DEVELOPING A LOW INTERACTION HONEYPOT DETECTION SYSTEM USING LIVE ENVIRONMENT AND NETWORK ANALYSIS.

by

ALLAN MUHUMUZA MUTABAZI
Reg. No: 2012/HD05/871U
BSc IT (Mak)
Department of Computer Science
School of Computing and Informatics Technology, Makerere University
allanmuh@yahoo.com/+256782172570

A Project Report Submitted to the College of Computing and Informatics Technology for the Study Leading to a Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science in Computer Science of Makerere University

Supervisor
Department of Computer Science
School of Computing and Informatics Technology, Makerere University
Dr. Joyce Nakatumba-Nabende jnakatumba@cis.mak.ac.ug
+256-701-726-338, Fax:+256-41-540620

August 2018

Contents

List of Abbreviations

CIM	Common Information Model
WMI	Windows Management Infrastructure

Declaration of Authorship

I, Muhumuza Allan Mutabazi, declare that this document titled, Developing a Low Interaction Honeypot Detection System Using Live Environment and Network Analysis, and the work presented therein are my own and have never been submitted at any academic institution for an award of any form.

Signature....[illegible]..

Date....07-12-2018..

Supervisor Approval

I confirm that the work presented in this document, A Honeypot Detection System Using Live Environment and Network Analysis is authentic and has been accomplished under my supervision.

Signature.[illegible]..

Date.....07-December-2018...

Abstract

As the technological change leads to the use of information systems to store and process data, the need to protect the systems becomes very important as it can cause to data leakage, disruption in processing among other things which then lead to financial losses and endangering of the persons whose information has been leaked. Honey Pots are one of the ways these information systems can be protected from authorized access through understanding the motives of the attacker and the methods being used to try access the systems. With this information collected, the production system can be hardened and information protected.

This project helped develop a Honeypot detection system that helps the honeypot developers check the built honeypots and harden them to prevent them from being detected by the attackers. The honeypot detection tool was tested on windows honeypots that are currently in use, a production server was used a control environment and the results showed that honeypots in production can be detected by the hackers.

Honeypots that are not easily detectable are important as they help organizations to collect lots of data on the methods the attackers are likely to use, and the information being targeted by the hackers. The loop holes in these areas can then be patched to keep the information systems safe.

The project also proposes a high level model on how honeypots for windows systems can be made undetectable and lead to the collection of more information about the attack, this model can also be replicated in the systems that are being operated by the organizations.

Acknowledgment

This thesis would not have been possible without the valuable contribution and guidance of certain people. First and foremost I offer my sincerest gratitude to my supervisor, Dr. Joyce Nakatumba Nabende, who has supported me throughout my thesis with her patience and knowledge. Secondly, I would like to express my gratitude to my employers during my study time, AH Consulting, EY Uganda and Uganda Revenue Authority for the knowledge/experience impacted during my time of study, this knowledge/experience was a valuable input to the course. I am also grateful to family for always supporting and encouraging me throughout the course duration, this would not be possible without you.

CHAPTER 1

BACKGROUND

This thesis is about developing system to detect low interaction honeypots in a networked environment through analysis of production environments and network analysis. The thesis uses a combination of system processing properties and network analysis of a computing environment, this will enable developers to strengthen the characteristics of low interaction honeypots during development in order to build honeypots that are not easily detected.

Today organizations are networked and hence rely on servers to provide computing resources and relay information to the different clients accessing the network. This centralized control provides gains such as access control of system resources, logging, patching, scalability and easy system maintenance. They also enable the organization system to all pick data from one source of truth and hence data integrity of the services they offer. These environments are very critical due to the value of information accessed and transmitted over the networks.

However these processing environment face risks such as downtime, information leakages, unauthorized access to the systems and encryption of data by hackers. This is mainly due to the networks being internet facing. These attacks are usually carried out through vulnerable networks that are not patched, open ports, insure services and gaining user access through social engineering. These systems are susceptible to being attacked due to the demand for information that is being transmitted and stored. According to the Global Cyber Report 2018, sub Saharan Africa lost about 2.5 billion dollars whereas the worldwide lost about 600billion dollars in cyber related crimes.

Risks that lead to network attacks can only be reduced but cannot be eliminated completely but are always defended in different forms. Attacks are being defended by in the following ways: - Patching of systems, deploying secure configurations, deploying intelligent firewalls to filter and block malicious traffic, honeypot deployment, and user education. However these methods of defense face challenges such as zero day malware, costs of implementations of the defense is very high, need to use configurations that can be exploited to deliver services on the network. The perfect decoy, they often containing false information, without providing access to any live data, however if poorly designed and detected by the hackers, it could act as a source of information about the production environment which could then facilitate attacks.

A honeypot is a system designed to learn how “black-hats” probe for and exploit weaknesses in IT systems. According to Anjari et al ,2011, it can be defined as an information system resource whose value lies in unauthorised or illicit use of that resource, and it is used as a decoy, put out on a network as bait to lure attackers [1]. One of the purposes of a honeypot is to lure the attacker into interacting with into the honeypot in order to gather information about emerging threats or attack vectors so that the organization’s defences can be updated. New tools can be discovered, attack patterns can be determined, and the very motives of the attackers can be noted and used to improve and protect the networked environment [2] [3].

Being able to detect honeypots is important to security professionals as it enables them to improve the honeypots developed in thus future making them undetectable. A honeypot is basically a system that emulates a weak or venerable system in an effort to try and attract a potential attacker to try and crack or break the machine while the honeypot is logging all the activities that occur [1].

This project led to the development of a honeypot detection system that will be used by honeypot developers to test whether honeypot systems developed can be detected. This will enable the developers' tweak and configure honeypot to make them undetectable.

1.2 Problem Statement

Honeypots that are detectable don't serve the purpose of luring the attackers to the systems and collecting as much information from them as regards to the information they need from the system, and the type of attack used. Several honeypot detection methods have been proposed by the several academicians, however the proposed methods cover either network analysis or system analysis, and we hence intend to implement both methods in a single system to be able to detect the honeypots.

The fact that low interaction honeypots do not implement a complete feature set (which a real system does) and also that emulated environments have a significant software overhead when multiple virtual machines are running on a single physical machine, we will use this to find honeypots by checking for real system [4].

The research developed a honeypot detection system that has tools that test the components that make honeypots discoverable, this will guide future honeypot developers' patch up these loopholes hence making honeypots more effective in gathering information from the attackers.

This research led to the development of a honeypot detection system that will enable identification of low interaction honeypot systems through identification of virtualization systems on which they are implemented and techniques such as port scanning, services review, network analysis service scanning.

1.3 Research Objectives

1.3.1 General Research Objective

The main objective of this research was to develop a system to enable detection of a honeypot in a network environment to help honeypot designers develop stronger honeypots.

1.3.2 Research specific objectives

The following were the research objectives of the study.

1. To study attack and defense mechanisms in a networked computing environment through review of literature
2. To develop and test a tool to detect honeypot systems in a network environment.

1.4 Scope

The honeypot detection system covered virtual honeypots that are currently in use and will support the the windows operating system.

The developed honeypot detection system was developed and tested on a windows environment due to the share of the windows production environment.

1.5 Significance of the Study

- The main significance of the research proposal was to study attack and defense mechanisms through the review of literature .The study will guide the developers of honeypots improve on the components that lead to the detection of honeypots in order to develop undetectable honeypots.

- The research also developed and tested a system to be used in detecting honeypots through the use network analysis and production analysis review. The implementation included different tools that led to the detection of ot based on their outputs.

- The research will help the developers of honeypots make improvements that can make honeypots detectable by the attackers. The honeypot detection tool helps honeypot developers test honeypots being developed for the identified vulnerabilities, hence leading to the development of honeypots that are undetectable or hard to detect by the attacker and hence collection of lots of information by the organisations that have deployed the honeypots.

CHAPTER 2

LITERATURE REVIEW

This chapter discusses the literature reviewed regarding the development of a honeypot detection system in a networked environment. It explains what honeypot systems are and how they are being put in use in networked environments, it also highlights the different techniques that have been used in honeypot detection by researchers previously and the structures on which the low interaction honeypots are developed. It contains three sections which include honeypots, virtual honeypots and honeypot detection;-

2.1 Honeypots

A honeypot [5] is an inveigler environment that is monitored strictly by the network safeguards. It is used to attract the network attacks by true or simulative network services. The purpose of using the honeypot is to protect the important object hosts. Honeypot Systems are decoy servers or systems setup to gather information regarding an attacker or intruder into your system, it is important to remember that honeypots do not replace other traditional Internet security systems; they are an additional level or system [6]. A honeypot works by fooling attackers into believing it is a legitimate system; they attack the system without knowing that they are being observed covertly. When an attacker attempts to compromise a honeypot, attack-related information, such as the IP address of the attacker and the method of attack, are collected. This activity done by the attacker provides valuable information and analysis on attacking techniques, allowing system administrators to “trace back” to the source of attack if required.

Honeypots are run to gather information about the motives and tactics of the Blackhat community targeting different networks [7]. These honeypots do not add direct value to a specific organization; instead, they are used to research the threats organizations face and to learn how to better protect against those threats.

The two popular reasons or goals behind setting up a honeypot is learning how intruders probe and attempt to gain access to your systems he adds that since a record of the intruder’s activities is kept, you can gain insight into attack methodologies to better protect your real production systems. The other reason as to gather forensic information required to aid in the apprehension or prosecution of intruders by law enforcement officials with the details needed to prosecute [6].

Figure 2.1 shows an example of a honeypot on in a networked environment and potential locations of honeypot systems which can be on the internal network, the DMZ or on the internet facing computing environment.

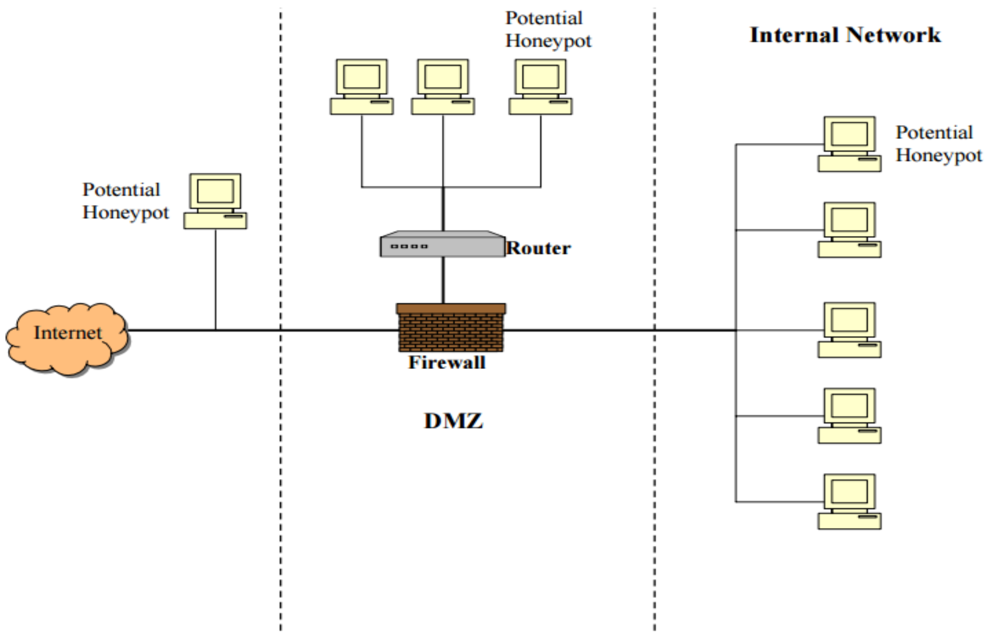


Figure 2.1. Structure of a honeypot in a networked environment adapted from SANS [6]

Production Honey Pots

Production honeypots emulate real production systems and have attackers spend time and resource attacking them as opposed to the production or critical systems and to learn the way they exploit vulnerabilities in production environment. Production honeypots mainly emulate specific services and sometimes operating systems to invite attackers. They can also emulate different backdoors, viruses and trojans to lure the attackers. For an example to examine attacks on web servers a production honeypot emulating the Web server and fake services can be deployed [1]. This research will focus on operating system based production honeypots, we will hence develop a system to find honeypots in production environments

The concept of production honeypots is to emulate real production systems and have attackers spend time and resource attacking them as opposed to the production or critical systems and to learn the way they exploit vulnerabilities in production environment [8]. Production honeypots mainly emulate specific services and sometimes operating systems to invite attackers. They can also emulate different backdoors, viruses and trojans to lure the attackers. It is easier to break the honeypot phases into groups and the Bruce Schneier model is good for understanding the honeypots [10]. He groups the security issues into several steps, which are prevention, detection and response.

Production honeypots add extensive value to the organization's detection capability they are designed for. Often organizations are so overwhelmed with production activity, they don't have time and resources to spend through gigabytes of system logs for detecting the attacks. Even if they happen to surf through all the logs, still it won't be sufficient for complete detection because the extensive logs generated by security technologies suffer by false positives and false negatives.

Production honeypots are designed in such a way that either there is no false positive or very few because all the activities on production honeypots is taken as illegitimate, hence all the logs are relevant, important and reveal some problem, attack or any attempt made for the same. They are also at par with the risk of false negatives, when IDS systems fail to detect a valid attack. It is possible to launch an unknown attack that may not be detected by other security technologies but honeypots addresses this issue very well because they always detect any connection made to them via a known or unknown way by the virtue of system activity, not signatures. A connection made to the production honeypot, is most likely a malicious activity like probe, scan or attack. If the honeypot initiates a connection, most likely means the system is successfully compromised. Thus, due to their elementary design they are best suited for detection. But they can never replace any technology for detection because they can't be placed on production systems.

The thesis seeks to support and reinforce these properties by developing a system that will enable the developers discover any loopholes that may exist in the implementation of low interaction honeypots which may then be fixed making honeypots detectable, below main advantages of production honeypots in a networked environment [9]:

- Production honeypots carry lots of tangible and intangible advantages for an organization. Especially they add some advantages, which no other existing security technology provides.
- Production honeypots collect small amount of information. Instead of logging 1 GB of data a day, they log only 1 MB of data. So it becomes much easier to analyze the data and derive value from it. They are designed to capture any tool, method or exploit which they have never seen before.
- Information collected by them is of high value and no other technology can match some of the collected information. The gathered data can be used to learn about the attack, existing vulnerabilities and the ways intruders use to probe and gain access to the systems. The gathered data can be provided as legal proofs in the apprehension and prosecution of intruders.
- They are conceptually smooth, so there are fewer chances of mistakes in configuring and deploying them. They aid in flexible data gathering and have lots of configurable options. They can log data locally, to a central log server, put an alarm at the time of intrusion, send an e-mail to intrusion response group and can make entry in the incident database.

2.2 Virtual Honeypots

In low interaction honeypots services are simulated in such a way that they cannot be exploited to gain complete access of the honeypot [10]. In these types of honeypots there is no operating system for the attackers to interact with [11]. The deployment and maintenance process of low interaction honeypots is comparatively simple then medium and high interaction honeypots. Their functionality is very similar to passive IDS as they do not have any interaction with the attackers. Virtual honeypots can minimize risk but at the same time their functionality is limited. However they can still be used for analyzing spammers and can also be used as active countermeasure against worms.

Virtual honeypots contrast with hardware-based honeypots, which are dedicated computers, networks or network segments designed to serve the same purpose. Virtual honeypots can be thought of as virtual machines (VMs) which may exist in multiple configurations on a single computer or appliance to emulate various systems and vulnerabilities [12]. Virtual honeypots are cheaper to deploy and more secure than hardware-based systems. In some cases, for example, real honeypots have been infiltrated by intruders who were able to use them to attack the corporate network. However, because a virtual honeypot is an emulator, it doesn't function exactly as a real system does and hackers may be able to pick up on cues that indicate the difference [12]. We will leverage on these positives of virtualization to create and test the virtual honeypots. As virtual machines are a major component in the deployment of low interaction honeypots here are the main benefits of using virtualization [14]:

- A virtualized information technology infrastructure will change the old way of disaster recovery by providing a fast, dependable and low budget disaster recovery plan through hardware independent, server consolidation and easy test scenarios. This will support the research proposal as it will enable us setup a test environment cheaply to support the honeypot detection toll that we will have developed.
- Testing a new software in an OS can cause problems and cause file-system damage. With virtualization software developers can easily test new software in a virtualized environment and, if any damage is caused to the system, it is possible to rollback the system to its original state without any problems.
- Software developers can easily test their products in different OSs with just a few clicks. Having all OSs up and running in one place is something which software developers can use to their advantage while saving time. During the testing most of the test environments were setup in on the virtual environments due to the easy of setup and will enable the testing of the honeypot detection tool.
- On most servers only one application can run because if an application crashes the whole system will crash and, if there are any other applications on that server, they will stop functioning as well. To solve that problem system administrators usually run each application individually on different servers to minimize system failure. However, with virtualization, multiple applications can run at once on the virtual server leading to the savings money and resources.

In conclusion, literature reviewed will guided us in understanding why developers preferred to build the honeypot on virtual machine and we'll hence look at the role of virtualization in the detection of honeypots. We will also check how these properties of honeypots can help us achieve our goal of honeypot detection in a networked environment.

2.3 Honeypot Detection

a) **Bot Infection**

Honeypots can be detected by through malware infection using bot programs, a computer is compromised and a bot program is installed, some bot programs will continuously try to infect other computers in the Internet [15]. In this case, a honeypot must modify or block the outgoing malicious traffic to prevent infecting others. Based on this liability constraint imposed on honeypot security professionals, a botmaster could let compromised computers send malicious infection traffic to her sensors. However they noted that this honeypot detection technique is difficult for honeypot defenders to deal with. Honeypot defenders cannot block or even modify the outgoing infection traffic. Without accurate binary code analysis, honeypot defenders will not be able to know which target IPs belong to the botmaster's sensors [15].

b) **TCP/IP Stack**

Honeypots can be detected by the reviewing the to find inconsistencies in in TCP/IP stack (remotely detectable), using Tools like hping can be used to detect incorrect TCP/IP stack emulations indicating the use of a low-interaction honeypot for example the results would be as follows:- Normal RH9: TTL=64, window=0, id=0, DF for a live operating system, RH9 on vmware installed operating system: TTL=64, window=0, id=0, DF and RH9 on honeyd virtual honeypot: TTL=64, window=1460, id=0, DF. This method works even better on Unix systems emulating Windows and vice versa: Normal Win2k SP4: TTL=128, window=0, id=+, DF and honeyd emulating Win2k SP4: TTL=64, window=1460, id=0, DF [16]. This is a strong honeypot detection technique as the emulated TCP/IP stack can be easily identified and can conclude that the honeypot has been detected.

c) **Network Analysis**

Network analysis can be used to detect honeypots. An ideal honeypot will mirror a real system exactly and is thus difficult to detect but unfortunately existing honeypot technology is far from ideal. In general there are several high level "features" that honeypots possess but real production systems do not [4]:

- There should be no network activity on the honeypot
- All interactions with the honeypot are logged extensively
- Bandwidth is often restricted to prevent a compromised honeypot from damaging other network
- Low interaction honeypots do not implement a full feature set
- Emulated environments have multiple virtual machines running on a single physical machine or have significant software overhead when compared to real systems.

However, network analysis maybe is hard to detect without long term monitoring of the honeypot's local network traffic. It is worth noting that the only way to detect an ideal or "pure" honeypot at the network level is to monitor local traffic and even then there is a danger for a high false positive rate [4].

Service exercising is used to detect a honeypot by testing or "exercise" the services it provides. Some environments (especially low interaction honeypots) do not implement a full feature set and by selecting uncommon features or operations we may be able to determine if we are working with a legitimate system or a part of the network defenses [4]. However, some low interaction honeypots may create services that emulate those that run in a production networked environment, these can be detected by getting the time stamps of the running services which are usually absent in the low interaction honeypots.

d) **Timing Analysis**

Timing analysis of ICMP ECHO requests is a detection system builds on a simple observation that most honeypot software responds slower to ICMP ECHO (ping) requests compared to non-emulated systems. In effect this doubles the delay from the operating system. Other delays could be introduced when multiple virtual machines are present on a single guest operating system and the guest operating system must route packets between several processes. This is one of the few features that distinguish virtual machines from real systems [4]. Since most of the honeypot systems are built on virtual systems, this can be used for the detection of the environment on which the operating systems are installed which can then lead to the detection of the honeypots. However timing analysis requires a lot of information to be sent in order to have accurate results.

e) **Finger Printing**

TCP/IP finger printing: active finger printing is used to collect the data for analysis. For each of the TCP/IP connection, 49 various quantitative and qualitative features were extracted [4].

Being able to detect honeypots is important to malicious users as well as security professionals. The stealthy-ness of a honeypot is an important factor to consider in an organization's overall security strategy but more importantly honeypot developers have few tools with which to test their products [18]
.

Physical device fingerprinting can also be used to identify virtual honeypots based on the skews of the devices' physical clocks. This approach can be used to determine whether different addresses correspond to virtual hosts on the same physical machine. However, that approach would not work if the hosts being fingerprinted do not provide timestamps (e.g., with TCP timestamp option disabled) [19].

f) **Pattern Recognition**

Virtual honeypots detection using the pattern recognition technique uses classifiers to classify an unknown pattern as belonging to one of several existing patterns, classes, this is done through comparisons of latency between the honeypot and the production environment. This framework suggests two phases to accomplish the task which included (a) off-line training and (b) on-line recognition as shown in figure 2.2 [20].

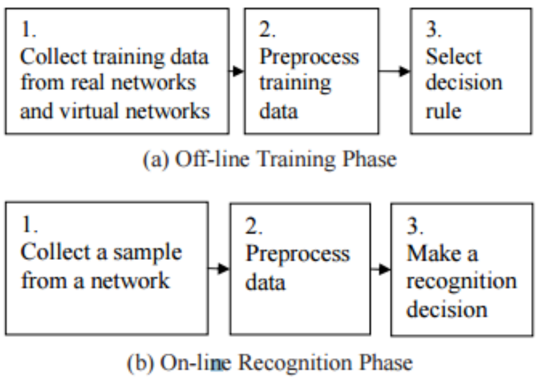

1. Collect training data from real networks and virtual networks
2. Preprocess training data
3. Select decision rule
(a) Off-line Training Phase
1. Collect a sample from a network
2. Preprocess data
3. Make a recognition decision
(b) On-line Recognition Phase

Figure 2.2 Virtual honeypots based on pattern recognition adapted from [20].

Phase 1 is the procedure for the off-line training phase [20]:

Collect training data from real networks and virtual networks: An attacker applies the probing traffic to known real networks and virtual networks. They collect the RTT data samples for both real links and virtual links.

Phase 2 is the procedure for the Preprocess training data

The attacker derives two classes of link latency data sample from the RTT data these include real link latency sample and virtual link latency sample. From these two classes of link latency sample, the attacker is able to derive two distributions: real link latency distribution, and virtual link latency distribution. They use an assumption that the attacker uses kernel based density estimation approach to derive the distribution numerically.

