Adaptable Exploit Detection through Scalable NetFlow Analysis

Alan Herbert
Rhodes University
Grahamstown, RSA
Email: g09h1151@campus.ru.ac.za

Barry Irwin
Rhodes University
Grahamstown, RSA
Email: b.irwin@ru.ac.za

***Abstract*—Full packet analysis on firewalls and intrusion detection, although effective, has been found in recent times to be detrimental to the overall performance of networks that receive large volumes of throughput. For this reason partial packet analysis technologies such as the NetFlow protocol have emerged to better mitigate these bottlenecks through log generation. This paper researches the use of log files generated by NetFlow version 9 and IPFIX to identify successful and unsuccessful exploit attacks commonly used by automated systems.**

These malicious communications include but are not limited to exploits that attack Microsoft RPC, Samba, NTP (Network Time Protocol) and IRC (Internet Relay Chat). These attacks are recreated through existing exploit implementations on Metasploit and through hand-crafted reconstructions of exploits via known documentation of vulnerabilities. These attacks are then monitored through a preconfigured virtual testbed containing gateways and network connections commonly found on the Internet. This common attack identification system is intended for insertion as a parallel module for Bolvedere in order to further the increase the Bolvedere system's attack detection capability.

***Index Terms*—Digital forensics, Network security, Intrusion detection**

I. Introduction

This research builds on the module repertoire that is compatible with the Bolvedere platform currently in development by the authors of this paper. The Bolvedere platform is a highly adaptable and scalable NetFlow processor intended for distributed identification of malicious network activity. All modules developed for this platform run concurrently, be it remote to the Bolvedere host system or within it. The implementation brought forward in this research is in respect to development of a new module to run on the Bolvedere system [1].

A. Problem Statement

Issues that need to be addressed within this area of research are that of latent security in large scale networks. Included in this is network traffic monitoring to protect users from vulnerabilities that can cause their systems to perform malicious tasks within the network. Exploitation of these vulnerabilities can lead to inclusion within botnets, proxying of malicious data, and other malicious activities that lead to disruption of natural flow of data within the Internet.

For this reason automated systems are required to monitor interactions within networks that source traffic onto the Internet. These systems should then provide a feedback mechanism to better detect and mitigate malicious activities introduced into the Internet.

B. Research Goals

This research in its entirety aims to develop a modular platform for which modules that process network flows can interface in order to discern events on a network. Adaptability is taken into account in the form of system resources required to run a Bolvedere system. Bolvedere is able to execute itself on a single host machine that is Linux OS (Operating System) compatible as well as further scale out over multiple threads, multiple processes, multiple processors and multiple separate physical hosts. This adaptability and scalability also includes support for multiple languages as well as hardware technologies such as GPUs (Graphical Processor Unit) and FPGAs (Field-Programmable Gate Array). Furthermore, this scalability ensures the ability for Bolvedere to take on the task of Internet level network flow discernment as to whether a network flow is malicious or not.

The first question proposed by this research was if one could use NetFlow logs to detect a malicious exploit. With this question in mind this research's first goal was to collect NetFlow logs generated from network flows created by malicious network exploits. Sections IV-B1 and IV-B2 discuss how these resultant NetFlow logs were observed and what distillation process occurred when attempting to produce rule sets from different exploits.

The application of these rule sets within a Bolvedere module is then brought to light in Section IV-C which discusses the accuracy of such an automated mechanism, as well as where this module falls short and how these failings can be mitigated.

This paper begins with a literature review in Section II which deals with background knowledge required to better understand this research and why it is necessary. Following this, Section III discusses how this Bolvedere module was implemented through an understanding of the tools and libraries used within it. Finally, results are discussed in Section IV and a final conclusion is presented in Section V.

II. Literature Review

As the main data descriptor of this research's implementation is based on NetFlow [2], the first topic dealt with in this paper is the aforementioned technology. Once this technology is outlined this paper then moves on to discuss and explain current malicious activities that occur on the Internet and how they are accessed and acted upon.

A. NetFlow

The NetFlow protocol is best described as a means of logging network flows that pass through a flow monitoring

978-1-5090-2473-8/16/$31.00 ©2016 IEEE

device in a communication pair's route. A network flow is defined as a unidirectional connection and communication between a host and any other host, multicast group, or broadcast domain in the form of a sequence of packets [3]. A flow monitor implementing the NetFlow protocol can collect fields out of these communications, write them into predefined fields (restricted either by NetFlow protocol version or by use of a known template) and then transmit them to a logging host for analysis or storage [4]. There have been multiple versions of NetFlow with wide-spread support over multiple firewall and routing devices on the Internet.

