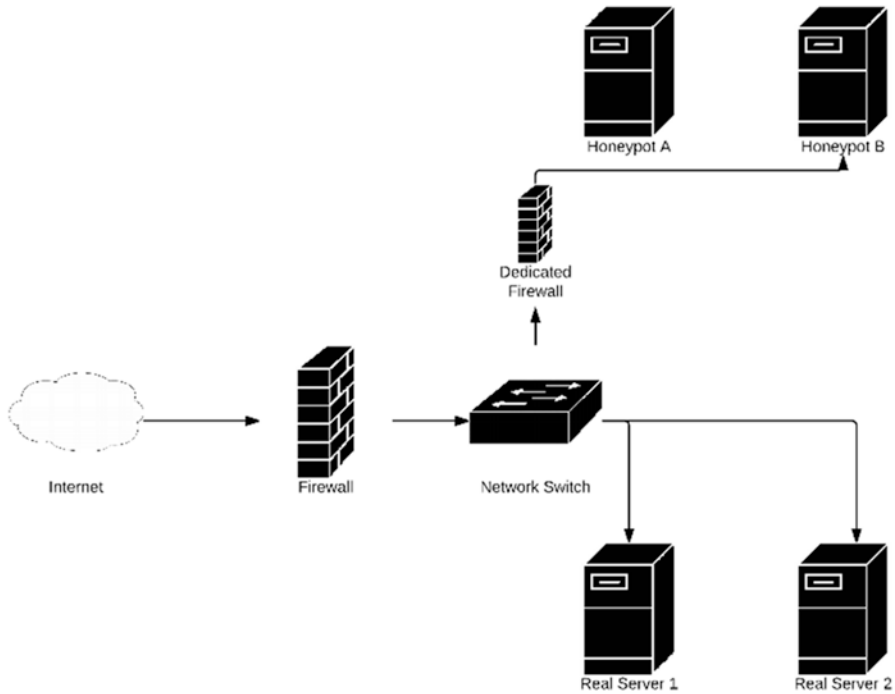
#### **2.5.1.5 Analysis of (Application) Log Files**

It is common practice for devices and applications to maintain records of events related to different operational aspects in the form of log files.

Log files analysis is a secondary approach of botnet detection system. The basic analysis is done using network devices log files, which come as a basic match method from entire network devices; this analysis can be done in parallel over entire range of network devices.

#### **2.5.1.6 Honeypots**

Figure 2.6 depicts a network architecture with two honeypots (A and B). A honeypot is a dedicated machine with a purpose of exposure to outside (i.e., Internet) with a focused goal to attract attackers and learn attacking methods or even to get compromised by malicious activities. In this setup, the network is always secured in the backend and only honeypots are kept visible by exposing them to the outer world. Honeypots help administrator to understand the attackers' techniques against the network and develop and deploy adequate security policies and mechanisms for protection.



**Fig. 2.6** Honeypot network

There are different types of honeypots including the following:

- *Client and server honeypots*
- *Low interaction honeypots*

The main motive for using honeypots in botnet analysis is the opportunity to collect different data about the practices and strategies used by inventors of malware and hackers. In general, two types of data can be collected by honeypots:

- Types of attack vectors in OS and software used for attacks, as well as the real exploit code which links to them.
- Actions done on an exploited workstation. These can be noted, while malware loaded on to the workstation can be conserved for further analysis.

### 2.5.1.7 Evaluation of Antivirus

This approach simply consists of relying on existing antivirus software capability. Different antivirus products have different signature databases, with some overlapping signature set. New generation of antivirus systems not only pushes updates

regularly to their clients, but they also learn from new instances of viruses occurring at specific endpoints, by pulling information from the clients. So it is a two-way communication stream.

#### **2.5.1.8 Software Feedback**

Software installed in the user work stations and data flow in network are analyzed and automated feedbacks of software reported to vendors. In this scenario of network each host machine acts as a sensor and the entire network is converted into a big sensor network (Plohmann et al. 2015).

### **2.5.2 Active Techniques**

The group of active methods contains methods that involve communication with the information sources being observed. While these allow deeper probing and analysis, their application may leave traces that impact consequences or include events that can be observed by the concerned. This can cause counter-reactions, such as a DDoS attack or trigger other attempts at evading detection.

#### **2.5.2.1 Sinkholing**

This is a process of mitigating botnets by cutting off the source and breaking of communication between bots and C&C server.