Phase 3 is the procedure for the selection decision rule

The attacker selects an appropriate classification rule based on the training data and the two trained link latency distributions. They use an assumption that the attacker uses a classifier based on the Neyman Pearson theory.

Figure 2.2 (b) is the procedure for the on-line recognition phase. The procedure is similar to the off-line training phase. An attacker collects a link latency sample from a suspect network. Then the attacker uses the trained classifier to decide whether the link is real or is a virtual link.

Honeypots can be detected by vividly simulating the routing topology and services of a virtual network by tailoring honeyd's response [10]. GenII honeynets allow a limited number of packets to be sent out from an infected honeynet. From the botmaster's perspective, some hardware or software specific means have always been available to detect infected honeypots [10].

2.4 Conclusion

The literature reviewed was important in helping us understand the works that have been done in the detection of low interaction honeypots in the IT industry. This knowledge helped us understand these methods which range from network analysis, measure of the packets being transmitted, network analysis, virtual machine detection and service scanning, we will use this knowledge to develop a comprehensive tool that can use some of the above methods and do a comparison to find out if the system being tested in a honey pot or not. We also noted the use of bots and spam were common methods in the detection of honeypots though spam would require user interaction and wouldn't be successful in honeypot detection unless the users click links to the honeypot or respond to mails.

The literature reviewed showed that most of the work previously done focuses on one component either network analysis or system properties, hence our honeypot detection system will focus on several components of live environment and network analysis hence covering both the system properties of the developed system and the transmission aspect of the environment. The detection of honeypots is very important for the future of honeypot development as it will enable the honeypot developers' test the developed products at one go which will lead to the development of honeypots that are undetectable and able to serve the purpose for which they will have been developed.

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter presents detailed discussion about the design process and the methodology used to develop the system for the detection of low interaction honeypots that was proposed for this research. It showed methods used to come up with tools to detect honeypots using live environment and network analysis. The methodology consisted of the four steps of the software development lifecycle which included requirement analysis, design, application development and final testing. Figure 3.0 showed the software development cycle which shows the processes that were be used in the implementation of the research proposal.

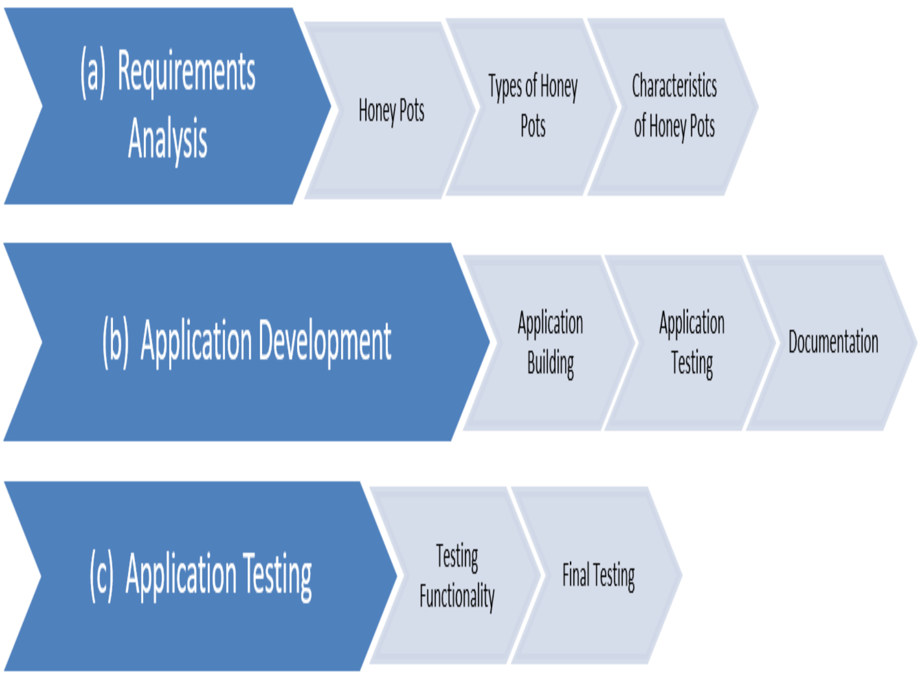


Figure 3.1: The software development lifecycle adapted from The Software Life Cycle Development (SDLC)

It described the different stages of lifecycle development that were applied during the development of the honey pot detection tool as further explained in the below:-

3.2 Design and Development of Software Application

The system was designed using PowerShell and Shell Scripts which are preinstalled on windows and will run on Linux Operating Systems through the PowerShell for Linux respectively, these were used to carry out analysis of live production environments and network analysis, Kali Linux a Debian-based Linux distribution aimed at advanced penetration testing and security auditing was used to deliver the developed tools to the operating systems that was tested, however the testing was done on the end user side and not on the delivery to the environment. Kali contains several hundred tools aimed at various information security tasks, such as Penetration Testing, Forensics and Reverse Engineering [23]. The development machine was a Windows 7 machine with VMare installed that hosted the Windows Development Environment which consisted of the PowerShell Integrated Scripting Environment (ISE) which was used to develop and test the windows based scripts. The honeypot detection tool consisted of several modules that enabled the detection of honeypots on a network such as network traffic analysis, open ports on the network, running services, running processes, virtual machine detection and hardware analysis.

PowerShell has grown in the last few years and is still rapidly becoming more popular. The development of PowerShell has greatly increased during these few years and the ability to include Linux bash inside it is the reason why it excels more than the other operating system languages [30]. PowerShell is a scripting language built on .NET framework, it is object-oriented language which utilizes cmdlets to perform various tasks. These cmdlets are small programs that can be called directly from the command line or from a PowerShell script file [31].

PowerShell is prebuilt on the Windows Operating Systems and is being used by system administrators to manage systems, it's being seen as a major tool for the future of systems administration. Hence we will use the PowerShell functionality to develop the honeypot detection components for the windows operating system. [31]

PowerShell is built on top of Microsoft's .NET Framework (Windows PowerShell versions 1 – 5) or .NET Core (PowerShell Core, version 6). All output returned by PowerShell commands consists of .NET Framework objects. A major part of PowerShell's capability comes from .NET Framework's object-oriented approach, it is PowerShell's main differentiating point compared to traditional shells. [24]

PowerShell provides a scripting language for creating more programs with various windows functionalities. The scripting language is similar to C# , the .NET programming language, supporting basic programming language features, designed specifically for a shell-based environment. Scripting allows combining multiple commands into singular parameterized scripts or functions, providing features such as conditional execution, looping, variables and arithmetic operations. [33]

Multiple sequentially executed commands can be placed into a function or script to make the functionality available by typing a single command. Functions and scripts can be distributed and executed from script files, which have a .ps1 file extension. Scripts can be executed by inputting a full file path to the .ps1 file or with the dot sourcing operator

"." (period). When the dot sourcing operator is used, the script is run in the current PowerShell session scope, making functions defined inside the script available until the session is closed. [35]

When a script file is executed, its code is run as-is in the current session. The difference between a script and a function is that functions can remain (with dot sourcing) available in the interactive session as commands.

Scripts and functions can define parameters to be used as input values similarly to cmdlets having parameters. Parameters are defined with the same syntax for functions and scripts. Parameter definitions can be placed in a Param() block. The definitions are similar to variable definitions, containing the variable name with the $ sign and optionally the object type in brackets. There are also additional options to control how the parameter is handled. [34]

PowerShell's capability can be extended by using modules. Modules enable developers and software vendors to easily integrate into PowerShell or extend its native capability by offering customized cmdlets. Related functionalities can be packaged as a module to be easily shared and installed. Modules consist of code, dependencies and manifest files. Modules can be divided into two types: script modules and binary modules. Script modules contain any valid PowerShell code. Binary modules contain compiled code written in a .NET programming language such as C# . Manifest files can optionally be used to store related metadata, such as versioning, dependencies and author information. [24]

PowerShell modules can contain scripts or cmdlet definitions, which are made available either by importing the module manually by using the Import-Module cmdlet or by using the autoloading feature, which imports modules placed into directories defined in the PSModulePath environment variable. Whenever a cmdlet in PSModulePath directory is called, PowerShell automatically loads the module into the session and executes the requested command. [24]

3.3 Application Development

Using the application development environment setup and configured we will wrote the algorithm to achieve the objective of being able to detect honey pots. Several tests were carried out at the unit levels to ensure that the project objective is being achieved, after the development of each module it was tested to find out if it achieves the desired role. The system comprises of different tools which are tested individually to make sure they can achieve the expectations these algorithms are as follows:-

The virtual machine detector checks/tests whether the system being tested is a virtual machine and which virtual machine has been deployed to host this computing environment.

The process start reader was developed to check the system and find out which processes are running and for how long they have been running. This will help the system come up with results to be reviewed in order to conclude on whether the machine being tested is a honeypot of not after combining it with other features.

The Logins algorithm was also tested to find out how many users has used the system and when they last accessed the system to help the tool users deduce on whether the system is a honeypot or a production environment.

We tested for hardware algorithm and network ports active to determine whether they are hosted on a virtual environment or server box.

Traffic analysis was also done to determine the amount of traffic flow versus the organization under review.

The port and services scanner was used to scan the given systems for open ports and running services on the given systems. Given that most honeypot have the default ports open, was be used to review the open port and was important in deducing on whether the given system is a honeypot or not.

The high level review of the honeypot detection tool described above is shown in Figure 3.2

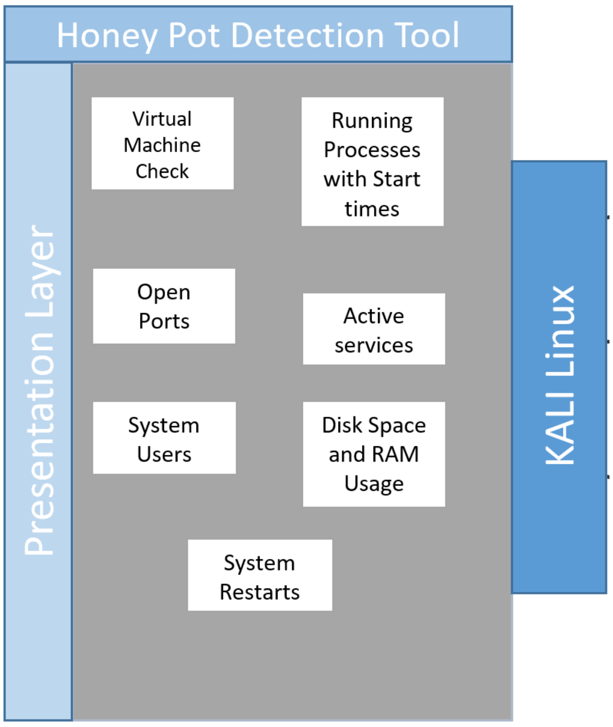


Figure 3.2: High-level architectural design for the honeypot detection tool.

All the functionalities described above are were then bundled to create the final system that will is being used to conclude on whether a system are honeypots or not from the results returned by the algorithms. The system provides different results that can be assessed to help conclude on whether the machine under review is a honeypot or a production system.

As part of the application development documentation is done, through the comments in the code being written and side by side documentation on the functionality of the program being written.

3.4 Application Testing

After the development of the application, several tests were carried out to find out, if it serves the purpose of its development using the test cases developed. Both functional and non-functional testing were carried out on the application. Functional testing included the testing functional requirements of a program and its components and also covered how well (if at all) the system executes its functions.

Nonfunctional testing was also done to specifically evaluate the readiness of a system according to the various criteria which are not covered by functional testing such as execution time of the program. The test were documented and added the to this report.

The honeypot system was tested in two phases that is individual functionality and integrated functionality. The algorithms developed was bundled to come up with one system with a presentation layer. The algorithms were tested for functionality before being integration into on system.

We developed a testing environment that included a Honey drive virtual machine hosting a Windows 7 virtual honeypot, Windows Production Environment and Windows Virtual Operating System as shown in figure 3.3

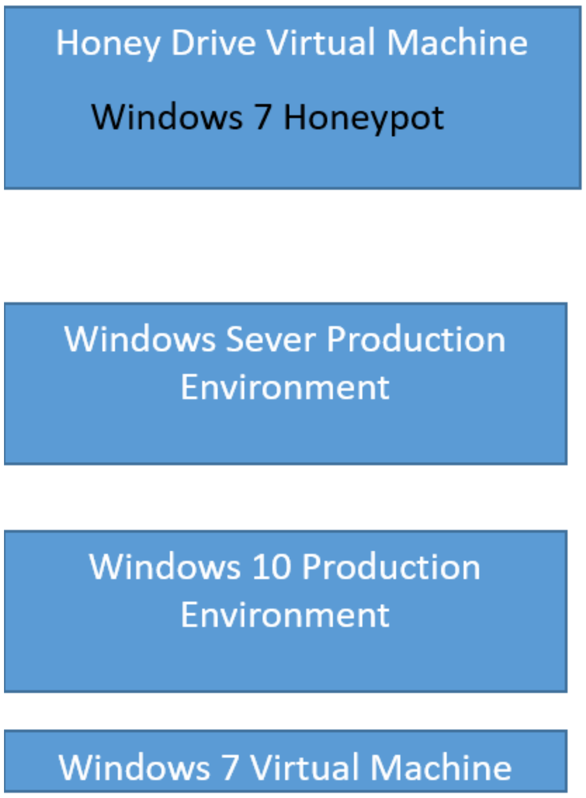

Honey Drive Virtual Machine
Windows 7 Honeypot
Windows Sever Production Environment
Windows 10 Production Environment
Windows 7 Virtual Machine

Figure 3.3: Testing environment for the honeypot detection tool.

The honeypot testing environment was setup as follows:

This research focused on developing a tool to discover production honeypots, hence we set up and low interaction honey pot using a honeypot linux distro called honey drive. Honey drive is a virtual appliance (OVA) with Xubuntu Desktop 12.04.4 LTS edition installed. It contains over 10 pre-installed and pre-configured honeypot software packages such as Kippo SSH honeypot, Dionaea and Amun malware honeypots, Honeyd low-interaction honeypot, Glastopf web honeypot and Wordpot, Conpot SCADA/ICS honeypot, Thug and PhoneyC honey clients and more. [27]

Additionally it includes many useful pre-configured scripts and utilities to analyze, visualize and process the data it can capture, such as Kippo-Graph, Honeyd-Viz, DionaeaFR, an ELK stack and much more. Lastly, almost 90 well-known malware analysis, forensics and network monitoring related tools are also present in the distribution [27].

This was installed as a guest operating system on windows 7 computer using the Oracle VM VirtualBox. VirtualBox is one of the most popular virtualization solutions for desktop computers [28]. Providing a virtualization solution to virtualize the X86 platform, this popular virtualization technology is now developed by Oracle Corporation. VirtualBox can run multiple guest operating systems under the host. Each of these hosts can pause and resume each guest at will, and is able to take snapshots of each of these guests for backup purposes [29]. Each of the virtual machines can be configured independently and can run in either software emulation mode or hardware assisted mode, taking use of Intels VT-X technology or AMD AMD-V technology. In addition VirtualBox emulates ethernet network adapters, which enables each guest to connect to the internet through a NAT interface.
Architecturally VirtualBox uses a kernel module that acts as the hypervisor, on top of which the VirtualBox API communicates with the hypervisor. From the API the end-user applications that is the default GUI and the VBoxManage command line tool can be built. The architecture that VirtualBox then builds upon is then not so different from what we have seen in other mentioned virtualization suites [29].

Virtual box is light and works well with honey drive, hence the choice of this of the virtualization software. This infrastructure will enable us exploit a one-to-many relationship between hardware and end-user respectively in order to obtain the economic benefits of large-scale resource sharing since it is open source [30]. The Honey Drive enabled us set put both the Linux and Windows honeypot on the same system virtual machine and operating systems for testing of the developed system for detecting honeypots.

The developed tool was then be used on the testing environment to come up with conclusions on the virtual honeypot and virtual machine to find out if the objective of the research proposal has been achieved. Figure 3.4 shows the high level testing architecture for the testing of the honeypot detection system developed.

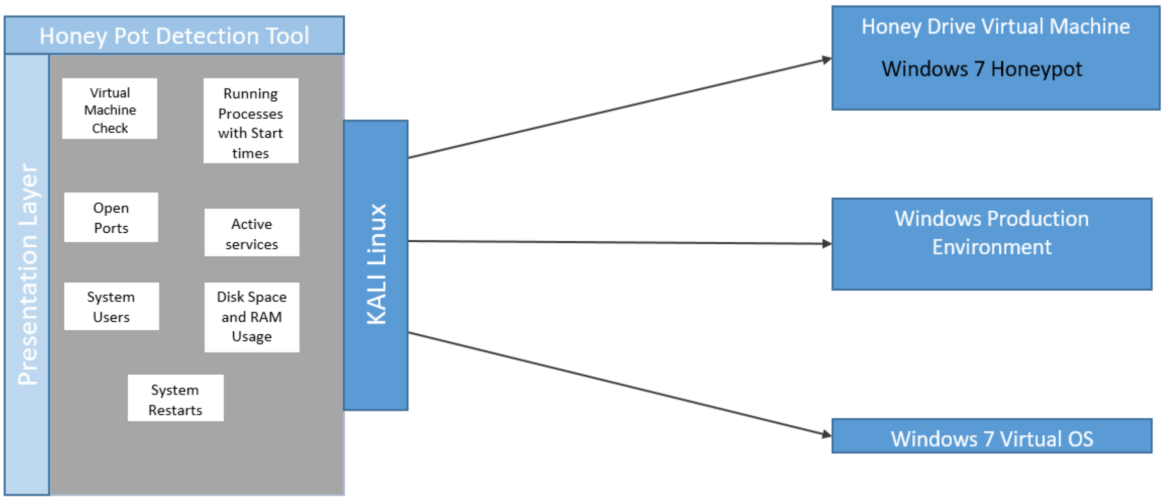


Figure 3.4: High-level architectural design for the honeypot detection tool testing.

CHAPTER 4:

HONEYPOT DETECTION ALGORITHM

An algorithm was designed to aid in the detection of honeypot systems, this chapter explains how the designs of the algorithm.

The honeypot detection algorithm was designed based on weaknesses in the current honeypot systems that makes them detectable in today's production environment, the algorithm focuses on areas whose results can help identify where a system is a honeypot or not, these include Virtual Machine Detection, Login Details, Status of the Ports, Screen Capture, Installed Programs, Processes and services, Logins, Hardware Properties and Active Network Connections.

The section below explains the implementation of the Honeypot Detection Algorithm breaking down the various components that make up the tool and the data flow.It is explains the functionality of the algorithm designed using a series of flowcharts and later illustrates the functionality and output of the different algorithms when run on a system.

1) Virtual Machine Detection

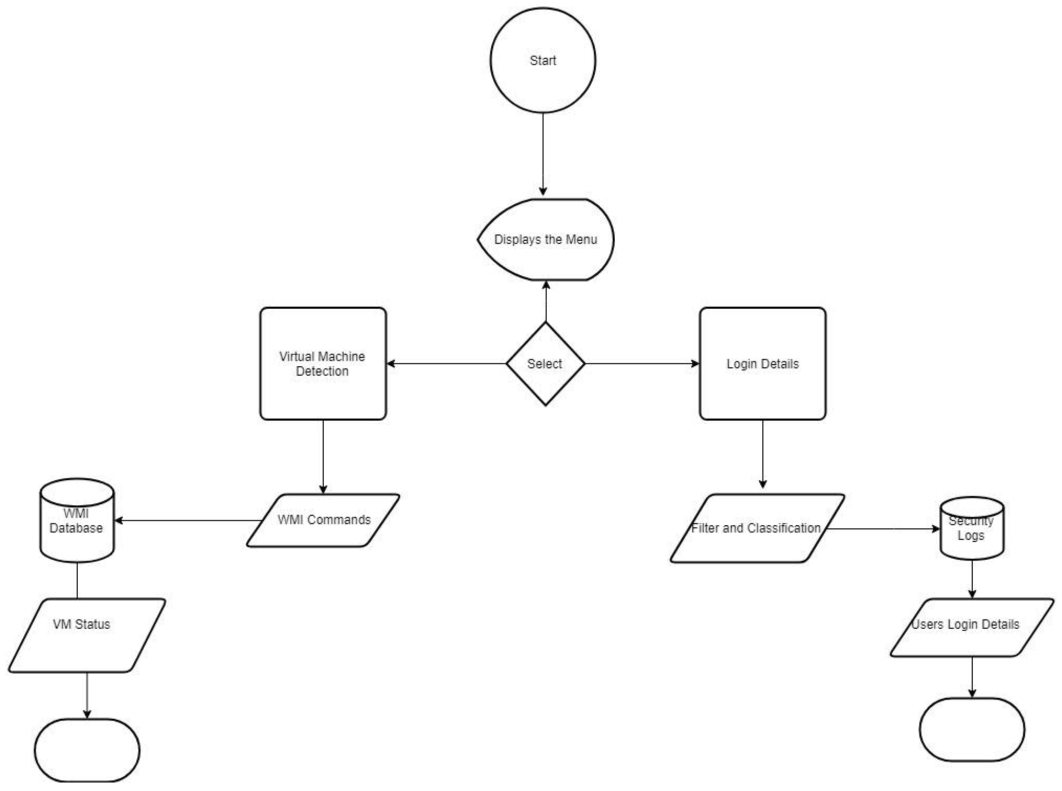


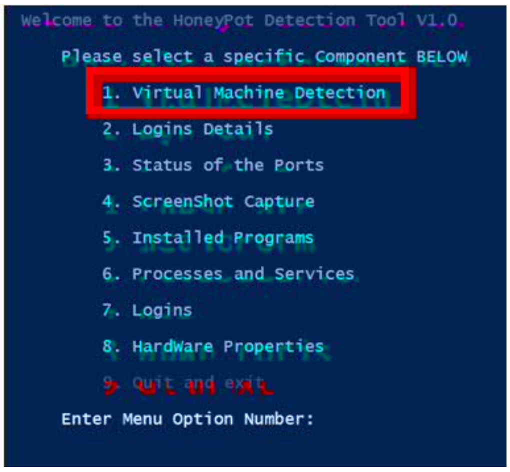
Figure 4.1 Flow Chart of the Virtual Machine Detection and Login Detail Algorithm

Virtual Machine detection algorithm was developed due to the fact that most of the honey pot in the production environment run on virtual environments and hence finding out if a machine is running on a virtual environment or not would be the first step in the detection of a honey pot machine. Hardware is expensive and hence most organizations are not able to spend resources in acquisition of hardware for honeypots and hence end up hosting them on Virtual Machines. Some virtual machine environment are free and if found as the host of the environment probability of the machine being a honeypot increases, these include Windows Virtual PC and Virtual Box.