The need to update the NetFlow protocol over the years arose from multiple factors. First, the addition of IPv6 (Internet Protocol version 6) [5] that was brought about by the IP address exhaustion [6] of the IPv4 (Internet Protocol version 4) [7] space required amendments to be added to the NetFlow protocol. Furthermore, the need for better use of network resources grew as the amount of traffic passing through flow monitor points increased. Finally, the requirement to adapt these records to one's needs gave way to updating the NetFlow protocol to give users the ability to break out of the predefined logging fields determined by older versions of NetFlow into a dynamic space that allows for field collection through a user created predefined and distributed template [4]. This method of log collection also has additional unused record space for later allocation of new network protocols released at future dates.

Major updates to the original NetFlow standard have included the addition of new fields and further standardisation and this resulted in version 5 of the protocol. This version allowed for logging subnet masks and AS (Autonomous System) numbers [8]. Version 8 saw inclusion of aggregation of records that were first defined in version 5 [9]. More recently, version 9 continued to build on the freedom brought forth by version 8 through the addition of the template packet to the NetFlow protocol. This template packet allowed one to define the fields to be logged in a record and the order in which they are logged from a flow [4].

These templates are coupled with template identification numbers that allow for use of multiple templates and is defined by the 2-byte-long template identification field within the NetFlow protocol. Although IDs 0 through to 255 are reserved for use by specific predefined flow templates, template identification numbers 256 through to 65535 are available for public use; a fairly large template space. This template count further extends the memory requirements of these devices and most devices supporting NetFlow version 9 limit the number of templates that can be stored to a count far less than the available 65535 due to memory and performance limitations [2].

B. Malicious Attacks

A malicious event acted upon something or someone is simply defined as an action with intent to do harm to that entity. Within the Internet these entities are commonly endpoint hosts. These hosts are typically targeted in order to collect information or prevent other entities on the Internet from accessing the information provided by the host; be it private or public. This section intends to focus on two main reasons for gaining access to a system through malicious means and these are denial of service and information theft.

1) Denial of Service: The goal of a DDoS (Distributed Denial of Service) attack is the same as that of a (DoS Denial of Service) attack in that they both aim to bring down a service that exists on a network. The key difference between a DDoS attack and a DoS attack is that a DDoS attack uses multiple physical source hosts rather than a singular host. These hosts usually exist within a botnet [10]. Note that a DoS attack can appear to be a DDoS attack through the spoofing of multiple IPs; this makes detecting a DDoS attack difficult.

2) Identity Theft: Identity theft refers to any form of theft that enables the thief to perform actions of that of the victim while holding all the credentials needed to be identified as that victim. This on the Internet includes usernames and passwords, private identifying information and man-in-the-middle attacks to gain access to tokens passed between hosts in order to gain access to the victims session [11].

3) Information Theft: This is simply stealing information that is private, be it from a single person, group or company. This information is usually targeted and sensitive to public viewing or viewing by a competitor.

4) Information Destruction: On the other end of the spectrum when compared to information theft, information destruction can be performed through ransomware or simply destruction of a targets information store, be it deletion of a database or project. Ransomware aims to encrypt data with a key that the attacker holds and typically requires payment in order to retrieve the key to decrypt one's data with. If one fails to acquire this key, be it because of the event timing out or one's unwillingness to pay for the data, then the data that is encrypted is effectively as good as destroyed [12]. Complete outright destruction of data is usually a method used to bring down companies as if a company were to lose its database of operations, it would be as if the company never existed from that point on. It is note worthy that the cost of data recovery in the event of data destruction can be enough to put a company under in itself [13].

C. Common Attack Vectors

There are multiple ways to perform analysis on potential targets and multiple methods to go about executing an attack on these targets. These all fall into specific categories of which the ones this research is aimed to analyse and identify are listed below.

1) Brute-Force Attacks: This form of an attack is applicable occurs in multiple forms and is defined by an attack method having little to no intelligence [14]. The method of this attack is to simply to try every combination from start to finish of the attack space until a solution is found. As this attack intends to attempt every combination as input to a system in order to gain access to information held by the system, this approach is time consuming and thus attempts to leverage high throughput hardware in order to achieve this task in a acceptable time [15] (this hardware includes GPU through the use of CUDA and OpenCL).

This form of attack can however be broken down into two subsets, these are offline and online brute-force attacks. A offline brute-force attack is typically performed against a data that exists on the attackers storage devices and is locally accessible; these include databases of hashed passwords. For this example an attacker would typically attempt to recover the password used to generate a hash through generating every

possible input for the respective hashing algorithm until the output matches that of the hash that one is trying to recover a password for. At this point the input used is the original password that the attacker is looking for.

An online brute-force attack refers to an attack on a remote service or system. Typically this is done through guessing login credentials until access is gained. This kind of approach can be seen used on systems that require a username and password such as SSH or a website. One would try to generate combinations of username and password until a successful login occurs.

