MasterBlaster: Identifying Influential Players in Botnet Transactions

Napoleon C. Paxton
College of Computing and Informatics
UNC Charlotte
Charlotte, NC 28223
ncpaxton@uncc.edu

Gail-Joon Ahn
School of Computing, Informatics
and Decision Systems Engineering
Arizona State University
Tempe, AZ 85281
gahn@asu.edu

Mohamed Shehab
College of Computing and Informatics
UNC Charlotte
Charlotte, NC 28223
mshehab@uncc.edu

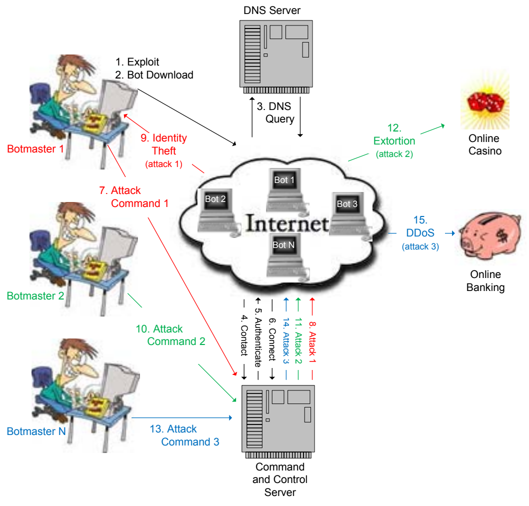
***Abstract*—Botnets continue to be a critical tool for hackers in exploiting vulnerabilities of systems and destructing computer networks. Botnet monitoring is a method used to study and identify malicious capabilities of a botnet, but current botnet monitoring projects mainly identify the magnitude of the botnet problem and tend to overt some fundamental problems, such as the diversified sources of the attacks. Most malicious botnets have the ability to be rented out to a broad range of potential customers, allowing each customer to launch different attacks from the other. Consequently, under the control of multiple botmasters, various attacks and transactions at different times attempt to damage networked infrastructures. In this paper we propose a multi-layered analysis system called *MasterBlaster* which identifies the communication characteristics of a botmaster in botnet transactions and correlates those characteristics with evolutionary changes within botnet communication channels. Our results show the level of involvement of the monitored botmasters within a botnet as well as their general motives. Our system clearly indicates that the investigation of each botmaster and analysis of botmaster interactions are essential to cope with net-centric attacks caused by botnets.**

***Keywords*-Botnet; Botnet Monitoring; Attribution; Botnet Analysis**

I. Introduction

Botnets are networks of compromised machines called bots that carry out the commands of botmasters through communication mediums–such as the Internet Relay Chat (IRC), P2P, social networks, and so on. Botnet monitoring has been an effective method to garner in-depth information about the threat of botnets. The idea behind botnet monitoring is to capture and modify a bot, allow the bot to connect to its command and control center while ensuring the modified bot will not be part of any subsequent attacks, and then monitor actual communications that take place on the botnet. This approach helps understand the magnitude of the botnet problem as a whole, but a fine-grained analysis technique with applicable protection mechanisms is still needed to defend against discovered botnet threats. In this paper we extend botnet monitoring techniques based on the interactions between botmasters and their botnets. Figure 1 shows the basic operation of an IRC botnet and addresses a deficiency present in previous publications which only show a botmaster sending commands to a botnet, giving the impression that there is only one botmaster in control of the botnet. In fact, most botnets are controlled by multiple botmasters. In Figure 1, we notice a botmaster 1 initially creating the botnet. Once the botnet is created, botmasters 1, 2, and N (which represent all other botmasters commanding the botnet) have their own attack agenda. Discovering these agendas and the roles played by each botmaster is the challenging research task.

