YAŞAR UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

MASTER/PHD THESIS

SOFTWARE DEFINED IMPLEMENTATION OF CYBER ATTACK DETECTION AND PREVENTION

MERT CAN KILIC

THESIS ADVISOR: ASST. PROF. IBRAHIM ZINCIR

COMPUTER ENGINEERING

PRESENTATION DATE: 10.11.2017

BORNOVA / İZMİR
December 2017

We certify that, as the jury, we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Jury Members: **Signature:**

Asst. Prof. Dr. İbrahim ZİNCİR
Yaşar University

Asst. Prof. Dr. Samsun M. BAŞARICI
Adnan Menderes University

Assoc. Prof. Dr. M. Süleyman ÜNLÜTÜRK
Yaşar University

Prof. Dr. Cüneyt GÜZELİŞ
Director of the Graduate School

ABSTRACT

SOFTWARE DEFINED IMPLEMENTATION OF CYBER ATTACK DETECTION AND PREVENTION

Kilic, Mert Can
Msc, Computer Engineering
Advisor: Assist.Prof. Ibrahim Zincir, Ph.D.

December 2017

Computer networks and computational communication technologies have been improving very fast since the first connection was established between two computers by ARPANET in 1969. The daily routines are becoming digitalized day by day. This transformation provides easiness, but at the same time it causes some security problems. The security mechanisms such as authentication, authorization and recognition that a human brain can automatically execute, can be manipulated in digital environments. The people who have the motivation for stealing information, profiting in illegal ways, blackmailing and so on, use a lot of manipulative methods by making use of computer networks and the systems that are based on these networks. These methods are changing and being updated very rapidly, so it is very difficult to detect and prevent that kind of attacks. Even the new generation tools that have current electronic control mechanisms can be exposed to that kind of attacks, so that it is known that this may cause crucial destructions including death.

The security experts who provide service for defending systems against these complicated and sophisticated attacks, may be unaware and uninformed about the security flaws that are being used by the people who have the criminal

motivations. The penetration tests that are being conducted periodically, are mostly for the revealed security flaws. Namely, the security flows are updated more frequently than the penetration tests.

The systems that are not maintained or operated by the qualified security experts are very open to the old-fashioned attacks, and these poorly maintained systems are avoiding the costs of the sophisticated detection and prevention software.

The main goal of this work is to use a x86 based embedded system which hosts a customized Linux based operating system with the dynamic analysis of the both remotely and locally gathered/enumerated logs as well as implementing network security functionalities of the conventional network equipments provide. Thus allowing to gather and analyze information about the local or remote network resulting automated reporting for the IT administrators.

ÖZ

SIBER SALDIRI TESPIT ETME VE ONLEME YAZILIM UYGULAMASI

Kilic, Mert Can

Yüksek Lisans Tezi, Bilgisayar Mühendisliği

Danışman: Yrd.Doç.Dr. Ibrahim Zincir

Aralik 2017

Bilgisayar ağları ve sayısal haberleşme teknolojileri, ilk bilgisayarlar arası bağlantının 1969 yılında ARPANET ile başlamasından bu yana çok hızlı bir şekilde gelişmeye devam etmekte. Bir kaç yıl önce toplumun hayatında karşılaştığı veya bazı rutin işleri halledebilmek adına izlediği genel işler günbegün değişmekte ve farklı formlara bürünerek hayatları sayısal ortama taşımaktadır. Bu gelişme veya dönüşüm beraberinde kolaylıkların yanı sıra bir çok güvenlik problemi getirmektedir. Fiziksel dünyada insan beyni, diğer duyu organlarından aldığı ve işlediği görüntü, ses vb girdilerle doğrulama, hatırlama ve yetkilendirme mekanizmalarını kullanmaktayken, bu mekanizmaların yanıltılabileceği ve manipüle edilebileceği sayısal ortamda benzeri doğrulamaları sağlamak güçleşmekte. Finansal çıkar, bilgi çalmak, şantaj yapmak ve benzeri motivasyonu olan kimseler, bu tarz manipülatif yöntemleri hem bilişim ağlarında hem de bu ağlar üzerinden sağlanan hizmetlere uyarlamak için çalışmaktalar. Fark edilme sürecine kadar yeni sürümleri geliştirilen karmaşık saldırıları tespit etmek çok zorlu bir işlem olduğu gibi, daha sınırları belli olmayan muhtemel saldırılara karşı önlem almak, sonsuz büyüklükteki bir olasılık kümesindeki tüm çıktılara karşı genel geçer bir yöntem bulmak kadar zordur. Güncel ve elektronik kontrol mekanizmasına sahip yeni nesil bir araç dahi bu tarz saldırılara maruz kalabildiği gibi sonucu ölüme varan büyük yıkımlara sebebiyet verebileceği bilinmektedir.

Bu denli karışık ve kademeli saldırılara karşı profesyonel olarak destek veren güvenlik uzmanları, ana gayesi kriminal amaçlar veya haksız kazanç sağlamak olan kimselerin güncel olarak kullandığı ve istismar ettiği güvenlik zaafiyetlerine karşı habersiz ve bilgisiz olabilmektedir. Belirli aralıklarla uygulanan penetrasyon testleri, çoğunlukla kullanılması bırakılmış veya ifşa olmuş saldırı vektörlerine karşı önlem alma amacıyla yapılmaktadır. Günlük mertebede güncellenen bu saldırı vektörlerinin hedefinde bir şirket veya kuruluşun yer alması, bir sonraki olağan zaafiyet testine kadar güvenli olarak kabul edilmesi algısını ortadan kaldırmaktadır. Bu güvenlik zaafiyetlerinin büyük hasarlar verdiği bir çok örnek ve haber çıkmasına rağmen aylar sonra dahi hala aynı zaafiyeti taşıyan sistem ve ağlar bulunabilmektedir. Özellikle bu saldırı tekniğinin sahibi bilgisayar korsanları tarafından paylaşılması üzerine çok daha az teknik bilgiye sahip kimseler, basitçe aynı saldırıyı kendi iç bilgiye sahip oldukları daha ufak ve zaafiyete sahip sistemlere yüksek başarı oranıyla uygulayabilmektedir. Bünyesinde yeterli nitelikte güvenlik uzmanı bulundurmayan sistemler, bir çok geçmiş saldırıya açık kaldığı gibi, sofistike güvenlik cihazlarının işletimsel ve güncelleme maliyetlerinden kaçınmaktadırlar.

Bu tezin amacı, özelleştirilmiş tek bir x86 tabanlı gömülü sistem üzerine, özel derlenmiş ve yazılımsal işlevsellikler eklenmiş bir Linux tabanlı işletim sistemi kurarak, otonom ve kompleks ilişkilendirmeler kurabilen bir çözümü denemektir. Geleneksel tüm ağ ve güvenlik işlemlerinin yazılımsal ve işletim sistemi katmanında kontrol edildiği bu çözümde aynı zamanda savunma odaklı ve katı bir güvenlik algısından yana saldırgan ve dış ağa bilinçli olarak zayıf gösterilen sistemler sayesinde olası saldırganları tespit etme ve bilgi toplama işlemleri yapılmaktadır.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to my advisor Ibrahim Zincir for all his support and patience during this research project. He has been guiding me towards the goals that I am willing to achieve. His encouragment and his faith in this research always inspired and motivated me to pursue even in the most challenging parts of my projects.

Special thanks to Graeme Hanssen for his mentorship and his enormous design experience that definetly helped me a lot to overcome many challenges that I have faced throughout my journey. He spent his time, effort and assets both personally and his GO Mechatronics Company.

I would like to thank Yasar University Scientific Research Project comittee those who not only supported my financial and infrastructrual needs, also provided me with great feedback and brainstorming during the hardware board development phase.

Many thanks to Minerva Business Incubation Center for providing me my infrastructrual needs in order to conduct my research and experiments as well as introducing me with many business professionals allowing me to get very valuable feedback. I would also like to thank Gediz Electric Inc. for allowing me to test my research on a live infrastructure as well as providing professional feedback that helped me to improve some functionalities.

I thank my parents who always supported and motivated me for my goals. They were always eager to ease out the overwhelming effort that I have been putting.

My friends and colleagues, Burak Kaymaz, Cenk Ozdag, Mecit Selek and Ilter Yuksel played a huge part in my research with their invaluable contributions, with their amazing ideas. They never gave up or rejected my brainstorming requests. I would like to specially thank Gamze Orhon for her guidance and amazing support during my work.

Many thanks to Offensive Security Inc., creators of Kali-Linux. Their ”OffSec Philosophy” and ”Try Harder” motto always gave me strength and their vision allowed me to improvise new approaches towards my goals.

Last but not least, I thank to all anonymous open-source software developers and their great projects that allowed me to improve my methodologies and techniques. Those who are free in the mind, as in free-software. Their modesty is a fine example for those who still believe in individualism and greed.

Mert Can Kilic
İzmir, 2017

TEXT OF OATH

I declare and honestly confirm that my study, titled "SOFTWARE DEFINED IMPLEMENTATION OF CYBER ATTACK DETECTION AND PREVENTION" and presented as a Master's Thesis, has been written without applying to any assistance inconsistent with scientific ethics and traditions. I declare, to the best of my knowledge and belief, that all content and ideas drawn directly or indirectly from external sources are indicated in the text and listed in the list of references.

Mert Can Kilic

Izmir, 2017

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Computer networks and digital communication technologies has been advancing rapidly ever since the first computer-to-computer link was established by the ARPANET (Advanced Research Projects Agency Network)(Lukasik, 2010) back in 1969. This advancement has been so fast throughout the years, it is nearly impossible to avoid or ignore interaction with any form of inter-connected device during daily lives. post-offices, local stores and even banks evolved into digitally available and accessible services regardless of the Geo-location on Earth. Conventional daily chores, duties and even jobs are and has been transforming into an ”E-” form. Nowadays finding new friends, enrolling a new course or even sports are transformed into E-Friends, E-Course and E-Sports. Shopping from a local store considered to be more time consuming than ordering them off of an online store and often called old-fashioned.

Back in days, special memories were kept in private until they are shared with legitimate visitors or friends. Nowadays countless visual and written content and personal information are uploaded to the online world. Precious bank accounts and credit cards details are traveling back and forth on the realm of inter-connected computers. Thus causing major privacy and security breaches daily while easing out the daily lives of individuals and businesses. Regardless of the motivation, malicious activities targeting corporations, individuals and governments are and always been a major issue ever since the Internet became available. This is not only causing massive privacy leaks or unauthorized financial transactions, there has been recorded countless security breach incidents on major companies that some lead to their collapse.

Even though individuals' privacy is constantly under a threat on the Internet, digital security breaches on the corporations and critical infrastructures are effecting masses and their security as well. Conventional cyber-security approaches have never been sufficient enough to stop these malicious activities once and for all since new features are constantly being developed. Every new feature or an implementation may prevent older malicious techniques, but also potentially bears new and undetected attack methodologies and techniques, considered to be a "new challenge" in cyber criminals realm. Solutions like bug bounty programs and internal tests of the developed software are not adequate enough to pin point every possible security flaw on a system, yet most of the testers are conducting these tests from an engineering point of view where malicious attackers are abusing even the smallest unturned stones that are either ignored or never even thought about.

All in all, controversial total attack proof systems and discovered or abused security flaws on a daily basis are highly dynamic and quite unpredictable. Major corporations and many critical infrastructures that have any form of interconnection such as SCADA systems, are being constantly monitored and patched for recently discovered security flaws by their security experts. On the other hand small to medium enterprises are the ones that heavily rely on conventional security equipment on their networks such as Firewalls, Mitigators, Spam Filters which are proven to be inadequate in this era where nearly everything is online.

1.1 PROBLEM CONTEXT

Identification procedure of the possible security or implementation flaws on a network is called Penetration testing. These tests are both performed by internal IT experts as well as qualified cyber security experts on monthly, quarterly or yearly basis. Despite the fact that these tests are either overwhelmingly detailed or in a quick regular check form, security experts are aware of that these tests are designed to prevent well known attacks or to improve previous implementations that might cause security flaws. Constantly found brand new attack vectors also known as Zero Days are the biggest threat to nearly all digital entities. Since its impossible to foresee where and when the next zero-day will emerge, most of these attack vectors are announced publicly after the attackers took advantage of it or their technique is discovered by security experts. Thus also points out clearly that penetration tests and conventional security equipment are not sufficient enough to protect systems that are not monitored constantly (Bellovin, 1989).

IT Security experts, often called White-Hat hackers, are playing an important role in analysis and detection of the system logs populated by the network devices. Most malicious attack techniques are based on manipulation of the legitimate connection or authorized actions. Conventional network security devices are not capable of making predictions or analyzing obfuscated data leaks but they can be considered as just regulators on a system that are enforcing predefined rules depending on the setup. IT Security experts collect and correlate different information from various sources in order to prevent or identify possible attacks and the potential owner of the on-going or prior malicious attack.

Because of the IT security experts are not sharing the same knowledge background, their professional precautions and methodologies also vary. Especially against the malicious attackers who are only concerned about to find a way into their target system with their extremely sophisticated techniques, it is nearly impossible to protect a system without knowing where the next flaw will emerge (Portokalidis, Slowinska, & Bos, 2006).

1.2 THESIS STATEMENT

Rapidly increasing computational and functional needs for conventional security entities, such as Firewall, Mitigator, Sandbox and IPS/IDS, Small-Medium enterprises are not only avoiding to use some of these equipment also extending the penetration tests frequency in order to reduce operational and upgrade budgets.

Aim of this work is to overcome some of the problems that are defined previously. By utilizing technologies and concepts such as Network Function Virtualization and Software Defined Networking as well as an Offensive-Security approach is to reduce the hardware and vendor dependency while adding basic penetration testing check lists that are automatically performed and periodically updated for the new security flaws that are discovered.

