Lightweight C&C based botnet detection using Aho-Corasick NFA

Udhayan J[1], Anitha R[2] and Hamsapriya T[3]

[1]Department of Information Technology, Karunya University, Coimbatore, India
udhayan@karunya.edu, udhayangodwin@rediffmail.com
[2]Department of Mathematics and Computer Applications, PSG College of Technology, Coimbatore, India
[3]Department of Information Technology, PSG College of Technology, Coimbatore, India

ABSTRACT

Botnet distinguishes itself from the previous malware by having the characteristics of a C&C channel, using which a Botmaster can control the constituents of the botnet. Even though protocols like IRC, HTTP and DNS are exploited to incorporate C&C channels, previous analysis have shown that the majority of the botnets are usually based on IRC. Consequently in this paper the Aho-Corasick NFA based detection is proposed to detect the C&C instructions which is exchanged in IRC run botnets. However the ability to detect botnet is limited to the existing bot commands. Therefore a counting process which analyses every IRC messages is introduced to detect the existence of malicious codes. This detection method and various existing methods have been evaluated using real-world network traces. The results show that the proposed C&C Instruction based IRC detection method can detect real-world botnets with high accuracy.

KEYWORDS

Botnet; IRC, C&C, Flow based detection, Behaviour based detection, Signature based Detection

1. INTRODUCTION

The botnet is the present day threat on Internet [12]. It is a network of infected end-hosts (bots) controlled by the remote Botmaster [2][8]. Remote Botmaster controls the entire botnet through C&C (Command and Control) server. The C&C server is the software installed by the Botmaster in the remote server with or without the knowledge of the server administration. Botmaster then spreads out bot agents using the C&C server. The bot agent is a piece of code spread to exploit the vulnerable computers and make them as the bots (compromised computer). Like viruses, bot agents can automatically scan the environment, propagate themselves and compromise the vulnerable computers [9]. The more the bot agents propagate, the bigger the size of the botnet will be. The difference between the virus and the bot is that, while viruses act individually according to an inflexible program, bots respond to external commands and then execute the attacks as commanded. Furthermore through C&C instructions the Botmaster can update the entire network with new capabilities as they are developed.

Bots usually lurk silently within ordinary desktop computers, inert and undetected, until C&C server issues orders to strike. Bots are devoted to carry out several nefarious tasks as well. For instance (**DDoS) distributed denial-of-service attack** [18] an attempt to make a computer resource unavailable to its intended users. Most of the Modern days DDoS are constituted from botnet which constitutes millions of zombies (compromised computers). As the result, even the low rate traffic generated concurrently by computers in botnet can accumulate a mammoth

DOI : 10.5121/ijnsa.2010.2413

volume of packet at the receiving end and force the target server to cut off either temporarily or indefinitely.

In all the scenarios of botnet hosted attacks C&C Instructions play a vital role in passing over commands to bots. Until the bots receive C&C instructions it remains inactive and harmless. Most of the cases IRC channels are used to distribute C&C instruction [1]. In this paper the sec.2 shares the background information about IRC and C&C channels. Sec.3 discusses about the related works which presents various detection techniques along with their limitations. The C&C instruction based botnet detection over the IRC channels is discussed is sec. 4.1. Moreover C&C detection is capacitated with Aho-Corasick NFA detection procedure which is discussed in sec.4.2. In addition to that a counting process based verification technique is proposed to reduce the false positives and to help updating the new detection signatures which is discussed in sec.4.3. The sec.5 presents a complete overview in the proposed detection methodology. Moreover various detection techniques are evaluated through scrutinizing the real-world IRC C&C traces in sec.6 & sec.7. Eventually the concluding remarks is present in the sec.8.

2. BACKGROUND

Most of the cases, botnet is established, controlled, monitored, managed and manipulated through IRC C&C Channel [4][12]. Therefore the following sections discuss about the IRC C&C channels and its strength and weakness.

2.1. IRC C&C Channel

IRC is a real time chatting system that exchanges the client messages via channels (chat rooms) [9]. Channels can be global to entire Internet or local to a single network.

Generally Botmaster prey on the IRC server for hosting the C&C channel, this helps the Botmaster to gain control over the bots through (C&C) IRC channels by inviting the bots to join the C&C channel. Once the Botmaster assigns the C&C channel for the bots, he can send commands as genuine IRC chat messages and receive acknowledgements as chat messages from the bots through C&C channel.

Initially to establish the channel for the bots, like the genuine IRC clients, IRC bots need the login details like User ID and password to connect with the C&C channel. These details are normally issued by the Botmaster through C& C channel as PRIVMSG (private messages) [4]. PRIVMSG are used by channel admin to control the channel activates. Once the login details are provided to the bots, the bots will login to the specified C&C channel and stay connected, and listening for the C&C instructions from the Botmaster to arrive. This is possible because of the inherent multicast feature of IRC protocol enables the channels with one-to-many communication. Therefore the C&C instructions sent by the Botmaster will be delivered to all the bots listening to that specific channel. This helps the Botmaster to co-ordinate the bots to zero-in the devastating attacks like DDoS attacks. The C&C instructions issued from the Botmaster are usually the IRC Private Messages. This is because the Private Messages are used by the owner of the group to control the group and it is high priority messages.

