Implementation and Evaluation of A Low-Cost Intrusion Detection System For Community Wireless Mesh Networks

A Thesis Submitted to the
College of Graduate Studies and Research
in Partial Fulfillment of the Requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Wojciech Quibell

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science
176 Thorvaldson Building
110 Science Place
University of Saskatchewan
Saskatoon, Saskatchewan
Canada
S7N 5C9

Abstract

Rural Community Wireless Mesh Networks (WMN) can be great assets to rural communities, helping them connect to the rest of their region and beyond. However, they can be a liability in terms of security. Due to the ad-hoc nature of a WMN, and the wide variety of applications and systems that can be found in such a heterogeneous environment there are multiple points of intrusion for an attacker. An unsecured WMN can lead to privacy and legal problems for the users of the network. Due to the resource constrained environment, traditional Intrusion Detection Systems (IDS) have not been as successful in defending these wireless network environments, as they were in wired network deployments. This thesis proposes that an IDS made up of low cost, low power devices can be an acceptable base for a Wireless Mesh Network Intrusion Detection System. Because of the device's low power, cost and ease of use, such a device could be easily deployed and maintained in a rural setting such as a Community WMN. The proposed system was compared to a standard IDS solution that would not cover the entire network, but had much more computing power but also a higher capital cost as well as maintenance costs. By comparing the low cost low power IDS to a standard deployment of an open source IDS, based on network coverage and deployment costs, a determination can be made that a low power solution can be feasible in a rural deployment of a WMN.

Contents

List of Tables

List of Figures

List of Abbreviations

CIDS	Cross-layer Intrusion Detection System
CPU	Central Processing Unit
RPi	Raspberry Pi
DOL	Distributed Object Locator
DOS	Denial Of Service
DDOS	Distributed Denial Of Service
DPI	Deep Packet Inspection
DSR	Dynamic Source Routing
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HDD	Hard Disk Drive
ICMP	Internet Control Message Protocol
IDS	Intrusion Detection System
IP	Internet Protocol
ISP	Internet Service Provider
MAC	Media Access Control
MANET	Mobile Ad-hoc Network
WANET	Wireless Ad-hoc Network
PCAP	Packet Capture
RAM	Random Access Memory
SAHN-IDS	Suburban Ad-Hoc Networks Intrusion Detection System
SSH	Secure Shell
SMTP	Simple Mail Transfer Protocol
SD	Secure Digital
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
VM	Virtual Machine
WMR	Wireless Mesh Router
WMN	Wireless Mesh Network

Chapter 1

Introduction

Wireless Mesh Networks (WMN) have gained popularity as a cost effective deployment for rural and developing areas [5, 11]. WMNs are utilized for the facilitation of communication and commerce, between internal and external users [5]. The openness of the WMN can give a malicious attacker many different methods with which to compromise the WMN. A compromise in the security and or privacy of the WMN can lead to numerous online and real world problems for those affected. Personal data and financial data can be exposed and identities can be compromised and taken over. A security system is needed for such a network. In corporate and scholar networks, the systems are usually homogeneous and as such can be monitored and fixed by applying universal remedies; however, not all WMNs are homogeneous. A great number of them are made of varying Operating Systems (OS) and hardware. WMNs allow many different devices to connect to the network. The variety of the Operating Systems and hardware is wide and may require a variety of different drivers to simply communicate with the network, and as a result, require security to be added on after deployment due to the lack of homogeneity. Therefore, establishing adequate and cost-effective security mechanisms, such as intrusion detection, provides a significant challenge to the deployment and effectiveness of a WMN.

1.1 Characteristics of Wireless Mesh Networks

As shown in Figure 1.1, Wireless Mesh Networks (WMN) are communications networks in which the routing infrastructure is comprised of nodes (usually WIFI-connected) that are organized and deployed in a mesh topology. Wireless Mesh Networks consist of End User Systems, mesh routers and gateways. The End-User systems can consist of desktops, laptops, cell phones and other wireless devices, while the mesh routers can be special purpose devices that forward traffic to and from the gateways which in turn may, but need not, connect to the Internet [5, 60]. The WMN uses radio frequency communications, usually WIFI but there have been WiMax deployments in the past [59]. A Wireless Mesh Network can help communities with only a limited number of consistent and reliable broadband connections to the Internet build up a network that would facilitate Internet connectivity for the entire community. The open nature of the WMN, however, could lead to possible problems in regards to information security for the community's user base.

If a WMN is open, anybody can access it, even people driving within radio range (up to 300 ft) of the

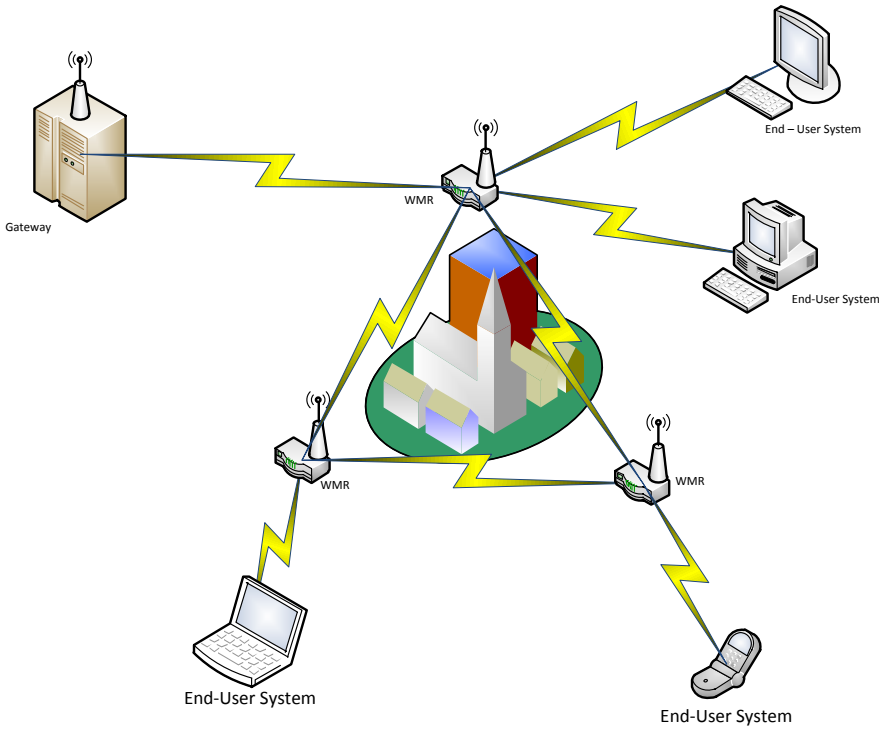


Figure 1.1: An Example Wireless Mesh Network

community [38]. This can lead to multiple points of attack and not only through the regular gateway to the Internet. A successful intrusion could lead to the compromise of user data and identity theft. Identity and data theft can lead to financial and legal implications for the victims [8]. Even if the users of a WMN do not think that there is anything of personal or financial value held in their computers, the computing resources themselves can be the target.

Access to the network and maintaining connectivity depends on the routing nodes of the network working together to forward and route packets from one end of the network to the other. Mesh networks are designed to be reliable as they incorporate commonly-used fail-over techniques such as Ad hoc On-demand Distance Vector(AODV) [28] and have the possibility of built-in fault tolerance. Wireless mesh networks can make use of various wireless technology including 802.11, 802.16, WiMax cellular technologies or an amalgam of the multiple protocols.

A WMN is made up of two separate parts, the Wireless Mesh Routers (WMR) and the End-User Systems [33, 5, 43, 60, 41]. The End-User machines are simply regular consumer machines that connect to the network

and use it as they would any other networked connection [5]. End User Systems connect to the routing boxes to access the Internet; wireless connectivity can be used for surfing the Internet, gaming and business applications, etc. The WMRs make up the backbone of the network as they route the packets from each node to the gateway for the Internet Service Provider(ISP). WMRs can be old converted consumer machines or they can be specially built low-power machines that are specially configured for the purpose of routing packets to other nodes. Converted WMRs have the drawback of using more power than specially-built WMRs [5]. Some specially-built WMRs can often be powered by batteries. They can also be located quite a distance from a persistent power source, facilitating easy routing and network connection [5]. Routing boxes forward the packets for all the applications that are being run on the End User machines. Figure 1.1 shows an example layout of a WMN.

There are plenty of uses for Wireless Mesh Networks. They are found in regular usage in different Municipal Wireless networks that provide free connectivity to their residents and businesses. Examples of Municipal Wireless Networks in the last ten years include 1) Calgary, Alberta, 2) Brookline, Massachusetts, 3) Rochelle, Illinois, 4) Thorold, Ontario, 5) Milton Keynes, UK, 6) Oulu, Finland, and 7) Galatsi, Greece [11, 56]. The municipal networks are usually a joint venture between a local government and a private company, and allow for free connectivity to anyone within a certain radius around the city centres.

Another application for WMNs is the use of Community Wireless Mesh Networks. These networks differ in that they are setup and maintained by the residents of the community themselves, with little or no help from the municipality. Places such as Wray [33, 43], Berlin [62], and Boston's MIT RoofNET [4] have established community-driven WMNs with a bit of help from local universities for the maintenance and technical expertise.

WMNs provide the same network connectivity and applications as traditional wired networks do. Unlike urban areas, the cost for an Internet connection in rural or remote areas may be considerably higher as there is a smaller subscriber pool from which ISPs can recover deployment costs or make profits. In other cases, the rural community may be so remote that traditional wired service cannot be provided to everyone in the community. The distance issue may require a more costly solution to the connectivity problem and it is possible that only the more wealthy in the community could afford it. By deploying a community WMN, the cost of the network is spread out amongst the community. With more people connected, the per-user costs decrease as each new person is added to the WMN, amortizing the initial infrastructure costs, enabling the community to use the WMN to perform regular Internet activities that were previously cost prohibitive.

For the rural community of Wray [33, 43], deployment was a costly, time consuming, and risky endeavour. The main costs of deployment were with the mesh boxes that acted as routers for the peer packets to be forwarded; however there is also a cost in securing the network against intrusion, data loss, and disruption due to connectivity and availability issues.

1.2 Security Requirements in Community Wireless Mesh Networks

By using the computing resources of the WMN, attackers can launch distributed attacks against other networks. By installing a Trojan program, attackers can issue commands remotely to the local user's computers causing them to initiate connections to foreign networks [61]. These remote instructions can steal cycles from users, generate more network traffic for the WMRs to route [64], and perhaps lead to legal troubles for users. Generating more traffic for the WMRs to route can lead to increased power consumption on a WMR, and subsequently leading to a shorter battery life. Decreased battery life means that the network nodes may become disconnected more frequently leading to service disruption. Also, more traffic generation has the ability to tie up the network resources, thereby lowering the throughput of the WMN and leading to congestion in network packets [64].

The security function of the WMN can potentially be more costly than initializing the WMN. Securing WMNs is complicated, due to issues with power consumption, multi hop routing, authentication of router nodes, and the encryption needed to ensure trust between nodes [60]. Also there is the issue that security is an ongoing and evolving process with new and different exploits being discovered as technology matures.

According to Dourish and Redmiles, security is an End-User problem [14]. Most breaches of security are either caused by or affect the end user. Due to the variety of different end-user programs that are available for users of the WMN and the multitude of bugs and exploits associated with those programs, the easiest part of a system to compromise is the end-user system. In a network such as a WMN, the large number of devices that can connect to the network and the various intentions of the connecting end users, combine to form an emergent heterogeneous network. Also, since different nodes can be run by individuals in the community or run on battery power to facilitate connectivity, a WMN can be classified as an Ad-Hoc Network, as nodes can join and leave with no set schedule or warning.

A community WMN will not always have technical expertise available locally to defend against the numerous possible security threats [65, 66, 41]. Due to potential lack of advanced technical knowledge on the part of the community [66], individual routing nodes that provide the vital connectivity of the network may be initialized, in many cases, with the default setup. Use of the default setup can lead to the nodes of the network not having adequate security protection software, resulting in many exploits that could be open to potential intruders.

There is an identified requirement for the network itself to provide additional security to novice users, and to discourage its use in an external attack [60, 43, 33]. Any compromise of a node system in the WMN could lead to disconnectivity, loss of personal information (identity fraud, credit card numbers, mail fraud, etc.), and resulting legal issues.

1.3 WMN Security Issues in Detail

Security is not a single component of a system or network. As most modern systems now cooperate with many different components, each component can be the target of a security breach. An attacker can compromise any component of a system with a multitude of possible attacks. If he or she is successful in compromising one of the components that interact or are a part of the system, then the system itself can be compromised.

Aspects of Security

There are five separate components that comprise information systems security: Confidentiality, Authentication, Integrity, Availability and Non-Repudiation [74]. For an attacker to be successful, he or she only needs to compromise one of the five components. The defending network, on the other hand, needs to secure all five; thus providing the required security is a challenging task.

Confidentiality is the ability to keep information readable to only the select users who are authorized, and required to view it. In a WMN the transmission is open; therefore, the data is readable by anyone. To allow for complete confidentiality, cryptography must be used. Cryptography is a resource-intensive strategy for providing confidentiality, and the machines that perform it must have the available resources to accommodate it; otherwise, the packet forwarding and routing task efficiency of the network suffers.

Authentication is the ability to correctly identify a valid user of the data or service. Authentication is usually done through a central authority such as an authentication service like Kerberos [50]. In a WMN, the network is distributed, with no central authority. The WMN relies on its components to communicate and authenticate each other and due to possible environmental and power fluctuations it is not always possible to connect to a third party server.

Integrity is the ability for the data to remain unchanged from its original state during transmission or viewing. In a WMN, it is easy for an attacker to alter the network data, as the nodes act as relays to the rest of the network. It is therefore possible to compromise integrity by changing packets' payloads at a malicious router.

Availability requires the resource or data to be available when the service or data is requested. Attackers use Denial of Service (DOS) attacks to remove availability. DOS attacks usually tie up resources of a system so that valid users have delayed or no access to the network resources. By generating a large number of packets from a single node or bringing a high coupling [1] node down, an attacker can cause a DOS attack.

Non-repudiation is the inability for a node to deny or disown data sent in to the network. Non-repudiation is a legal term that has been adapted to fit in to the computing security context. Again, since a WMN is a multihop network, it is possible for attackers to spoof IP or MAC addresses. There is an implicit trust between nodes, and this kind of spoofing can lead to the nodes disowning routing data that other nodes

[1] High Coupling Node: a node in a network that is connected to multiple other nodes with more connections than the average node in the network.

think they have sent. Since this is true data that is being disowned, the spoofed node loses reputation, since it is providing unreliable information, and other nodes would not trust the routing done by spoofed nodes. If no nodes trust the others, then the routing capability of the network would fail.

All these components are as vital to securing a rural WMN as they are to securing a city-wide or company-wide network. A community WMN has difficulties in providing the same security as larger city and corporate networks. The heterogeneity of the community usage gives a number of attack vectors for malicious operators to attempt.

Due to the fact that the systems are generally considered open, there is also the possibility that an attack can come from within the network [24]. An attack from within the network creates the added problem that nodes cannot implicitly trust the data that is originating from other local user nodes and possibly the routers themselves, and as a result need to mitigate the security flaws.

1.3.1 Common Attacks Against WMNs

The attack vectors that are used by malicious intruders are much the same as those that are launched against a regular network.

The Denial of Service attack involves an attacker trying to compromise a network not by gaining entry into the network but by preventing the regular users of that network from normal access [70]. The attacks are difficult to identify as quite often the attack sends valid data to the network, but the DOS sends it at a rate that a network with resource constraints can have difficulty serving it and the valid requests simultaneously.

A **scanning** attack is essentially a precursor to an attacker gaining entry to a system. The attacker uses a tool to scan the ports on a network to determine what applications are running on the network and if they have any communication ports open, they could receive malicious commands from the attacker.

A **Man-in-the-middle Attack** is comprised of an attacker intercepting all or part of the communication, changing the content, and sending back as legitimate replies. Both speaking parties are not aware of the attacker's presence and believe the messages they receive are from each other.

Spoofing is used to masquerade as a person, program or an address as other than oneself by altering the address data for the purpose of unauthorized access.

The various attack vectors a malicious computer user can undertake can range from simple to quite sophisticated; the sheer number of potential attack vectors requires that the user of a network has help in identifying and mitigating the risks that he or she is exposed to every day while on the network. The users can have help via software or expertise means.

1.4 Potential Solution: Intrusion Detection Systems

An Intrusion Detection System (IDS) is a tool that analyzes the traffic on the network. The primary purpose of an IDS is to determine if a node in the network has been compromised or is attempting to send traffic that is not typically sent within the network [39]. Such a system can also be used to alert the community network in case of odd behaviour. Traditional IDSs in wired systems sit on a central point of a network and detect packets being sent through. Since the wired systems are usually on the same subnet, an administrator can set up an IDS on the gateway so that it may filter and detect all the traffic that is entering from outside the network.

Traditional IDSs are usually found in larger, mostly corporate networks [14]. Small public networks do not usually have the resources to hire a professional to deploy and monitor an IDS. These networks have a much different user and resources base than a corporate network [7]. Barman *et al.* investigated the mono-culture in IT environments and found that users can be grouped by specific contextual network usage. Barman *et al.* also stated that the idea of one-size-fits-all for IDS configuration in this context can lead to numerous problems. Therefore, since a centrally designed and focused IDS does not work for a network that is mostly made up of homogeneous clients and routers, then the challenge would be much greater to configure a single IDS for a diverse network such as a WMN. In a WMN, an Intrusion Detection System needs an adaptable configuration to deal with a variety of different traffic patterns [7]. The users of a WMN have the potential to use any program that can be networked, and in turn can offer numerous protocols that must be secured. This wide variety of usage can lead to false positives[2] or in certain cases, false negatives.[3] In the case of false positives, this could lead to a preemptive disconnection or cause panic among the user base.

In a rural setting, there is frequently no full time system administrator that can configure and monitor an IDS [35, 66, 65]. Instead, a rural community relies on phone support, intermittent visits from administrators from outside the community, or self taught members of the community to maintain the system [66]. An intermittent visit is in most cases too long to wait for an intrusion to be detected or mitigated. There needs to be a way that the system can detect these intrusions and potentially heal itself, without much help from outside, as there is the distinct possibility of connection disruption or power failure [66].

Certain attacks can permit attackers full access to the network and they only require a few hours or minutes to conceal the evidence of a system compromise, meaning that even a trained professional may not be able to detect and mitigate the intrusion. Due to the concealment aspect of a possible attack, it is even more imperative that the IDS should be monitoring at all times in order to prevent an attack.

The scope and range of attacks that a single node IDS can cover can be a challenge. While it may cover its own subnet in the WMN, the rest of the network may be open to undetected attack. Unless packets

[2]False Positive: situation in which the software would detect an intrusion where there was none

[3]False Negative: situation in which the software does not detect an Intrusion event, where one is present

come into radio range for the single IDS, even outbound packets from the WMN can infect other parts of the network. In a WMN, devices can connect to any part of the network, allowing for a potential to compromise the network from within.

Rather than have a single IDS system listening to the network and performing deep packet inspection on packets, a system that includes multiple low cost devices performing the IDS responsibilities is proposed. A single IDS may only be able to monitor the outgoing traffic; the rest of the network could be compromised without its knowledge. Due to the hybrid nature of the WMN, and relatively low computational power that Wireless Mesh Routers(WMR) require, the low-cost devices will be able to off-load the work allowing for the WMRs to solely focus on packet forwarding.

1.5 Thesis Statement

This thesis proposes that an IDS made up of low cost, low power devices can be an acceptable base for a Wireless Mesh Network Intrusion Detection System. Experiments evaluating the system with typical WMN traffic demonstrate that the Raspberry Pi can handle the processing requirements for deployment in a small to midsize system in a flexible and low-cost manner.

1.6 Contribution

The proposed system offers potential solutions and tradeoffs as to the layout of a WMN for greatest possible security and mitigation and minimal cost. The proposed low cost device IDS can be implemented to better protect a WMN using low cost hardware available to most community members. With lower cost devices and a readily implementable platform, the on-site technical expertise is not as crucial to the security of a community WMN. This thesis will analyze the feasibility of using the low cost Raspberry Pi device as an IDS by comparing it to a standard desktop operated IDS.