2) Vulnerability Exploitation: Humans are not perfect and as hardware and software are developed by people, there are imperfections introduced into the systems. These imperfections lead way to unexpected behaviour that when acted upon lead to results that fall outside of the systems intended operation. These result can range from simple errors in output to code being remotely uploaded and executed on the system or the system being shut down completely.

Malware such as Blaster Worm [16], Conficker [17] and SQL Slammer [18] use these vulnerabilities to upload and execute themselves upon remote hosts. These vulnerabilities exploited by these malware include MS03-026 [19], MS03-039 [20], MS08-067 [21] and the Microsoft SQL Server Resolution Service [18]. Even though these are well known vulnerabilities that have long since been fixed there are still many existing systems on the Internet that are still vulnerable to these attacks due to improper maintenance of said systems. These iconic attack methods were chosen to show their age (dating back to January 2003) and further exclaim the neglect shown by some system administrators.

3) Social Engineering: There are many methods both physical and digital of social engineering but this text will focus on two methods used on the Internet to explain what it is. These methods are phishing and baiting. The idea behind social engineering is to attack the human psyche through misleading someone to act in a way they would usually not, or exploiting one's natural characteristics into acting upon something that should not be acted upon. The former is prevalent in phishing where the latter is used in baiting.

Phishing gets someone to give up private information by pretending to be something or someone it is not. A simple example of a phishing attack is a website pretending to be an existing bank that it is not. If it were to successfully trick someone into entering their banking details the third-party that set up the website would then gain access to that private information [22].

Baiting on the other hand relies on a human trait known as greed. If someone really wants something that they can't get and a malicious source offers that something, it opens up a vector of attack. This is commonplace in pirated software available on the Internet. Someone wants to use a piece of software that they have to pay for, why not offer it for free and attach malware to the executable. Why even hide behind the faade of a executable when one could just rename the malicious executable and change the display icon to mimic that of the original software; once the user clicks run it doesn't really matter what the user sees next as the attack is already successful [23].

D. Availability of Attacks

The large number of malicious attacks occurring on the Internet on daily base is due to two reasons. The first is the ease of which one can perform these attacks. For the most part someone with little to no knowledge of how a vulnerability is exploited on a network can simply get hold of tools and scripts to run at the press of a button that one points at a target. The second is that many malicious attacks are automated, be it via botnet [10] or by a malicious preconfigured system.

There is a market for these systems, botnets and zero day attacks (a vulnerability that is yet to be exploited and is unknown to the vendor) and these can fetch a high price depending on the capabilities of the exploit, however there are freely available tools for configuring and performing malicious exploits of systems on a network [24]. These exploits are for the most part well known and fixed and for a target to fall victim to these exploits is due to their own negligence in terms of keeping their system up to date. This section will deal with the two common ways exploits are performed in a legal penetration testing environment.

1) Metasploit: This software suit that is directed at penetration testers to test the security of networks and users on it. It houses a wide variety of tools that allow for assessment of software and systems running on a network, as well as the awareness of the users on a network through features like the generation of phishing campaigns to test users on a network. Coupled with these features is a database of fully functional exploits that are known to the penetration testing community. This means that one can install Metasploit, which is free, and launch these attacks on a target host on a network at one's leisure [25]. This can of course be for Metasploits intended purpose, that being security conciousness, or for malicious reasons.

For intended reasons Kali Linux exists as a penetration testing operating system that comes pre-installed with Linux based software that one would use for penetration testing of a system or network [26]. Furthermore, the Metasploit community supports this Linux distribution and thus regular updating of the distribution and tools on it is freely available.

2) Reuse of Code in the Wild: There is a need to understand existing malware on the Internet and for this collection and reuse of such malware in a safe environment, while monitoring the characteristics of the captured malware, can be invaluable in combating it [27]. There are multiple methods of capturing malware and the class of software that typically performs this action is referred to as a honeypot; although this is not the only way to capture a piece of malware. A honeypot acts as a vulnerable system in order to attract malicious attacks [28]. Any attacks that are targeted at the honeypot are then logged and any data upload is too. From this point one can set up an environment such as a virtual network, or if one has the resources a live, environment in which to rerun these malware and analyse their characteristics.

Other methods of malware collection range from malware sharing communities to collecting the remnants of uploaded scripts and programs to a server that may have resulted in a failed or successful attack. Either way, deletion of these malware are a loss to the community trying to combat these forms of malicious attack and one should attempt to pass on the malware to a third-party that has a use for it (hopefully not malicious in nature).