Following are the few reasons why Botmaster are attracted towards IRC

1 Two-way communication
2 IRC is well established and understood protocol
3 Freely available IRC software and channel control program modules
4 Interactive and offers redundancy with linked IRC servers
5 Simplicity and distributive nature [21]
6 Remote control feature.

Moreover, segregating the IRC from other traffic at the network vantage point is difficult [12] because IRC uses the TCP/IP protocol, whereas the IRC header is embedded within the TCP payload. This will therefore look like a genuine TCP packet.

2.2. Advantage of eliminating C&C

In IRC-based botnets, if the traffic is not encrypted, capturing the packets from the compromised node will reveal the name of the IRC network, the name of the IRC server and the name of the IRC channel [4].

Once if the IRC C&C channel is identified, the IRC server can be tracked [17]. The C&C channel application can be killed instantly by informing the administrative circle of the IRC server. The Botmaster literally losses the control over the bots as soon as the central C&C channel is removed. This is called as single-point-of-failure problem [16][20]. Thus the common method in botnet fighting is to shutdown the known C&C channels to prevent infected machines from receiving further commands. Although the botnet is successfully disabled, the zombies remain infected. This vulnerability either helps the previously possessing Botmaster to recapture his botnet from another IRC Server or helps the new Botmaster to augment his botnet.

For instance if a new Botmaster approaches the previously infected system to find a way to compromise it, if he check for the existence of bot agent with commands like “bot.about” the existing bot agent will respond with the version details so that the Botmaster can exploit it to constitute that particular system to the botnet. Therefore the network based detection method capacitates the local network administration with the feature of alerting the infected computers. As the result, the owners of the contaminated machines can be informed and is able to clean it, before getting exploited again.

3. RELATED WORK

Since the C&C instructions are the software generated messages, it exhibits a pattern different from the genuine IRC client messages. Therefore, botnet can be detected by analyzing the IRC messages. In this section, the analysis of various previous detection methods for IRC based botnet is presented.

3.1. Signature based detection

Signature based detection [6][19] is nothing but pattern matching. This extracts the features from the IRC packet and performs the cross check with the existing IRC C&C signatures stored in the database. If a match is found then it is declared as attack. The process of this method is easy because this compares simple byte sequences only. Moreover this kind of detection produces less or no false detections.

However the frequent update is essential to stay put with the evolving bots to prevent them from evading. This is a daunting task because modern Botmaster, by periodically altering the source code bring changes to the signature which thwarts the signature based detection, and increases the number of false negative detections.

Moreover a separate database should be maintained which will augment from time to time, whenever there is a signature update. This induces performance problems and increases management cost.

Moreover, searching of the signature in database for every incoming packet, delays the detection and demands lot of buffering space to cache the incoming packets, which annoys the detection. Therefore if it is performed online, it introduces the delay in the communication. If it is performed offline, it permits evasion of few infected packets before detection.

3.2. Behaviour based detection

The interaction pattern or behaviour of bots varies from human [13]. Human interaction occurs frequently and with varying Intervals. If the log of the bot traffic is examined, bots stay idle for a long time; once it receives the command from the Botmaster, it responds quickly and then stays idle until it receives the next command [14]. Therefore the C&C channel detection becomes feasible by using spatial-temporal reasoning [11]. The inter arrival time between the C&C instructions of one bot will not vary or vary marginally when compared against another.

Drawback

This type of detection has its limitations too, while manipulating by spatial-temporal reasoning, the genuine IRC clients may also exhibit the same pattern. Hence to avoid false positives this method has to look for multiple interactions. Therefore this type of detection can be eluded a little by altering the delay between events. However introducing the delay will reduce the performance of botnet [5].

Additionally, the spatial-temporal reasoning based detection cannot detect one-to-one conversation between Botmaster and bot. For example the Botmaster can deliver a C&C instruction through private channel to an individual client to install the keylogger and copy the password.

Moreover this is a network based detection method; it can be implemented either in server or router but not in client.

However this method can detect the botnet over the encrypted messages with moderate number of false positives. Therefore the behaviour based detection will yield a better result if combined with other detection methods.

3.3. Nick name based detection

Nick name based C&C channel detection is introduced in [4]. As it is not allowed to have same nicknames within the network to avoid naming collision [13, 16] each bot has to JOIN the IRC network with different names. A common method used by bots to avoid naming collision is to concatenate a certain word or phrase with a random number and this is where this method capitalizes. For instance sometimes bots assigns nick name in a standard and predictable manner such as [country name| random number] (e.g. USA|221038). In most of the cases the Nick names are generated with the constant part and the random part.