1.7 Overview

The rest of this thesis is organized as follows. Chapter 2 will detail the related work in regards to the other IDS that have been attempted for WANETs, MANETs, and WMNs. Chapter 3 introduces the proposed low cost low power system. Chapter 4 will detail the experiments and the types of metrics that will be collected and analyzed. Chapter 5 will examine the results and analyze the data collected. Chapter 6 will give a conclusion and provide ideas and new directions for future work.

Chapter 2

Background and Related Work

2.1 Security Issues in Wireless Networks

There has been much research in regards to ad-hoc and sensor network security. Wireless Mesh Networks have not been covered as extensively due to their relatively recent deployment. WMNs and Ad-Hoc/Sensor networks have some similarity and because of this similarity, the security research overlaps between the two. There have been different approaches to IDS systems proposed in the literature. The remainder of this chapter contains an evaluation of these approaches and their applicability to Community Wireless Mesh Networks.

2.1.1 Terminology & Definitions

Computer security involves usage of a lot of attacks software, mitigation terminology that can have obtuse naming. As a result their purpose may not have an obvious meaning or understanding. The terminology is defined in the next subsections.

Attack Definitions

There are many different attacks that are possible in a WMN. Attacks in a WMN can target a number of different technologies in a network. Among the attacks that can target a WMN are: Teardrop attack, DNS Tunneling, Computer Worms including the slammer worm, and Trojan Horses including the `Trojan.tbot`, and `Poison Ivy CnC`.

Teardrop Attack. The teardrop attack is a denial of service (DOS) attack that targets TCP/IP fragmentation reassembly codes [45]. Fragmented packets overlap one another on the host receipt and the attack malforms them. When the host attempts to reconstruct them during the process it fails. Gigantic payloads would be sent to the targeted systems, causing the system to crash.

DNS Tunneling. DNS Tunneling was discovered in 1998 and allows an attacker to bypass outbound security controls and devices by tunneling IP over DNS [25]. The would-be attacker has control of an external DNS server. The attacker then sends a malicious or infectious email to the victim. The victim

sends a request for a certain host name within a domain. The compromised server responds with a modified RData field that will act as a command to the victim. The modified DNS queries can then be used to transfer files or execute remote commands.

Computer Worm. A computer worm is a malware computer program that replicates itself in order to spread to other computers [19]. Worms have often been responsible for massive disruptions in the Internet and many data compromises. Worms also attempt to replicate and send themselves out into the Internet in the hope that the exploit they use is found on other hosts. Once a new host is infected, the worm attempts the reproduction process again. Slammer is a computer worm that caused a denial of service on Internet hosts and impeded Internet traffic on January 25, 2003 [51, 19]. Slammer exploited a buffer overflow vulnerability in Microsoft's SQL server and Desktop Engine database suite. The slammer worm's behaviour consisted of generating random IPs and sending a version of itself to the generated IPs. The key components of the Slammer worm were that it was very small so that it fit inside a single packet, and it used the UDP protocol to send out potentially hundreds of packets per second. The rapid packet generation and traffic caused certain routers on the Internet to crash and become unusable. The result of this was that other routers that had not failed updated their own routing tables to remove the crashed routers. The failed routers then were restarted triggering a massive routing table update after their announcement to other routers. This caused regular traffic to slow down on the Internet as most routers were busy configuring and updating their routing tables.

Trojan Horse. A Trojan is a non-self-replicating type of malware program that typically causes loss or theft of data, and possible system damage. Trojans often use social engineering, presenting themselves as a non-threatening and useful piece of software so that an end user will install them. There are many various trojans that exploit networks, two of them are listed below.

The `Trojan.tbot` was a recent trojan that would create a directory on a Windows-based OS with a random name and then rename itself with a random string. A Trojan is considered to be a non-self-replicating type of malware program containing malicious code that carries out actions typically causing loss or theft of data.[1] A Trojan often acts as a backdoor, contacting a controller which can then have unauthorized access to the affected computer. The `Trojan.tbot` injected itself into an svchost.exe process and terminated the original process. It would then listen to a newly created IRC(Internet Relay Chat) session for command and control commands [17].

`Poison Ivy CNC` uses a stack-based buffer overflow in `CoolType.dll` in Adobe Reader on Windows and Mac OSX. It allows remote attackers to execute arbitrary code or cause a denial of service (application crash) via a PDF document with a long field in a Smart Independent Glyphlets (SING) table in a TTF font. `Poison Ivy CNC` was exploited in the wild in September 2010 [17].

[1] http://usa.kaspersky.com/internet-security-center/threats/trojans

Software Terminology

Blacklist. A blacklist is a list that holds the various IP addresses of malicious sources. It is generally used to disallow traffic from the sources into the network. An IDS would use a blacklist to save on resources by not needing to scan the packet from the source, simply disallowing it into the network.

Whitelist. A Whitelist is a list that holds the various IP addresses of trusted sources, it is the opposite of a blacklist. It is generally used to allow traffic from the sources into the network, as the traffic is considered trusted. It is used in IDS to make scanning more efficient, and prevent resources from being utilized to process already trusted sources.

Deep Packet Inspection. Deep Packet Inspection (DPI) is the filtering and categorizing of a network packet data part in addition to the header of a packet. It attempts to verify if the packet contains viruses, spam, intrusions, or other predefined harmful criteria. While a lot of IDS check the headers of a packet, DPI allows an IDS to cover more possibilities of intrusion.

Software Tools Techniques

Rsyslog. `Rsyslog` is an open-source software utility used in Linux systems for forwarding log messages in a network. It is an extension of the syslog protocol, by allowing it to filter the logs based on certain text properties, and allows for TCP to be used as a transport mechanism.

Nmap. `Nmap` is a program used to discover hosts and services on a computer network [29]. `Nmap` sends specially crafted packets to the target host and then analyzes the responses. It is often used by attackers to map out a network or host and determine what open programs are running on them. There are a few preset scans available to `Nmap`: Quick Scan, Regular Scan, Intense Scan, and Slow Comprehensive scan.

Quick Scan. The Quick Scan will execute a TCP connect() port scan on the target host and report on open ports. It is difficult to detect as it only initiates the connection and does not follow up on it.

Regular Scan. The Regular Scan means it will issue a TCP SYN scan for the most common 1000 TCP ports, it also uses an ICMP Echo request (ping) for host detection. The ping is what will give the scan away as pinging more than one port in succession is a common rule in `snort`.

Intense Scan. The Intense Scan scans the most common TCP ports. It will make an effort in determining the OS type and what services and what versions of the services are running. This also includes the ping scan as well as the OS profiling, so it is easy to detect.

Slow Intensive Scan. The Slow Intensive Scan has a large number of options enabled. The main focus of the scan is for host detection, by not terminating the scan if the initial ping request fails. It uses three different protocols in order to detect the hosts; TCP, UDP and SCTP. If a host is detected it will do its best in determining what OS, services, and application versions the host are running based on the most common TCP and UDP services. The scan camouflages itself as a DNS request.

2.1.2 Availability

A DOS attack greatly affects the availability of a network or system. If a valid method for detecting frequent excessive valid requests can be determined, then a potential to mitigate Denial of Service (DOS) attacks can be implemented with an IDS. Other methods have suggested to increase the bandwidth of a system in case of a DOS attack [70]; the feasibility of the aforementioned methods can be disputed, however, because of the lack of available bandwidth in a lot of rural areas. Another method of DOS attack mitigation is computational puzzles introduced by Aura *et al.*, where the performance trade-off for the clients is having the client compute puzzles to delay replay of attacks, and to tie up DOS nets [6]. Due to the battery and processing power requirements of mesh boxes, computational puzzles could consumer power unnecessarily.

2.1.3 Non-Repudiation

Research into non-repudiation has mostly been done at the protocol level with researchers introducing new protocols to achieve non-repudiation. Meng *et al.* introduced a fair non-repudiation protocol that used hash values of the ciphers of plain text messages into the non-repudiation tokens, thereby reducing the amount of data sent over the network, while still achieving non repudiation with a third-party authority [47]. Yang et al. introduced a Behaviour Authentication Code (BAC) in wireless mesh network user postings [77]. They manipulated the data frame on the MAC layer in a WMN by including keys that included group keys, and user addresses. Teh group key would be used to decrypt the BAC code that is embedded in the packets, and would be compared to the BAC on file at the end point. They simulated the approach and found that they were able to secure and disown any data that was trying to be sent without the BAC embedded. However they did not go beyond the simulation phase of the project, and the scope of the project was only limited to user postings on a BBS type server. Adikari introduced the Efficient Non-Repudiation Protocol (ENRP) [2] for non-repudiation in techno-informational environments. They used cryptography with multiple trusted third party authentication services that allowed for the binding of the certificates to be high[2] by making forging very difficult and making sure the message cannot be altered. The methodologies introduced in the design of both protocols can improve the non-repudiation. The protocols also require that each node in the network would be aware and utilize the protocols. This would be a very difficult task to deploy successfully because of the ad-hoc nature of a WMN.

[2]Physically bonded to an entity with a unique identification or evidence is stored and managed by a trusted third party [2]

2.1.4 Confidentiality

Confidentiality in a WMN is a difficult concept to realize as most of the solutions for confidentiality would require a full network solution for the WMN and could not be added after system deployment and in an ad-hoc manner. Wang et al. implemented a new Simple Authentication and Key Distribution Protocol (SAKDP) that would be used to distribute authentication keys in a WMN [71]. With SAKDP utilized symmetric keys and a trusted network center to handle the generation of the authentication keys. The clients would communicate with the network center using one way functions that would then be exchanged for the symmetric keys to the other end point, thereby allowing the two sides to communicate without someone eaves dropping on the exchange. Wu *et al.* provided a solution that would encrypt each message in the WMN [73]. They also introduced a penalty-based routing algorithm that would prevent the discovery of any emerging traffic patterns that could potentially be exploited by an intruder. The algorithm would send traffic to its destination through nodes in a predefined list of nodes that would not always take the most efficient path; instead, it would choose a path that would be able to disrupt the traffic pattern to outside observers. The results of this would be that the traffic entropy would increase to the outside observers, making it more difficult to determine the source and type of traffic.

2.1.5 Authentication

Studies in regards to the user authentication of the WMNs have been mostly done at the routing protocol level. Different routing protocols have been introduced to authenticate the individual routing nodes to the network. Numerous authentication schemes were introduced to allow new users to be authenticated when they joined the WMN. Fu *et al.* introduced an authentication scheme that allowed peers to authenticate nodes as they joined, provided that they had already registered previously with an offline certificate authority [21]. The offline certificate authority would then issue the public and private keys and select existing nodes from the network that can handle extra processing overhead that would then act as a virtual authority online that would manage the predefined keys. This solution is quite secure, but requires extra overhead and limits the ad-hoc nature of the WMN, and the redundancy of the the Certificate Authority can be compromised. Li and Nguyen introduced a fast authentication scheme in which the client and the mesh access point authenticate each other [40]. Mobile devices can then travel and connect from node to node by use of tickets that are generated by the nodes between themselves. The mobile hardware would use the tickets to allow authentication on most of the nodes on the network. While this methodology would limit the need for the mobile devices to perform intensive computing for authentication, and remove the need for a central certificate authority, it can still be bypassed by an attacker commandeering a node via different means.

2.1.6 Integrity

There is research that deals with providing algorithmic solutions to the security problems [76], and the processing capacities of WMNs [75] required for intensive tasks such as packet analysis. Hubaux *et al.* investigated securing the communication in ad-hoc networks; having each user generate and distribute their own certificates, allowing for better authentication between the nodes of the network. Since the nodes are using their own certificates and can verify the validity of the sender with other nodes, the packets would have to be de-crypted in order to change the data. With the certificates acting as a trust source, the system achieved data integrity, preventing attackers from spoofing the packets thereby compromising the integrity of the packets. When nodes want to communicate, they exchange their certificate directories to determine if they are trustworthy. Srinivasan *et al.* looked into determining security by cooperation with a Generous Tit-for-Tat algorithm, determining which nodes are acting in a normal manner [63]. Here again, all solutions provided required an increase in processing overhead and more technical knowledge to deploy the systems.

2.2 Intrusion Detection Systems

Researchers have shown that cooperation can be useful in securing WANETs and there have been a few IDS systems developed that use a distributed and cooperative scheme to provide intrusion detection functionality in the WMN.

2.2.1 Agent-based IDS

Zhang and Lee provided a framework for the deployment of an IDS in a wireless mesh network [79]. They proposed an IDS running on each of the nodes in the network, with communication and cooperation between the nodes. Each node acts as an IDS agent on behalf of the entire system. Each system or node has a local trace that it collects and monitors. If strong evidence of an intrusion (i.e. a detected malware signature or anomalous network behaviour) is found, the IDS signals an alarm to the rest of the network. If an anomaly is detected, then it must coordinate with other nodes to confirm an intrusion event has taken place. If the majority of the nodes around an inquiring node detect something suspicious, then the inquiring node can then issue an alarm and act as though an intrusion has taken place. If an intrusion is determined to have taken place, then the nodes can reorganize to exclude the offending node from the network. This can be accomplished by issuing a re-authentication for all nodes, or to have all nodes remove the offending node from the network. This approach was shown to be more resource intensive than the agents could handle, and found a high false alarm rate [36]. Each IDS would have to be rule-limited for the IDS in order to work on each node and still not tie up all the resources on the node. Another weakness was that most of the IDS proposed were light-weight; deep packet inspection is not implemented in most cases as it as been shown to use up 31% of the processing time for `SNORT` [18].

Kachirski *et al.* also devised a distributed IDS with agents monitoring a subset of the network [36]. Their stated goal was to provide an IDS that could function in a low power and low bandwidth mobile environment. Agents can provide coverage for basic IDS functions as well as using cooperation to exclude nodes from the network [36, 49]. To save on power and computing resources, they implemented the agents in a modular IDS deployment, meaning that not every node received an IDS agent that monitored the network; instead each agent monitored their host on both the system and application level. Only some nodes were given network monitoring duties. To determine which hosts received the network monitoring duties, they divided the nodes into clusters and then each cluster would vote on the node to be the monitor, based on the number of connections that each node had to the rest of the cluster. In their scheme, any node that determined it was under attack could raise an alarm. The network monitors would raise alarms if they or any member of their cluster were sending or receiving suspicious traffic. No mitigation strategy was given. The authors also stated that there would still be a high false positive rate due to the nature of the network in which it was deployed. To extend the mobile agents and give them the resources to perform deep packet inspection would require them to be installed on lower powered Mesh Boxes in a WMN that would then not be capable of processing the required tasks [23].

One problem with mobile agents is that they require a high security clearance and would have to be self-updating. Installing updates frequently on different areas of the network could be misappropriated by an attacker and could lead to a compromise in part or all of the WMN [12]. If an attacker could spoof the agent signature, they would be granted elevated privilege in the network.

2.2.2 Cluster-based IDS

Yi *et al.* proposed an IDS based on a finite state machine that uses cluster-based intrusion detection for a Wireless Ad-Hoc Network to detect DSR protocol exploits and intrusions [78]. They argued that such a system would be more efficient than one in which all of the nodes acted as an IDS. The proposed algorithm consisted of one node being promoted to the position of Monitor for a cluster of nodes. As in the previous implementation, an election algorithm is used to select the monitoring node. Once the Monitor has been elected it monitors all the incoming routing messages for the zone for which it is responsible. If an incoming REQUEST message is deemed to have been tampered with, then the monitor rejects the route request message and does not forward it to the rest of the zone for which it is responsible. The drawbacks to this system are that the election algorithm can seemingly be exploited by an attacker by acting as a monitor node and simply forging their ID to become a monitor node. If an attacker achieved monitor status, he or she could then alter the routes at their leisure. Another drawback is that the cluster-based approach proposed only functioned with the DSR algorithm and required that there be a significant number of nodes from which to choose a monitor [78].

2.2.3 CIDS

Bose *et al.* created a cross-layer IDS (CIDS) that is capable of detecting DOS attacks across multiple stack layers [10]. As DOS attacks can be successful at different layers of the IP stack, they decided to permit an IDS to analyze all the available layers in order to detect malicious nodes. They implemented a multi-level approach to detection. In the first level, if a node is detected to be misbehaving, it gets flagged and put on a NO TRUST REGION and is then passed on to level 2 for further analysis. Level 2 then gathers information from different layers in the network. Level 2 analyzes the amount of energy resources at a node, the overall congestion and then any malicious intent of the packets to determine if an alarm should be raised. By analyzing different responses at different layers, they were able to successfully detect a DOS attack 100% of the time. They eliminated the false positive rate by combining analysis from different tests such as energy levels, packet drops, sequence packets lost by intrusions, etc. when looking at a maximum of 55 nodes in a simulated environment. This research was only specialized in the detection of DOS attacks, however, and did not attempt to detect other types of malicious traffic. Also, it did not show a distributed IDS but rather a single stationary one that receives information relayed by the rest of the network, leading to a potential single point of failure in the security of the network.

2.2.4 Routing Protection

Marti *et al.* used `Watchdog`, and `pathrater`, two tools that help to monitor the network and mitigate against routing algorithm exploits [46]. They developed a way to detect DOS attacks in a Wireless Ad-Hoc Network that routes with DSR; they would only work on wireless networks as `Watchdog` would need to listen to the traffic of nearby nodes to work correctly. If a node was found to misbehave (by not transmitting or delaying the packets), it would be removed from the routing table [46]. `Watchdog` can detect packets being dropped at the forwarding level, as well as the link layer. The `pathrater` program is run on each node in the network and it combines the `Watchdog` information to build a more reliable network path for all nodes that are running it. The `pathrater` algorithm assigns the shortest path to a node and its intermediary steps between nodes with a neutral value and always rates the local node higher than the neutral value. After a certain time interval, the algorithm then increments the rating system on all paths until a predetermined maximum value that is still a bit less than its own path rating. When a node connection breaks, then each time slice is decremented until the floor of 0 is reached. If, however, a node is flagged by `Watchdog`, then its rating is set at a high negative value (-100); this allows for false positive nodes to eventually work their way back into the network.

However, the `Watchdog` system could be fooled by a node dropping packets at a lower rate than the `Watchdog` threshold [49]. Also, `Watchdog` can't function very well in an environment where a lot of natural collisions are occurring as it can be fooled by low-powered or failing nodes. As well, `Watchdog` can be fooled by a node falsely reporting that another node is misbehaving. `Watchdog` also did not take into account Deep

Packet Inspection (DPI) because of its resource requirements. This can lead to complex malware slipping through the system and infecting other nodes.

2.2.5 Anomaly-Based Ad Hoc IDS

Islam *et al.* [34] developed an intrusion detection system for Suburban Ad-Hoc Networks (SAHN-IDS). SAHN-IDS focused on two specific areas of attack. The first attack vector was the misuse of the transmission channel, and the second was anomalies in packet forwarding [34]. Among the characteristics of misusing the transmission channel were the following:

- Non-compliance with the MAC protocol,
- Jamming the transmission channel with packets of unknown format or random bit-patterns and static noise, and
- Not complying with the bandwidth reservation scheme.

The anomalies in packet forwarding were dropped packets, and delayed packet transmissions. SAHN-IDS uses traffic volume and a preallocated bandwidth rating to determine if there is a misuse of the transmission channel. A limitation of this system is that it can only detect an attack if only one neighbour node is misbehaving.

SAHN-IDS uses timestamps and expected transmission and processing delay to determine whether or not there is a delay attack present in the network. If an attack is discovered, then the routing protocol is altered to avoid the misbehaving node. They deployed SAHN-IDS on each node in the network, and ran a simulation of the system in GloMoSim with 70 nodes. They found the detection was fairly fast and effective and was able to maintain a high throughput of the network traffic. The authors did note however that there was a 6% increase in traffic overhead from the IDS itself passing messages.

2.2.6 Grid-Based IDS

Donadio *et al.* proposed a distributed IDS using grid computing middleware [13]. Each node would contain an agent that would provide a continuous monitoring service, a decision making service, an action making service, and a communication service. The monitoring service would be a passive monitoring system that would be able to share traffic statistics and other information over the grid. The main processing is handled by the open source framework CoMo [27]. The nodes would each have their own decision making system and would provide coverage for other node clusters. The processing would be done by each node and the IDS would be distributed throughout the network using the GLOBUS [54] open source communication service. The Grid-based approach required all of the nodes to be homogeneous.

This type of system would be difficult to deploy in a WMN consisting of heterogeneous nodes, namely the lower-powered routing nodes and higher-powered end nodes. There is little mention of nodes listening

on other nodes traffic as a means of monitoring, and while the authors state they built a prototype, no experiments were performed on it. One particular problem is the possibility of letting too much information become available to the other nodes.

