BotViz: A Memory Forensic-Based Botnet Detection and Visualization Approach

Iman Sharafaldin, Amirhossein Gharib, Arash Habibi Lashkari, *Member, IEEE,*
Ali A. Ghorbani, *Senior Member, IEEE*
Canadian Institute for Cybersecurity (CIC), University of New Brunswick (UNB), Canada
isharafa@unb.ca, agharib@unb.ca, a.habibi.l@unb.ca, ghorbani@unb.ca

***Abstract*—Nowadays, there are many serious cyber security threats such as viruses, worms and trojans but without a doubt botnets are one of the largest threats. Although there are numerous ways to discover botnets and mitigate their effects, most methods have problems effecting detection, due to their evasive characteristics. Also, the majority of previous research uses only one data source (e.g. network traffic), which makes the botnet detection process very difficult over a network. This paper proposes a detection and visualization system, BotViz, to visualize botnets by using memory forensics analysis and a new domain generation algorithm detector. BotViz utilizes machine learning techniques to detect anomalous function hooking behaviors. We established a live Zeus botnet to evaluate the efficiency of the BotViz.**

***Keywords*-Botnet Detection, Botnet Visualization, Domain Generating Algorithm, Memory Forensics**

I. Introduction

A botnet is a network of infected computers remotely controlled by one or more management hosts to accomplish certain tasks. Botnets are used for many online crimes such as DDoS (Distributed Denial of Service) attacks, identity theft, click fraud, stealing information, phishing and spam emails. The key to the operation of a botnet is the availability of one or more functional command centers, usually called Command and Control (C&C), to coordinate and control slave hosts. Previously administrators could easily deactivate botnets by blacklisting their C&Cs. But the recent advanced botnets such as Conficker, Kraken and Torpig have designed and developed a new method for botnet operators to control their bots and evade this deactivation by DNS domain-fluxing [7].

In DNS domain-fluxing, each bot feeds their DGA (Domain Generation Algorithm) with a starting seed to generate a large set of random domain names and then queries them until one is resolved. Then the bot contacts the corresponding obtained IP address which is typically used to host the (C&C) server. All bots generate the same random domain names every day, in order to make it harder for a security vendor to pre-register the domain names.

On the other hand, one of the most important attack vectors of botnets is their function hooking mechanism on a computer. The term "API/Functions hooking" includes a range of techniques that are used to modify the behavior of an operating system or applications by capturing function calls, messages or events passed between software components. Botnets hook functions for different reasons such as spying, getting confidential information and stealing critical information. Sometimes hooks are used to obtain alerts for the arrival of new sensitive data, to capture victims keystrokes, or to steal users' passwords. Botnets may also use hooks to get confidential information from web pages. Also botnets implant hooks to change critical system information to hide their presence in the system. Similarly, they can hook network functions to build a stealthy communication channel with their C&C servers [2]. Furthermore, there are numerous papers published on malware and function hooks mainly on monitoring function calls ([44], [45], [46], [47]) , and finding hooks ([48], [50], [51], [52], [53]). To the best of our knowledge, none of the aforementioned papers worked on clustering botnets or malwares by their hooking behaviors.

Also, it has been shown in numerous papers that visualizations will allow analysts to complement machine learning algorithms and hugely improve the detection of the abnormal activities by themselves [42]. Also, as it is shown in the section on related works, one of the major short comings of botnet visualizations is that they do not consider host-based information to visualize security threats.

In this paper, we present BotViz, a framework for detecting and visualizing botnets. BotViz contains the following modules:
1) Data Collector Module (DCM): to collect DNS logs and memory dumps.
2) DGA Detector Module (DDM): a new fast DGA detector algorithm with high accuracy and low false positive rate.
3) Memory Forensics Module (MFM): to analyze memory dumps and extract different types of hooks.
4) Analyses Modules (AM): to encode hosts to vectors and clustering them to distinguish suspicious hosts.
5) Visualization Module (VM): to visualize suspicious hosts on network topology diagram.

For the evaluation part, we evaluate DDM separately, because it can work as an independent module. We use Alexa's top 10000 domains as our benign dataset. For the final evaluation, we compiled Zeus botnet leaked source code and established a live botnet. The main reason for establishing a live botnet is the difficulty of finding a live botnet, because after any samples are discovered the authorities start to bring down the C&Cs, and the sample would become useless immediately. Furthermore, we used Zeus because after source code leakage it became the mother of many future botnets like SpyEye, Ice IX, Citadel, Carberp, Bugat, Zeus Gameover, Shylock and Torpig [39].

978-1-5386-1585-0/17/$31.00 ©2017 IEEE

The contributions include the following:
- To propose a new fast and scalable DGA detector with high accuracy and low false positive rate
- To present a novel botnet detector and visualizer, which is based on detecting anomalous function hooking behaviors.

II. Related Works

As we have three contributions in this paper, consequently we divide related works into three parts. The first part reviews the available botnet detection techniques for memory forensics (Sub-Section II-A), the second part focuses on the DGA detection algorithms (Sub-Section II-B) and the third part shows the available botnet visualization methods and enumerates their weaknesses and strengths (Sub-section II-C).

A. Available Botnet Detection Techniques Using Memory Forensics

Among numerous approaches that have been proposed for botnet detections, some researchers utilize memory forensics techniques in their proposed model. Law *et al.* [11] propose collecting relevant digital traces from a new host-based investigation approach for botnet detection. Their approach utilizes the observed similarities of bot infected machines at the local level with established network based investigation techniques for collecting traces. In the memory forensics part, they just used memory tool for detecting suspicious process without providing any self-defined heuristics.