For instance constant part holds certain information about the kind of bot, the kind of operating system running on the bot, or the location of the contaminated machine. Random part is mostly composed with the set of integers.

Drawback

However the Botmaster can evade this kind of detection with simple source modification which uses different naming strategy and changes nick names time to time. In future all the botnets are expected to have the encrypted communication. In such cases this detection technique becomes ineffective.

3.4 Flow based detection

Flow characteristics like packets per flow (ppf) and average, bytes per packet (bpp), bytes per second (bps), packets per second (pps) have been used in segregating the botnet traffic from the TCP traffic [1, 19]. However these parameters can only help in segregating aggressive flow. Modern attackers keep the rate as low as possible to masquerade the attack flow as normal.

Hence to segregate the low rate attack flow, the flow per IP address is correlated to find out the similar behavior among the flows.

Drawback

However one cannot completely rely on this parameter for detection because the genuine IRC packets can also be circulated with similar flow characteristics. Moreover the correlation analysis is passive and it fails to initiate the timely action against the attack flow.

Other segregation methods like *Port based detection,* is inept nowadays because the ports (source & destination) which are used on IRC can be freely changed [9].

4. PROPOSED C&C DETECTION

The main design issue of botnet is how the bots receive the commands from the Botmaster. In IRC based botnets, the Botmaster communicates the bots through private messages (messages visible to only the channel members). Once the channel has been established by the Botmaster, he only needs to type in the command for the bots listening in that channel, following the previously defined syntax. This therefore will follow a different format than the genuine IRC messages, which can be differentiated as C&C Instruction.

More Importantly IRC Botmaster communicates the bots using IRC channel topics and through dispersing the PRIVMSG messages in the channel [12]. Botmaster will embed the C&C Instructions as PRIVMSG. Therefore monitoring the PRIVMSG will offer a platform to detect botnet as follows.

4.1. Bot Command (C&C) based detection

The PRIVMSG from the Botmaster is usually composed of C&C Command and supplementary Information. The C&C Instruction format therefore is **[Command | Information]**.

Anyhow the Information portion of the bot instruction contains parameters like IP addresses, Port Numbers, URL, options etc. which indeed is the user defined parameter and varies from instruction to instruction but the bot command is predefined. Therefore altering the command lends to misinterpretation by bots.

The command part mostly comprises 1) Main command 2) Sub command and normally with **(.)** as a delimiter or separator between them and rarely uses **(-)** as the delimiter. The delimiter marks the end of main command and the start of the sub command. Table1 presents a set of bot commands.

Table 1. IRC based bot commands

Command	Description
bot.dns	Resolves an IP/hostname
bot.execute	Runs *.exe* file on remote computer
bot.about	About the existing bots
bot.open	Opens a file on remote computer
bot.command	Runs a command with system()
ddos.udpflood	Starts a UDP flood
ddos.synflood	Starts a SYN flood
ddos.pingflood	Starts a Ping flood
ddos.phaticmp	Starts a PHAT ICMP flood
pctrl.list	List of processes
pctrl.kill	Kills the process

The bot commands generally have the following format.

[Main_Command | . - | Sub_Command]

Hence the detection procedure for IRC based C&C channel through PRIVMSG is obvious through extracting and comparing the bot commands.

4.2. Detection using Aho-Corasick NFA

Aho-Corasick NFA is used to design lightweight signature based detection, which will perform botnet detection online and in real-time.

Botnet detection through bot commands reduces the size of the signature which indeed reduces the storage requirement. At the same time, instant search for bot commands against the arriving IRC packet is essential and it is crucial to process the IRC packets at high rate to make the detection work online. Moreover the database should be dynamic and be able to accommodate any number of bot commands instantaneously which demands augmentable storage and the scalable method to bring the updated entities in to effect.

For this purpose the Aho-Corasick algorithm [10] is employed because it is a multi-pattern matching algorithm and it is scalable too [15]. Given a set of bot commands, as the pattern to search for, in the arriving IRC packets, the algorithm constructs a non-deterministic finite automation (NFA), which is employed to match all patterns at once.

The following is a portion of constructed NFA, due to the space constraint, not all the bot commands are included in it. However all the possible bot commands can be added to the NFA in the similar fashion, since the Aho-Corasick NFA is easy to construct.

