ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

BOTNET ATTACK IN COMPUTER NETWORK SECURITY

S.Nagendra Prabhu
Department of CSE,
Malla Reddy College of Engineering and Technology
Hyderabad, India

D.Shanthi
Department of CSE
PSNA College of Engineering
Dindigul, India

Article DOI: https://doi.org/10.36713/epra4905

ABSTRACT

Among the various forms of malware, Botnet is the serious threat which occurs commonly in today's cyber attacks and cyber crimes. Botnet are designed to perform predefined functions in an automated fashion, where these malicious activities ranges from online searching of data, accessing lists, moving files sharing channel information to DDoS attacks against critical targets, phishing, click fraud etc. Existence of command and control(C&C) infrastructure makes the functioning of Botnet unique; in turn throws challenges in the mitigation of Botnet attacks. Hence Botnet detection has been an interesting research topic related to cyber-threat and cyber-crime prevention in network security. Various types of techniques and approaches have been proposed for detection, mitigation and preventation to Botnet attack. Here I discusses in detail about Botnet and related research including Botnet evolution, life-cycle, command and control models, communication protocols, Botnet detection, and Botnet mitigation mechanism etc. Also an overview of research on Botnets which describe the possible attacks performed by various types of Botnet communication technologies in future.

KEYWORDS— ***Bot; Botnet; C&C mechanism; communication protocols; honeynet; passive traffic; attacks; defense; preventaation; mitigation***

I. INTRODUCTION

The term "Bot" is nothing but a derived term from "ro-Bot" [2] which is a generic term used to describe a script or sets of scripts designed to perform predefined function in automated fashion. Botnet is the collections of bots or collection of compromised computers that are remotely controlled by its BotHerder [1]. Even though Botnets shows the trace of existence for several years ago, Botnet have only recently sparked the interest of the research community.

Generally *Botnet* is used to define networks of infected end-hosts, called *bots* that are under the control of a human operator commonly known as a *Botmaster*. Botnets recruit vulnerable machines using methods utilized by other classes of malware (*e.g.,* remotely exploiting software vulnerabilities, social engineering, etc.) [3], these machines create a C&C infrastructure between them to perform malicious activity.

Now in general the main difference between Botnet and other kind of malwares is the existence of C&C infrastructure. Hence in the mechanism of detection of Botnet, if we identify the location of C&C then Botnet can be detected, removed and prevented from various types of cyber-crimes. But this depends on the weakness and strengths in communication protocol which is adopted by Botnet to perform malicious attacks. Now on the other side, bots are used by search engines to spider online website content and by online games to provide virtual opponents.

More specifically on Internet relay chat (IRC) network bot"s function in channels include managing access lists, move files, share users, share channel information, anything else if right scripts are added. IRC bots are automated and controlled by events which could be commands given in a channel by other IRC bot or client with necessary privileges.

In this paper, an overview of current Botnets technology research has been provided. The remainder of the paper is organized as follows: Section 2 discusses background of Botnets. Section 3described about literature review, in this section, Botnet characteristics, and Botnet life-cycle are explained to provide better understanding of Botnet technology, Classification of bots & also describe about the communication protocols used by Botnet to communicate. In Section 4, classifies Botnet detection approach which is explained in two classes: First identify the cryptographic key of the botnet communications for figuring out the botnet operations. We then compromise the botnet entities for tracing back to the botmaster across the stepping-stones. In section 6, concludes the total work done in this paper and we explain about the further Botnet attacks possible or Botnet developments in future.

II. BACKGROUND OF BOTNET

Botnets have been in existence for about 10 years [13].Security experts have been cautioning the public about the threat posed by botnets for some time. Still, the scale and magnitude of the problem caused by botnets are underrated and most users do not comprehend the real threat they pose[13].

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

A. How Does a Botnetworkwork ?

Most botnets are designed as distributed-design systems, with the main botnet operator (*botmaster*) issuing instructions directly to a small number of systems. These machines propagate the instructions to other compromised machines, usually via Internet Relay Chat (IRC) [14]. The constituents of a typical botnet include a server program, client program for operation, and the program that embeds itself on the victim's machine (*bot*). All three of these usually communicate with each other over a network and may use encryption for stealth and for protection against detection or intrusion into the botnet control network. Botnets are effective in performing tasks that would be impossible given only a single computer, single IP address, or a single Internet connection. Originally, botnets were used for distributed denial of service attacks. (See Figure 1) Most modern web servers have developed strategies to
combat such DDoS attacks, making this use of a botnet less effective [14]. When infecting a computer, the bots connect to IRC servers on a predefined channel as visitors and waited for messages from the botmaster. The botmaster could come online at any time, view the list of bots, send commands to all infected computers at once, or send a private message to one infected machine.