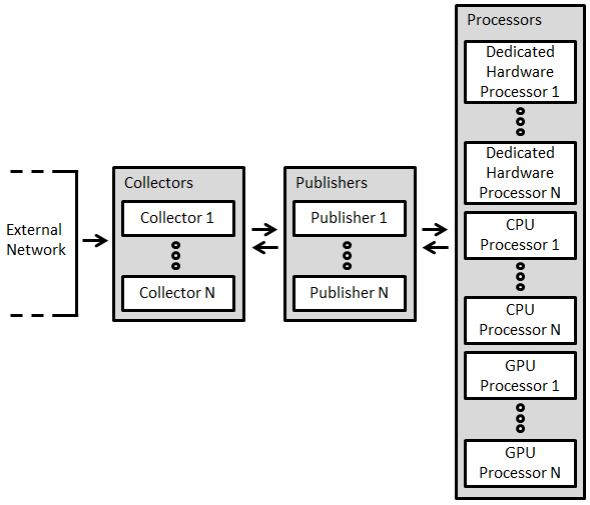

Fig. 1: Bolvedere System Overview

III. Bolvedere Module Implmentation

This research collects and analyses NetFlow logs in order to discern whether a network flow could have attempted a malicious attack or not. It is understood that a non-malicious network flow can give the same NetFlow log results as NetFlow log generated from a malicious flow. It was decided that dealing with false positives was better than dealing with false negatives in this system as this would mean that a successful attack would go unrecognised. The results of these true and false positives would both be presented to the user with all related information for the user to discern, be it through further processing or if the traffic is low enough by one's self, as this module is intended to identify malicious attacks and not to take mitigating action.

In order to achieve this and ensure adaptability and scalability, this module intended for use with Bolvedere required some careful planning. A short discussion on tools used and method of configuration will be pointed out in this section as to help the reader better understand how this intended adaptability and scalability was achieved.

A. Tools Used

The two major potential bottlenecks identified in this system was on the network interface and the rule set processor. Referring to Figure 1 with the understanding that this module is a processor in the Bolvedere system means that the network component of the system has to act in a distributed manner between a publisher and the connected processors. For this the use of a broadcast groups was used to allow for a single network packet sent by a publisher to arrive at multiple destinations.

The library used for handling these broadcast groups was ZMQ (Zero Message Queue). The ZMQ library gives access to a distributed networking model that makes use of sockets and broadcasting to allow for increased concurrency within a system. Furthermore, these transmitted messages ensure atomicity over the entire broadcast group. The ZMQ transport layer can be set to in-process, inter-process, TCP and multicast modes depending on whether the communications are happening within a process or between processes on a single host, or between processes on separate hosts [29]. Furthermore, ZMQ is a library that is supported by over 30 languages and the entirety of the protocol is documented. this means that no matter what processor module one intends to implement for Bolvedere, one can choose the best language for the job.

In order to quickly access the rule sets to discern whether a network flow is malicious or not, a SQL (Structured Query Language) database was used. There are many varients of SQL databases which range from a file on disk, as SQLite [30] implements, to entire databases loaded into RAM to achieve maximum throughput, as implemented MemSQL [31]. Given these implementations all share a common language, one can swap out the back-end of this module according to the needs and/or limitations of the host system.

The rest of the software which applies the rule set to incoming NetFlow logs of this module was written in C and a Python based prototype also exists.

B. Configuration

Three stepsC are required for configuration which are listed below:

- Select publishers that the module should listen to in order to receive processed NetFlow logs.
- Load in which rules the module should implement in order to discern these received NetFlow logs so that detection potential malicious activity can occur.
- Select a method of notification for when detection occurs.

The first point expanded upon will be that of publisher selection. A module in the Bolvedere system is allowed to subscribe to more than one publisher to receive its processed NetFlow logs from. The format in which the information arrives is predetermined at configuration time of a publisher and is intended to be in a format best suited for use by listening modules.

The second point is that of loading the rule sets. One of these malware detection modules may attempt to detect one or many forms of malware signatures. If the rule set gets too large and one wishes to break up the rule set into smaller chunks of work in order to better scale the systems ability to process this module, all one simply needs to do is start more of these modules that only deals with a subset of the entire rule set. Furthermore, these new modules can run concurrently with the rest of the modules on a separate physical host completely, or on a separate processor thread within the same host.

Lastly the method in which these findings are presented to the user is optional. Currently the options of receiving daily reports via an email or presenting findings to a terminal at runtime are available for selection. The use of a terminal allows piping of outputs from this module into other applications for further processing. Any option in between can trivially be implemented at a later stage, however the question of did this implementation achieve its goal did not rely on this functionality.