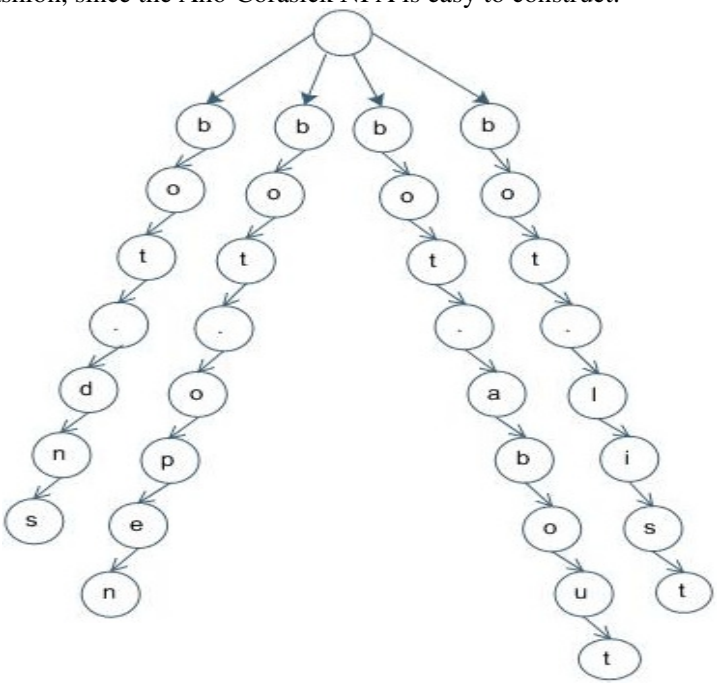


Figure 1. Aho-Corasick NFA.

The Aho-Corasick NFA as in Figure 1 is a forward NFA. If the detection engine traverses through, that marks the completion of proper detection.

The construction of NFAs must be done before the procedural invocation of Aho-Corasick NFA engine. The algorithm for Aho-Corasick is not discussed because it has been well studied over the period and has the rich collection of programming library.

However the papers [3][11] mention the existence of commands in the bot instructions but none of them develop the detection mechanism out of it. Reason may be, this kind of detection may fall short if the Botmaster uses different string combination for the bot commands, can only detect the bot commands that exist in the database whenever the new bot commands arrive there is no chance for detection. To overcome these constraints the principle of counting process as proposed in Section 5.3 can be handy.

4.3. Counting process to detect bot

At present, limited number of commands is used by the Botmaster. In future, Botmaster may try to derive new set of commands or alter the existing commands. In such cases lexical analysis like word frequency analysis can help to identify the existence of new bot commands which doesn't exist in the NFA. Once if the new C&C instruction is identified, they can be updated in the NFA for active attack detection.

The IRC messages have to be transmitted instantaneously because one actor doesn't afford to make another wait which may terminate their communication prematurely this is mostly the case with legitimate IRC chatting. Therefore the users truncate the words in such a way that it may convey the same meaning, which helps to improve the communication speed.

The bot commands are even shorter than the genuine chat messages. For instance, the C&C instruction is usually composed of bot command, IP address, port number etc, as per the pre-determined format.

Even though the chat messages and the bot commands are IRC messages they differ in the usage of words and framing of the sentence. This motivated us to perform lexical analysis to differentiate bot messages from the chat messages. Mostly used lexical analysis is word frequency [22]. For instance, word frequency helps in finding out which word is used mostly in the document, whether verb is used more than the adverb or not? Etc.

However word frequency is effective, only if the input is a book or big article, for a short message it is inconclusive. Because the integrity of the word frequency is determined by the sample size, therefore bigger the sample size higher the integrity will be [22]. The chat messages are normally a single sentence or few sentences with very less number of words which in no means suits the need for word frequency analysis.

Moreover, due to the smaller size of the chat messages, there will be considerable number of messages with not even a single words repeated twice. This even complicates the scenario of using word frequency for analyzing the IRC messages.

Even though the words may not be reiterated in the IRC messages, the words with same number of letters {1-letter words, 2-letter words, 3-letter words up to n-letter words} may be reiterated in the IRC chat messages.

Therefore instead of finding the word frequency, a method called as counting process is introduced and used. It is given as follows.

Count (i) = number of words with 'i' Character $\quad i = 1, 2, 3, 4, 5 \ldots n.$

Since the C&C message uses limited words, that to with special character and numbers. The words constituted with the alphabets alone are counted for every (i). Presume that the C&C instructions have low count and the genuine words have high count. The outcome of the counting process can be therefore deemed as follows.

Count (i) = {0, 1} then C&C instrn is + ive
Count (i) > 1 then C&C is – ive genuine chat
Where $1 < i < n$

Measuring counting process creates a chance for updating the new bot messages in the Aho-Corasick NFA.

5. OVERVIEW OF C&C DETECTION

Initially the bot commands have to be stored in the NFA. Once the bot commands are saved, the Aho-Corasick engine or detection engine should be started and to be applied right behind the IRC Capture, which segregates a range of TCP packets as possible IRC packets from other packets. The filtrate from the NFA engine is analyzed by the counting process for the possible new command.

