Discovering and Analyzing Deviant Communities: Methods and Experiments

Napoleon C. Paxton*, Dae-il Jang**, Ira S. Moskowitz*, Gail-Joon Ahn**, Stephen Russell* and Myong Kang*

*Information Technology Division, Naval Research Laboratory, Washington, DC
**School of Computing, Informatics, Decision Systems Engineering, Arizona State University, Tempe, AZ

***Abstract*—Botnets continue to threaten the security landscape of computer networks worldwide. This is due in part to the time lag present between discovery of botnet traffic and identification of actionable intelligence derived from the traffic analysis. In this article we present a novel method to fill such a gap by segmenting botnet traffic into communities and identifying the category of each community member. This information can be used to identify attack members (bot nodes), command and control members (Command and Control nodes), botnet controller members (botmaster nodes), and victim members (victim nodes). All of which can be used immediately in forensics or in defense of future attacks.**

The true novelty of our approach is the segmentation of the malicous network data into relational communities and not just spacially based clusters. The relational nature of the communities allows us to discover the community roles without a deep analysis of the entire network. We discuss the feasibility and practicality of our method through experiments with real-world botnet traffic. Our experimental results show a high detection rate with a low false positive rate, which gives encouragement that our approach can be a valuable addition to a defense in depth strategy.

I. Introduction

There has been a significant amount of research devoted to discovering botnet traffic within computer networks, but an equally important area of research is the analysis of the data once it has been discovered. Security analysts in charge of decyphering information gathered after a botnet attack always begin at a disadvantage due to the everchanging landscape of botnet administration and the custom and decentralized methods used to analyze the data. Because of this, many botnets such as *Mariposa (7 months), Kelihos (8 months), and Rustock (5 years)*, can continue to operate for a significant amount of time after an attack has been discovered, while waiting for actionable information gathered from current analysis techniques [5]. This "wait" is significant because current analysis methods have to decipher the commands used to administer the botnets. Experts agree that this is time consuming and non-trivial even for experienced and skilled analysts [13], [4]. A preliminary analysis that will allow analysts to take immediate steps, *(such as identifying and blocking key actors in the botnet)*, can lessen the effects of continued operations until a more detailed analysis can be completed.

Malicious botnets, which are networks of compromised machines, continue to be among the top threats found on the Internet [16]. Attacks performed using botnets include: Distributed Denial of Service (DDoS), Identity Theft, Click Fraud, Phishing, Spam, and so on. Each type of attack can cause significant harm to their victim and consume considerable bandwidth of the networks they operate in. For example, in the case of spam, a recent security report from Trustwave found that 75.2% of all inbound emails are considered spam sent by botnets [12]. Additionally, ten percent of those spam emails contain malicious content, which will infect vulnerable machines of users who click on the email's embedded link.

Defense In Depth is a strategy used by nearly every network security professional to defend against threats on the Internet. This strategy involves multiple layers of protective solutions such as anti-virus, anti-spyware, firewalls, and intrusion detection/prevention systems. This strategy also includes layers of analysis methods which generally consist of custom tools that analyze malware and network traffic. Each layer works in concert to defend systems. Information discovered from the analysis layers are turned into signatures that are fed into the protection layers. Current botnet analysis techniques, which are based on the custom tools, are effective in discovering fine-grained details about botnets. However, due to the amount of time that takes place between protection layers and analysis layers, an additional analysis layer is still needed to reduce the effects of the botnet while the more conclusive analysis takes place.

Methods for finding communities have been studied extensively in a variety of networks including the Internet [6]. In each method, all networks are represented generically as graphs composed of vertices (nodes) and edges (links). The concept of communities does not have a widely accepted definition. For the purpose of this article communities are described as nodes in the network that communicate with each other through links more than they do with any other nodes in the same network. In previous research, discovering the communities and the relationships between

their nodes has revealed key facts about the purpose of the community creation. In this article we investigate how this method can be further extended to perform botnet analysis.

This article is organized as follows: the overview of botnet analysis approaches is described in Section II. Section III discuss our botnet community analysis model which is based on *k*-clique constructs. In Section IV, we discuss our tool and evaluation results on both IRC and HTTP botnets. To evaluate the effectiveness of our approach, we compare information discovered in our analysis with the results of a manual analysis of the same botnets. Our comparison shows community analysis methods can accurately uncover preliminary information with a low false positive rate. This information can be eventually used to reduce the time between discovery of botnet activity and identification of actionable intelligence. The future directions of our approach are elaborated in Section V. Section VI concludes the paper.