As shown in Figs. 2.7 and 2.8, sinkholing consists of redirecting requests from the bot to the sinkhole (typically a server under control of the good guy) rather than letting such communications go through to the C&C server.

If one or more domains with fixed IP addresses are used by the malware, then discovering and blacklisting them will quarantine the specific malware examples that rely on them, making those useless. By using the direct IP addresses, there is no need of the DNS queries and the botnet can be terminated by deregistering the domain name (Plohmann et al. 2015).

This approach could help discover more malicious activities beyond the initial detection. For example, if a domain is identified as malicious, it is known that all incoming queries for this entry are given out by infected hosts with high probability.

#### **2.5.2.2 DNS Cache Snooping**

As shown in Fig. 2.9, DNS Cache Snooping approach leverages the caching property implemented and used by several DNS servers. If a DNS server is asked for a domain for which it has no entry defined, it will issue a query towards the responsible authoritative name server on behalf of the querying client and store the resultant data record later in a local cache. Caching is mainly used to increase the performance of a name server and reduce its traffic load.

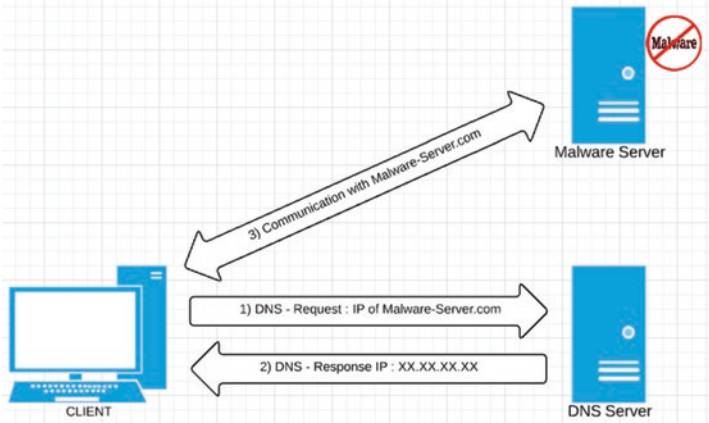


Fig. 2.7 Sinkhole attack

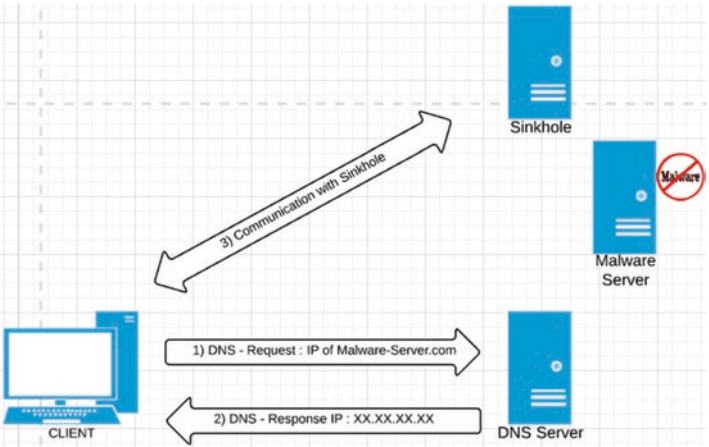
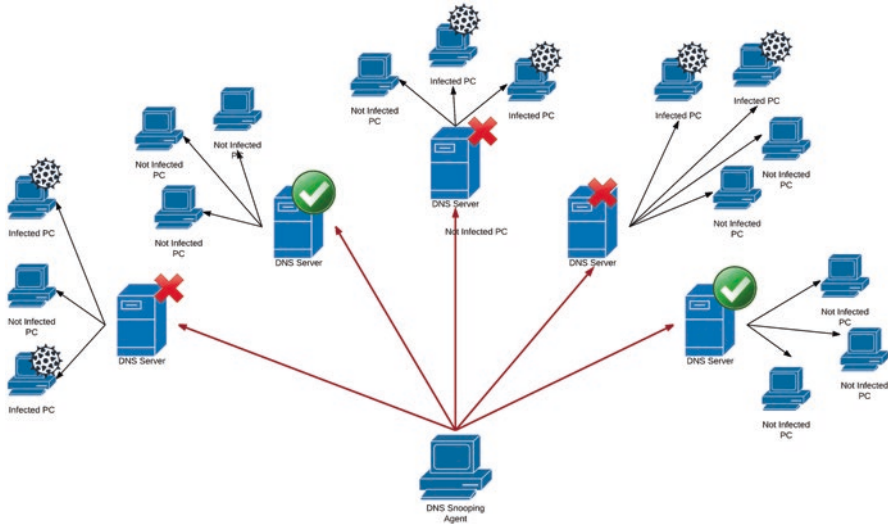


Fig. 2.8 Sinkhole redirection



**Fig. 2.9** DNS cache snooping

Cache snooping approach consists of analyzing the caches to identify illegitimate or unexpected DNS queries, which potentially could point to botnet presence.