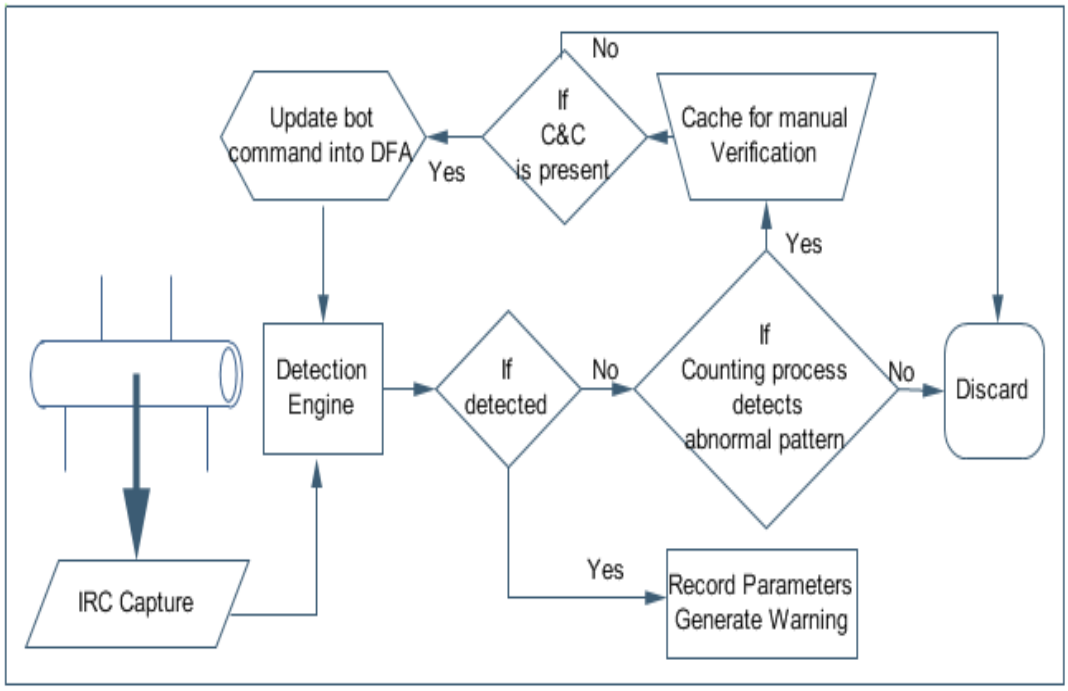


Figure 2. IRC C&C instruction based detection overview.

Figure 2 illustrates the logical structure of the proposed solution. Even though IRC Capture is a single unit but capturing IRC packets or segregating it from other TCP packets is not a simple task. Since the Protocol field of IP header helps in segregating TCP from other protocols, other than port number 6667 no other fields in TCP header reveals the presence of the IRC protocol. Early days the IRC Client uses port 6667 and 6668 no other application uses these ports, so there was a possibility to segregate the IRC traffic by looking into the ports. However, nowadays the IRC source port and destination port are altered randomly by the attackers to make it looks like a different application. This makes the segregation process even worse since relying on port 6667 to segregate IRC packets from other TCP packet will not be effective. Therefore a sequence of methods as defined in [1][7][21] is useful in segregating the IRC traffic from the other TCP packets with marginal deviation. They are,

1) Packet size greater than 300B can be exempted for IRC C&C instruction. Since the transmitted IRC commands are too small and can be contained in packet size less than 300B.
2) Packets with the TCP flags (ACK, FIN, SYN, RST) can be exempted, since they do not carry real data.

This will improve the performance of detection; by easing out the job of looking into every TCP packets for possible IRC C&C Instruction.

Once the segregation is done then the content of the incoming and outgoing IRC packets are analyzed using Aho-Corasick NFA engine. If the detection is positive, then the result is stored for generating the warning to the bots.

However if the NFA engine doesn't detect the presence of the bot commands, then the counting process will be applied to that packet and if the counting process result exhibits the presence of C&C instruction, then the particular message is stored in the temporary storage for passive manual verification.

Manual verification like checking for the information part of the C&C instruction, nick names etc. will helps to identify the existence of the bot instruction. Thus creates a chance for updating the suspicious bot commands in the NFA without being communicated to the signature providers which is the time sensitive process needs timely update to detect new attacks and it is time consuming too.

6. EXPERIMENTAL RESULT ANALYSIS

The implementation of IRC based malicious botnet detection is done using SNORT and Java. Snort is used because it is the most widely deployed intrusion detection and prevention technology worldwide, and has become the de facto standard for the industry. The snort rule is written to capture the IRC packets based on the IRC packet behaviour.

Snort Rule: alert tcp any any <> $HOME_NET any (msg:"TCPPACKET"; dsize:300<>40; flag:!FSR sid:1000989; rev:1;)

This rule captures the TCP Packets. Snort has many configuration variables and options, but the two most important ones are $HOME_NET and $EXTERNAL_NET. $HOME_NET is a variable that defines the network or networks to be protected, while $EXTERNAL_NET is the external, untrustworthy networks to which you are connected. These variables are used in virtually all rules to specify criteria for the source and destination of a packet. The default of both variables is "any," which allows monitoring of all the incoming and outgoing packets. Therefore choosing $HOME_NET enables monitoring packets from any network to home network and vice versa. The packet size less than 300 bytes and greater than 40 bytes are only monitored, because this is the range where the bot commands fall. Through traffic analysis we found out that the packet size less than or equal to 40 bytes can be refrained for IRC bot command based detection because it only have the TCP header and no pay load, which means it conveys the status of the communication with any of the flags FIN, SYN, RST, ACK set. The SID is the signature id unique for this rule and rev is the revision number.