Not only maintaining basic network functionalities from a single hardware or performing predefined penetration tests, more offensive procedures are aimed to be pursued in order to achieve high-availability of the services that are behind. Offensive approach involves adding new software functionalities for the hardware platform so that it impersonates a malicious attacker both within the local network as well as outbound connections. By the help of this approach, it is possible to prevent possible security flaws that may not been discovered or detected before.

1.3 ROADMAP

In this thesis there are 5 chapters starting with this chapter accompanied by additional appendix where scripts, outputs and other referenced outcomes are located.

Structure is as follows:

- **Chapter 2 - Background** provides some key concepts and technologies that are required to assemble pieces of their corresponding part/parts related with the thesis.

- **Chapter 3 - Implementation** contains the actual step by step process for building a customized Linux distribution as well as the hardware specifications of the base device that is used as a hardware platform in order to meet specified functionalities and needs.

- **Chapter 4 - Testing** includes various test scenarios and their corresponding results.

- **Chapter 5 - Conclusion** embodies the actual usability by comparing security products and solutions that are on the market. As to prove and analyze the outcome of this work, evaluation in technical aspect and in financial point of view is provided within.

CHAPTER 2

TECHNICAL BACKGROUND

This chapter includes background information that is necessary to correlate and explain the approach that is being pursued in the following chapters. Sections are there to explain their basic definition and related roles in this thesis. Some concepts that are located below are not fully covered in detail but after defining their key role to the reader, it's relation with this approach is presented.

2.1 COMPUTER NETWORKS CONCEPTS

Software defined networks, network function virtualization and soft-networking are commonly misunderstood and confuse even professionals today. Many commercial applications that are available out on the market are utilizing more then one of these technologies as a foundation, but still there are not many strict boundaries that differentiates each other because of the emerging and constantly expanding application areas and new features.

There are variety of Software Defined Networking implementations and usage areas that are currently in use on live-Networks such as load-balancers, Virtual Hosts, Traffic generators and many others. But since there are nearly no boundaries of the Software Defined Network implementations, Unified Security Manager can be considered as the parent category and the most simple definition that is given by SDN implementation (Hollabaugh, 2002).

Software Defined Network (SDN)

Software defined networking is the concept that allows administrators to be able to deploy, initialize and program network functionalities as needed in a flexible manner. Legacy devices are not capable of achieving such tasks since they are mostly based on ASICs which is defined in 2.2 with re programmable FPGA[1] or NVRAM[2] (Han, Gopalakrishnan, Ji, & Lee, 2015).

In the scope of this thesis, Software defined networking concept is deployed in the Operating system by regulating network flows and rules within the O/S that is explained in section 3.2.1. By utilizing network interface configuration and manipulation tools that are available for *NIX based operating systems such as `iptables` and `ifconfig`, achieving software defined networking features in operating system shell layer became possible with shell scripting (Williams & Bergmann, 2004).

Virtualization

Virtualization is a concept of utilizing same hardware for sharing multiple operating systems or applications. Virtualization concept emerged from the need of allowing mainframes to run multiple applications simultaneously. Before the virtualization technology, commercial server system utilization considered to be mostly slack operation. Both commercial and open-source solutions allows running multiple operating system simultaneously. In the subsequent parts of this thesis the term "Host" is used to describe actual hardware that runs virtual applications or operating systems. Where as the term "Guest" is to describe virtual application or operating system that runs on a specific host (Pfaff et al., 2009).

Apart from the dedicated conventional network equipments, x86 based generic computing unit is used to utilize different virtual guest operating systems that is used for various applications like sandboxing incoming executable files and for small server instances of SMB/NFS (Joshi & Benson, 2016).

[1] Field Programmable Gate Array

[2] Non-Volitalie Random Access Memory

Network Function Virtualization (NFV)

Network Function Virtualization, known as ”NFV” is the concept that utilizes hardware virtualization technologies and concepts; in order to virtualize network nodes in a system that are capable of connecting simultaneously to any other network node despite the fact that they reside in the same host device or share same network interfaces (Joshi & Benson, 2016). Leading network device manufacturers and service providers are providing licenses for well known embedded O/S and commercially available security solutions that can be deployed in seconds to a generic hardware board. Those embedded O/S was once can only be used within its governed company devices. By the help of network function virtualization, one generic embedded board or a network device can be switched into full-stack security solution within minutes. By eliminating the need to update and develop prior devices, a sandbox device with all necessary peripherals allows manufacturers to only provide licensing and subscription services by allowing customer to pick variety of generic boards for various computational and performance needs (Bugnion, Devine, Rosenblum, Sugerman, & Wang, 2012).

Hardware implementation and technical specifications can be found in appendix section. This generic embedded platform is where network function virtualization is applied. Number of network interface cards, persistent storage unit, central processing unit and random access memory are crucial for determining hardware limitations that can be used for different NFVs (Martins et al., 2014) (Mijumbi et al., 2016) (Han et al., 2015) (Pfaff et al., 2009).

2.2 HARDWARE

Hardware portion of this research aims to establish a standardized system configuration on the host board that will be used to run software implementations that are covered in chapter 3. Regarding this hardware, key concepts and elements are described in the following sections.

Application Specific Integrated Circuit (ASIC)

Application Specific Integrated Circuits also referred as ASICs are designed to accomplish certain tasks rather than general usage. ASICs are designed to achieve a certain task with maximum efficiency (Einspruch, 2012).

Network Interface cards on the hardware board are ASICs. They are to encode and decode the digital transmission in a very strict fashion. Error correction and data de-capsulation is achieved by a processing unit(Lee et al., 2004).

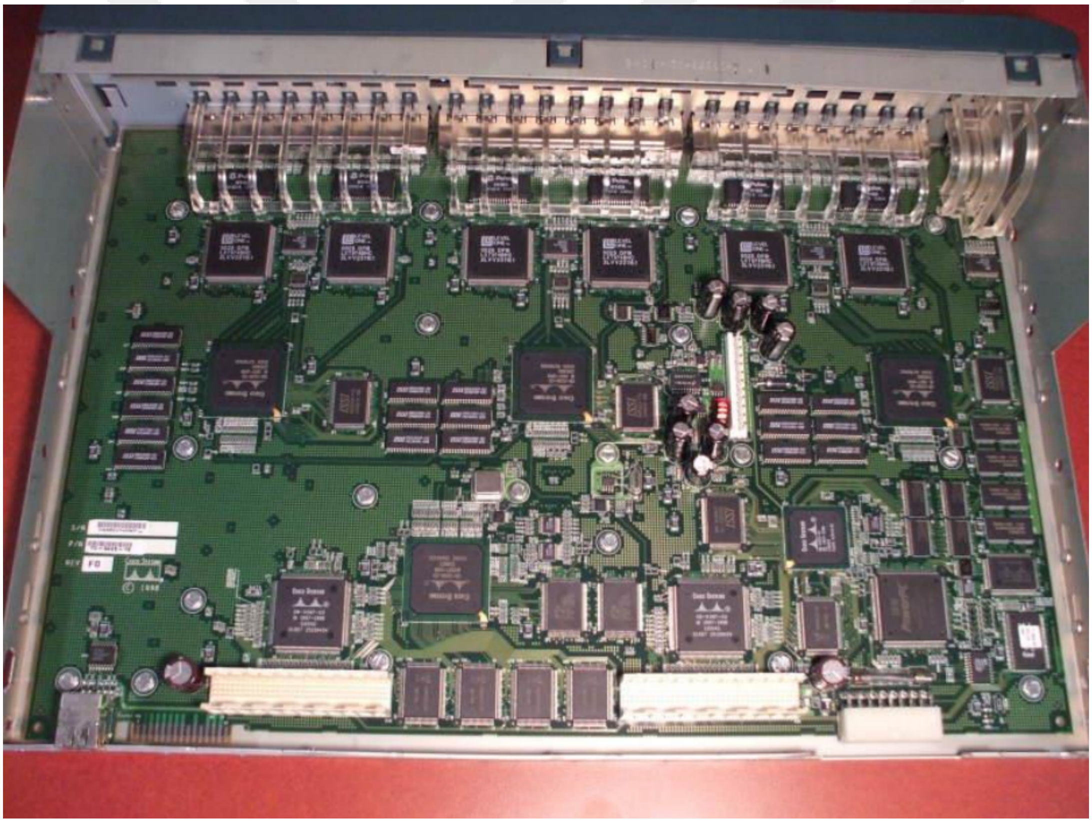


Figure 2.1: ASIC Network Switch's Mainboard

Conventional network devices that are build by ASICs are task specific and capable of performing limited array of actions. By the time that computer central processing units were not capable of achieving virtualization and simultaneous tasks, ASIC boards considered to be the only feasible solution. But in order to achieve aims of this research, general purpose central processing capabilities are required. Regular x86/64 personal computer is a fine example to point out the difference between ASICs and CPU operations.

Field Programmable Gate Array (FPGA)

FPGA consists of various size of inter-connected programmable logic blocks that can be customize in order to achieve certain tasks. These array of logic blocks can be configured by various combinations by the help of hardware description language also known as HDL (Baker, Asami, Deprit, Ousterhout, & Seltzer, 1992).

Non-Volatile Random Access Memory (NVRAM)

Non-Volatile Random Access Memories main feature is that NVRAM retains its contents even after power off or system halt. Some examples are included but not limited to EEPROM and flash memory. Its also known as persistent data medium. (Baker et al., 1992)

Unified Security Management (USM)

Unified Security Management is an overall solution to handle multiple security features such as firewall, Sandbox and IPS/IDS all in one. By the help of network function virtualization USM devices are mostly preferred to reduce MTBF[3] and operational cost of managing different devices in a network (Agham, 2016).

Since the main goal is to provide all in one security solution on a single device with external update and self penetration testing capabilities, Unified Security Management and it's inherited features is the key concept and definition for this work (Ericsson, 2010).

[3] Mean Time Between Failure

2.3 CYBER SECURITY ASPECT

In this section, frequently used and referred cyber security definitions are presented. Categories and the effects of these defined attack are provided. Concepts such as enumeration includes countless techniques as well as their combination with other attack vectors are nearly impossible to predict and will vary for each scenario. More refined attack prevention methodologies and definitions can be located in Chapter 3 (Neuman, 2009) (Liu, Xiao, Li, Liang, & Chen, 2012) (Ramim & Levy, 2006).

Denial Of Service (DoS)

Denial-Of-Service also known as ”DoS” is a cyber attack concept that is based on flooding the server side with illegitimate requests. This allows attackers to disrupt the victim's target services, such as Web Request or API[4] Communications (Senie & Ferguson, 1998).

Denial of service attacks are unpredictable and they vary on the magnitude of the attack. Precautions are based on the monitored data of the given network by analyzing the legitimate network traffic as well as high and low ends of the network. By the help of this boundaries, any extreme connection attempt or excessive drop packages compared to normal values can be considered as malicious and interrupted. But without knowing the extreme values, TCP/IP connections are flagged as safe and legitimate, therefore understanding an incoming denial of service attack can be challenging (Martin, 2008).

Distributed Denial of Service (DDoS)

Distributed Denial of Service often known as DDoS, is another form of Denial of Service attacks that is designed to conduct Denial of Service attack by multiple clients that are distributed over the network or networks in order to flood target

[4] Application Programming Interface

system or service. Main characteristic of DDoS attack is that most of the workers or zombies[5] are distributed over the networks in a fashion that they all have different global IP addresses and different network bandwidths. Thus it becomes much harder for network administrators to differentiate legit connections from malicious connection requests (Batsell, Rao, & Shankar, 2005). Conceptual of a distributed Denial Of Service attack topology is as shown in Figure 2.2.

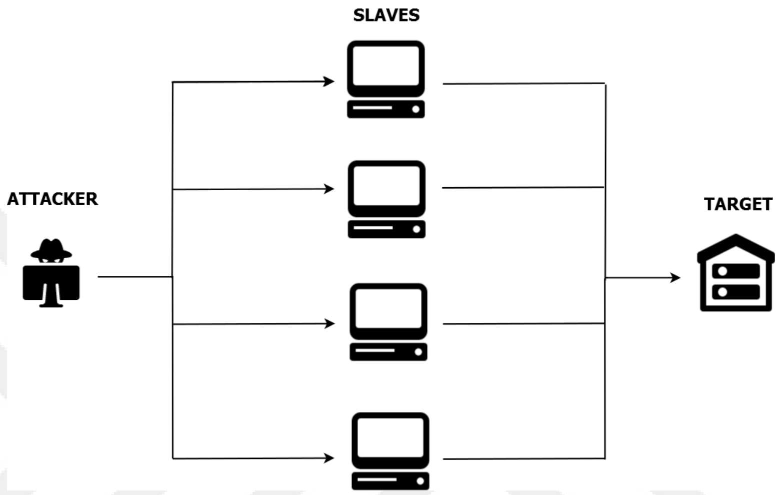


Figure 2.2: Conceptual Distributed Denial of Service Scenario

In Figure 2.2, Slave computer nodes also known as Zombies which are basically compromised network devices that are commanded by the attacker, in order to flood target server by utilizing their different network throughput capabilities.

There are various methods to conduct denial of service attacks. All these methods can be considered in two different kinds.

- Connection Oriented; The attack occurs once a connection between server and client has been established under certain standard protocols such as TCP/IP.

- Connectionless; To conduct the attack, fully established and ensured connection is not necessary. In this type of attacks, attacker floods the

[5] Nodes that are intentionally or unintentionally contributing to DDoS Attack

traffic regardless of its transmission status. UDP protocol is a fine example for this type of attacks.

Dividing DDoS attacks into two high level categories from network point of view is not enough to draw the big picture. There are also three different main categories where these type of attacks can be identified from Cyber-Security point of view.