II. Current Botnet Analysis Methods

Botnets have the ability to conduct attacks or other malicious activities within minutes or even seconds. Current botnet analysis techniques designed to discover enough information to slow down or stop botnets are effective once the semantics of the data captured from the botnet can be discovered [11]. Unfortunately, the deciphering of the data normally takes days, and in some cases months, to complete.

Signature Based Analysis. Most botnet analysis techniques today require a manual pre-processing step, which involves reviewing the normal format of the command and control protocol being used, and comparing it with the format of the actual command structure found in the botnet data. This is a mandatory step because many botnet administrators modify their command and control protocols to evade detection or monitoring. As mentioned before, this step can be very time consuming depending on the sophistication of the protocol modification and the skill level of the analyst [4], [5]. The products of this type of analysis are signatures which are used to power mechanisms such as anti-virus programs, firewalls, and blacklists. Signatures block botnet traffic based on a defined ruleset. Again, the problem with this approach is the length of time involved in the signature discovery. In normal cases botnet administrators slightly change their malware to avoid a detection engine, this simple change places the requirement on the analyst to repeat the tedious signature based process each time a change occurs.

Anomaly Based Analysis. Anomaly based solutions such as sandnets perform their analysis based on patterns discovered during execution of the data [14]. Existing approaches such as botsniffer [9] and bothunter [8] require a malicious activity response module for each type of botnet action that is to be monitored. Today botnets have become sophisticated in disguising their activities so that it could make defining malicious activity response modules both difficult and impractical. Since our approach is based on physical structure and communication patterns, we do not require this step for botnet analysis.

Sinkholing. Sinkholing is the current method of choice for botnet analysis and defense [3]. In this approach, the analyst deceives bots into taking orders from the analyst instead of their normal command and control servers. Once this is done the analyst receives all traffic packets passed from the bots to the botnet. There are several drawbacks to sinkholing and shutting down botnets. The biggest issue is the complexity and time involved in conducting a sinkholing campaign. Normally, sinkholing involves a coordinated effort from the analyst, ISPs, and law enforcement officials. Another major concern is after a botnet is sinkholed, there is only a small window to conduct an analysis of the entire botnet. This is because the command and control server and the master of the botnet are no longer controlled by the deviant users whom the botnet used to belong to. Shutting down the botnet also has its drawbacks. Although the loss of the botnet is a setback for the administrators and botmasters that controlled the botnet since the perpetrators behind the botnet are normally not captured, they are free to regroup and create new and more resilient botnets.

Each of these methods are useful in a defense in depth approach over time, but new methods are needed to discover information about botnets earlier so more immediate actions can be taken to address the threats caused by the botnet activity.

III. Botnet Community Analysis

There exist several approaches in extracting and analyzing communities within the network data. Early algorithms, such as the approach by Borgatti et. al. [2] and the *k*-core algorithm [15] showed how communities could be extracted, but did not allow nodes to be part of more than one community within a network. Networks such as botnets do contain nodes that belong to multiple communities so these methods could not be easily adapted for our approach. CFinder is an algorithm that allows community members to overlap and it is the most studied method in the literature. It is based on *k*-clique percolation, which was introduced by Palla et. al. [10]. This algorithm builds communities based on *k*-cliques and has been proven effective in identifying the semantics of many network communities such as social networks and normal Internet traffic networks [6]. However, this approach is not sufficient for our purposes in its native form because it does not consider communication direction.

Attribute Selection. Analyzing botnet data using our method is dependent on selecting attributes for two elements, *(1) Discovering nodes* and *(2) Discovering links that connect the nodes*. Discovering these elements is dependent on the format of the available data, but not the structure of the command and control commands. For example, the most common data format collected for all protocols is packet capture data (PCAP) which is based on the packets created by the transport layer of the TCP/IP protocol stack

when messages are sent back and forth over the Internet. This means that if PCAP data is available for IRC, P2P, HTTP, and any other botnet administration protocol it is structured the same, the only preparation that needs to be made before analysis is the selection for the attributes to define the nodes and links.