As shown in figure 4.1, when launched the virtual machine tool uses Windows Management Instrumentation (WMI), which is a set of specifications from Microsoft for consolidating the management of devices and applications in a network from Windows computing systems, it then uses the WMI-Object property and filter out the properties which include the Manufacturer, Model, Computer Name, Current Time Zone, Primary Owner Name, Workgroup and the Domain. These are the results that are presented to the users before the program exits.

This make the virtual machine detection an important component of the honey pot detection system, it cannot determine whether the host is a honey pot or not and hence the other tools in the the honeypot system need to in the evaluation to come up with a correct conclusion.

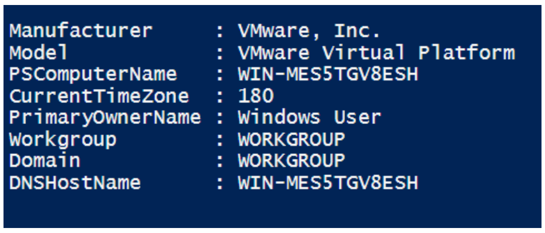
The figure below shows how the algorithm works when launched.



```
Welcome to the HoneyPot Detection Tool V1.0.
    Please select a specific Component BELOW
        1. Virtual Machine Detection
        2. Logins Details
        3. Status of the Ports
        4. ScreenShot Capture
        5. Installed Programs
        6. Processes and Services
        7. Logins
        8. HardWare Properties
        9. Quit and exit
    Enter Menu Option Number:
```

Figure 4.2 - Honey Pot Detection Tool

The tool displays the manufacturer and model of the operating system being probed, if hosted on a virtual environment, it displays the virtual environment software that the operating system is hosted on. Figure 4.3 below show the results from the honey pot detection tool.



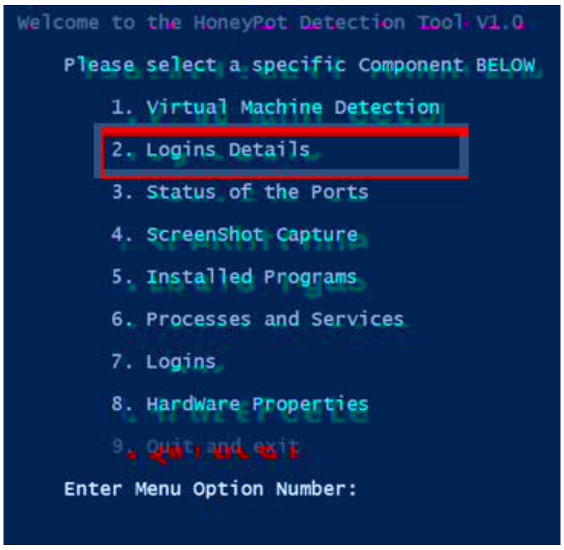
```
Manufacturer      : VMware, Inc.
Model             : VMware Virtual Platform
PSComputerName    : WIN-MES5TGV8ESH
CurrentTimeZone   : 180
PrimaryOwnerName  : Windows User
Workgroup         : WORKGROUP
Domain            : WORKGROUP
DNSHostName       : WIN-MES5TGV8ESH
```

Figure 4.3 – Results from the Honey Pot Detection Tool

2) Login Details

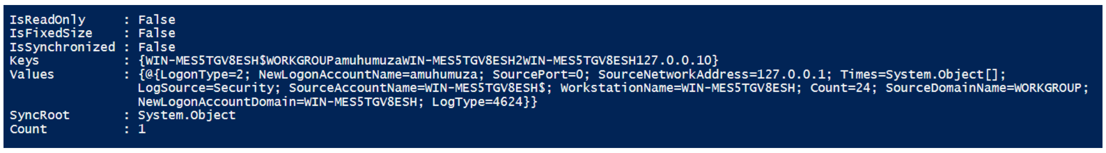
The Login Details algorithm enables display of information that regards the number of user's accounts that have logged in and the number of times operating system has been logged into, this information is extracted for the security log and filters out the logs with the label 4646 which is the record for logging into the operating system events as shown in figure 4.1. Production servers usually have very few logins as they are not used for daily use whereas the honey pots are likely to have been logged in several times by different users due to their ease of access.

Figure 4 below shows the display of the Login Details Algorithm launched from the honey pot detection system.



```
Welcome to the HoneyPot Detection Tool V1.0

    Please select a specific Component BELOW

        1. Virtual Machine Detection

        2. Logins Details

        3. Status of the Ports

        4. ScreenShot Capture

        5. Installed Programs

        6. Processes and Services

        7. Logins

        8. HardWare Properties

        9. Quit and exit

    Enter Menu Option Number:
```

Figure 4.4 - Login Details Algorithm when launched

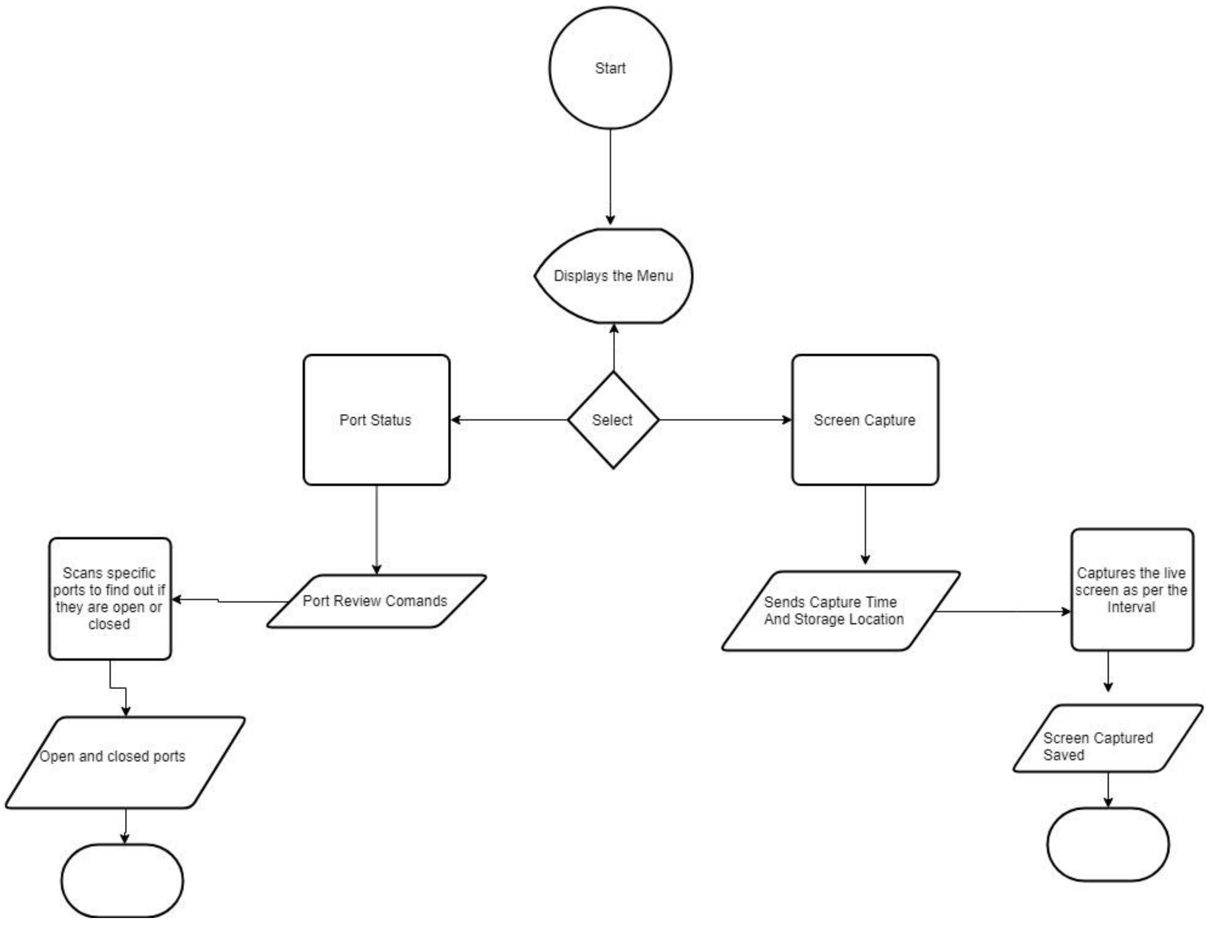


```
IsReadOnly     : False
IsFixedSize    : False
IsSynchronized : False
Keys           : {WIN-MES5TGV8ESH$WORKGROUPamuhumuzaWIN-MES5TGV8ESH2WIN-MES5TGV8ESH127.0.0.10}
Values         : {@{LogonType=2; NewLogonAccountName=amuhumuza; SourcePort=0; SourceNetworkAddress=127.0.0.1; Times=System.Object[];
                 LogSource=Security; SourceAccountName=WIN-MES5TGV8ESH$; WorkstationName=WIN-MES5TGV8ESH; Count=24; SourceDomainName=WORKGROUP;
                 NewLogonAccountDomain=WIN-MES5TGV8ESH; LogType=4624}}
SyncRoot       : System.Object
Count          : 1
```

The Login Details algorithm also displays whether the operating system is connected on a domain or is part of a workgroup. Today organizations mainly operate their networked servers using domains since they are supposed to provide services to other workstations in the organization, hence organization servers would be expected to have the operating system connected to a domain and not a workgroup. Figure 4.5 below shows the sample results from the Login Details algorithm.

Figure 4.5 - Login Details algorithm results

3) Status of the Ports



Start
Displays the Menu
Port Status
Select
Screen Capture
Scans specific ports to find out if they are open or closed
Port Review Comands
Sends Capture Time And Storage Location
Captures the live screen as per the Interval
Open and closed ports
Screen Captured Saved

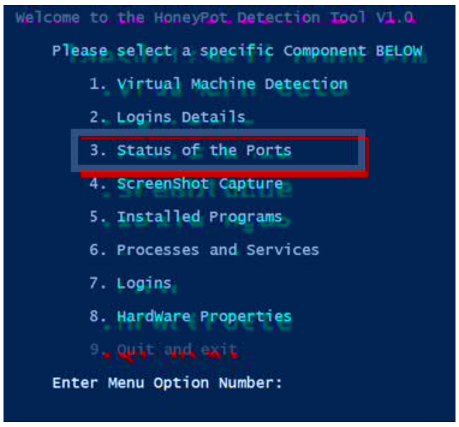
Figure 4.6

Flow Chart of the Port Status and Screen Capture Algorithm

The Status of the Ports tool displays information of the ports that are open on the operating system that is being probed. As shown in the Figure 4.6 above, the algorithm scans the top 1000 ports on the computer system to find out if the ports are open or closed. Particular ports have different functionalities in the operating environments for example oracle databases run on port 1521 whereas the SQL

Databases run on port 1433. So through the review of the open and closed ports one is able to tell the services that are running on a particular operating system. If the service being portrayed by the server is not present among the open ports then there is a high likelihood that the host is a honeypot.

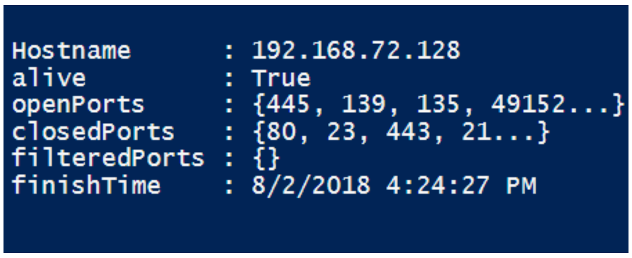
Figure 4.7 below the Status of the Ports algorithm presented on the menu of the Honey Pot Detection System.



```
Welcome to the HoneyPot Detection Tool V1.0

    Please select a specific Component BELOW

        1. Virtual Machine Detection
        2. Logins Details
        3. Status of the Ports
        4. ScreenShot Capture
        5. Installed Programs
        6. Processes and Services
        7. Logins
        8. HardWare Properties
        9. Quit and exit

    Enter Menu Option Number:
```

Figure 4.7 - Status of the Ports Algorithm when run

The tool returns the host IP, the live status, open ports, closed ports and the filtered ports. Filtered ports are common in production systems as used to give services to only preselected computing systems on a given network.



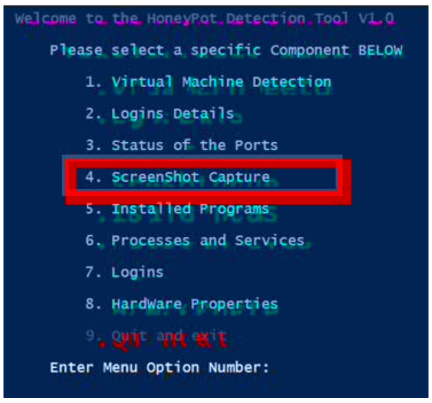
```
Hostname      : 192.168.72.128
alive         : True
openPorts     : {445, 139, 135, 49152...}
closedPorts   : {80, 23, 443, 21...}
filteredPorts : {}
finishTime    : 8/2/2018 4:24:27 PM
```

Figure 4.8- Status of the Ports Sample Results

4) Screenshot Capture Algorithm

The Screenshot Capture algorithm is used to capture the screenshots of the current activity on the on the operating system that is being interrogated. The screen capture algorithm requires a time for capture to be set and the location to store captured images as shown in the figure 4.6, it then captures the computer screen and saves the images to the provided location.

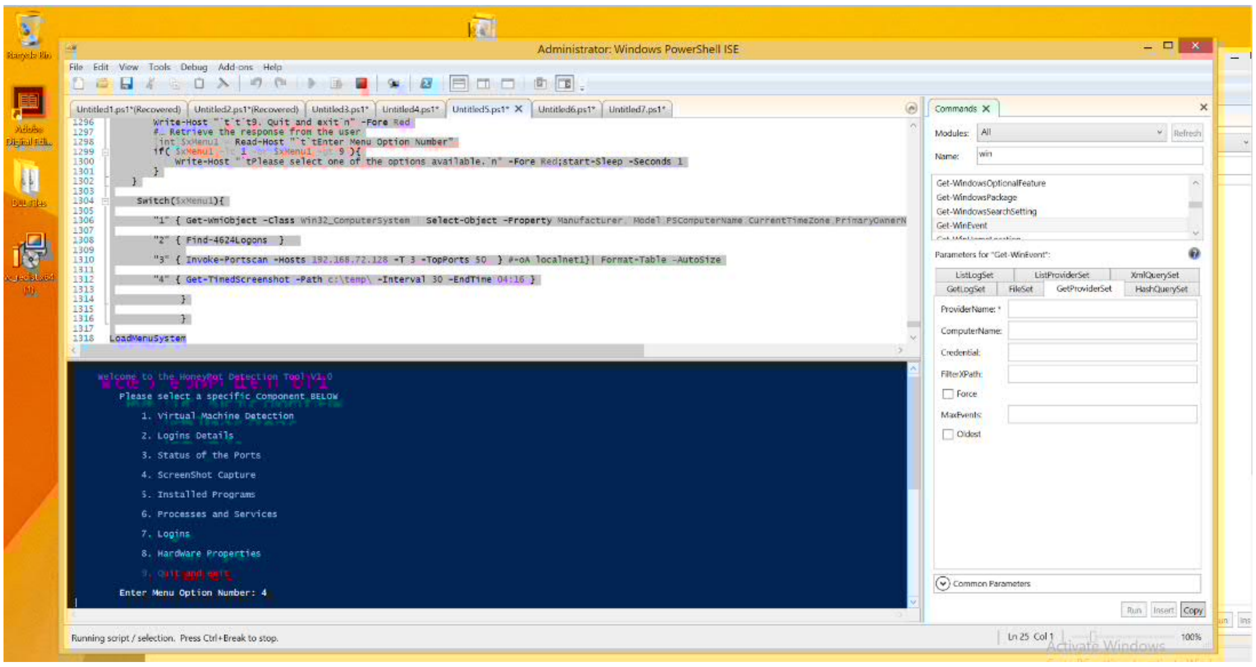
Little or no activity is expected on the processing server, as server management is carried or some configurations updated once in a while and not daily. Similarly honeypot system are meant to have little or no activity however they are easy to hack into which may lead to a lot of activity especially relating to querying what operating system and programs are running, hence it is likely to be a honeypot system. Figure 4.9 below the Screen Capture Algorithm when running.



```
Welcome to the HoneyPot Detection Tool V1.0

    Please select a specific Component BELOW

        1. Virtual Machine Detection
        2. Logins Details
        3. Status of the Ports
        4. ScreenShot Capture
        5. Installed Programs
        6. Processes and Services
        7. Logins
        8. Hardware Properties
        9. Quit and exit

    Enter Menu Option Number:
```

Figure 4.9 Screen Capture Algorithm when running

It captures screen shots at set time and interval and saves them to a defined location for review. This is to help review the amount of activity on the server to help identify if it's a honeypot or a production environment. Below is the sample results from the Screen Capture Algorithm



```
        Write-Host "`t`t`t9. Quit and exit`n" -Fore Red
        #… Retrieve the response from the user
        [int]$xMenu1 = Read-Host "`t`tEnter Menu Option Number"
        if( $xMenu1 -lt 1 -or $xMenu1 -gt 9 ){
            Write-Host "`tPlease select one of the options available.`n" -Fore Red;start-Sleep -Seconds 1
        }
    }

    Switch($xMenu1){

        "1" { Get-WmiObject -Class Win32_ComputerSystem | Select-Object -Property Manufacturer, Model PSComputerName CurrentTimeZone PrimaryOwnerN

        "2" { Find-4624Logons  }

        "3" { Invoke-Portscan -Hosts 192.168.72.128 -T 3 -TopPorts 50  } #-oA localnet1}| Format-Table -AutoSize

        "4" { Get-TimedScreenshot -Path c:\temp\ -Interval 30 -EndTime 04:16 }

            }

            }
LoadMenuSystem
```

```
Welcome to the HoneyPot Detection Tool V1.0
    Please select a specific Component BELOW
        1. Virtual Machine Detection
        2. Logins Details
        3. Status of the Ports
        4. ScreenShot Capture
        5. Installed Programs
        6. Processes and Services
        7. Logins
        8. Hardware Properties
        9. Quit and exit
    Enter Menu Option Number: 4
```

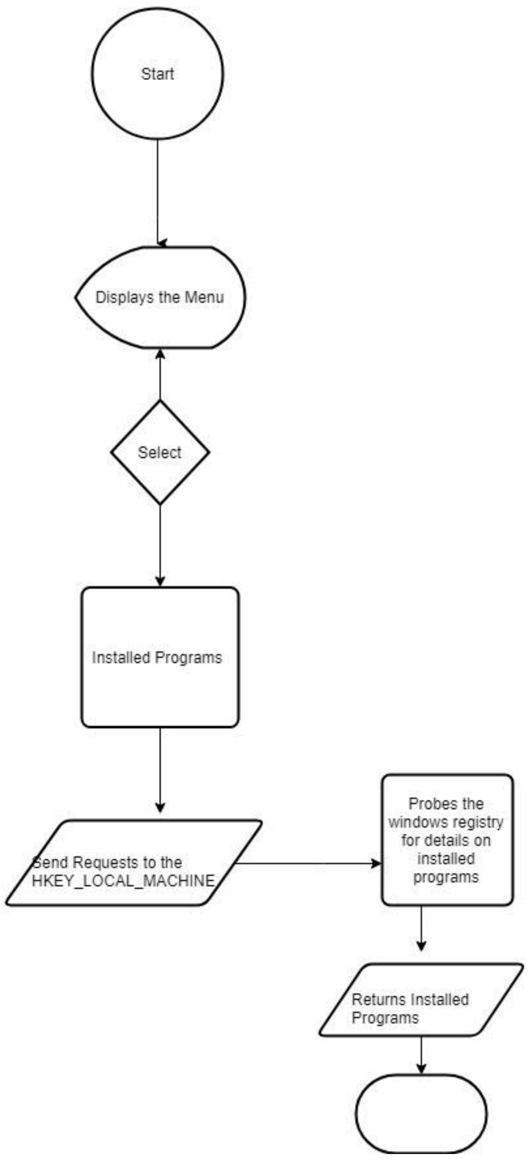
Running script / selection. Press Ctrl+Break to stop.

Figure 4.10 Sample Results from the Screen Capture algorithm.