- Application Layer (OSI Layer 7) Attacks are mostly connection based depending on the target application that is being used. The purpose of the application layer attacks is to monopolize or dominate target service by establishing low traffic rate and mostly legitimate connections without actually utilizing the service thus exhausting the targeted service or server.
- TCP State Exhaustion attacks are performed to abuse and disrupt limited TCP connection states that can be handled by the target. Therefore any extra TCP connection attempts that is generated as a legitimate connection will be queued until target device can reply/handle more TCP slots.
- Volumetric Attacks also known as ”flooding” are generating massive loads of connectionless traffic causing saturation of the traffic or the bandwidth. (Bellovin, 1989)

Enumeration (Information Gathering)

Enumeration is the process of gathering as much information as possible about a system in order to conduct more effective penetration testing process. Enumeration is the key for all cyber-analysis whether intention is offensive or defensive. In Cyber Security discipline its well known that success rate of a malicious attack highly depends on how much information is out there and can be gathered about victim system or company without alerting the target (Martin, 2008). Enumeration also refereed as Information Gathering. Three types of information gathering or enumeration phases are as follows;

Active Information Gathering Active information gathering is the process of collecting as much information as possible about target. Active information gathering process includes but not limited to DNS Enumeration, Port Scanning[6], active host scanning[7] and vulnerability scanning (Jibao, Huiqiang, & Liang, 2006) (Xi, Jin, Yun, & Zhang, 2011).

Key distinction of Active Information Gathering then other enumeration categories is active information gatherings are potentially detectable and/or traceable. Phone calls, security cameras, firewall logs are always there to conduct counter correlation attack to figure out the details about the origin (Yin, Yurcik, & Slagell, 2005).

Passive Information Gathering Passive Information gathering considered as any act of collecting information about a target without communicating directly with the target. Whois[8], background check and public company information are some examples of Passive information gathering.

Open Source Intelligence (OSI) Open Source Intelligence often known as OSI can be considered a subsection of Passive Information Gathering. OSI mainly involves gathering publicly available information about a target organization. Attackers tend to browse target organization's website, look for organizations economical activities, identify structure of the organization and contact information in that organization.

[6] Scanning all 65535 or a subsection of TCP-IP Ports on target to identify running services or weaknesses

[7] Identifying live host IP addresses on a network

[8] Process of checking publicly announced information about a domain name to gather information like Registration contact, Name Server address and administrative contacts.

Exploitation

Any action that enforces another application to misbehave in a way that target application malfunctions as attacker is configured to be. Thus abusing behavior of the victim application to accomplish certain task such as remote shell, local file inclusion or buffer overflow attacks (Portokalidis et al., 2006).

Cyberspace

Cyberspace considered as the online world of computer networks that includes all interconnected peers as well as the area where all events takes place within those interconnected nodes (Benedikt, 1991).

Zero-Day / 0Day

Zero Day is a technical term that is used to describe cyber attacks that are not yet been used before and/or has not been detected before. Zero Days are usually appear in exploitation format rather than a new technique (Syversen, 2006)(Alazab, Venkatraman, Watters, & Alazab, 2011).

CHAPTER 3

IMPLEMENTATION

Computer networks and network security is not a brand new subject yet concepts that are covered in Chapter 2 can be considered as new approaches to overcome the current limitations and drawbacks within these fields. Motivation of this research is to combine these technologies that are described in Chapter 2 and orchestrate them on a *NIX based operating system that is custom build with script-hooks. As described ASICs are capable of performing predefined set of functions whereas *NIX based O/S and its Sandbox nature allows to mimic these functionalities.

In this chapter, both hardware and software implementations and their corresponding details are described. The procedures that are followed in this chapter aims to build a sufficient hardware platform in order to meet SDN and Virtualization needs as well as additional I/O devices that can be utilized for different tasks. Since the base hardware platform is a generic x86 architecture, controlling necessary hardware assets within the operating system became possible. Therefore hardware implementation of this work is to extend I/O interaction for both hardware interfaces and the software interrupts. Building the customized O/S for the related or preferred architecture is to utilize the hardware more effectively compared to a pre-compiled known O/S.

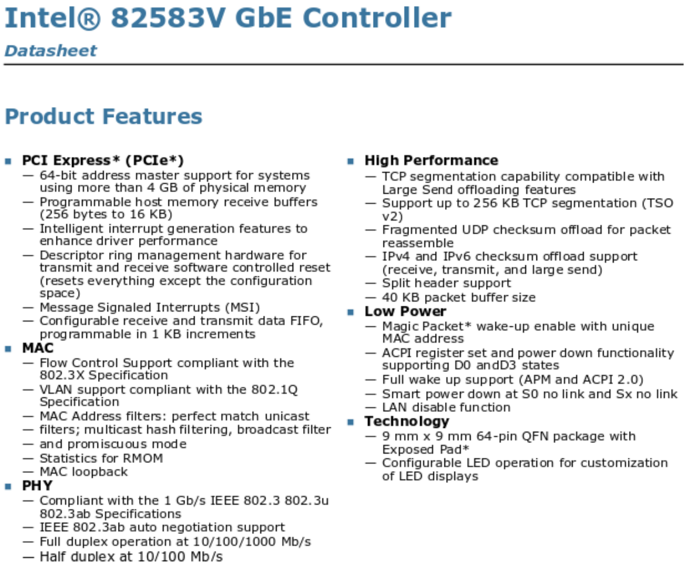
Software implementation phase not only consists of the operating system customization and building but also implementation and usage of the coded functionalities that is covered in this thesis.

As mentioned in Chapter 2, possible features or functionalities that can be implemented on the custom USM devices, behaviors such as port scanning, vulnerability analysis and honeypot deployment are also implemented which are considered to be offensive actions in cyber security point of view.

3.1 HARDWARE IMPLEMENTATION

The hardware platform that is used in this research is a customized x86 embedded board with multiple integrated NICs, Fiber Optic SFP slots, integrated SIM Slot and every other regular personal computer peripheral I/O such as SATA connection and USB ports etc.

The board is designed as a Yasar University Scientific Research Project 014 and manufactured in Shenzen.China. Base platform utilizes an Intel®Atom D525 Dual Core 4 Threads 1.8 Ghz Processor as the central processing unit. 8 GB of DDR3 RAM and six different embedded 100/1000 Mbps IEEE 802.3 Ethernet NICs.[1]. Detailed specifications regarding the NICs can be seen in Figure 3.1



Intel® 82583V GbE Controller

Datasheet

Product Features

- **PCI Express* (PCIe*)**
  — 64-bit address master support for systems using more than 4 GB of physical memory
  — Programmable host memory receive buffers (256 bytes to 16 KB)
  — Intelligent interrupt generation features to enhance driver performance
  — Descriptor ring management hardware for transmit and receive software controlled reset (resets everything except the configuration space)
  — Message Signaled Interrupts (MSI)
  — Configurable receive and transmit data FIFO, programmable in 1 KB increments
- **MAC**
  — Flow Control Support compliant with the 802.3X Specification
  — VLAN support compliant with the 802.1Q Specification
  — MAC Address filters: perfect match unicast
  — filters; multicast hash filtering, broadcast filter
  — and promiscuous mode
  — Statistics for RMOM
  — MAC loopback
- **PHY**
  — Compliant with the 1 Gb/s IEEE 802.3 802.3u 802.3ab Specifications
  — IEEE 802.3ab auto negotiation support
  — Full duplex operation at 10/100/1000 Mb/s
  — Half duplex at 10/100 Mb/s
- **High Performance**
  — TCP segmentation capability compatible with Large Send offloading features
  — Support up to 256 KB TCP segmentation (TSO v2)
  — Fragmented UDP checksum offload for packet reassemble
  — IPv4 and IPv6 checksum offload support (receive, transmit, and large send)
  — Split header support
  — 40 KB packet buffer size
- **Low Power**
  — Magic Packet* wake-up enable with unique MAC address
  — ACPI register set and power down functionality supporting D0 andD3 states
  — Full wake up support (APM and ACPI 2.0)
  — Smart power down at S0 no link and Sx no link
  — LAN disable function
- **Technology**
  — 9 mm x 9 mm 64-pin QFN package with Exposed Pad*
  — Configurable LED operation for customization of LED displays

Figure 3.1: Intel®PCI-E 1Gb 82583v Specifications

Necessity of designing a customized embedded board is to implement SDN based

[1] https://www.intel.com/content/www/us/en/embedded/products/networking/82583v-gbe-controller-datasheet.html

security solution was emerged by the hardware limitations of regular PCs. End user network interface cards are usually designed to accelerate certain amount of sockets at a time, whereas in scope of this project, NICs must be able to handle server grade sockets and connections simultaneously.

There are many alternative out-of-box embedded boards for many other special needs and application areas that could have been used as a computational base platform. Both ensuring hardware reliability of the embedded design of the scientific project, and manipulating the embedded board interrupt addresses more freely, the board was gathered from well known integrated circuit groups.

North Bridge and the South Bridge of the embedded board were already in wide use in other embedded designs and proven to be much more flexible and compatible with other peripherals. Designing an VLSI embedded SoC[2] from ground up is beyond the scope of both SRP014 project as well as this research therefore utilizing south-bridge GPIO[3] pins for extra peripherals such as NICs and GSM 900/1800 module is pursued.

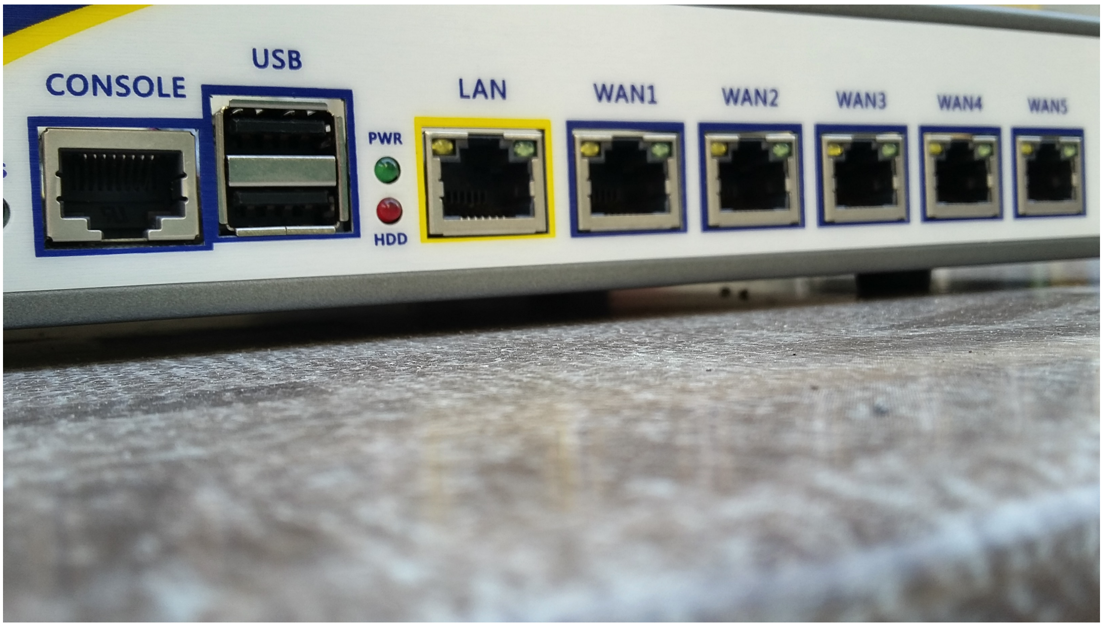


Figure 3.2: Hardware Platform From Front Port Side

[2] System on a Chip
[3] General Purpose Input Output

Input/Output ports are located on front side panel as shown in Figure 3.2. Console port is in the left hand side of the front panel. Cooling fans and PDU power socket is shown in Figure 3.3.