Figure 1. Basic Botnet Operation



In this paper, our contributions are manifold as follows: first, within the monitored data we attribute each transaction to the botmaster and categorize the transactions based on a modified version of the reflective-impulsive model [1]. Our reasoning is, although a botnet is destructive, it is still just a tool, and a tool is only as useful as the way it is used with the intentions of the person who uses it. We categorize the botmaster interactions as social characteristics since there is a 2-way correspondence between the botmaster and the node in a botnet that responds to him. There are five categories of nodes in our system:

- *Botmaster node:* The entity that controls actions on the botnet.

- *Bot node:* The entity that carries out attacks and queries.
- *Compromised Machine node:* The machine that was originally attacked and turned into a bot node.
- *Storehouse node:* The node that provides a download service to the botmaster node or the bot node.
- *Victim node:* The node that is attacked.

Second, we identify the evolution of the physical characteristics (size) of a botnet. In most situations the size of a botnet determines the magnitude of the botnet's attack vector.[1] When it comes to size, botnets also have the same characteristics as other networks, like human social networks which are constantly in a state of flux. These communication networks are born, grow, shrink, and also disappear. We explore botnet evolution to track and garner physical characteristics from each evolutionary stage of a botnet. Third, we correlate the discovered social characteristics and the evolutionary characteristics to shed light on the role each botmaster plays in a botnet. To the best of our knowledge, this is the first attempt to identify the evolutionary characteristics of a botnet, and also to analyze a botnet based on its botmasters.

The rest of the paper is presented as follows. Section 2 discusses the scope of our research and Section 3 gives an overview of the system. In Section 4, we discuss our implementation details and our results. We overview the related work in Section 5. Section 6 gives a discussion on the current state of botnets and limitations of our work. The conclusions and future research directions are given in Section 7.

II. Scope of Research

Since botnets are normally massive in size, it has been relatively easy to covertly infiltrate a botnet and monitor its transactions. Because of this, botnet monitoring has become a common way to analyze and identify botnets and the destruction they cause. Most research goals in this area have been to identify the command and control of the botnet and shut it down (e.g., [10]), or to monitor the botnets for statistics without taking actions (e.g.,[5], [11]). In this paper we introduce the novel idea of monitoring botnet traffic to identify the roles each botmaster has in the botnet. This work builds on our earlier work in [4], [6], [7]. As mentioned earlier in the paper, the botnet is just a tool. The botmaster is the one that conducts the attack. When botnets are shut down, the botmasters need only to regroup and create another botnet such as the spam bot in [3]. Our goal is to discover motives and characteristics, which lead to discovering the root cause behind the botnet which is the botmaster.

[1]The compromised machine bandwidth is also a major factor of attack vector.

III. Master Blaster: System Overview

This section overviews the operation of system components of MasterBlaster as shown in Figure 2.

A. Bot Capture

In order to capture and analyze a bot we pretend to be a legitimate vulnerable machine that belongs to a network. Our bot capture component has three elements:

1) *Socket manager*: The attacker attempts to connect to a port through the socket manager.
2) *General shell code handler*: General shell code handlers are created to receive the data.
3) *Perl regex shell code handler*: General shell code handler passes the code to the Perl regex shell code handler to determine what type of code it is. After determining the type of code, the code is downloaded without executing it.

More details of the bot capture component could be available in [4].

