Developing a Network Science Based Approach to Cyber Incident Analysis

Napoleon Paxton
Center for High Assurance Computer Systems
Naval Research Laboratory
4555 Overlook Ave SW
Washington, DC 20375
UNITED STATES

napoleon.paxton@nrl.navy.mil

Ira S. Moskowitz, Stephen Russell, Paul Hyden
Information Management and Decision Architectures Branch
Naval Research Laboratory
4555 Overlook Ave SW
Washington, DC 20375
UNITED STATES

{ira.moskowitz, stephen.russell, paul.hyden}@nrl.navy.mil

ABSTRACT

Adversaries that conduct cyber crime continue to enjoy a significant head start on analysts who are tasked with discovering important information which can deter and ultimately defeat their attacks. A major reason for this problem is the slow process of the current analysis methodology. In this paper we present a new method of incident analysis which is artefact driven and not process driven. In our method, key aspects of the incident are revealed dynamically through the tracking of the interactions between the artefacts. With the discovered information, many attacks that are in progress can be stopped and new incidents can be prevented in a fraction of the time it would take to discover this information through traditional analysis. This new method builds a community for each individual incident found within the network. We evaluate our approach on two botnet data traces. Our preliminary results show that the communities built based on the artefact interactions shed light on the roles of each contributing botnet participant. Discovering these roles gives the analyst expedient options in responding to the attack. We believe this work has the potential to significantly help cyber incident analysis by reducing the time gap between identifying an incident and discovering actionable intelligence from it.

1.0 INTRODUCTION

Cyber incident analysis can be a difficult and time consuming endeavour in large part because of the process driven approach of most analysis techniques. Currently the status quo requires malware analyst to first discover an attack, next identify the malicious software (malware) behind the facilitation of the attack, then the malware is manually analyzed at the machine level to determine what it is capable of doing. Finally, the malware is dynamically analyzed in a sandbox environment to determine what its basic behaviours are. The dynamic step can be further complicated by protocol customization, which is a method malware network administrators frequently use to make outside analysis of their methods more difficult. Although this process is currently necessary for a long term solution; an intermediate, less invasive incident analysis could prove beneficial in the short term, and is needed in order to reduce the damage caused by cyber attacks. In this paper we present an approach that utilizes community detection algorithms as a way to analyze malicious events. Our approach is built upon the scientific principles of Network Science which is a multidisciplinary field based on graph theory.

2.0 CYBER INCIDENT ANALYSIS ISSUES

Currently, the process driven approach that is in place for cyber incident analysis suffers from what is often a significant time gap between discovery of the incident and gathering enough meaningful information (actionable intelligence) to stop or prevent future attacks based on the infecting agent. Issues that contribute to this gap in time are the manual analysis of the malware, the skill level of the analyst, code obfuscation, and slow and low infection techniques.

2.1 Manual Analysis

Manual analysis, which is also called static analysis, refers to an analyst receiving malware and using ad-hoc tools to discover the internal functionality of the code [1]. Manual analysis does not take into account the behavior of external files or processes after an infection has taken place (this is the job of a dynamic analysis which is beyond the scope of this paper). Tools that are normally used for static analysis are: a disassembler, a virus scanner, and payload analysis tools. Most analyst use IDA Pro for disassembly [2], but the virus scanner [3 - 6], and payload analysis tools [7 - 10] vary greatly. Because of this lack of uniformity, collaborating analysts which use different methods must spend time to understand the analysis process of the other analysts. Furthermore, manual analysis is a very slow process because of the multiple steps involved and the sophistication of present day malware. One example of this sophistication is the malware that was used to attack the Iranian nuclear centrifuge. In the most comprehensive analysis to-date Langner discusses how even after three years the complexities of stuxnet malware has not been solved [11]. One analyst believes it will take at least ten years to fully analyse stuxnet [12]. Malware this sophisticated can continue to threaten networks during this time.

2.2 Analyst Skill Level

Analyst skill level greatly affects the time of analysis. In theory more experienced analysts can perform malware analysis quicker than less experienced analysts [13]. There are several reasons why this should be the case. For instance, a more experienced analyst should be more familiar with the machines and the network in which the malware has infiltrated. They can then count on that experience to notice “strange” changes in the system or network. Experienced analysts should also be more familiar with their tools and the various ways they can be used. Another advantage of an experienced analyst over a less experienced analyst is their ability to explain their network or tools to others. Each of these reasons assumes the best case scenario and is not typical for most organizations, but even with the best case scenario the analysis process is normally lengthy due to the amount of steps involved.