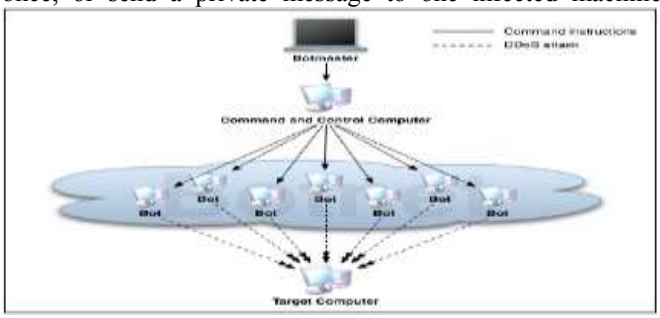

Figure 1 Example of Botnet Attack

B. Why are Botnets dangerous today?

Botnets today are one of the most dangerous species of network-based attack because they use large, coordinated groups of hosts to execute both brute-force and subtle attacks. A collection of bots, when controlled by a single command and control (C&C) infrastructure, forms a botnet [15]. Since the bots work together in large groups taking orders from a centralized botmaster, they can cripple a large-scale networks in a short time. A lot of work has been done trying to mitigate the efforts of botnets to avoid data and financial loss. However hard the industry works towards patching the known vulnerabilities in hosts and networks, there are always more unpatched or unknown vulnerabilities that malicious developers and cyber criminals may exploit.

III. LITERATURE REVIEW

This section reviews selected literature to discuss the current research that has been published about botnets. We first identify the motivations behind building and operating botnets and how these motivations have evolved over time. Then, we discuss the current research on how to track and disable botnets.

A. Botnet Life cycle

A typical Botnet can be created and maintained in five phases. This is depicted in Fig. 2.

1. In first phase, firstly Botmaster infect victim host with Bot through the social engineering, mail attachments, automatic scan, exploit and compromise etc mechanisms.
2. In second phase, Bot connected to command and control channel
3. In third phase, Botmaster send command through IRC/HTTP/P2P C&C Channel to bots
4. In fourth phase, repeat, soon the Botmaster has a large number of army bots to control from a single point.
5. And in last phase, bots are updated with a new version or new business functionally through their operator which issue payload command.

Hence the above discussion elaborates all five steps about how a bot is infected to other hosts. In addition it also gives insight into how the Bot increase their quantity means its capacity on a network to perform malicious activity and harm the users.

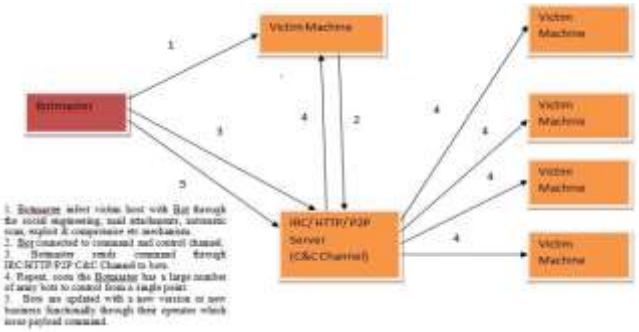

Figure 2 A Typical Botnet Life-cycle

B. Classification of Botnet

1. Based on Network Protocols

For a Botmaster to send commands to a bot, it is essential that a network connection must be established between the zombie machine and the computer transmitting commands to control it. Here all network connections are based on protocols that define rules for the interaction between computers on the network. Botnets can be classified according to network protocols follow as:

a. *IRC-oriented:* This is one of the very first types of Botnet in which bots are controlled via IRC channels. Each infected computer connected to the IRC server (master) indicated in the body of the Bot program, and waited for commands [48]

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

from its master on a certain channel (eg-IRC Botnet).