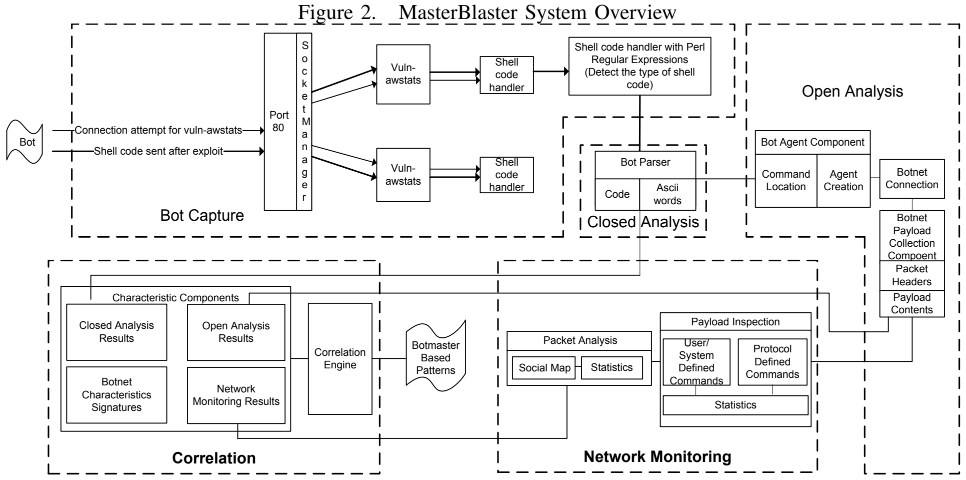
B. Closed Analysis

In [1], Strack et al. introduced the reflective-impulsive model that describes the social behavior of people. We adapt and modify their approach to botnets. We depict social behaior as a joint function of the two systems, *Reflective* and *Impulsive* systems denoted by the expression $S_R \Longleftrightarrow S_I$. The *Reflective System* is built by responses of knowledge on facts and their decisions denoted by the expression $S_R = set$ F that is composed of k-subsets: $\{f_{d_1}, f_{d_2}, ..., f_{d_{k-1}}, f_{d_k}\}$, which include a finite amount of facts f and their decisions d. In our closed analysis, we discover the ASCII text in the bot codes which are the reflective keywords. These keywords represent the facts. We derive the semantics of the facts from the command and control protocol. For this perspective, we use RFC 1459 and RFC 1812 since they define the IRC protocol. This helps us determine our protocol based keywords. All other keywords are user/system based. It is important to point out that bots on the same botnet normally have the same programmed capabilities and will react in the same way with the same command [8]. Therefore, keywords from one bot also represent keywords from the other bots in the botnet. To capture this, the parser identifies and stores all ASCII texts as keywords in the bot code. As [10] states, botmasters change commands using bot updates. Because of this, each update of the code is also parsed and changes are stored using this element. The decisions d of the facts largely depend on the Impulsive System S_I.

C. Open Analysis

Here we extract the general packet information from the botnet data. As stated in [13], botnet monitoring is always possible, since all information about the initial bootstrapping has to be included in the bot binary and thus can be cloned. This component has three elements that carry out the

Figure 2. MasterBlaster System Overview



analysis: bot agent, botnet connection, and botnet payload collection.

1) *bot agent*: The bot is stripped of its ability to attack victim machines.
2) *botnet connection*: The bot agent to connects to the command and control locations.
3) *botnet payload collection*: Captures all the readable contents of the payload.

More details of open analysis component could be available in [7].

D. Network Monitoring

In the Network Monitoring component we analyze the ASCII readable data in the payload discovered by the open analysis component and extract characteristic elements from the content of the data. The payload of each packet is inspected to discover conversations initiated by commands between the bot master node and the other nodes in the botnet. The structure of these conversations are discovered in commands based on the command and control protocol. In the case of this study we used the structure for `RFC 1459 and RFC 1812`. Within these conversations we discover the Impulsive System and the Evolutionary Characteristics.

1) Impulsive System: This is where we model the Impulsive System which integrates with the Reflective System discussed earlier to discover the social characteristics of the botnet. The Impulsive System S_I is built on associative links and motivational drives. To model these commands we use the following notation, $S_I \equiv S = m_1 \cup m_2 \cup m_3$, where S is the ground set of motivations based on 3 k-subsets of motivations M, *Destructive* (M_1), *Monetary* (M_2), and *Other* (M_3) and $m_i \subseteq M_i$. In our model, each command given by the botmaster is one impulsive human initiated command. Each subset (m_1, m_2, m_3) is composed of a set of commands. The associative links are the semantic connections of each command to another that meet a defined criteria for the subset. This means that each command that resides in a k-subset is linked to each other. In our framework, after the finite value of each k-subset is discovered, the upper-bound k-subset determines what the motivation of the botmaster is. Here we define the k-subsets that make up the motivational drives:

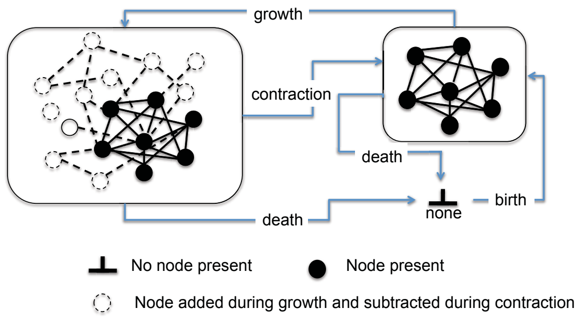
- *Destructive*: Concerned with causing damage that can physically affect potential victims' systems. This can also include extortion or other forceful ways to get money from potential victims.
- *Monetary*: Concerned only with covertly stealing money.
- *Other*: All unknown motives.

The operation of our reflective-impulsive process is as follows: once an impulsive command e in a set S is matched to a reflective keyword f in a set F then we determine two entities, e and f, to be one characteristic E which conjoins two systems, S_R and S_I.

2) Evolutionary Characteristics: In [12], Palla et al. introduce the concept of social networking evolution to study the constant evolution that happens in networks over time. It is important to note that in our work, evolution refers to the physical construction of the botnet based on nodes entering and leaving botnet channels. An evolutionary change is the addition or subtraction from the botnet by one of the botnet nodes. Each evolutionary change (JOIN or QUIT, or PART) is considered a characteristic. Figure 3 illustrates a life-cycle of botnet evolution. Each stage of evolution is defined as the following:

- *Birth*: The addition of a new botnet channel due to the creation of the first node(s).
- *Growth*: Node/s being added to a botnet channel after it is born.
- *Contraction*: Node/s being subtracted from a botnet channel after it is born. Other nodes still remain in a botnet channel.

Figure 3. Botnet evolution



- *Death*: All nodes subtracted from a botnet channel.

E. Correlation

The correlation component is where the results from the other components are combined and formed into patterns based on the botmaster. The output of this component is what allows us to discover what role each botmaster plays in the botnet. The three elements used are component correlation, botmaster characteristic statistics, and correlation engine.

1) *Component correlation*: Each result from the components has a timestamp. Using this timestamp and the botmaster name, the results of the components are correlated.
2) *Botmaster characteristic statistics*: The reflective-impulsive characteristics are analyzed to discover whether the characteristics are more protocol based or user/system based.
   - *Evolutionary characteristic statistics*: We use the auto-correlation function, $C(t)$, to discover the number of botnet nodes that consecutive timesteps t have in common:

$$C(t) \equiv \frac{|\beta(t_0) \cap \beta(t_0 + t)|}{|\beta(t_0) \cup \beta(t_0 + t)|} \tag{1}$$

   The number of botnet nodes that are present in both timesteps is $|\beta(t_0) \cap \beta(t_0+t)|$ and $|\beta(t_0) \cup \beta(t_0+t)|$ is the number of timesteps that have both botnet nodes in common.
   - *Reflective-impulsive characteristic statistics*: We discover the ratio of protocol based commands to user/system based commands to distinguish script generated commands from human generated commands which are more personalized.
3) *Correlation engine*: Correlates the results of the closed analysis component, the open analysis component, the network monitoring component, and the botnet characteristic component to discover the botmaster based patterns.