IV. Results

This paper sought to collect NetFlow logs generated by malicious network flows on a network and then analyse them for generating rule sets for use in a Bolvedere system module to detect further attempts of these attacks. For this reason this results section will be broken down into two major parts. The first part will deal with the collection and analysis of NetFlow

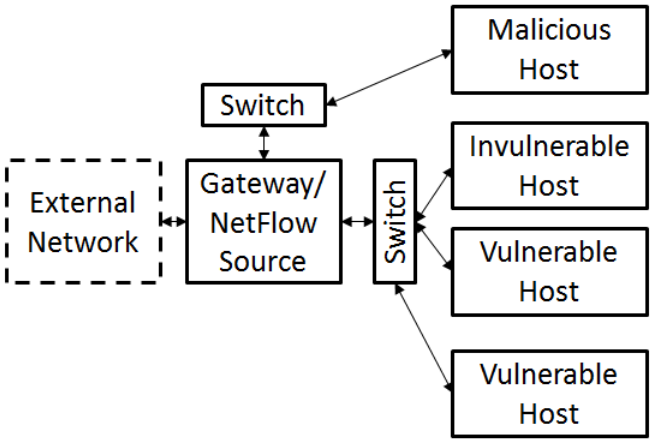

Fig. 2: Virtual Network Overview

logs generated by controlled malicious attacks in order to build rule sets to detect further attempts of these attacks. The second part of these result will be targeted at the running of the automated Bolvedere system module that implements these rule sets to detect repetitions of these recorded malicious attacks.

A. Environment

These tests are all performed within a virtual environment. A basic configuration of the virtual network this virtual environment uses can be seen in Figure 2. One can see that hosts are broken up into 4 groups and these are listed below with a short description:

- Malicious Hosts: These hosts launch malicious attacks on vulnerable and invulnerable hosts typically through the use of Metasploit or custom written code as to the documented exploitation attack vector.
- Vulnerable Hosts: These hosts are built to be vulnerable to a monitored attack.
- Invulnerable Hosts: These hosts are built to be made resistant to a monitored attack.
- Gateway/NetFlow Source: This host acts as a gateway to an external network (advertises it is connected to the Internet) and also runs the NetFlow log generator to store the network flows that pass through it. This host uses softflowd to generate all NetFlow logs [32].

One can observe that in order for the malicious host to communicate with a vulnerable or invulnerable system it has to first pass through the gateway system running softflowd. This means that the network flows generated between these hosts can be fully logged and analysed into rule sets at a later point.

B. NetFlow Logs and Rules Generated

The purpose of this section is to collect the NetFlow logs generated by softflowd when observing the network flows caused by malicious attacks. The defining features of these generated logs are then extracted by an automated rule generator and used to form rule sets that can discern further attempts of the attacks that generated the network flow. The rule generator simply observes a repeated attack and collects relevant metadata about the attack before generating a rule for that form of attack. The NetFlow logs generated that the automated rule generator observed have been tabulated and can be referred to in Tables I to VIII. These results were gathered over 6 iterations of which 3 iterations were designed to be successful and 3 were designed to fail. The failures did not show a large variance in results and so due to the space limitations of this paper have been omitted but will still be discussed later in this section. Also, due to the nature of networking technologies there is some variance in the collected results, this is handled through displaying the result as a range rather than a set value where necessary.

Terminology used in these results is explained below:

1) Attacker: The host that is implementing an exploitation.
2) Target: The host which the attacker is attempting to exploit.
3) Victim: A third-party that is affected due to an Attackers exploit.
4) A, B and C: These refer to randomly assigned ports by the operating system when a connection is created without being told to use a specific port.
5) N: This refers to all numbers after the last until process is terminated.
6) X and Y: These refers to counters of varying size relating to packet and byte counts.
7) Exploit: Used to define the stage of the exploit in which the exploitation is being attempted.
8) Payload: Used to define the stage of the exploit in which the payload is being transferred and executed on the system.
9) Runtime: Used to define the stage of the exploit in which the payload is running.

1) Results: Unsurprisingly the first point to note is that it is the attacker that always starts the communications in these exploits. The method is usually performed through fingerprinting a target to identify which services are running on a system (these logs are not interesting and so have been omitted). Once a vulnerable service has been identified the exploit then is executed and if successful the payload is then uploaded to the vulnerable host and an attacker gains access to the target in their chosen method. As these vulnerabilities are found in services running on a host and these services run on specific ports, it is noteworthy that these specific ports is what an exploit targets.

A significant point that arose when an exploit was repeated was that the initial NetFlow log's packet count, byte count and service port were consistent (the service port consistency is important as some services utilize multiple ports). This means that one can say that for a new NetFlow log between two hosts, if a set port is connected to that receives a set packet count with set total byte count, one should check that targeted host for an occurrence of an attack that is represented by this signature. Although one should also note that as a NetFlow log only contains the metadata of a network flow, a perfectly legitimate network flow could also cause this NetFlow log to be generated.