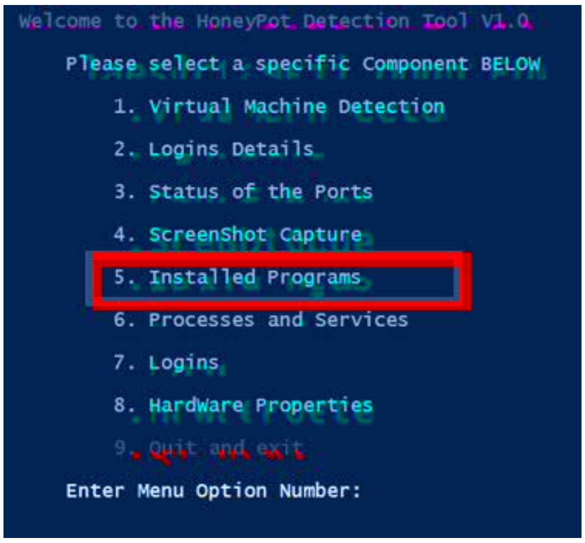
5) Installed Programs

Figure 4.11

Flow Chart of the Installed Programs

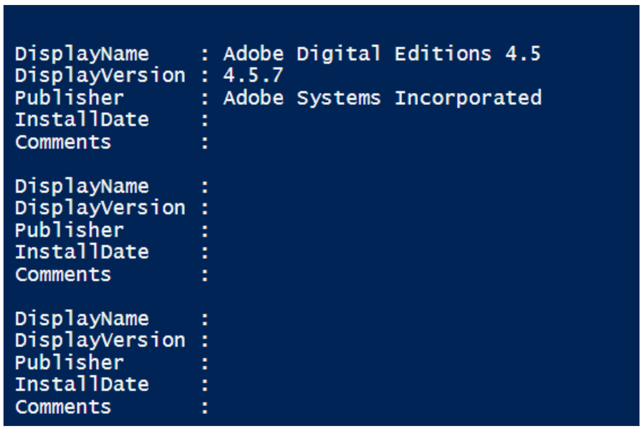

Start
Displays the Menu
Select
Installed Programs
Send Requests to the HKEY_LOCAL_MACHINE
Probes the windows registry for details on installed programs
Returns Installed Programs

The Installed Programs algorithm extracts from the registry a list of programs that are installed on the operating system. The installed programs are stored in the registry under the H_KEY_MACHINE, so the algorithm requests for the information from the registry and filters out the properties and only presents the Display Name, Version, Publisher Install date and Comments as shown in the flow chart in 4.11 above. The production servers always run particular programs that are related to the service that is being offered by the given server, however these applications are not installed on honeypots due to the expenses that come with licensing of the applications. This leads to honey pots not having any or having demonstration versions of the production software or none. Figure 4.12 below the Installed Programs algorithm being run.



```
Welcome to the HoneyPot Detection Tool V1.0

    Please select a specific Component BELOW

        1. Virtual Machine Detection

        2. Logins Details

        3. Status of the Ports

        4. ScreenShot Capture

        5. Installed Programs

        6. Processes and Services

        7. Logins

        8. HardWare Properties

        9. Quit and exit

    Enter Menu Option Number:
```

Installed Programs shows the installed program, the version of the program, Publisher of the Program and the date of installation of the given software. Below is a sample result from the Installed Programs algorithm.



```
DisplayName    : Adobe Digital Editions 4.5
DisplayVersion : 4.5.7
Publisher      : Adobe Systems Incorporated
InstallDate    :
Comments       :

DisplayName    :
DisplayVersion :
Publisher      :
InstallDate    :
Comments       :

DisplayName    :
DisplayVersion :
Publisher      :
InstallDate    :
Comments       :
```

Figure 4.13 - Program Installed Algorithm

6) Processes and Services

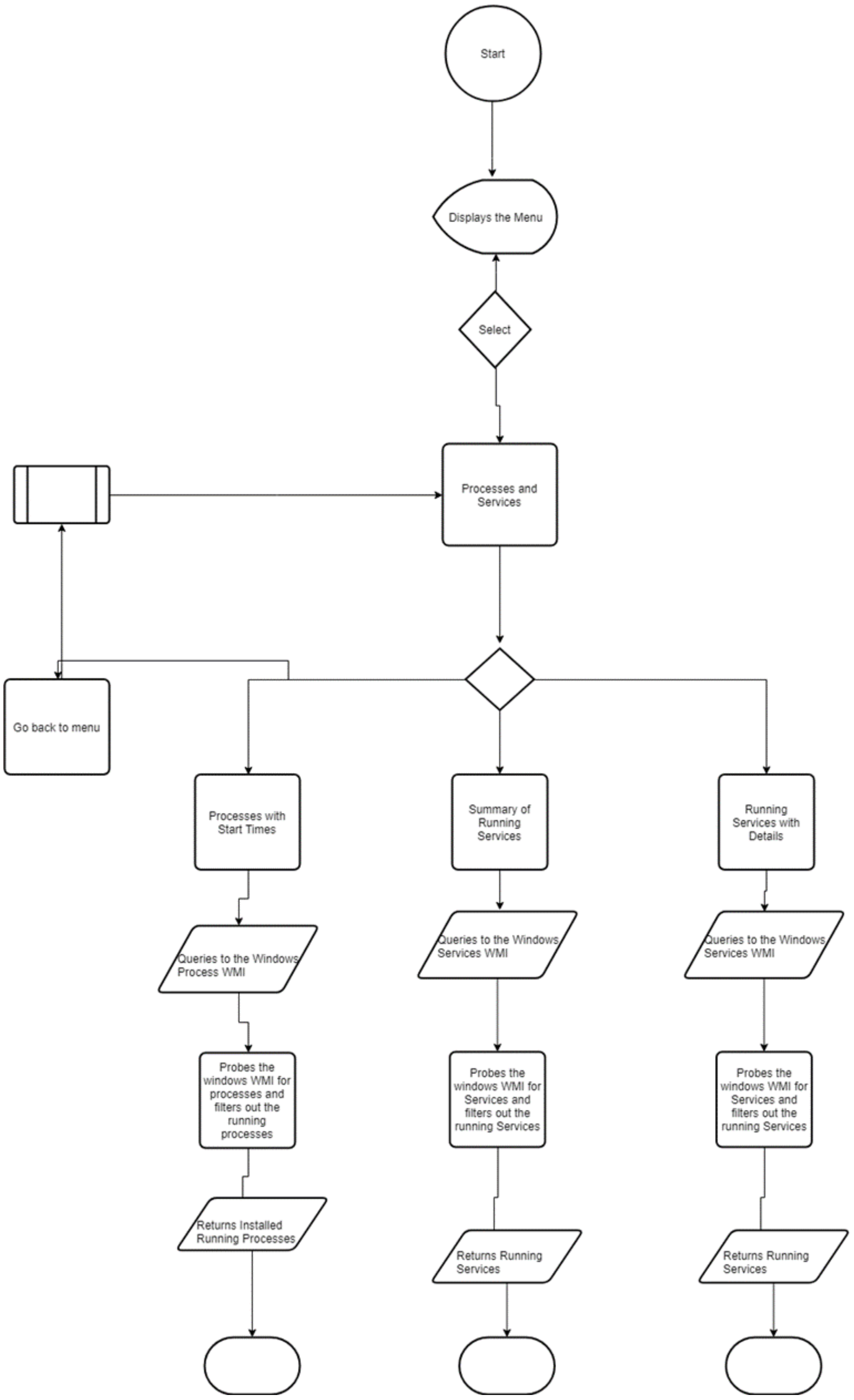
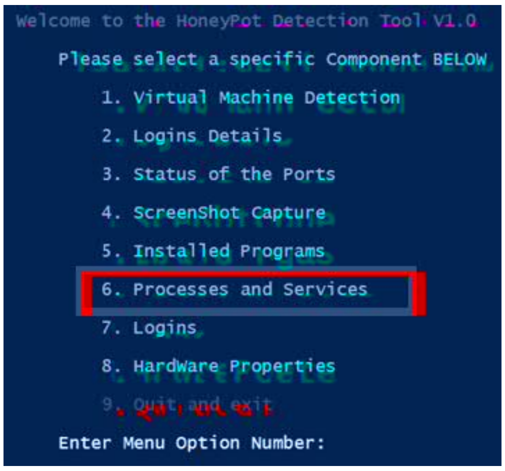

Start
Displays the Menu
Select
Processes and Services
Go back to menu
Processes with Start Times
Summary of Running Services
Running Services with Details
Queries to the Windows Process WMI
Queries to the Windows Services WMI
Queries to the Windows Services WMI
Probes the windows WMI for processes and filters out the running processes
Probes the windows WMI for Services and filters out the running Services
Probes the windows WMI for Services and filters out the running Services
Returns Installed Running Processes
Returns Running Services
Returns Running Services

Figure 4.14

Flow Chart of Processes and Services Algorithms

The Processes and Services algorithm is meant to provide the running processes and services on the operating system that is being probed. Processes are managed using the WMI for windows. The algorithm hence queries the WMI for processes that are running on the computer system and computes the running time which is then converted to DDHHMMSS and presents them with the CPU usage and start date, as shown in the flow chart in figure 4.15. For the running services it still queries the WMI for the process but uses different filters to display results for each algorithm.

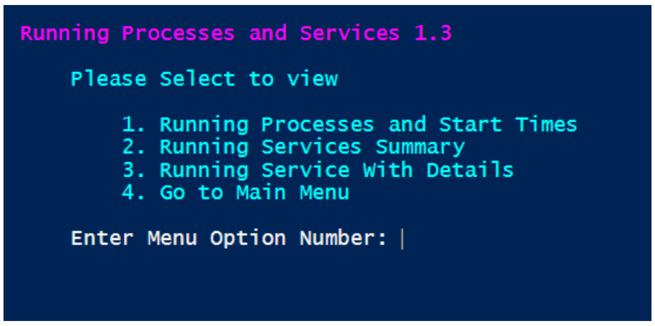
Production environments always have services and processes running in order to provide services to the other workstations on the network. Figure 4.16 below shows the Processes and Services on running the algorithm.



```
Welcome to the HoneyPot Detection Tool V1.0

    Please select a specific Component BELOW

        1. Virtual Machine Detection

        2. Logins Details

        3. Status of the Ports

        4. ScreenShot Capture

        5. Installed Programs

        6. Processes and Services

        7. Logins

        8. HardWare Properties

        9. Quit and exit

    Enter Menu Option Number:
```

Figure 4.16 Processes and Services Algorithm

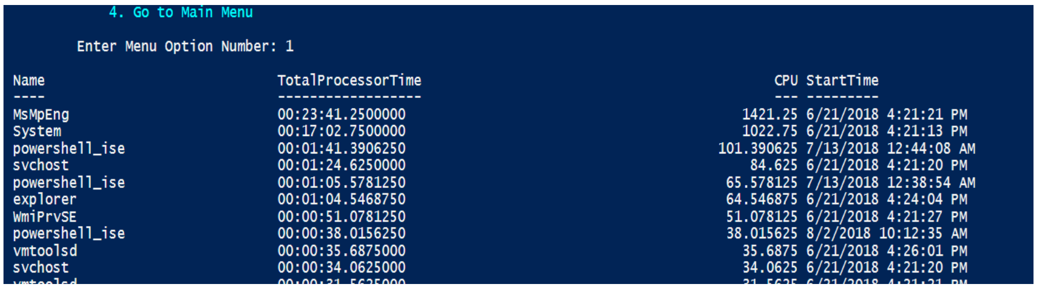
The Processes and functions is sub divided into three components which include Running Processes and Start Times, Running Services Summary and Running Service with Details. Figure 4.16 below shows the output from the selection of the Processes and Services Algorithm shown above.



```
Running Processes and Services 1.3

    Please Select to view

        1. Running Processes and Start Times
        2. Running Services Summary
        3. Running Service With Details
        4. Go to Main Menu

    Enter Menu Option Number: |
```

Figure 4.17 Output from the Processes and Services Algorithm

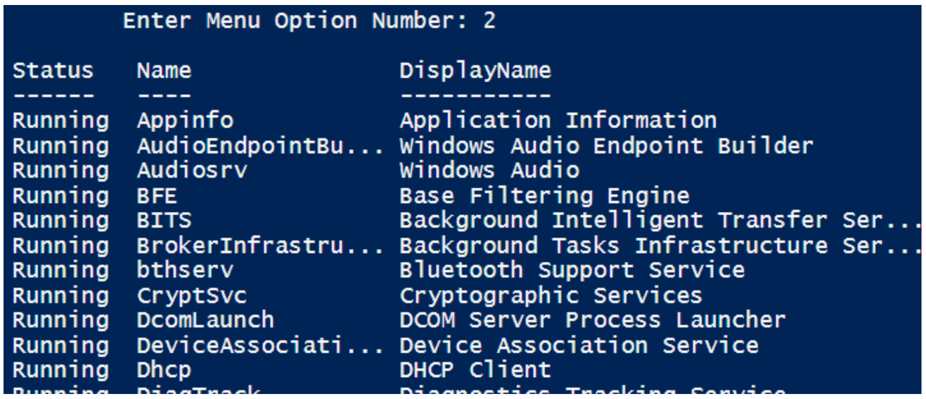
The Running Processes algorithm provides details of the Name of the process, Total Processor Time showing for how long a particular process has been running, CPU showing the usage of the different processes and StartTime showing the time the process started running. Figure 4.17 below shows the output from the running processes algorithm.



```
            4. Go to Main Menu

        Enter Menu Option Number: 1

Name                             TotalProcessorTime                                  CPU StartTime
----                             ------------------                                  --- ---------
MsMpEng                          00:23:41.2500000                                1421.25 6/21/2018 4:21:21 PM
System                           00:17:02.7500000                                1022.75 6/21/2018 4:21:13 PM
powershell_ise                   00:01:41.3906250                             101.390625 7/13/2018 12:44:08 AM
svchost                          00:01:24.6250000                                 84.625 6/21/2018 4:21:20 PM
powershell_ise                   00:01:05.5781250                              65.578125 7/13/2018 12:38:54 AM
explorer                         00:01:04.5468750                              64.546875 6/21/2018 4:24:04 PM
WmiPrvSE                         00:00:51.0781250                              51.078125 6/21/2018 4:21:27 PM
powershell_ise                   00:00:38.0156250                              38.015625 8/2/2018 10:12:35 AM
vmtoolsd                         00:00:35.6875000                                35.6875 6/21/2018 4:26:01 PM
svchost                          00:00:34.0625000                                34.0625 6/21/2018 4:21:20 PM
```

Figure 4.17 Running Processes

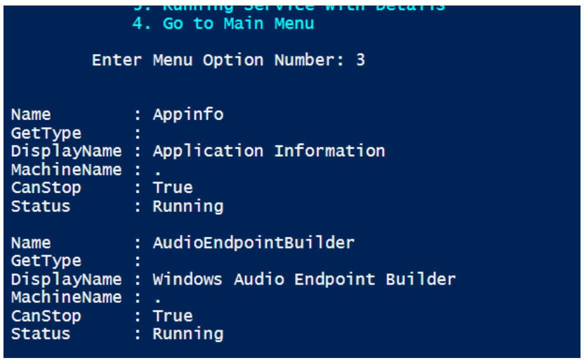
The Running Services Summary algorithm provides the status of the running services and the Name of the Services. This is meant to help point out particular services operating services, for example the oracle or SQL services on a database that provides Database Services. Figure 4.18 below shows the output from the running processes algorithm.



```
        Enter Menu Option Number: 2

Status   Name               DisplayName
------   ----               -----------
Running  Appinfo            Application Information
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  Audiosrv           Windows Audio
Running  BFE                Base Filtering Engine
Running  BITS               Background Intelligent Transfer Ser...
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  bthserv            Bluetooth Support Service
Running  CryptSvc           Cryptographic Services
Running  DcomLaunch         DCOM Server Process Launcher
Running  DeviceAssociati... Device Association Service
Running  Dhcp               DHCP Client
```

Figure 4.18 Running Services Algorithm Sample output

Running Services Details Provides and the status of the services that are running and if the services can be stopped by the user or not. Figure 4.19 Show the output of the running services details.



```
            4. Go to Main Menu

        Enter Menu Option Number: 3


Name        : Appinfo
GetType     :
DisplayName : Application Information
MachineName : .
CanStop     : True
Status      : Running

Name        : AudioEndpointBuilder
GetType     :
DisplayName : Windows Audio Endpoint Builder
MachineName : .
CanStop     : True
Status      : Running
```

Figure 4.19 Running Services Details Sample output

7) Logins

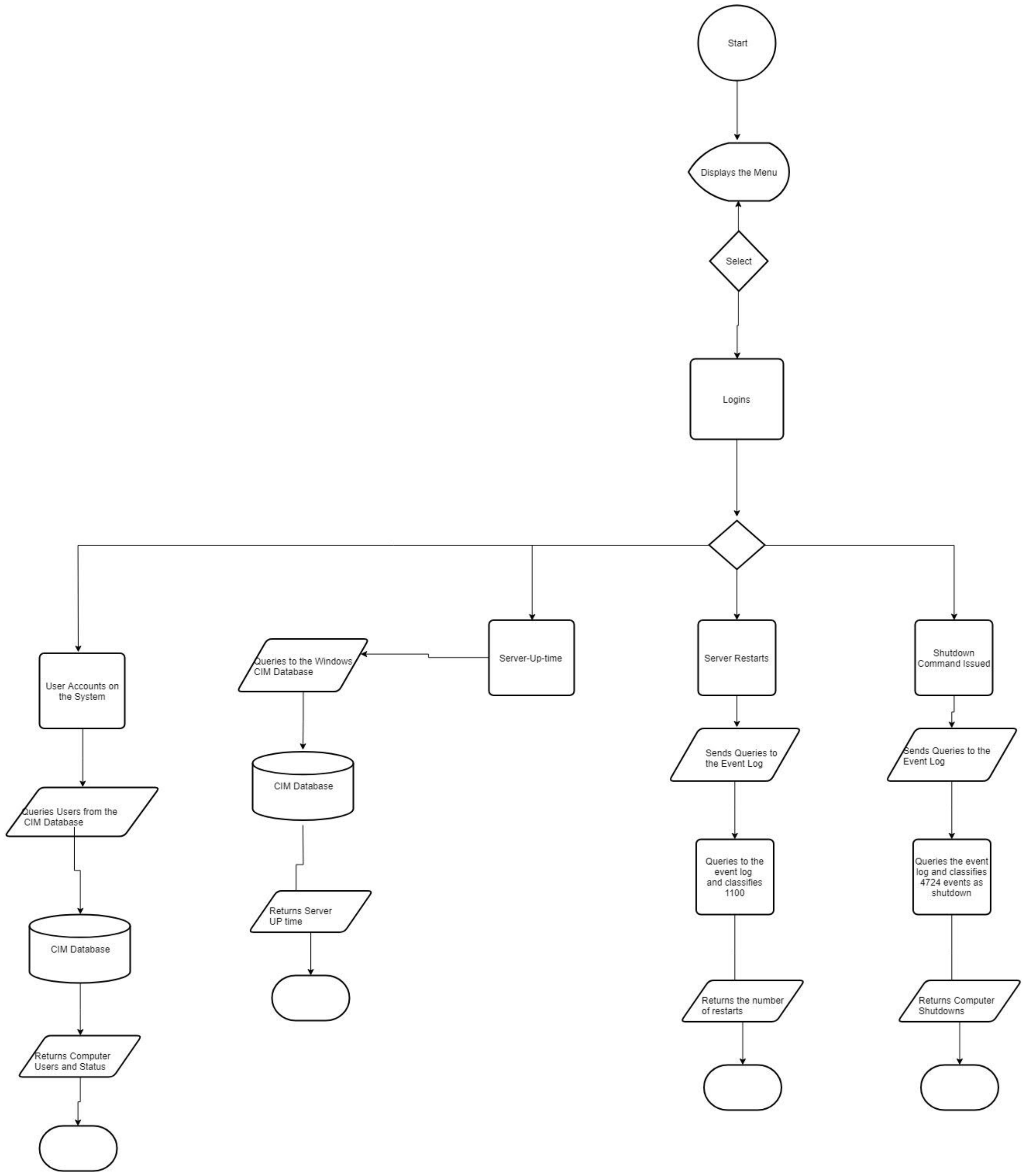
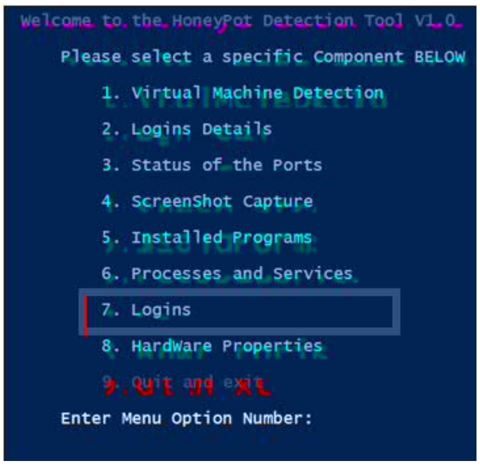

Start
Displays the Menu
Select
Logins
User Accounts on the System
Queries Users from the CIM Database
CIM Database
Returns Computer Users and Status
Server-Up-time
Queries to the Windows CIM Database
CIM Database
Returns Server UP time
Server Restarts
Sends Queries to the Event Log
Queries to the event log and classifies 1100
Returns the number of restarts
Shutdown Command Issued
Sends Queries to the Event Log
Queries the event log and classifies 4724 events as shutdown
Returns Computer Shutdowns

Figure 4.20 Flow Chart of the Login Algorithm

Logins provides information on server up-time or for how long the server has been running, server restarts or number of times the restart command has been issued to the server, number of time and when the shutdown command was given to the servers and User Accounts that are used to log onto the server. It uses the event log to filter out information and later classification to present the Server shutdowns and server restarts. It then interacts with WMI to find out for how long the system has been running and the user accounts on the given computer system.

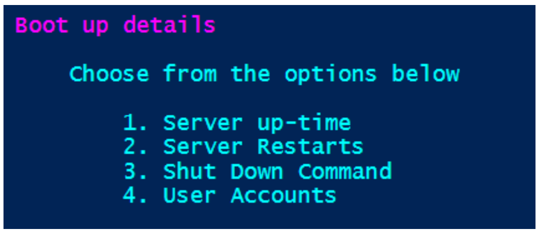
Honeypot machines are not usually protected from the power fluctuation and power loss and hence can always be turned off inconsistent power. Figure 4.21 shows the Logins Menu on the Honeypot detection system.



```
Welcome to the HoneyPot Detection Tool V1.0
    Please select a specific Component BELOW
        1. Virtual Machine Detection
        2. Logins Details
        3. Status of the Ports
        4. ScreenShot Capture
        5. Installed Programs
        6. Processes and Services
        7. Logins
        8. HardWare Properties
        9. Quit and exit
    Enter Menu Option Number:
```

Figure 4.21 Logins Menu

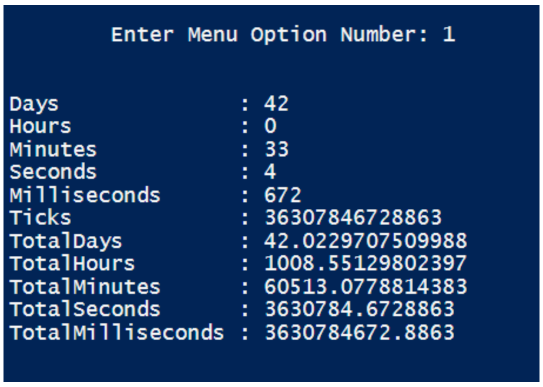
Servers are not supposed to be restarted often as that would lead to downtime which directly affects the services being provide to the users. Hence review of the uptime, restarts and shutdown would help review the system up uptime. If it's being restarted or shutdown often then it's highly likely to be a honeypot. Figure 4.22 show the Boot up detail menu which is the stems from the Login Menu Algorithm



```
Boot up details

    Choose from the options below

        1. Server up-time
        2. Server Restarts
        3. Shut Down Command
        4. User Accounts
```

Figure 4.22 Boot up Details Algorithm Results

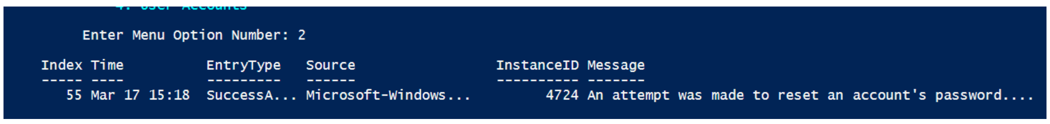
The server-up time algorithm queries the system to puck out for how long this system has been running, since servers are not always powered on and off and rarely face power failure challenges, the server uptime should be up to about 3 months and not a few days or weeks. Hence if a server has been up for about 3 months there are high chances that the server is not a honeypot. Figure 4.23 below shows the sample output for the server uptime algorithm.



```
        Enter Menu Option Number: 1


Days              : 42
Hours             : 0
Minutes           : 33
Seconds           : 4
Milliseconds      : 672
Ticks             : 36307846728863
TotalDays         : 42.0229707509988
TotalHours        : 1008.55129802397
TotalMinutes      : 60513.0778814383
TotalSeconds      : 3630784.6728863
TotalMilliseconds : 3630784672.8863
```

