

Utilizing Network Science and Honeynets for Software Induced Cyber Incident Analysis

Abstract

Framing the scene and investigating the cause of a software induced cyber-attack continues to be one of the most difficult yet important endeavors faced by network security professionals. Traditionally, these forensic pursuits are carried out by manually analyzing the malicious software agents at the heart of the incident, and then observing their interactions in a controlled environment. Both these steps are time consuming and difficult to maintain due to the ever changing nature of malicious software. In this paper we introduce a network science based framework which conducts incident analysis on a dataset by constructing and analyzing relational communities. The construction of these communities are based on the connections of topological features formed when actors communicate with each other. We evaluate our framework using a network trace of the malware network, BlackEnergy, captured by our honeynet. We have found that our approach is accurate, efficient, and could prove as a viable alternative to the current status quo.

1. Introduction

Today the importance of developing effective methods to analyze and defend against cyber attacks is no longer in doubt. Incidents such as the recent data breaches at Target and PF Chang as well as discovered and alleged attacks against Nation states, bring world-wide attention to the cyber attack problem [1-3]. This problem has been growing exponentially despite constant research geared towards reducing its impact. One of the major issues causing this upward trend in attacks is the dependence on an outdated analysis system [4]. Modern day attacks are often sophisticated and carried out at high speeds. In order to effectively understand and defend against these attacks, it is necessary to identify key attributes and actions quickly, before the originators of the attack can cover their tracks and attack other targets.

When dealing with attacks where malicious software (malware) directly interacts with a victim computer, current methods of cyber incident analysis

involve two major steps. Step one is to discover and manually analyze the malware that was sent to the infected system. Step two is to run the discovered malware in a closed simulated network (sandbox) and evaluate its actions based on the intelligence that was learned in step one. In theory these steps represent the most effective method to conduct a detailed analysis of the incident if the analyst conducting the manual analysis is a highly competent expert in malware analysis. Unfortunately, modern malware continues to evolve and become increasingly complex. Because of this, a detailed analysis is likely to take a significant amount of time for even the most highly qualified analyst to complete [4]. Furthermore, most sandbox analysis systems are limited in regards to simulating a network. For instance, in order to completely simulate an environment, the sandbox would need to be configured to include every item in the network that can be modified in any way [5]. This level of customization is unrealistic for most organizations.

We believe a beneficial approach which could improve the efficiency and effectiveness of analyzing a cyber incident should be able to identify the important actors involved in the incident without requiring a detailed internal description of the actor. This would reduce the time consuming roadblock of manually analyzing convoluted malware code. We also believe this approach needs to be able to identify and track the transactions that take place between actors. This would reduce the dependence on sandboxes for the discovery of behavioral characteristics. Community based analysis from the multi-disciplinary field of network science provides techniques that can satisfy these requirements. Here, actors and their interactions are modeled using graph theory where each actor is represented by a vertex V and each interaction between actors is represented by an edge E . Communities are then formed based on the context of the discovered interactions [6]. One of the findings we present in this paper shows that the discovered community structures and the relationships within the communities reveal the underlying meanings of the network as a whole. This is a finding that echoes the results of networks in other fields of study, such as biology and social networking [7, 8].

In this paper, we present and discuss our honeynet based framework that captures network traces of cyber incidents, identifies key actors within the network trace, and constructs communities based on the interactions of the discovered actors. Current tests are very encouraging and show that a meaningful analysis is possible without an in depth knowledge of the inner workings of the malware code.

The remainder of the paper is structured as follows. In Section 2 we discuss related work in the area of software based cyber incident analysis, honeynet technology, and network science based community detection. Section 3 discusses the components of the framework. In Section 4 we discuss the results, and Section 5 concludes the paper.

2. Background

Cyber incident analysis in general includes the gamut of cyber attacks conducted on a computer network. One category of these incidents include reflection attacks, such as the NTP reflection attack. Here an attacker sends a small request to a vulnerable NTP server or group of servers, but asks for the reply to be sent to a separate target computer. The NTP server then responds with a large amount of data which causes a denial of service in the target computer. Software bug exploitation is another form of a cyber incident. Here an attacker has knowledge of a flaw in software installed on a computer system. The attacker then uses this flaw to gain access to the system and is free to copy or take anything available based on the level of access he has achieved. Attacks such as these are especially troubling because it may not be possible to identify the intrusion. This is the case with the recently discovered heartbleed bug [9]. The analysis of reflection attacks, bug exploitations, and other incidents that do not involve software based malware directly interacting with a targeted network are beyond the scope of this paper, but in theory can be analyzed using our network science based approach. We leave the analysis of these types of networks for future works.