b. *IM-oriented*: This type of Botnet is not particularly common. It differs from IRC-oriented Botnets only in that it uses communication channels provided by IM (instant messaging) services such as AOL, MSN, and ICQ etc and due to the difficulty of creating individual IM accounts for each bot. The Biggest problem in this, Bots should be connected to the network and must remain online all the time [4] and each bot needs its own IM account to perform malicious activity. As result, owners of IM-oriented Botnets only have a limited number of registered IM accounts at their disposal, which limits the number of bots that can be online at any one time. Of course, they can arrange for different bots to share the same account, come online at predefined times, send data to the owner's number and wait for a reply for a limited period of time, but this is inefficient because it takes such networks too long to respond to their masters' commands to perform an activity.

c. *Web-oriented*: This is a relatively new and rapidly evolving type of Botnet designed to controlling zombie networks over the World Wide Web. A bot connects to a predefined web server (master), receives commands from it and transfers data to it in response. And wait to get a signal from its master to perform some activity for eg-HTTP Botnet.

d. *Other:* In this, there are other types of Botnets that communicate via only their own protocol that is only based on the TCP/IP stack, i.e., they only use transport-layer protocols such as TCP, ICMP and UDP.

2. Based on communication topologies

In this section we will describe about "how bot communicate" between each other. So according to the C&C channel, we categorized Botnet topologies into three different models, the Centralized model and the Decentralized model and Unstructured C&C Model [3].

a. Centralized model: Hossein et al [5] explain the model where, one central point (C&C server) has been used for exchanging commands and data between the Botmaster and Bots. Actually C&C server runs certain network services such as IRC or HTTP. So advantage of this model is small message latency which cause Botmaster easily arranges Botnet and launch attacks. Here, all connections and action performs through the C&C server; therefore, the C&C is a critical (weak) point in this model. If somebody manages to discover and eliminates the C&C server, the entire Botnet will be useless and ineffective.

b. Decentralized model: In this model the communication system does not completely depend on some selected servers, for discovering and destroying a number of Bots. As a result, attackers exploit the idea of Peer-to-Peer (P2P) communication as a Command-and Control pattern which is more resilient to failure in the network. Figure 4 shows that, depicts the decentralized (P2P) model where there is no Centralized point for communication. In this, each bot keeps some connections to the other Bots of the Botnet where Bots act as both Clients and servers. A new bot must know some addresses of the Botnet to connect there. Here if Bots are offline, the Botnet can still continue to operate under the control of Botmaster. Since P2P Botnets usually allow commands to be injected at any node in the network, the authentication of commands become essential to prevent other nodes from injecting incorrect commands [5] for eg: DNS, P2P protocol based botnet.

C. Communication Protocol in Botnet

A communications protocol is a system of digital message formats and rules for exchanging those messages in or between computing systems and in telecommunications [6]. Today Botnet usually use well defined communication protocols to perform attack. So studying about communication protocols can help us determine the origins of a Botnet attack and decode conversations between the bots and the Botmasters [3].

Communication protocol can be classified in three different categories:

1. **IRC protocol**: A most common protocol used by Botmasters to communicate with their Bots. IRC protocol mainly designed for one to many conversations but can also handle one to one, which is very useful for Botmasters control their Botnet. However, security devices can be easily configured to block IRC traffic [3].

Weaknesses of IRC bots:

- Usually unencrypted
- Easy to get into, take over or shut down
- Due to the dependability more on C&C Server, Single point of failure is there [7].

2. **HTTP protocol**: Generally HTTP protocol is a popular Botnet due to its communication method by sending message as HTTP response and HTTP GET response to perform attack which is difficult to be detected. So Using the HTTP protocol, Botnet usually bypass security devices.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

Weaknesses of HTTP based bots:

- Due to the dependability more on C&C Server, single point of failure is there [7].
- Bypass attack possible

3. **P2P protocol**: Recently, more advanced Botnet used decentralized model for their communications [3, 8]. For eg; Phatbot[7] , Storm, Nugache [7], Peacomm [7], Conficker and Slapper[9] used P2P communication protocols to perform malicious activity.

Weaknesses of P2P based bots:

- Strict Dependent ability on previous or others nodes
- These will not generate a sound Botnet
- Not mature
- If these have poor connectivity then easily traced
- Compared to HTTP Botnet, these have no hardly encryption /authentication code
- For large number of nodes, creates a complex structure and generates a large amount of traffic
- WASTE P2P protocol [8] is not scalable across a large network.