2.2.7 SNORT

SNORT is an open source network intrusion detection system created by Martin Roesch. It is used worldwide as a primary IDS to secure networks of leading companies in the Aerospace Industry, wireless industry and Military [57]. SNORT performs protocol analysis, content searching, and content matching. SNORT is a packet sniffer that monitors network traffic in real time, analyzing each packet to detect a dangerous payload or suspicious anomalies. It can be run in three different modes, packet logger, sniffer, or intrusion detection mode. SNORT is based on libpcap, a tool that is widely used in TCP/IP traffic sniffers and analyzers. Through protocol analysis and content searching and matching, SNORT is capable of detecting many different attack methods, including denial of service, buffer overflow, CGI attacks, stealth port scans, and SMB probes. When suspicious behavior is detected, SNORT sends a real-time alert to syslog, and a separate alerts file. There are also many rules and administration managers available for SNORT that allow the rules to be updated as new exploits are confirmed. As a result of SNORT's widespread use, it makes an ideal platform from which to base a low-cost, low-power IDS. The open source nature of SNORT makes the software capital cost nil and the large adoption rate allows for a well documented and large knowledge base. This allows non-experts to be able to configure or troubleshoot the SNORT IDS in environments like WMNs.

2.2.8 OpenLIDS

Hugelshofer *et al.* [26] investigated lightweight intrusion detection research on WMN systems. They showed that conventional IDS software such as BRO [55] and SNORT [57] can severely hamper the performance of an underpowered mesh routing box. The CPU utilization could reach 100% with processor load reaching over 1.0 meaning that processing time would continue to be delayed and the processing queue would keep growing, resulting in the routers dropping packets. The resulting dropped packets would negatively impact the application performance of the end nodes. A detection mechanism was developed that could be deployed on the lower-powered node machines called OpenLIDS. They used simple detection mechanisms to detect anomalies that could be an indication of a mass mailing worm, in order to save on space and resources. Deep packet processing was removed due to the inability of the mesh boxes to perform the necessary pre-processing on the packets and still route the packets efficiently. They used DNS MX queries and SMTP traffic to detect the mass mailing worms and spam. They collected the number of packets being sent, ratio of incoming and outgoing traffic, average packet size, and average flow size. They used the collected data to infer whether or not an attack was taking place. OpenLIDS was tested with traces taken from Wray, as well as traces of a variant of the conficker worm test detection. The authors indicated that systems running BRO and SNORT were dropping packets in the preprocessing phase when analyzing the Wray traces. OpenLIDS was able

to allow the boxes to function on the low powered mesh boxes by not performing deep-packet inspection. The drawback to not attempting the deep packet inspection is that more sophisticated attacks, worms and exploits can still be used against the `OpenLIDS` system.

2.3 Low Power Devices Security Deployments

There have been many deployments using low-power devices in various different deployments. Usually the low power devices are used for single-task functionalities like monitoring or logging; the Raspberry Pi, however, has been able to load an operating system, and be used as a small multitasking platform.

The Raspberry Pi (RPi) has been used for a variety of projects, including as a database server, in numerous home automation projects, as a learning device for low income areas of the world, and even a as security monitoring system. The processing power and flexibility of the device has been applied to higher processing tasks such as decryption and video processing [32]. Low power devices are desirable due to low power consumption and low cost, facilitating ease of replacement and low overhead when deploying new devices into a WMN.

Other devices are also low powered such as the Arduino [58] and the Texas Instruments MSP430 Launch Pad [30], however they require more technical knowledge to program and deploy and do not have as much software available for the platform. The Arduino platform is a series of boards and microcontrollers created at the Interaction Design Institute Ivera, that can be assembled together with different peripherals to perform various tasks. The Arduinos are usually shipped with just the board and usually no operating system type software. The developer is usually required to program the board for the desired task. Much like the Arduino, the MSP430 launchpad was designed by Texas Instruments to provide microcontrollers with significant lower power consumption than previous generations of previous hardware. As with the Arduino, the MSP430 requires the developer to compile directly on to the board to perform the desired task. This requires more technical knowledge than a typical user in a WMN has. The RPi was chosen in this work for its durability and low cost as well as complete ease of use and fully functional, plug and play capabilities.

2.3.1 WiFiPi

Low power devices were also used to track and simulate the movements of people in crowds at a mass event [9]. The researchers deployed Raspberry Pi devices in a popular summer concert series and around their own university. Using the Raspberry Pi to process the incoming connections from people's wireless devices they were able to process and model the crowd dynamics. The researchers did not let the people attending know that the devices were present in the field. The devices were able to process a coverage area of 400m by 500m. They were able to identify 137,899 unique devices over a span of three days. They mentioned that the processing power of the Raspberry Pi needed to be able to filter the incoming packets to identify and log the unique devices, and the devices were able to keep up to the task.

2.3.2 Drone Penetration Testing

A Raspberry Pi has been used for security penetration testing. Abramov *et al.* attached a Raspberry Pi to a flying drone, and used it to launch attacks against different WIFI networks [1]. Using the Raspberry Pi it was able to use PwnPi suite to launch attacks, as well as using the Raspberry Pi to perform logging and flight recording. They were able to crack both open networks and encrypted networks using a processor intensive program `AirCrack`. The research showed that the Raspberry Pi, is able to perform in the processor intensive application of security.

2.3.3 CreepyDOL

The `CreepyDOL` [53] detection network was built by Brendan O'Connor and presented at both BlackHat2013 and DefCon 21. `CreepyDOL` was a project in which Multiple RPi devices were used to connect to public WIFI hotspots and essentially track individuals. The project worked by distributing the RPi devices over a certain geographical area and logging the hardware and software information of laptops, smart phones and other WIFI enabled gadgets that connected to the access points. They logged all of the wireless connections and data that was readily available in the vicinity of the hotspot and sent it to logging server. After a few days the information gathered was able to provide the basic information to identify the people attached to the sniffed devices. After a month of data collection, meaningful insight was provided into the private lives of the owners of the devices. For instance the personal details of a person such as their marital status, what bank they used, their browsing habits, and other more personal details depending on what sites they accessed could be extrapolated from the collected data. The RPi devices themselves did not masquerade as access points, but merely sniffed and logged the information that was being sent in plain text between the devices and hotspots. They were deployed in arbitrary locations determined by the desired geographical coverage area where they could obtain power from the nearest wall socket. The distributed deployment matches that of a WMN; thus, the feasibility of the `CreepyDOL` deployment showed the potential of the Raspberry Pi to host an application in wireless security domain.

2.4 Summary

A great deal of the related research has only attempted to develop an IDS, but there has been little or no research attempted with regards to an automated DPI (Deep Packet Inspection) mitigation system to be deployed in a remote rural area. Due to the multitude of possible attacks, a system that can attempt DPI is needed to detect increasingly sophisticated malicious behaviour in order to better protect the WMN. By providing another layer of security defense for the WMN, the success of the WMN to function as a communication outlet for rural communities greatly increases. The Raspberry Pi has been shown to be able to handle various and differing processing requirements, to have a large software library, and that its

platform can be possibly applied to an IDS.

Chapter 3

The Raspberry Pi Intrusion Detection System

3.1 Overview

An IDS would be very beneficial to a WMN with respect to protecting the end-user's data and privacy. The hardware limitations of the WMN that provide benefits such as low overhead, inexpensive and easily replaceable hardware and battery power as the main potential source of power can prove deploying such an IDS difficult. A low cost, low power device can be augmented to the nodes of the WMN, however, to provide an IDS function. As shown in Chapter 2, the Raspberry Pi has been developed so it can meet such a need for niche computational usage. Using the Raspberry Pi platform [20], a low-cost, low-power and low-overhead device, an IDS based on `SNORT` can be deployed in a distributed fashion over a large physical distance. The distribution of the IDS nodes would function to augment the security of the WMN at the node level.

The RPi platform provides a flexible environment for an IDS. No commercial software is required to be installed on the systems, and Linux-based operating systems are readily available. The device is easily configurable and have built-in Ethernet ports (Model B). This port is used in the system for physical connection, but Wireless connectivity is also available [20]. The devices could be deployed throughout the network to listen in on the network traffic providing a real-time intrusion detection service.

Once deployed, the RPi acts as a packet sniffer[1] and scans all the packets being sent from the node, even packets being forwarded from other nodes. By monitoring each node, the RPi machine provides a barrier from internal exploits. By default, the end stations in the network would not trust each other. Internally generated packets would no longer be blindly accepted by the WMRs; instead, all packets would be scanned. This policy prevents static trust exploits [79] by having all of the data analyzed. A compromised node would not be able to impersonate a cooperating node without more effort to hide its true identity or nature.

Due to the RPi's portability and ability to run on battery power, non-traditional node locations are also possibilities for deployment. The devices would be distributed to the Mesh nodes' locations to provide coverage in routing node areas, rather than deploying one IDS to monitor the entire network. Thus, detection would be spread out into the network, allowing for more robustness in security rather than relying on a single node that may fail, leaving the entire network unsecured. Should a misbehaving computer be detected,

[1] A program and/or device that monitors data traveling over a network.

the devices would then communicate with each other to alert the entire network about the inappropriate computer behaviour, therefore preventing the spread of the intrusion to other sections of the network that may or may not be aware of the security breach.

3.2 Design

The RPi's operating system is based off a version of Debian (`Raspbian`). The RPi can be altered to work with open source intrusion detection programs [20, 37]. The devices analyze the packets that are being transported on the network to and from any device connected to the WMR. Since the RPi is monitoring the incoming traffic of a node, it receives everything that the node receives before forwarding the traffic. This will allow the analysis to be done while not burdening the network nodes with scanning tasks and allowing the WMR to only route packets. An overview of this can be seen in Figure 3.1. The RPi machine runs the `SNORT` software, which performs the scans of the packets in real time. Due to `SNORT`'s reliability, large community [57], and relative ease of deployment it was chosen as an ideal platform on which to base an IDS.

One of the reasons that `SNORT` was chosen as the test IDS is that it is capable of Deep Packet Inspection (DPI). DPI is a form of packet filtering that not only examines the header but also the data portion of the packet to determine if there are any attempts to intrude or compromise the system in the packet payload. IDSs that provide DPI are able to reveal more malicious attacks than IDS that do not offer it. The extra detection comes at a price of more processing time and resources being used to analyze the packets against all rules.

Should the detection program running on the RPi device detect anomalous or malicious behaviour, the device would then flag it as malicious and then notify the rest of the network or routing nodes with an alert. Once the anomalous traffic is detected, an alert is sent out to the network. The RPi nodes then begin to look out for packets coming from the detected intruder. By initiating a large scale alert, a mass attack can be detected and mitigated before it infects a majority of the WMN. The RPis will be able to add a new rule to the `SNORT` ruleset to automatically drop packets coming from the detected malicious source. This would attempt to prevent an attacker from getting further access via an exploited node. Currently mediation of the alarm would have to be attempted manually, meaning that the threat an black list would have to be added to the routing nodes, as there currently is no mechanism for the RPi devices to message the WMRs automatically.

3.3 Raspberry Pi Hardware

The Raspberry Pi was created with the goal of providing a fully functional computing tool at a low cost to help facilitate educating the general public about computing [20]. The Model B runs off of a 5v micro USB and can be used in conjunction with a 1.2A power supply to run the majority of linux based utilities and

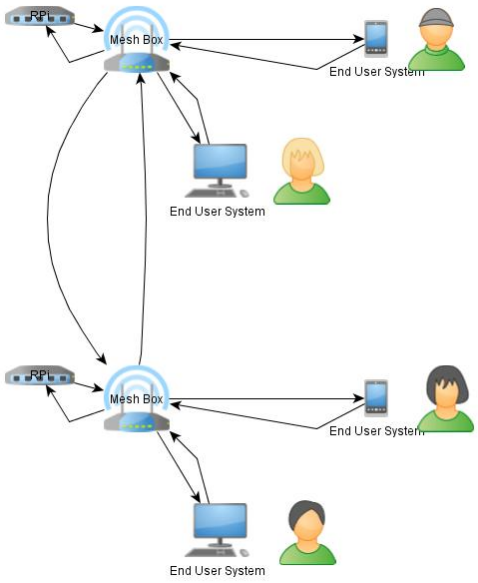


Figure 3.1: Communication Between Two Nodes with RPi Attached.

non-computationally intensive applications [20]. The maximum power consumption for the Raspberry Pi is 1 ampere. As shown in Figure 3.2, the Raspberry Pi Model B has 2 USB ports usually used for a keyboard and mouse, regular audio output, S-video out, HDMI output, and SD card slot (this is used as the main hard drive). The HDMI connection comes standard and can connect to any monitor or television that includes either HDMI or DVI ports. The USB ports will only output up to 1 ampere, so certain USB devices may need to be powered, or if more than two devices are to be used, a powered USB hub is recommended.

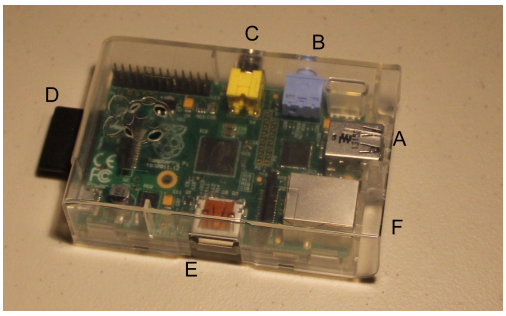
There are many flavours of Operating Systems that are tailor-made for the Raspberry Pi. The `Raspbian` OS is made specifically for the Raspberry Pi and is based off of the Debian Linux distribution. Arch Linux can also be installed directly on the Raspberry Pi. Most of the Operating Systems come with `python` installed as the main tool for programming. With the `python` PIP libraries, rapid application development on the Raspberry Pi is achievable.

3.4 Implementation

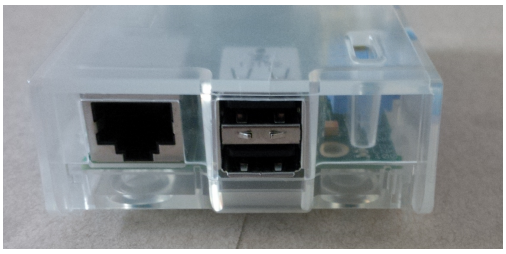
3.4.1 SNORT on the Raspberry Pi

With some system alterations to the setup of the Rasbian OS, `SNORT` was installed on the RPi [37]. The `SNORT` program was running along side of an RPi communication program that works by sharing detection results amongst the RPis.

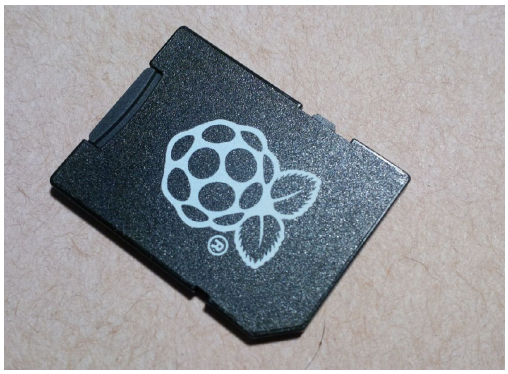
The setup for the RPi machine involved first building a working image of the finished RPi IDS on one



(a) A) Dual USB ports B) Audio Out C) Video Out D) SD Card (8GB) E) HDMI out F) Ethernet



(b) Side View.



(c) SD Card the "Hard Drive" for the RPi

Figure 3.2: RPi Hardware

instance and then using that instance as a clone template for any future machines. The raspbian OS was installed on the SD card that functioned as the RPi's hard drive. The configuration of the OS was standard, except for the following three changes. First, SSH was upgraded on the OS. The SSH session would be the only way of altering or communicating with the RPi machine once deployed. Second, there was a need to deallocate memory from the RPi's GPU back to the system memory as the configuration would provide more memory for the system to run `SNORT`. The RPi would not require the GPU memory as it would not be displaying graphics to the user. Finally, the iptables service was configured to only allow SSH messages through, any other packets incoming to the RPi machine would be dropped as a preventative measure against compromise of the IDS itself.

Once the machine was configured, `SNORT` was installed. `SNORT` was installed on the RPi raspbian OS using the built in `apt-get` utility. `SNORT` was configured to run in intrusion detection mode, with the logs of the packets being sent to a separate log file and in command line only mode to reduce CPU processing overhead for a GUI not required for this deployment. In intrusion detection mode, `SNORT` was given permission to drop traffic in real time. The `SNORT` program was also run in fast mode, meaning that the PCAP files are not be decoded from their binary form to a `wireshark` and other text editor compatible plain text format. Fast mode allows for faster execution of the detection algorithm. The `SNORT` rules were updated via a utility for `SNORT` called `PulledPork`. `PulledPork` is a rule manager for `SNORT` and `Suricata`. It automates the process of downloading and installing/updating VRT SNORT rules, Shared Object rules or Emerging Threats rules. The init.d service was configured to allow the `SNORT` process to be run on start-up of the machine.

When the machine was booting with the `SNORT` service active on start up, the image was cloned. Using a clone of the image, made it easier and faster to configure other RPi devices without having to go through the same setup steps for each machine.

The clone was made using a combination of `parted` and `rsync`. `Parted` is a disk partitioning and partition resizing program. It facilitates creation, destruction, resizing, movement and copying of ext2, linux-swap and FAT partitions. `Rsync` is a fast file copying tool. It can copy locally to and from another host over any remote shell. As shown in Figure 3.3, partitioning the SD card with two partitions (one for the boot sector, and the other for the regular OS image) facilitated the loading of the cloned files to the SD Card.

3.4.2 Deployment

Once the device setup was completed, the RPi IDS was attached to the individual WMR. They were connected, due to their physical proximity to the WMR, via wired connection to the WMR as an ethernet connection is available on the Model B. In a deployment scenario the model A versions of the Raspberry Pi could be used which only support wireless connectivity. All traffic to the WMR is routed through the Raspberry Pi and the RPi acts like a filter for all the traffic to the WMR. Due to the low power requirements,

```
root# parted /dev/sda
GNU Parted 2.3
Using /dev/sda
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) mklabel msdos
Warning: The existing disk label on /dev/sda will be
                destroyed and all data on this disk will be lost.
Do you want to continue?
Yes/No? y
(parted) mkpart primary fat16 1MiB 64MB
(parted) mkpart primary ext4 64MB -1s
(parted) print
Model: Single Flash Reader (scsi)
Disk /dev/sda: 8069MB
Sector size (logical/physical): 512B/512B
Partition Table: msdos

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  64.0MB  62.9MB  primary  fat16        lba
 2      64.0MB  8069MB  8005MB  primary


root# mkfs.vfat /dev/sda1
mkfs.vfat 3.0.13
root# mkfs.ext4 -j /dev/sda2
mke2fs 1.42.5
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
488640 inodes, 1954304 blocks
97715 blocks (5.00\%) reserved for the super user
First data block=0
Maximum filesystem blocks=2004877312
60 block groups
32768 blocks per group, 32768 fragments per group
8144 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632

Allocating group tables: done
Writing inode tables: done
Creating journal (32768 blocks): done
Writing superblocks and filesystem accounting information: done
```

Figure 3.3: Partitioning the SD Card

```
root# rsync -av --one-file-system / /boot /tmp/newpi/
var/mail/
var/opt/
var/spool/
var/spool/rsyslog/
var/tmp/
var/www/

sent 2204324795 bytes  received 1303348 bytes  2608667.23 bytes/sec
total size is 2199215758

root# df -h
Filesystem               Size  Used Avail Use% Mounted on
rootfs                    15G  2.3G   12G  17% /
/dev/root                 15G  2.3G   12G  17% /
devtmpfs                 215M     0  215M   0% /dev
tmpfs                     44M  248K   44M   1% /run
tmpfs                    5.0M     0  5.0M   0% /run/lock
tmpfs                     88M     0   88M   0% /run/shm
/dev/mmcblk0p1            60M  9.4M   51M  16% /boot
tmpfs                     10M     0   10M   0% /tmp
/dev/sda2                7.3G  2.3G  4.6G  33% /tmp/newpi
/dev/sda1                 60M  9.4M   51M  16% /tmp/newpi/boot
```

Figure 3.4: Copying the Image to a New Card.

the RPis could be run off of the same type of battery setup as the WMR, or via traditional grid connections. The setup for these experiments were run off of regular grid power.