In this section we discuss previous research that focused on software induced incidents.

2.1. Related Work

In a recent survey on network-based botnet detection methods by Garcia et al. [10], the authors took an in-depth look at the most widely used and researched botnet detection tools available. In this survey there was a discussion which highlighted the fact that bot detection mechanisms have different

requirements than botnet detection mechanisms due to the difference in detecting one machine as opposed to a group of machines. Detecting a bot fits in the realm of this paper since a bot is software that induces the cyber incident. Of these methods, "BotMiner" [11], "BotSniffer" [12], "N-gram" [13], and "Tamed" [14], were similar to our method based on their detection approach. Our method is different from these and all the other approaches found in this survey because once the software is discovered in our system it is analyzed based on communities, which can be analyzed to determine relationships between actors located in the same community and throughout multiple communities. The methods presented in this survey employ clustering in their detection algorithms. Clustering is conducted based on a metric of distance and does not take into account relationships [6].

In another recent survey, which compared dynamic malware analysis techniques, Egele et al. conducted a study on methods that look to dynamically analyze malware to reduce the time gap between discovery of the malware to gaining intelligence from the malware [15]. In this survey the authors acknowledge that most forms of malware analysis still rely heavily on manual or static based analysis. They also discuss the major forms of malware discovered on the Internet today. The methods discussed in this work seek to conduct an analysis of the malware without first performing the manual static analysis step. They also use clustering and automated dynamic analysis reports to describe observed actions. These actions are then turned into behavioral profiles. Our approach also focuses on behaviors, but instead of using clustering, we use communities which allows us to identify relationships for a more fine grained analysis.

Two works which developed systems to detect and analyze software induced malware networks are "In Mining Botnet Behaviors on the Large-scale Web Application Community" [16] and "Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals" [17]. In both these systems the authors used machine learning to discover patterns within the network that explain botnet behaviors. Our framework has the same goal of identifying patterns, but machine learning techniques suffer from their dependence on generating a training set of data. Also, this type of analysis does not make any connections between identified nodes of interest.

2.2. Honeynet Technology Overview

Honeynets are networks composed of machines that are geared to attract and capture transactions of malicious users [18]. If one machine is configured to collect this data it is called a honeypot. This

technology has proven very useful in studying and defending against malicious networks. In most cases a general honeypot with a simple vulnerability will receive many attack attempts within minutes. Virtually any analysis tool or method can be plugged into the honeynet in order to run analysis on the discovered attacks. Honeynets can be configured in a variety of ways, based on the desired level of interaction from the malicious software. Simple implementations seek only to capture traces of automated attacks which search for vulnerabilities within systems on a subnet. These types of honeynets are considered low interaction. Other forms of honeynets allow the malware to connect to locations outside of the network. These types of honeynets are considered high interaction. The connections are normally limited so as not to allow the malware to damage outside sources from the honeynet. These types of implementations can become very elaborate. It is possible to create a framework which is an exact replica of a production network. Creating such a honeynet would allow the system administrator or researcher to investigate the effect a piece of malware would have on the corresponding production network. The goal of a successful honeynet is to record data and discover patterns in malicious traffic, without alerting the attacker, in order to discover a way to render the attack useless. Researchers and security professionals have used these methods to identify and shutdown attacks from all over the world.

2.3. Network Science Based Community Detection

Network Science is an inter-disciplinary field that studies the network representations of physical, biological, and social phenomena. It includes methods and theories from a wide range of fields. Detecting the community structure of graphs is a graph theoretic based approach that fits into this area. A network is said to have community structure if the nodes of the network can be grouped into multiple sets of nodes, where the sets of nodes are more connected internally than externally [19].