Figure 4.23 Server-up time algorithm

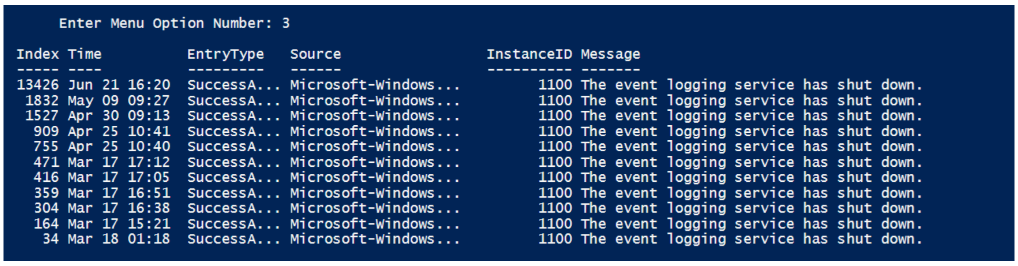
The server restarts algorithm queries the operating system to find the times the server has been restarted including the date and time of the server restart. If the server is often restarted there is a high probability of being a honey pot as production servers are not restarted often. Figure 4.24 below shows the sample output of the server restart algorithm.



```
      Enter Menu Option Number: 2

 Index Time          EntryType   Source                 InstanceID Message
 ----- ----          ---------   ------                 ---------- -------
    55 Mar 17 15:18  SuccessA... Microsoft-Windows...         4724 An attempt was made to reset an account's password....
```

Figure 4.24 Server Restarts Algorithm Results

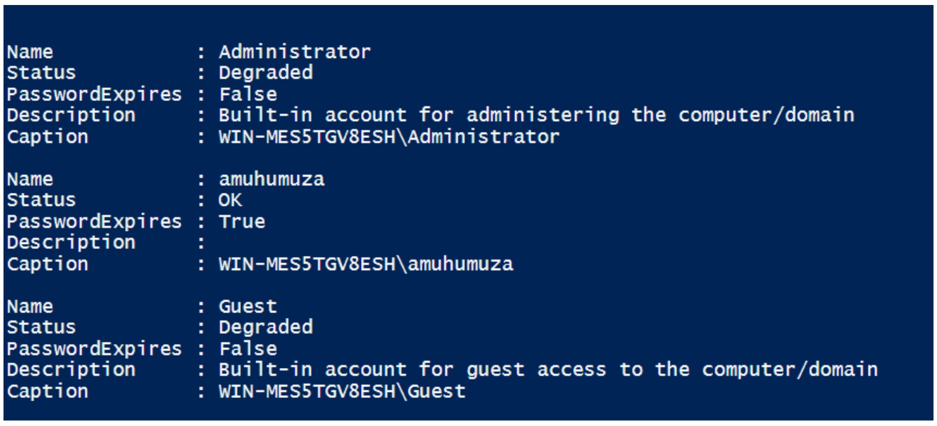
The shutdown command algorithm queries the operating system to find out the different times the shutdown command has been issued to the system. Production servers are rarely shut down and hence if the system is being shut down often then it may be a honey pot and not a production system. Figure 4.25 below shows a sample output from the shutdown command algorithm.



```
     Enter Menu Option Number: 3

Index Time          EntryType   Source                 InstanceID Message
----- ----          ---------   ------                 ---------- -------
13426 Jun 21 16:20  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
 1832 May 09 09:27  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
 1527 Apr 30 09:13  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  909 Apr 25 10:41  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  755 Apr 25 10:40  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  471 Mar 17 17:12  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  416 Mar 17 17:05  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  359 Mar 17 16:51  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  304 Mar 17 16:38  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  164 Mar 17 15:21  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
   34 Mar 18 01:18  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
```

Figure 4.25 Shut Down Command Algorithm Results

The user accounts algorithm displays the accounts that are able to log into the particular system. If the computer is connected a domain, it pulls all the user names on the domain as they could be used to log into the given server with different user roles. It shows the Users Names, Status(if log on is possible), and the possible connection. If the system is expected to be on a domain then a couple of users should be displayed, else the computer could be a honeypot. Figure 4.26 below show the sample output for the user account algorithm.



```
Name            : Administrator
Status          : Degraded
PasswordExpires : False
Description     : Built-in account for administering the computer/domain
Caption         : WIN-MES5TGV8ESH\Administrator

Name            : amuhumuza
Status          : OK
PasswordExpires : True
Description     :
Caption         : WIN-MES5TGV8ESH\amuhumuza

Name            : Guest
Status          : Degraded
PasswordExpires : False
Description     : Built-in account for guest access to the computer/domain
Caption         : WIN-MES5TGV8ESH\Guest
```

Figure 4.26 User Accounts Algorithm Results

CHAPTER 5:

PRESENTATION OF RESULTS

This chapter presents and analyses the results from the functionality testing of the honeypot detection system.

The honeypot detection system was tested on various environments, these included;

1) Honey Drive which is a Linux distro that emulates both Linux and windows production environments.
2) Windows Virtual Machine which emulates a honeypot that is deployed as a redundant computer with no applications and simple configurations on a network to attract attackers.
3) A windows Workstation that is used to carry out day-to-day activities in an organization, this is to be used to eliminate production workstations from being confused with production servers.
4) Windows Production server which is used as a control to show the server setting and what honeypots should be configured to emulate.

The test results are shown in the tables below.

1) Virtual Machine Test

   Table 5.1 Virtual Machine Test Results

	Hosted on a Virtual Machine	Appendices
Honey Drive(Virtual Honeypot)	N/A	
Windows Virtual Machine	Yes	Appendix 1
Windows Workstation	No	Appendix 2
Production Server	Yes	Appendix 3

   The table above shows the results from the virtual machine check, the test was not applicable to the Honey Drive as it doesn't support the running of PowerShell scripts which would lead to a deduction that the computer is a honeypot. The results also indicated that the Windows Virtual Machine and the Production Server were both hosted a virtual machine whereas the Windows workstation was hosted on hardware. The test has to be supported by other tests in order to confirm if the computer is a honeypot.

2) Running Services

   Table 5.2 Running Services

	Number of services Running	Database Services	Applications	Appendices
Honey Drive(Virtual Honeypot)	N/A	N/A	N/A	
Windows Virtual Machine	65	No	No	Appendix 4
Windows Workstation	115	No	Yes	Appendix 5
Production Server	69	Yes	No	Appendix 6

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results also indicated that the windows virtual machine was running 65 services but didn't include any application or database services, this creates a high like hood that it's a honeypot as it is unable to provide services to any other systems on the network. The results also indicated that the Windows WorkStation had 115 services running, with no database service but application services which means that it is used for day to day activities and hence is not a server that hold organizational critical information. The production server had 69 services running with no daily use applications running but with a running database. This test confirms that computers with no applications will be used as honeypots due to the costs involved in licensing the proprietary software used to run organizations.

3) Running Processes

Table 5.3 - Running Processes

	Running Processes	Database Processes	Applications	Appendices
Honey Drive(Virtual Honeypot)	N/A	N/A	N/A	
Windows Virtual Machine	Yes	No	No	Appendix 7
Windows Workstation	Yes	No	Yes	Appendix 8
Production Server	Not Tested	Not Tested	Not Tested	

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot.

The results also indicated that the Windows Virtual Machine had running processes but no database processes but contained application processes. The test was not done on the production server not to expose the server processes. The tests however showed that the windows Workstation is used for daily work whereas the Windows Virtual Machine is not a server as no application or database process are running.

4) Open Ports

Table 5.4 - Open Ports

	Open Ports	Database Ports	Applications Ports	Appendices
Honey Drive(Virtual Honeypot)	N/A	N/A	N/A	
Windows Virtual Machine	Yes	No	No	Appendix 9
Windows Workstation	Yes	No	No	Appendix 10
Production Server	Yes	Yes	Yes	Appendix 11

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results showed that the rest of the computer systems all had open ports however only the production serve had open ports that deliver database and application services, this is an indication that both the workstation and virtual machine are not used as production servers.

5) Hardware Properties

Table 5.5 Hardware Properties

	Hard Drive Capacity	Free Space	RAM	Processor	Appendices
Honey Drive(Virtual Honeypot)	N/A		N/A	N/A	
Windows Virtual Machine	60 GB	45.5 GB	2GB	Core i7	Appendix 12
Windows Workstation	463.2 GB	33.2 GB	35GB	Core i7	Appendix 13
Production Server	399	196 GB	25 GB	3 Intel Xeox	Appendix 14

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results also showed that the Windows Virtual Machine had 60 GB of hard disk space and 2GB RAM unlike the server and work station that had 463.2 and 399 GB with 35GB and 25GB of RAM respectively. The results also indicated that the processors being used were core i7 for both windows virtual machine and the workstation whereas the production server was using 3 Intel Xeox processors. From the analysis the production server shows processing capability and room for data growth with the free disk space indicating that the other two are not production servers.

6) Boot-Up Details

Table 5.6 Boot-Up Details

	Server-Uptime (Days)	Hours	Shut Downs	Appendices
Honey Drive(Virtual Honeypot)	N/A	N/A	N/A	
Windows Virtual Machine	11	12	11 between March and June	Appendix 15
Windows Workstation	3	9	Not Tested	Appendix 16
Production Server	165	13	Not Tested	Appendix 17

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results also indicate that the virtual machine has be running for 11 days and 12 hours whereas the Windows workstation is been running for 3 days and 9 hours, they also show that the server has been running for the longest time which is 165 days and 13 hours. From these results we are able to tell that the virtual machine and the work station are not server as servers have to keep running to provide services.

7) Login Details

Table 5.7 - Login Details

	Number of Logins	users	Appendices
Honey Drive(Virtual Honeypot)	N/A	N/A	
Windows Virtual Machine	24	1	Appendix 18
Windows Workstation	Not Tested	Not Tested	N/A
Production Server	Not Tested	Not Tested	N/A

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results from the Windows Virtual Machine indicated that there were 24 logins and only one user can log into the machine. The User functions display all the users that can log into the system, if the system is connected to the domain it shows all the users with potential to log onto the network, hence if only one person is able to log onto that network the computer is a honeypot. The system was not tested on the production server and workstation as results can be used to interrupt production on the server.

8) Screenshot Captures

Table 5.8 - Screenshot Capture

	Screen Activity	Appendices
Honey Drive(Virtual Honeypot)	N/A	
Windows Virtual Machine	Nil	Appendix 19
Windows Workstation	Busy	Appendix 20
Production Server	Nil	Appendix 21

The results indicated that the test was not able to run on the Honey Drive honeypot due to the inability to the PowerShell programs hence we rightly concluded that is a honeypot. The results from the Windows Virtual Machine and the production server show that there is no activity on the on the screens of the system. Hence if there is no desktop activity there is need to review the other tools to ascertain where the environment is a honeypot or not. The result shows that there is activity on the windows workstation as it is used for carry out work on a daily basis.

With the different tools shown above the tool was able to find to systems that were honeypots, however more than one test has to be carried out to ascertain whether a system is a honeypot or a production server. The production server acted as a control to should what the production environment is like, for a honeypot to be undetectable it should be able to emulate the server as shown in the results above.

CHAPTER 6

FUTURE WORK

As digitization and use of systems takes over most roles that were being done manually, the need to protect organization assets grows. Effective honeypot system are hence very important in understanding the hacking methods being used to access the different systems in order to enhance the protection in the areas that are being attacked.

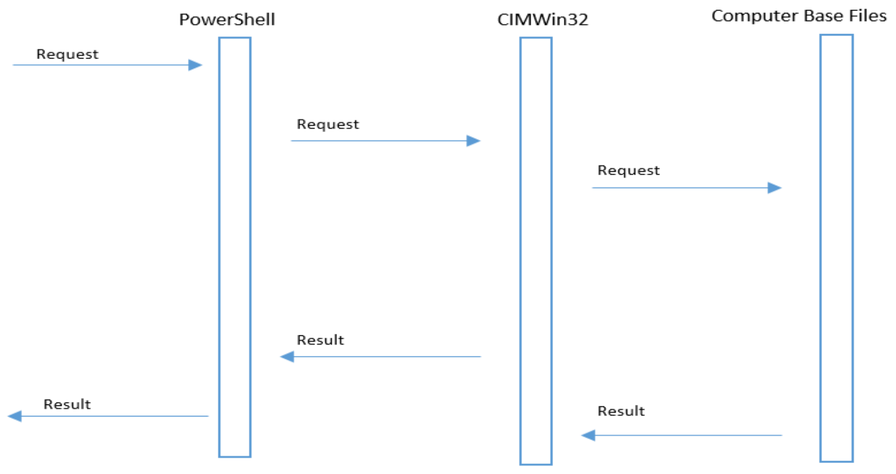
We propose one of the ways honeypots can be made undetected is by editing the files where the PowerShell commands pick particular information when requested. For example information about the Common Information Model (CIM) is stores information about dll the different windows components including the behavior of managed resources such as storage, network, or software components in the CIMWin32.dll file. We propose that systems built as honeypots should have their .
files that affect the results edited to help cater for specifications that fit the server processing environment whenever probed.

This proposed module is illustrated in the figures below. Figure A illustrates a simplified version of the workings of the current PowerShell system in the honey pot systems. The requests are made on the PowerShell interface and sent to the CIMWin32.dll which then sends the requests to the computer base files. The computer base file contain information about the computer properties such as the motherboard, hard disk storage, the installed processors and RAM. This information is sent back the PowerShell through the CIMWin32.dll and presented to the receiver.

This paper presents the design and development of a honeypot detection system to help honeypot developers develop honey pots that cannot be detected by hackers hence enabling the honeypot implementers to collect a lot of information about possible attacks and the tactics being used by the potential hackers to attack the system and help improve the protection of the systems.

From the tests carried out using the honeypot detection system we were able to note that all the honeypots in the current production environments can be detected using system analysis and network traffic, this hinders the effective operation of honeypots as they may not collect the important information they are meant to collect.

We hence recommend that future honeypot be built and tested with the honeypot detection system before being connected onto the networks in order to collect a lot of information from the attacker.



We further recommend that future honeypot harness the power of the PowerShell programing language to build scripts that can present desired results to the attackers by creating results that mirror server results when probed.

Figure 6.1 PowerShell Information Flow

The proposed model seeks the replacement of where information source from being the computer base file to a honey pot script designed to emulate a production server.

Figure B below shows the simplified proposed design, the information request flows as the one in figure A, however the information pick up is change to a script that emulates the actual system production system and can be edited to fit server specifications.

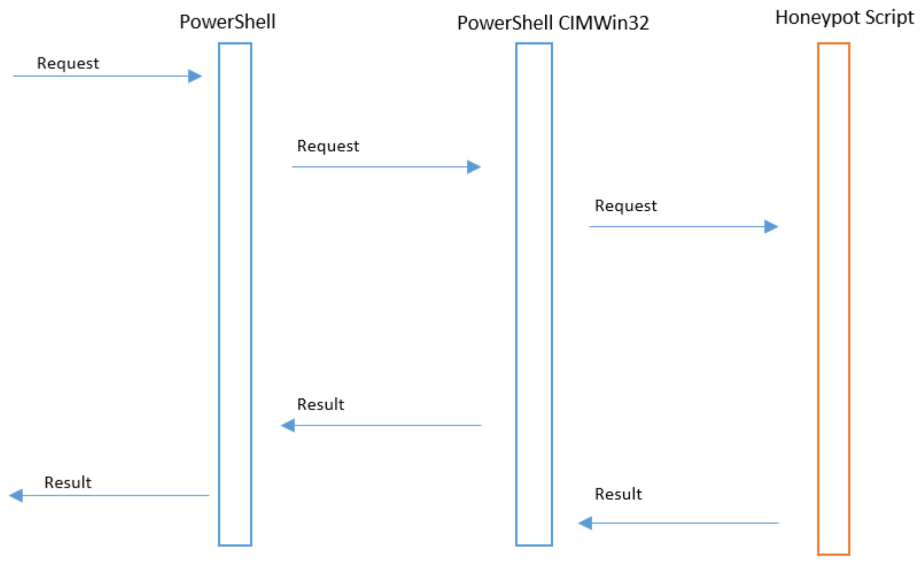


Figure 6.2 Simplified proposed design

REFERENCES

[1] L. Spitzner, HoneyPots, Boston: Addison-Wesley, 2002.

[2] "Honeynets: The Honeynet Project's Know Your Enemy Series," *Konw Your Enemy,* 2005.

[3] "Monitoring hacker activity with a honeynet," *International Journal of Network Management,* vol. 15, no. 2, pp. 123 - 134, 2005.

[4] M. S, Y. K, B. R, S. K. M and S. H. A, "Detection of Virtual Environments and Low Interaction Honeypots," New Mexico.

[5] L. Jiangzhou and H. Bin, "An active network defense policy-honeypot and its technologies," *Computer knowledge and technology,* no. 9, 2007.

[6] L. R. Even, "What is a Honeypot," SANS, 12 July 2000. [Online]. Available: https://www.sans.org/security-resources/idfaq/what-is-a-honeypot/1/9. [Accessed 04 October 2016].

[7] E. Nissan, Computer Applications for Handling Legal Evidence, Police, Springer, 2012.

[8] A. Verma, "Production Honeypots: An Organization's view," 23 October 2003. [Online]. Available: https://www.giac.org/paper/gsec/3585/production-honeypots-organizations-view/105831. [Accessed 03 October 2016].

[9] L. Spitzner, "Honeypots – Tracking Hackers," Addison-Wesley, 2002.

[10] P. N, "A virtual honeypot framework," in *Proceedings of the 13th USENIX Security Symposium*, 2004.

[11] R. Baumann and C. Plattner, "White Paper: Honeypots," *Swiss Federal Institute of Technology: Zurich,* 2002.

[12] I. Wigmore, Techtarget, September 2014. [Online]. Available: http://whatis.techtarget.com/definition/virtual-honeypot. [Accessed 3 October 2016].

[13] Archana and N. Gandhi, "Different Ways to Define Virtualization," *International Journal of Advanced Research in Computer Science and Software Engineering,* vol. 5, no. 9, pp. 331- 334, 2015.

[14] A. A. Masjedi, "Virtualization Programs," Auckland, 2012.

[15] W. Ping, W. Lei , C. Ryan and Z. C. Cliff , "Honeypot Detection in Advanced Botnet Attacks," *Information and Computer Security,* 2008.

[16] P. Krisztian and W. Sebastian , "Honeypot forensics," in *Risk Advisory Services*, Berlin, 2004.

[17] S. Anjali and R. C. Josh, Anti HoneyPot Technology, CRC Press, 2011.

continued on next page

continued from previous page

[18] D. Wanda and D. Nina , "A Honeypot Detection Method Based on Characteristic Analysis and Environment Detection," 2012, pp. 201-206.

[19] A. B. k. c. Tadayoshi Kohno, "Remote physical device fingerprinting," San Diego.

[20] X. Fu, W. Yu, D. Cheng, X. Tan, K. Streff and S. Graham, "On Recognizing Virtual Honeypots and Countermeasures," 2006.

[21] N. Krawetz, "Anti-honeypot technology," IEEE Security & Privacy Magazine, 2004.

[22] O. Hayatle, A. Youssef and H. Otrok, "Dempster-Shafer Evidence Combining for (Anti)-Honeypot Technologies," Electrical and Computer Engineering Department, Abu Dhabi.

[23] "Kali Linux Documentation," Offensive Security, [Online]. Available: http://docs.kali.org/introduction/what-is-kali-linux. [Accessed 05 October 2016].

[24] Microsoft, "Understanding a Windows PowerShell Module.," Microsoft, [Online]. Available: https://msdn.microsoft.com/en-us/library/dd878324(v=vs.85).aspx. [Accessed 29 01 2018].

[25] "NMAP," NMAP.ORG, [Online]. Available: www.nmap.org. [Accessed 05 October 2016].

[26] "KALI TOOLS," Offensive Security, [Online]. Available: http://tools.kali.org/tools-listing. [Accessed 05 October 2016].

[28] J. Gray, "Readers choice awards 2010," February 2010. [Online]. Available: http://www.linuxjournal.com/content/readers-choice-awards-2010. [Accessed 03 October 2016].

[29] M. Jan and O. Granberg , "Open-source virtualization," OSLO, 2013.

[30] G. Wang and T. S. Eugene, "The Impact of Virtualization on Network Performance of Amazon EC2 Data Center.," in *Proceedings of the 29th Conference on Information Communications*, NJ, USA, 2010.

[31] "Microsoft TechNet," Microsoft, 1 November 2013. [Online]. Available: https://technet.microsoft.com/en-us/library/dn383626(v=ws.11).aspx. [Accessed 04 October 2016].

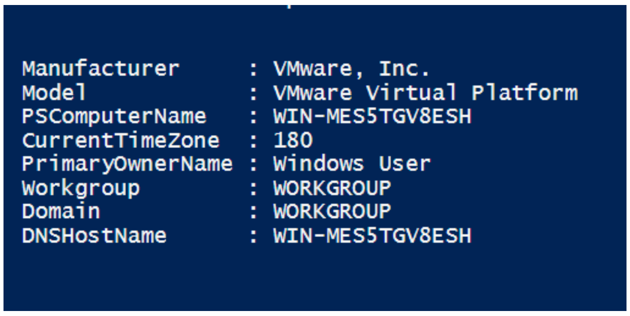
[32] "Red Hat Enterprise Linux Operating System Requirements," Redhat Community, [Online]. Available: https://access.redhat.com/documentation/en-US/Red_Hat_Directory_Server/8.2/html/Installation_Guide/Installation_Guide-RHEL-Requirements.html. [Accessed 4 October 2016].

[33] Microsoft, "PowerShell Overview.," Microsoft, [Online]. Available: https://docs.microsoft.com/en-sg/powershell/scripting/powershellscripting. [Accessed 29 01 2018].