The housing requirements for the RPi were minimal due to the device's size and weight. The RPi could be fit into most existing housings. With multiple housing availability, the RPis can be put in more secure or sheltered locations. For ones deployed in more open spaces, the small size will allow it to be placed in optimal locations. Optimal locations indicate a location that facilitates the maximization of the signal power and transmission radius, such as the tops of trees and/or poles in the middle of fields within line of sight of another node. The RPis were deployed and continued to run until manually shut down or the power supply was removed from the board.

3.5 Discussion

3.5.1 Benefits

The RPi devices themselves are resilient and cheap. They are small and require very little in actual power usage, they can be accessed remotely and can run a lot of different beneficial anti-intrusion programs. They are easy to conceal and can be placed on or near the WMN nodes. Each RPi primarily uses a 4 GB flash card that holds its OS and file system. If the device breaks down or is compromised, then the flash card can be replaced and the RPi would function as new. Development for different accessory programs on the Pi

is relatively easy as most of the development is done with Python and is extensively documented. Python comes standard with the raspbian OS.

3.5.2 Concerns and Limitations

The most notable concern is the resource constraints found on the RPi devices. While they are incredibly powerful, the devices still have some limitations in processing power and storage space. The devices are also limited by the power supply available in the area. The Raspberry Pi can run off of batteries but not for an indefinite amount of time.

As the traffic level grows, the detection can take longer as the amount of information to go through can be limited by the processor in the RPi. If the rate of traffic rises to a level where the Pi cannot keep up with the rate, then a delay between initiation of attack and detection will increase. One of the test plans included the ability of the RPi to keep up with increasing traffic.

3.6 Summary

In order to help secure a WMN, a low cost, low power device called the Raspberry Pi was chosen as the platform to host an IDS. The Raspberry Pi was chosen due to the ease of use and deployment, readily available documentation, and knowledge bases. The Raspberry Pi made the deployment of an IDS very easy and quick with abilities to expand and evolve the ruleset for the IDS.

Chapter 4

Experimental Design

A number of experiments were established to verify the effectiveness of the RPi as an IDS. The experiments consisted of simulations of a WMN environment that is being attacked by a sampling of attacks consisting of malicious exploitative code in the payload of the in-transit packet, discovery probes, and other various commonly found historical attacks that have been used as benchmark exploits in previous research in the last 10 years. The experiments were run on two platforms running `SNORT` IDS: a desktop machine serving as a baseline system and the experimental RPi system.

4.1 Goals

The premise of the experiments was to emulate a WMN environment that could be attacked by various attack vectors. The responses of the system determined how well the RPi handled detection of the malicious traffic at different traffic thresholds in addition to how well it alerted the rest of the system of the intrusion. One of the goals was to show that a low cost solution would be economical and effective in a community WMN. The economic characteristics of the system were tested by comparing the RPi system to a similar, more common desktop solution.

The `SNORT` software is not a standard install on the Raspberry Pi. To install `SNORT`, a few tweaks needed to be made to allow it to run as a daemon [37]. The Raspberry Pi `SNORT` was configured to run in fast mode and to have a zero size swap file so that the virtual memory would not be allocated to save on memory usage. It was imperative to ensure that the installation did not alter the operation of `SNORT`. On the baseline, the `SNORT` system was set up with a simple install via `apt-get` and one configuration change to allow it to run as a daemon on start up. The `SNORT` program on the baseline was unaltered, there were no tweaks unlike the install on the RPi. Due to this difference, the RPi would be tested against the baseline system with a subset of common packets to show detection commonality between the two systems.

To show that an RPi system would be a competent solution vs. that of a regularly deployed IDS, the RPi machine would have to perform within detection and traffic tolerances that would be common in a WMN. The detection performance would be based off of the detection time of a comparable system that would be performing intrusion detection. If the detection time is still comparable with those from the baseline system, then the RPi can be considered to detect the attack before the attack can spread. The traffic tolerances refer

to the amount of and type of traffic that would be found in a community WMN. Most of the time, this would be traffic created by commonly used consumer programs such as Skype, YouTube, and `BitTorrent` [69]. The detection times would be based off of the detection times for the baseline system, and while they are expected to be slower, they should still be able to detect malicious packets in a time that is not detectable to the end-user and that would prevent an entire attack to propagate throughout the network. To be considered successful, the RPi would have to be able to scan the traffic at the WMR, without interrupting a normal end user experience. This would mean that, in the case of the streaming videos and the Skype calls, the experience would be uninterrupted communication and video playback outside of normal network variability. For the `BitTorrent` client, the end user should not experience longer than usual download times.

The RPi would also have to perform intrusion detection services on a WMN, monitoring the traffic and guarding different attacks at a substantially lower cost than that of a traditional IDS. The goal of the experiments was to show that the designed characteristics for the detection and mitigation system layout would be proven to be robust and able to be easily deployed in a WMN with minimal cost and user expertise.

Evaluation of the RPi device was performed in a closed network environment using packet captures of malicious attacks and penetration scans. The packet types, attack frequency, and duration of attack were varied to allow for testing of the RPi at various levels of usage and throughput. Among the measurements taken were the following: the number of false positives and negatives, the number of attacks and penetrations detected, the time it takes to detect from the issuing of the attack, the throughput of the system, and the power consumption of the device. The system would raise an alert throughout the entire network and the alert frequency was analyzed for the time of detection[1] and whether the alarms were reasonable.[2] Finally, the cost to security ratio was evaluated by comparing the IDS and other similar commercial products. The cost to security ratio was the percentage of the network that would have an IDS monitoring all of its traffic.

4.2 Environment

All simulations were done on a private internal network with RPi monitoring packets being sent throughout the network. The RPi machines each had 8GB of HD space from a class 6 SD card. Each RPi consisted of a `Raspbian` Linux operating system with a modified version of `SNORT` running on each machine. The desktop machine had an Intel Core I3 processor running with 8GB of RAM and 1TB of HD space running Linux Mint, a lightweight distribution of the Linux operating system also running `SNORT`. The data sets contained a variety of packets that were found in a variety of networks. The packets chosen for the tests were divided into two groups: attack packets, and regular non-malicious packets. The data sets consisted of packet captures from the following sources:

[1]The detection time of a malicious packet measures from the time it was launched into the network to the time an alert was raised.

[2]False positive and false negative rates.

- `Wireshark` Sample Captures (*http://wiki.wireshark.org/SampleCaptures*) (25 captures),
- DARPA Intrusion Detection Data Sets (*http://www.ll.mit.edu/mission/communications/cyber/CSTcorpora/ideval/data*) (10 captures),
- OpenPacket.org Capture Repository (*https://www.openpacket.org/capture/list*) (17 Captures).

4.2.1 Attacks

The `Wireshark` sample sets are captures of traffic packets. They are captures of normal protocols that could be found on community WMNs. Traffic would include SAMBA,[3] HTTP, UDP and Appletalk. The `Wireshark` data sets also include some captures of a teardrop attack, DNS exploits, worms including the slammer worm, and the DNS remote shell anomaly. They were discussed in Section 2.1.1.

The DARPA Intrusion Detection Data sets are captured Internet traffic packets that are used by an IDS to tune and configure itself. The purpose of the packets is to be used as a training data set to allow the IDS to detect common attack vectors and abnormal data in packets to facilitate anomaly detection. The DARPA data sets are from 1998 and 1999, so the packet data is outdated, but it is still used today as a baseline qualification for IDS testing.

Malicious packet captures were run against the system, specifically the packet captures of a Teardrop attack, Trojan.tbot, and Poison Ivy CNC. These were chosen because they were examples of some common types of attacks that would compromise an end user system's security. All of the attacks are defined in Section 2.1.1.

4.2.2 Regular Traffic

Regular application traffic was also submitted to the network to show that the IDS would not confuse it for malicious packets; therefore not letting legitimate packets through into the network. Openpacket.org had packet captures of various different packet types much like the previous two examples; unfortunately the site has since been discontinued, therefore many of the packet types are outdated as of 2011. Openpacket.org had a wide variety of captures as it was a community based repository of packets due to all sorts of users submitting PCAPs to the repository. This meant that more unusual traffic was monitored and captured and then submitted to the repository.

First, regular packets of varying protocols at different layers were run through the network to verify that false positives would not be produced. The protocols that were tested were NFS, HTTP, BitTorrent, Kaspersky Update Protocol, Kerberos, mDNS, SSL, UDP, WIFI, and H.223. Because of their popularity in heterogeneous wireless networks, the rest of this section describes the details of these protocols and their

[3]Samba is a popular freeware program that allows SMB/CIFS to access and use files, printers, and other shared resources on a local network.

use cases. NFS is an application layer protocol used in distributed file systems, primarily those that are Linux-based. HTTP is the common protocol for retrieving web pages and is used on a WMN whenever a user uses a web browser. BitTorrent is a protocol that allows users to share files with peers. Kaspersky Update Protocol is used by an anti-virus brand Kaspersky to update their software. Kerberos is an authentication protocol that users would use to verify themselves with each other in a secure manner. The mDNS Apple Rendezvous is a data link layer protocol used to identify and share Apple product resources and files over the network. SSL is used to encrypt communication between a client and server at the application layer, it is used in places like online banking and shopping. SSL can also be used in conjunction with other protocols to provide security for sessions that involve those protocols. UDP is a transport layer protocol commonly found on a WMN when users use the protocol for online gaming [15], VoIP [52] and multi-cast video streaming [22]. WIFI (802.11) is the standard MAC-layer protocol that would be used by a lot of machines on the WMN as it would be the protocol that allows the different devices on the local network to communicate. H.223 is used by video phones to send data over publicly switched telephone networks and can be captured in the data-link layer.

4.2.3 Typical Usage Testing

Typical community usage was simulated with the common applications that are most frequently found on typical end user networks such as a community WMN [69]. These applications include `BitTorrent`, Skype, and HTTP traffic including sites like Facebook, and YouTube. By modeling the typical usage of the network, the goal was to show that the RPi would be able to handle the common traffic requirements of a modern WMN.

The typical usage was simulated with three typical user applications to see if the RPi machine could handle the traffic and still function effectively as an IDS. The simulation of normal day-to-day activity included downloading a `BitTorrent` with approximately 200 peers, making a Skype call with both video and audio, and watching a full YouTube video. The `BitTorrent` ran for 29 minutes of continuous download and upload. The total size of the transferred file was 316.9 MB, and the transfer time was approximately 27 minutes. The max download rate was 218KB/sec with a max upload rate of 40KB/sec. The Skype call lasted for almost 1 hour, and was averaging about 40 packets per second. Both video and audio were on between two laptops that had motion happening in front of their web cameras. The YouTube video was 7 minutes 18 seconds long running at 720p resolution. It was accessed via a `Firefox` web browser and decoded with a flash plug-in. The video data itself was compressed. The total size of the video that was downloaded to the client was approximately 204 MB.

The performance data was collected by using a bash script that ran a combination of `date`, `vnstat` and `top`. Also wireshark was monitoring on the endpoint to determine the total traffic received at the node, without taxing the RPi. The CPU, memory and traffic volume were sampled every ten seconds during operation so as not to inflate the working data set. The script can be found in Figure 4.1.

```
#!/bin/bash

while true;do
        process=‘top -b -d 1 -p $(pgrep -d’,’ snort) -n 1‘;
        date=‘date +%D%T%N‘;
        sleep 1;
        echo $process $date $traffic >> snort_log.log
done
```

Figure 4.1: Bash Script for Logging performance

4.3 Test Setup Details

The test setup system used the Raspberry Pi running a version of SNORT specifically tuned for the low power devices. The RPi machine was set up as a network bridge between the machine and the network that it would be monitoring. Using the RPi as a wired network bridge was required due to a limitation on the model B RPi that prevents the network card from entering promiscuous mode. When a data packet is transmitted in non-promiscuous mode, all the LAN devices listen to the data to determine if the network address included in the data packet is theirs. When the promiscuous mode is enabled, the RPi's WIFI adapter would connect to the network and sniff all the data that was flowing through the wireless network to the mesh box. The benefit of promiscuous mode would be that the adapter would intercept and read all packets regardless of which device they were addressed to. This is how the IDS is able to filter and check for malicious packets that are not intended for the IDS. If there was no promiscuous mode then the IDS detection capabilities would be severely limited.

4.3.1 Single Packet Worms

The following attacks are susceptible to being discovered by DPI: Buffer Overflows, Single packet Worms [72], and DOS attacks. Examples of these attacks were included in the testing sample. The attack packet traces were available in many forms and can be downloaded in PCAP form from the Wireshark sample captures page as mentioned above in Section 4.2.

A scanning attack was attempted by using NMAP to port scan the protected network. Depending on the responses received from hosts, it is possible to determine the running applications on the host and the operating system that the host is running. When the IDS detected the attack, it was evidence indicating a successful deployment and provided protection against the most common first step for an attacker.

TCPReplay [68] was installed on a separate machine internal to the network. It was already assumed that the attacker was on the internal network. The attacker node then sent malicious packets from a trace file into the network at a specified rate. Specific nodes could be targeted within the network or packets

could be sent to the network blindly in hopes that there is a host on the network susceptible to the attack. `TCPreplay` suite also comes with `tcpreplay-edit` that allows the user to replay the attacks with varying frequency and arbitrary speed [68].

4.3.2 Detection & Remediation

The average time for a successful specific attack was recorded and the process was repeated in an attempt to infect further nodes. Based on the detection time determined, the listening nodes would then begin shutting down the attacking node by updating a blacklist and passing the blacklist on to the rest of the network. A *blacklist* is a list that holds the various IP addresses of malicious sources. It is generally used to disallow traffic from the sources into the network. `SNORT` has the ability to drop packets entering the network if it is being used as a network bridge. Conversely, if `SNORT` is being used in a WIFI environment, it would create an alert that it would send to the other nodes. The other nodes would then disallow the traffic from the blacklisted IP into their local network.

4.4 Factors

The purpose of varying the input factors was to demonstrate that the RPi system is applicable in different settings and can be considered a robust intrusion detection system. In order to be considered a robust IDS, the detection must be able to scale to the size of the network. Each run had variables that were tweaked to simulate a more broad operating environment as can be found in an actual WMN. The parameters that were varied for the testing phase of the experiment were the following:

- Packet Type,
- Frequency of Attack, and
- Volume of Attack.

4.4.1 Packet Type

The packet type ranged from the standard packets that facilitate normal program network traffic to specifically-sculpted malicious packets. By testing a variety of packets and protocols against the IDS, it can be shown that the system would be compatible with the non-homogeneous nature of a community WMN. Both systems were subjected to the same packet types to measure whether the `SNORT` implementation was stable and the same on both systems. The packets that were analyzed were the following:

- NFS Protocol Family,
- Hypertext Transport Protocol (HTTP),

- Telnet,
- Bittorent Protocol,
- Kaspersky Update Protocol,
- SSL,
- UDP, and
- WIFI/ Wireless LAN Captures/ 802.11.

Also the following malicious packets were analyzed:

- Teardrop Attack,
- Trojan.Tbot, and
- PoisonIvy CnC.

4.4.2 Frequency of Attack

The frequency of the attacks was altered to ensure that the processing limitations of the RPi could be measured and applied to a real world scenario. By increasing the frequency of attacks, the IDS should be able to determine that the attacks are more frequent and not regard the packets as an increase in normal traffic. Frequency of attacks and overall network load was measured with Microsoft Network Monitor monitoring tool that is free to use for local systems [48]. To alter the frequency of attacks, the attack packets were deployed at an increasing rate from 1 per minute to 1 per second.

4.4.3 Volume of Attack

The volume of the attacks was altered to see how the RPi machine could handle the processing requirements needed to detect malicious packets that can be hidden with carefully sculpted packets. This is where the DPI inspection occurs and, as a result, can handle more intricate attacks. With an increasing volume of traffic the device will have to be able to still scan all the packet payloads to determine if the packet is truly malicious or not. This was to simulate a DOS attack, where the packet flood can severely hamper an IDS's ability to perform correctly.

To measure the total volume that a RPi would be able to handle, 10 runs of 10 seconds of sustained packet packet traffic were run against the RPi and the desktop system. The each run was at different packet/sec intervals from 50 up to 900. `Top` was run on each system at a 1 second interval. `Wireshark` was used to collect the traffic throughput at the destination node. The averages of the runs were then collected and measured for the packet/sec and the MBit/s measurements.

4.5 Metrics

The metrics that were tracked belonged to two categories:

- Performance of the system as an IDS
- Cost comparison.

4.5.1 Performance of the RPi as an IDS

The following metrics were tracked to indicate the performance and reliability of the RPi:

- The number of attacks and penetrations detected. The number of attacks and penetrations detected would provide a simple metric for determining whether or not the system would be considered more secure by utilizing the IDS. The detection was measured over a period of simulated use.
- The time it takes the system to detect an intrusion attempt. The detection time was used to show whether or not the IDS would be able to provide warnings in real time before the attack had a chance to complete.
- The throughput of the system. The throughput would measure whether or not the RPi would be able to handle the workload of the normal traffic and day-to-day use of the WMN. This was measured in units of KB/sec or packets/sec as well as total throughput over a period of time.
- The number of false positives and negatives. The number of false positives and negatives would measure the accuracy of the IDS and general usability level of the RPi. The number of non-malicious packets that the system allowed in without raising an alert was measured to ensure against false positives. Malicious packets that did not trigger an alert would be classified as a false negative.

4.5.2 Cost Comparison

The cost comparisons were chosen to show that an RPi would be a cost effective platform for deployment in a community WMN. The cost of deploying and maintaining an IDS system would be borne by the community and in most cases that community would want to minimize the costs of the system. The metrics that compared the RPi with a traditional system are the following:

- The lifetime of a battery powered IDS device. The lifetime of battery powered IDS devices would show that the RPi machine would be able to function in a remote environment and perform its tasks well for an acceptable amount of time.
- The power consumption of the IDS devices. The power consumption would show that the RPi running costs would be less than that of a comparable system.

- The cost for the devices. The cost for the device would show that it is cheaper to purchase and replace in the case of a failure.

4.6 Expected Analysis and Results

The selected PCAPs were run against the RPi system and the baseline desktop system. The captured packets and throughput were varied identically for each capture set. When both systems completed processing the capture sets, the `SNORT` logs were measured for performance and accuracy. The data was then analyzed for the following characteristics:

- Ability to handle the various amounts of throughput on the network.
- The efficiency and accuracy of the detection.
- The cost of implementation and maintenance of the system.

The expected analysis and results are that the RPi would allow a system being attacked to respond in a correct and timely manner and will not hinder the functionality of the WMN. Also, that the detection success rate between the two systems will be comparable in that both systems would detect the same amount of malicious packets at the same rate (RPi vs. Desktop). The Raspberry Pi is also expected to have a throughput threshold that is not overly inferior to the baseline system to be able to say that it is efficient or robust enough to be deployed in an actual WMN. The main expected difference between the two systems will be in the ability to handle large data sets in a timely manner. Remediation Strategies are beyond the scope of this thesis.

4.7 Summary

The testing phase compared the proposed RPi system vs. a standard desktop deployment. Malicious and regular packet samples were used to determine the accuracy of each system. The same packet captures were run on both systems with changing input variables of packet type, attack frequency and volume of attack. The evaluation of the RPi system was measured by the cost and efficiency of the proposed RPi system vs. the standard desktop deployment. By analyzing the performance of the IDS (throughput, detection rate, and detection accuracy) and the cost variables (power consumption, procurement costs, battery life), the suitability of the RPi as an IDS for a community WMN was evaluated.

Chapter 5

Experimental Results

The results from the experiments described in Chapter 4 were promising for verifying the suitability of the RPi as an IDS platform. The RPi machines were able to keep up with scanning when the traffic approached the typical packet level of day-to-day usage for most end users. For typical end user use, the RPi machine was at the same packet loss as the baseline desktop system.

5.1 Overview of Results

The results of the baseline system were comparable in terms of accuracy and memory usage with that of the RPi machine. The places where the RPi lacked was in the processing power, disk space and I/O performance. This affected the speed at which the RPi processed packets and read and wrote to the disk. The RPi machine performed just as well as the desktop baseline in basic testing. When preprocessors are used for DPI on packets coming through the system, the RPi platform performed slightly slower than the baseline.

5.2 SNORT Detection Results

5.2.1 Attacks

As described in Chapter 4, the SNORT IDS was run on both the baseline and the RPi machine. The attacks were all launched at the same time and each attack described was successfully detected.