### 2.5.2.3 Infiltration

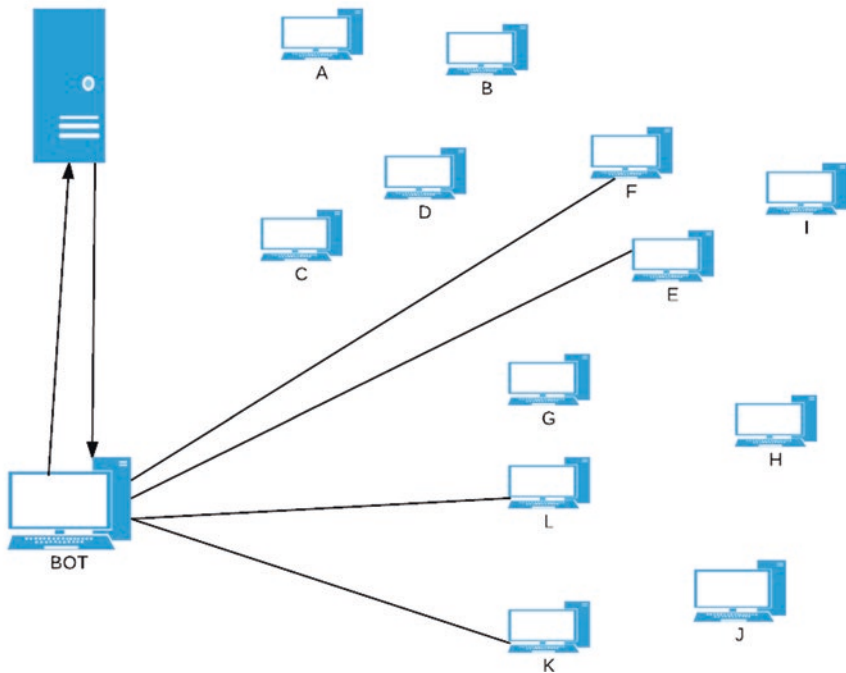
Infiltration techniques can be divided into software- and hardware-based techniques. Software-based infiltration technique can be used to monitor the traffic and bots executable to achieve control of bots in network whereas hardware-based infiltration allows to access command and control server and also to wiretap the communication between the nodes.

This usually requires the reverse engineering of the botnet infrastructure. This infiltration is a precise analysis which is useful for identification of potential weakness of infrastructure. The extracted knowledge is always very useful to achieve a commanding position in fighting back against botnet infection (Plohmann et al. 2015).

### 2.5.2.4 Tracking of Fast-Flux Network

Fast-flux networks consist of linking a single or few domain names with a large pool of IP addresses controlled by the botmaster, as illustrated by Fig. 2.10. Botnets use fast-flux networks to introduce secrecy of their actions and grow the consistency of their network and command configuration. This increases the stealth of the botnet, making detection of the C&C server much harder. Fast-Flux networks use promptly altering DNS records, indicating at a large number of hosts, and substitute as supplementary

DNS Server owned by  
botmaster



**Fig. 2.10** Fast-flux network attack

proxy layer to hide the actual content delivery systems. The proxy nodes are usually compromised workstations of the botnet itself (Plohmann et al. 2015).

Typically, the IP address assigned by the botmaster DNS Server is valid for only a few minutes that is indicated by the Time to Live (TTL) value.

The detection approach used in this case consists of monitoring and identifying the DNS server with low TTL values. Correlating such information with other parameters could expose the presence of botnet activity.

## 2.6 Defense Against Botnet Using Network Security Devices

Traditional network security appliances and devices (i.e., IDS, firewall, antivirus) play an important role in defending against botnet. Although taken in isolation these devices may not be enough, but they are essential components in any protection strategy. However, appropriate configuration must be performed for these devices to be effective in the fight against botnets.