Communities are distinct from clustering because communities group nodes based on context and not just distance [6]. This fact becomes especially useful when considering node overlap. Node overlap is when a node is a member of two distinct communities at the same time [20]. By investigating the context and the semantics of both community memberships we can begin to understand the purpose of the memberships. We can also begin to understand the connections between the two communities.

There are several methods used to construct the communities. The most prevalent methods are hierarchy based [21], modularity (null model) based [22], information theory based [23], and clique based [24]. Each model has its drawbacks and advantages. In this paper we utilized the clique based method due to its natural ability to discover overlaps within communities. Other hybrid based methods are also able to detect overlaps and may be more efficient, so we will explore those options in future work.

3. Cyber Incident Analysis Framework

Our incident analysis framework is composed of five modules which can act as a standalone method for malware analysis. In this particular application we are conducting an analysis of a botnet which is a network of computers that has been infected by agents. This network is controlled by one or more commanders, which are called botmasters or botherders. We chose a botnet as our first test dataset because the important actors (Bots, Botnets, and Command and Control Centers) are well defined and network traffic concerning each actor can be detected by our honeynet. These factors made it a good choice for community detection.

3.1. Traffic and Log Collection Module

The traffic and log collection module is the entry point to the framework. It aims to capture and monitor the traffic at the edge of networks and each sensor of installed honeypots within the honeynet. This module has two modes. It can capture live traffic from a network in the form of packets by using the pcap library and it can load captured files stored on a file system. Currently the types of stored traffic that can be read with this module is pcap, netflow, argus, and sebek. Other forms of log data can be formatted to be included in our framework. Collected data is forwarded to the Traffic processing module.

3.2. Traffic Processing Component

The Traffic processing component aims to pre-process the traffic for flow correlation. The Traffic separator module within the component separates the traffic into 5 minute time windows in order for us to observe changes in behavior over time. The packet parsing module extracts header level information from the traffic such as source IP, destination IP, source port, destination port, TCP/UDP payload size except 0 size of TCP length, source to destination packet count and data size, destination to source packet count and