IV. Botnet Detection Techniques

In this paper, We present our Pebbletrace scheme for the traceback to the botmaster. It first identifies cryptographic keys of the botnet communications for configuring botnet operations and then traces back to the botmaster. First identify the cryptographic key of the botnet communications for figuring out the botnet operations. We then compromise the botnet entities for tracing back to the botmaster across the stepping-stones.

A. Key Identification

Amajor difficulty for analyzing botnet attack traffic is that communication between bots and C&C servers are usually encrypted, and the encryption keys are to be identified first. Traditional memory forensic key identification problem was studied (e.g. [15]), however, we have to meet the following new challenges:

- *No source code.*

Traditional key identification schemes usually investigate source code, e.g. [16]. However, it is hard to obtain bot source code — often not even the bot binaries. Static analysis on source code and binaries cannot be conducted as in memory forensic.

- *Abnormal code pattern*

Attackers do not follow the standards to implement their encryption schemes even though they are mathematically equivalent. Identification schemes based on standard key words, such as prefixes and formats usually do not apply.

- *Hard to verify candidate keys*

– Traditional memory forensic often verifies candidate keys either by checking key scheduling properties and entropy of keys, or by applying them to encrypted text to obtain plain text that is meaningful for manual checking. The former scheme is prone to false positives. The latter one does not work well either, for the plaintext of the sniffed botnet traffic may not be meaningful by the design of the botmaster. Therefore, one cannot verify a candidate key by manual checking for decrypted botnet traffic.

– Keys should be identified for traceback before attackers disappear from the C&C servers and stepping-stones. Therefore, manual checking is often ruled out due to the short duration of botnet attacks, and we have to consider automated verification of candidate keys.

- *Low false positives*

It alerts the attacker if we apply an incorrect key to manipulate the botnet traffic for a traceback; he will quickly tear down the C&C server and stepping-stones. Therefore, low false positives are required, and it is not well studied or emphasized in the published literature.

B. Stepping-stones

Attackers usually hide behind stepping-stones from Web proxy, VPN and SSH tunneling. The latest botnets also leverage social networks and anonymous networks. In this article, we mainly consider proxy, VPN and SSH tunneling. We are focused on the two main challenges: key identification and stepping-stones, and present our Pebbletrace method for tracing back to botmasters in the network.

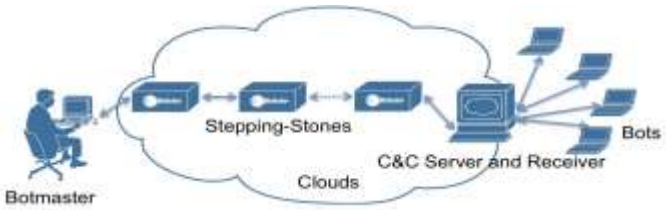

Figure 3: A Botnet Attack

C. Pebbletrace solution for Botnet attack

In the advent of a botnet attack we collect information of
attack traffic for an analysis. We first identify the encryption keys for decrypting the traffic. We then intercept the botnet traffic by entering our code – Pebbleware that enables us to
traceback to the botmaster.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

A. Information Collection

To traceback the botmaster, victims' local network administrator collects information from victim machines. Intrusion detection before information collection is out of scope of this paper. We assume that a local network administrator can obtain network traffic between bots and C&C servers, which communicate periodically. A network administrator can also collect memory image of victim machines and basic information of the attack event. The information includes hostname of the C&C server, the drop-zone URL and the types of botnets (e.g. Zeus). The local network administrator submits collected information — traffic records, memory image, and other basic information to traceback server and requests for a traceback service.

B. Key Identification and Extraction

When the traceback server receives traceback requests with the needed information, it first identifies the encryption key of attack traffic for: (i) Decrypting the attack traffic and figure out the needed information for traceback; and ( ) Embed Pebbleware in the botnet traffic and traceback
botmasters through stepping-stones. Our key identification scheme works without source codes and with vague traffic patterns only, is time efficient, and has low false positives. Given a memory image of victim machines, network traffic (ciphertext) and the type of botnets, we propose a threephase detection scheme that consists of (i) a pattern filter; ( ) an entropy analyzer; and (i) a verifier for identifying the symmetric keys used by bots. Figure 4 is an overview of the key identification scheme.