### ***2.6.1 Intrusion Prevention and Detection Systems***

Intrusion Detection Services are performed on three different platforms: some instances filter intrusions on each individual node of network with help of applications which are called host-based intrusion prevention systems, whereas some other scenarios consist of a central device acting as an intrusion prevention system and a single device serving the entire network needs. In very high-risk infrastructure a combination of Network and Host Intrusion prevention is used to detect Botnets including when encrypted data is involved, as Network-Based Intrusion Prevention cannot detect Botnet in Encrypted traffic (Andriesse and Bos [2014](#); Ollman [2009](#)).

### ***2.6.2 Network Firewalls***

Most of the network firewalls enabled with Botnet traffic filtering provide reputation-based control in network based on ratings of IP address or domain name. This integrates with an external central repository of database of known malicious devices and domains, and dynamically stops the attacks originating from these sources. For unknown attack sources, the firewall always checks for traffic flowing to/from communication potential botnet C&C server reports/logs such occurrences.

Network firewalls filter traffic with the following components (Stawowski [2015](#)).

#### **2.6.2.1 Dynamic and Administrator Blacklist Data**

Filtering is done using a central database of malicious domains and IP addresses from central repository. This database is maintained by different vendors like cisco, Websense, and IronPort (Cisco White Paper [2015](#)).

#### **2.6.2.2 Traffic Classification and Reporting**

For classification of Botnet Traffic, the active filter associates the source and destination addresses of user data besides the IP addresses that have been revealed for the several lists and logs and accounts the administrator and dynamic database (Cisco White Paper [2015](#)).

#### **2.6.2.3 Domain Name System Snooping**

To ensure the binding of IP addresses to domains that are listed in central repository of database, the Network Firewall uses DNS Snooping in combination with DNS Inspection. The Firewall matches DNS Snooping lookup with DNS replies.

Firewall builds a reverse cache, which compares the IP address in user replies to actual known legitimate domain; if the domain matches then it is considered as clean traffic else it is flagged as bot traffic (Paquet 2015).

## 2.7 Security Measures Against Botnets

### 2.7.1 Network Design

Network design must be done in such a way that intruders and malware are not able to exploit existing susceptibilities. Defense in depth strategy in network against Botnet helps to mitigate even zero day attacks on network and helps to streamline security operations.

This involves making use of layered security systems on each segment to ensure security against bots (Boyles CCNA Security Study Guide) (Fig. 2.11).



**Fig. 2.11** Security measures chart

### **2.7.1.1 Advance Threat Protection**

Advanced threat protection systems must be used to mitigate layers 3 and 4 traffic and block all unintended traffic and allow only traffic initiated for trusted network and enable data control on edge of network (Cisco White Paper [2015](#)).

### **2.7.1.2 Intrusion Prevention and Detection System**

It enables the capability of deep packet inspection and anomaly detection by analyzing data from layer 4 up to layer 7 of network and correlated the events to protect network against botnets. Intrusion prevention is the most important component of network filtering. Thus it is always good to deploy both host and also network-based appliances, as while network-based detection fails to mitigate encrypted attacks, it enables synchronizing with a central repository of malicious patterns of intrusions and protects network against it [25, 26].

### **2.7.1.3 Email Security Systems**

Email being most important component of work flow it's very important to allow mails and also filter threats associated with botnet infection using email filtering engines.

### **2.7.1.4 Forensic Analysis**

Forensic-enabled devices allow correlating the security events in network and trace the origin of attack. This allows administrators to act efficiently on bots and protect other network users against them (SANS Institute InfoSec Reading Room [2015](#)).

### **2.7.1.5 Security Event Monitoring**

Event monitoring allows keeping track of all events in network and gives a comprehensive report of threats against network and also enables the transparency in network monitoring (Stawowski [2015](#)).

## **2.7.2 Application Usage**

Botnet can be prevented by monitoring and establishing normal patterns of usage for applications on individual nodes. The following application natures can be used to mitigate botnet against host.

### 2.7.2.1 HIPS (Host-Based Intrusion Prevention System)

Host-based intrusion prevention systems are application model of network-based IPS services to prevent network attacks on host (Scarfone and Mell 2007).

### 2.7.2.2 End Point Security

End point security applications monitor and protect the host against known viruses and malware and also observe and identify malicious activities in the behavior of the host computer; once if any abnormal activity is observed in the computing process, the application itself stops the suspected processes (Scarfone and Mell 2007).

### 2.7.2.3 Application Firewall

Application firewall can be used to block unwanted ports especially common ports of botnet attack such as ports 25 and 21 which are generally used by bots to transfer data.