Memarian *et al.* [12] present the EyeCloud as a detection system for members of botclouds that abuse cloud resources. The authors take advantage of data mining and virtual machine introspection techniques to group malicious and non-malicious VMs. For the training machine learning techniques, they use six features, namely the existence of hidden processes, shutting down the firewall, inappropriate commands in the history of the command shell, existence of hidden DLLs, shutting down windows defender, and shutting down automatic update.

Soltani *et al.* [29] address the lack of real world botnets detection. They review architecture, protocol, type of infection, communication interval, attacks and evasion techniques of eight botnets Conficker, Kraken, Rustock, Storm, TDL4, Torpig, Waledac, and Zeus. Also, they study the drive-by download attacks available detection techniques and some of the evasion techniques used by botnets to delay and hinder shutdown attempts.

Sharma *et al.* [30] study botnet detection frameworks and propose a generic framework based on the approach of passively monitoring network traffic. In their propose framework they focus on monitoring different kinds of similar behavior and patterns for detecting botnet behavior on four types of traffic: P2P, IRC, HTTP and SMTP. Their developed system has three components, *detector* which detect the bots from the network traffic, *analyzer* which groups bots into different similar activities and *action* which works on the various stages such as collection, retention and observation.

Kebande and Venter [31] believe that virtualization as a component of cloud computing is providing the users an immediate way of accessing limitless resources, but botnets are one of the most dangerous intruders in this domain. The authors present a new approach to infection detection of a robot network in the cloud environment. For matching whether the botnet belongs to self or non-self pattern in the proposed algorithm they use a negative selection algorithm and in the future they want to use the genetic algorithm for improving it.

Kwon *et al.* [32] introduce a scalable and fast approach for malicious behaviour detection within a large volume of DNS traffic and named it PsyBog. It uses power spectral density (PSD), a signal processing technique, to discover major frequencies of DNS queries of botnets. The proposed approach covers groups of hosts with similar malicious behaviour patterns. In the evaluation part they use two large datasets and utilize the malware traces as the ground truth. The detection accuracy is about 95% including 23 unknowns and 26 known botnets with 0.1% false positive rate.

B. Available Botnet Detection Techniques Using DGA Detector

Several reports have shown that the bot masters utilized DGA for elasticity improvement between bot and C&C servers nowadays, many researchers are focusing on this technique and are developing botnet detection systems by using DGA [7], [14], [16], [15], [17], [10], [18], [19], [20]. They believe Domain Generated Algorithms from one algorithm have similar attributes and these attributes are different from those of legitimate domains. Based on these facts, they adopted classification and clustering algorithms for detection.

Yadav and Ranjan [7] propose a new method for DGA domains detection, which botnets are using for domain fluxing. For classifying the malicious websites, they use Kullback-Leibler divergence, Levenshtein edit distance and Jaccard index. Finally, an analysis of a set of legitimate domain names and DNS traffic from Tier-1 ISP has been performed. In this research, the Jaccard index produces the best results followed by the Levenshtein edit distance and Kullback-Leibler divergence.

Antonakakis *et al.* [14] train models of benign and malicious domains with a feature vector consisting of 18 network based features, 17 zone based features, and 6 evidence-based features. They score new domains with these trained models. In their other work [17] in 2012, they introduce a detection method of clustering NXDomains which are queried by multiple infected hosts and share similar linguistic features.

Ron *et al.* [10] design a new method to detect threats in DNS logs. They generate 13028 domains and tag as malicious, and also select the top one million ranked sites from “Alexa.com” and tag them as benign domains. Their language model uses these datasets to learn “normal” and “abnormal” domains. They utilize Variable-order Markov Models (VMMs) in their proposed method because the running time of input is linear. They indicate high detection rates in their experiments with less than 5% false-negative and 1% false-positive rates.

Zhou *et al.* [18] present a technique to detect DGAs by analyzing patterns in DNS NXDomain traffic. They cluster all domains extracted from DNS traffic, according to second TLD

and IPs. Then, they look into the clustered domains to see if the domains in one cluster have similar lifetime spans and visit patterns. They use this approach to detect DGAs since every domain name in the domain group generated by one botnet is often used for a short period of time, and has similar life time and query style.
Schiavoni *et al.* [20] suggest using Phoenix to detect DGAs domains from other domains. In their approach they extract linguistic and IP-based features.

C. Available Botnet Visualization Techniques

Security visualization has been an active area of research for more than a decade [42]. Out of many papers which are published on network security visualization, a few of them specifically targeted botnet visualization.
Conti *et al.* in 2005 [21] presents a visualization tool with a view of a large number of network packets to support both real-time and forensic analysis of packet-level data. Although authors aim at using novel methods of displaying network data, but in many cases, the visualizations are only suitable for a particular kind of attack and would not be applicable for an analysts' everyday use.
Portal [34] and HoNe [35] investigate the monitored hosts and try to correlate TCP connections with the host processes that produce them, creating an end-to-end visualization of communications between distributed processes.
Mukosaka and Koike [37] present the visualization technique for security mechanisms in large-scale LANs. Their proposed technique provides a strong filtering mechanism along with 3D visualization. The proposed 3D visualization integrates logical, geographical and temporal information together with a strong filtering mechanism. The main criticism of the proposed model is the large false detection rate and the failure to support inbound traffic and reliance on the minimum information for anomaly detection.

Cremonini and Riccardi [22] offer Dorothy which can monitor the activity of a botnet infiltration. They infiltrate and monitor Siwa bot and characterize its behaviour through a set of parameters such as functional structure, geographic distribution, communication mechanisms, command language and a graphical representation. Their system can just illustrate the plain and unencrypted IRC communication.

Mansman *et al.* [23] visualize essential events from a large set of alerts by considering different information from each alert by using different techniques such as trees and graphs. The authors explain the usability of the proposed system through three case studies, including analysis of service usage in a network, the detection of a distributed attack, and identification of hosts that are susceptible to communication with external IPs (for malicious activity).