Once the IRC packets are segregated then the Aho-Corasick NFA which is programmed by java is applied. If the packet is found to carry a bot command, then the corresponding source and destination IP's and port numbers will be recorded for generating warning. If Aho-Corasick NFA detects nothing, then the counting process is applied to the packet to confirm the genuineness of the packet.

The attack is simulated in the SSE lab and the detection engine found very successful in detecting them. To perform large scale analysis in real world, the raw data is accumulated from three different dataset providers. They are EFNET as dataset-1, QGIS as dataset-2 and Eris Free as dataset-3. Those datasets contains genuine traffic as well as the botnet traffic.

6.1. Real-time Simulation & detection result analysis

The NFA based IRC C&C detection has been examined in this section. The detection results of Aho-Corasick NFA against IRC messages are as follows.

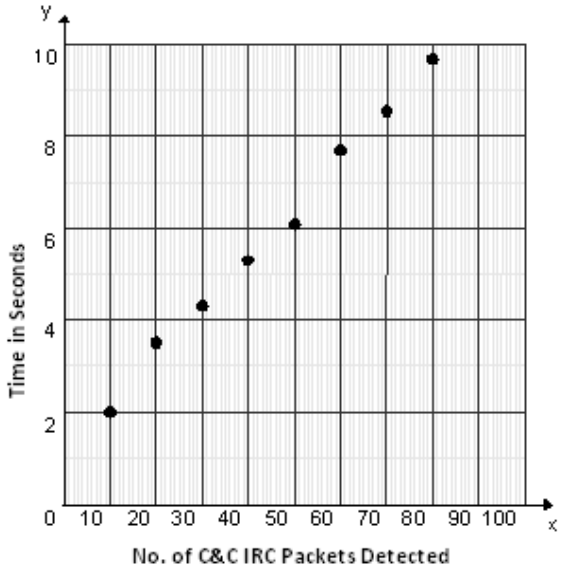


Figure 3. Real-time detection using Aho-Corasick NFA.

Figure 3 gives the number of real time detections completed by the detection engine. However the time taken in detecting the bot commands varies widely depending on the size of the words. The detection time is measured for patterns with varying length are analyzed.

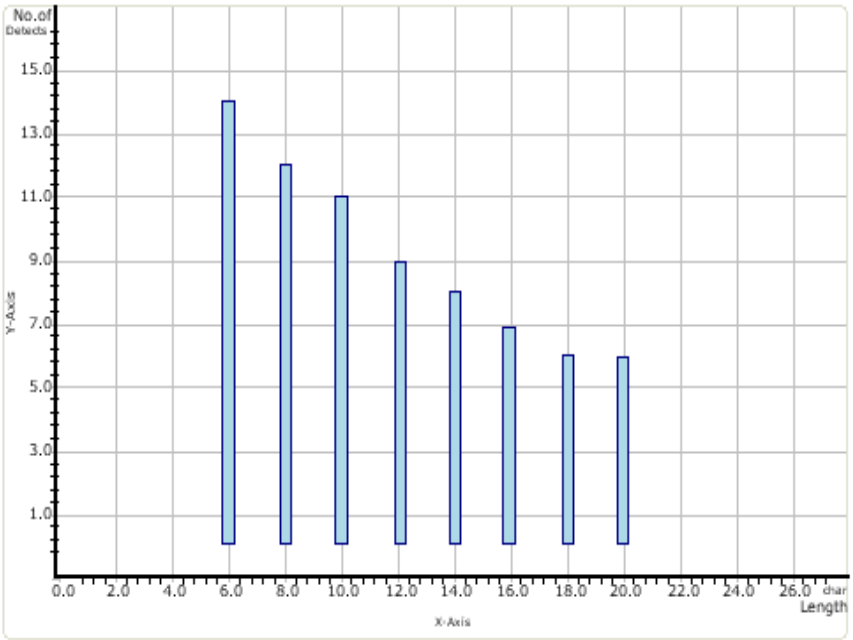


Figure 4. Detections per minute for various pattern length.

Figure 4 shows the influence of pattern length over the detection. Around 15 infected packets are generated for every 1 minute and the pattern length is increased after every minute, this had a effect on number of detection made. The result proves that, smaller the size of the bot commands results in quicker detection.

6.2 Examining the Real World Data

The dataset-1, dataset-2 and dataset-3 were examined individually using the counting process by incrementing the 'i' for every message. The max value 'n' for 'i' can only be fixed by analyzing the behaviour of chat room. Since apart from chitchat the professional chat room uses more complex terms where the word length may go up. In such cases 'n' may take the value up to 12. The test condition Count (i) ≤ 1 not only segregates the C&C packets but also a moderate number of genuine packets as C&C packets.