APPENDICES

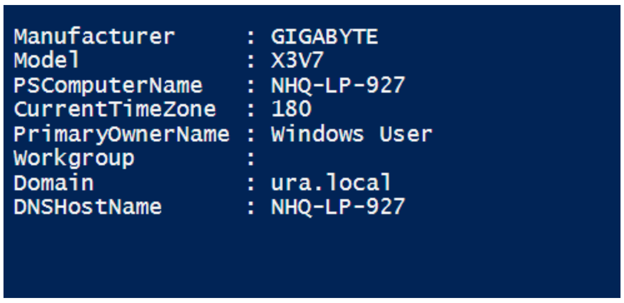
APPENDIX 1

Appendix 1 Virtual Machine Test Results Windows Virtual Machine



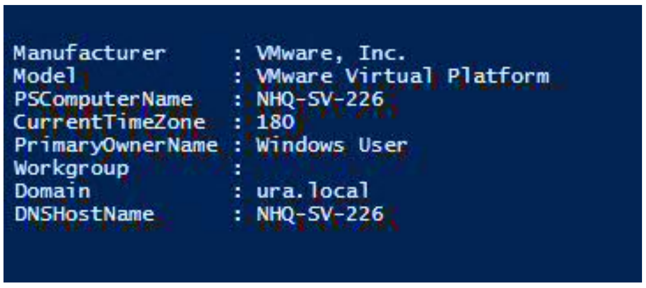
```
Manufacturer      : VMware, Inc.
Model             : VMware Virtual Platform
PSComputerName    : WIN-MES5TGV8ESH
CurrentTimeZone   : 180
PrimaryOwnerName  : Windows User
Workgroup         : WORKGROUP
Domain            : WORKGROUP
DNSHostName       : WIN-MES5TGV8ESH
```

Appendix 2 Virtual Machine Test Results Windows Work Station



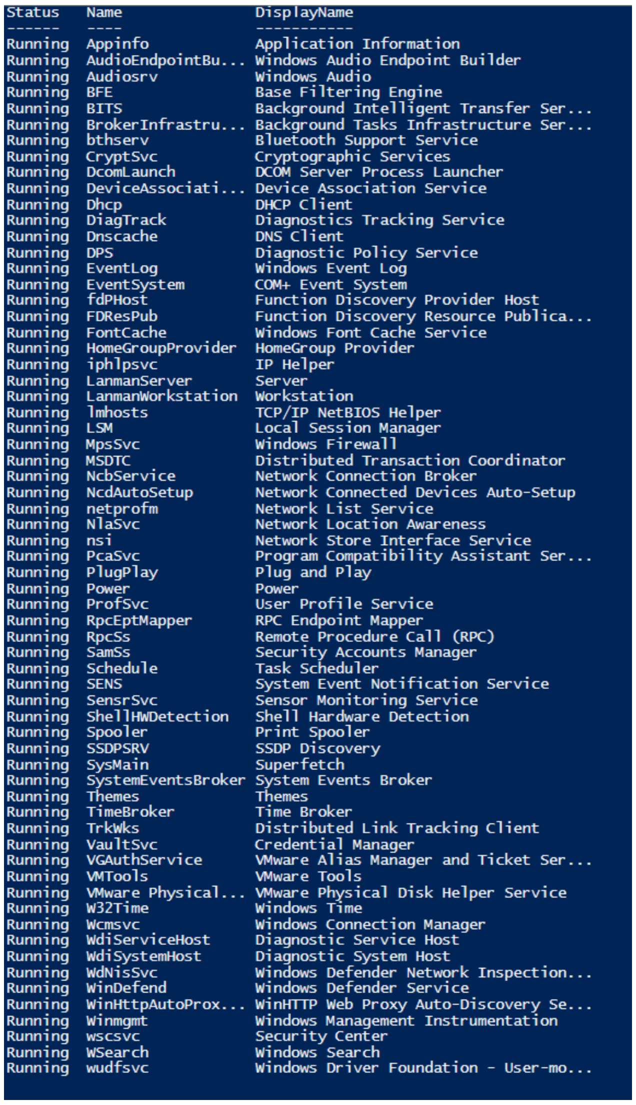
```
Manufacturer      : GIGABYTE
Model             : X3V7
PSComputerName    : NHQ-LP-927
CurrentTimeZone   : 180
PrimaryOwnerName  : Windows User
Workgroup         :
Domain            : ura.local
DNSHostName       : NHQ-LP-927
```

Appendix 3 Virtual Machine Test Results Productions Server



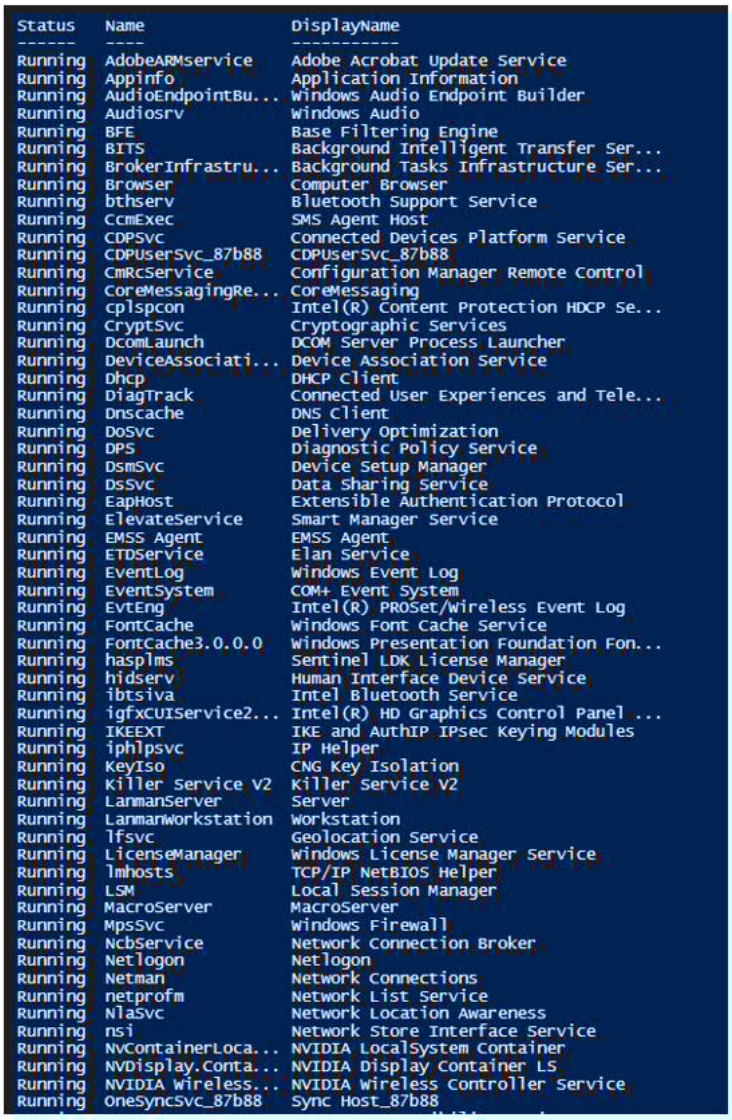
```
Manufacturer      : VMware, Inc.
Model             : VMware Virtual Platform
PSComputerName    : NHQ-SV-226
CurrentTimeZone   : 180
PrimaryOwnerName  : Windows User
Workgroup         :
Domain            : ura.local
DNSHostName       : NHQ-SV-226
```

Appendix 4 Running Services Windows VM

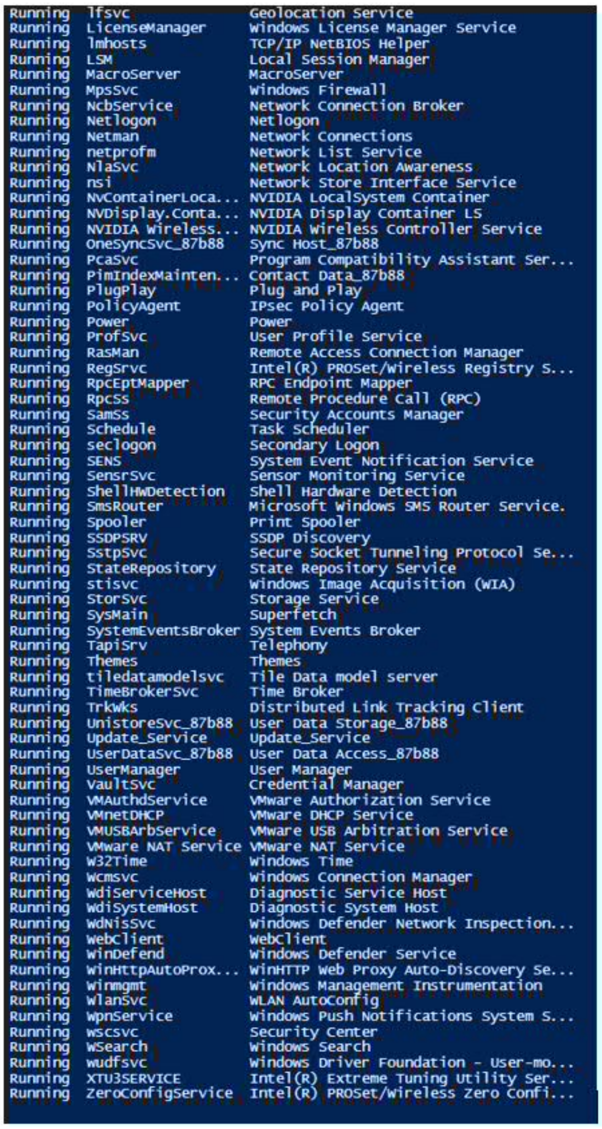


```
Status   Name               DisplayName
------   ----               -----------
Running  Appinfo            Application Information
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  Audiosrv           Windows Audio
Running  BFE                Base Filtering Engine
Running  BITS               Background Intelligent Transfer Ser...
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  bthserv            Bluetooth Support Service
Running  CryptSvc           Cryptographic Services
Running  DcomLaunch         DCOM Server Process Launcher
Running  DeviceAssociati... Device Association Service
Running  Dhcp               DHCP Client
Running  DiagTrack          Diagnostics Tracking Service
Running  Dnscache           DNS Client
Running  DPS                Diagnostic Policy Service
Running  EventLog           Windows Event Log
Running  EventSystem        COM+ Event System
Running  fdPHost            Function Discovery Provider Host
Running  FDResPub           Function Discovery Resource Publica...
Running  FontCache          Windows Font Cache Service
Running  HomeGroupProvider  HomeGroup Provider
Running  iphlpsvc           IP Helper
Running  LanmanServer       Server
Running  LanmanWorkstation  Workstation
Running  lmhosts            TCP/IP NetBIOS Helper
Running  LSM                Local Session Manager
Running  MpsSvc             Windows Firewall
Running  MSDTC              Distributed Transaction Coordinator
Running  NcbService         Network Connection Broker
Running  NcdAutoSetup       Network Connected Devices Auto-Setup
Running  netprofm           Network List Service
Running  NlaSvc             Network Location Awareness
Running  nsi                Network Store Interface Service
Running  PcaSvc             Program Compatibility Assistant Ser...
Running  PlugPlay           Plug and Play
Running  Power              Power
Running  ProfSvc            User Profile Service
Running  RpcEptMapper       RPC Endpoint Mapper
Running  RpcSs              Remote Procedure Call (RPC)
Running  SamSs              Security Accounts Manager
Running  Schedule           Task Scheduler
Running  SENS               System Event Notification Service
Running  SensrSvc           Sensor Monitoring Service
Running  ShellHWDetection   Shell Hardware Detection
Running  Spooler            Print Spooler
Running  SSDPSRV            SSDP Discovery
Running  SysMain            Superfetch
Running  SystemEventsBroker System Events Broker
Running  Themes             Themes
Running  TimeBroker         Time Broker
Running  TrkWks             Distributed Link Tracking Client
Running  VaultSvc           Credential Manager
Running  VGAuthService      VMware Alias Manager and Ticket Ser...
Running  VMTools            VMware Tools
Running  VMware Physical... VMware Physical Disk Helper Service
Running  W32Time            Windows Time
Running  Wcmsvc             Windows Connection Manager
Running  WdiServiceHost     Diagnostic Service Host
Running  WdiSystemHost      Diagnostic System Host
Running  WdNisSvc           Windows Defender Network Inspection...
Running  WinDefend          Windows Defender Service
Running  WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Running  Winmgmt            Windows Management Instrumentation
Running  wscsvc             Security Center
Running  WSearch            Windows Search
Running  wudfsvc            Windows Driver Foundation - User-mo...
```

Appendix 5 Running Services Windows Workstation

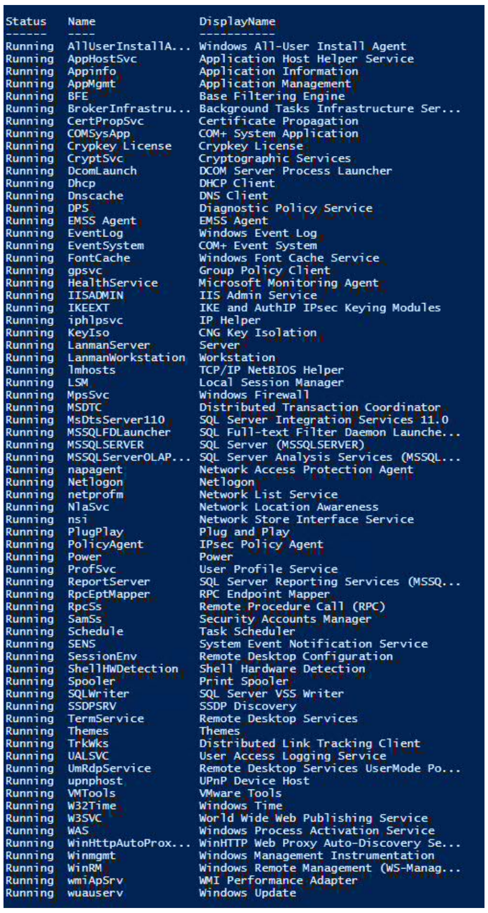


```
Status   Name               DisplayName
------   ----               -----------
Running  AdobeARMservice    Adobe Acrobat Update Service
Running  Appinfo            Application Information
Running  AudioEndpointBu... Windows Audio Endpoint Builder
Running  Audiosrv           Windows Audio
Running  BFE                Base Filtering Engine
Running  BITS               Background Intelligent Transfer Ser...
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  Browser            Computer Browser
Running  bthserv            Bluetooth Support Service
Running  CcmExec            SMS Agent Host
Running  CDPSvc             Connected Devices Platform Service
Running  CDPUserSvc_87b88   CDPUserSvc_87b88
Running  CmRcService        Configuration Manager Remote Control
Running  CoreMessagingRe... CoreMessaging
Running  cplspcon           Intel(R) Content Protection HDCP Se...
Running  CryptSvc           Cryptographic Services
Running  DcomLaunch         DCOM Server Process Launcher
Running  DeviceAssociati... Device Association Service
Running  Dhcp               DHCP Client
Running  DiagTrack          Connected User Experiences and Tele...
Running  Dnscache           DNS Client
Running  DoSvc              Delivery Optimization
Running  DPS                Diagnostic Policy Service
Running  DsmSvc             Device Setup Manager
Running  DsSvc              Data Sharing Service
Running  EapHost            Extensible Authentication Protocol
Running  ElevateService     Smart Manager Service
Running  EMSS Agent         EMSS Agent
Running  ETDService         Elan Service
Running  EventLog           Windows Event Log
Running  EventSystem        COM+ Event System
Running  EvtEng             Intel(R) PROSet/Wireless Event Log
Running  FontCache          Windows Font Cache Service
Running  FontCache3.0.0.0   Windows Presentation Foundation Fon...
Running  hasplms            Sentinel LDK License Manager
Running  hidserv            Human Interface Device Service
Running  ibtsiva            Intel Bluetooth Service
Running  igfxCUIService2... Intel(R) HD Graphics Control Panel ...
Running  IKEEXT             IKE and AuthIP IPsec Keying Modules
Running  iphlpsvc           IP Helper
Running  KeyIso             CNG Key Isolation
Running  Killer Service V2  Killer Service V2
Running  LanmanServer       Server
Running  LanmanWorkstation  Workstation
Running  lfsvc              Geolocation Service
Running  LicenseManager     Windows License Manager Service
Running  lmhosts            TCP/IP NetBIOS Helper
Running  LSM                Local Session Manager
Running  MacroServer        MacroServer
Running  MpsSvc             Windows Firewall
Running  NcbService         Network Connection Broker
Running  Netlogon           Netlogon
Running  Netman             Network Connections
Running  netprofm           Network List Service
Running  NlaSvc             Network Location Awareness
Running  nsi                Network Store Interface Service
Running  NvContainerLoca... NVIDIA LocalSystem Container
Running  NVDisplay.Conta... NVIDIA Display Container LS
Running  NVIDIA Wireless... NVIDIA Wireless Controller Service
Running  OneSyncSvc_87b88   Sync Host_87b88
```



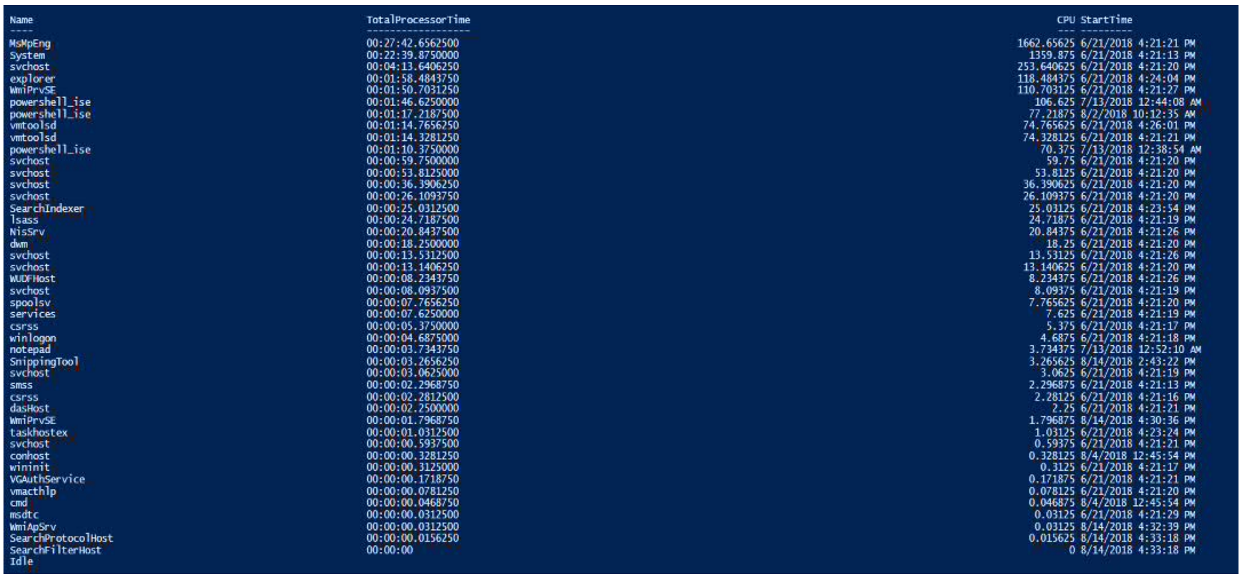
```
Running  lfsvc              Geolocation Service
Running  LicenseManager     Windows License Manager Service
Running  lmhosts            TCP/IP NetBIOS Helper
Running  LSM                Local Session Manager
Running  MacroServer        MacroServer
Running  MpsSvc             Windows Firewall
Running  NcbService         Network Connection Broker
Running  Netlogon           Netlogon
Running  Netman             Network Connections
Running  netprofm           Network List Service
Running  NlaSvc             Network Location Awareness
Running  nsi                Network Store Interface Service
Running  NvContainerLoca... NVIDIA LocalSystem Container
Running  NVDisplay.Conta... NVIDIA Display Container LS
Running  NVIDIA Wireless... NVIDIA Wireless Controller Service
Running  OneSyncSvc_87b88   Sync Host_87b88
Running  PcaSvc             Program Compatibility Assistant Ser...
Running  PimIndexMainten... Contact Data_87b88
Running  PlugPlay           Plug and Play
Running  PolicyAgent        IPsec Policy Agent
Running  Power              Power
Running  ProfSvc            User Profile Service
Running  RasMan             Remote Access Connection Manager
Running  RegSrvc            Intel(R) PROSet/Wireless Registry S...
Running  RpcEptMapper       RPC Endpoint Mapper
Running  RpcSs              Remote Procedure Call (RPC)
Running  SamSs              Security Accounts Manager
Running  Schedule           Task Scheduler
Running  seclogon           Secondary Logon
Running  SENS               System Event Notification Service
Running  SensrSvc           Sensor Monitoring Service
Running  ShellHWDetection   Shell Hardware Detection
Running  SmsRouter          Microsoft Windows SMS Router Service.
Running  Spooler            Print Spooler
Running  SSDPSRV            SSDP Discovery
Running  SstpSvc            Secure Socket Tunneling Protocol Se...
Running  StateRepository    State Repository Service
Running  stisvc             Windows Image Acquisition (WIA)
Running  StorSvc            Storage Service
Running  SysMain            Superfetch
Running  SystemEventsBroker System Events Broker
Running  TapiSrv            Telephony
Running  Themes             Themes
Running  tiledatamodelsvc   Tile Data model server
Running  TimeBrokerSvc      Time Broker
Running  TrkWks             Distributed Link Tracking Client
Running  UnistoreSvc_87b88  User Data Storage_87b88
Running  Update_Service     Update_Service
Running  UserDataSvc_87b88  User Data Access_87b88
Running  UserManager        User Manager
Running  VaultSvc           Credential Manager
Running  VMAuthdService     VMware Authorization Service
Running  VMnetDHCP          VMware DHCP Service
Running  VMUSBArbService    VMware USB Arbitration Service
Running  VMware NAT Service VMware NAT Service
Running  W32Time            Windows Time
Running  Wcmsvc             Windows Connection Manager
Running  WdiServiceHost     Diagnostic Service Host
Running  WdiSystemHost      Diagnostic System Host
Running  WdNisSvc           Windows Defender Network Inspection...
Running  WebClient          WebClient
Running  WinDefend          Windows Defender Service
Running  WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Running  Winmgmt            Windows Management Instrumentation
Running  WlanSvc            WLAN AutoConfig
Running  WpnService         Windows Push Notifications System S...
Running  wscsvc             Security Center
Running  WSearch            Windows Search
Running  wudfsvc            Windows Driver Foundation - User-mo...
Running  XTU3SERVICE        Intel(R) Extreme Tuning Utility Ser...
Running  ZeroConfigService  Intel(R) PROSet/Wireless Zero Confi...
```