Shahrestani *et al.* ([24], [25]) presents a combination of data mining and visualization for network flow analysis, wherein the malicious data pass through several trust models, and after re-evaluation of the flows, the data aggregate to detect malicious traffic through visualization. They use the human, intellectual and conceptual ability of their proposed model for analyzing the visualized information to gain useful knowledge about botnet activities for further precaution and validation. The visualization techniques used in this system consist of Graphs, Scatter plots, and Histograms which are easy to interpret and suitable for large scale datasets.
These techniques can be used to visualize a limited set of invariant bot behavior. First, fast response time: Bots reply to the botmasters command very quick. In other words, human response to a command or request is always much slower than a bot. Second, small size commands: the lengths of command packets are typically very small. Despite the normal packets that have unbounded size, a typical command packet from botmaster has a small size of 1KB or less. Third, instant execution of commands: bots may launch an executable application on the infected host machine immediately after receiving the botmasters command. They just cover theoretical aspects and do not prove the ability of the system in the real world scenarios.
Abdullah *et al.* [26] develop a novel visualization system, IDS RainStorm, to address the problem of the flourishing number of alerts generated from intrusion detection systems in large networks. They argue that manually traversing textual logs is not only a frustrating and time consuming task. They insist that using information visualization in network security would assist the analyst in gaining new insights and would aid in identifying patterns of anomalous network behavior. The develop system consists of a main view which displays an overall representation of the network and a zoom view that provides a detailed display of a user selected range of IP addresses.
The main view depicts in eight columns in a top to bottom manner which is a contiguous set of IP addresses (such as 20) and allocated to a single row. The proposed method allows a 2.5 class B block to be represented in a single display screen for a full 24 hours. Three level alert is represented also as color coded pixels: red for high, yellow for medium, and green for low-severity. The authors evaluate their system by showing various usage scenarios, each carefully chosen to display the diverse set of activities that can occur, internally or externally, against a large network. Based on evaluation results is possible to finalize that Presented columnar view is not an ideal method for presenting attacks.
Kim *et al.* [27] introduce development of a visualization mechanism using DNS traffic. The proposed mechanism enables network administrators to discover botnets and malicious activities within normal traffic intuitively. The system reveals hidden botnets and obtained distinct patterns by using the filter mechanism. However, as some false-positive patterns cause legitimate programs look like botnets, it would be possible for a botnet to modify its behavior to bypass this mechanism, enabling the system to generate false positives.
Zhao *et al.* [28] present a network security visualization framework, IDSRadar, to assist in understanding IDS alerts and in identifying the abnormal pattern behavior in overwhelming false positives. They use five catalogs of entropy functions to describe how to analyze the attack pattern and recognize the false positives. Finally, the authors evaluate the proposed model with attacks provided by VAST challenge and have shown how the framework can be used to illustrate the attacks

and visually correlate the events. The proposed model was not effective against attacks as the five categories of entropy are constant.

III. The Proposed Framework

Figure 1 shows the overall structure of the proposed system. There are five modules: Data Collector Module (DCM), DGA Detector Module (DDM), Memory Forensics Module (MFM), Analyze Module (AM) and Visualization Module (VM). In the next five sub-sections we explain these modules.

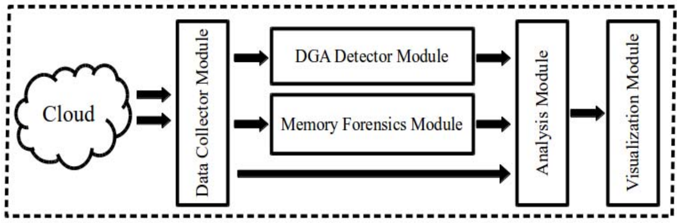


Fig. 1: The proposed framework

A. Data Collector Module (DCM)

DCM is responsible for collecting DNS logs and memory dumps of virtual machines and sending them to other modules for further processing. The proposed framework "BotViz" uses the following four different features from a DNS reply packet:
1) Host IP - the IP address of the host which requested specific domain name;
2) Target name - which is a domain name of the server (also can be C&C name);
3) Target IP - which is the IP address of requests domain name;
4) TTL (Time to Live)- the time interval for caching.

DCM also interacts with a hypervisor and collects memory dumps of virtual machines using two methods. The first method uses a software development kit (SDK) for that specific hypervisor, while the second method uses any type of Virtual Machine Introspection (VMI), which is a common technique to examine the memory space of a virtual machine from a secure point. In this work, VirtualBox is used as the hyperviser; DCM connects to virtual machines through VirtualBox SDK, and collects memory dumps of virtual machines and sends them to the MFM module for further processing.

This model can also be modified for a non-virtualized environment. To achieve this goal, it is necessary to develop and execute agents for collecting memory dumps of computers (monitored hosts) and forward them to MFM. In the proposed model, the VirtualBox is used as our test bed. Therefore, it is not required to install and execute agents on the hosts which is the main advantage of using a virtualized environment.