IV. Implementation and Results

A. Implementation

All the components are implemented on a Pentium III cpu with 3GB RAM and 360GB of disk space. The results from all the components are stored on a Dell PowerEdge 2900 server using the relational database. We built our bot capture component on top of the Nepenthes platform [14] which is housed on a Debian OS Virtual Machine (VM). The closed analysis component is also implemented on the VM which houses the bot capture component. The open analysis component is implemented on two VMs. One VM is the client running Windows XP operating system with a vulnerable awstats service. It receives the bot from bot collection and allows each bot to connect to its command and control center. The other VM is the server running Debian Linux operating system. It monitors activities on the client system and prevents it from participating in attacks using `iptable` firewall rules. The network monitoring component resides on a VM with Windows XP. It takes the packets discovered in the open analysis component as input. It consists of a Java program that parses through the packets and returns the social characteristics located in the payload as well as the header information in each packet. The correlation component resides on the same VM as the network monitoring component and consists of a Java program that correlates the results from all the components and separates them into patterns based on the botmaster.

B. Results

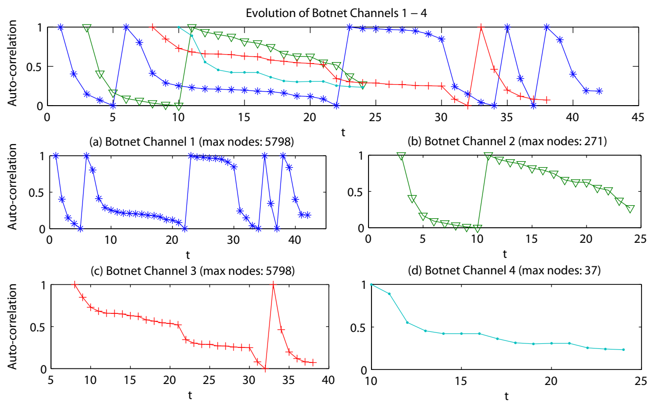
The results of bot capture and the input into our system was identified as an IRC shellbot written in Perl. Closed analysis was performed based on the amount of files downloaded from storehouse nodes. Changes in commands of the bots are discovered and updated with each bot code update. The following scripts in one version of the bot codes were identified by closed analysis:

```
if(/^\:$owner!.*\@.*PRIVMSG.*:!who(.*)/){
  print $sock "who ".$channel."\n";}

if(/^:.+?\s+352\s+\S+\s+\S+\s+(.+?)$/) {
  my $nicks = $1;
  #$nicks =~ s/\n//;
  #$nicks =~ s/\r//;
  push(@WHO, split(/ /,$nicks));
  print STDOUT "$who[1]\n";}

if(/^\:$owner!.*\@.*PRIVMSG.*:!dccflood(.*)/){
  for (1 .. 10) {
  print $sock "PRIVMSG ".$mescalina.": \001DCC
  CHAT chat 1121485131 1024\001\n";}

if(/^\:$owner!.*\@.*PRIVMSG.*:!hop (.*)/){
  print $sock "JOIN ".$1." : ".$2."\n";
  for (1 .. 10) {
  print $sock "PART ".$1." : ".$2."\n";
  print $sock "JOIN ".$1." : ".$2."\n";}
```

Table I
TOP 10 COMMANDS AND RATIO

Botmaster	Commands	Protocol:User/System
Botmaster 1	65535	18702:46833
Botmaster 2	2836	695:2141
Botmaster 3	1672	395:1277
Botmaster 4	1082	109:973
Botmaster 5	216	194:16
Botmaster 6	100	84:16
Botmaster 7	48	48:0
Botmaster 8	48	48:0
Botmaster 9	48	48:0
Botmaster 10	40	40:0

Figure 4. Auto-Correlation Statistics



These graphs show the auto-correlation of the nodes over the timesteps in the botnet
(a) Bot channel 1 is the largest of the botnet channels and is also the most dynamic. The auto-correlation value decayed completely 4 times, representing 4 complete recycle periods of botnet nodes.
(b) Bot channel 2 recycled once and had a steady decay. This is most likely due to it being small in size.
(c) Bot channel 3 decayed slowly until timestep 30. During this time the botnet grew tremondously in size do to bot recruitment. It became more dynamic after it grew in size.
(d) Bot channel 4 decayed the slowest of all. This is most likely do to it being the smallest

Reflective keywords extracted from these results are `PRIVMSG` which is found in line 123, 133, 135, and 138. Also, or `dccflood` was found in line 133. There were a total of 23124 unique reflective keywords discovered in the captured bot nodes. In our open analysis component, we discovered ∼5 GB of packets over one month. Each packet was stored as a row in a database with the sections of the packet as the fields. Using our framework, we were able to identify 558 botmasters completing transactions on the botnet. Table 1 shows the number of impulsive commands generated by the top 10 botmasters.