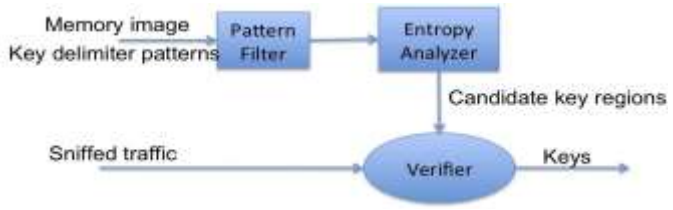

Figure 4: An Overview of Key Identification Scheme

Phase 1: Pattern Filter identifies suspected regions, which contain the key. We know the key size (number of bytes) yet not its location. However, we have information of the bit pattern before and after the key — its delimiters. We obtain such information from the previous works on botnets of their particular symmetric encryption schemes. Unfortunately, the delimiter pattern information is supposed to be vague so that it can adapt to multiple versions of the botnets. For example, a delimiter pattern for Zeus botnet where RC4 is used for encryption could be 2 consecutive zero bytes followed by 256 to 400 other bytes and another 2 consecutive zero bytes. The 256 to 400 bytes include RC4 256-byte S array (equivalent to symmetric key) and other overheads for Zeus. Given a memory image of a victim machine, we apply the Pattern Filter that contains key delimiter pattern information and obtain suspected regions, which may contain the key. However, the number of the suspected regions is usually large and we reduce it by an entropy analysis in Phase 2.

Phase 2: Entropy Analyzer further analyzes the suspected regions from Phase 1 and identifies several candidate keys for verification. The design idea is: a strong symmetric encryption scheme achieves security goal by constructing a pseudorandom string for encrypting a plaintext by an XOR with it. Hence a region that contains a pseudorandom string has high entropy value. Entropy search is first proposed in [12]. The entropy value of a string is:

$$\mathbf{E} = - \sum_{i=0,\ldots,255} p_i \log (p_i)$$

where p_i is the empirical probability of value & in the string. Since each byte has 8 bits of values from 0 to 255, a pseudorandom string has an entropy value closed to log (256) = 8. We compute entropy of each suspected region from Phase 1 and rule out the ones with entropy lower than a threshold value, thus greatly reduce the number of suspected regions, resulting in candidate key regions for further analysis. The threshold value can be varied by different types of botnets.

Phase 3: Verifier validates the candidate key regions and identifies the key in two steps. We first examine each candidate key region with a sliding window of the key size and check whether all properties of key scheduling are satisfied. For example, for RC4, it checks whether there are 256 bytes with 256 different values; it is the property of RC4 key scheduling. Similar approaches have been used in [18]. If we consider all of them, the false positives are too high. The second step of the verifier explores the idea that normal plaintext and customized encryption are usually weak for crypto-analysis. That is, the randomness of ciphertext mostly comes from symmetric key encryption schemes, rather than a plaintext of customized encryption. Therefore, if a correct key is found and applied to decrypting ciphertext traffic string, the string will have a significant drop in the entropy values after decryption. On the contrary, if an incorrect key is used for decryption, the entropy value remains high. Consequently, only the right key significantly reduces the entropy value by decrypting the ciphertext.

Algorithm I summarizes the algorithm of the key identification that contains three phases. The time to check a memory with size N is $O(C_0N + C_1M_1 + C_2M_2)$

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

where C0 is the time to check by Prototype Filter, C1 and C2 are time for computing entropies and time for verifying keys, and M1 and M2 are the number of suspected regions after the Pattern Filter and Entropy Explorer, respectively. Note that C1, C2 ≫C0, and N ≫M1 ≫M2. Our algorithm avoids expensive checking on the whole memory image by narrowing down candidate keys at each phase.