B. DGA Detector Module (DDM)

Botnets feed their DGA with a starting seed to generate random domains. For instance, Conficker-A [3] bots generate 250 domain names every three hours using the current date and UTC time (in seconds) as the seed, which in turn is obtained by sending empty HTTP GET queries to a few legitimate sites, such as google.com, baidu.com and answers.com. But, all bots generate the same domain names every day. In order to make it harder for a security vendor to pre-register the domain names, the subsequent version, Conficker-C [4], increased the number of randomly generated domain names per bot to 50K. As an another example, Torpig [5] bots employ an interesting trick by using a seed for the random string generator based on one of the most popular trending topics in Twitter.
To handle this obstacle, a string-based approach has been proposed to classify malicious domains generated by DGAs. DGA domains are generated with a high degree of randomness, and this module takes advantage of it.
DDM calculates a score for domains based on its lexical features. First, it looks to extract the words in top level domain(TLD); then, it scores the domain by counting the number of alphabets in detected words divided by the total length of the domain. Note that if two or more words overlap, the overlapping characters will be counted only once. In this approach, short words such as "am", "is", and "I" are escaped. This boundary is necessary because including many short words can increase the score and hence reduce the accuracy. Usually, the length of malicious domains is greater than eight characters since the generated domain should be available (the botmasters will encounter problems to buy domains less than eight characters, because majority of them are already sold). Therefore, the module assumes that domains shorter than 8 characters are benign.

Domains	Remaining Characters	Score	Malicious
stackoverflow.com	....	1.00	False
caribbeancompr.com	............pr	0.86	False
dtaxnrxjgyqjfeq.com	d...nrxjgyqjfeq	0.2	True
yyibabelhlhscatjs.com	yy......hlh....js	0.58	True

TABLE I: Sample Domains

Table I shows four domains; the first two domains are valid and the second two are generated by DGA. Once all words in a domain are processed, then it is possible to give it a score based on the number of remaining characters and length of the domain. In Table I the characters of the detected words have been replaced with ".".

To create a comprehensive dictionary, English, French, Italian, Japanese, and Spanish are combined into one. Since not all of the words are in English, symbols and accent letters have been converted to the English alphabet.

C. Memory Forensics Module (MFM)

Based on our investigation of hooking mechanism for two different famous botnets, Zeus and Stuxnet it is clear that botnets in the real world need to hook different functions to achieve their goals (We considered these two because they are predecessors of many prominent botnets [38], [39]).

Stuxnet is a malicious computer worm that is exclusively designed to target programmable logic controllers that Duqu, Flame, and Gauss are its descendants [38]. To achieve the objective of hiding the threat related files Stuxnet hooks APIs from kernel32.dll and Ntdll.dll (e.g. FindFirstFileW, FindFirstFileExW, FindNextFileW, NtQueryDirectoryFile). Also it

```
{
  "rows": [
   [
      "Usermode", -> Hook mode
      "Inline/Trampoline", -> Hook type
      "explorer.exe", -> Victim process
      1664, -> Victim PID
      "CRYPT32.dll", -> Victim module
      2007498752, -> Address in memory
      2008117248, -> Address in memory
      "CRYPT32.dll!PFXImportCertStore at 0x77af02af", -> Function
      15316876, -> Hook Address
      "<unknown>", -> Hooking module
    ],
...
```

Fig. 2: Snippet of MFM output

hookes Ntdll.dll to monitor the requests for loading specially crafted file names (e.g. ZwMapViewOfSection, ZwCreateSection, ZwOpenFile, ZwCloseFile).

Zeus is a trojan horse malware package that runs on different versions of Microsoft Windows. While it can be used to carry out many malicious and criminal tasks, it is often used to steal banking information by man-in-the-browser keystroke logging and form grabbing. Furthermore Zeus Gameover, SpyEye, Ice IX, Citadel, Carberp, Bugat, Shylock and Torpig are well known descendants of this botnet.

Zeus is also used to install the CryptoLocker ransomware. The last descendant of the Zeus family, Pinkslipbot, is also employs hooking techniques. This varient of the Zeus botnet is reported in 2016 [36].

Zeus is spread mainly through drive-by downloads and phishing schemes. It hooks low level socket communication functions (e.g. recv, send), higher level WININET Internet communication routines (e.g. HttpQueryInfoA, HttpQueryInfoW, HttpSendRequestA, HttpSendRequestExA, HttpSendRequestExW, HttpSendRequestW), crypto routines (e.g. SealMessage, EncryptMessage), along with clipboard capturing methods (GetClipboardData).

Normally botnets hook functions for different reasons such as spying, getting confidential information and stealing critical information. Sometimes hooks are used to obtain alerts for the arrival of new sensitive data or to capture victims keystrokes, or to steal user's passwords. Botnets may also use hooks to get confidential information from web pages. Also botnets implant hooks to change critical system information to hide their presence in the system. Similarly, they can hook network functions to build a stealthy communication channel with their C&C servers [2]. For instance, Sality as one of the most famous botnets also is using hook methods to evade anti-viruses in its last version on 2016 [43]. This information shows that it is possible to define some heuristic techniques by using hooking behaviours of botnets to filter suspicious connections. Based on this research, a module has been proposed to process the memory dumps received from DCM, and detect available hooks for each virtual machine. Volatility Framework [1] is used to detect hooks from memory dumps. Volatility is an advanced open source memory forensic framework which proposes multiple plugins and functions for extracting data from memory snapshots and dumps. To find user mode and kernel mode hooks, BotViz uses APIhooks plugin. This plugin finds several types of hooks such as IAT (Import Address Table), EAT (Export Address Table), and Inline style hooks. The collected hooks' information for each VM are sent to AM in JSON format. Figure 2 illustrates a snippet of the JSON file.

D. Analyze Module (AM)

This module is responsible for detecting suspicious hosts and domains by processing the information gathered by the MFM and the DDM. The AM detects suspicious hosts based on abnormal hooks. It uses K-Means clustering algorithm to cluster hosts based on the hooking behaviours. The AM encodes each host to a feature vector by using JSON files received from the MFM. To create this feature vector, for each hook object in each host, it creates a tuple which includes five elements hook mode, hook type, victim process, victim module, and function. Figure 2 illustrates one tuple for the host which includes "User-mode", "Inline/Trampoline", "explorer.exe", "CRYPT32.dll", "CRYPT32.dll!PFXImportCertStore".