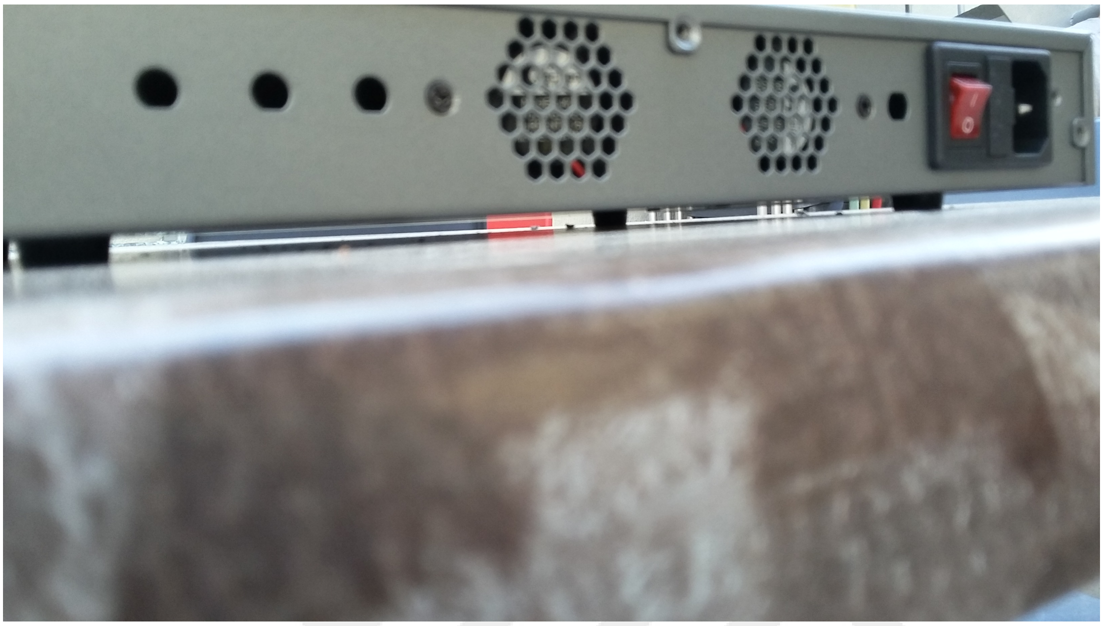


Figure 3.3: Hardware Platform From Backpane Side

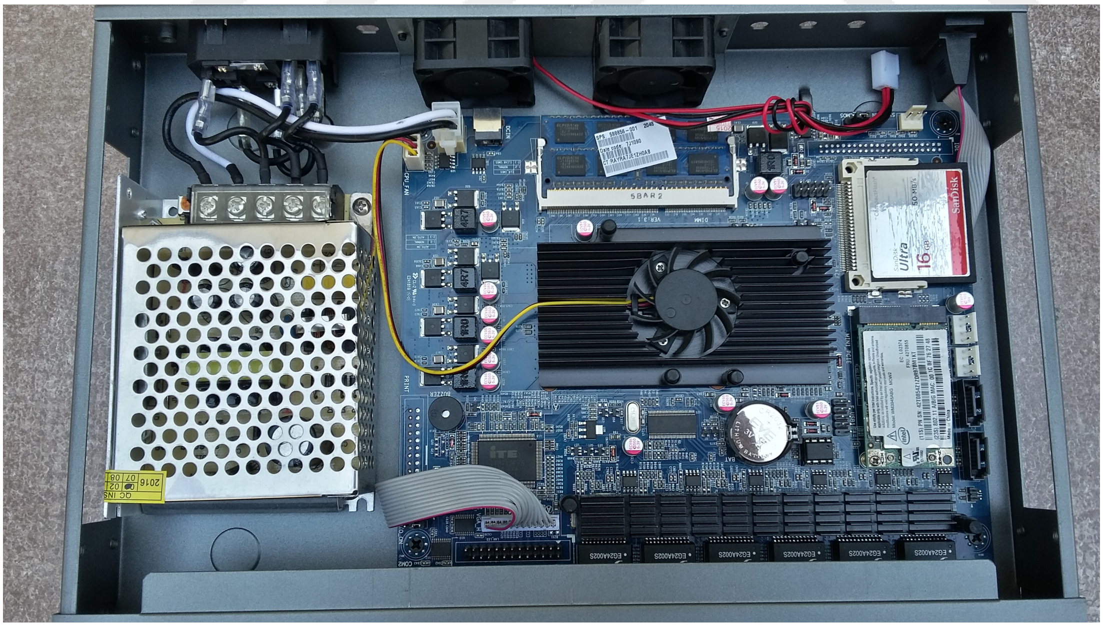


Figure 3.4: Hardware Platform From Top Interior View

Contents of the Operating system are deployed in the Compact Flash card that is shown in the Figure 3.4. Available SATA ports on the board also utilized for cache server and logging features. Logs and outputs that are generated by

Component	Details	Usage
CPU	Intel®Atom D525 Dual Core 4 Threads 1.8 Ghz	Main Processing Unit
Chipset	Intel®I/O Controller Hub 8 (Intel®ICH8M)	Chipset Family
RAM	8GB DDR3 1333Mhz SODIMM	Random Access Memory
Storage Unit	SanDisk®16GB CF Card	Storage Area for OS to Store
Storage Unit 2	SanDisk®128GB SSD	Storage Area for Logs and Software
Network Interface 1	Intel®PCI-E 1Gb 82583v	Ethernet Interface 1
Network Interface 2	Intel®PCI-E 1Gb 82583v	Ethernet Interface 2
Network Interface 3	Intel®PCI-E 1Gb 82583v	Ethernet Interface 3
Network Interface 4	Intel®PCI-E 1Gb 82583v	Ethernet Interface 4
Network Interface 5	Intel®PCI-E 1Gb 82583v	Ethernet Interface 5
Network Interface 6	Intel®PCI-E 1Gb 82583v	Ethernet Interface 6
Power Supply	60W 12V/5A Switching PSU	Power Source for entire Device

Table 3.1: Hardware Specifications

the applications such as `tcpdump` are stored in the secondary storage device that can be installed on the base-platform. During the boot sequence, if there is not any secondary storage device available, Log verbosity is reduced and are stored in the Compact Flash up to allocated storage available. Despite the fact that logs and network dumps are playing a huge role for the automated penetration testing phase and further reporting, Lack of logs or unavailable storage space is not blocking the basic network functionalities.

3.1.1 x86 Architecture For Unified Security Management

Unified Security Management is the concept that allows single network device to be able to handle multiple security tasks such as; Firewall, IPS[4], IDS[5], Web Application Firewall[6], DDoS mitigation and many more. Since USM is based on x86 Generic Architecture, most Unix based Operating system can be installed and interact with peripheral devices. Thus allowing Security Metric Assessment and Reporting Tool to utilize more specialized hardware such as IEEE 802.3 Ethernet Interfaces.(Bugnion et al., 2012) More detailed output and dmidecode output of the architecture under a unix operating system can be found in Appendix 5

[4] Intrusion Prevention System

[5] Intrusion Detection System

[6] Special firewall based on Web traffic and enforcing rules to clients about how they can interact with services.

3.2 SOFTWARE IMPLEMENTATION

x86 Based Hardware platform that is described above is nothing more than an ordinary computer with extra network interface's and other peripherals that is packed into one small embedded board. In order to achieve all SDN capabilities, software design of the concept is the key (Cooper, 2010). Time critical and fully deterministic decision mechanisms are implemented within the software. This implementation aims to harmonize both hardware platform and the soft-features that are defined within SDN software. In this work, Software design and implementation is accomplished by two phases. First phase was to build a custom Linux-Kernel and configuring proper drivers in order to utilize all peripherals that are used on the hardware platform. Second phase was to implement high level functionalities such as automated penetration testing and raw data analysis(Mazurak & Zdancewic, 2007). Detailed SMBIOS output in an ASCII format is provided by the help of `dmidecode` application and is provided in Appendix 5 `dmidecode` is a tool for gathering ASCII readable SMBIOS—DMI of a computer system. SMBIOS stands for System Management BIOS, while DMI stands for Desktop Management Interface. By the help of `dmidecode` tool, detailed hardware information can be gathered on SMART embedded board. This listing can also be referenced as a hardware configuration of the base platform as needed (Brown, 2004).

Second phase of the implementation was to harmonize all peripherals and predefined tasks in a sequential flow. To be able to achieve flexible testing and preproduction, Bash scripting is used. bash scripting is easy to use and can maintain complex scripting tasks. Most of the peripheral module management is coded in Bash scripts and controlling Linux command line applications. Main bash modules are added as scheduled startup programs or as a cron jobs (Solomon, 2007).

3.2.1 Operating System Installed On The Prototype

Operating System development from scratch was way beyond the scope of this thesis. Therefore Linux-Kernel-3.4.113[7] is used as bare-bone operating system. There is nothing much that can be done with bare-bone Linux-Kernel thus necessary drivers and utilities are installed accordingly and tested for any catastrophic driver compatibility issues. Full drivers list can be found in Appendix 5. Further testing and necessary changes is done in testing phase.

Advantages of using *Nix based Operating system as a base are limitless but main advantages are including but not limited to, Open-Source driver/kernel codes that can be configured for specific needs easily, Extensive documentation available for tinkering, Scripting and virtualization technologies are quite powerful on *Nix based systems (Love, 2005) (Bovet & Cesati, 2005) (Henkel, 2006) (Winter, 2008).

Compilation phase is achieved on a Debian[8] based ”Kali Linux Rolling 2.0”[9] distribution. Necessary files and drivers were put in custom build .ISO file during the compilation. Since Kali-Linux is a specifically crafted operating system that is a well known for penetration testing and cyber security needs, Third party applications such as; Nmap, Etterape, Wireshark, were included in this compilation phase. List of the applications and dependencies that are used during the compilation phase can be located in Appendix 5.

In further testing and development in first Alpha release of the entire software bundle, There has been major bugs found in basic operating system functionalities. Especially hardware virtualization and resource handling was not working properly enough to meet minimum expectations even for proof-of-concept version.

[7] Long-Term Support Stable Release
[8] https://www.debian.org/
[9] https://www.kali.org/

pfSense-CE-2.x[10] is an Open-Source firewall and Router operating system distribution based on FreeBSD [11]. Despite the fact that nearly all peripherals of the hardware platform were automatically detected and utilized by the pfSense, customized programs and scripts were unable to function properly on pfSense platform. Therefore as a standalone Firewall/Routing functionalities were overwhelmingly successful but trying to modify pfSense's predefined security precautions and privileges caused pfSense to malfunction beyond recognition (Williams & Bergmann, 2004)(Hollabaugh, 2002).

Kali-Linux-2016.2 [12] is not designed to be used as a permanent operating system yet it is considered as a full-stack penetration testing tool. But in this scope of modifying Kali-Linux and building a custom .ISO bundle with networking capabilities and functionalities as well as hardened linux security kernel was the picked to execute codes and scripts on.

On a persistent and fully operational Kali-Linux distribution, procedures that are used to create an .ISO file are shown below.

```
$ apt install curl git live-build cdebootstrap
$ git clone git://git.kali.org/live-build-config.git
$ cd live-build-config
$ ./build.sh --distribution kali-rolling --verbose
```

Code 3.1: Building OS From Source

Since this process takes a while even on a high end PC, configuration and script hooks are specifically designed to ensure healthy boot sequence. In order to create fast and reliable bootable installation medium for the platform after every release change, bash functions are shown below.

Some of the bash functions are presented below are used during the disk

[10] https://www.pfsense.org

[11] https://freebsd.org

[12] https://docs.kali.org/introduction/what-is-kali-linux

operations where building and cloning of the operating system can be time consuming. By using these functions, human error during this fragile step is eliminated. create-disk 3.2.1 function is the main menu where other disk operations can be used such as privacy wipe 3.2.1 of the storage device. Functions such as disk-image 3.2.1 is there to standardize the bit by bit file copy and formatting operations during and after the custom operating system compilation.

```
function create_disk(){
    echo "${DARKGRAY}"
    echo "+----------------------------------------------------------------------------+"
    echo "| ${LIGHTRED}Warning${DEFAULT}${DARKGRAY} ! Please Thinkg Twice of Your Actions ! |"
    echo "+----------------------------------------------------------------------------+"
    echo "# [1) List Disks | 2) Format a Disk | 3) Privacy Cleanup | 4)Create Disk |"
    echo "+----------------------------------------------------------------------------+"
    echo "${DEFAULT}"
    echo -en "${bold}${RED}EVE${RESET}${normal}${BLUE}->${RESET} "
    read t
        case $t in
        1)  lsblk ;;
        2)  format_disk ;;
        3)  privacy_cleanup ;;
        4)  create_disk_image ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
        echo "Please select a valid option !"
    esac
    #echo "Enough Crypt!"
    pause
}
```

Code 3.2: codes/create_disk.sh

```
function format_disk(){
    echo "${lightyellow}"
    lsblk | grep disk
    echo "${DEFAULT}"
    main_drive=$(lsblk | grep disk | cut -d " " -f1)
    echo "${RLS}You Should Not be Picking your Resident Drive ${RED}$main_drive${DEFAULT}"
    echo "${GLS}Please enter your device [NOT Partition if Image] {ie./dev/sdc} = "
    read padisk
    echo "${RLS}Are you sure ? Please enter again to confirm = "
    read pbdisk
    if [[ "$padisk" = "$pbdisk" ]];
        then
        echo "${GLS} Formatting is Commencing in 5, You can Still Unplug it !"
        sleep_indicator 5
        dd if=/dev/zero of=$padisk bs=1M status=progress && sync
    fi
}
```

Code 3.3: codes/format_disk.sh

```
function create_disk_image(){
    echo "${lightyellow}"
    lsblk | grep disk
    echo "${DEFAULT}"
    main_drive=$(lsblk | grep disk | cut -d " " -f1)
    echo "${RLS}You Should Not be Picking your Resident Drive ${RED}$main_drive${DEFAULT}"
    echo "${GLS}Please enter your device [NOT Partition if Image] {ie./dev/sdc} = "
    read iadisk
    echo "${RLS}Are you sure ? Please enter again to confirm = "
    read ibdisk
    echo "${GLS}Please pick a image file"
    sleep 1
    image_file=$(pick_single_file)
    if [ ! -z "$image_file" ];
            then
            if [[ "$iadisk" = "$ibdisk" ]];
                 then
                    read -r -p "Are you sure? [y/N] " response
                    case "$response" in
                        [yY][eE][sS]|[yY])
                            #echo "${GLS} "
                            sleep_indicator 5 "Creating the image Commencing in 5, You can Still Unplug it !"
                            echo -ne ""
                            dd if=$image_file of=$iadisk bs=1M status=progress && sync
                            ;;
                        *)
                            pause
                            ;;
                    esac
             fi
    fi
}
```

Code 3.4: codes/disk_image.sh

By utilizing simple yet effective command line tool, `dd` [13] every custom build .ISO releases were copied bit by bit to first boot sector of the target medium. CompactFlash CF cards and 2.5 SATA SSD drives were tested throughout the development phase. (Code 3.1)

```
dd if=$image_file of=$iadisk bs=1M status=progress && sync
```

Command takes input of an .IMG or an .ISO file as a source and destination for the targeted medium that will be installed to the platform.`status` parameter is used to ensure total bytes I/O is equivalent to custom build image file.