Table 2. Result of segregation done by counting process

IRC Dataset	Overall percentage of IRC packets segregated		
	Genuine Message	Suspicious Messages	C&C Instruction
Dataset-1	72.45%	25.2%	2.35%
Dataset-2	70.54%	26.45%	3.01%
Dataset-3	69.72%	25.2%	5.08%

Table2 shows moderate number of suspicious messages which are mostly genuine messages. The chat messages which carry URLs, messages without repeated words etc are all segregated. For instance "I am going home". Through observing suspicious messages manually, additional conditions are added along with the counting process, to shun the presence of the genuine chat messages. They are

If the message has only the words then it is not a bot command because bot command carries delimiter and numeral values in additional information. E.g. Hello, hai, hi etc are single word message sent without any punctuations, moreover chatting introduces new way of writing e.g. 'I m fine' instead of writing I am fine. 'ciao' instead of see you etc.

If the message has only numbers then it can be considered genuine messages. Message with more than four words can be eliminated from the suspicion. Applying all this conditions improved the performance as shown in the following Table 3.

Table 3. Resultant segregation

IRC Dataset	Overall percentage of IRC packets segregated		
	Genuine Message	Suspicious Messages	C&C Instruction
Dataset-1	92.45%	5.2%	2.35%
Dataset-2	90.54%	6.45%	3.01%
Dataset-3	89.72%	5.2%	5.08%

However the test condition Count(i) > 1 is successful in eliminating the range of genuine messages from suspicion as shown in Table4.

Table 4. Result obtained by varying i

IRC	Percentage of genuine IRC messages segregated from the suspicion for various ' i'							Messages with no symbol
	1	2	3	4	5	6	7	
	Overall (%) Segregated							
Dataset-1	5	12	50	27	20	10	9	20.4%
Dataset-2	7	7	52	21	18	13	10	12.2%
Dataset-3	4	17	50	30	15	9.	11	16.7%

7. COMPARISON RESULTS

Few other botnet detection techniques has been applied to the dataset and the results has been analyzed.

Table 5. Result obtained by applying various detections

IRC Dataset	Size MB	IRC Packets	Bot command based		Nick Name based detection		Behaviour Based detection		Flow based detection	
			Detect	FP %	Detect	FP %	Detect	FP %	Detect	FP %
Dataset-1	20	48,832	22210	0	17121	6.28	25832	4.63	23632	13.87
Dataset-2	15	35,878	19790	0	15023	5.52	23790	3.75	21075	12.37
Dataset-3	12	28,450	11124	0	9197	4.91	18116	3.42	12624	11.5

Table 5 records the number of detection made on three datasets using different detection techniques.

The method applied for bot command based detection is the Aho-Corasick NFA engine with the set of bot commands as in Table 1 and the counting process. Set of Nick names and possible nicks as discussed in [4] are applied for Nick name based detection, This detection is quick because it can detects the botnet activity in the 'JOIN' message itself, which is the inception stage of botnet formation. For behaviour based detection seven successive communication patterns between the Botmaster and the bot is identified and the inter-arrival time is calculated and stored in temporary database. Then the comparison is made with the other communication patterns. The packet size less than 100 bytes is accounted as bot command for flow characteristics based detection. Then the five successive one way communication with packet size less than 100 B is marked as detection, because the Botmaster issued C&C Instruction mostly framed with packet size less than 100 B but the bots sends the output that may sometimes go beyond the packet size 100 bytes. However the bot command based detection stands out without being produced any false positives.

8. CONCLUSION

In this paper the background information about C&C channels and its communication pattern over IRC for various botnets are comprehended. It also presents an elaborate study over various detection techniques and their limitations. As the result, we were able to device a brand new detection strategy which makes use of Aho-Corasick NFA algorithm to detect C&C instruction over the IRC channels, because C&C instruction is the definite trait of botnet. In addition to that, a counting process based verification technique is proposed to reduce the false positives and to help updating the new detection strings. An effective detection engine is emerged through the realization of both detection and update procedure. Finally various detection techniques and the proposed detection engine are evaluated through using them against the real-world IRC C&C traces. The results show that the proposed lightweight detection procedure is way ahead of other detection methods in performance.

REFERENCES

[1] Robert Walsh, David Lapsley, and W. Timothy Strayer. 2009. Effective Flow Filtering for Botnet Search Space Reduction. Cybersecurity Applications & Technology Conference for Homeland Security, pp.141-149.

[2] Ping Wang, Sherri Sparks and Cliff C. Zou. 2010. An Advanced Hybrid Peer-to-Peer Botnet. IEEE Transaction on Dependable and Secure Computing, pp. 113-127.

[3] Paul Barford and Mike Blodgett. 2007. Toward Botnet Mesocosms. First workshop on Hot topics in Understanding Botnet.

[4] Jan Goebel and Thorsten Holz. 2007. Rishi: Identify Bot Contaminated Hosts by IRC Nickname Evaluation. USENIX workshop on Hot topics in Understanding Botnets.