After AM constructs all tuples for all hosts, it creates a sorted list of unique tuples from all hosts. By using this list AM can construct a feature vector for each host (an array of 0s and 1s in this case), by putting "1" for the available tuples in the host and putting "0" for missing tuples. To distinguish benign and suspicious clusters a ground truth should be defined which is the feature vector created from a benign host. Also, user can define multiple ground truths. At the end, AM uses K-Means with Hamilton distance function to create clusters of hosts and ground truth(s).

Now, AM can classify hosts which are in the same group with ground truth(s) as benign hosts and the rest, which are not in the same groups, as suspicious hosts. Also, AM would flag domains which need more consideration based on their relation with suspicious hosts. AM will flag any domains which are requested exclusively by all suspicious hosts, also it ignores the 10,000 top domains of Alexa. (The first the 10,000 Alexa domains have been considered as benign [6])

AM also can detect suspicious domains based on two other methods. The first method is based on the results of DDM, if DDM recognizes any domains as a random string, AM would flag that domain as a suspicious one. Furthermore to detect fast-flux botnets AM uses TTL values in DNS logs, and if it is less than or equal to a threshold (threshold is 0), it classifies this domain as a distrustful entity. At the end AM sends suspicious hosts and domains to the Visualization Module for visualizing botnets.

E. Visualization Module (VM)

This module helps the users to detect anomaly patterns. By identifying patterns and anomalies the user can gain new knowledge and insight for further explorations. Also, visualization is very efficient and effective at communicating information. BotViz design is inspired by the Shneidermans information visualization mantra [13]: "overview first, zoom and filter, then details on demand". By selecting each domain the system shows the hosts which are connected to that specific domain and positions of those hosts in the network layout.

Furthermore, a user can check the details to gain more insight on suspicious hosts or domains. For example, the user can see details of suspicious hooks or random domains which cause problems.

IV. Experiments

A test environment has been created to demonstrate the ideas outlined in this paper. The main goal of this environment is to focus on showing the possibility of visualizing botnets with the proposed framework and also demonstrate the feasibility of distinguishing infected hosts from benign ones. For hardware specification, we used a desktop machine with 16 GB of RAM and an Intel core i7-4790 processor. Also, VirtualBox is used to establish a virtualized environment, with Ubuntu as the host operating system and Windows as the guest operating system.

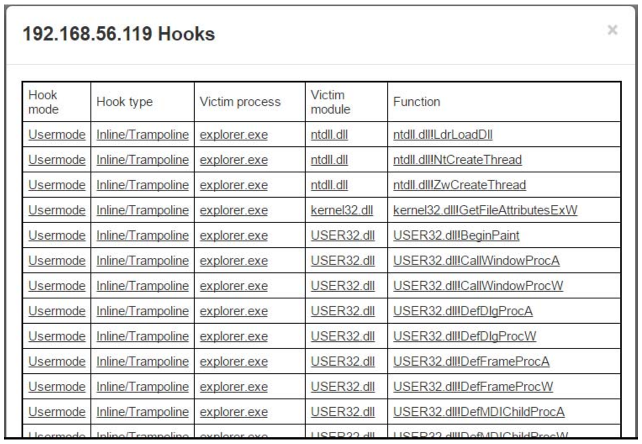
Based on the proposed model, DCM collects memory dumps from virtual machines by using VirtualBox SDK, and MFM processes memory dumps using Volatility Framework 2.4 to extract the hook information. The AM uses Python Scikit-learn library for clustering the hosts (K = 2) and detecting suspicious hosts. For visualization section the “D3.js” library has been used to visualize hosts and domains.

Ten virtual machines where deployed as guest operating systems. The Zeus botnet where used to infect three virtual machines out of ten. The reason behind selecting the Zeus botnet is because the source code leaked and it is available for download. Due to stochastic behaviors of botnets the best botnet to analyse is the one owned by researchers. Also, as mentioned in the introduction dozens of famous botnets are generated from this leaked source code. The Zeus source code [40] contains two important modules agent builder and C&C server (PHP). The first one is used to get the configuration settings and generate one malicious file to infect victims. The second one is a control panel for bot administrator which communicates with bots also.

To evaluate the proposed DDM, six real DGAs [9] have been used to generate a list of malicious domains (Torpig, ZeusBot, Cryptolocker, Necurs, Symmi and Ranbyus). It takes the name of the algorithm and date as a seed and generates random domains using that algorithm. 4155 malicious domains have been generated in one day (they use the date as their initial seed) by using these algorithms, are used to calculate the accuracy of DDM.

To measure the false positive rate of DDM, the top 10k domains gathered from Alexa [8] are used. Note that domains that are smaller than 8 characters are filtered from this dataset. The reason is that correctly classifying small domains is nearly impossible. For example, Ebay is a legitimate but meaningless domain. Furtunately, removing these domains cannot bring our approach under question since the majority of domains with length equal or less than 8 characters are already reserved [6]. By considering the mentioned filtering, 4949 domains are removed and the size of the Alexa dataset reduced to 5051 domains.

Also, a simple simulator was developed to simulate DNS requesting behavior of the Torpig botnet, to request set of domains based on Torpig algorithm [9].