Appendix 6 Running Services Production Server



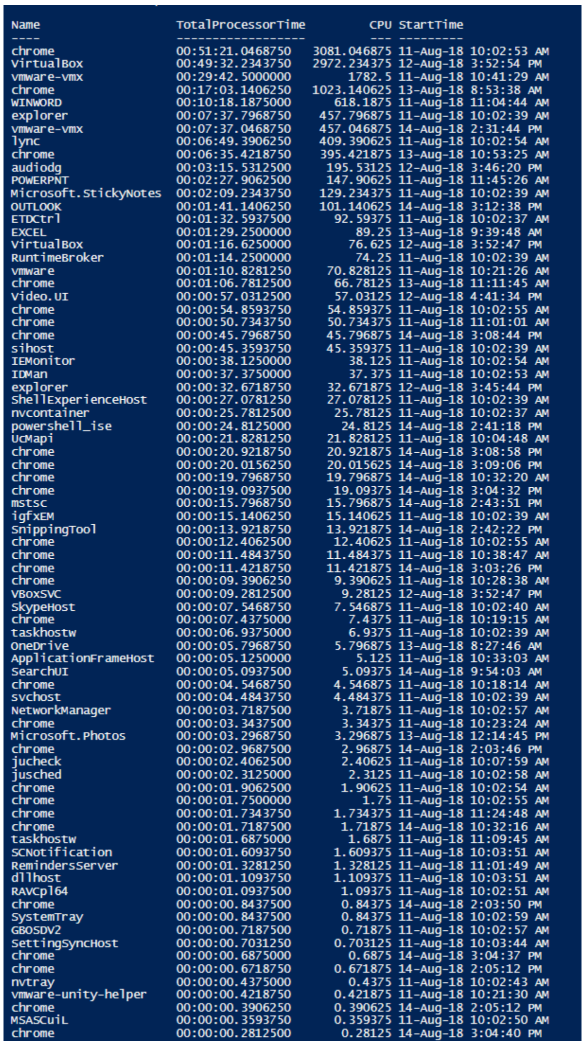
```
Status   Name               DisplayName
------   ----               -----------
Running  AllUserInstallA... Windows All-User Install Agent
Running  AppHostSvc         Application Host Helper Service
Running  Appinfo            Application Information
Running  AppMgmt            Application Management
Running  BFE                Base Filtering Engine
Running  BrokerInfrastru... Background Tasks Infrastructure Ser...
Running  CertPropSvc        Certificate Propagation
Running  COMSysApp          COM+ System Application
Running  Crypkey License    Crypkey License
Running  CryptSvc           Cryptographic Services
Running  DcomLaunch         DCOM Server Process Launcher
Running  Dhcp               DHCP Client
Running  Dnscache           DNS Client
Running  DPS                Diagnostic Policy Service
Running  EMSS Agent         EMSS Agent
Running  EventLog           Windows Event Log
Running  EventSystem        COM+ Event System
Running  FontCache          Windows Font Cache Service
Running  gpsvc              Group Policy Client
Running  HealthService      Microsoft Monitoring Agent
Running  IISADMIN           IIS Admin Service
Running  IKEEXT             IKE and AuthIP IPsec Keying Modules
Running  iphlpsvc           IP Helper
Running  KeyIso             CNG Key Isolation
Running  LanmanServer       Server
Running  LanmanWorkstation  Workstation
Running  lmhosts            TCP/IP NetBIOS Helper
Running  LSM                Local Session Manager
Running  MpsSvc             Windows Firewall
Running  MSDTC              Distributed Transaction Coordinator
Running  MsDtsServer110     SQL Server Integration Services 11.0
Running  MSSQLFDLauncher    SQL Full-text Filter Daemon Launche...
Running  MSSQLSERVER        SQL Server (MSSQLSERVER)
Running  MSSQLServerOLAP... SQL Server Analysis Services (MSSQL...
Running  napagent           Network Access Protection Agent
Running  Netlogon           Netlogon
Running  netprofm           Network List Service
Running  NlaSvc             Network Location Awareness
Running  nsi                Network Store Interface Service
Running  PlugPlay           Plug and Play
Running  PolicyAgent        IPsec Policy Agent
Running  Power              Power
Running  ProfSvc            User Profile Service
Running  ReportServer       SQL Server Reporting Services (MSSQ...
Running  RpcEptMapper       RPC Endpoint Mapper
Running  RpcSs              Remote Procedure Call (RPC)
Running  SamSs              Security Accounts Manager
Running  Schedule           Task Scheduler
Running  SENS               System Event Notification Service
Running  SessionEnv         Remote Desktop Configuration
Running  ShellHWDetection   Shell Hardware Detection
Running  Spooler            Print Spooler
Running  SQLWriter          SQL Server VSS Writer
Running  SSDPSRV            SSDP Discovery
Running  TermService        Remote Desktop Services
Running  Themes             Themes
Running  TrkWks             Distributed Link Tracking Client
Running  UALSVC             User Access Logging Service
Running  UmRdpService       Remote Desktop Services UserMode Po...
Running  upnphost           UPnP Device Host
Running  VMTools            VMware Tools
Running  W32Time            Windows Time
Running  W3SVC              World Wide Web Publishing Service
Running  WAS                Windows Process Activation Service
Running  WinHttpAutoProx... WinHTTP Web Proxy Auto-Discovery Se...
Running  Winmgmt            Windows Management Instrumentation
Running  WinRM              Windows Remote Management (WS-Manag...
Running  wmiApSrv           WMI Performance Adapter
Running  wuauserv           Windows Update
```

Appendix 7 Running Processes Windows VM



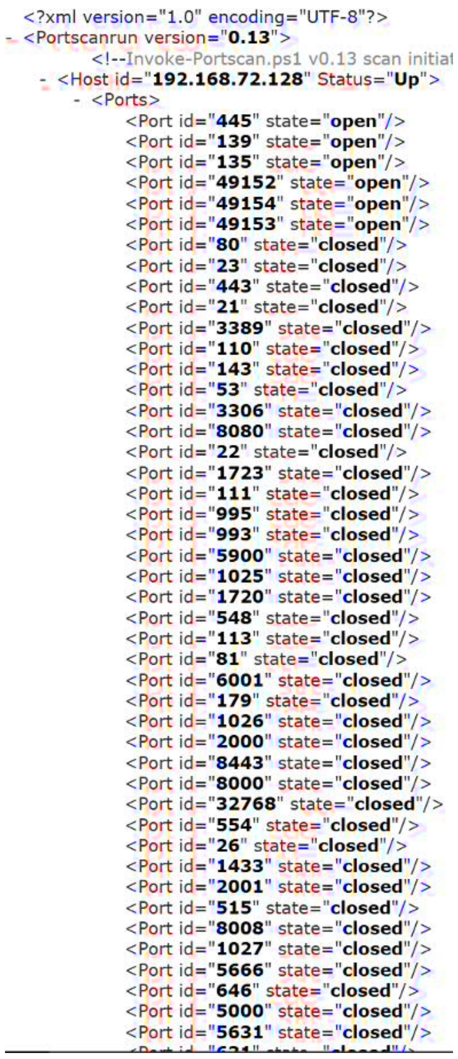
```
Name                 TotalProcessorTime           CPU StartTime
----                 ------------------           --- ---------
MsMpEng              00:27:42.6562500      1662.65625 6/21/2018 4:21:21 PM
System               00:22:39.8750000        1359.875 6/21/2018 4:21:13 PM
svchost              00:04:13.6406250      253.640625 6/21/2018 4:21:20 PM
explorer             00:01:58.4843750      118.484375 6/21/2018 4:24:04 PM
WmiPrvSE             00:01:50.7031250      110.703125 6/21/2018 4:21:27 PM
powershell_ise       00:01:46.6250000         106.625 7/13/2018 12:44:08 AM
powershell_ise       00:01:17.2187500        77.21875 8/2/2018 10:12:35 AM
vmtoolsd             00:01:14.7656250       74.765625 6/21/2018 4:26:01 PM
vmtoolsd             00:01:14.3281250       74.328125 6/21/2018 4:21:21 PM
powershell_ise       00:01:10.3750000          70.375 7/13/2018 12:38:54 AM
svchost              00:00:59.7500000           59.75 6/21/2018 4:21:20 PM
svchost              00:00:53.8125000         53.8125 6/21/2018 4:21:20 PM
svchost              00:00:36.3906250       36.390625 6/21/2018 4:21:20 PM
svchost              00:00:26.1093750       26.109375 6/21/2018 4:21:20 PM
SearchIndexer        00:00:25.0312500        25.03125 6/21/2018 4:23:54 PM
lsass                00:00:24.7187500        24.71875 6/21/2018 4:21:19 PM
NisSrv               00:00:20.8437500        20.84375 6/21/2018 4:21:26 PM
dwm                  00:00:18.2500000           18.25 6/21/2018 4:21:20 PM
svchost              00:00:13.5312500        13.53125 6/21/2018 4:21:26 PM
svchost              00:00:13.1406250       13.140625 6/21/2018 4:21:20 PM
WUDFHost             00:00:08.2343750        8.234375 6/21/2018 4:21:26 PM
svchost              00:00:08.0937500         8.09375 6/21/2018 4:21:19 PM
spoolsv              00:00:07.7656250        7.765625 6/21/2018 4:21:20 PM
services             00:00:07.6250000           7.625 6/21/2018 4:21:19 PM
csrss                00:00:05.3750000           5.375 6/21/2018 4:21:17 PM
winlogon             00:00:04.6875000          4.6875 6/21/2018 4:21:18 PM
notepad              00:00:03.7343750        3.734375 7/13/2018 12:52:10 AM
SnippingTool         00:00:03.2656250        3.265625 8/14/2018 2:43:22 PM
svchost              00:00:03.0625000          3.0625 6/21/2018 4:21:19 PM
smss                 00:00:02.2968750        2.296875 6/21/2018 4:21:13 PM
csrss                00:00:02.2812500         2.28125 6/21/2018 4:21:16 PM
dasHost              00:00:02.2500000            2.25 6/21/2018 4:21:21 PM
WmiPrvSE             00:00:01.7968750        1.796875 8/14/2018 4:30:36 PM
taskhostex           00:00:01.0312500         1.03125 6/21/2018 4:23:24 PM
svchost              00:00:00.5937500         0.59375 6/21/2018 4:21:21 PM
conhost              00:00:00.3281250        0.328125 8/4/2018 12:45:54 PM
wininit              00:00:00.3125000          0.3125 6/21/2018 4:21:17 PM
VGAuthService        00:00:00.1718750        0.171875 6/21/2018 4:21:21 PM
vmacthlp             00:00:00.0781250        0.078125 6/21/2018 4:21:20 PM
cmd                  00:00:00.0468750        0.046875 8/4/2018 12:45:54 PM
msdtc                00:00:00.0312500         0.03125 6/21/2018 4:21:29 PM
WmiApSrv             00:00:00.0312500         0.03125 8/14/2018 4:32:39 PM
SearchProtocolHost   00:00:00.0156250        0.015625 8/14/2018 4:33:18 PM
SearchFilterHost     00:00:00                       0 8/14/2018 4:33:18 PM
Idle
```

Appendix 8 Running Processes Windows Workstation



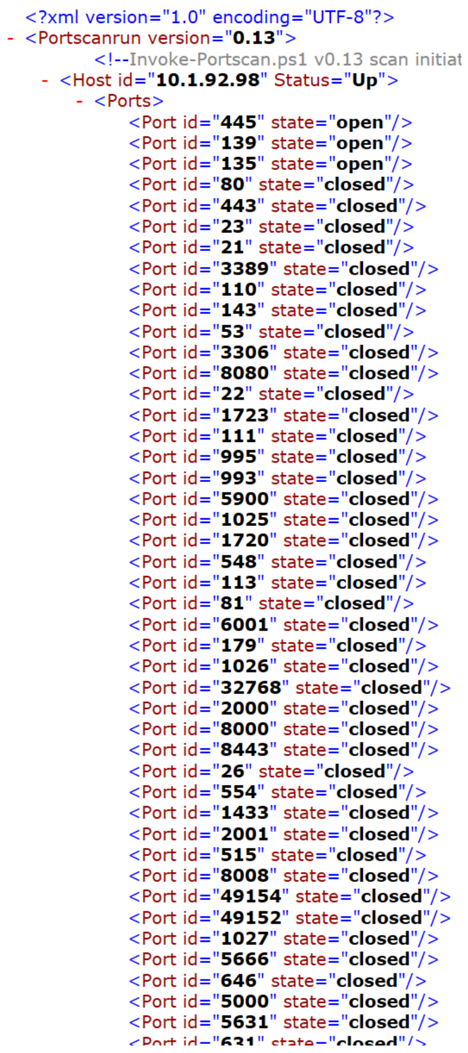
```
Name                    TotalProcessorTime            CPU StartTime
----                    ------------------            --- ---------
chrome                  00:51:21.0468750   3081.046875 11-Aug-18 10:02:53 AM
VirtualBox              00:49:32.2343750   2972.234375 12-Aug-18 3:52:54 PM
vmware-vmx              00:29:42.5000000        1782.5 11-Aug-18 10:41:29 AM
chrome                  00:17:03.1406250   1023.140625 13-Aug-18 8:53:38 AM
WINWORD                 00:10:18.1875000      618.1875 11-Aug-18 11:04:44 AM
explorer                00:07:37.7968750    457.796875 11-Aug-18 10:02:39 AM
vmware-vmx              00:07:37.0468750    457.046875 14-Aug-18 2:31:44 PM
lync                    00:06:49.3906250    409.390625 11-Aug-18 10:02:54 AM
chrome                  00:06:35.4218750    395.421875 13-Aug-18 10:53:25 AM
audiodg                 00:03:15.5312500     195.53125 12-Aug-18 3:46:20 PM
POWERPNT                00:02:27.9062500     147.90625 11-Aug-18 11:45:26 AM
Microsoft.StickyNotes   00:02:09.2343750    129.234375 11-Aug-18 10:02:39 AM
OUTLOOK                 00:01:41.1406250    101.140625 14-Aug-18 3:12:38 PM
ETDCtrl                 00:01:32.5937500      92.59375 11-Aug-18 10:02:37 AM
EXCEL                   00:01:29.2500000         89.25 13-Aug-18 9:39:48 AM
VirtualBox              00:01:16.6250000        76.625 12-Aug-18 3:52:47 PM
RuntimeBroker           00:01:14.2500000         74.25 11-Aug-18 10:02:39 AM
vmware                  00:01:10.8281250     70.828125 11-Aug-18 10:21:26 AM
chrome                  00:01:06.7812500      66.78125 13-Aug-18 11:11:45 AM
Video.UI                00:00:57.0312500      57.03125 12-Aug-18 4:41:34 PM
chrome                  00:00:54.8593750     54.859375 11-Aug-18 10:02:55 AM
chrome                  00:00:50.7343750     50.734375 11-Aug-18 11:01:01 AM
chrome                  00:00:45.7968750     45.796875 14-Aug-18 3:08:44 PM
sihost                  00:00:45.3593750     45.359375 11-Aug-18 10:02:39 AM
IEMonitor               00:00:38.1250000        38.125 11-Aug-18 10:02:54 AM
IDMan                   00:00:37.3750000        37.375 11-Aug-18 10:02:53 AM
explorer                00:00:32.6718750     32.671875 12-Aug-18 3:45:44 PM
ShellExperienceHost     00:00:27.0781250     27.078125 11-Aug-18 10:02:39 AM
nvcontainer             00:00:25.7812500      25.78125 11-Aug-18 10:02:37 AM
powershell_ise          00:00:24.8125000       24.8125 14-Aug-18 2:41:18 PM
UcMapi                  00:00:21.8281250     21.828125 11-Aug-18 10:04:48 AM
chrome                  00:00:20.9218750     20.921875 14-Aug-18 3:08:58 PM
chrome                  00:00:20.0156250     20.015625 14-Aug-18 3:09:06 PM
chrome                  00:00:19.7968750     19.796875 14-Aug-18 10:32:20 AM
chrome                  00:00:19.0937500      19.09375 14-Aug-18 3:04:32 PM
mstsc                   00:00:15.7968750     15.796875 14-Aug-18 2:43:51 PM
igfxEM                  00:00:15.1406250     15.140625 11-Aug-18 10:02:39 AM
SnippingTool            00:00:13.9218750     13.921875 14-Aug-18 2:42:22 PM
chrome                  00:00:12.4062500      12.40625 11-Aug-18 10:02:55 AM
chrome                  00:00:11.4843750     11.484375 11-Aug-18 10:38:47 AM
chrome                  00:00:11.4218750     11.421875 14-Aug-18 3:03:26 PM
chrome                  00:00:09.3906250      9.390625 11-Aug-18 10:28:38 AM
VBoxSVC                 00:00:09.2812500       9.28125 12-Aug-18 3:52:47 PM
SkypeHost               00:00:07.5468750      7.546875 11-Aug-18 10:02:40 AM
chrome                  00:00:07.4375000        7.4375 11-Aug-18 10:19:15 AM
taskhostw               00:00:06.9375000        6.9375 11-Aug-18 10:02:39 AM
OneDrive                00:00:05.7968750      5.796875 13-Aug-18 8:27:46 AM
ApplicationFrameHost    00:00:05.1250000         5.125 11-Aug-18 10:33:03 AM
SearchUI                00:00:05.0937500       5.09375 14-Aug-18 9:54:03 AM
chrome                  00:00:04.5468750      4.546875 11-Aug-18 10:18:14 AM
svchost                 00:00:04.4843750      4.484375 11-Aug-18 10:02:39 AM
NetworkManager          00:00:03.7187500       3.71875 11-Aug-18 10:02:57 AM
chrome                  00:00:03.3437500       3.34375 11-Aug-18 10:23:24 AM
Microsoft.Photos        00:00:03.2968750      3.296875 13-Aug-18 12:14:45 PM
chrome                  00:00:02.9687500       2.96875 14-Aug-18 2:03:46 PM
jucheck                 00:00:02.4062500       2.40625 11-Aug-18 10:07:59 AM
jusched                 00:00:02.3125000        2.3125 11-Aug-18 10:02:58 AM
chrome                  00:00:01.9062500       1.90625 11-Aug-18 10:02:54 AM
chrome                  00:00:01.7500000          1.75 11-Aug-18 10:02:55 AM
chrome                  00:00:01.7343750      1.734375 11-Aug-18 11:24:48 AM
chrome                  00:00:01.7187500       1.71875 14-Aug-18 10:32:16 AM
taskhostw               00:00:01.6875000        1.6875 11-Aug-18 11:09:45 AM
SCNotification          00:00:01.6093750      1.609375 11-Aug-18 10:03:51 AM
RemindersServer         00:00:01.3281250      1.328125 11-Aug-18 11:01:49 AM
dllhost                 00:00:01.1093750      1.109375 11-Aug-18 10:03:51 AM
RAVCpl64                00:00:01.0937500       1.09375 11-Aug-18 10:02:51 AM
chrome                  00:00:00.8437500       0.84375 14-Aug-18 2:03:50 PM
SystemTray              00:00:00.8437500       0.84375 11-Aug-18 10:02:59 AM
GBOSDV2                 00:00:00.7187500       0.71875 11-Aug-18 10:02:57 AM
SettingSyncHost         00:00:00.7031250      0.703125 11-Aug-18 10:03:44 AM
chrome                  00:00:00.6875000        0.6875 14-Aug-18 3:04:37 PM
chrome                  00:00:00.6718750      0.671875 14-Aug-18 2:05:12 PM
nvtray                  00:00:00.4375000        0.4375 11-Aug-18 10:02:43 AM
vmware-unity-helper     00:00:00.4218750      0.421875 11-Aug-18 10:21:30 AM
chrome                  00:00:00.3906250      0.390625 14-Aug-18 2:05:12 PM
MSASCuiL                00:00:00.3593750      0.359375 11-Aug-18 10:02:50 AM
chrome                  00:00:00.2812500       0.28125 14-Aug-18 3:04:40 PM
```

Appendix 9 Open Ports Windows VM



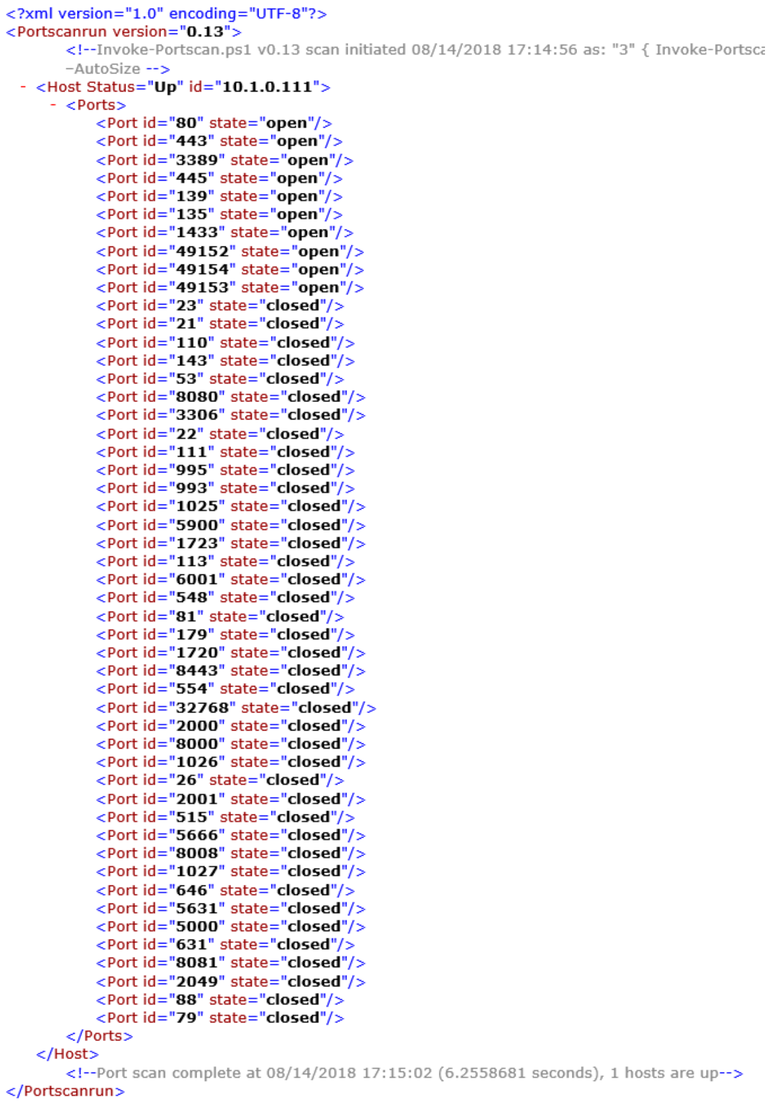
```
  <?xml version="1.0" encoding="UTF-8"?>
- <Portscanrun version="0.13">
        <!--Invoke-Portscan.ps1 v0.13 scan initiat
  - <Host id="192.168.72.128" Status="Up">
    - <Ports>
          <Port id="445" state="open"/>
          <Port id="139" state="open"/>
          <Port id="135" state="open"/>
          <Port id="49152" state="open"/>
          <Port id="49154" state="open"/>
          <Port id="49153" state="open"/>
          <Port id="80" state="closed"/>
          <Port id="23" state="closed"/>
          <Port id="443" state="closed"/>
          <Port id="21" state="closed"/>
          <Port id="3389" state="closed"/>
          <Port id="110" state="closed"/>
          <Port id="143" state="closed"/>
          <Port id="53" state="closed"/>
          <Port id="3306" state="closed"/>
          <Port id="8080" state="closed"/>
          <Port id="22" state="closed"/>
          <Port id="1723" state="closed"/>
          <Port id="111" state="closed"/>
          <Port id="995" state="closed"/>
          <Port id="993" state="closed"/>
          <Port id="5900" state="closed"/>
          <Port id="1025" state="closed"/>
          <Port id="1720" state="closed"/>
          <Port id="548" state="closed"/>
          <Port id="113" state="closed"/>
          <Port id="81" state="closed"/>
          <Port id="6001" state="closed"/>
          <Port id="179" state="closed"/>
          <Port id="1026" state="closed"/>
          <Port id="2000" state="closed"/>
          <Port id="8443" state="closed"/>
          <Port id="8000" state="closed"/>
          <Port id="32768" state="closed"/>
          <Port id="554" state="closed"/>
          <Port id="26" state="closed"/>
          <Port id="1433" state="closed"/>
          <Port id="2001" state="closed"/>
          <Port id="515" state="closed"/>
          <Port id="8008" state="closed"/>
          <Port id="1027" state="closed"/>
          <Port id="5666" state="closed"/>
          <Port id="646" state="closed"/>
          <Port id="5000" state="closed"/>
          <Port id="5631" state="closed"/>
```