2.3 Code Obfuscation

A common way malware writers complicate the tasks of analysts is by obfuscating their malware code. Today most malware is encrypted and depending on the encryption; hours, days, or even weeks can pass by before the internal code of the malware can be analyzed. This delay in discovering intelligence is currently unavoidable, but if we can discover artefacts without needing to investigate the internal composition of the malware code, the time of analysis will be reduced drastically. Our community based approach aims to accomplish this task.

2.4 Slow and Low Techniques

Advanced malware typically operates in a way that is out of the norm. One approach is a delayed execution cycle called slow and low. In this technique malware is programmed to delay its execution for a period of time or until an event happens within the system or network [14]. Discovering information from this type of malware is often very difficult without a complicated dynamic analysis which involves multiple simulated services and programs. Even with this highly dynamic system, it is possible that the process or program needed to trigger the malware is not installed on the system. Developing an effective analysis method for this type of malware is ongoing. Since certain actions are only made evident when artefacts of the malware interact, the current manual approach is very limited. We believe our approach can be beneficial in analyzing this malware because our approach is based on artefact interaction and not the malware code.

3.0 NETWORK SCIENCE BASED MALWARE ANALYSIS

Network Science is an inter-disciplinary field that studies the network representations of physical, biological, and social phenomena [15]. It includes methods and theories from a wide range of fields. Detecting community structure is a graph theoretic approach that fits into this area. The first step in conducting this analysis is determining whether or not the network being analyzed has community structure. 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 [16]. Communities are distinct from clustering because communities group nodes based on context and not just distance [17].

3.1 Community Structure Overlap

Community structure overlap occurs when a node, or nodes, is part of multiple communities at the same time. The concept of time is relative to the analyst. In our approach time refers to a selected interval that can be manipulated to generate a larger or smaller graph of communities.

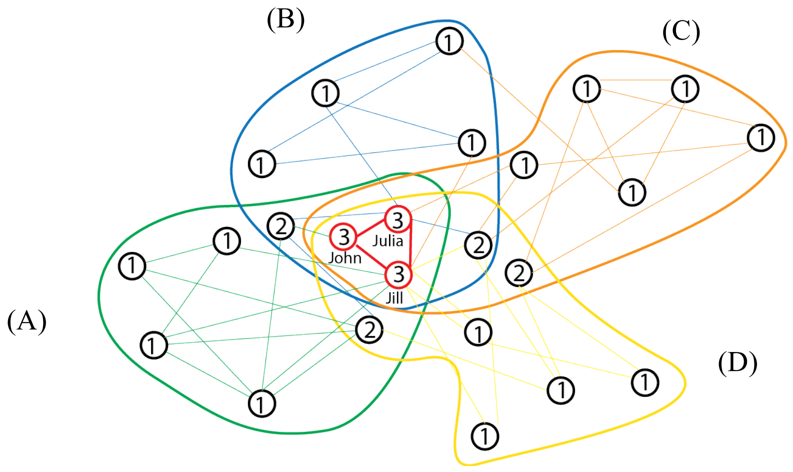

Figure 1: Community Structure Overlap Example - There are 16 nodes in this network that only belong to one community, 4 nodes that belong to 2 communities, and 3 nodes that belong to all 4 communities.

Case Study: In figure 1 we have a network with the communities (A, B, C, and D). There are 16 nodes that only belong to one community (4 in each community), 4 nodes that belong to two of the communities, and John, Jill, and Julia (outlined in red) are the only nodes that belong to all four of the communities. The immediate reaction to this graph is: Who are the three nodes that belong to all four communities, and how are the communities related? Further inspection would show that John, Jill, and Julia are leaders in each of the communities. Furthermore, Jill seems to be the most important node in the network because she has the most connections to and from other nodes. In other words, she is the node with the highest degree. This type of analysis can be very beneficial when trying to determine relationships between nodes, and when trying to decide a course of action on nodes.

3.2 Differences between Community Detection Methods

There are several types of algorithms used in community detection. The most popular methods are Modularity Based, Hierarchical Based, Information Theoretic Based, and Clique Based. Although each algorithm is structurally different, they all follow the basic principles of community detection; nodes in a community are more connected than nodes outside of a community, communities can be visually represented on a graph by revealing the nodes and edges, and nodes are related based on a semantic edge connection and not a distance based metric [18]. We believe this level of cohesion is sufficient enough to provide a structured malware analysis platform which will make it easier for analysts to share results as long as each malware network exhibits community structure.