192.168.56.119 Hooks

Hook mode	Hook type	Victim process	Victim module	Function
Usermode	Inline/Trampoline	explorer.exe	ntdll.dll	ntdll.dll!LdrLoadDll
Usermode	Inline/Trampoline	explorer.exe	ntdll.dll	ntdll.dll!NtCreateThread
Usermode	Inline/Trampoline	explorer.exe	ntdll.dll	ntdll.dll!ZwCreateThread
Usermode	Inline/Trampoline	explorer.exe	kernel32.dll	kernel32.dll!GetFileAttributesExW
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!BeginPaint
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!CallWindowProcA
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!CallWindowProcW
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefDlgProcA
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefDlgProcW
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefFrameProcA
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefFrameProcW
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefMDIChildProcA
Usermode	Inline/Trampoline	explorer.exe	USER32.dll	USER32.dll!DefMDIChildProcW

Fig. 3: Hooks

Furthermore, for retrieving longitude and latitude of an IP address to show that on the world map, the Geoip2 Python package has been used. This package uses the free MAXMIND web service.

V. Experimental Results and Analysis

In this section we present and analyze the results of our experiments. Three hosts where infected by Zeus botnets and one with the Torpig simulator (it just generates domain requests). Figure 4 illustrates BotViz visualization for the post infection phase. As can be seen, there are 4 different suspicious hosts. The Three in red (double underlined) are hosts with abnormal hooking behavior and all connected to a fake C&C. The other one which is orange (underlined) shows three algorithmic generated domains and the associated host. The system detects C&C and unb.ca, because all of the red hosts connect to these domains.

The system uses red color (dangerous) to show the C&C because just suspicious hosts connected to this domain and also it uses black color for unb.ca because other hosts in the network also connected to this domain so it can not be counted as a suspicious domain and just shows this domain to inform the administrator (furthermore consider that BotViz filter the top 10,000 domains of Alexa and unb.ca is not in the top 10,000 domains of Alexa). Besides, the system provides network layout and geographical map of the domains to present situational awareness for administrators to detect parts of organization which are more vulnerable against botnets and also the location of attackers.

As Figure 4 shows, a parallel coordinates illustrate suspicious domains and IPs (Part A), a network topology presents the suspicious hosts (Part B), and a world map locates malicious domain’s owners’ position for the user (Part C). Also a user can get further insights of suspicious processes and hooks (details on demand) for each hosts (figure 3). The Zeus infected hosts have 267 unique hooks and benign hosts have 2 unique hooks. Botviz is a framework to detect botnets based on DNS logs and memory forensics. To our knowledge Botviz is the first system which uses memory forensics and DGA detector module to visualize botnets. To detect botnet by using memory forensics, BotViz is better than the previous works

TABLE II: Performance of DDM in case of TPR and FPR

Families	**Necrus**	**ZeusBot**	**Cryptolocker**	**Torpig**	**Symmi**	**Ranbyus**	**All**
Number of samples	2048	1000	1000	20	64	40	4172
True positive	0.847	0.998	0.982	1.00	0.67	1.00	0.923
Some false negatives	lowusoheu.com oxegnusaen.com cuprybmeatskye.org keygtobetheld.es						
Some false positives	aljazeera.tv peyvandha.ir munrvscurlms.com uludagsozluk.com						

(none of them used hook detection system to detect botnets and did not provide any datasets for comparison), EyeCloud [12] just uses six static heuristics and is mainly focused on detecting changes on system policies (for example it detects disabled firewall) which can cause high false positive rate and low accuracy.

Frank Y.W *et al.* [11] proposed statically hidden process measurement by using Memoryze tool [33] which was not enough to detect botnets. For visualizing botnets there are some studies which are mentioned in the previous works section, BotViz is superior to them in many ways. For example, they did not use DGA detector based module or hook detection system to detect botnets. Moreover, they did not use any host-based information to fortify their visualizing system. Our proposed visualization system provides a hybrid framework to visualize botnets. Just [34] and [35] present a visualization system which visualizes processes on each hosts and their connections to external IPs (it means that they use some sort of host-based information), without providing any analytics.

As Table II shows, DDM has a high detection rate (0.935) and low false positive rate (0.02). Furthermore, some of the false positives such as “aljazeera.tv” contains words from other languages (which we did not included their dictionary, here “aljazeera” means Algeria in Arabic). By adding more dictionaries to DDM, it can improve its detection rate and reduce the false positive rate.

One of the possible counter moves for malware authors to evade DGA detectors is to generate domain names that are pronounceable yet not in the dictionary and much more likely to be available for registration. While in this case most of the previous works cannot detect random domains efficiently, our proposed approach is using a dictionary to label domains and therefore immune to this category of evasions. Also, one could employ a dictionary-based DGA and uses words to generate domains. But, the problem for this evasion arise when they want to register the domain at a domain registrar. Hence, random domains generated with a dictionary based DGA would be long enough to be easily detected. Since the task of detecting DGA can be divided into independent subtasks, DDM has the potential to distribute processes over multiple processors with near linear speedup.