Now we present the results of our correlation as a whole. First, we discovered the impulsive command ratio shown in Table 1. Our results show that active botmasters (botmasters with a larger number of commands generated) generated more human user/system commands while botmasters that were not very active generated more protocol based commands. This is a key finding which means most of the impulsive commands generated by the active botmasters are human based and therefore are more apt to reflect the true intentions of the botmaster. Figure 4 shows the auto correlation results for 4 significant botnet channels. Here we divide the temporal periods of the botnet into 50 timesteps. The auto correlation discovered the amount of decay of the botnet nodes in the channels over the timesteps. Whenever the auto correlation value decreases from 1 to 0, the original bots have left the channel. We call this event a recycle of bots. As shown in Figure 4 larger channels decayed more rapidly. This was the case in both botnets.

This phenomenon has also been observed in other types of networks [12], which opens up the door for future work in modeling botnets using social networking techniques.

In Table II we show the patterns that result from the correlation of one represented botmaster in each tier. As shown, masters are often present on multiple channels within the botnet. Each botmaster is grouped into tiers based on their level of interaction with the botnet as a whole. We discovered that more active botmasters had a higher ratio of human initated elements to protocol based elements. This is very important since it means the botmaster is using his own intuitions in this channel and most of the transactions are not by scripts. As discussed in [15], this is important since human error continues to be the best way to catch botmasters or malware writers in general. Another important finding shows that each attack by a botmaster is performed in the growing stage of evolution and at the time when the size of the botnet is near maximum capacity in each case of attack.

V. RELATED WORK

In this section we discuss other research approaches that are similar to our work with respect to botnet monitoring and defense. To the best of our knowledge we are the first to analyze botnet traffic based on the botmaster and to incorporate evolutionary factors in botnet analysis.

Multifaceted approach to understanding the botnet phenomenon [5]: In this work the authors introduce their system which monitored botnets on a wide scale on the Internet. During their study they tracked 192 unique botnets of a multitude of sizes and show that botnets represent 27% of the Internet's unwanted data which is now much higher [3]. Their approach to monitoring the data is similar to ours in that they have a system setup that gets infected and securely connects back to its command and control and is then under the control of the botmaster while being monitored by the security professional. In this work, the authors focus on the identification of the widescale problems botnet present, but do not provide a fine-grained analysis approach as mentioned in our work.

Table II
CORRELATED PATTERNS

Name	Botnet	Msg No.	Attacks	Evolution	User/System	Protocol	Motivation
Tier 1							
Master 1	Channel 1	300	DDoS: 56	Grow: 500	500	320	Destruction
	Channel 2	1434	DDoS: 228	Grow: 5000	5274	1475	Destruction
	Channel 2	697	Identity Theft: 30	Grow: 5092	2379	1052	Monetary
	Channel 3	35	None	None: 253	60	22	Other
Tier 2							
Master 2	Channel 1	734	DDoS: 120	Grow: 493	2393	1299	Destruction
	Channel 2	245	Identity Theft: 13	Grow: 5121	432	211	Monetary
	Channel 3	12	None	None: 255	25	13	Other
Tier 3							
Master 9	Channel 16	8	None	None: 254	0	8	Other
	Channel 19	5	None	None: 152	1	12	Other
	Channel 20	5	None	None: 171	1	12	Other
	Channel 22	8	None	None: 75	1	12	Other
	Channel 23	5	None	None: 172	1	12	Other
Tier 4							
Master 245	Channel 34	0	None	None: 228	25	0	Other
	Channel 35	0	None	None: 54	45	0	Other
	Channel 36	0	None	None: 119	35	0	Other