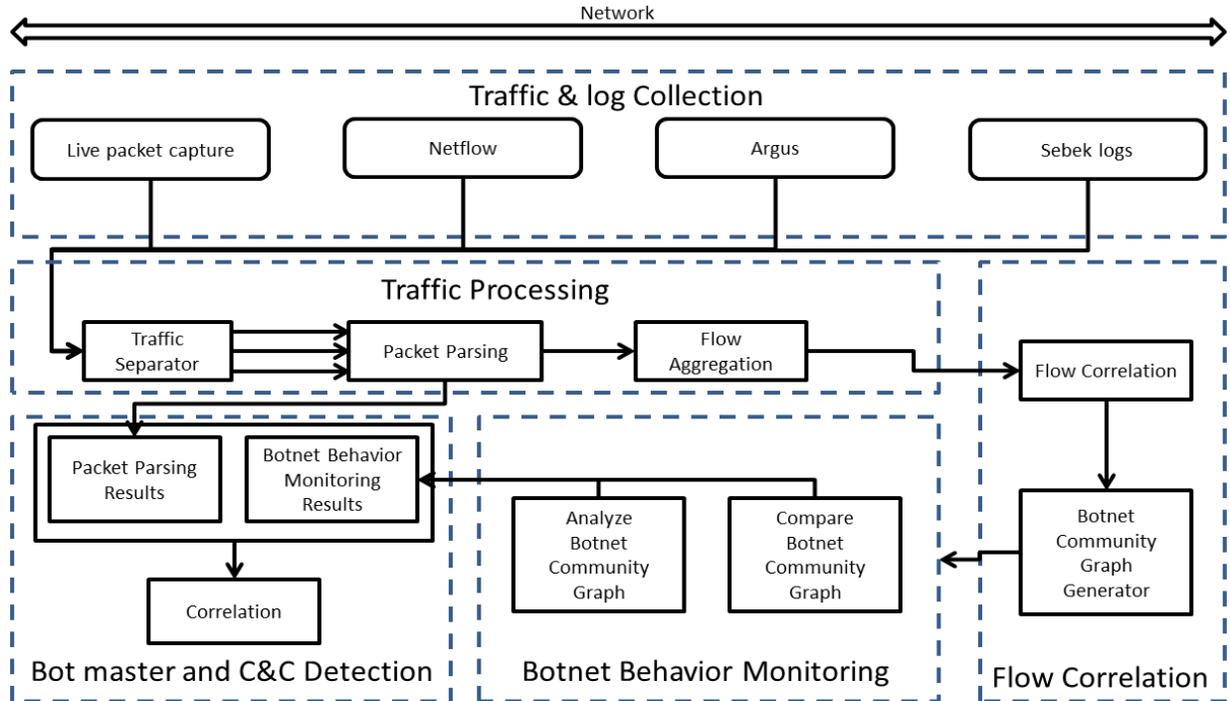


Figure 1: Community Based Cyber Incident Analysis Framework

data size, and session interval. And this module generates txt file and CVS file based on above information. The Flow aggregation module generates an input file for flow correlation. The matrix for correlation is {srcIP, dstIP}, {srcIP, TCP/UDP payload size}, and {TCP/UDP payload size, dst IP}.

3.3. Flow Correlation Component

The flow correlation component creates communities from the normalized data inputted from the flow aggregation module located in the traffic processing component. As mentioned earlier, we use the clique based method to generate the communities. This method relies on constructing a matrix which is built from completely connected sub graphs called cliques. In order to generate the cliques we first need to define what a node is. In our approach we used IP addresses as nodes and the message between a srcIP dstIP pair as the link.

The botnet community graph generator module inside the flow correlation component produces a visual image of the communities within a certain time period. This graph is then passed to the botnet behavior monitoring component.

3.4. Network Behavior Monitoring Component

In the network behavior monitoring component we analyze the current community graph and then compare it to previous graphs to determine dynamic changes that have occurred over time. Here we can discover the evolution of nodes and fluid relationships within the same community and across multiple communities.

3.5. Bot Master and C&C Detection

The Bot master and C&C detection component is where the results from the other components are combined and correlated to produce intelligence.

The botnet behavior modeling results module gives a display of what has changed over time within the community graphs.

The packet parsing results module captures and displays parsing results from each data source that was used in the analysis. This module was included to provide the analyst with a lens into the lower level data in case there is a question about the validity of the monitoring results. Here the analyst can manipulate the data before correlation is facilitated.

The correlation module adjusts the output of the behavior monitoring results depending on what has changed in the packet parsing results. If no change has been made the correlation results will be identical to the input for the behavior monitoring results. If a

change has been made, the results will change accordingly.

4. Results

Our preliminary results have shown that we can detect various types of data streams in our traffic and log collection module. All other modules are also functional, but results are currently being evaluated and will be presented in the next required revision.

4.1. Discovered Communities

Our clique based community detection algorithm was able to identify communities within the BlackEnergy data. This proves that the data is not random since it has a defined community structure. We were also able to track distinct nodes across multiple community graphs. This was made possible by performing a correlation of IP addresses with distinct user IDs which were discovered from a manual static analysis of the dataset. Without this step, distinct node discovery would be impossible due to the constant changing of IP addresses over time. Statistics showing the comparison of IP addresses and distinct username IDs will be completed before the next revision of this paper.

5. Conclusion

Here we introduced a cyber incident analysis framework which is based on the detection of communities within discovered attack data. This framework does not need to perform a time consuming manual analysis step or a closed system dynamic analysis step to identify intelligence from the data. Instead, all that is needed is high level identification data and knowledge of communications between the identified actors. Preliminary results have been very promising and have revealed community structures within the tested BlackEnergy data. We were also able to identify relationships between the communities due to the overlap discovered. We are currently collecting analysis results to be included in this paper and we will have a complete analysis by the next revision.