Some finer details to notice is that MS08-067 (exploitation of Microsoft RPC (Remote Procedure Call) service) exploitations tend to have their packet count and byte count vary more than exploits utilizing other vulnerabilities in these results. Another point is that the NTP (Network Time Protocol) monitor list attacks were generated using 3 separate monitor

TABLE I: NetFlow Logs Generated by a Successful ms08_067_shell Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	445	43 - 47	9900 - 10100
2	Exploit	target → attacker	445	A	42 - 44	7600 - 7700
3	Payload	attacker → target	B	Set in Exploit	8	695
N	Runtime	Bi-Directional	B/C	Set in Exploit	X	Y

TABLE II: NetFlow Logs Generated by a Successful ms08_067_vnc Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	445	43 - 47	9900 - 10100
2	Exploit	target → attacker	445	A	42 - 44	7600 - 7700
3	Payload	attacker → target	B	Set in Exploit	278	416549
N	Runtime	Bi-Directional	B/C	Set in Exploit	X	Y

TABLE III: NetFlow Logs Generated by a Successful java_rmi_server Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	1099	6 - 7	358
2	Exploit	target → attacker	1099	A	7	567
3	Payload	attacker → target	A	Set in Exploit	7	7400 - 7500
N	Runtime	Bi-Directional	A/B	Set in Exploit	X	Y

TABLE IV: NetFlow Logs Generated by a Successful distcc_exec Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	3632	7	656
2	Exploit	target → attacker	3632	A	4	276
3	Payload	attacker → target	B	Set in Exploit	4	216
N	Runtime	Bi-Directional	B	Set in Exploit	X	Y

TABLE V: NetFlow Logs Generated by a Successful samba_symlink_traversal Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	445	10	975
2	Exploit	target → attacker	445	A	8	790 - 800
N	Runtime	Bi-Directional	B	Set in Exploit	X	Y

TABLE VI: NetFlow Logs Generated by a Successful samba_usermap_script Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	139	7	733
2	Exploit	target → attacker	139	A	4	356
3	Payload	attacker → target	B	Set in Exploit	3	164
4	Payload	target → attacker	Set in Exploit	B	2	135
N	Runtime	Bi-Directional	B	Set in Exploit	X	Y

TABLE VII: NetFlow Logs Generated by a Successful unreal_ircd_3281_backdoor Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	Set in Exploit	3	164
2	Exploit	target → victim	Set in Exploit	A	2	135
N	Runtime	Bi-Directional	A	Set in Exploit	X	Y

TABLE VIII: NetFlow Logs Generated by a Successful ntp_mon_list Exploit

Flow Index	Exploit Stage	Direction	Source Port	Destination Port	Packet Count	Byte Count
1	Exploit	attacker → target	A	123	1	60, 90 or 234
2	Exploit	target → victim	123	A	up to 10	up to 4460

list request packets, these were of size 60, 90 and 234 as found out in the wild [33]. As this attack is UDP based reflection attack, the attacker did not receive any feedback as to success or unsuccess of their attack and instead only a response was generated by an NTP server to the victim which the attacker intended to DDoS. For this reason the attacker also requires the uses of a third-party discovery tool, such as ping, to see whether the victim was still reachable or not (these ping logs were not shown as they are not part of the exploit tested however did exist in the communications).

The failures of these exploits for the most part resulted in a TCP reset at some point in the exploit attempt. The resulting NetFlow logs depict this with an initial flow from the attacker with a response flow of 1 packet that is 46 bytes in length (this represents a TCP reset). The only two exceptions to this were the MS08-067 based attacks, which showed a response flow from the target before a follow up flow was generated in order to access the payload of which was responded to with a flow of the aforementioned TCP reset. The second was the NTP based attacks which because they were UDP based, showed no response to the exploit in any form.

2) Rule Sets Generated: Tables I to VIII in the Results section, Section IV-B1, are in the format of the rule sets that will be given to the Bolvedere module that will attempt to discern these exploits[1]. It is notable that these rules outlined by these tables require far fewer checks in an attempt to discern a network flow than deep packet analysis does; this is due to the fact that every packet in a network flow doesn't get analysed but rather the existence of a network flow. Coupling this with the sheer reduction in the amount of throughput the overall system has to handle as a NetFlow log only contains the metadata of a network flow. This allows for multiple NetFlow source nodes to sink their generated logs into a fewer hosts running Bolvedere than the equivalent amount of hosts required for a deep packet analysis solution.

C. Automated Module in Action

In order to check proper functionality and usability of this Bolvedere module, one has to provide control data for the results to be compared against. For this reason legitimate network traffic is required to the services running on the vulnerable host. In this testing, the legitimate connections and use of the services on the vulnerable host was performed by bots. These bots were programmed to perform simple tasks that required use of these services at random times ranging between 500 milliseconds and 10 seconds. One must note that the vulnerable target host was running every exploitable service in which the rule sets were generated for allowing for ease of testing[2]. Furthermore, Microsoft Windows services were made available on this system through use of a Windows

[1]The tables were developed this way to save space.