Node attribute selection seeks to identify each member of a community that sends or receives a message. We selected *IP address* as our node attribute because each node is required to have an IP address assigned to it. A well known issue with IP addresses is that they tend to change frequently, which makes identifying nodes uniquely impossible when considering only the IP address. Our current research is only concerned with discovering the communities and the semantics of the communications between the nodes in each community. Hence, the uniqueness of each node is not required. We leave identifying unique nodes to future work.

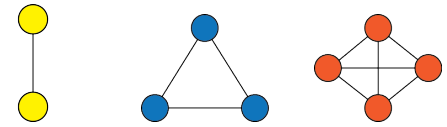
Link attribute selection in our approach is concerned with identifying identical links. Because of this, we selected *Payload Length* as our link. Payload length meets our requirement since periodically bot nodes in a botnet will either receive or send an identical message. Payload length is also readily available and useful regardless of the readability of the payload. For instance, if the payload messages of a botnet are encrypted, we will not be able to understand the communication across the network without a time consuming decryption step, but since many payload messages in a botnet tend to have the same content, these identical messages which are encrypted using the same algorithm will be the same size.

Node Category Discovery. To discover information that can aid in botnet defense, we place each node in a category based on their actions in the communities. These categories are: *Master Node*-which initiates all commands to the botnet, *Command and Control Node*-which acts as a proxy between the Master Node and the rest of the botnet, *Bot Node*-which carries out commands received from the Command and Control and returns responses periodically or when prompted, and finally the *Victim Node*-which is the target of attacks and does not send a response recorded in our data.

A. Botnet Community Model

We select *k*-clique constructs that are based on Palla et. al. [10], to build our communities. Our intuition behind this is two-fold: (1) our community identification is local, which means if a node or link outside of the community were to be removed, then the local community would not be effected. (2) It allows overlaps, which means a node can be part of more than one community at the same time. In clique percolation, there is a graph with nodes and links, (of weight 1 and non-directional), which represent a means of communication or contact without self-edges.

Definition 3.1: A **clique** is a set of nodes that have a link to every other node in the set.



(a) k=2 (b) k=3 (c) k=4

Fig. 1. Example k-cliques: In order to be a k-clique each node needs to have a connection to every other node in the clique. (a) Two fully connected nodes (b) Three fully connected nodes (c) Four fully connected nodes.

The k in k-clique represents the number of fully connected nodes in the clique. Fig. 1 displays three cliques with different k values. All cliques are equal to or are a subset of a maximal clique. In addition, there can be many cliques in a maximal clique, but a maximal clique cannot belong to another clique. Cliques *percolate* into each other by being *adjacent* and communities are built from adjacent cliques. We define several terms as follows:

Definition 3.2: The largest amount of fully connected nodes k found in a particular community is known as the **maximal clique**.

Definition 3.3: Two k-cliques are **adjacent** if they have $k-1$ nodes in common.

Definition 3.4: A **community** is a set of two or more adjacent cliques.

Note that others have relaxed the definition of adjacency by varying $k-1$ to $k-i$, and is dependent on the method being used [10]. An important element captured using the clique percolation method is community overlap.

Definition 3.5: **Community overlap** occurs when at least one node is part of multiple communities.

In order to identify communities using this method we first discover the maximal cliques in the botnet. This step takes into account the botnet as a whole and finds the maximal cliques built from fully connected nodes within the botnet. Each maximal clique now represents a node which will or will not be paired with other maximal cliques. To identify the links between cliques we create an adjacency matrix O with adjacent cliques i and j. In order to find all cliques of size k that are percolating into each other and forming communities, all values of O that are equal to or greater than k-1 are given the value 1 in the matrix and all other values of O are set to 0. After this process, each value of 1 in the matrix represents a community and the community overlap is discovered by identifying the number of vertices shared by clique (i and j).