Following script was used to compare overall hash value for both source and destination in an automated manner.

```
function hash_em_all() {
        file_to_hash=$(pick_single_file)
        if [ ! -z "$file_to_hash" ];
            then
            md5sam=`hash_md5sam "$file_to_hash"`
            write_header "This is MD5Sum"
            printf "\v%s\n" "$md5sam"
            sha160=`hash_sha160 "$file_to_hash"`
            write_header "This is SHA160"
            printf "\v%s\n" "$sha160"
            sha224=`hash_sha224 "$file_to_hash"`
            write_header "This is SHA224"
            printf "\v%s\n" "$sha224"
            sha256=`hash_sha256 "$file_to_hash"`
            write_header "This is SHA256"
            printf "\v%s\n" "$sha256"
            sha384=`hash_sha384 "$file_to_hash"`
            write_header "This is SHA384"
            printf "\v%s\n" "$sha384"
            sha512=`hash_sha512 "$file_to_hash"`
            write_header "This is SHA512"
            printf "\v%s\n" "$sha512"
        else
            echo "Nothing Selected !"
        fi
    fi
    pause
}
```

Code 3.5: Hash Comparision

After confirming a successful power on self test and boot sequence, releases were checked for crucial basic operating system functionalities. All peripherals and

[13] http://www.gnu.org/software/coreutils/dd

internal mechanisms of the board were checked for any compilation or cloning mistakes. `dmidecode`[14] command line software provided all peripheral devices' interrupt request mappings and reserved address spaces on the memory. Since heavy modifications were made on custom drivers, miscalculated address blocks were automatically detected as kernel-panic and linux-kernel immediately flagged it as a possible buffer overflow on system address space.

3.2.2 S.M.A.R.T

S.M.A.R.T stands for "Security Metric Assessment and Reporting Tool", which lays in the very core of the NFV-Based security concept that is being presented. Main purpose of this module is to orchestrate other peripheral modules in harmony so that S.M.A.R.T can analyze incoming telemetry logs and take action based on those logs.

Despite the fact that representation of any generic data or information can be broken down to bits or bytes, there are no security measurement units to answer "How Much" questions. Therefore S.M.A.R.T's Metric is relative. This relative unit is derived by collected logs and analysis of security breach logs and submitted security incident reports to S.M.A.R.T database. Therefore "metric" measurement is not a valid statement yet it's a relative definition to score entries specifically predefined to analyze and correlate incoming logs.

S.M.A.R.T runs its automated vulnerability and exploitation scenarios based on a finite state decisions. Each outcome and action analyzed by the S.M.A.R.T by the predefined procedures and boundaries in order to decide next possible action on the chart that is presented in Appendix 5

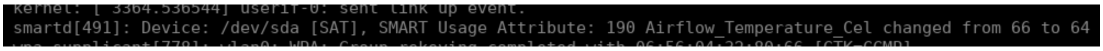
[14] https://linux.die.net/man/8/dmidecode

Administrative Logs Generated by the S.M.A.R.T

IT Administrative logs are one of the must-have in an organization IT infrastructure. Administrative logs can be gathered in different detail level depending on the device that produces them. In our scope Administrative logs will be based on the logs that are generated by the Security Metric Assessment and Reporting Tool, and mainly involves security flaws, possible impact ratings and possible solution procedures if applicable.

Implementation of remote log gathering is accomplished by using `rsyslog`. Rsyslog allows to gather system logs from remote nodes as well as sending them over network. All the log message from the kernel and the operating system applications are distributed to the logs of the related files under the `/var/log` directory.

Security metric assessment and reporting tool is capable of collecting logs such as network traffic, user activity and possible security breaches in a categorized fashion, where these logs can be set individually in terms of verbosity of the corresponding task. Maintaining the system health status as well as environmental changes also under the administrative logs category in the scope of this research. Example airflow and CPU temperature also is fed to Administrative logs as can be seen in Figure 3.5



```
kernel: [ 3364.536544] userif-0: sent link up event.
smartd[491]: Device: /dev/sda [SAT], SMART Usage Attribute: 190 Airflow_Temperature_Cel changed from 66 to 64
wpa_supplicant[778]: wlan0: WPA: Group rekeying completed with 06:56:04:22:80:66 [GTK=CCMP]
```

Figure 3.5: Temperature Airflow Change for the Fans

Honeypots

Honeypots are deployed for any reverse search may occur during active penetration testing phase. Also allowing administrative logs that are generated by Security Metric Assessment and Reporting Tool to include all malicious and unauthorized access attempts with detailed information (Zou & Cunningham,

2006) (Wang, Wu, Cunningham, & Zou, 2010).

Honeypots that are deployed by S.M.A.R.T is mostly based on well-known ports but can also be customized to emulate most TCP and UDP services. Because of that, ports that are not in use will be redirected to a Honey pot with no internal access yet only for logging all connection attempts with detailed geo-location and IP address information.

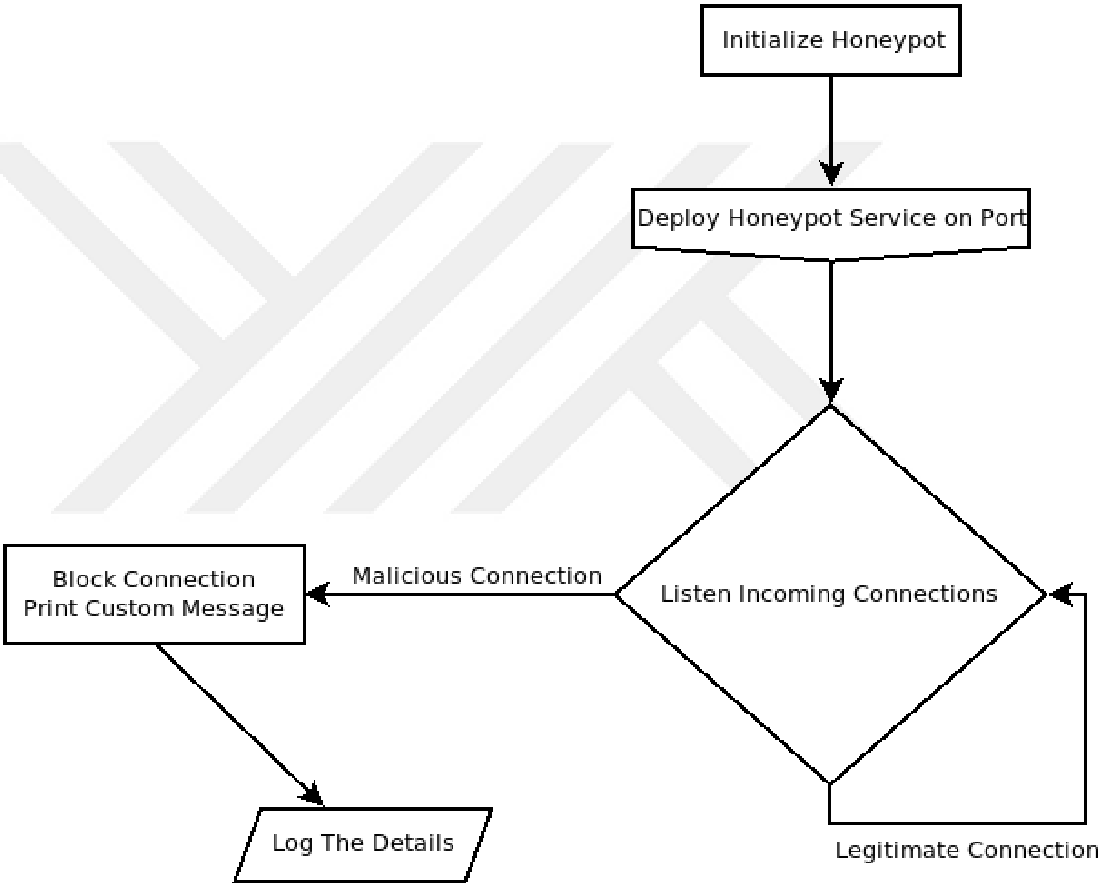
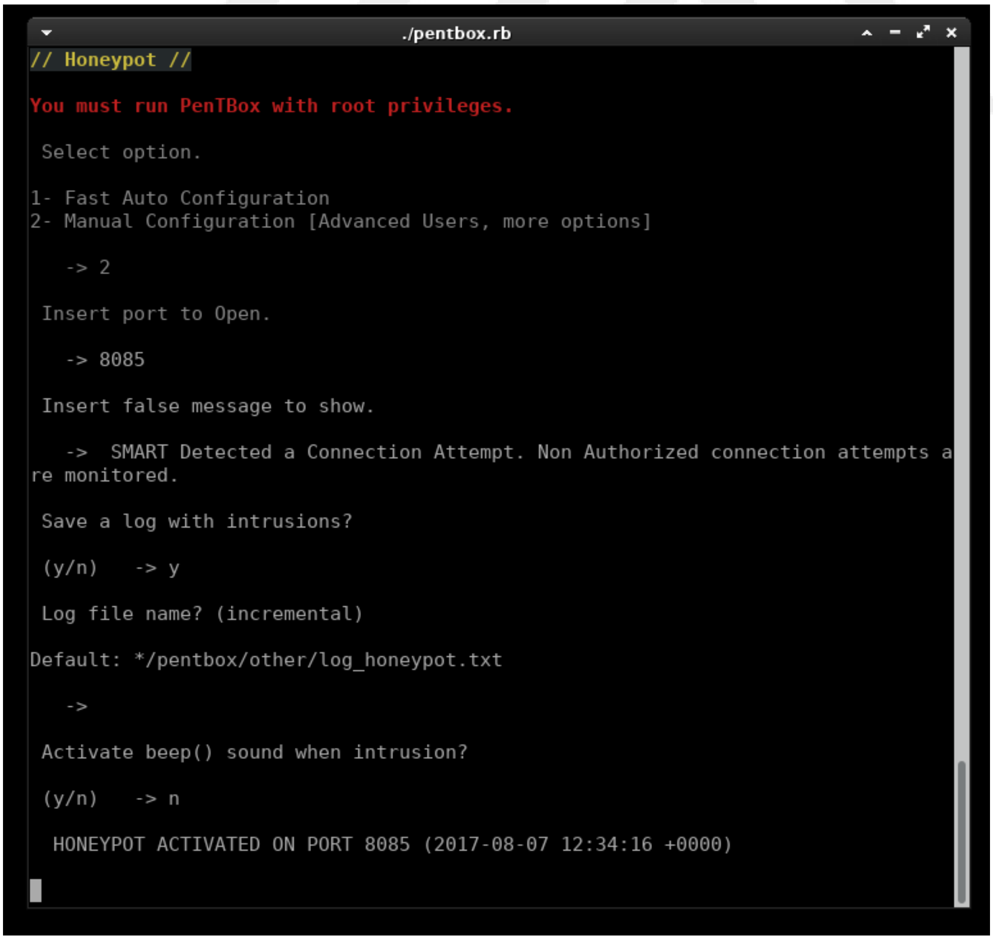


Figure 3.6: Flowchart of the Honeypot Sequence

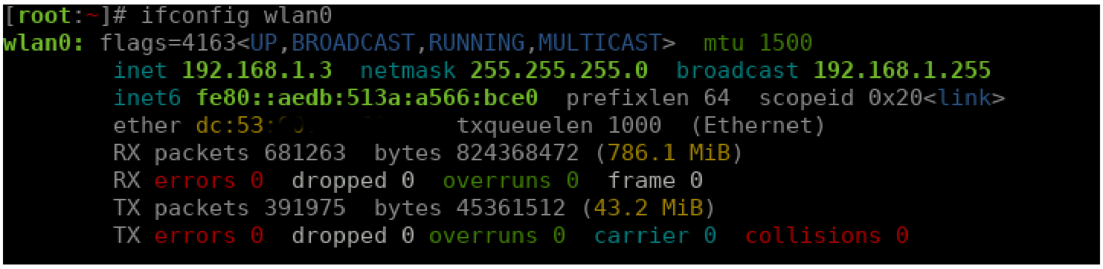
Context level flow chart is as shown in Figure 3.6. Regardless of the manual and automated deployment of the honeypot, initial parameters are provided either by the user or the S.M.A.R.T's predefined rules. In Figure 3.7, manual deployment parameters are provided thus honeypot deployed. After this operation, a listener runs in an infinite loop, until it is interrupted, constantly checking for matching true positive attempt in order to take further actions. Unless the connection does not match the provided parameters, Cycle continues and pattern matching is performed. In case of any Malicious connection detection, S.M.A.R.T blocks the connection easily since its the gateway router, and pushes custom warning message to the client. After logging the details of the incoming attack, Honeypot can be configured to halt or keep alive with same configurations.

By the help of deployed honey pots throughout the network, dramatically improves the identification of possible breach attempts. Custom Honeypot Deployed on TCP 8085 Port. As shown in Figure 3.8 local ip address of the wireless network interface is 192.168.1.3 with subnet of 255.255.255.0 under the 192.168.1.0 Network. Outbound connection is performed both within the Local network and port-mapped WAN attempt.



```
// Honeypot //

You must run PenTBox with root privileges.

 Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

   -> 2

 Insert port to Open.

   -> 8085

 Insert false message to show.

   ->  SMART Detected a Connection Attempt. Non Authorized connection attempts a
re monitored.

 Save a log with intrusions?

 (y/n)   -> y

 Log file name? (incremental)

Default: */pentbox/other/log_honeypot.txt

   ->

 Activate beep() sound when intrusion?

 (y/n)   -> n

  HONEYPOT ACTIVATED ON PORT 8085 (2017-08-07 12:34:16 +0000)

```

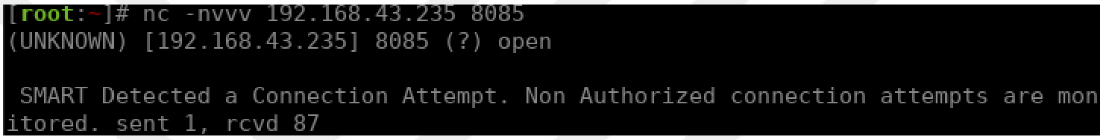
Figure 3.7: TCP 8085 Honeypot Manual Deployment



```
[root:~]# ifconfig wlan0
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.3  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::aedb:513a:a566:bce0  prefixlen 64  scopeid 0x20<link>
        ether dc:53:[redacted]        txqueuelen 1000  (Ethernet)
        RX packets 681263  bytes 824368472 (786.1 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 391975  bytes 45361512 (43.2 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3.8: Local Interface IP Address

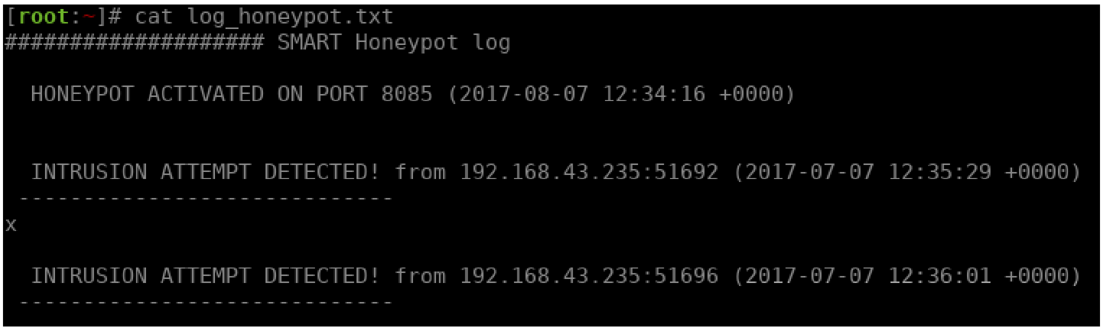
Customized honeypot can be easily deployed and since it is a Ruby script it also can be automated within the SMART just like any other script that is being automatically performed (Weiler, 2002).