[2]This system is provided by RAPID7 and is available for download at https://information.rapid7.com/metasploitable-download.html

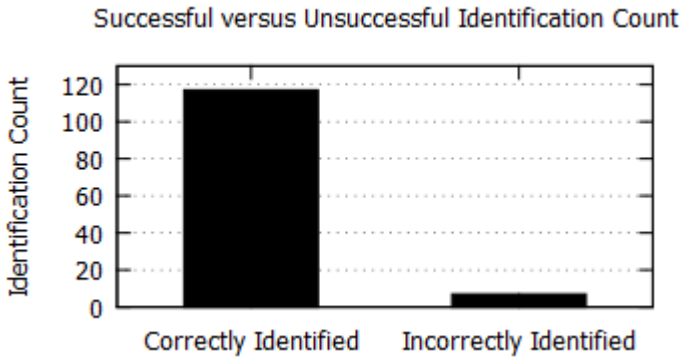

Fig. 3: Comparison of Success versus Failure of Network Flow Identifications

XP virtual machine runnining on the vulnerable host that was configured to bridge its network interface with that of the vulnerable host system. At runtime, the bots were first enabled to start communicating with the vulnerable host and then the attacks were manually launched and results observed through the terminal display of the Bolvedere module.

Listing 1: Terminal Output of Bolvedere Module

```
[10.42.0.45:45677 -> 10.42.0.33:445,
 Size:9991,
 Count:44]: Potential ms08_067_shell

[10.42.0.13:34782 -> 10.42.0.33:445,
 Size:15232,
 Count:113]: No malicious activity found

[10.42.0.68:40029 -> 10.42.0.33:123,
 Size:60,
 Count:1]: Potential ntp_mon_list

[10.42.0.13:37087 -> 10.42.0.33:139,
 Size:23011,
 Count:146]: No malicious activity found

[10.42.0.103:28928 -> 10.42.0.33:445,
 Size:18002,
 Count:174]: No malicious activity found
```

Testing occurred over 124 separate connections consisting of multiple network flows depending on the task at hand. The success or failure of a result was considered on a per connection bases and were discerned as to whether a connection was malicious by the NetFlow logs generated by the entire connection. Terminal output of this module can be referred to in Listing 1 which includes a false positive regarding the

detection of a ntp_mon_list attack. This was in fact a legitimate request for the monitor list from the NTP server. Referring to Figure 3 one can see the results produced by these 124 separate connections.

Of these 124 connections 117 were successfully identified as either a malicious or legitimate connection where the only 7 failures were false negatives produced when trying to determine whether a monitor list request from the NTP service was legitimate or part of a DDoS attack. This means that the rule set produced for this Bolvedere module is 94.355% accurate when attempting to discern the exploits recorded in Tables I to VIII under controlled test conditions.

These results suggest that detection of malicious activities when legitimate network flows closely resemble that of malicious flows becomes difficult. In the case of ntp_mon_list, this is because a legitimate request is used to exploit an amplification attack on a victim which is near impossible to detect against other legitimate requests. For this reason it is suggested that further revisions take into account previous connections made by an IP address, however memory usage should be considered before this step is taken.

Considering the high level of accuracy produced by this module when considering the given rule set and non-ntp_mon_list exploit and legitimate connections, these results hold promise into extension into detection of other malicious connections as well as extension into real world implementation. In all, the fact that no malicious connections were missed even though there were false positives means that this Bolvedere module has successfully achieved the goal set out by this research.

V. Conclusion

This research aimed to answer two questions, this being whether NetFlow logs can be used to discern malicious exploits and whether rule sets can be generated from these results and automated as a Bolvedere module. After execution of known exploits through a NetFlow node in a controlled environment, NetFlow logs were recorded and distilled into a rule set. Once this rule set was implemented within a Bolvedere module the accuracy of this module was shown to be 94.355% when discerning whether a network flow was legitimate or a specific malicious exploit. Although there were false positives, no executed exploits were recorded as false negatives. Given that every network flow containing a malicious exploit was detected by this Bolvedere module, this research was deemed successful and further development into its rule sets and fine tuning of the module itself shows promise for future iterations and use in live environments.

Acknowledgment

The authors wish to acknowledge the joint support of the Council for Scientific and Industrial Research (CSIR) and Rhodes University for the financial support and access to facilities for this research.

References

[1] A. Herbert and B. Irwin, "FPGA Based Implementation of a High Performance Scalable NetFlow Filter," in *Southern Africa Telecommunication Networks and Applications Conference*, D. F. Otten and M. R. Balmahoon, Eds., 2015, pp. 177 – 182.

[2] Cisco. (2003) NetFlow V9 Export Format. Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://tinyurl.com/ond3pe8

[3] D. R. Kerr and B. L. Bruins, "Network flow switching and flow data export," Jun. 5 2001, uS Patent 6,243,667.

[4] B. Claise, "Cisco systems NetFlow services export version 9," *IEEE Networking Group RFC*, 2004.

[5] S. E. Deering, "RFC 2460: Internet Protocol, version 6 (IPv6) specificat," 1998.