Once communities are discovered, we integrate directional data based on the Source and Destination IPs. This step is straight forward and forms the basis of our analysis by identifying how nodes communicate amongst each other. Previous research has shown that bots make up at least 50% of the total nodes found in a botnet [9]. Using this key metric as a threshold, we discover the category of each node. Our **node categorization algorithm** performs as follows:

1. *Discovering Coordinated Activity*–Identify all mes-

sages sent to or received by at least 50% of the nodes in a community graph and this step terminates if a coordinated link is not discovered or no nodes in a current community have been previously categorized;

2a. *Identifying Command and Control Nodes*–Identify all nodes sending messages to or receiving and forwarding messages from coordinated nodes;

2b. *Discovering Victim Node*–Identify all nodes that receive messages from coordinated nodes and do not respond;

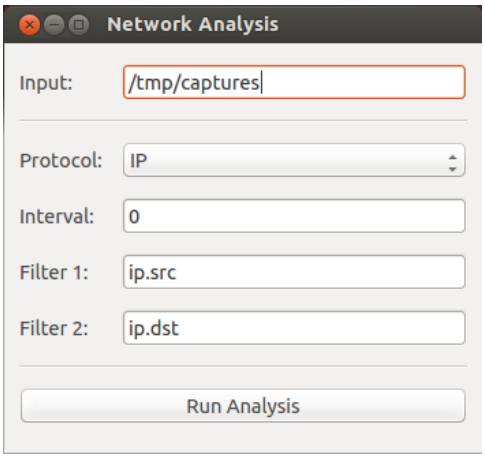
3. *Discovering Master Nodes*–Identify all nodes that sends/receives non coordinated messages to the command and control nodes without receiving a coordinated reply; and

4. *Discovering Bot Nodes*–Identify all nodes that are part of the coordinated group that sent or received coordinated messages.

A community graph is a set of all communities discovered in a temporal window. Each community graph represents a timestep in our botnet analysis. We have discovered that many botnets have session intervals of less than 10 minutes, for this reason we chose 10 minute time intervals for each community graph. This means that all nodes and links that are discovered within a 10 minute increment are part of the same community graph. We also consider adjacent community graphs to account for sessions that span across multiple community graphs. Assuming that we are analyzing the initial community, *step 1* of our algorithm checks each community in the graph for discovering identical links in one direction that reaches at least 50%. If the community does not have a coordinated group of at least 50% we do not consider it useful for botnet analysis. In step 2, we check to see if the discovered coordinated links have a one to many relationship. *Step 2* is split into *(a)* and *(b)* because at this point the single node that has been identified will be classified as a Command and Control Node if messages are being sent and received which is identified as *Step 2a*. If messages are only being received, the node will be classified as a Victim, which is identified in *Step 2b*. At this point, the command and control nodes have been identified. Nodes that send non-coordinated links to the command and control nodes are identified in *Step 3*, and are considered bot masters because no other node type recorded in the community will send non- coordinated messages without receiving a coordinated reply in the allocated time interval. *Step 4*, identifies the nodes that are 50% or over of members that sent or received a coordinated link. There has to be at least 2 nodes in this category.

IV. Botnet Community Analysis Evaluation

All of the components in our analysis are implemented on commodity hardware using Inter Core i7 and 8GB memory through Cygwin. This is important because no modification of the hardware is required for our method to operate. The data collection and analysis were conducted



(a) GUI

```
$ python3 analyze.py -h
usage: analyze.py [-h] [-1 FILTER1] [-2 FILTER2] [-p PROTOCOL] [-i INTERVAL]
                  target

PCAP Network Analysis

positional arguments:
  target       input file or folder

optional arguments:
  -h, --help   show this help message and exit
  -1 FILTER1   First Wireshark filter expression
  -2 FILTER2   Second Wireshark filter expression
  -p PROTOCOL  Filter analysis protocol
  -i INTERVAL  Maximum session interval time filter value
```

(b) CUI

Fig. 2. Current Proof of Concept Tool: (a) Graphical Interface and (b) Command-line

in Virtual Machines (VMs) designed to capture botnet traffic. Our datasets consist of PCAPs from the HTTP based BlackEnergy botnet and an unnamed IRC botnet.

Tools and Implementation. To conduct our analysis, we created tools based on the Python and Perl programming languages. We based our community extraction on the open source tool, CFinder[1], to discover communities. Fig. 2 shows the proof-of-concept prototype of our graphical and command line tools.

The front end of our tool is designed to provide options in choosing the node.

A. Analysis of an IRC botnet

First we demonstrate our approach by analyzing network traffic from a botnet controlled by the IRC protocol. As a reminder, the verticies (nodes) are identified by IP address and the edges (links) are identified by the message payload size in bytes. Example (21B is a message of size 21 bytes).