5.2.2 Common Usage

In the traffic throughput tests as described in Chapter 4, the system was run with commonly-used protocols that would be found in a community WMN (Varied between 1.32 Mbits/s [26] to approximately 11 Mb/s [4]).[1] The RPi was able to keep up with the regular traffic load while performing the IDS inspection; however, the desktop used far less in terms of percentage of memory and processing capacity as the processing power of the desktop can process a simple steady stream of data very efficiently (Figure 5.1 and Figure 5.4). The memory was shown not to be a factor in the processing performance of either system as neither system

[1] roofnet showed expected possible throughput, in reality the traffic throughput was lower.

exceeded more than 15% of memory being used by the SNORT process (Table 5.1). This could potentially allow other less processor intensive applications to be run on the RPi to augment the SNORT IDS. The tests were done in single sample runs to model a typical end user experience. A BitTorrent session can have differing traffic patterns from use to use depending on the amount and location of peers it connects to. The time was sampled at an interval of one second and then averaged over ten seconds to reduce the working data set to a more manageable level.

In the test for video streaming sites such as YouTube the desktop utilized less than 0.45% of total processor capacity (Figure 5.1). On the baseline, any processor spikes were negligible amounting to less than a percentage of total processor capacity. The RPi utilized more CPU than the desktop but as shown in Figure 5.2, the RPi can handle the traffic demands and at the same time not utilize more than 50% of processor usage. As seen in Figure 5.2, the RPi processor usage peaked at approximately 29% utilization when the YouTube Video was being downloaded and the most amount of traffic was being sent through the RPi. As the video went on, the traffic rate lowered and so did the processor utilization of the RPi. The rate picked up near the end causing another processor usage spike as the video hadn't fully downloaded yet (Figure 5.3).

BitTorrent is also responsible for a large percentage of community network traffic. The BitTorrent protocol opens up multiple connections to many peers increasing the traffic rate significantly, this allows for a faster download than a single direct connection. The baseline desktop handled the scanning of the BitTorrrent traffic utilizing 15% of processor capacity for processing (Figure 5.4). The RPi was run during a BitTorrent swarm with the CPU utilization shown in Figure 5.5 and the packet volume shown in Figure 5.6. The processor usage spiked whenever the packet volume did and lowered when the packet rate lowered. The memory usage was consistent at approximately 15% no matter how many extra packets were being processed on the RPi. This indicates that the resource constraint would be the processor and not the memory. The RPi handled the throughput of the torrent without exceeding 45% CPU usage (Figure 5.5).

Table 5.1: Memory Usage

Application	Baseline Snort Memory Usage	RPi Snort Memory Usage
Skype	2.9%	15.1%
YouTube	2.9%	15.1%
Bittorrent	2.9%	15.2%

5.2.3 Attack Scenario

The attack scenarios were executed to see if the RPi could handle detecting the attacks that would be coming into the node. The baseline desktop detected the attacks and alerted the system in all cases. The RPi was also able to detect the scans in the attack scenario. In all cases, the malicious packets were identified as malicious and an alert was logged to the alerts file. The scans described in Section 2.1.1 were all detected and

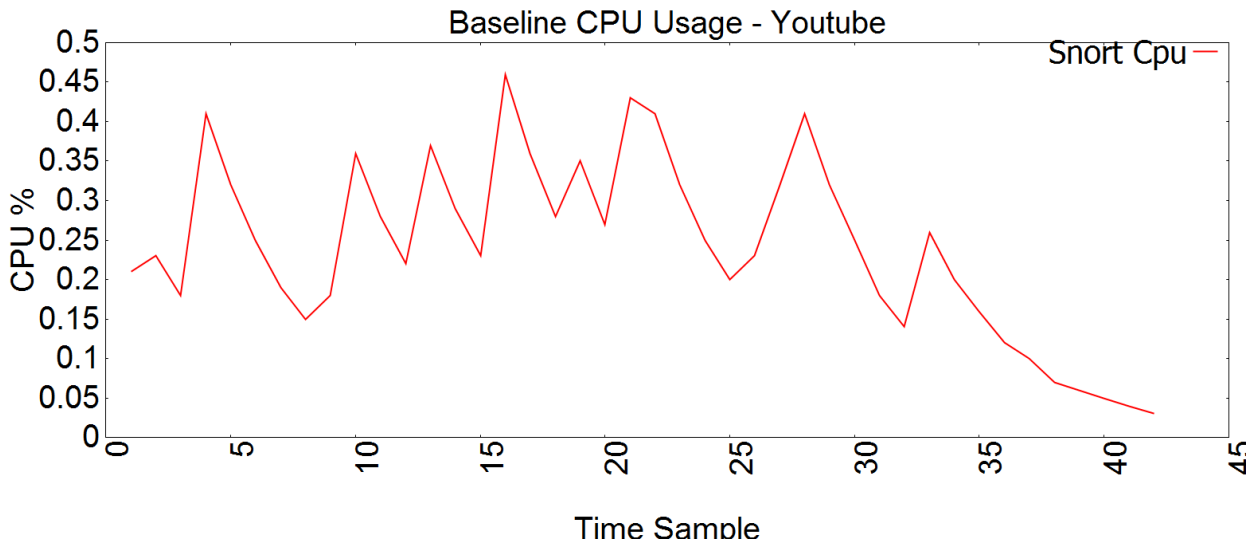


Figure 5.1: Baseline CPU Usage YouTube

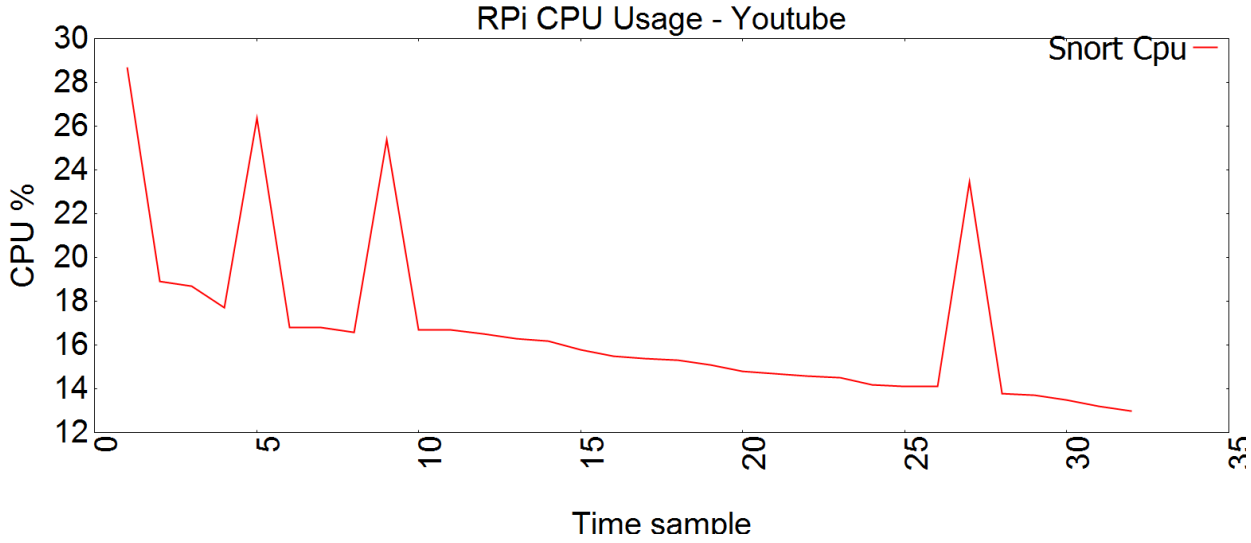


Figure 5.2: RPi CPU Usage YouTube

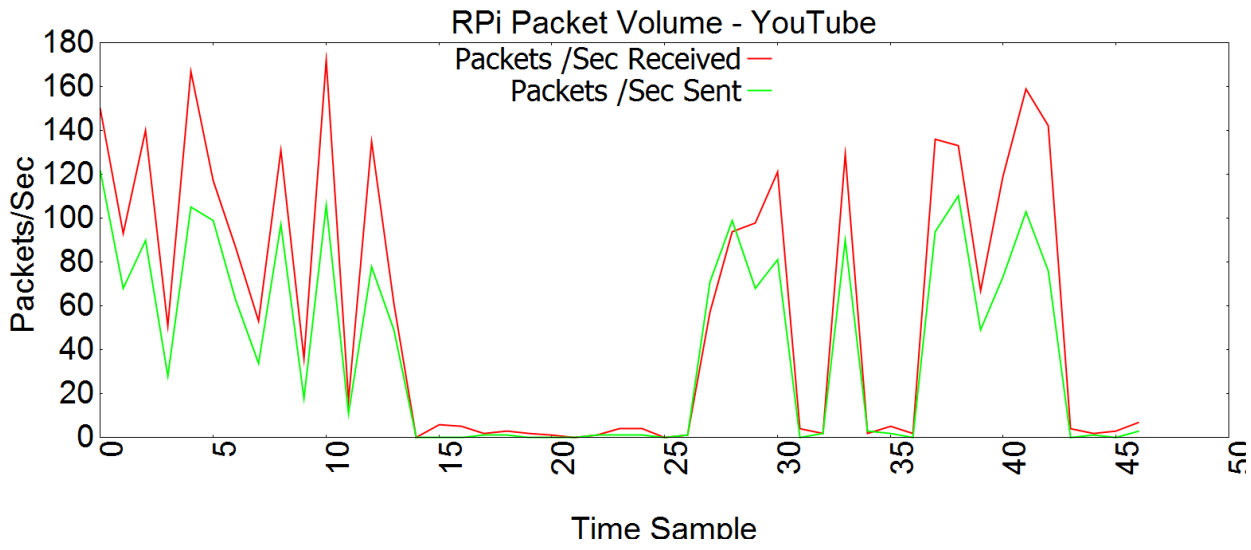


Figure 5.3: RPi Packet Volume - YouTube

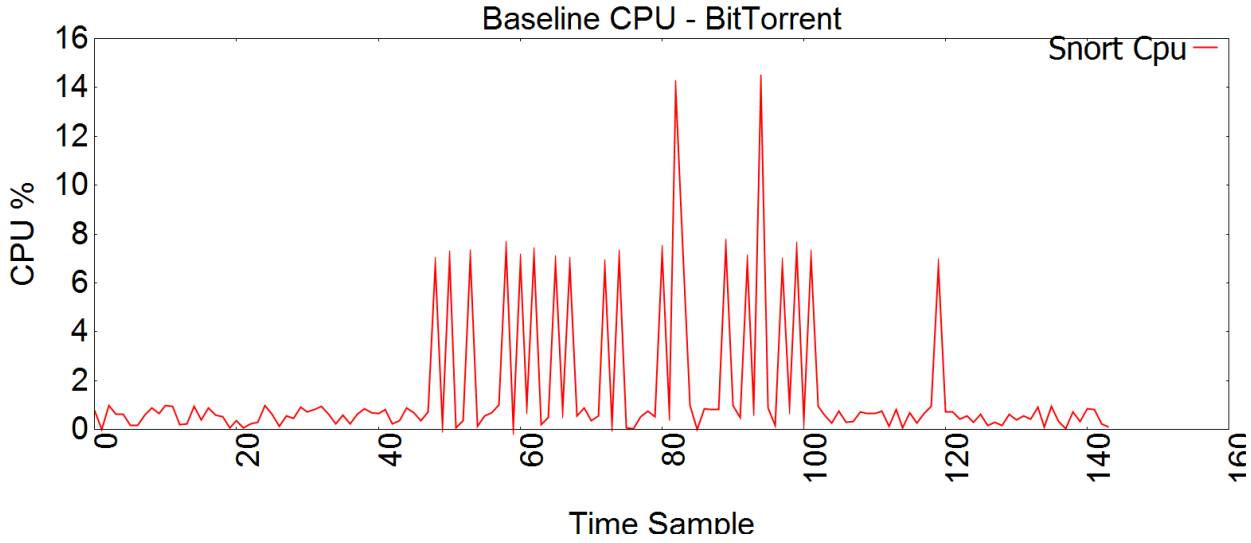


Figure 5.4: Baseline CPU Usage `BitTorrent`

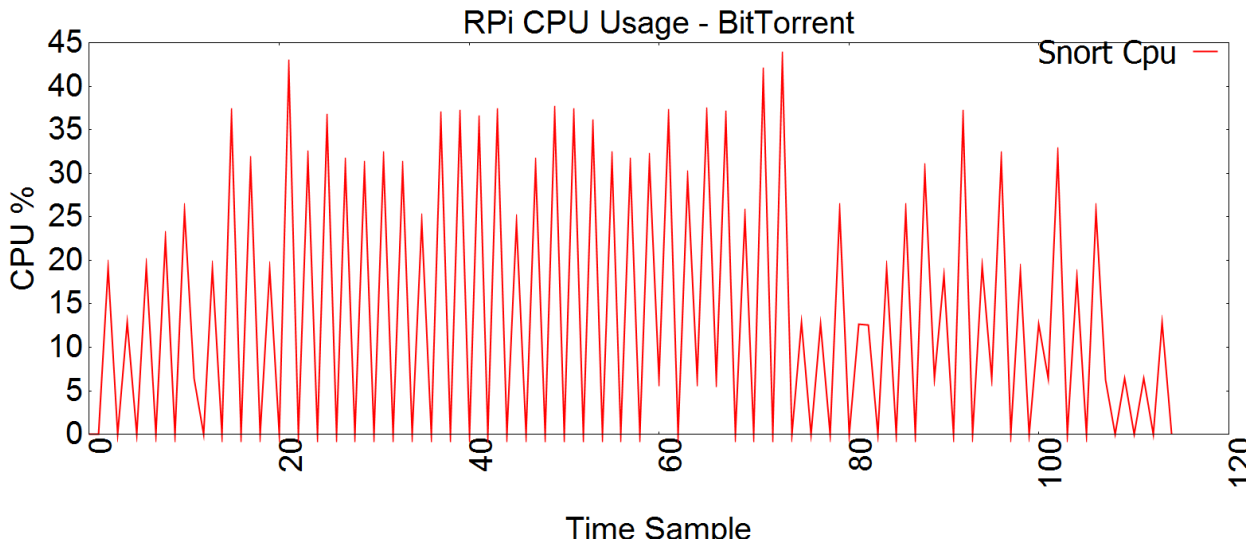


Figure 5.5: RPi CPU Usage `BitTorrent`

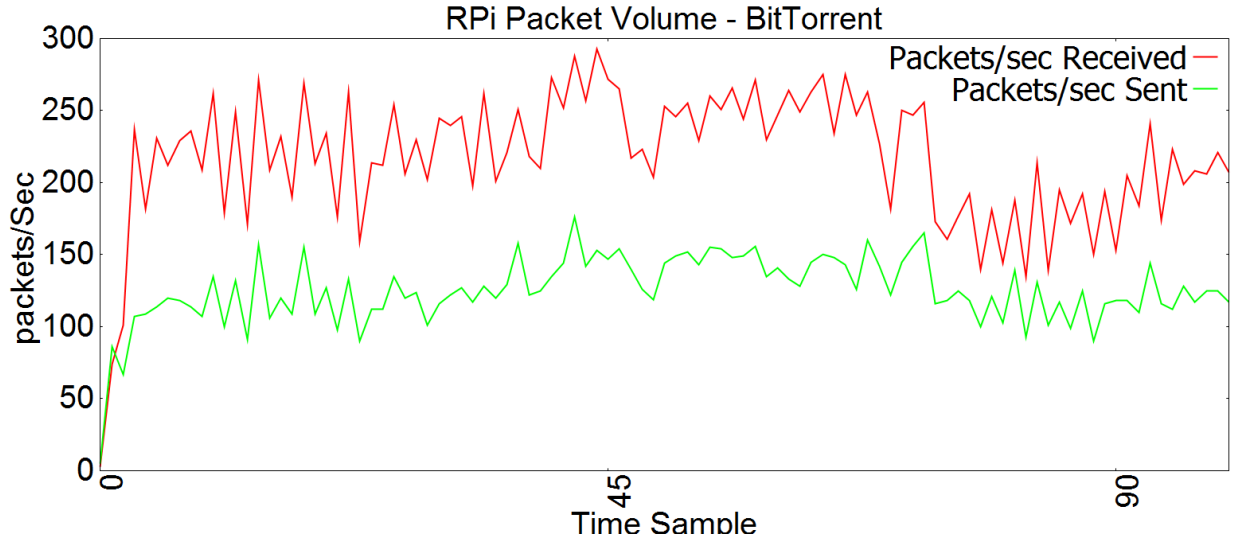


Figure 5.6: RPi Packet Volume - `BitTorrent`

all except the Quick (stealth) scan were detected before they were completed scanning.[2] As both systems were using SNORT, this result was not a surprise.

5.2.4 Detection Results

The detection results were categorized by:

- Detection success rate
- False Positives
- False Negatives.

The first tests were run with an emphasis on detecting the common scanning techniques that an attacker would use to map out a potential victim network. The tests were done with NMAP, an open source network scanning tool [29]. Both systems were attacked at the same time by the NMAP scan. The alert logs were monitored with a small python script. When an alert was logged, the system would increase the alert count. ICMP and UDP heartbeat negatives were sometimes classified as alerts in the SNORT log.

The SNORT dump from Appendix A.2 shows that all packets on the scans in the RPi machine were caught and processed during a Skype call. In Appendix A.1, the total percentage of scanned packets was 100%. No packets were dropped or let through without being scanned. All packets were accounted for in the SNORT logs, and the RPi was able to keep up with the traffic volume.

The download and upload rates were the same in both test systems. The RPi machine showed no dropped packets and a 99.99%[3] analyzed packet rate (Appendix A.3). The BitTorrent had some connections that were still in a CLOSE WAIT state to other computers when the BitTorrent and SNORT applications was closed to retrieve the summary log. The RPi was able to detect all the scans that were sent against the network at the same success rate as that of the baseline desktop machine.

5.2.5 False Positives and Negatives

This section describes the testing for regular data on regular PCAPs and then mixing in malicious or malformed PCAPs. During regular usage of the YouTube stream, Bittorrent swarm and Skype calls, the false alerts rate was measured (Table 5.2). In each of the cases the false positive rate was below 1% with the YouTube and BitTorrent well below 0.1%. The Skype call was greater than the other two at 0.78%, due to a rule that was finding the advertising in the Skype program suspicious. Both systems had approximately the same false positive rate (Table 5.2).

[2]The stealth scan would never actually connect to the system, as a result, no information was truly sent to be scanned and could not be identified as malicious; instead the detection occurred when the malicious packet payload was sent to the target computer after the scan completed.

[3]The outstanding packets in the BitTorrent swarm were associated with data connections that were still active with a close wait status when the SNORT application was shut down.

Table 5.2: False Positive Detection

Application	False Positive Percentage
Skype	0.78%
YouTube	0.04%
Bittorrent	0.07%

It is more difficult to determine a false negative rate without coming up with a new methodology to counteract the SNORT rule set, which is updated frequently. Since the rules are updated frequently much like an anti-virus program, there are new rules that are added at each iteration. This makes it difficult to have older attacks not be detected as once an attack is discovered and studied, a rule is usually made to counteract it. When the malicious packets (Teardrop, Tbot, and Poison Ivy) were run against the system, all packets were flagged as malicious by the RPi system and the baseline desktop.

5.2.6 Frequency of Attack

The frequency of the attack (Section 4.4.2) did not affect the detection capabilities of the RPi. As the number of attacks increased, as long as the packets/sec did not increase past the 700 packets/sec threshold as shown in Section 5.2.7, the RPi would be able to handle the throughput for the system. Each malicious packet was detected in the traffic stream no matter how many times it was inserted into the stream.