[5] Elizabeth Stinson and John C. Mitchell. 2008. Towards Systematic Evaluation of the Evadability of Bot/Botnet Detection Methods. Proceedings of the 2nd conference on USENIX Workshop on offensive technologies.

[6] Urjita Thakar, Nirmal Dagdee and Sudarshan Varma. 2010. Pattern Analysis and Signature Extraction for Intrusion Attacks on Web Services. International Journal of Network Security & Its Applications (IJNSA), pp.190-205.

[7] Akshay Dua James and R. Binkley Suresh Singh. 2009. Finding IRC-like Meshes Sans Layer 7 Payloads. *PSU TR*.

[8] Ping Wang Sparks and S. Zou, C.C. 2010. An Advanced Hybrid Peer-to-Peer Botnet. IEEE Transaction on Dependable and Secure Computing, pp. 113 - 127.

[9] Jianwei Zhuge, Thorsten Holz, Xinhui Han1, Jinpeng Guo and Wei Zou. 2007. Characterizing the IRC-based Botnet Phenomenon. Technical Report, University of Mannheim.

[10] Xinyan Zha Sahni, S. 2008. Highly Compressed Aho-Corasick Automata For Efficient Intrusion Detection. IEEE Symposium on Computer and Communication, pp. 298-303.

[11] Guofei Gu, Junjie Zhang, and Wenke Lee. 2008. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. Proceedings of the 15th Annual Network and Distributed System.

[12] J. Rayome, S. Romig. 1998. IRC on Your Dime? What You Really Need to Know About Internet Relay Chat. Technical Report, U.S. Department of Energy, Lawrence Livermore National Laboratory.

[13] Yuji Kugisaki, Yoshiaki KASAHARA, Yoshiaki HORI and Kouichi SAKURAI. 2007. Bot Detection based on Traffic Analysis. IEEE International Conference on Intelligent Pervasive Computing, pp: 303 – 306.

[14] Zhenhua Chi and Zixiang Zhao. 2007. Detecting and Blocking Malicious Traffic Caused by IRC Protocol Based Botnets. IFIP International Conference on Network and Parallel Computing Workshops. pp. 485-489.

[15] Yanbing Liu, Yifu Yang, Ping Liu and Jianlong Tan. 2009. A Table Compression Method for Extended Aho-Corasick Automaton. Lecture Notes in Computer Science, pp. 84-93.

[16] Jing Liu, Yang Xiao, Kaveh Ghaboosi, Hongmei Deng and Jingyuan Zhang. 2009. Botnet: classification, attacks, detection, tracing, and preventive measures. EURASIP Journal on Wireless Communications and Networking.

[17] Young June Pyun, Young Hee Park, Xinyuan Wang, Douglas S. Reeves and Peng Ning. 2007. Tracing Traffic through Intermediate Hosts that Repacketize Flows. IEEE International Conference on Computer Communications, pp: 634-642.

[18] Claus R. F. Overbeck. 2007. Botspy- Efficient Observation of Botnets. Hack.lu Security conference.

[19] Mohammed Misbahuddin, Sachin Narayanan and Bishwa Ranjan Ghosh. 2009. Dynamic IDP Signature Processing by fast eliminating using DFA. International Journal of Network Security & Its Applications (IJNSA), Vol 1, No 2.

[20] Andre Fucs, Augusto Paes de Barros and Victor Pereira. 2007. 'New Botnets trends and threats. Whitepaper.

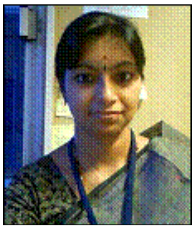
[21] Claudio Mazzariello. 2008. IRC traffic analysis for botnet detection, Proceedings of the Fourth International Conference on Information Assurance and Security, pp; 318-323.

[22] R. Harald Baayen, Counting process distribution, Kluwer Academic Publications.

Authors

J. Udhayan is graduated in Network and Internet Engineering from Karunya University, India in the year 2006. He is now pursuing PhD in Anna University Coimbatore, India. He is also working as Assistant Professor in Department of Information Technology, Karunya University, India. He has published papers in National/International Conferences and Journals. His area of interest includes Network Security, Computer Ethics, Grid Computing, Sensor Networks and Wireless Networks.

R. Anitha received the PhD Degree from Bharathiyar University, India in 1999. Presently she is working as Assistant Professor in Department of Mathematics and Computer Application, PSG College of Technology. She has published 9 papers in International journals and 2 papers in national journals, also having 20 years of teaching experience. Her area of specialization includes Queuing theory, graph theory, cryptography and network security.



T. Hamsapriya received the PhD Degree from Anna University, India. She is working as Professor & Head in Department of Information Technology, PSG College of Technology. She has published 12 papers in reputed International journals and 13 papers in International Conferences and more. She is having 15 years of teaching experience. Her area of specialisation includes Parallel and Distributed Computing, Networks, Information Systems, Database Technologies, Modelling and Simulation.
.