Community Graph Analysis (IRC): Figure 3 shows six randomly selected nodes from a community graph. The applied directional links show that nodes *(x.x.x.16, x.x.x.17, x.x.x.18, x.x.x.19, x.x.x.10)* all receive a (21B) request (shown in dotted lines) from *(node x.x.x.194)*. All nodes except for node *x.x.x.194* returned a (20B) response (shown in solid lines). Following *step 1* of our algorithm, nodes *x.x.x.10, x.x.x.16, x.x.x.17, x.x.x.18, and x.x.x.19* all received and sent a message at the same time and since this represents over 50% of the nodes found in the community graph this qualifies as a coordinated communication event. Based on *step 2a* of our algorithm, node *x.x.x.194* is a

COMMAND AND CONTROL node because it is the node that sent and received the coordinated message. In this community there were no nodes that received coordinated messages without returning a message *step 2b* and there were no nodes that sent non-coordinated messages to the command and control node *step 3*. All the other nodes were part of the group that made up the coordinated nodes which only sent and received coordinated messages. According to *step 4* of our algorithm, these nodes all belong to the *BOT* node category.

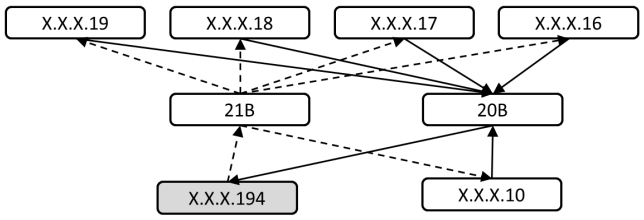


Fig. 3. Sample Community of IRC Botnet in a Community Graph: Command and control node *x.x.x.194* sends a link (21B) to all bots in the community. Bots respond to the command and control node with identical links of (20B).

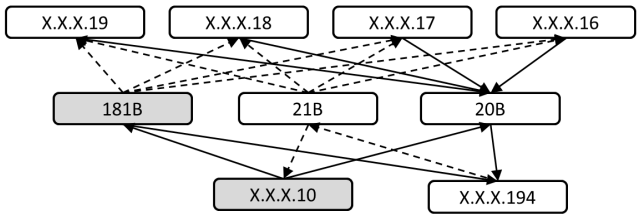
In Figure 4 we first see the (20B) and (21B) messages sent previously *step 1*. Node *x.x.x.194* still qualifies as a *COMMAND AND CONTROL* node based on *Step 2a*. Node *x.x.1.19* received coordinated messages but did not return any data, so based on *Step 2b* it fits in the *VICTIM* node category. Next we see node *x.x.x.10*, which was one of the nodes found in the bot category in the previous community graph and sent a non-coordinated message, (181B), to the command and control node *x.x.x.194*. According to *Step 3* this places the node in the *BOTMASTER* node category. In this new graph all the other nodes in the graph sent a message (9B) to the node *x.x.1.19*, which places it in the *VICTIM* category. The nodes that sent the coordinated message represent more than 50% of the nodes in the graph and they only sent coordinated messages so according to *Step 4* these nodes belong to the *BOT* category.

Manual Analysis (IRC): The IRC botnet we analyzed followed specifications from RFC 2812 and included all message types. Link (21B) translated to a PING message and link (20B) translated to a PONG message. PING/PONG messages are sent periodically to determine what nodes are considered available within the IRC network. In Figure 4(a), we can notice that the PING links (21B) were sent throughout the network to see what nodes were available and the PONG links (20B) gave the reply. In Figure 4(a), a node x.x.x.10 sent an attack command to node x.x.x.194 with the destination node set to ALL. Figure 4(b) shows the result of the command (ALL), where each of the bots in the botnet sent a (9B) message to the node x.x.x.19.