Algorithm I: Suggested Key Identification Scheme

```
Require: memory_image, network_traffic,
key_delimiter_prototype,
      Lowest_entropy = 8;
     Detected_key = NULL;
     while EOF != (byte=read_a_byte(memory_image))
do
suspect_region = update(suspect_region, byte);
if true!= check_pattern(suspect_region,
key_delimiter_prototype) then
      continue;
else if entropy(suspect_region) < entropy_threshold then
      continue;
else
       while key = sliding_window(suspect_region) do
if false== satisfy_key_scheduling(key) then
      continue;
else
      plaintext = decrypt(key, network_traffic);
                                  key.entropy =
entropy(plaintext);
      if key.entropy < lowest_entropy then
                                       lowest_entropy =
key.entropy;
           detected_key = key;
      end if
end if
       end while
end if
      end while
  return detected key;
```

D. Pebbleware for Pebbletrace

A traceback is possible when a botmaster wants to communicate with victims to obtain information. However, botmaster usually does not communicate directly with the victims or C&C servers. Instead, he may communicate through one of his accomplices through stepping-stones. Consequently, we can initiate our traceback process from the victim and/or receiver machine. We want to keep track of the communication path from the victim/receiver machine to the botmaster by spreading "pebbles" — Pebbleware. A Pebbleware is a piece specially designed executable code that reveals its host machine information. We piggyback Pebbleware on the communication packets from the victim/receiver to the botmaster. When it reaches the botmaster, it reports to us the host machine information.

Figure 5 shows an overview of our Pebbletrace scheme for cloud-based botnets. The trackback starts from the receiver, i.e. C&C server, or from the victim. A local network administrator submits a request to traceback server providing sniffed traffic, memory image and basic information of victim. Traceback server then extracts encryption key of botnet communication by our key identification algorithm. After that traceback server creates a Pebbleware and encrypts it with the detected botnet key.

Figure 5: An Overview of Pebbletrace for Cloud-based Botnets

Pebbleware is then spread (piggybacked) from the receiver or the victim to the botmaster's machine along the same path as the responses from the victims while pabbleware is executed at its host machine. As a result, the botmaster's machine is forced to send its IP address to the traceback server when Pebbleware reaches and executes there. Sometimes botmaster launches attacks outside their home domain, e.g. Starbucks. In this case private IP address does not reveal his identification, and Pebbleware should also collect other information, such as the hostname, routing table, files, directories, and screen snapshots.

Pebbleware is designed by exploring the zero day vulnerabilities of: (1) Vulnerabilities of C&C servers. As other software developers, botnet developers can also make mistakes, leaving vulnerabilities on C&C servers. In [18], a vulnerability of Zeus C&C server due to careless input examination is reported. In [19], vulnerabilities of Unreal IRCd botnet are studied. The vulnerabilities enable us to take down botnets and also to compromise C&C servers for a Pebbletrace through stepping-stones. (2) Vulnerabilities of clients. Client side zero-day vulnerabilities can be explored to traceback through stepping-stones across clouds, if botmaster uses the client in his local machine. Typical client side vulnerabilities include web browser vulnerabilities and SSH client vulnerabilities. Pebbleware, which exploits such vulnerabilities, is usually embedded in the application layer payload to be relayed by stepping-stones. Finally, when it reaches the botmaster's machine it is executed and sends the host machine IP address to the traceback server.

ISSN (Online): 2455-3662
EPRA International Journal of Multidisciplinary Research (IJMR) - Peer Reviewed Journal
Volume: 6 | Issue: 8 | August 2020 || Journal DOI: 10.36713/epra2013 || SJIF Impact Factor: 7.032 ||ISI Value: 1.188

V. CONCLUSION

As well discussed above since 1988, Botnet have evolved from the beginning assistant tool to the predominant threat in modern internet and as discussed in this paper, in 1988 Botnet was not a malicious activity but later in 1998 , attacker use the bot to perform malicious activity via cyber crime. That is Botnets pose a significant and growing threat against cyber-security as they provide a key platform for many cybercrimes such as DDoS attacks against critical targets, malware dissemination, phishing, and click fraud etc. Although the number of bots to each Botnet seems to be decreasing, the monetary damaging power of the Botnets is continuously increasing given the development of internet bandwidth due to change in technology.

This study is focused on the attacks that a botmaster attempts to steal sensitive data from the victim machines and we can spread our tracing pebbles along with the stolen data all the way back to the botmaster. However, if a botmaster only wants to communicate with the victims, more intelligence has to be integrated in the Pebbleware. As most of the botnet traffic is encrypted by symmetric keys for efficiency, we only study symmetric key identification.

Asymmetric key identification is a challenging research topic in general. However, our results on symmetric key identification and the specific botnet application environment may shed light on future investigation.

Anonymous network and social network services, such as Twitter and Facebook, might be moved into clouds. Their security, particular defense against botnets, is an intriguing research topic.