4.0 OUR APPROACH

The intuition behind our method is that all networks can be represented as graphs (the associated adjacency matrix only has 0 or 1 entries, is symmetric, and is all zeros down the main diagonal). Malware infested networks are no different in that regard. However, not all networks have community structure, so the question we need to answer is: Do networks composed of malware exhibit community like behavior? To conduct our analysis we follow these steps shown in table 1.

Table 1: Community Detection Based Analysis Steps

Step	Action
1	**Model malware networks as graphs:** *G = (V, E).* • V = {Botmaster, Bot, Command and Control, Victim} • E = {SrcIP, DstIP,MsgLen}
2	**Segment the malware network traffic into groups or "communities" based on non-directional transactions:** Multiple communities are discovered in a community graph which is based on a set time interval. Direction is not considered when discovering communities.
3	**Apply directional data to the communities:** Direction initiates at the node sending the message and terminates at the node that receives the message.
4	**Perform Analysis:** Discover features that an analyst can use to defend against malware networks.

4.1 Experimental Malware Based Approach

To evaluate our community detection based malware analysis method, we analyze packet capture data from a HTTP based botnet and an IRC based botnet. Both botnets are considered centralized because bots connect to a command and control server in order to receive their commands. The difference between the two botnets is: the HTTP based botnet has a dedicated connection to the botnet and the IRC based botnet is administered by broadcast.

4.1.1 HTTP Botnet

Each node (*V)* in a HTTP botnet is discovered on a message to message basis. Bots in a HTTP based botnet send a POST message to the command and control server using the HTTP protocol. After this transaction our first connection is discovered, (*Bot connected to Command and Control*). If the authentication is successful, the command and control server sends a message to the bot with instructions on what it should do, (*Command and Control to Bot).* The bot then sends periodic messages to the command and control for further instructions, (*Bot to Command and Control*). Messages are programmed into the command and control by the bot master, but the bot master does not directly interact with the command and control in real time as is the case with the IRC botnet (*Botmaster to Command and Control*). Victims are the results of messages being sent to them from bots that were instructed based on their previous instructions after checking in with the command and control, (*Bot to Command and Control, Command and Control to Bot, Bot to Victim*). The Botmaster to Command and Control connection is not present because in most cases it will occur outside of the connection interval window. This correlation will have to be completed later in the analysis.

4.1.2 IRC Botnet

Nodes in the IRC based botnet are discovered in a more straight forward way than with a HTTP based botnet. The botmaster, command and control, and all the bots subscribe to an IRC channel where all commands are sent. When attacks are made, Victims are discovered. Bots that subscribe to a botnet channel are sent a message from the command and control directly after the bot master gives the command. IRC based data is easier to obtain and analyze, because one node can observe traffic from every other node in the botnet channel. Using this method we can also track the evolution cycle of the botnet, which will be accomplished at a later time.

5.0 EXPERIMENTAL RESULTS

Using the methodology described in table 1, we were able to identify the roles each node plays within both botnet datasets. We compared our results with the findings of a manual analysis of both datasets, which served as our ground truth.

5.1 HTTP Results

For the HTTP botnet success rates were: Bots – 99%, Victims – 91%, Command and Control – 100%, and Botmaster – 31%. Further analysis of the Botmaster results revealed that each undetected node was due to the lack of a malicious act performed by the Botmaster. In other words, the Botmasters communicated with each other instead of sending commands for attacks. We were able to later identify these Botmasters with a secondary analysis which included aggregating the message length of the communications and correlating the IP addresses with previous attack IP addresses. Further testing is needed to determine if this is a sustainable addition to our analysis method.

5.2 IRC Results

IRC botnet results were: Bots – 99%, Victims – 87%, Command and Control – 100%, and Botmasters – 67%. Although the detection of Botmasters was better in the IRC botnet, the same issue was at fault for the low detection rate compared to the other roles. We were also able to detect the missing Botmasters using the aggregation and correlation method, which gives us confidence that this could be a viable addition to our approach.

6.0 RELATED WORK

Two works which developed systems to detect and analyze malware using artefacts instead of an adhoc process are "In Mining Botnet Behaviors on the Large-scale Web Application Community" [19] and "Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals" [20]. 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, which is an important benefit of community detection.