```
[root:~]# nc -nvvv 192.168.43.235 8085
(UNKNOWN) [192.168.43.235] 8085 (?) open

 SMART Detected a Connection Attempt. Non Authorized connection attempts are mon
itored. sent 1, rcvd 87
```

Figure 3.9: RAW TCP netcat connection on 8085 Port

Raw TCP connection to 8085 port became available after the honeypot deployment and even transmitted a custom message to the client. (Figure 3.9)



```
[root:~]# cat log_honeypot.txt
#################### SMART Honeypot log

  HONEYPOT ACTIVATED ON PORT 8085 (2017-08-07 12:34:16 +0000)


  INTRUSION ATTEMPT DETECTED! from 192.168.43.235:51692 (2017-07-07 12:35:29 +0000)
 -----------------------------
x

  INTRUSION ATTEMPT DETECTED! from 192.168.43.235:51696 (2017-07-07 12:36:01 +0000)
 -----------------------------
```

Figure 3.10: Contents of a log file

In honeypot logs Intrusion attempts and even possible breaches are listed with their corresponding timestamps. These logs are fed into the Administrative logs before generating a report file for the IT administrators. As another example for the legitimate TCP 22 SSH port same manual deployment procedures are shown in Figure 3.11 (Zhang, Zhou, Qin, & Liu, 2003) (Mairh, Barik, Verma, & Jena, 2011).

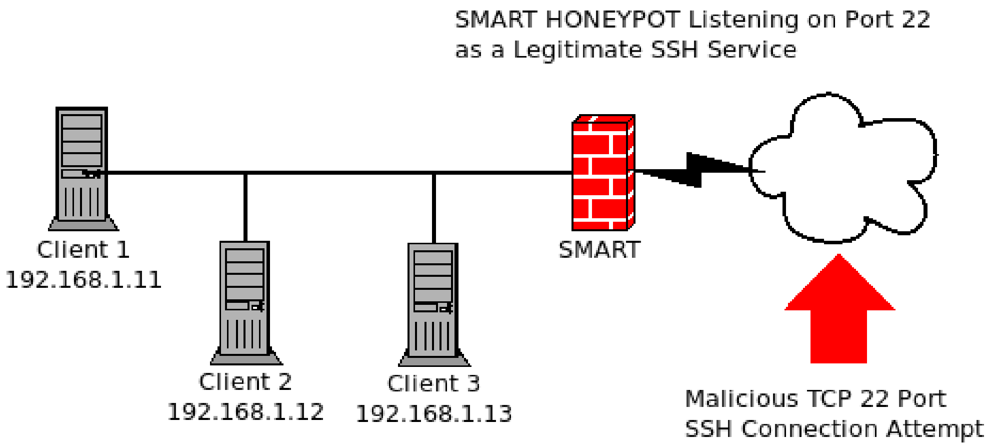
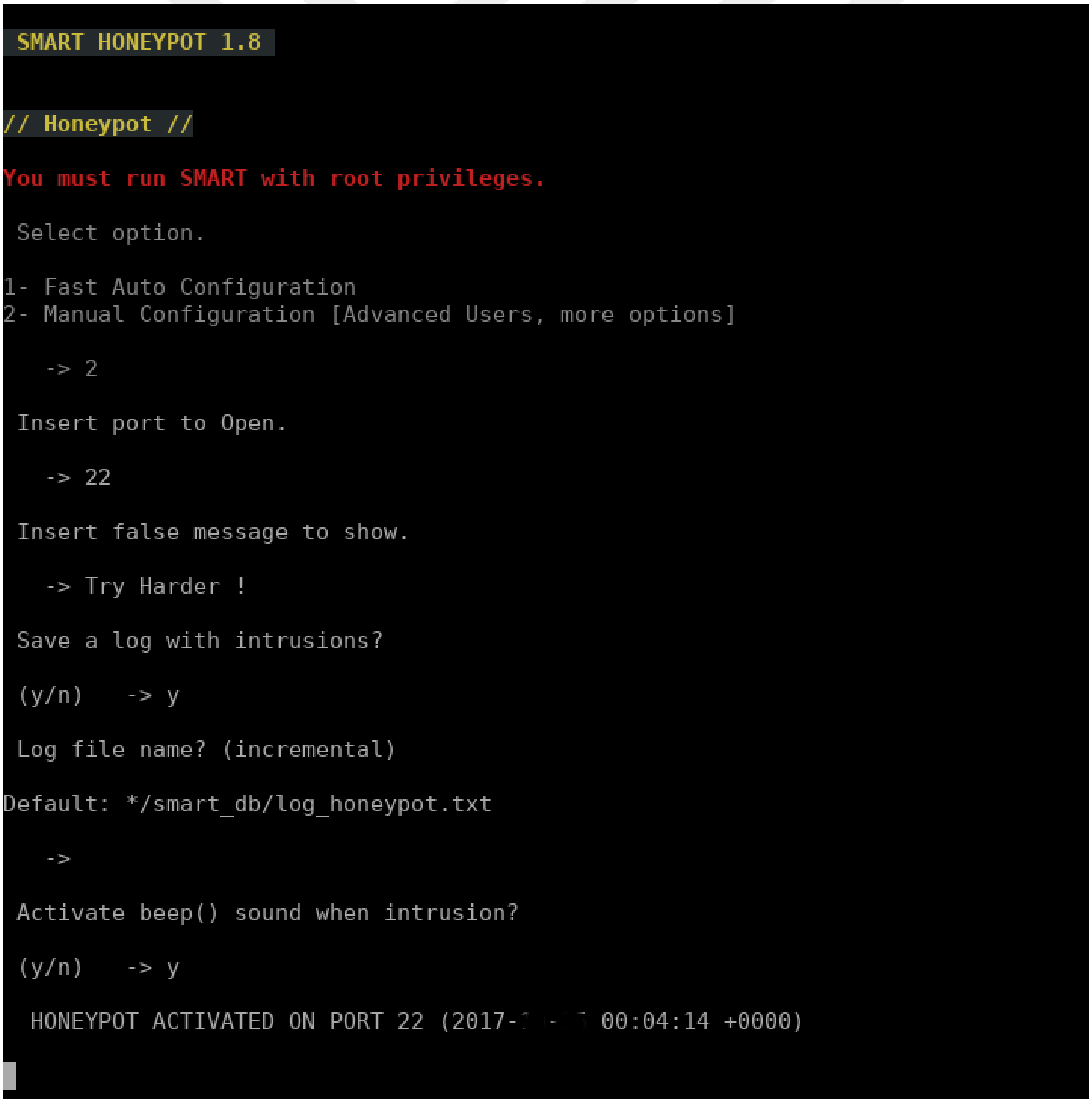
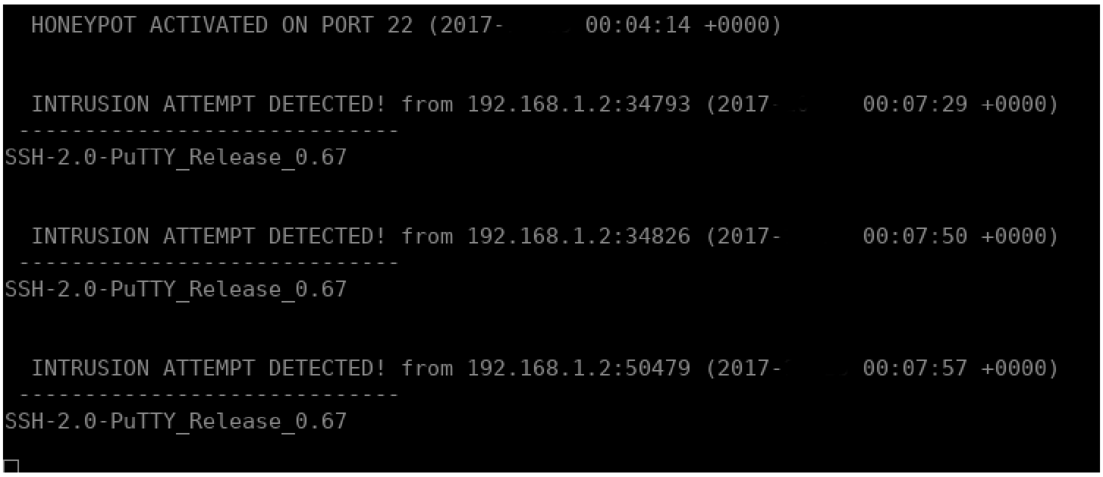


Figure 3.11: Honeypot SSH TCP/22 Example From WAN



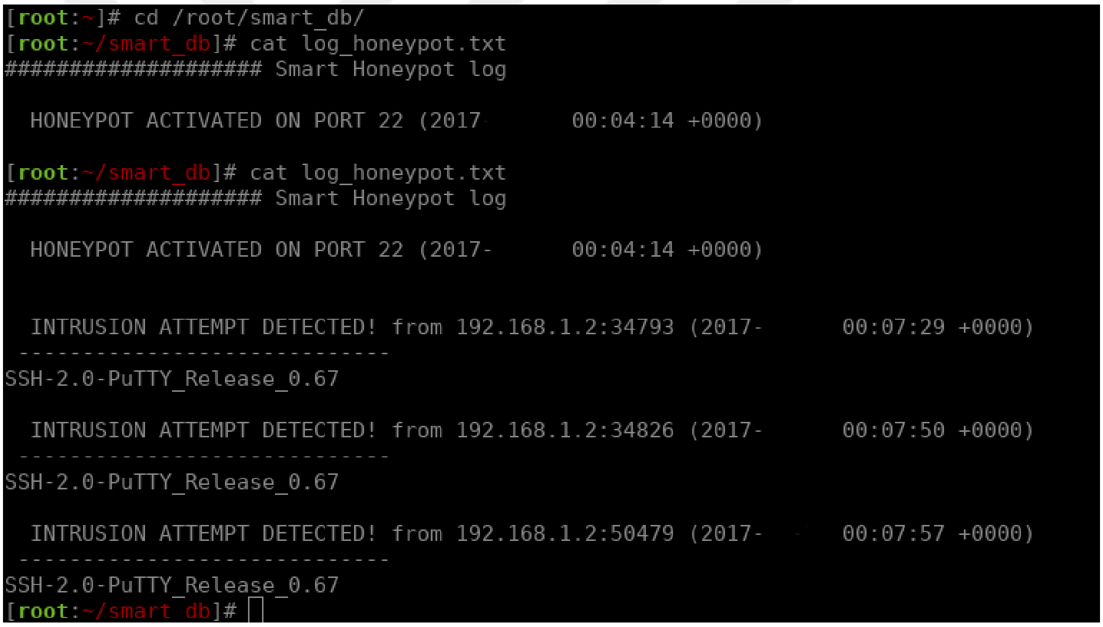
```
 SMART HONEYPOT 1.8 


// Honeypot //

You must run SMART with root privileges.

 Select option.

1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]

   -> 2

 Insert port to Open.

   -> 22

 Insert false message to show.

   -> Try Harder !

 Save a log with intrusions?

 (y/n)   -> y

 Log file name? (incremental)

Default: */smart_db/log_honeypot.txt

   ->

 Activate beep() sound when intrusion?