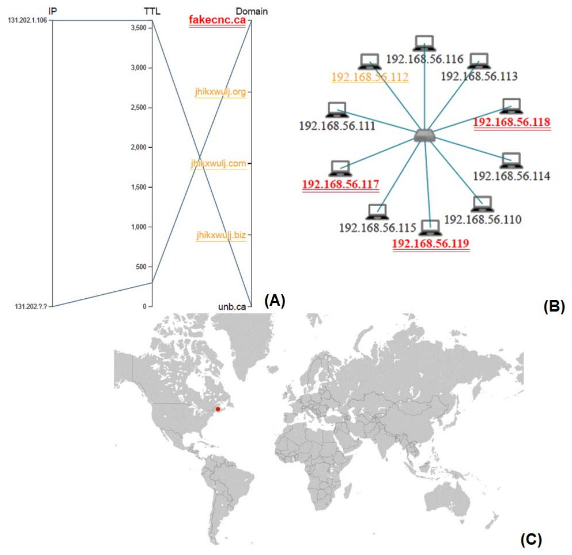


Fig. 4: BotViz visualization

VI. Conclusion

With the increase in complexity of network attacks through botnets, current security monitoring tools will be not suitable for efficient botnet detection. Hence, a tool that can detect existence of bot in a network is highly needed. The proposed framework “BotViz”, provides a hybrid visual approach for botnet detection in small to medium size networks. To the best of our knowledge, it is the first botnet visualization tool which uses suspicious hooks on the hosts to empower its botnet detection algorithm. Also BotViz uses its own DGA detector algorithm to increase the detection rate of botnets. Likewise BotViz framework is capable but not limited to run on cloud environments (it can use virtual machines memory dump to detect botnets), furthermore if Data Collector module changes (gathering memory dump section) it can run on non-cloud environments too. This work does not consider P2P botnets which are flourishing nowadays. BotViz could have false-positives if a legitimate application uses many unusual hooks (like anti-malwares), by adding these softwares to the ground truth’s memory image the problem could be solved but it affects the accuracy rate.

Acknowledgment

The authors generously acknowledge the funding from the Atlantic Canada Opportunity Agency (ACOA) through the Atlantic Innovation Fund (AIF) and through grant from the National Science and Engineering Research Council of Canada (NSERC) to Dr. Ghorbani.

References

[1] Volatitlity foundation, https://goo.gl/fDVSj1, Retrieved October 31 , 2016
[2] Liang Z, Yin H. and Song D., HookFinder: Identifying and understanding malware hooking behaviors, Proceeding of the 15th Annual Network and Distributed System Security Symposium (NDSS’08), 2008
[3] P.A. Porras, H. Saidi, and V. Yegneswaran, ”An Analysis of Conficker’s Logic and Rendezvous Points,” SRI Technical Report, 2009
[4] P.A. Porras, H. Saidi, and V. Yegneswaran, ”Conficker C Analysis,” Technical report, 2009
[5] B. Stone-Gross, M. Cova, L. Cavallaro, B. Gilbert, M. Szydlowski, R. Kemmerer, C. Kruegel, and G. Vigna, ”Your Botnet is my Botnet: Analysis of a Botnet Takeover,” In ACM Conference on Computer and Communications Security (CCS), 2009

[6] Namazifar M., ”Detecting Randomly Generated Strings”, DEFCON23, 2015

[7] Sandeep Yadav, Ashwath Kumar Krishna Reddy, A.L. Narasimha Reddy, and Supranamaya Ranjan. ”Detecting algorithmically generated malicious domain names”, In Proceedings of the 10th ACM SIGCOMM conference on Internet measurement (IMC ’10), 2010

[8] Alexa - Top websites on the web. https://goo.gl/vc1y - Retrieved October 31, 2016

[9] DGA Collection - A collection of known Domain Generation Algorithms. https://goo.gl/M9itmf - Retrieved October 31, 2016

[10] R. Begleiter, Y. Elovici, Y. Hollander, O. Mendelson, L. Rokach and R. Saltzman, ”A fast and scalable method for threat detection in large-scale DNS logs,” Big Data, 2013 IEEE International Conference on, pp. 738-741, 2013

[11] Law, Frank Y. W., Chow, K. P., Lai, Pierre K. Y. and Tse, Hayson K. S., ”A Host-Based Approach to BotNet Investigation?”, ”Digital Forensics and Cyber Crime: First International ICST Conference, ICDF2C, pp. 161-170, 2010

[12] M. R. Memarian, M. Conti and V. Leppnen, ”EyeCloud: A BotCloud Detection System,” Trustcom/BigDataSE/ISPA, 2015 IEEE, Helsinki, pp. 1067-1072, 2015

[13] Shneiderman, B. ,The eyes have it: A task by data type taxonomy for information visualizations. Proceedings IEEE Symposium on in Visual Languages, pp. 336-343, 1996

[14] Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for dns. In: USENIX security symposium. pp. 273290, 2010

[15] Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: Finding malicious domains using passive dns analysis, NDSS , 2011

[16] Yadav, S., Reddy, A.N.: Winning with dns failures: Strategies for faster botnet detection. Security and privacy in communication networks. Springer, 2012

[17] Antonakakis, M., Perdisci, R., Nadji, Y., Vasiloglou II, N., Abu-Nimeh, S., Lee, W., Dagon, D.: From throw-away traffic to bots: Detecting the rise of dga-based malware. USENIX security symposium , 2012

[18] Zhou, Yong-lin and Li, Qing-shan and Miao, Qidi and Yim, Kangbin ,DGA-Based Botnet Detection Using DNS Traffic. Journal of Internet Services and Information Security (JISIS), vol. 3, pp. 116-123, 2013

[19] Mowbray, M., Hagen, J.: Finding domain-generation algorithms by looking at length distribution. In: IEEE International Symposium on Software Reliability Engineering Workshops, 2014

[20] Schiavoni, S., Maggi, F., Cavallaro, L., Zanero, S.: Phoenix: Dga-based botnet tracking and intelligence in detection of intrusions and malware, and vulnerability assessment, pp. 192211. Springer, 2014

[21] Krasser, S., Conti, G., Grizzard, J., Gribschaw, J., and Owen, H., Real-time and forensic network data analysis using animated and coordinated visualization, In Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop, pp. 42-49, 2005

[22] Cremonini, M., and Riccardi, M., The Dorothy project: an open botnet analysis framework for automatic tracking and activity visualization, IEEE European Conference on Computer Network Defense (EC2ND), pp. 52-54, 2009

[23] Mansmann, F., Fischer, F., Keim, D. A., and North, S. C., Visual support for analyzing network traffic and intrusion detection events using TreeMap and graph representations, In Proceedings of the Symposium on Computer Human Interaction for the Management of Information Technology, p. 3, 2009