6. References

[1] Krebs, Brian, "The Target Breach, By the Numbers", <http://krebsonsecurity.com/2014/05/the-target-breach-by-the-numbers/>, May 14, 2014

[2] Krebs, Brian, "P.F. Chang's Breach Likely Began in Sept. 2013", <http://krebsonsecurity.com/2014/06/p-f-changs-breach-likely-began-in-sept-2013/>, June 14, 2014

[3] Geers, Kenneth, Kindlund, Darien, Moran, Ned, and Rachwald, Rob, "World War C: Understanding Nation-State Motives Behind Today's Advanced Cyber Attacks", <http://www.fireeye.com/resources/pdfs/fireeye-wwc-report.pdf>, 2013

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

[5] Rossow, Christian, Dietrich, Christian, J., Bos, Herbert, Cavallaro, L., Steen, Maarten van, Freiling, Felix, C., and Pohlmann, Norbert, "Sandnet: Network Traffic Analysis of Malicious Software", ACM, 2011

[6] Fortunato, Santo, "Community detection in graphs", Physics Reports, 2010

[7] Jia, Yuntao, Garland, Michael, and Hart, John, C., "Social Network Clustering and Visualization using Hierarchical Edge Bundles", Computer Graphics Forum, December 2011

[8] Leydesdorff, Loet and Ahrweiler, Petra, "In Search of a Network Theory of Innovations: Relations, Positions, and Perspectives", Journal of the American Society for Information Science and Technology (JASIST), 2013

[9] Codenomicon, "The Heartbleed Bug", <http://www.heartbleed.com>, April, 2014

[10] Garcia, Sebastian, Zunino, Alejandro, and Campo, Marcelo, "Survey on network-based botnet detection methods", Security and Communication Networks, May 2014

[11] Guofei, Gu, Perdisci, Roberto, Zhang, Junjie, and Lee, Wenke, "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection, USENIX, 2008

[12] Guofei, Gu, Zhang, Junjie, and Lee, Wenke, "Botsniffer: Detecting botnet command and control channels in network traffic.", In Proceedings of the 15th Annual Network and Distributed System Security Symposium, NDSS, 2008

[13] Abou-Assaleh, T., Cerone, N, Keselj, V, and Sweidan, R., "N-gram-based Detection of New Malicious Code.", In Proceedings of the 24th Annual International Computer Software and Applications Conference (COMPSAC 2004), Hong Kong, 2004

[14] Yen, Ting-Fang and Reiter, Michael, K., "Traffic Aggregation for Malware Detection", Detection of Intrusions and Malware, and Vulnerability Assessment Lecture Notes in Computer Science, Volume 5137, 2008

- [15] Egele, Manuel, Scholte, Theodoor, Kirda, Engin, and Kruegel, Christopher, "A Survey on Automated Dynamic Malware Analysis Techniques and Tools, ACM Computing Surveys, Vol. 44, No. 2, 2012
- [16] Garant, Daniel and Lu, Wei, "Mining Botnet Behaviors on the Large-Scale Web Application Community", In Proceedings of 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA), IEEE, 2013
- [17] Zhao, David, Traore, Issa, Sayed, Bassam, Lu, Wei, Saad, Sherif, Ghorbani, Ali, and Garant, Dan, "Botnet detection based on traffic behavior analysis and flow intervals", 27th IFIP International Information Security Conference, Computers and Security, Volume 39, Part A, November 2013
- [18] HoneyNet Project, "Know Your Enemy: GenII HoneyNets", <http://old.honeynet.org/papers/gen2/>, 2005
- [19] Lancichinetti, Andrea and Fortunato, Santo, "Community detection algorithms: A comparative analysis", Phys. Rev. E 80, 056117, November 2009
- [20] Xie, J., Kelley, S., and Szymanski, B., "Overlapping community detection in networks: the state of the art and comparative study. In Social and Information Networks, ACM, 2012
- [21] Wang, Jianxin, Li, Min, Chen, Jianer, and Pan, Yi, "A Fast Hierarchical Clustering Algorithm for Functional Modules Discovery in Protein Interaction Networks, Computational Biology and Bioinformatics, IEEE/ACM Transactions, 2011
- [22] Perry, Patrick and Wolfe, Patrick, "Null Models for Network Data", Available at <http://arxiv.org/abs/1201.5871v1>, 2012
- [23] Rosvall, Martin and Bergstrom, Carl, T., "An information-theoretic framework for resolving community structure in complex networks", Proceedings of the National Academy of Sciences of the United States of America, 2007
- [24] Palla, Gergely, Derenyi, Imre, Farkas, Illes, and Vicsek, Tamas, "Uncovering the overlapping community structure of complex networks in nature and society", Nature, June 2005