 (y/n)   -> y

  HONEYPOT ACTIVATED ON PORT 22 (2017-[redacted]-[redacted] 00:04:14 +0000)
```

Figure 3.12: Honeypot SSH Manual Deployement



```
  HONEYPOT ACTIVATED ON PORT 22 (2017-[redacted] 00:04:14 +0000)


  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34793 (2017-[redacted] 00:07:29 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67


  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34826 (2017-[redacted] 00:07:50 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67


  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:50479 (2017-[redacted] 00:07:57 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67
```

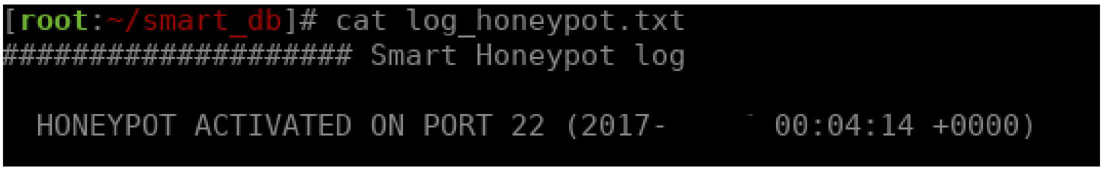
Figure 3.13: Output of the Terminal App



```
[root:~]# cd /root/smart_db/
[root:~/smart_db]# cat log_honeypot.txt
#################### Smart Honeypot log

  HONEYPOT ACTIVATED ON PORT 22 (2017[redacted] 00:04:14 +0000)

[root:~/smart_db]# cat log_honeypot.txt
#################### Smart Honeypot log

  HONEYPOT ACTIVATED ON PORT 22 (2017-[redacted] 00:04:14 +0000)


  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34793 (2017-[redacted] 00:07:29 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67

  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34826 (2017-[redacted] 00:07:50 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67

  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:50479 (2017-[redacted] 00:07:57 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67
[root:~/smart_db]#
```

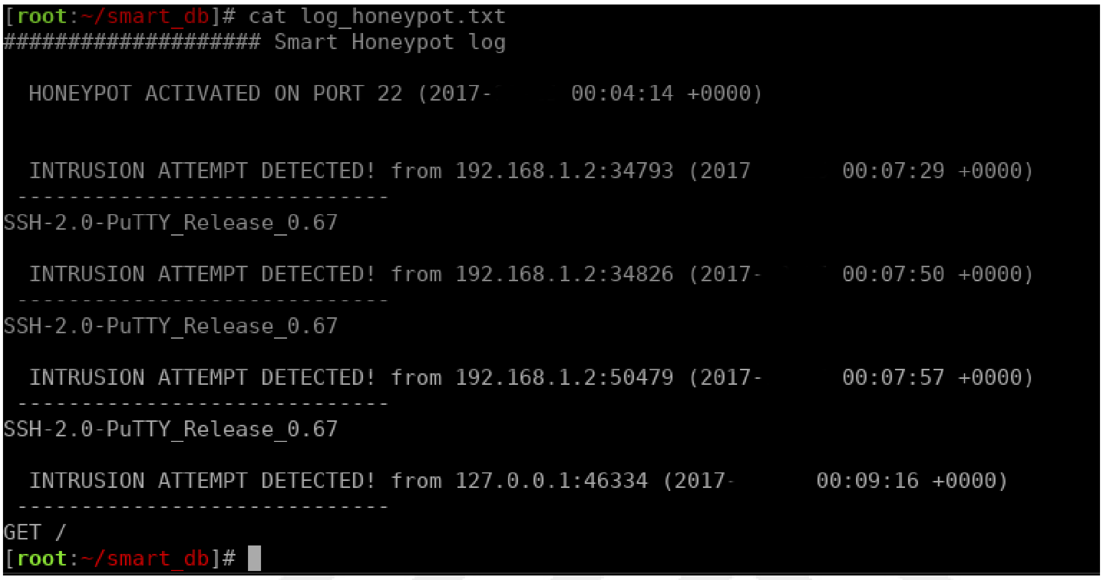
Figure 3.14: Contents of the honeypot's log file

Now incoming TCP 22 SSH connection from loopback interface 'lo' created a raw `netcat` connection (Figure 3.14). Despite the fact that its a virtual interface and located on the same host device, by the help of *nix based file operations, any connection attempt regardless of the origin, will be flagged as a malicious attempt and will perform the same procedures as in any other outbound connection (Krishnaprasad, 2017).



```
[root:~/smart_db]# cat log_honeypot.txt
#################### Smart Honeypot log

 HONEYPOT ACTIVATED ON PORT 22 (2017-[redacted] 00:04:14 +0000)
```

Figure 3.15: Raw Netcat TCP connection to listening Loopback Interface

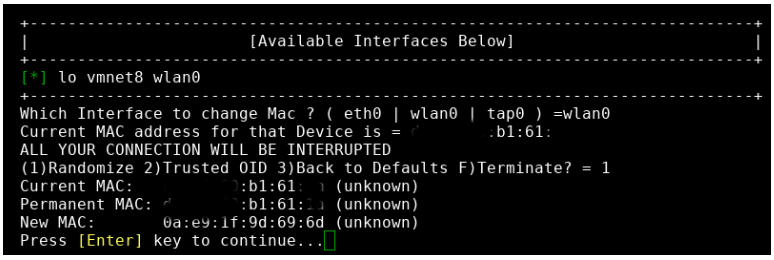


```
[root:~/smart_db]# cat log_honeypot.txt
#################### Smart Honeypot log

  HONEYPOT ACTIVATED ON PORT 22 (2017-[redacted] 00:04:14 +0000)


  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34793 (2017[redacted] 00:07:29 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67

  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:34826 (2017-[redacted] 00:07:50 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67

  INTRUSION ATTEMPT DETECTED! from 192.168.1.2:50479 (2017-[redacted] 00:07:57 +0000)
 -----------------------------
SSH-2.0-PuTTY_Release_0.67

  INTRUSION ATTEMPT DETECTED! from 127.0.0.1:46334 (2017-[redacted] 00:09:16 +0000)
 -----------------------------
GET /
[root:~/smart_db]# 
```

Figure 3.16: Contents of the Log file after local Malicious connection attempt

Between each operation, Hashing scripts shown in 3.2.1 are executed and store in a separate checksum file in order to detect any tempering with the log file. These generated hashes can be secured using various data leak prevention techniques (Provos et al., 2004) (Teodorczyk, 2013).

MAC Address Disguise



```
+--------------------------------------------------------------------------+
|                        [Available Interfaces Below]                        |
+--------------------------------------------------------------------------+
[*] lo vmnet8 wlan0
+--------------------------------------------------------------------------+
Which Interface to change Mac ? ( eth0 | wlan0 | tap0 ) =wlan0
Current MAC address for that Device is = [redacted]:b1:61:[redacted]
ALL YOUR CONNECTION WILL BE INTERRUPTED
(1)Randomize 2)Trusted OID 3)Back to Defaults F)Terminate? = 1
Current MAC:     [redacted]:b1:61[redacted] (unknown)
Permanent MAC: [redacted]:b1:61:[redacted] (unknown)
New MAC:         0a:e9:1f:9d:69:6d (unknown)
Press [Enter] key to continue...
```

Figure 3.17: Override the Permanent MAC Address

In order to avoid legitimate local users' antivirus software or any other client side precautions, MAC address of the network interface where active penetration testing is performed, can rapidly and automatically be manipulated just like many malicious attackers do by default of actions.

This implementation can be extended with Link-Layer attacks as well as to deploy Link-Layer secure frame communication during a more high level attack in a peer to peer fashion. But in this research, MAC address manipulation is used solely and can be seen in Figure 3.17.

```
# {*} Function status = Finished
# {*} Function Desc = Spoof Mac Adress for any interface
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None

function change_mac() {

    echo ""
    echo "+--------------------------------------------------------------------------+"
    echo "|                   [Available Interfaces Below]                   |"
    echo "+--------------------------------------------------------------------------+"
    echo -ne "${GLS} "
    echo ‘ifconfig | grep flags | cut -d ":" -f1‘
    echo "+--------------------------------------------------------------------------+"

    read -p "Which Interface to change Mac ? ( eth0 | wlan0 | tap0 ) =" chcInf

    if [ -z "$chcInf" ];

        then
        echo "${RLS} No Interface Selected !"

    else

        echo -n "Current MAC address for that Device is = "
        curr_mac=‘ifconfig $chcInf | grep ether | cut -d " " -f 10‘
        echo "$curr_mac"
        echo "ALL YOUR CONNECTION WILL BE INTERRUPTED"

        read -p "(1)Randomize 2)Trusted OID 3)Back to Defaults F)Terminate? = " ce

        case "$ce" in

            1) sudo ifconfig $chcInf down; sudo macchanger -r $chcInf; sudo ifconfig $chcInf up;;
            2) sudo ifconfig $chcInf down; sudo macchanger -e $chcInf; sudo ifconfig $chcInf up;;
            3) sudo ifconfig $chcInf down; sudo macchanger -p $chcInf; sudo ifconfig $chcInf up;;
            f|F ) echo " Terminated !" ;;
            * ) echo "No Input Provided";;

        esac

    fi

    pause

}
```

Code 3.6: codes/mac.sh

Go Turtle

Under an ongoing cyber attack where the attack is identified as a denial of service attack, legitimate local network traffic even the Web traffic may come to a stopping point, Especially under the circumstances where possible breach is confirmed but is not exactly been pinpointed, limiting all network traffic to 80 and 443 TCP/IP ports as well as logging other connection attempts is the approach that S.M.A.R.T will perform.

In case of ongoing cyber attack detected, Running all countermeasures might not be sufficient. In order to allow HTTP[15] and HTTPS[16] traffic regardless of the attacks magnitude, `IPTABLES` configuration is deployed. Since this configuration will block all other connection attempts it serves as a last resort to keep LAN[17] operational.

```
# {*} Function status = To be Tested
# {*} Function Desc = Man gotta protect himself right ?
# {*} Function To do = None
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None

function go_turtle() {

 # allow only 1.1.1.0/24 and ports 80,443 and log drops to /var/log/messages

  iptables -A INPUT -s 1.1.1.0/24 -m state --state RELATED,ESTABLISHED,NEW -p tcp -m multiport --dports
       80,443 -j ACCEPT
  iptables -A INPUT -i eth0 -m state --state RELATED,ESTABLISHED,NEW -j ACCEPT
  iptables -A INPUT DROP
  iptables -A OUTPUT -o eth0 -j ACCEPT
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A OUTPUT -o lo -j ACCEPT
  iptables -N LOGGING
  iptables -A INPUT -j LOGGING
  iptables -A LOGGING -m limit --limit 4/min -J LOG --log_prefix "EVE_DROPPED_CONN "
  iptables -A LOGGING -j DROP

}
```

Code 3.7: codes/go_turtle.sh

SMB Null Session Checker

Local networks are configured as a secure location where file sharing and printing services are presented. During penetration testings in various company networks, unauthorized file sharing caused by misconfiguration is a fine example for malicious attacker to gain access to the root file system. SMB null session attacks can be performed on the network and needs to be checked regularly. Code snippet that is provided below is to automate the scan on the local network regularly by conducting the search on the list of IP addresses provided within.

```
# {*} Function status = Finished
# {*} Function Desc = SMB Null Session Checker
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None

function nullmein(){

    if [ -z "$1" ]; then

        echo "[*] Try SMB Null Session for specific ip or range"
```

[15] Hyper Text Transfer Protocol

[16] Hyper Text Transfer Protocol Secure

[17] LocalArea Network

```
      echo "[*] Usage : $0 <file_to_read>"

   exit 0
   fi

   source_file=$1;

   for ips in $(cat $source_file); do

      printf "Scanning for Null Session @ %s\n""$ips"
      output=‘bash -c "echo ’srvinfo’ | rpcclient $ips -U%"‘
      echo $output

   done

}
```

Code 3.8: codes/nullmein.sh

Checks any given IP range for SMB[18] Null Sessions.

3.3 ACTIVE PENETRATION TEST BY THE S.M.A.R.T

Code snippets and bash functions that are covered prior to this section is to emphasize the S.M.A.R.T approach and its possible benefits over the manual penetration testing phases. Entire system is made of key elements such as hardware platform, custom operating system and S.M.A.R.T bundle. All these elements are gathered specifically in order to add autonomous penetration testing capability to the S.M.A.R.T and chosen as a security approach in this scope.

The concept of active penetration testing that is being presented, is a reference to describe procedures of a penetration testing pursued by the Security Metric Assessment and Reporting Tool. This feature aims to achieve basic penetration testing procedures in an autonomous fashion. These produces are including but not limited to, Port Scanning, Web Vulnerability analysis and enumeration. By following these procedures in a customizable list format, Security Metric Assessment and Reporting Tool is able to perform these tasks with regular periods that are defined by `crontab` [19]

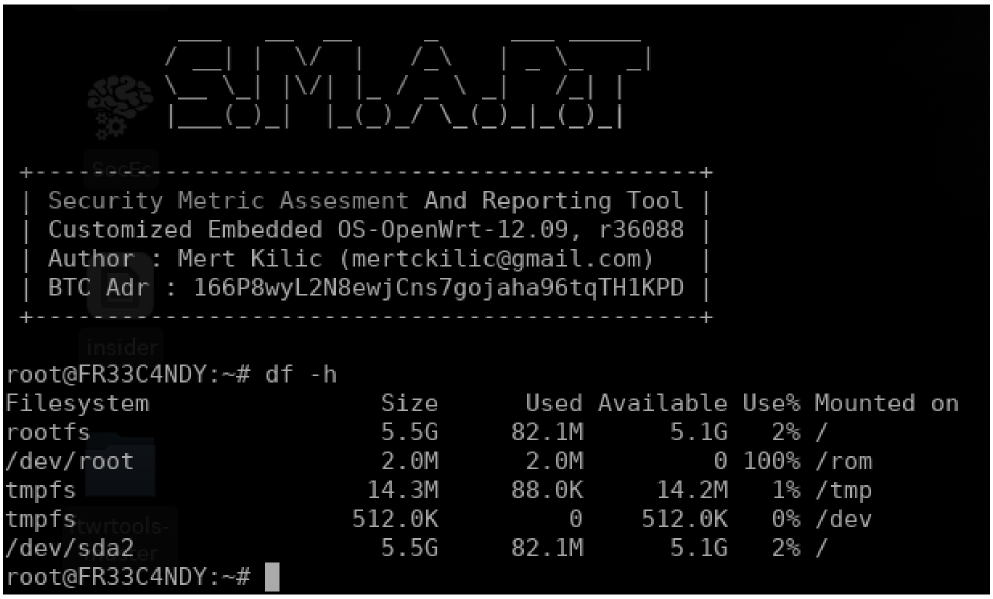
[18] Server Message Block

[19] UNIX utility that allows scheduling tasks for running a script,software or a command within defined intervals.

Predefined set of security checks are utilized in order to conduct and analyze current well known security flaws of a certain network. These security flaws might appear on a daily basis in a growing network. In order to prevent the bulky work that is needed to analyze overall security structure, Active penetration testing phase allows to conduct these tests without any administrative support.

As a result of Active Penetration testing, at the end an analysis report is presented and logged to System administrator in order to further investigate or take actions on the problems that are beyond of the abilities of S.M.A.R.T . These extra logs or possible security flaws that are not directly fixable, are presented to Administrators, daily, weekly and monthly.

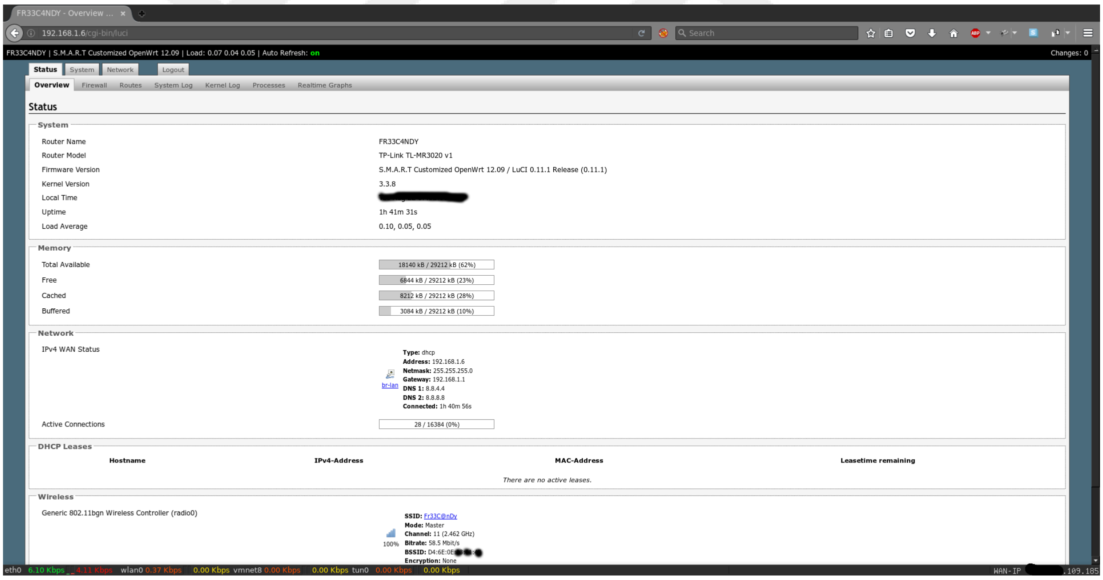
Peripheral equipments are also can be used to feed external information to main S.M.A.R.T database. These peripheral equipment can be easily deployed using customized OpenWRT router to conduct Wireless Grade Security analysis. By Utilizing customized OpenWRT, small embedded router can be turned into a mobile penetration testing device with central management system, S.M.A.R.T (Figure 3.18) (Fainelli, 2008) (Petullo, 2010).