## 2.8 Conclusion

In this chapter, we presented Zeus, Koobface, and Windigo as some of the most impactful botnets. There are many other different ones which have their own targets (such as desktop end users or mobile devices), and also use diverse avoidance techniques. The chapter also summarizes the passive and active countermeasures against botnets as well as some defensive mechanisms which can be implemented on the network.

## References

- Andriess D, Bos H (2014) An analysis of the ZeuS peer-to-peer protocol. IR-CS-74, rev
- Andriess D, Rossow C, Stone-Gross B et al (2013) Highly resilient peer-to-peer botnets are here: an analysis of Gameover Zeus. VU University of Amsterdam, Amsterdam
- Baltazar J, Costoya J, Flores R (2009) The real face of KOOBFACE: the largest web 2.0 botnet explained. Trend Micro Threat Research
- Bilodeau O, Bureau P, Calvet J et al (2015) Operation Windigo. [http://www.welivesecurity.com/wp-content/uploads/2014/03/operation\\_windigo.pdf](http://www.welivesecurity.com/wp-content/uploads/2014/03/operation_windigo.pdf). Accessed 22 July 2015
- Boyles T (2010) *CCNA Security Study Guide*. Indiana: Wiley Publishing, Inc., 2010
- Cisco White Paper (2015) Combating botnets using the cisco ASA botnet traffic filter. [http://www.cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/white\\_paper\\_c11-532091.pdf](http://www.cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/white_paper_c11-532091.pdf). Accessed 26 July 2015
- Falliere N, Chien E (2009) Zues: King of the bots. Symantec Corporation, Cupertino, CA

- Ferguson R (2015) The history of botnet—part I. <http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-i/>. Updated 24 Sept 2010. Accessed 20 July 2015
- Ferguson R (2015) The history of botnet—part II. <http://countermeasures.trendmicro.eu/the-history-of-the-botnet-part-ii/>. Updated 24 Sept 2010. Accessed 20 July 2015
- Fortinet White Paper (2013) Anatomy of a botnet. Fortinet, Sunnyvale. [www.fortinet.com](http://www.fortinet.com)
- SANS Institute InfoSec Reading Room (2015) Defense in depth. <https://www.sans.org/reading-room/whitepapers/basics/defense-in-depth-525>. Accessed 30 July 2015
- Irani D, Balduzzi M, Balzarotti D et al (2011) Reverse social engineering attacks in online social networks. DIMVA 2011, 8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment, Amsterdam, The Netherlands, 7–8 July 2011. Also published in “Lecture Notes in Computer Science”, Vol 6739/2011, doi: [10.1007/978-3-642-22424-9\\_4](https://doi.org/10.1007/978-3-642-22424-9_4)
- Khattak S, Ramay NR et al (2014) A taxonomy of botnet behavior, detection, and defense. *IEEE Commun Surv Tutor* 16(2):898–924
- Ollman G (2009) Botnet communication topologies, understanding the intricacies of botnet command-and-control. Damballa Inc., Atlanta
- Paquet C (2015) Network security concepts and policies. <http://www.ciscopress.com/articles/article.asp?p=1998559>. Accessed 1 Aug 2015
- Plohmann D, Gerhards-Paddila E, Leder F (2015) Botnets: detection, measurement, disinfection & defense. <https://www.enisa.europa.eu/publications/botnets-measurement-detection-disinfection-and-defence>. Accessed 30 July 2015
- Scarfone K, Mell P (2007) Guide to Intrusion Detection and Prevention Systems (IDPS). National Institute of Standards and Technology, Gaithersburg
- Shin S, Gu G (2010) Conficker and beyond: a large-scale empirical study. In: Proceedings of annual computer security applications conference (ACSAC)
- Sophos Press Release (2007) Sophos Facebook ID probe shows 41% of users happy to reveal all to potential identity thieves. <https://www.sophos.com/en-us/press-office/press-releases/2007/08/facebook.aspx>
- Stawowski M (2015) Practical defense-in-depth protection against botnets. <http://www.cllico.pl/services/practical-defense-in-depth-protection-against-botnets>. Accessed 31 July 2015
- Thomas K, Nicol DM (2010) The Koobface botnet and the rise of social malware. Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware, 2010, pp 63–70

Information Security Practices

Emerging Threats and Perspectives

Traore, I.; Awad, A.; Woungang, I. (Eds.)

2017, VII, 104 p. 51 illus., 31 illus. in color., Hardcover

ISBN: 978-3-319-48946-9