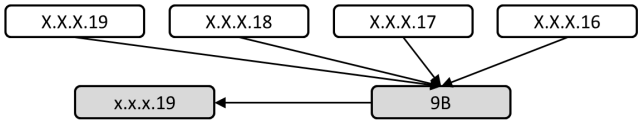
Summary of Overall IRC Analysis. The overall results discovered using our botnet community overlap method were similar to those discovered using the manual analysis. Bots were correctly classified (99%) of the time during the evaluation and Command and Control Nodes were fully identified. Botmaster Nodes were identified with a (67%) detection rate, but this result was a bit misled because 77 of those botmaster Nodes identified by manual analysis connected to a Command and Control Node once and did not commit any subsequent transactions. Victim Nodes had a success rate of (87%), but attack detection had a rate of (68%). The reason for the disparity here is many attacks only involved one or two bots in the botnet. These are reconnaissance attacks that are sent out before more large-scale coordinated attacks are conducted. Currently our approach does not capture these reconnaissance attacks in the communities. Overall our analysis result was meaningful and promising and it shows our method was conducted nearly three days faster than the manual analysis. Community overlap was relatively high in our study. This is interesting because normal Internet traffic is said to have a relatively low overlap rate.

Method	Time	Bots	C&C	botmasters	Victims	Attacks
Bot Com	30 min	4323	34	234	61	219
Manual	3 days	4310	34	351	70	321

TABLE I
IRC COMMUNITY METHOD RESULTS



(a) IRC Command Propagation Found in Community Graph 1



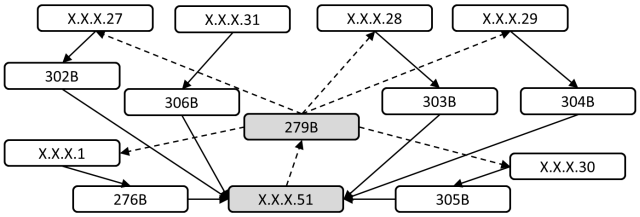
(b) IRC Command Execution Found in Community Graph 2

Fig. 4. Sample Result of Command Propagation and Attack Across Adjacent Community Graphs: (a) Botmaster node *x.x.x.10* sends a link (181B) to command and control node *x.x.x.194* to send a message to all bots in the botnet. (b) All the bots in the botnet send a (9B) attack to victim node *x.x.x.19*.

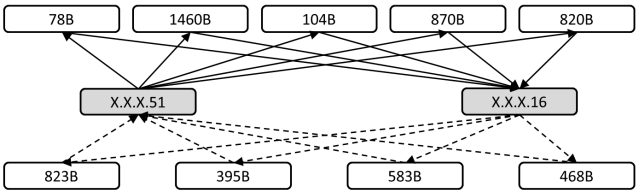
B. Analysis result of HTTP botnet

The HTTP botnet used in this article followed standard protocol procedures in HTTP RFC 2616 [7]. Figure 5 shows three partial communities across three community graphs that were constructed from randomly selected nodes.

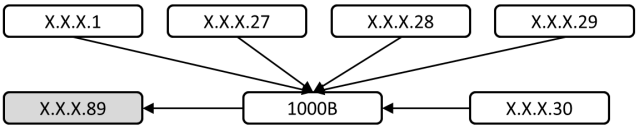
Community Graph Analysis (HTTP): Here we show an example of our analysis across three community graphs which are shown Figure 5. Figure 5(a) shows a segment of a community graph where a node *x.x.x.51* sent a coordinated link to all the other nodes in the community.



(a) HTTP Bots and Command and Control Server Interactions Found in Community Graph 1



(b) HTTP Botmaster and Command and Control Server Interactions Found in Community Graph 2



(c) HTTP Attack Discovered From Multiple Bots to a Victim Found in Community Graph 3

Fig. 5. Sample Result of Adjacent HTTP Botnet Community Graphs: (a) Command and control node *x.x.x.51* sends a coordinated link (279B) to all the bot nodes in the botnet. (b) Botmaster node *x.x.x.16* sends a series of links (823B, 395B, 583B, 468B) to command and control node *x.x.x.51*, which then sends a series of replies (78B, 1460B, 104B, 870B, 820B) back to the botmaster node.