Appendix 10 Windows Workstation



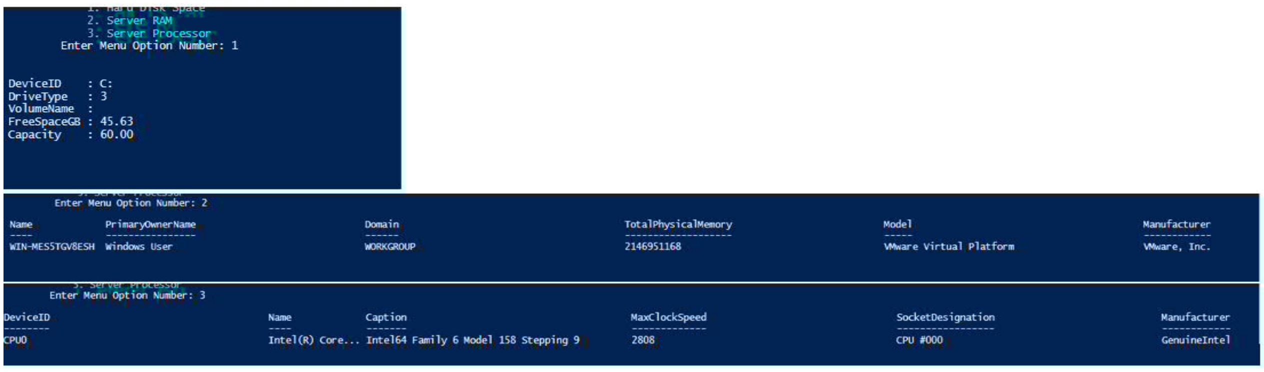
```
  <?xml version="1.0" encoding="UTF-8"?>
- <Portscanrun version="0.13">
        <!--Invoke-Portscan.ps1 v0.13 scan initiat
  - <Host id="10.1.92.98" Status="Up">
    - <Ports>
          <Port id="445" state="open"/>
          <Port id="139" state="open"/>
          <Port id="135" state="open"/>
          <Port id="80" state="closed"/>
          <Port id="443" state="closed"/>
          <Port id="23" state="closed"/>
          <Port id="21" state="closed"/>
          <Port id="3389" state="closed"/>
          <Port id="110" state="closed"/>
          <Port id="143" state="closed"/>
          <Port id="53" state="closed"/>
          <Port id="3306" state="closed"/>
          <Port id="8080" state="closed"/>
          <Port id="22" state="closed"/>
          <Port id="1723" state="closed"/>
          <Port id="111" state="closed"/>
          <Port id="995" state="closed"/>
          <Port id="993" state="closed"/>
          <Port id="5900" state="closed"/>
          <Port id="1025" state="closed"/>
          <Port id="1720" state="closed"/>
          <Port id="548" state="closed"/>
          <Port id="113" state="closed"/>
          <Port id="81" state="closed"/>
          <Port id="6001" state="closed"/>
          <Port id="179" state="closed"/>
          <Port id="1026" state="closed"/>
          <Port id="32768" state="closed"/>
          <Port id="2000" state="closed"/>
          <Port id="8000" state="closed"/>
          <Port id="8443" state="closed"/>
          <Port id="26" state="closed"/>
          <Port id="554" state="closed"/>
          <Port id="1433" state="closed"/>
          <Port id="2001" state="closed"/>
          <Port id="515" state="closed"/>
          <Port id="8008" state="closed"/>
          <Port id="49154" state="closed"/>
          <Port id="49152" state="closed"/>
          <Port id="1027" state="closed"/>
          <Port id="5666" state="closed"/>
          <Port id="646" state="closed"/>
          <Port id="5000" state="closed"/>
          <Port id="5631" state="closed"/>
          <Port id="631" state="closed"/>
```

Appendix 11 Open Ports Production Server



```
<?xml version="1.0" encoding="UTF-8"?>
<Portscanrun version="0.13">
      <!--Invoke-Portscan.ps1 v0.13 scan initiated 08/14/2018 17:14:56 as: "3" { Invoke-Portsca
      -AutoSize -->
  - <Host Status="Up" id="10.1.0.111">
    - <Ports>
          <Port id="80" state="open"/>
          <Port id="443" state="open"/>
          <Port id="3389" state="open"/>
          <Port id="445" state="open"/>
          <Port id="139" state="open"/>
          <Port id="135" state="open"/>
          <Port id="1433" state="open"/>
          <Port id="49152" state="open"/>
          <Port id="49154" state="open"/>
          <Port id="49153" state="open"/>
          <Port id="23" state="closed"/>
          <Port id="21" state="closed"/>
          <Port id="110" state="closed"/>
          <Port id="143" state="closed"/>
          <Port id="53" state="closed"/>
          <Port id="8080" state="closed"/>
          <Port id="3306" state="closed"/>
          <Port id="22" state="closed"/>
          <Port id="111" state="closed"/>
          <Port id="995" state="closed"/>
          <Port id="993" state="closed"/>
          <Port id="1025" state="closed"/>
          <Port id="5900" state="closed"/>
          <Port id="1723" state="closed"/>
          <Port id="113" state="closed"/>
          <Port id="6001" state="closed"/>
          <Port id="548" state="closed"/>
          <Port id="81" state="closed"/>
          <Port id="179" state="closed"/>
          <Port id="1720" state="closed"/>
          <Port id="8443" state="closed"/>
          <Port id="554" state="closed"/>
          <Port id="32768" state="closed"/>
          <Port id="2000" state="closed"/>
          <Port id="8000" state="closed"/>
          <Port id="1026" state="closed"/>
          <Port id="26" state="closed"/>
          <Port id="2001" state="closed"/>
          <Port id="515" state="closed"/>
          <Port id="5666" state="closed"/>
          <Port id="8008" state="closed"/>
          <Port id="1027" state="closed"/>
          <Port id="646" state="closed"/>
          <Port id="5631" state="closed"/>
          <Port id="5000" state="closed"/>
          <Port id="631" state="closed"/>
          <Port id="8081" state="closed"/>
          <Port id="2049" state="closed"/>
          <Port id="88" state="closed"/>
          <Port id="79" state="closed"/>
      </Ports>
    </Host>
      <!--Port scan complete at 08/14/2018 17:15:02 (6.2558681 seconds), 1 hosts are up-->
</Portscanrun>
```

Appendix 12 Hardware Properties Windows Virtual Machine

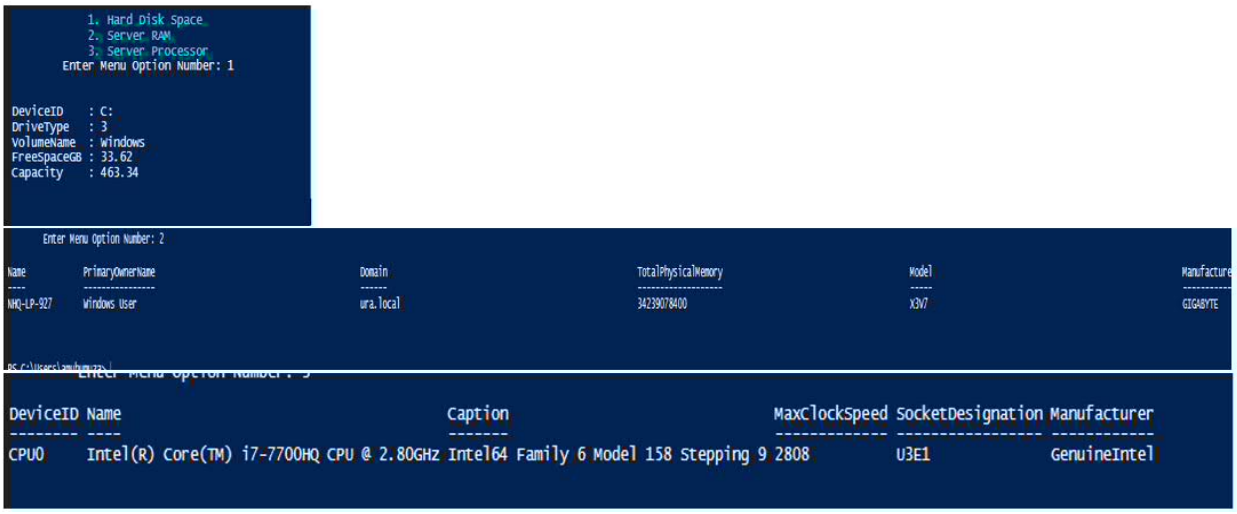


```
             1. Hard Disk Space
             2. Server RAM
             3. Server Processor
         Enter Menu Option Number: 1


DeviceID    : C:
DriveType   : 3
VolumeName  :
FreeSpaceGB : 45.63
Capacity    : 60.00
```

```
             3. Server Processor
        Enter Menu Option Number: 2

Name             PrimaryOwnerName                                Domain                                          TotalPhysicalMemory                             Model                                           Manufacturer
----             ----------------                                ------                                          -------------------                             -----                                           ------------
WIN-MES5TGV8ESH  Windows User                                    WORKGROUP                                       2146951168                                      VMware Virtual Platform                         VMware, Inc.
```

```
             3. Server Processor
        Enter Menu Option Number: 3

DeviceID                                         Name            Caption                                         MaxClockSpeed                                   SocketDesignation                               Manufacturer
--------                                         ----            -------                                         -------------                                   -----------------                               ------------
CPU0                                             Intel(R) Core... Intel64 Family 6 Model 158 Stepping 9          2808                                            CPU #000                                        GenuineIntel
```

Appendix 13 Hardware Properties Windows Workstation

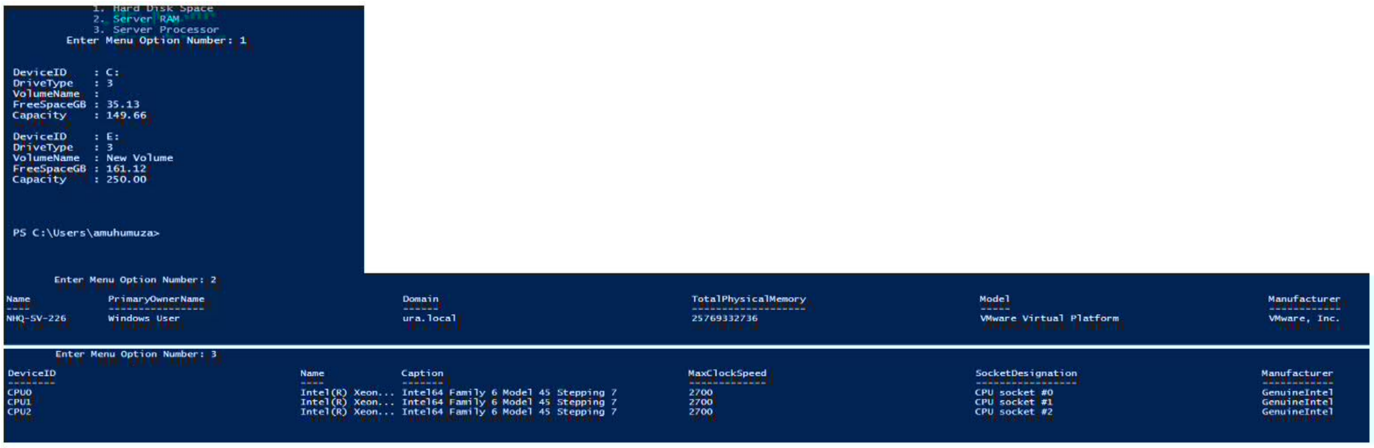


```
            1. Hard Disk Space
            2. Server RAM
            3. Server Processor
        Enter Menu Option Number: 1


DeviceID    : C:
DriveType   : 3
VolumeName  : Windows
FreeSpaceGB : 33.62
Capacity    : 463.34
```

```
      Enter Menu Option Number: 2

Name         PrimaryOwnerName            Domain         TotalPhysicalMemory        Model          Manufacture
----         ----------------            ------         -------------------        -----          -----------
NHQ-LP-927   Windows User                ura.local      34239078400                X3V7           GIGABYTE


PS C:\Users\amuhumuza>
```

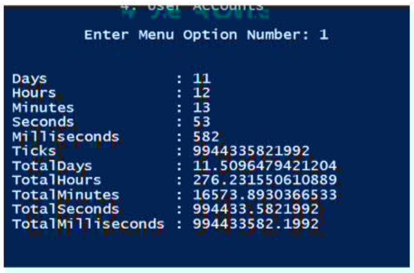
```
        Enter Menu Option Number: 3

DeviceID Name                                     Caption                                MaxClockSpeed SocketDesignation Manufacturer
-------- ----                                     -------                                ------------- ----------------- ------------
CPU0     Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz Intel64 Family 6 Model 158 Stepping 9 2808          U3E1              GenuineIntel
```

Appendix 14 Hardware Properties Production Server

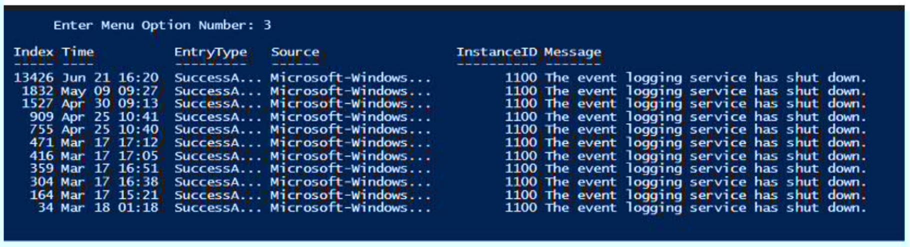


```
            1. Hard Disk Space
            2. Server RAM
            3. Server Processor
        Enter Menu Option Number: 1


DeviceID    : C:
DriveType   : 3
VolumeName  :
FreeSpaceGB : 35.13
Capacity    : 149.66

DeviceID    : E:
DriveType   : 3
VolumeName  : New Volume
FreeSpaceGB : 161.12
Capacity    : 250.00



PS C:\Users\amuhumuza>



      Enter Menu Option Number: 2

Name          PrimaryOwnerName          Domain          TotalPhysicalMemory          Model                    Manufacturer
----          ----------------          ------          -------------------          -----                    ------------
NHQ-SV-226    Windows User              ura.local       25769332736                  VMware Virtual Platform  VMware, Inc.

      Enter Menu Option Number: 3

DeviceID                 Name           Caption                             MaxClockSpeed      SocketDesignation     Manufacturer
--------                 ----           -------                             -------------      -----------------     ------------
CPU0                     Intel(R) Xeon... Intel64 Family 6 Model 45 Stepping 7  2700            CPU socket #0         GenuineIntel
CPU1                     Intel(R) Xeon... Intel64 Family 6 Model 45 Stepping 7  2700            CPU socket #1         GenuineIntel
CPU2                     Intel(R) Xeon... Intel64 Family 6 Model 45 Stepping 7  2700            CPU socket #2         GenuineIntel
```

Appendix 15 Boot-Up Details Windows Virtual Machine

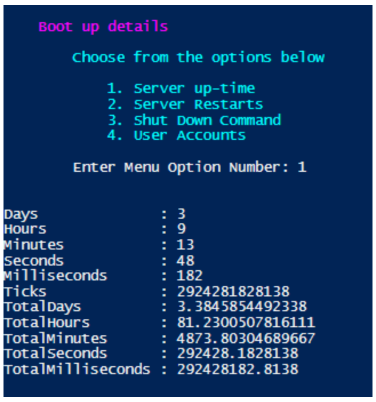


```
        Enter Menu Option Number: 1


Days              : 11
Hours             : 12
Minutes           : 13
Seconds           : 53
Milliseconds      : 582
Ticks             : 9944335821992
TotalDays         : 11.5096479421204
TotalHours        : 276.231550610889
TotalMinutes      : 16573.8930366533
TotalSeconds      : 994433.5821992
TotalMilliseconds : 994433582.1992
```



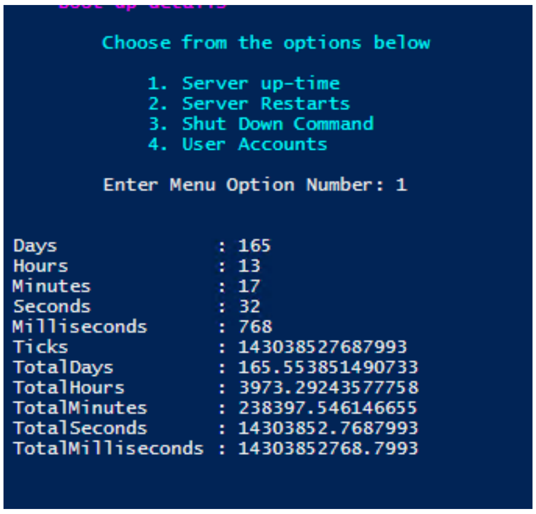
```
      Enter Menu Option Number: 3

Index Time          EntryType   Source                 InstanceID Message
----- ----          ---------   ------                 ---------- -------
13426 Jun 21 16:20  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
 1832 May 09 09:27  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
 1527 Apr 30 09:13  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  909 Apr 25 10:41  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  755 Apr 25 10:40  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  471 Mar 17 17:12  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  416 Mar 17 17:05  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  359 Mar 17 16:51  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  304 Mar 17 16:38  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
  164 Mar 17 15:21  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
   34 Mar 18 01:18  SuccessA... Microsoft-Windows...         1100 The event logging service has shut down.
```

Appendix 16 Boot-Up Details Windows Workstation



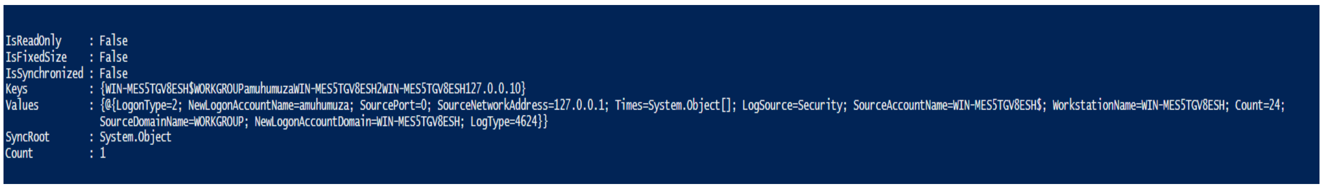
```
    Boot up details

        Choose from the options below

            1. Server up-time
            2. Server Restarts
            3. Shut Down Command
            4. User Accounts

        Enter Menu Option Number: 1


Days              : 3
Hours             : 9
Minutes           : 13
Seconds           : 48
Milliseconds      : 182
Ticks             : 2924281828138
TotalDays         : 3.3845854492338
TotalHours        : 81.2300507816111
TotalMinutes      : 4873.80304689667
TotalSeconds      : 292428.1828138
TotalMilliseconds : 292428182.8138
```

Appendix 17 Boot-Up Details Productions Server



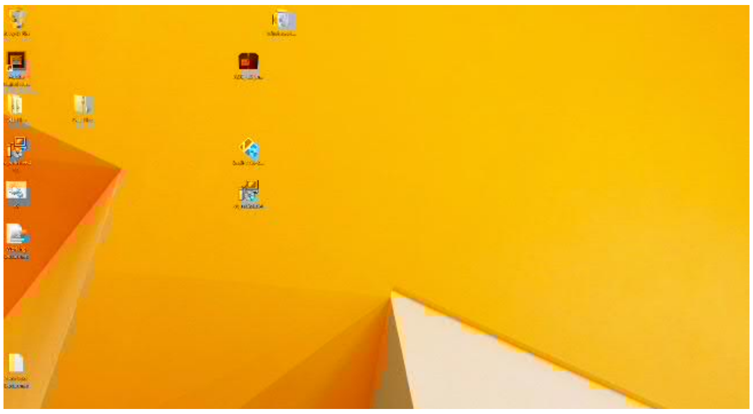
```
        Choose from the options below

            1. Server up-time
            2. Server Restarts
            3. Shut Down Command
            4. User Accounts

        Enter Menu Option Number: 1


Days              : 165
Hours             : 13
Minutes           : 17
Seconds           : 32
Milliseconds      : 768
Ticks             : 143038527687993
TotalDays         : 165.553851490733
TotalHours        : 3973.29243577758
TotalMinutes      : 238397.546146655
TotalSeconds      : 14303852.7687993
TotalMilliseconds : 14303852768.7993
```

Appendix 18 Login Details Windows Virtual Machine

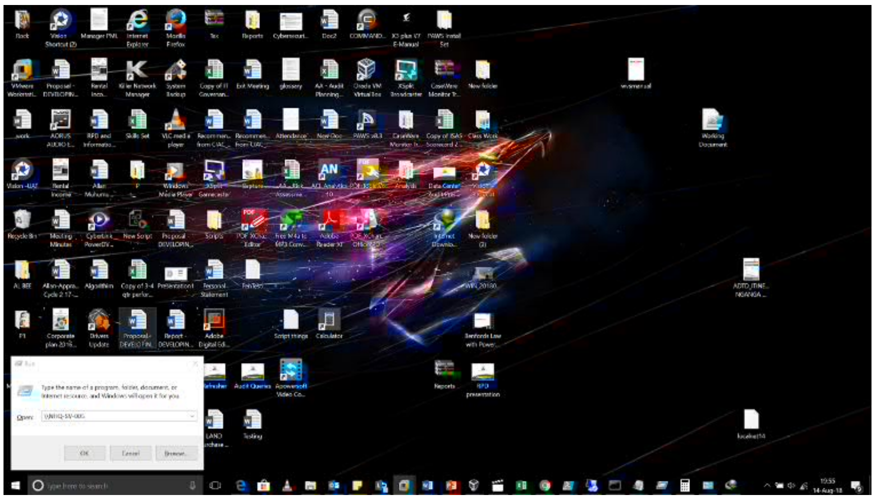


```
IsReadOnly     : False
IsFixedSize    : False
IsSynchronized : False
Keys           : {WIN-MES5TGV8ESH$WORKGROUPamuhumuzaWIN-MES5TGV8ESH2WIN-MES5TGV8ESH127.0.0.10}
Values         : {@{LogonType=2; NewLogonAccountName=amuhumuza; SourcePort=0; SourceNetworkAddress=127.0.0.1; Times=System.Object[]; LogSource=Security; SourceAccountName=WIN-MES5TGV8ESH$; WorkstationName=WIN-MES5TGV8ESH; Count=24;
                 SourceDomainName=WORKGROUP; NewLogonAccountDomain=WIN-MES5TGV8ESH; LogType=4624}}
SyncRoot       : System.Object
Count          : 1
```

Appendix 19 Screenshot Capture Windows Virtual Machine



Appendix 20 Screenshot Capture Windows workstation



Appendix 21 Screenshot Capture Production Server