Bot countermeasures [16]: In this work the authors attempt to identify and disrupt bot-like activity in a production system. Their method is to setup a system that includes a honeypot on the DMZ of a network. When the firewall of the network detects IRC botnet communications, a rule is triggered to redirect the traffic to the honeypot to interact with the botnet. The authors then use the information discovered to learn the command structure and submit commands to disable or uninstall the bots from the botnet. Their monitoring of a botnet is much like our botnet monitoring approach. As mentioned in [3], shutting down the botnet only forces the botmasters to recruit more bots and continue botnet activities elsewhere. Our approach focuses on identifying the patterns of the botmasters toward the attack attribution.

Proactive botnet countermeasures: an offensive approach [13]: In this work the authors discuss proactive ways to defend against botnets. They categorize their approaches in three areas; addressing, command, and exploitation layers. In each area they describe ways to connect to and shutdown the botnet. The case studies presented in this work display the effectiveness of the approaches, but as mentioned before our approach is focused on monitoring the botnet traffic to identify the role each botmaster plays so we can eventually stop those who are responsible for using the botnet.

VI. DISCUSSION

Botnets vary greatly in terms of command and control and the level of sophistication. In this section we discuss the current state of botnets and some limitations of our work.

A. Current state of botnets

Although IRC based botnets are still destructive and still need to be researched, botnets with other forms of command and control are becoming more prevalent and destructive. Http-based, P2P-based, and hybrid botnets are more difficult to defend. This is due mainly to the flexibility the botmaster has in controlling the botnet. In IRC based botnets, the command and control represents a single point of failure meaning if the command and control is shut down the botnet is also inactive. In the newer forms of botnets such as P2P, any of the nodes in the botnet can become the command and control. In this case if the command and control gets shutdown, one of the other nodes can now act as the command and control. The command and control protocol should not have an effect on our method, since the initial bootstrapping of any botnet node must be present in the binary so a clone can be made to join the botnet and monitor its activity [13]. We leave the monitoring of more advanced command and control protocols for the future work.

B. Limitations

A key limitation of our work is we can only identify the botmaster characteristics of transactions that have been decrypted. We are aware that many botnets now encrypt their malware and communications between them and their command and control servers, but there are many research projects aimed at decrypting payloads and malware data. In the future we may incorporate some of these techniques such as doing a memory dump of an encrypted bot as addressed in [9], but currently we are concentrating on the analysis of the data after it is decrypted, so encrypted payloads are beyond the scope of this paper. Our method of identifying the commands is based on the interpretation of the protocol based on RFCs and known system commands. Currently we have to manually identify how the commands are structured so that we can automatically extract the characteristics which is another limitation of our work. At the moment we are analyzing only IRC based botnet monitoring data and as shown with the recent DDoS attack on Twitter [18], IRC based command and control botnets are still very dangerous threats to the Internet. Hence, it is inevitable to continuously monitor the relevant activities.

VII. Conclusions

The real threats from botnets come from the botmasters that use them. In our research we addressed this critical issue by monitoring the traffic between the botmasters and the botnet and identifying the roles each botmaster played in the botnet. Since multiple botmasters are found on each separate botnet channel, automatically discovering the role each botmaster plays significantly helps reduce analysis time and provide a list of suspects to aid in the attribution of attacks. Our approach also enabled us to identify the generalized motives for each botmaster as well as the level of involvement each botmaster has on each botnet channel. Our results indicated that the physical construction of the botnet has a significant effect on botnet transactions since most attacks occurred during times where the botnet was at its largest size. We believe our work is an important and encouraging start to eventually identifying comprehensive patterns created by the transactions of botmasters.