[24] Shahrestani, A., Feily, M., Ahmad, R., and Ramadass, S. , Architecture for applying data mining and visualization on network flow for botnet traffic detection, Computer Technology and Development, pp. 33-37, 2009

[25] Shahrestani, A., Feily, M., Ahmad, R., and Ramadass, S., Discovery of invariant bot behavior through visual network monitoring system, In Fourth International Conference on Emerging Security Information, Systems and Technologies, pp. 182-188, 2010

[26] Abdullah, K., Lee, C. P., Conti, G. J., Copeland, J. A., and Stasko, J. T., IDS RainStorm: Visualizing IDS Alarms, p. 1, 2005

[27] Kim, I., Choi, H., and Lee, H., BotXrayer: Exposing Botnets by Visualizing DNS Traffic, In KSII the first International Conference on Internet (ICONI), 2009

[28] Zhao, Y., Zhou, F., Fan, X., Liang, X., and Liu, Y., IDSRadar: a real-time visualization framework for IDS alerts, China Information Sciences, vol 56(8), pp. 1-12, 2013

[29] Somayeh Soltani, Seyed Amin Hosseini Seno, Maryam Nezhadkamali and Rahmat Budirato, A Survey On Real World Botnets And Detection Mechanisms, International Journal of Information & Network Security (IJINS), Vol.3, No.2, pp. 116-127, 2014

[30] Punit Sharma, Sanjay Tiwari, Anchit Bijalwan, Emmanuel Pilli, Botnet Detection Framework, International Journal of Computer Applications, Volume 93 No.19, 2014

[31] V. R. Kebande and H. S. Venter, ”A cognitive approach for botnet detection using Artificial Immune System in the cloud,” Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2014 Third International Conference on, Beirut, pp. 52-57, 2014

[32] Jonghoon Kwon, Jehyun Lee, Heejo Lee, Adrian Perrig, PsyBoG: A scalable botnet detection method for large-scale DNS traffic, Computer Networks, Volume 97, Pages 48-73, 2016

[33] Memoryze tool from FireEye, https://goo.gl/7d6lQW - Retrieved October 31, 2016

[34] G. Fink, P. Muessig, and C. North, Visual Correlation of Host Processes and Network Traffic, Proc. IEEE Workshop Visualization for Computer Security (VizSEC 05), pp. 11-19, 2005

[35] G. Fink, V. Duggirala, R. Correa, and C. North, Bridging the Host-Network Divide: Survey, Taxonomy, and Solution, Proc. 20th USENIX Conf. Large Installation System Administration, pp. 247-262, 2006

[36] McAfee malware report - https://goo.gl/jMs4Au - Retrieved October 31, 2016

[37] Mukosaka, S., and Koike, H., Integrated visualization system for monitoring security in large-scale local area network, 6th International Asia Pacific Symposium on Visualization APVIS’07, pp. 41-44, IEEE, 2007

[38] Bencsth, B., Pk, G., Buttyn, L., & Felegyhazi, M. . The cousins of Stuxnet: Duqu, flame, and gauss. Future Internet, 4(4), 971-1003, 2012

[39] Different Zeus variants- https://goo.gl/ux1xOA - Retrieved October 31, 2016

[40] Zeus source code, https://goo.gl/wPy5r - Retrieved October 31, 2016

[41] R. Ball, G. A. Fink, and C. North, Home-centric Visualization of Network Traffic for Security Administration, in Proceedings of the 2004 ACM Workshop on Visualization and Data Mining for Computer Security, VizSEC/DMSEC 04, pp. 5564, 2004

[42] Shiravi H, Shiravi A, Ghorbani AA. A survey of visualization systems for network security. IEEE Transactions on visualization and computer graphics. 18(8):1313-29, 2012

[43] Sality Bot, https://www.microsoft.com/security/portal/threat /encyclopedia/Entry.aspx?Name=Win32/Sality, Retrieved October 31, 2016

[44] Yin, Heng, et al. ”Panorama: capturing system-wide information flow for malware detection and analysis.” Proceedings of the 14th ACM conference on Computer and communications security. ACM, 2007

[45] Inoue, Daisuke, et al. ”Malware behavior analysis in isolated miniature network for revealing malware’s network activity.” Communications, 2008. ICC’08. IEEE International Conference on. IEEE, 2008

[46] Kirat, Dhilung, Giovanni Vigna, and Christopher Kruegel. ”BareBox: efficient malware analysis on bare-metal.” Proceedings of the 27th Annual Computer Security Applications Conference. ACM, 2011

[47] Song, Dawn, et al. ”BitBlaze: A new approach to computer security via binary analysis.” International Conference on Information Systems Security. Springer Berlin Heidelberg, 2008

[48] Wang, Zhi, et al. ”Countering persistent kernel rootkits through systematic hook discovery.” International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2008

[49] Xuan, Chaoting, John Copeland, and Raheem Beyah. ”Toward revealing kernel malware behavior in virtual execution environments.” International Workshop on Recent Advances in Intrusion Detection. Springer Berlin Heidelberg, 2009

[50] Riley, Ryan, Xuxian Jiang, and Dongyan Xu. ”Multi-aspect profiling of kernel rootkit behavior.” Proceedings of the 4th ACM European conference on Computer systems. ACM, 2009

[51] Wang, Zhi, et al. ”Countering kernel rootkits with lightweight hook protection.” Proceedings of the 16th ACM conference on Computer and communications security. ACM, 2009

[52] Liang, Zhenkai, Heng Yin, and Dawn Song. ”HookFinder: Identifying and understanding malware hooking behaviors.” Department of Electrical and Computing Engineering, 2008

[53] Yin, Heng, et al. ”Hookscout: Proactive binary-centric hook detection.” International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. Springer Berlin Heidelberg, 2010