Based on *step 1*, this qualifies as a coordinated event since more than 50% of the nodes received a link of the same size (279B). Based on *step 2(a)*, node *x.x.x.51* is a *COMMAND AND CONTROL* node because it sent the link. In this community graph there are no nodes that received coordinated messages without returning a response so *step 2(b)* does not apply and there were no nodes that sent a non-coordinated link without receiving a coordinated response, so no botmasters were discovered using *step 3*. Since all nodes *x.x.x.27, x.x.x.31, x.x.x.28, x.x.x.29, x.x.x.1, and x.x.x.30* received a coordinated link (279B), these nodes are classified as *BOT* nodes by *step 4* of our algorithm. Note that after all the bot nodes received a coordinated link they returned similar, but non-identical replies. These replies are updates and are currently used as metadata, but not to identify the category of the nodes.

In the community graph illustrated in Figure 5(b), a new node is discovered which sends a series of non-coordinated links to a node *x.x.x.51*. The node *x.x.x.51* was previously identified as a command and control node in an adjacent community graph. In this community graph *steps 1,2 (a), 2(b)*, and *4* were not utilized, because no coordinated links were observed, but since the node *x.x.x.51* was already identified in a previous graph, we were able to use its previous state and apply *step 3* which placed a new node, *x.x.x.16*, in the *BOTMASTER* node category because it sent non-coordinated links without receiving a coordinated response.

In the community graph shown in Figure 5(c), nodes that were previously identified as belonging in the bot category also sent a coordinated message to a new node *x.x.x.89*. Since the node *x.x.x.89* did not respond to the coordinated link, it is placed in the *VICTIM* category. Also, if the nodes that sent the coordinated message were not previously identified as bots, they would have been identified as bots in this graph because of the coordinated message that was detected.

Manual Analysis (HTTP). Manual analysis of the community graph in Figure 5(a) shows that nodes *x.x.x.27, x.x.x.31, x.x.x.28, x.x.x.29, x.x.x.1, and x.x.x.30* are bots sending requests to a command and control node *x.x.x.51* for obtaining command instructions. Figure 5(b) shows a series of communications between a botbotmaster *x.x.x.16* and a command and control server *x.x.x.51*. In Figure 5(c), bot nodes *x.x.x.1., x.x.x.27, x.x.x.28, x.x.x.29, and x.x.x.30* all sent an attack message to the victim node *x.x.x.89*. These results also are consistent with the analysis conducted using our community graph analysis.

Summary of Overall HTTP Analysis. The overall results of our method compared to the manual analysis for this botnet were nearly identical. Just like the analysis of the IRC based botnet, bots were correctly classified (99%) and Command and Control Nodes were fully identified. In the case of the bots, the nodes that were not identified did not participate in the coordinated group activity for some reason. Since they did not perform anything malicious they were not important from the attack analysis perspective. Botmaster nodes were partially identified (31%) but this result is again misled because all 9 of the botmasters that were not discovered only connected to each other and did not perform a malicious act. For victims we actually had 2 false positives. The manual analysis revealed that we incorrectly classified two botmasters as victims because multiple bot nodes sent a link of the same size to the botmasters within the same time interval. Finally we had a success rate of (91%) for discovering attacks. All the attacks that we missed were small-scale activities so they were not captured in our community analysis. Our community overlap rate was once again high across communities, which suggests we may be able to leverage this attribute in future analysis studies.

Method	Time	Bots	C&C	botmasters	Victims	Attacks
Bot Com	12 min	2123	5	4	45	59
Manual	2 days	2102	5	13	43	65

TABLE II
HTTP COMMUNITY ANALYSIS RESULTS

V. Discussion

Here we discuss future directions of our approach along with potential approaches.

Attack Reconnaissance Discovery. An issue with discovering botnet communities using our method is the possibility of not discovering all botnet transactions. For our analysis we set the value of k to 3, which allows us to identify all communities where at least three nodes can communicate with each other. The communities are not captured if it involves direct messages between two nodes, such as botmaster to botmaster communications. These transactions are usually used to discuss malicious plans or send test attacks towards a target. One way to solve this problem is to reduce the k value to 2, but this also greatly enlarges the size of the communities discovered. Instead of reducing the k value across the entire community, we will experiment with conducting an analysis that reveals all adjacent links and nodes of identified botmaster and command and control nodes. This should reveal the "silent" botmasters that only connect to other botmasters and command and control nodes.