Our future work would focus on other forms of botnets such as http-based, P2P-based, and hybrid attacks. In addition to automatic identification of command structures, it is crucial to create a taxonomy of botmaster patterns and corresponding botnet threat level alerts that will help identify the specific threats each botmaster poses within a monitored botnet. We will also integrate static malware analysis into the closed analysis component to identify a higher level of granularity in our reflective keyword discovery.

Acknowledgements

This work was supported, in part, by funds provided by National Institute of Justice.

References

[1] F. Strack and R. Deutsch. Reflective and impulsive determinants of social behavior. *In Proceedings of Person. Soc. Psychol. Rev 8*, 8:220–247, March 2004.

[2] IC3. IC3 Report. Available at http://www.ic3.gov/IC3Report.pdf, March 2010.

[3] m86security. Security labs report. Available at http://www.m86security.com, January 2010.

[4] N. Paxton, G-J. Ahn, B. Chu. Towards practical framework for collecting and analyzing network-centric attacks. In *IRI '07: Proceedings of the IEEE International Conference on Information Reuse and Integration*, pages 73 - 78, Las Vegas, NV, USA, 2007. IEEE.

[5] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis. A multifaceted approach to understanding the botnet phenomenon. In *IMC '06: Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pages 41–52, New York, NY, USA, 2006. ACM.

[6] N. Paxton, G-J. Ahn, R. Kelly, K. Pearson, B. Chu. Collecting and analyzing bots in a systematic honeynet-based testbed environment. In *CISSE '07: Proceedings of the 11th Colloquium for Information Systems Security Education*, 2007.

[7] G-J. Ahn, N. Paxton, K. Pearson. Understanding irc bot behaviors in network-centric attack detection and prevention framework. In *ICIW '08: Proceedings of the 3rd International Conference on i-Warefare & Security*, 2008.

[8] G. Gu, V. Yegneswaran, P. Porras, J. Stoll, and W. Lee. Active botnet probing to identify obscure command and control channels. In *ACSAC 09: Proceedings of the 2009 Annual Computer Security Applications Conference*, pages 241–253, Washington, D.C., 2009. IEEE.

[9] Y. Park and D. S. Reeves. Identification of bot commands by run-time execution monitoring. In *ACSAC '09: Proceedings of the 2009 Annual Computer Security Applications Conference*, pages 321–330, Washington, D.C., 2009. IEEE.

[10] F. Freiling, T. Holz, and G. Wicherski. Botnet tracking: Exploring a root-cause methodology to prevent denial of service attacks. In *ESORICS'05: Proceedings of ESORICS 2005*. IEEE, 2005.

[11] honeynet. Know your enemy: Tracking botnets. Available at http://www.honeynet.org/papers/bots, August 2008.

[12] G. Palla, A.-L. Barabasi, and T. Vicsek. Quantifying social group evolution. *Nature*, 446:664–667, April 2007.

[13] F. Leder and T. Werner. Know your enemy: Containing conflicker. *The Honeynet Project*, April 2009.

[14] P. Baecher, M. Kotter, T. Holz, M. Dornseif, and F. Freiling. The nepenthes platform: An efficient approach to collect malware. In *RAID '06: Proceedings of the 9th International Symposium on Recent Advances in Intrusion Detection*, pages 41–52, 2006.

[15] J. Calvet, C. Davis, and B. Pierre-Marc. Malware authors don't learn, and that's good. In *MALWARE'09: Proceedings of the Fourth Annual Conference on Malicious and Unwanted Software*. IEEE, 2009.

[16] V. Thomas and N. Jyoti. Botnet countermeasures. In *Journal in Computer Virology*, 2007.

[17] L. Corrons. Mariposa botnet. Available at http://pandalabs.pandasecurity.com/, March 2010.

[18] J. Wortham and A. E. Kramer. Professor main target of assault on twitter. *The New York Times*, page B1, August 2009.

[19] N. J. Rubenking. New botnet may have infected half of fortune 1000. *PCMAG.COM*, August 2009.