5.2.7 Volume of Attack

As the volume traffic increased, the processing overhead increased. The packets consisted of TCP packets with a 1200 byte payload. The baseline system approached 7% CPU utilization when the packets/sec approached 700 packets/sec and a throughput of approximately 6.5 Mbit/sec the (Figure 5.7). Using extrapolation, the maximum for the baseline system would be around 90 Mbit/sec. In comparison, the max for the number of packets a second that the RPi could handle was around 7.5 MBits/sec (800 packets/sec) on average (Figure 5.8 & 5.9). In these figures, the data points represent the average of 10 experimental runs of traffic of 10 seconds, at various packet intervals from 50 packets/sec to 900 packets/sec. The RPi would still be able to handle the 7.5 Mbits/sec without increasing CPU load; however, once the data rate went up, the load of the CPU went over 100% meaning that the packet processing tasks were now being delayed and there would be no time for other processes. Once 100% CPU usage was reached, it would begin dropping packets in IDS mode. In the case when IDS mode is not configured, it began letting the packets through without dropping them, leading to a potential compromise. The results demonstrated that the desktop can handle much more throughput than the RPi. The desktop would not be deployed on a local WMR, however, thus it would not be able to detect the traffic that is being sent through the WMR. Also, as DDOS attacks are approaching packets/sec rate in the millions [42, 44], not even the baseline system would be able to keep up with the packet rates without utilizing 100% CPU. Overall CPU utilization increases with an increase

in packet volume, and the RPi CPU increases at a higher rate than the baseline. Measurement granularity can prevent exact volume to CPU usage mapping, however, the increase in CPU utilization trends upwards when the packet volume increases.

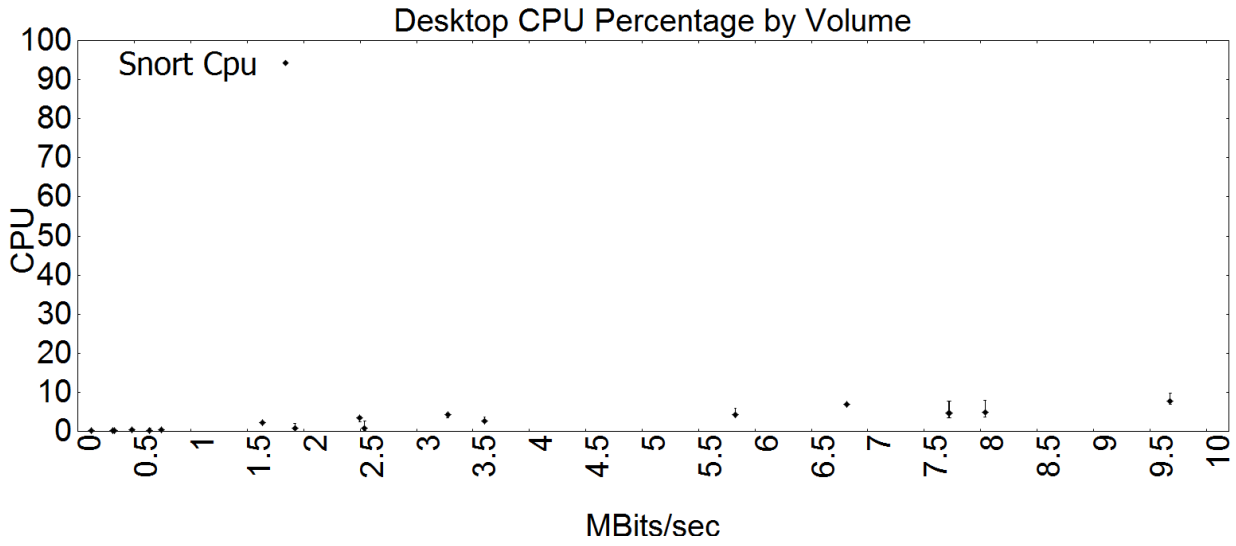


Figure 5.7: Baseline CPU Utilization by MBits/Sec

In all instances, the RPi detected the malicious packets in the traffic stream that it processed and raised an alert. When the volume of attack exceeded 100% CPU utilization on the RPi, the RPi in the experiment would drop or delay the packets from reaching the intended target. In a WMN environment, with the RPi not acting as a physical connection to the WMR, the packets that were in excess of the traffic threshold could be allowed into the system.

5.3 Processing

The processing requirements for the RPi machine were the key measure for the feasibility of deploying a low-cost, low-power IDS. To help the RPi use the least amount of resources, the RPi was run with fast processing mode. The fast processing mode allows the `SNORT` program to dump the logs in `TCPDump` format and produce only minimal alerts. This means that the logs were written in binary format rather than a text based human readable format. The logs would then have to be reanalyzed after the logging session is complete to view the true details of the alerts and sniffers and make the logs human readable.

When the network was idle and not a lot of information was coming through the router, the desktop machine had an average of 1.8% CPU usage but there were other processes or daemons that were running on the baseline system that could explain the extra usage as the `SNORT` process was not using any more than 1% of CPU. In contrast, the RPi was sitting on an average of 6.3% CPU usage and only used 98MB of the 485MB available. The maximum processor and memory utilization for both the baseline and RPi systems

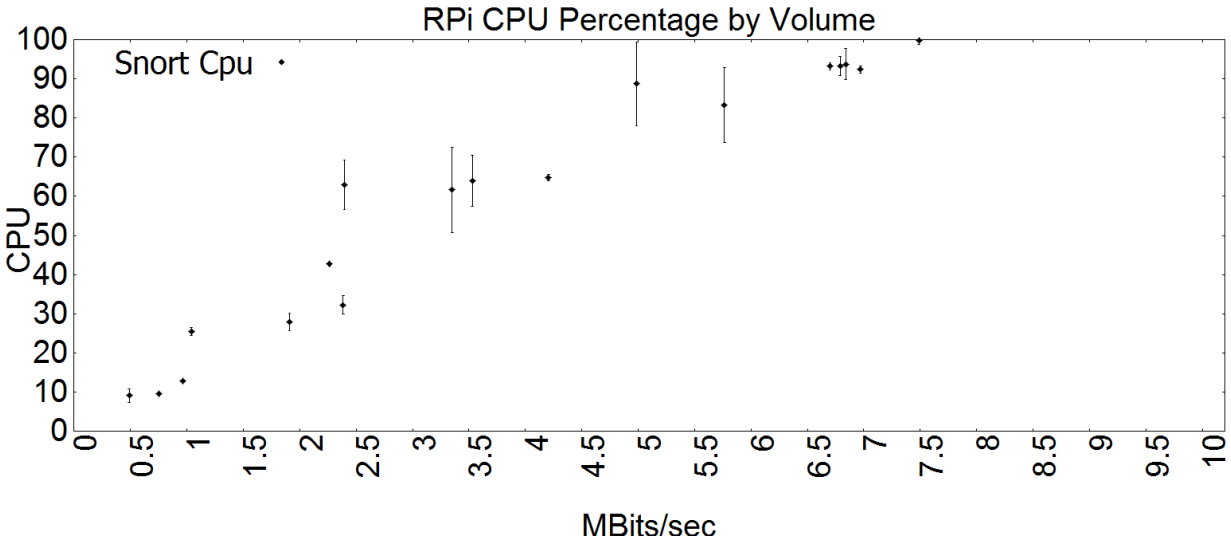


Figure 5.8: RPi CPU Utilization by MBits/sec

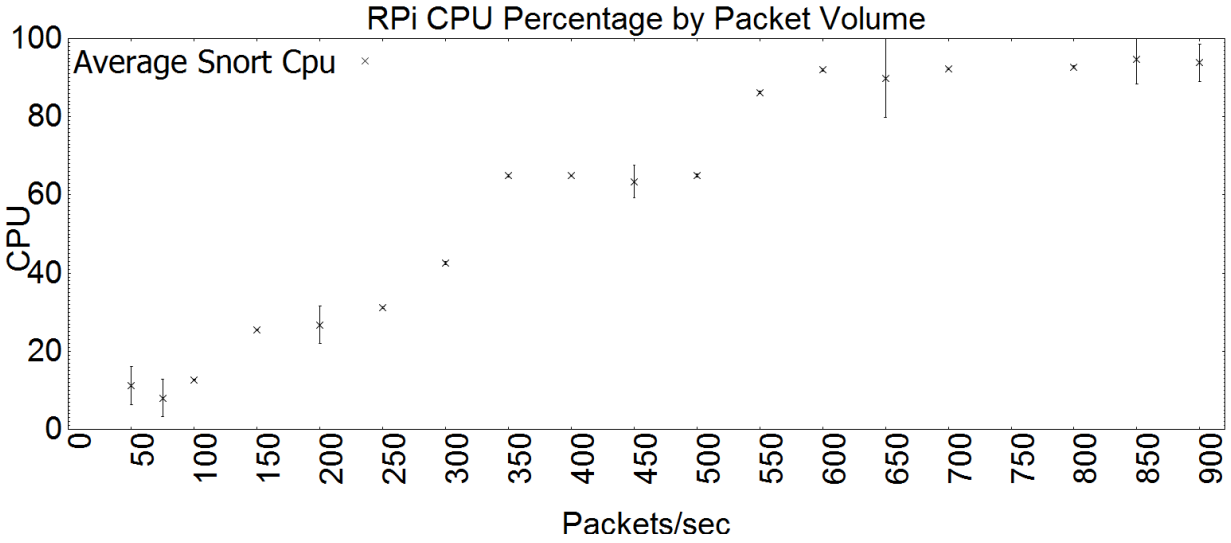


Figure 5.9: RPi CPU Utilization by Packet Volume

under different attack and regular usage scenarios can be found in Table 5.3. When an attack commenced that included scanning the SNORT protected systems, there was a high increase, typically 25-40%, in the CPU usage of the Raspberry Pi machine while the baseline desktop had a negligible usage increase of 1-2%. This was due to the slower processor and memory capacity of the RPi. The difference in processing power came when attempting to increase the throughput of the scanning system. The RPi began to start hitting more than 50% usage when the scans were running at an increased rate while the system was already under heavy network load (approximately 300 packets/sec). The CPU usage was acceptable for the RPi machine to perform the processing for a scanning attack.

Table 5.3: Processor & Memory Load Under Various Network Usage

Scan type	MAX CPU Utilization (Baseline)	MAX Memory Utilization (Baseline)	MAX CPU Utilization (RPi)	MAX Memory Utilization (RPi)
Quick Scan (11.2 sec)	1.3%	500 MB	7.5%	101MB
Regular Scan (37.3 sec)	1.3%	581 MB	17.5%	101MB
Intense Scan (73.12 sec)	1.3%	600 MB	43.5%	101MB
Slow Comprehensive Scan (1231.15 sec)	2.6%	600 MB	36.2%	101MB
YouTube video	1.3%	581 MB	15.5%	101MB
BitTorrent download	1.3%	500 MB	45.0%	167 MB
Skype call	2.6%	600 MB	16.2%	168 MB

As an interesting side note, initiating an SSH session for the RPi machine caused the RPi CPU to go above 50% usage on a regular basis. When attempting to process packets and defend against scans, the RPi had a noticeable slow down in terms of responsiveness in the SSH session. When the CPU usage begins to move upwards, this can lead to slow responsiveness and potential overheating of the device leading to device failure.

5.3.1 Time to Detect

The time to detect was compiled by noting the time that the malicious packet was received into the network, then the time that the item was flagged by the IDS was logged and the two times were subtracted from each other. The total attack time was considered to be the scan plus the launching of the malicious packets. In this case, it is assumed the attacker would begin to explore the network by scanning it with a tool such as Nmap. Once the attacker detected a vulnerability, they would initiate the sending of malicious data to the target computer. The time the attacker initiated the scan of the local node, attempting to find a host to exploit or network to map was logged as the start time. The time that a scan was finished and a malicious packet was sent to the target computer was considered the end time.

To measure the time it would take for a single packet to be detected (Poison Ivy, teardrop, etc.), the time that the log would show a new packet in the log and the time it was processed and matched with a SNORT rule would be logged. The difference between the two logged times is the detection time. With the

scanning being completed in real time, the scanning based off the SNORT syslogs indicated that in the case of a non-match for a regular packet, the time to process one of the packets would be, in most cases, lower than 20 ms. When a packet was matched to a rule, an alert was raised. The Baseline System was more effective in that matching was fairly consistent at 2 ms. The detection time would range from 2 ms to 18 ms on the RPi. Under low load (10 packets/sec) it was more frequently under 10 ms, while under high load (300 packets/sec) (Figure 5.10) the detection time could reach up to 20 ms. Under high load, the average detection time for the RPi was 9.14 ms and with a low load, the average time was 4.87 ms.

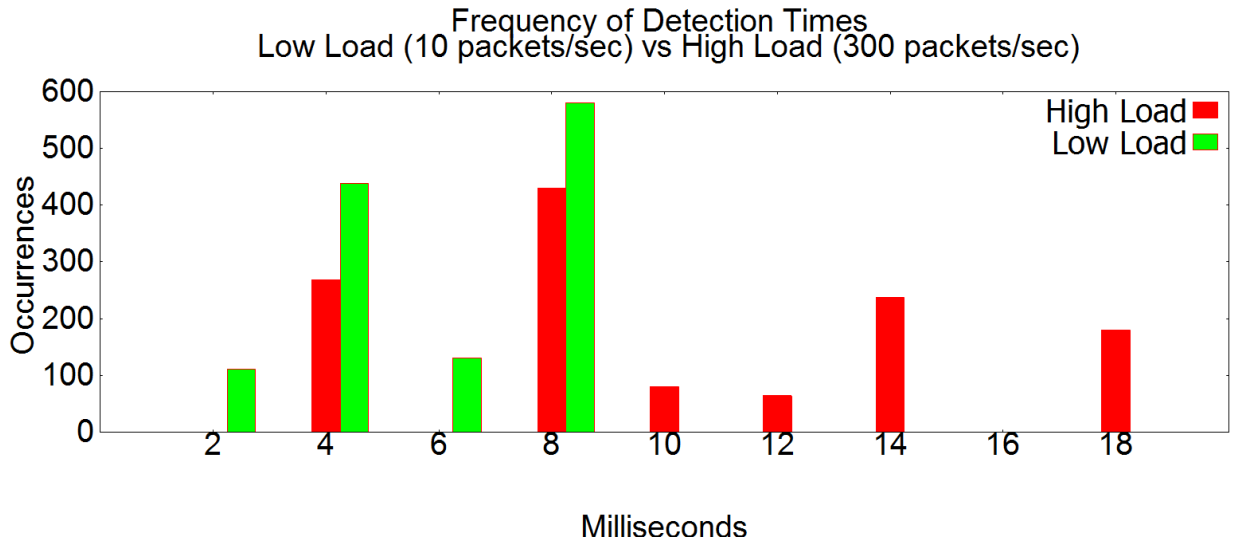


Figure 5.10: RPi Detection Time Under Various Loads

5.3.2 Remediation

The SNORT program was run on both systems, with an identical rules set-up. The network was closed off to outside traffic. Only the test machines and an attacker machine were present on the network. Assigned IP addresses were as follows: baseline desktop was assigned the IP address of 192.168.0.101 and the RPi machine had the IP address of 192.168.0.104. When an alert was raised by the SNORT IDS, the offending IP was logged in the alert log. When the rate of writing to the log reached over 30 occurrences per minute, a new rule was created to not allow any of the traffic coming from the IP address to be whitelisted (Section 2.1.1). The offending IP was also added to a non-enforced blacklist that would update a log when the IP would send traffic. In these cases, the attacker node would have been blacklisted in most cases, with most common scanning techniques performed via NMAP.

5.4 Economic Results

The economic evaluation of the system was one of the major system properties analyzed. Among the cost associated variables were the following:

- Power Consumption cost, and
- System Costs
  - Cost of the system for initial capital investment
  - Cost of replacement of various system components.

These variables were evaluated to determine what level of protection could be achieved by the comparative systems with equal financial resources.

5.4.1 Power Consumption

To measure power consumption when the CPU was fully utilized, a simple stress program was written: a worker process was spawned that would perform the `sqrt()` function repeatedly, thereby tying up the available CPU resources. The power consumption of the RPi machine was very much smaller than that of the desktop machine. Ten separate readings were taken when each device booted, after booting at idle, and at 100% power utilization. These results can be seen in Table 5.4. The power consumption readings on the desktop were also backed up by various benchmarking websites [31, 67]. The Raspberry Pi, even if running at 100% CPU capacity, would use far less power than the desktop running at idle.

Table 5.4: Power Consumption.

System Load	Raspberry Pi	Desktop
Idle	2 Watts	35 Watts
Typical IDS Usage (`SNORT`)	2 Watts	65 Watts
Maximum CPU usage	˜3 Watts	72 Watts

To figure out the amount of time a device can last off of a battery the following equation was used:

$$hours = (volts * amperehours)/watts. \tag{5.1}$$

Due to the low power requirements of the Pi, it is feasible that it could run off of a 12 volt 125 ampere-hour battery for approximately 750 hours at typical IDS usage (250 packets/sec), 500 hours if full CPU utilization is achieved (>450 packets/sec).

5.4.2 System Costs

The system costs of the RPi machine were compared to the desktop IDS. The key items were the price of acquiring the devices (initial system costs) and the cost of deploying and maintaining each IDS (maintenance cost and replacement costs).

Initial System Costs

The initial system cost for each Raspberry Pi basic kit, which included a power supply, case, and 8GB SD card with the Raspbian OS included, was $59.99 CDN plus tax. The system cost for the desktop computer with an Intel i3 processor and 4GB of RAM and 1 TB HD at the lowest price was $369.99. Approximately 6 RPi machines could be deployed for the same cost as this desktop configuration. The RPi configuration has the benefit of being able to protect the network by being deployed throughout the network. This would allow the network to be protected via multiple detection points. Multiple devices could warn each other if an alert is found and alter the capture rules accordingly. The costs of the RPi system and the desktop, as well as various component replacement costs are shown in Figure 5.5.[4]

Table 5.5: Capital Costs & Component Costs

Item	Unit Price
Raspberry Pi Model B Starter Kit	$63.31
32 GB SD Card	$19.00
8 GB SD Card	$9.50
Raspberry Pi Casing	$4.20
Raspberry Pi Model B Board	$35.00
intel i3 desktop	$369.99
80 GB HDD	$30.00
i3 CPU	$137.00
Mother Board	$56.47

The cost differences can be approximated with real world examples. The MIT Roofnet [4] had in the setup approximately 27 nodes with 5 wired gateways that allowed access to the internet. Assuming that each gateway would utilize a traditional desktop IDS to monitor the traffic, and that each node would contain an RPi to monitor traffic to and from the node, the total security capital costs for a WMN such as roofnet (27 RPi and 5 Desktops) would lead to a savings of $77.27. The benefit of the RPi is that they would also prevent attackers from inside the network and not just traffic that is coming in through the gateways.

For Wray's 10 WMRs and one gateway node [33, 43] the cost would be $633.00 for the RPi devices vs. the $369.00 for the single IDS desktop at the gateway. Using only a single gateway based desktop IDS in both of these examples, the only traffic that would be screened would be the traffic that uses the gateway as a go-between. Any traffic between the internal nodes would pass undetected.

[4]Prices are an average of prices taken from canada.newark.com, amazon.ca, and canakit.com for the Raspberry Pi. While the prices for the desktop system and its components are averages from prices taken from tigerdirect.ca , ncix.com, and amazon.ca

Maintenance Cost and Replacement costs

The costs to maintain and replace faulty parts on a device also need to be factored in to the economic comparison. If a RPi node goes down, depending on the point of failure, the parts are fairly easy to replace. The cost of replacement is limited to either an SD card or the RPi itself which currently costs approximately $35. The most common occurrence of failure on the RPi machine is the SD card. The 8GB replacements are available from many online retailers for an average price of $9.50 per card (Table 5.5), making it a trivial piece of hardware to replace, unlike a HD drive for a desktop system. If the RPi itself goes down (i.e mother board, CPU failure), then the cost of the replacement part is $35 vs. $100 or more for a CPU or motherboard on the desktop (Table 5.5). The low maintenance and replacement costs add to the viability of the RPi as an alternative to a traditional IDS.

Also, with the low power consumption of the RPi, the system will save money on power utilities. Using the equations below and a price of 10.2 cents per kilowatt hour (currently the price per kilowatt hour in the United States) [3], the power consumption costs can be considerably lower than a desktop system.

$$kiloWatthours = Watts * Hours/1000 \tag{5.2}$$

$$TotalPrice = Price * KiloWattHours \tag{5.3}$$

The monthly cost per RPi machine is $0.081 a month vs. $4.84 a month for the traditional desktop. The monthly costs can be estimated for existing WMNs. For MIT's Roofnet, the energy consumption costs using RPi devices would be approximately $2.27 per month vs. $24.20 per month for the desktops. Over a year of operation the RPi would cost approximately $263.19 less on power consumption alone. For the Wray project with 10 WMRs and one gateway node the cost would be $0.81 per month; a savings of $48 per year over the gateway IDS.