Botnet Traffic Detection. Currently our method only applies to botnet data after an attack has been discovered by other methods. In the future we plan to investigate expanding our approach to further discover botnet traffic in a set of unfiltered network data. The current state of our algorithm will not allow us to accomplish this goal because of the significant amount of communities that will be created, such as the case addressed in [6]. Furthermore, at $k = 3$, some of the communities created by our approach would not be related to botnet traffic. To address this issue we plan to modify our algorithm by adding additional conditions which are more selective for constructing communities. One possible option is to require the detection of homogeneous interaction between nodes before a community is extracted. Currently we consider the homogeneous properties of the traffic after the communities are discovered, which initiates our node classification. The high success rate in correctly identifying these nodes show that the homogenous property is of significant value and has a reasonable chance to succeed in differentiating botnet communities from other communities on the Internet.

VI. Conclusion

In this article we introduced a new approach for analyzing botnet traffic. In particular, we made two notable contributions:

Our first contribution was a novel algorithm for botnet analysis based on an extension of k-clique community finding constructs. By discovering communities in botnets, events such as attacks are systematically identified without having to conduct a time consuming, manual analysis of the commands used to administer the botnets.

Our second contribution was a method to identify the category that each node belongs to within the botnet based on their communication patterns. Discovering this information early in the overall analysis process gives analysts enough information to make preliminary decisions, such as blocking attacking IPs and identifying key nodes such as the command and control servers.

Current methods to analyze botnets require the manual step of reverse-engineering the command and control protocol used to administer the botnet. This is a nessecary step for a detailed analysis, but presents an opportunity for the botnet to continue its nefarious acts during the procurement of the process. Our approach has shown through a botnet analysis comparison of our method, and an expert based manual method, that our approach identifies the correct category for each node with a high percentage rate and a low false positive rate. Since our approach is command and control protocol independent it can perform this analysis without reverse-engineering the botnet administration structure. This makes our analysis much faster than the detailed analysis, and the information provided can prevent attacks while waiting for the more detailed analysis to complete. These results show that our approach shows great promise as a potential add-on-layer to a defense-in-depth network protection strategy.

References

[1] B. Adamcsek, G. Palla, I. J. Farkas, I. Derényi, and T. Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[2] S. Borgatti, M. Everett, and P. Shirey. Ls sets, lambda sets, and other cohesive subsets. In *In the Proceedings of Social Networks*, 1990.

[3] D. Bradbury. Fighting botnets with sinkholes. *Network Security*, 2012(8):12–15, 2012.

[4] C. Y. Cho, D. Babic, E. C. R. Shin, and D. Song. Inference and analysis of formal models of botnet command and control protocols. In *Proceedings of ACM Conference on Computer and Communications Security*. ACM, 2010.

[5] D. Dittrich. So you want to take over a botnet. In *Proceedings of 5th USENIX conference on Large-Scale Exploits and Emergent Threats*. USENIX, 2012.

[6] E. Gregori, L. Lezini, and C. Orsini. k-clique communities in the internet as-level topology graph. In *Proceedings of the 31st International Conference on Distributed Computing Systems Workshop*, pages 134–139. IEEE, 2011.

[7] N. W. Group. Hypertext transfer protocol request for comments. *http://www.ietf.org/rfc/rfc2616.txt*, 1999.

[8] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through ids driven dialog correlation. 2007.

[9] G. Guofei, J. Zhang, and W. Lee. Botsniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium*. NDSS, 2008.

[10] G. Palla, I. Derényi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[11] N. C. Paxton, G.-J. Ahn, and M. Shehab. Masterblaster: Identifying influential players in botnet transactions. In *35th Annual IEEE International Computer Software and Applications Conference (COMPSAC)*. IEEE, 2011.

[12] N. J. Percoco. 2013 trustwave global security report. In *www2.trustwave.com/rs/trustwave/images/2013-Global-Security-Report.pdf*. Trustwave, 2013.

[13] D. Plohmann and E. Gerhards-Padilla. Malware and botnet analysis methodology. In *Proceedings of 4th Annual Conference on Cyber Conflict*. CyCon, 2012.

[14] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. v. Steen, F. Freiling, and N. Pohlmann. Sandnet: Network traffic analysis of malicious software. ACM, 2011.

[15] S. Seidman. Network structure and minimum degree. In *In the proceedings of Social Networks*, 1983.

[16] Z. Zhao, G.-J. Ahn, and H. Hu. Examining social dynamics for countering botnet attacks. In *54th IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2011.