[6] G. Huston. (2014, February) IPv4 Address Report. Accessed 6th February 2014. [Online]. Available: http://www.potaroo.net/tools/ipv4/index.html

[7] J. Postel, "RFC 791: Internet Protocol," IETF, Tech. Rep., 1981.

[8] G. Huston. (2006) NetFlow Packet Version 5 (V5). Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://netflow.caligare.com/netflow_v5.htm

[9] ——. (2006) NetFlow Packet Version 8 (V8). Cisco Systems, Inc. Accessed 13th February 2015. [Online]. Available: http://netflow.caligare.com/netflow_v8.htm

[10] G. Gu, R. Perdisci, J. Zhang, W. Lee *et al.*, "Botminer: Clustering analysis of network traffic for protocol-and structure-independent botnet detection." in *USENIX Security Symposium*, vol. 5, no. 2, 2008, pp. 139–154.

[11] Y. Desmedt, "Man-in-the-middle attack," in *Encyclopedia of Cryptography and Security*. Springer, 2011, pp. 759–759.

[12] A. Gazet, "Comparative analysis of various ransomware virii," *Journal in computer virology*, vol. 6, no. 1, pp. 77–90, 2010.

[13] D. M. Smith, "The cost of lost data," *Journal of Contemporary Business Practice*, vol. 6, no. 3, pp. 1–9, 2003.

[14] D. A. Leslie, *Legal Principles for Combatting Cyberlaundering*. Springer, 2014, page 7.

[15] C. Paar and J. Pelzl, *Understanding cryptography: a textbook for students and practitioners*. Springer Science & Business Media, 2009.

[16] S. H. Chad Dougherty, Jeffrey Havrilla and M. Lindner. (2003, August) W32/Blaster worm. CERT. Accessed 9th March 2015. [Online]. Available: http://www.cert.org/historical/advisories/CA-2003-20.cfm

[17] Microsoft. (2009, April) Worm:Win32/Conficker.E. Accessed 31st October 2015. [Online]. Available: http://tinyurl.com/hdlorbc

[18] C. Shannon and D. Moore, "The spread of the witty worm," *Security & Privacy, IEEE*, vol. 2, no. 4, pp. 46–50, 2004.

[19] Microsoft. (2003, July) Microsoft Security Bulletin MS03-026 - Critical. Microsoft. Accessed 27th January 2016. [Online]. Available: https://technet.microsoft.com/library/security/ms03-026

[20] ——. (2003, September) Microsoft Security Bulletin MS03-039 - Critical. Microsoft. Accessed 27th January 2016. [Online]. Available: https://technet.microsoft.com/en-us/library/security/ms03-039.aspx

[21] ——. (2008, October) Microsoft Security Bulletin MS08-067 - Critical. Accessed 31st October 2015. [Online]. Available: https://technet.microsoft.com/en-us/library/security/ms08-067.aspx

[22] R. Dhamija, J. D. Tygar, and M. Hearst, "Why phishing works," in *Proceedings of the SIGCHI conference on Human Factors in computing systems*. ACM, 2006, pp. 581–590.

[23] S. Staniford, V. Paxson, N. Weaver *et al.*, "How to own the internet in your spare time." in *USENIX Security Symposium*, 2002, pp. 149–167.

[24] L. Bilge and T. Dumitras, "Before we knew it: an empirical study of zero-day attacks in the real world," in *Proceedings of the 2012 ACM conference on Computer and communications security*. ACM, 2012, pp. 833–844.

[25] D. Maynor, *Metasploit toolkit for penetration testing, exploit development, and vulnerability research*. Elsevier, 2011.

[26] J. Muniz, *Web Penetration Testing with Kali Linux*. Packt Publishing Ltd, 2013.

[27] R. Perdisci, A. Lanzi, and W. Lee, "Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables," in *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*. IEEE, 2008, pp. 301–310.

[28] N. Provos *et al.*, "A virtual honeypot framework." in *USENIX Security Symposium*, vol. 173, 2004, pp. 1–14.

[29] P. Hintjens, *ZeroMQ: Messaging for Many Applications*. " O'Reilly Media, Inc.", 2013.

[30] SQLite Consortium. (2016) Sqlite: Small. fast. reliable. choose any three. Accessed 30th April 2016. [Online]. Available: https://www.sqlite.org/

[31] MemSQL Inc. (2016) Make every moment work for you. Accessed 30th April 2016. [Online]. Available: http://www.memsql.com/

[32] D. Miller. (2016) Softflowd. Mindrot. Accessed 30th April 2016. [Online]. Available: http://www.mindrot.org/projects/softflowd/

[33] L. Rudman and B. Irwin, "Characterization and analysis of ntp amplification based ddos attacks," in *Information Security for South Africa (ISSA), 2015*. IEEE, 2015, pp. 1–5.