5.5 Advantages

The advantages of the RPi system are that the RPi can be deployed with any node in the network and not just the ones that are connected to the Internet acting like a backbone for the network. Therefore, 100% of traffic would be monitored, be it external to the network or internal to the network such as intruders walking within range of a local WMR, connecting to it, and attempting to compromise it from within. The total area coverage is greater than that of a single IDS that listens on a single gateway node. With each RPi covering an outdoor range of 100m outdoors and 35m indoors the RPi system could cover an area at least 109.9m^2 per node. To cover the village of Wray with their 10 nodes, the total cost would be approximately $633. With the desktop, a greater amount of potential traffic can be monitored at a single node, however, the internal WMN traffic is not monitored. To ensure all traffic is monitored, a desktop system would need to be installed at all nodes of the WMN, escalating the cost significantly. To cover Wray's 10 nodes with

desktop IDSs would cost over $3699.90 for the setup. Another advantage of the RPi system is that, due to its small size, it can be stored in a small sealable space that can be outside, whereas a desktop is not very well suited to outdoor operation.

5.6 Limitations

The largest limitation is, of course, the processing power of the RPi machine. The experiments in this thesis have shown that it is able to handle daily use traffic, without beginning to slow down or drop packets. There are two possible minor additional limitations with the RPi machines: the limited amount of disk space and the longevity of the SD cards. The disk space on a RPi is limited to the largest available SD card. At this time that is only 32 GB [20]. The SD card forms the basis of the stable storage of the RPi. An IDS can have multiple log files running at once, and it can read and write constantly to each of them. The number of writes and reads that can occur in such a system tend to cause the SD Card to fail sooner than SD cards under different usage, for instance, digital video, photo, and music storage.

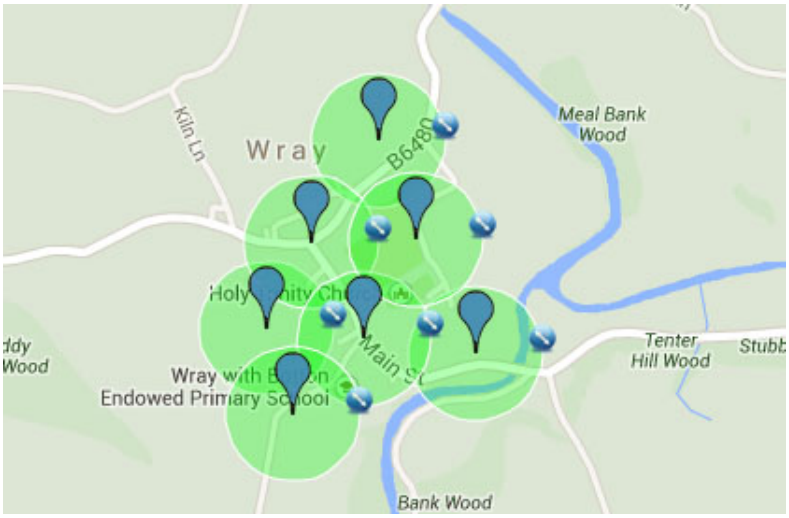
Both of these limitations can be resolved by moving the logging from the IDS to a remote logging device. This logging device could be located physically with the RPi and connected via USB, or it could be a network log storage that is easily configured on Linux with `rsyslog` (Section 2.1.1). This would both save on disk space and reduce the amount of writes and reads to the SD card. The latter method of using `rsyslog` would increase the amount of network traffic and could increase the burden on other nodes. There is, as with most network systems, an issue of scaling. As the number of nodes increases, so does the number of security nodes alongside them. If the number of deployed nodes increases and if there is not a proportional increase in gateway node IDS security, the capital cost becomes larger for the RPi devices vs. the desktop. Deploying an RPi at each node no longer becomes cost effective. Instead, gains can be made in terms of security by strategically placing the WMR nodes with an RPi system in locations that would provide greater coverage in terms of area. For example, in the case of Wray, depending on where the nodes are located, there can be a great improvement to the range of the WMR. A regular, effective WIFI signal[5] varies between 90ft radius and 330ft radius. If a 90ft radius is used, then approximately 25 WMR nodes would have to be deployed to cover the village of Wray (Figure 5.11). If, however, the WIFI nodes are located outside with minimal obstructions, then a more manageable 7 nodes are required (Figure 5.12). If we were to overlay a North American city block on the map, we can see that approximately 1 city block of 4.5 acres could be covered by one WMR node (Figure 5.13).

Depending on the layout and the routing of the nodes, the need for security on every node could be eliminated, placed instead only on every second node. This setup would allow for a node to be compromised from within its own connection area if an attacker connected and compromised the node itself; however, any

[5]This varies between -30 dBm (decibels) which is the maximum achievable rate to -70dBm which is the minimum signal strength for packet delivery [16]

Map Images ©maps.google.com

Figure 5.11: Wray Coverage with 90ft WIFI Radius



Map Images ©maps.google.com

Figure 5.12: Node Coverage with 330ft WIFI radius

Figure 5.13: Comparison of a Typical North American City Block with Wray.

connectivity between a compromised node and a security node would still be scanned. For instance, if there was a routing layout such as the one shown in Figure 5.14, the RPi would be included on Nodes 1, 3, and 7 and every interaction would be monitored from node to node. An attacker could not compromise more than one potential node without having to run their attack on another node that was running an IDS.

5.7 Summary

The Raspberry Pi has shown itself to be resilient for the purpose of monitoring a network and a good value for networks with limited financial resources. The cost effectiveness has proven that a low power, low cost device can handle a specific task with a reasonable performance rate and it will keep working at an effective rate. While the performance will lag behind a traditional system, the low cost and ease of replacement of the system outweigh the reduced performance.

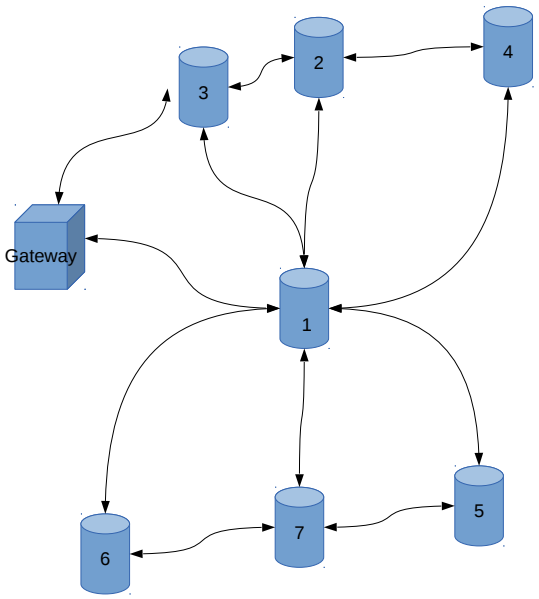


Figure 5.14: Example Routing Layout

Chapter 6

Conclusion and Future Work

6.1 Conclusion

This thesis has shown that a low cost, low power device can be used as a viable alternative to traditional desktop Intrusion Detection Systems. The cost of the system is substantially lower than a traditional IDS and the RPi can be adapted due to its low cost and low power consumption to run on battery systems in remote areas. This can allow community WMNs to cover more area as nodes can be placed in remote areas. By saving on costs and power, the RPi machine can fit into the basic security needs for a community WMN.

The main advantage to such a system is that the RPi setup will be able to monitor all network traffic at all nodes. A traditional IDS would only be able to monitor the one node in the system that is connected to the internet gateway. If an attacker is located within the network or a compromised device joins the network and attempts to compromise other nodes on the network, the RPi system set up would be able to detect the intrusion at the local connected node level. Also, due to the locality of the WMRs the offending device could be easier to locate physically, so variants of malicious devices such as `CreepyDOL`[53] would be easier to locate as well as the would-be malicious attackers that deployed them.

The advancement of new devices, from Raspberry Pis to phones (Android, Apple, Blackberry etc.) that have the same processing power as older desktop computers allows for deploying security solutions to these new devices. The versatility of these low power devices, their power consumption savings, and lower price points compared to traditional security deployments, can be beneficial to smaller rural community Wireless Mesh Networks. By utilizing these devices as components of a greater overall security strategy, even smaller networks can have improved digital security.

This thesis has evaluated the Raspberry Pi as a deployed IDS for a small WMN. With regular traffic usage, the RPi was able to keep up with the scanning requirements for common applications and common attacks that would be found in a community WMN. The RPi is a potential security augmentation for a community Wireless Mesh Network's security.

6.2 Future Work

Due to the RPi experiments being a proof of concept prototype, there are many potential applications of RPi or other low-cost, low-power devices with respect to security. By using the advantages of the low power devices, there are many different areas of research that can be explored.

6.2.1 Optimal Security Deployment Schemes in Ad-Hoc Networks

If a low cost, low power device cannot be deployed at each node due to financial or other environmental concerns, then future study of the deployment and security node placement could show an ideal deployment scheme for WMNs. Depending on the coverage area and the node configuration, studies could be done on different deployments to show how to deploy a security node to key nodes that would maximize the amount of traffic scanned, while minimizing the cost of deployment. This could lead to a more secure WMN in the future without having some of the scaling problems described in 5.6.

6.2.2 Self-Mitigating networks

By deploying the RPi machines in a WMN and then giving them the ability of detecting an intrusion and or virus, they can also be used to mitigate a compromised machine by removing it from the network, or applying a patch to the compromised software. The RPi machines will, however, have to be granted elevated privilege on these machines which could lead to some security questions. Since they are self mitigating, they would also require less oversight from actual humans they must be thoroughly examined to ensure that a compromised state could not be entered into by mitigating the security breaches to much.

6.2.3 Self-Upgrading Nodes

With the increase in computing power that will give future Raspberry Pis more hardware specs and performance, it is possible to utilize a deployment service such as `Ansible` to deploy a more comprehensive solution to each node. `Ansible` is software that deploys other software from a deployment server to one or multiple clients via SSH. The `Ansible` server can either have all the required installation and configuration files stored on the server or can download them from the Internet via install managers such as `yum` or `apt-get`. The server then copies and installs the files via remote commands issued via SSH login. There is no configuration that has to be configured on the clients other than having an account that can use the SSH connection. `Ansible` would allow for a potentially better mitigation and update system for the RPi. If a node is found to be compromised, a new install could be initiated or an update begun, without having to have physical access to the node itself.

6.2.4 Self-Repairing Security Nodes

Securing the nodes and repairing them once they have been compromised is also a priority for future research. When a node is compromised, it affects the throughput and security of the network. A cooperative methodology can be devised, such that when a node is compromised, it can re-flash itself and rejoin the network as a new node. With the node resetting itself and removing the old state of the system, it would prevent attackers from making long term use of a compromised node. The self-repairing nodes will be able to provide a robust security system for wireless mesh networks.

6.2.5 Mobile Security Devices

With the small dimensions of such devices there is the possibility that the devices could be attached to a mobile platform (a platform that moves i.e. a remote controlled car, a drone, etc., not an iOS or Android phone), and as such could be modified to handle a larger area of security with intermittent connectivity. For example, if a certain device monitoring certain parts of a network detects that there is more suspicious traffic coming from a distant part of the network that has no monitoring, it can then relocate itself to monitor the anomaly.

6.2.6 Summary

The future work with low-cost, low-power devices with a security application is very promising. With low-cost, low-power devices becoming more and more ubiquitous and common, applications for these devices will always be available. Security will be a constant concern for all networks new and old, large and small. With increasing digital interaction, there is the possibility that even smaller networks can be an entry point for malicious attackers to gain entry into larger more prized targets. Security will always need to keep up with the ever changing landscape of modern networks.

References

[1] Evgeny Abramov, Maxim Kobilev, and Oleg Makarevich. Using quadrocopter as a pentest tool. In *SIN '13 Proceedings of the 6th International Conference on Security of Information and Networks*, pages 404 –407, Aksaray, Turkey, Nov 2013.

[2] Jithra. Adikari. Efficient non-repudiation for techno-information environment. In *First International Conference on Industrial and Information Systems*, pages 454–458, Peradeniya, Sri Lanka, Aug 2006.

[3] U.S. Energy Information Administration. Table 5.6.A. Average Retail Price of Electricity to Ultimate Customers by End-Use Sector, by State, January 2014 and 2013 (Cents per Kilowatthour). *http://www.eia.gov/electricity/monthly/epm_table_grapher.cfm*, March 2014. Accessed Sept 2014.

[4] Daniel Aguayo, John Bicket, Sanjit Biswas, and Douglas De Couto. MIT Roofnet: Construction of a Production Quality Ad-Hoc Network, Poster. In *9th Annual International Conference on Mobile Computing and Networking*, San Diego, CA, Sept 2003.

[5] Ian F. Akyildiz, Xudong Wang, and Weilin Wang. Wireless Mesh Networks: A Survey. *Computer Networks and ISDN Systems*, 47(4):445–487, 2005.

[6] Tuomas Aura, Pekka Nikander, and Jussipekka Leiwo. DOS-resistant authentication with client puzzles. In *Revised Papers from the 8th International Workshop on Security Protocols*, pages 170–177, Cambridge, UK, 2001.

[7] Dhiman Barman, Jaideep Chandrashekar, Nina Taft, Michalis Faloutsos, Ling Huang, and Frederic Giroire. Impact of IT Monoculture on Behavioral End Host Intrusion Detection. In *WREN '09: Proceedings of the 1st ACM Workshop on Research on Enterprise Networking*, pages 27–36, Barcelona, Spain, Aug 2009.

[8] Hal Berghel. Identity theft, Social Security Numbers, and the Web. *Communications of the ACM*, 43(2):17–21, 2000.

[9] Bram Bonne, Arno Barzan, Peter Quax, and Wim Lamotte. Wifipi: Involuntary tracking of visitors at mass events. In *2013 IEEE 14th International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, pages 1–6, Madrid, Spain, June 2013.

[10] S. Bose and A. Kannan. Detecting denial of service attacks using cross layer based intrusion detection system in wireless ad hoc networks. In *International Conference on Signal Processing Communications and Networking*, pages 182–188, Chennai, India, July 2008.

[11] Nan Chen. Next steps in municipal wireless mesh networks. *http://searchnetworking.techtarget.com/feature/Next-steps-in-municipal-wireless-mesh-networks*, 2006. Accessed June 2014.

[12] Thomas Chen, Geng-Sheng Kuo, Zheng-Ping Li, and Gou-Mei Zhu. *Security in Wireless Mesh Networks*. CRC Press, Taylor and Francis Group, Boca Raton, FL, 2008.

[13] Pasquale Donadio, Antonio Cimmino, and Giorgio Ventre. Enhanced Intrusion Detection Systems in Ad Hoc Networks Using a Grid Based Agnostic Middleware. In *AUPC '08: Proceedings of the 2nd International Workshop on Agent-Oriented Software Engineering Challenges for Ubiquitous and Pervasive Computing*, pages 15–20, Sorrento, Italy, Jul 2008.

[14] Paul Dourish and David Redmiles. An Approach to Usable Security Based on Event Monitoring and Visualization. In *NSPW '02: Proceedings of the 2002 Workshop on New Security Paradigms*, pages 75–81, Virginia Beach, VA, Sept 2002.

[15] J.L. Duarte, D. Passos, R.L. Valle, E. Oliveira, D. Muchaluat-Saade, and C.V. Albuquerque. Management issues on wireless mesh networks. In *2007 Latin American Network Operations and Management Symposium.*, pages 8–19, Sept 2007.

[16] Jakob Eriksson, Hari Balakrishnan, and Samuel Madden. Cabernet: Vehicular content delivery using WiFi. In *Proceedings of the 14th ACM International Conference on Mobile Computing and Networking*, MobiCom '08, pages 199–210, San Francisco, CA, 2008.

[17] Contagio Exchange. Contagio exchange - contagio community malware dump. *http://contagiodump.blogspot.ca/2011/07*, July 2011. Accessed June 2014.

[18] Mike Fisk and George Varghese. Fast Content-Based Packet Handling for Intrusion Detection. Technical Report CS2001-0670, La Jolla, CA, 2001.

[19] Richard Ford. Malcode mysteries revealed [computer viruses and worms]. *IEEE Security & Privacy*, 3(3):72–75, May 2005.

[20] Raspberry Pi Foundation, E. Upton, R. Mullins, J. Lang, and A. Mycroft. Raspberry Pi FAQs. *http://www.raspberrypi.org/faqs*, 2013. Accessed Sept. 2014.

[21] Yingfang Fu, Jingsha He, Rong Wang, and Guorui Li. Mutual authentication in wireless mesh networks. In *IEEE International Conference on Communications*, pages 1690–1694, Beijing, China, May 2008.

[22] A. Hava, G.-M. Muntean, and J. Murphy. ABI: A mechanism for increasing video delivery quality in multi-radio Wireless Mesh Networks. In *2014 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–6, Beijing, China, June 2014.

[23] Guy Helmer, Johnny S. K. Wong, Vasant G. Honavar, Les Miller, and Yanxin Wang. Lightweight Agents for Intrusion Detection. *Journal of Systems and Software*, 67(2):109–122, 2003.

[24] Rich Henders and Bill Opdyke. Detecting intruders on a campus network: Might the threat be coming from within? In *Proceedings of the 33rd Annual ACM SIGUCCS Fall Conference*, SIGUCCS '05, pages 113–117, Monterey, CA, Nov 2005.

[25] M. Van Horenbeeck. Deception on the network: Thinking differently about covert channels. In *Proceedings of 7th Australian Information Warfare and Security Conference*, pages 174–184, Perth, Australia, Dec 2006.

[26] Fabian Hugelshofer, Paul Smith, David Hutchison, and Nicholas J.P. Race. Openlids: a lightweight intrusion detection system for wireless mesh networks. In *ACM MobiCom '09: Proceedings of the 15th Annual International Conference on Mobile Computing and Networking*, pages 309–320, Beijing, China, Sept 2009.

[27] Gianluca Iannaccone. CoMo: An Open Infrastructure for Network Monitoring Research Agenda. *http://como.sourceforge.net/*, 2008. Accessed Sept. 2013.

[28] M. Imani, B. Hassanabadi, and M. Naderi. Refinement of AODV routing algorithm for wireless mesh networks (WMNs). In *2011 19th Iranian Conference on Electrical Engineering (ICEE)*, pages 1–1, Tehran, Iran, May 2011.

[29] insecure.org. Nmap security scanner. *http://insecure.org/*, 2008. Accessed Sept 2014.

[30] Texas Instruments. Msp430 launchpad value line development kit. *http://www.ti.com/tool/msp-exp430g2*, 2013. Accessed Jan. 2014.

[31] Intel. Intel core i3-2130 processor. *http://ark.intel.com/products/53428/Intel-Core-i3-2130-Processor-3M-Cache-3-40-GHz*, August 2013. Accessed Feb 2015.

[32] V.M. Ionescu, F. Smaranda, and A.-V. Diaconu. Control system for video advertising based on Raspberry Pi. In *Networking in Education and Research, 2013 RoEduNet International Conference 12th Edition*, pages 1–4, Constanta, Romania, Sept 2013.

[33] Jonathan Ishmael, Sara Bury, Dimetrios Pezaros, and Nicholas Race. Deploying rural community wireless mesh networks. *IEEE Internet Computing*, 12(4):22–29, July-Aug 2008.

[34] Muhammad Mahmudul Islam, Ronald Pose, and Carlo Kopp. An intrusion detection system for suburban ad-hoc networks. In *TENCON 2005 IEEE Region 10*, pages 1–6, Melbourne, Australia, Nov 2005.

[35] David Lloyd Johnson and Kobus Roux. Building rural wireless networks: Lessons learnt and future directions. In *Proceedings of the 2008 ACM Workshop on Wireless Networks and Systems for Developing Regions*, pages 17–22, San Francisco, CA, Sept 2008.

[36] O. Kachirski and R. Guha. Intrusion detection using mobile agents in wireless ad hoc networks. In *Proceedings of the IEEE Workshop on Knowledge Media Networking*, pages 153 – 8, Kyoto, Japan, 2002.

[37] Guillaume Kaddouch. Raspberry Pi firewall and instrusion detection system. *http://www.instructables.com/id/Raspberry-Pi-Firewall-and-Intrusion-Detection-Syst/*, 2013. Accessed Dec. 2014.

[38] Minkyong Kim, Jeff Fielding, and David Kotz. Risks of using AP locations discovered through war driving. In *Proceedings of the 4th International Conference on Pervasive Computing*, Dublin, Ireland, May 2006.