Acknowledgment

We would like to thank Dr. VSK Reddy, Principal, Malla Reddy College of Engineering and Technology and Dr. D. Sujatha, HOD, CSE dept., MRCET for their kind help and motivation of doing research.

REFERENCES

1. *Hossein Rouhani Zeidanloo, Azizah Bt Manaf, Payam Vahdani, Farzaneh Tabatabaei, Mazdak Zamani, "Botnet Detection Based on Traffic Monitoring" IEEE transaction,2010.*
2. *SANS Institute InfoSec Reading Room provided a description on "Bot & Botnet: An overview" research on topics in information security, 2003.*
3. *Generation of a robust Botnet capable of maintaining control of its remaining bots even after a substantial portion of the Botnet population has been removed by defenders.*
4. *S. Nagendra Prabhu, Kemal Sultan Abdo & Gashaw Bekele Kabtimer, 'Introducing Proxy Cloud Storage Using Internet Information Services in University and Utilization Of Its Resources in the Academic Institution' in Journal of Network communications and Emerging Technologies, Volume 8, issue 2, February 2018.*
5. *Hossein Rouhani Zeidanloo, Farhoud Hosseinpour , Farhood Farid Etemad, "New Approach for Detection of IRC and P2P Botnets" , International Journal of Computer and Electrical Engineering, Vol.2, No.6, December, 2010, 1793-8163.*
6. *Prabhu S, Chandrasekar V & Shanthi S, Improving the performance of IDS using Arbitrary Decision Tree in Network Security, International Journal of Advanced Science and Technology Vol. 29, No. 3, (2020), pp. 3453 - 3462*
7. *Hossein Rouhani Zeidanloo, Azizah Abdul Manaf,"Botnet Command and Control Mechanisms", IEEE transactions, 2009.*
8. *Nagendra Prabhu, S & Shanthi, D 'A Survey on Anomaly Detection of Botnet in Network' published in Volume 2, Issue 1 of International Journal of Advance Research in Computer Science and Management Studies in the year of 2014.*
9. *Robert F. Erbacher, Adele Cutler, Pranab Banerjee, Jim Marshall,"A Multi-Layered Approach to Botnet Detection", IEEE conference, 2010.*
10. *Nagendra Prabhu, S & Shanthi Saravanan, D, 2017, 'An Efficient Botnet Detection System in Large Scenario Networks Using Adaptive Neuro Fuzzy Inference System Classifier', Journal of Computational and Theoretical Nanoscience Vol. 14, 1–5, 2017*
11. *P. Wang, S. Sparks, and C. C. Zou, "An advanced hybrid peer-to-peer botnet," in Proc. In Workshop on Hot Topics in understanding Botnets, 2010.*
12. *Prabhu, SN & Shanthi, D , 'Cloud Computing Defense Threats and Responses against DDOS Attack', published in Volume 5, Issue 4 in International Journal of Science, Engineering and Technology Research (IJSETR) in Vol. 5, 1–4, April 2016.*
13. *http://www.kaspersky.com/reading_room?chapter=207716701*
14. *S. Nagendra Prabhu, The title name " Examining Zeus Botnet by Adopting Key Extraction and Malicious Traffic Detection Framework using DNS" in International Journal of Applied Engineering Research ISSN 0973-4562 Volume 10, Number 3 (2015) pp. 6987-7007 © Research India Publications*
15. *K. Harrison and S. Xu, "Protecting cryptographic keys from memory disclosure attacks," in International Conference on Dependable Systems and Networks, 2007, pp. 137–143.*
16. *S. Nagendra Prabhu, S. Shanthi & V. Chandrasekar,' Botnet Revealing By Consuming Data Mining Techniques In Cloud System', International Journal For Recent Development In Science & Technology (IJRDST), Volume 03, Issue 12, Dec 2019 ISSN 2581 – 4575..*
- A. *Shamir and N. Van Someren, "Playing 'hide and seek' with stored keys," in Financial Cryptography, 1999, pp. 118–124.*
17. *J. Li et al., "Large-scale IP traceback in high-speed Internet: Practical techniques and theoretical foundation," in IEEE Symposium on Security and Privacy, 2004, pp. 115–129.*
18. *Y. Zhang and V. Paxson, "Detecting stepping stones," in USENIX Security Symposium, vol. 171, 2000, p. 184.*