7.0 CONCLUSION

In this paper we introduced a cyber incident analysis methodology which is based on the detection of communities within discovered attack data. This method does not need to perform a time consuming manual analysis step to identify intelligence from the data. Instead, all that is needed is high level identification data in the form of Nodes and knowledge of communications between the identified Nodes. Preliminary results have been very promising and have revealed community structures within the tested Botnet data, as well as roles of the nodes in the community structure. Identifying these nodes provides us with valuable information, such as the identity of the Command and Control (to shut down or monitor the botnet), Bots (to identify the infected machines in the botnet), Victims (to identify what machines are targeted), and Botmasters (to identify the possible human behind the attack). We believe further development of our approach can provide an intermediate step for incident analysis until a more comprehensive analysis can be conducted. This will allow the analyst to take immediate action on discovered intelligence. We also believe the structured nature of Network Science and in particular, Community Detection can lead to a more structured analysis environment, which will enable security analyst to better share their results.

8.0 REFERENCES

[1] Brodersen, R.W.,D. Plohmann and E. Gerhards-Padilla. Malware and Botnet Analysis Methodology. In Proceedings of 4th Annual Conference on Cyber Conflict, CyCon. 2012

[2] Chris Eagle. The Ida Pro book: the official guide to the world's most popular disassembler. William Pollock. Russell, S., Forgionne, G., and Yoon, V. 2009

[3] J. H. Wang, P. S. Deng, Y. S. Fan, L. J. Jaw. Virus Detection Using Data Mining Techniques. In the Proceedings of the IEEE 37th Annual 2003 International Carnahan Conference. October 2003

[4] O. Henchiri, N. Japkowicz. A feature Selection and Evaluation Scheme for Computer Virus Detection. In the Proceedings of the Sixth International Conference on Data Mining. December 2006

[5] A. Sulaiman, K. Ramamoorthy. Malware Examiner using Disassembled Code. In the Proceedings of the

Sixth Annual IEEE SMC Information Assurance Workshop. June 2005

[6] A. H. Sung, J. Xu, P. Chavez, S. Mukkamala. Static Analyzer of Vicious Executables (SAVE). In the Proceedings of the 20th Annual Conference on Computer Security and Applications. December 2004

[7] R Perdisci, D Ariu, P. Fogla, G Giacinto, and W. Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. In the proceedings of Computer Networks 53. 2009

[8] B. Biggio, I. Pillai, S. Rota Bulo, D. Ariu, M. Pelilo, and F. Roli. Is data clustering in adversarial settings secure? In the Proceedings of the 2013 ACM workshop on Artificial Intelligence and Security. 2013

[9] R. Perdisi, D. Ariu, and G. Giacinto. Scalable fine-grained behavioral clustering of HTTP-based malware. In the proceedings of Computer Networks. 2013

[10] D. Ariu, G. Giacinto. A Modular Architecture for the Analysis of HTTP Payloads based on Multiple Classifiers. 2011

[11] Ralph Langner. To Kill a Centrifuge. Langner. The Langner Group. November 2013

[12] Kim Zetter. (2012, May 28). Meet 'Flame,' The Massive Spy Malware Infiltrating Iranian Computers. Wired. Retrieved from http://www.wired.com/2012/05/flame/all/

[13] 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

[14] A. Alazab, M. Hobbs, J. Abawajy, A. Khraisat. Developing an Intelligent Intrusion Detection and Prevention System against Web Application Malware. Springer. 2013

[15] J. Fang, X, Wang, Q. Bi, Z. Di, X, Li. New Interdisciplinary science: Network Science. Progress in Science. 2007

[16] G. Palla, I. Derenyi, I. Farkas, T. Vicsek. Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society. Nature, 2005

[17] A. Lancichinetti, S. Fortunato. Community Detection Algorithms: A Comparative Analysis. Phys. Rev. E 80, 2009

[18] S. Fortunato. Community Detection in Graphs. Physics Reports, 2010

[19] D. Garant, W. Lu. Mining Botnet Behaviors on the Large-Scale Web Application Community. In the Proceedings of the 27th International Conference on Advanced Information Networking and Applications Workshops. 2013

[20] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, D. Garant. Botnet Detection Based on Traffic Behavior Analysis and Flow Intervals. In the Proceedings of the 27th International Information Security Conference. 2013