[39] Wenke Lee and Salvatore J. Stolfo. A framework for constructing features and models for intrusion detection systems. *ACM Transactions on Information and System Security*, 3(4):227–261, Nov 2000.

[40] Celia Li and Uyen Trang Nguyen. Fast authentication for mobile clients in wireless mesh networks. In *2010 23rd Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–8, Calgary, AB, Canada, May 2010.

[41] Lv Li, Yang Jiankang, and Huo Jinghe. Research on the security issues and counter measures of wireless mesh network. In *2012 8th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*, pages 1–3, Shanghai, China, Sept 2012.

[42] Yonggang Liu, Ya Liu, and Xiaohui Wang. Mid-year 2014 DDoS threat report. *www.nsfocus.com/SecurityReport/NSFOCUS%202014%20Mid-Year%20DDoS%20Threat%20Report.pdf*, Sept. 2014. Accessed Dec 2014.

[43] Dwight Makaroff, Paul Smith, Nicholas J.P. Race, and David Hutchison. Intrusion detection systems for community wireless mesh networks. In *5th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2008*, pages 610–616, Atlanta, GA, Sept-Oct 2008.

[44] Z. Morley Mao, Vyas Sekar, Oliver Spatscheck, Jacobus van der Merwe, and Rangarajan Vasudevan. Analyzing large DDoS attacks using multiple data sources. In *Proceedings of the 2006 SIGCOMM Workshop on Large-scale Attack Defense*, LSAD '06, pages 161–168, Pisa, Italy, Sept 2006.

[45] Gerald A. Marin. Network security basics. *IEEE Security & Privacy*, 3(6):68–72, Nov 2005.

[46] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 255–265, Boston, MA, Aug 2000.

[47] Bo Meng and QianXing Xiong. A securely fair non-repudiation protocol with TTP load lightly. In *The 8th International Conference on Computer Supported Cooperative Work in Design*, pages 13–17, Chicago, IL, May 2004.

[48] Microsoft. Microsoft network monitor 3.4. *http://www.microsoft.com/en-us/download/details.aspx?id=4865*, October 2013. Accessed Sept 2014.

[49] A. Mishra, K. Nadkarni, and A. Patcha. Intrusion detection in wireless ad hoc networks. *IEEE Wireless Communications*, 11(1):48 – 60, Feb 2004.

[50] MIT. Kerberos: The network authentication protocol. *http://web.mit.edu/kerberos/*, 2010. Accessed Dec 2012.

[51] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy*, 1(4):33–39, July 2003.

[52] D.S. Niculescu, S. Ganguly, Kyungtae Kim, and R. Izmailov. Performance of VoIP in a 802.11 wireless mesh network. In *25th IEEE International Conference on Computer Communications.*, pages 1–11, April 2006.

[53] Brendan O'Connor. CreepyDOL (Creepy Distributed Object Locator). *http://blog.ussjoin.com/2013/08/creepydol.html*, 2013. Accessed May 2014.

[54] The University of Chicago and Argonne National Laboratory. Globus alliance. *http://www.globus.org/*, 2008.

[55] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[56] Alison Powell and Leslie Regan Shade. *Community and municipal WiFi initiatives in Canada: evolutions in community participation.*, pages 183–201. Athabasca University Press, Edmonton, AB, Canada, 2012.

[57] Martin Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX Conference on System Administration*, pages 229–238, Seattle, WA, Nov 1999. USENIX Association.

[58] Arduino SA. What is Arduino? *http://www.arduino.cc/en/Guide/Introduction*, 2013. Accessed Jan. 2013.

[59] E. Sedoyeka, Z. Hunaiti, M. AL Nabhan, and W. Balachandran. WiMAX mesh networks for underserved areas. In *IEEE/ACS International Conference on Computer Systems and Applications*, pages 1070–1075, Doha, Qatar, March 2008.

[60] Muhammad Shoaib Siddiqui and Choong Seon v. Security issues in wireless mesh networks. In *MUE '07: Proceedings of the 2007 International Conference on Multimedia and Ubiquitous Engineering*, pages 717–722, Seoul, Korea, April 2007.

[61] Ed Skoudis and Lenny Zeltser. *Malware: Fighting Malicious Code*. Prentice Hall PTR, Upper Saddle River, NJ, 2003.

[62] R. Sombrutzki, A. Zubow, M. Kurth, and J.-P. Redlich. Self-organization in community mesh networks the Berlin roofnet. In *1st Workshop on Operator-Assisted (Wireless Mesh) Community Networks*, pages 1–11, Berlin, Germany, Sept 2006.

[63] Vikram Srinivasan, Pavan Nuggehalli, Carla F. Chiasserini, and Ramesh R. Rao. Cooperation in wireless ad hoc networks. In *Proceedings of IEEE Infocom*, pages 808–817, San Francisco, CA, Mar 2003.

[64] Brett Stone-Gross, Christo Wilson, Kevin Almeroth, Elizabeth Belding, Heather Zheng, and Konstantina Papagiannaki. Malware in IEEE 802.11 wireless networks. In *Passive and Active Measurement Conference (PAM)*, pages 222–231, Cleveland, OH, April 2008.

[65] Sonesh Surana, Rabin Patra, and Eric Brewer. Simplifying Fault Diagnosis in Locally Managed Rural WiFi Networks. In *Proceedings of the 2007 Workshop on Networked Systems for Developing Regions*, pages 1–6, Kyoto, Japan, August 2007.

[66] Sonesh Surana, Rabin Patra, Sergiu Nedevschi, Manuel Ramos, Lakshminarayanan Subramanian, Yahel Ben-David, and Eric Brewer. Beyond Pilots: Keeping Rural Wireless Networks Alive. In *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, pages 119–132, San Francisco, CA, Apr 2008.

[67] Bit Tech. Intel core i3 3220 power consumption. *http://www.bit-tech.net/hardware/2012/11/26/intel-core-i3-3320-review/7*, November 2012. Accessed Sept 2014.

[68] Aaron Turner. TCP-Replay. *http://tcpreplay.synfin.net/*, December 2013. Accessed June 2014.

[69] A Veldurthy, D Goel, H.D Mary, and P Leon. Network traffic analysis of the internet 2 using OSU flow-tools. *https://www.andrew.cmu.edu/user/pgl/I2NTA.pdf*, July 2007. Accessed Oct 2014.

[70] Michael Walfish, Mythili Vutukuru, Hari Balakrishnan, David Karger, and Scott Shenker. DDoS defense by offense. *Computer Communication Review*, 36(4):303, 2006.

[71] Jian Wang, Nan Jiang, Hui Li, Xinxin Niu, and Yixian Yang. A simple authentication and key distribution protocol in wireless mobile networks. In *International Conference on Wireless Communications, Networking and Mobile Computing*, pages 2282–2285, White Plains NY, Sept 2007.

[72] Nicholas Weaver, Vern Paxson, Stuart Staniford, and Robert Cunningham. A Taxonomy of Computer Worms. In *Proceedings of the 2003 ACM Workshop on Rapid Malcode*, pages 11–18, Washington, DC, Oct 2003.

[73] T. Wu, Yuan Xue, and Yi Chi. Preserving traffic privacy in wireless mesh networks. In *Proceedings of International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 459–461, Niagara Falls, NY, June 2006.

[74] Yang Xiao, Xuemin Shen, and Ding-Zhu Du. *Wireless Network Security (Signals and Communication Technology)*. Springer-Verlag New York, Inc., Secaucus, NJ, 2007.

[75] Peng Xue and Surendar Chandra. Revisiting multimedia streaming in mobile ad hoc networks. In *Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 1–7, Newport, Rhode Island, May 2006.

[76] Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang. Security in mobile ad hoc networks: challenges and solutions. In *IEEE Wireless Communications*, volume 11, pages 38–47, New York, NY, Feb 2004.

[77] Yatao Yang and B. Stiller. A non-repudiation scheme for user postings based on BAC in mesh networks. In *2012 IEEE Network Operations and Management Symposium (NOMS)*, pages 1187–1190, Maui, HI, April 2012.

[78] Ping Yi, Yichuan Jiang, Yiping Zhong, and Shiyong Zhang. Distributed intrusion detection for mobile ad hoc networks. In *Proceedings of the 2005 Symposium on Applications and the Internet Workshops*, pages 94–97, Trento, Italy, Jan 2005.

[79] Yongguang Zhang and Wenke Lee. Intrusion detection in wireless ad-hoc networks. In *Proceedings of the Sixth Annual International Conference on Mobile Computing and Networking*, pages 275 – 83, Boston, MA, Sept-Oct 2000.

Appendix A

Snort Logs

Listing A.1: Results of the RPi Packet Capture from SNORT with YouTube Video

```
===================================================
Run time for packet processing was 779.167510 seconds
Snort processed 63270 packets.
Snort ran for 0 days 0 hours 12 minutes 59 seconds
   Pkts/min: 5272
   Pkts/sec: 81
===================================================
Packet I/O Totals:
   Received: 63270
   Analyzed: 63270 (100.000%)
    Dropped: 0 ( 0.000%)
   Filtered: 0 ( 0.000%)
Outstanding: 0 ( 0.000%)
   Injected: 0
===================================================
Breakdown by protocol (includes rebuilt packets):
       Eth: 63286 (100.000%)
      VLAN: 0 ( 0.000%)
       IP4: 62223 ( 98.320%)
      Frag: 0 ( 0.000%)
      ICMP: 0 ( 0.000%)
       UDP: 931 ( 1.471%)
       TCP: 61292 ( 96.849%)
       IP6: 623 ( 0.984%)
    IP6 Ext: 665 ( 1.051%)
   IP6 Opts: 42 ( 0.066%)
      Frag6: 0 ( 0.000%)
      ICMP6: 77 ( 0.122%)
       UDP6: 546 ( 0.863%)
       TCP6: 0 ( 0.000%)
     Teredo: 0 ( 0.000%)
    ICMP-IP: 0 ( 0.000%)
      EAPOL: 0 ( 0.000%)
        GRE: 0 ( 0.000%)
       MPLS: 0 ( 0.000%)
        ARP: 440 ( 0.695%)
        IPX: 0 ( 0.000%)
All Discard: 0 ( 0.000%)
      Other: 0 ( 0.000%)
Bad Chk Sum: 0 ( 0.000%)
    Bad TTL: 0 ( 0.000%)
     S5 G 1: 16 ( 0.025%)
     S5 G 2: 0 ( 0.000%)
      Total: 63286
===================================================
Action Stats:
     Alerts: 10 ( 0.016%)
     Logged: 10 ( 0.016%)
     Passed: 0 ( 0.000%)
Limits:
```

```
      Match: 0
      Queue: 0
        Log: 0
      Event: 0
      Alert: 10
Verdicts:
      Allow: 61357 ( 96.976%)
      Block: 0 ( 0.000%)
    Replace: 0 ( 0.000%)
  Whitelist: 1913 ( 3.024%)
  Blacklist: 0 ( 0.000%)
     Ignore: 0 ( 0.000%)
==================================================
```

Listing A.2: Results of the RPi Packet Capture from SNORT with Skype Call

```
==================================================
Run time for packet processing was 3655.536045 seconds
Snort processed 148273 packets.
Snort ran for 0 days 1 hours 0 minutes 55 seconds
    Pkts/hr: 148273
   Pkts/min: 2471
   Pkts/sec: 40
==================================================
Packet I/O Totals:
   Received: 148273
   Analyzed: 148273 (100.000%)
    Dropped: 0 ( 0.000%)
   Filtered: 0 ( 0.000%)
Outstanding: 0 ( 0.000%)
   Injected: 0
==========================================================
Breakdown by protocol (includes rebuilt packets):
       Eth: 148273 (100.000%)
      VLAN: 0 ( 0.000%)
       IP4: 141816 ( 95.645%)
      Frag: 0 ( 0.000%)
      ICMP: 1128 ( 0.761%)
       UDP: 4799 ( 3.237%)
       TCP: 135889 ( 91.648%)
       IP6: 4268 ( 2.878%)
    IP6 Ext: 4383 ( 2.956%)
   IP6 Opts: 115 ( 0.078%)
      Frag6: 0 ( 0.000%)
     ICMP6: 1931 ( 1.302%)
      UDP6: 2337 ( 1.576%)
      TCP6: 0 ( 0.000%)
    Teredo: 0 ( 0.000%)
    ICMP-IP: 0 ( 0.000%)
      EAPOL: 0 ( 0.000%)
    IP4/IP4: 0 ( 0.000%)
    IP4/IP6: 0 ( 0.000%)
    IP6/IP4: 0 ( 0.000%)
    IP6/IP6: 0 ( 0.000%)
        GRE: 0 ( 0.000%)
    GRE Eth: 0 ( 0.000%)
   GRE VLAN: 0 ( 0.000%)
    GRE IP4: 0 ( 0.000%)
```

```
    GRE IP6: 0 ( 0.000%)
GRE IP6 Ext: 0 ( 0.000%)
   GRE PPTP: 0 ( 0.000%)
    GRE ARP: 0 ( 0.000%)
    GRE IPX: 0 ( 0.000%)
   GRE Loop: 0 ( 0.000%)
       MPLS: 0 ( 0.000%)
        ARP: 2187 ( 1.475%)
        IPX: 0 ( 0.000%)
   Eth Loop: 0 ( 0.000%)
   Eth Disc: 0 ( 0.000%)
   IP4 Disc: 0 ( 0.000%)
   IP6 Disc: 0 ( 0.000%)
   TCP Disc: 0 ( 0.000%)
   UDP Disc: 0 ( 0.000%)
  ICMP Disc: 0 ( 0.000%)
All Discard: 0 ( 0.000%)
      Other: 2 ( 0.001%)
Bad Chk Sum: 270 ( 0.182%)
    Bad TTL: 0 ( 0.000%)
     S5 G 1: 0 ( 0.000%)
     S5 G 2: 0 ( 0.000%)
      Total: 148273
===================================================
Action Stats:
     Alerts: 1149 ( 0.775%)
     Logged: 1149 ( 0.775%)
     Passed: 0 ( 0.000%)
Limits:
      Match: 0
      Queue: 0
        Log: 0
      Event: 0
      Alert: 0
Verdicts:
      Allow: 148273 (100.000%)
      Block: 0 ( 0.000%)
    Replace: 0 ( 0.000%)
  Whitelist: 0 ( 0.000%)
  Blacklist: 0 ( 0.000%)
     Ignore: 0 ( 0.000%)
===================================================
Frag3 statistics:
       Total Fragments: 0
     Frags Reassembled: 0
              Discards: 0
         Memory Faults: 0
              Timeouts: 0
              Overlaps: 0
             Anomalies: 0
                Alerts: 0
                 Drops: 0
    FragTrackers Added: 0
   FragTrackers Dumped: 0
FragTrackers Auto Freed: 0
    Frag Nodes Inserted: 0
     Frag Nodes Deleted: 0
===================================================
Stream5 statistics:
```

```
           Total sessions: 67848
             TCP sessions: 67747
             UDP sessions: 101
            ICMP sessions: 0
              IP sessions: 0
               TCP Prunes: 0
               UDP Prunes: 0
              ICMP Prunes: 0
                IP Prunes: 0
TCP StreamTrackers Created: 67755
TCP StreamTrackers Deleted: 67755
             TCP Timeouts: 0
             TCP Overlaps: 0
      TCP Segments Queued: 0
     TCP Segments Released: 0
       TCP Rebuilt Packets: 0
         TCP Segments Used: 0
              TCP Discards: 106
                  TCP Gaps: 0
     UDP Sessions Created: 101
     UDP Sessions Deleted: 101
             UDP Timeouts: 0
             UDP Discards: 0
                   Events: 0
          Internal Events: 0
          TCP Port Filter
                 Dropped: 0
                Inspected: 0
                  Tracked: 135718
          UDP Port Filter
                 Dropped: 0
                Inspected: 6936
                  Tracked: 101
===================================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods: 0
    GET methods: 0
    HTTP Request Headers extracted: 0
    HTTP Request Cookies extracted: 0
    Post parameters extracted: 0
    HTTP response Headers extracted: 0
    HTTP Response Cookies extracted: 0
    Unicode: 0
    Double unicode: 0
    Non-ASCII representable: 0
    Directory traversals: 0
    Extra slashes ("//"): 0
    Self-referencing paths ("./"): 0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed: n/a
    Gzip Decompressed Data Processed: n/a
    Total packets processed: 16
===================================================
SMTP Preprocessor Statistics
  Total sessions : 0
  Max concurrent sessions : 0
===================================================
dcerpc2 Preprocessor Statistics
  Total sessions: 0
```

```
==================================================
SIP Preprocessor Statistics
  Total sessions: 0
==================================================
```

Listing A.3: Results of the RPi Packet Capture from SNORT with BitTorrent

```
==================================================
Run time for packet processing was 1907.796110 seconds
Snort processed 589975 packets.
Snort ran for 0 days 0 hours 31 minutes 47 seconds
   Pkts/min: 19031
   Pkts/sec: 309
==================================================
Packet I/O Totals:
   Received: 589975
   Analyzed: 589975 (100.000%)
    Dropped: 0 ( 0.000%)
   Filtered: 0 ( 0.000%)
Outstanding: 0 ( 0.000%)
   Injected: 0
==================================================
Breakdown by protocol (includes rebuilt packets):
       Eth: 589977 (100.000%)
      VLAN: 0 ( 0.000%)
       IP4: 587359 ( 99.556%)
      Frag: 0 ( 0.000%)
      ICMP: 390 ( 0.066%)
       UDP: 525214 ( 89.023%)
       TCP: 61753 ( 10.467%)
       IP6: 1463 ( 0.248%)
    IP6 Ext: 1527 ( 0.259%)
   IP6 Opts: 64 ( 0.011%)
      Frag6: 0 ( 0.000%)
      ICMP6: 158 ( 0.027%)
       UDP6: 1305 ( 0.221%)
       TCP6: 0 ( 0.000%)
        ARP: 1155 ( 0.196%)
All Discard: 0 ( 0.000%)
      Other: 2 ( 0.000%)
Bad Chk Sum: 7 ( 0.001%)
    Bad TTL: 0 ( 0.000%)
     S5 G 1: 0 ( 0.000%)
     S5 G 2: 2 ( 0.000%)
      Total: 589977
==================================================
Action Stats:
     Alerts: 396 ( 0.067%)
     Logged: 396 ( 0.067%)
     Passed: 0 ( 0.000%)
Limits:
      Match: 0
      Queue: 0
        Log: 0
      Event: 0
      Alert: 0
Verdicts:
      Allow: 589959 ( 99.997%)
      Block: 0 ( 0.000%)
```

```
    Replace: 0 ( 0.000%)
  Whitelist: 16 ( 0.003%)
  Blacklist: 0 ( 0.000%)
     Ignore: 0 ( 0.000%)
===============================================
Stream5 statistics:
           Total sessions: 2286
             TCP sessions: 2267
             UDP sessions: 19
            ICMP sessions: 0
              IP sessions: 0
               TCP Prunes: 0
               UDP Prunes: 0
              ICMP Prunes: 0
                IP Prunes: 0
TCP StreamTrackers Created: 2344
TCP StreamTrackers Deleted: 2344
             TCP Timeouts: 0
             TCP Overlaps: 0
       TCP Segments Queued: 381
     TCP Segments Released: 381
       TCP Rebuilt Packets: 50
         TCP Segments Used: 54
             TCP Discards: 1306
                 TCP Gaps: 0
      UDP Sessions Created: 19
      UDP Sessions Deleted: 19
             UDP Timeouts: 0
             UDP Discards: 0
                   Events: 14
          Internal Events: 0
          TCP Port Filter
                 Dropped: 0
               Inspected: 0
                 Tracked: 61751
          UDP Port Filter
                 Dropped: 0
               Inspected: 526463
                 Tracked: 19
===============================================
HTTP Inspect - encodings (Note: stream-reassembled packets included):
    POST methods: 0
    GET methods: 20
    HTTP Request Headers extracted: 20
    HTTP Request Cookies extracted: 0
    Post parameters extracted: 0
    HTTP response Headers extracted: 19
    HTTP Response Cookies extracted: 0
    Unicode: 0
    Double unicode: 0
    Non-ASCII representable: 0
    Directory traversals: 0
    Extra slashes ("//"): 0
    Self-referencing paths ("./"): 0
    HTTP Response Gzip packets extracted: 0
    Gzip Compressed Data Processed: n/a
    Gzip Decompressed Data Processed: n/a
    Total packets processed: 31713
```