```
      ___   __  __     _     ___ _____
     / __| |  \/  |   /_\   | _ \_   _|
     \__ \_| |\/| |_ / _ \ _|   / | |
     |___(_)_|  |_(_)_/ \_(_)_|_(_)_|

 +-------------------------------------------+
 | Security Metric Assesment And Reporting Tool |
 | Customized Embedded OS-OpenWrt-12.09, r36088 |
 | Author : Mert Kilic (mertckilic@gmail.com)   |
 | BTC Adr : 166P8wyL2N8ewjCns7gojaha96tqTH1KPD |
 +-------------------------------------------+

root@FR33C4NDY:~# df -h
Filesystem                Size      Used Available Use% Mounted on
rootfs                    5.5G     82.1M      5.1G   2% /
/dev/root                 2.0M      2.0M         0 100% /rom
tmpfs                    14.3M     88.0K     14.2M   1% /tmp
tmpfs                   512.0K         0    512.0K   0% /dev
/dev/sda2                 5.5G     82.1M      5.1G   2% /
root@FR33C4NDY:~# 
```

Figure 3.18: Customized OpenWRT console connection

Web Interface that is hosted within the embedded device can be utilized to maintain actions and status of ongoing tasks. Classical and user friendly web UI is as shown in Figure 3.19. By the help of this management interface, on site tests and certain configurations of the peripheral device can be configured on the go. These configurations are including but not limited to; Management IP address, DHCP server configuration and Wifi Capture options. Also, for configuration of the OpenWRT box's system parameters, such as root file system, device drivers and USB Root Hub; serial connection or SSH connection directly to management IP will allow direct access to the running terminal session, which is shown in 3.19. Monitoring live stats such as memory usage, storage usage and network details are also located in OpenWRT's default Web interface, in customized version in smartWRT unrelated monitoring and management options are discarded.



FR33C4NDY | S.M.A.R.T Customized OpenWrt 12.09 | Load: 0.07 0.04 0.05 | Auto Refresh: on

Changes: 0

Status

System

Router Name	FR33C4NDY
Router Model	TP-Link TL-MR3020 v1
Firmware Version	S.M.A.R.T Customized OpenWrt 12.09 / LuCI 0.11.1 Release (0.11.1)
Kernel Version	3.3.8
Local Time	[redacted]
Uptime	1h 41m 31s
Load Average	0.10, 0.05, 0.05

Memory

Total Available	18140 kB / 29212 kB (62%)
Free	6844 kB / 29212 kB (23%)
Cached	8212 kB / 29212 kB (28%)
Buffered	3084 kB / 29212 kB (10%)

Network

IPv4 WAN Status	br-lan **Type:** dhcp **Address:** 192.168.1.6 **Netmask:** 255.255.255.0 **Gateway:** 192.168.1.1 **DNS 1:** 8.8.4.4 **DNS 2:** 8.8.8.8 **Connected:** 1h 40m 56s
Active Connections	28 / 16384 (0%)

DHCP Leases

Hostname	IPv4-Address	MAC-Address	Leasetime remaining

There are no active leases.

Wireless

Generic 802.11bgn Wireless Controller (radio0)	100% **SSID:** Fr33C@nDy **Mode:** Master **Channel:** 11 (2.462 GHz) **Bitrate:** 58.5 Mbit/s **BSSID:** D4:6E:0E[redacted] **Encryption:** None

```
eth0 6.10 Kbps 4.11 Kbps  wlan0 0.37 Kbps  0.00 Kbps vmnet8 0.00 Kbps  0.00 Kbps tun0  0.00 Kbps  0.00 Kbps        WAN-IP [redacted].109.185
```

Figure 3.19: Web Interface of smartWRT

3.4 MERGING THE FUNCTIONALITIES

Since both customized operating system instance and the hardware platform are presented, SMART code is easily deployed just like an ordinary bash script with `chmod 755` privileges for root execution.

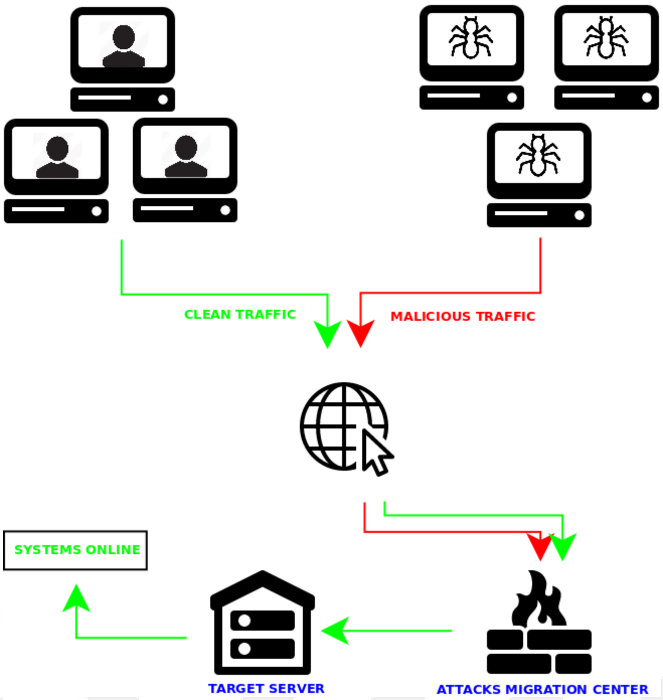


Figure 3.20: S.M.A.R.T's Topological Position

S.M.A.R.T's host device is placed as a Gateway to local network and separating the LAN and WAN access of any given network. By utilizing network concepts such as VLANs[20] or ACLs [21], S.M.A.R.T can also be utilized within the local networks but default behavior is configured as a Gateway. (Figure 3.20)

Initial installation of the SMART is not manually achieved yet its a generic update and dependency file that can be checked from a remote server in order to prevent any missing dependencies. As an intermediary software during installation and startup, Genesis.sh script runs every boot of the Operating system and is the only way to initialize S.M.A.R.T main program. By the help of this intermediary software certain check sequences and necessary software is verified. Detailed procedures and steps can be located in Code 3.4.

```
#!/bin/bash
### Needs to have an installation script.
#Default directory to store everything in one place.
source ~/smart/toolset/randomize_password.sh
source ~/smart/config/coloring_scheme.conf

#Introduce Config file

default_path=~/smart/config
default_config=$default_path/smart_def.conf
```

[20] Virtual Local Area Network

[21] Access Control List

```
config_file=$default_path/smart.conf


#Introduce Config Params
STEPS=0
TOTAL_STEPS=‘cat ∼/smart/config/lilith.list | wc -l ‘

VPN_DIREC=‘cat $config_file | grep vpn_dir | cut -d"=" -f2 | tr -d ’",’ ‘
RUN_CONFIG=‘cat $config_file | grep dont_ask | cut -d"=" -f2 | tr -d ’",’ ‘
OUTPUTS_DIR=‘cat $config_file | grep outputs_dir | cut -d"=" -f2 | tr -d ’",’ ‘
DEFAULT_PATH=‘cat $config_file | grep default_path | cut -d"=" -f2 | tr -d ’",’ ‘
STARTUP_CHECK=‘cat $config_file | grep startup_check | cut -d"=" -f2 | tr -d ’",’ ‘
EXPORT_TO_PATH=‘cat $config_file | grep export_to_path | cut -d"=" -f2 | tr -d ’",’ ‘
REQUIREMENTS_MET=‘cat $config_file | grep requirements_met | cut -d"=" -f2 | tr -d ’",’ ‘


###For debuging Purposes
#echo "$TOTAL_STEPS"
#echo "This is run_config $RUN_CONFIG"
#echo "This is OUTPUTS_DIR $OUTPUTS_DIR"
#echo "This is DEFAULT_PATH $DEFAULT_PATH"
#echo "This is REQUIREMENTS_MET $REQUIREMENTS_MET"
clear
if [[ ! $(id -u) == 0 ]]; then
    echo -e "${RLS} This script must be run as root"
    exit 1
fi

if [ "$STARTUP_CHECK" == "true" ]; then
    for pc in $(cat ∼/smart/config/lilith.list);do
        (( STEPS++ ))
        bin=‘echo "$pc" | cut -d "#" -f2‘
        echo -ne "${RLS} ${darkgray} Checking ${lightyellow}$bin${RESET} ${darkgray}[${GREEN}$STEPS${RESET}${
              darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}\r"
        sleep 0.1
        echo -ne "                                                        \r"
        if [[ -z $(which $bin) ]]; then
            echo "${RES} ${RED}Required files are not fully provided SMART will start Installing
                  Dependencies${RESET}"
            program_tbi=‘echo "$pc" | cut -d "#" -f1‘
            #echo "${RQS} ${darkgray}Checking ${lightyellow}$program_tbi${RESET}${darkgray} with ${
                  lightyellow}$bin${RESET} ${darkgray}binary name.${RESET}"
            #sed -i ’/dont_ask/c\dont_ask="false"’ $config_file
            #sed -i ’/startup_check/c\startup_check="false"’ $config_file
            #sed -i ’/requirements_met/c\requirements_met="false"’ $config_file
            echo -e "${RLS} ${RED}Unable to find ${lightyellow}$program_tbi${RESET}. ${RED}Installing it !${
                  RESET} "
            sleep 5
            apt-get -y -q=2 install $program_tbi 2>/dev/null
            echo -e "${GLS}${lightgreen}Successfully Installed${RESET} ${darkgray}[${GREEN}$STEPS${RESET}${
                  darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}${RESET}"
            (( STEPS++ ))
            #read -p "Press any key"
            #exit 0
        fi
        #(( STEPS++ ))
        echo "${GCS} ${GREEN}Located ${lightyellow}$bin${RESET} ${GREEN}[1]${RESET}${darkgray}[${GREEN}
              $STEPS${RESET}${darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}"
    done
    echo "${GLS} Succesfully Provided dependencies for SMART !"
    sed -i ’/requirements_met/c\requirements_met="true"’ $config_file
    #sed -i ’/startup_check/c\startup_check="true"’ $config_file
    echo ""
    #echo "${GCS} Startup Check is completed !"
    #sed -i ’/requirements_met/c\requirements_met="true"’ $config_file
    #STEPS=0
fi

if [ ! -d "$OUTPUTS_DIR" ]; then
    echo " ${RLS}Necessary log directories doesnt exist, Creating them."
    echo " ${GLS}Creating $OUTPUTS_DIR"
    mkdir /root/smart_db/
    echo " ${GLS}Creating $OUTPUTS_DIR/fmr"
    mkdir /root/smart__db/fmr
    echo " ${GLS}Creating $OUTPUTS_DIR/motion"
    mkdir /root/smart_db/motion
    echo " ${GLS}Creating $OUTPUTS_DIR/bugin"
    mkdir /root/smart_db/bugin
    touch /root/smart_db/motion.log
    echo " ${GLS}Directory Creation Accomplished [${GREEN}$STEPS${RESET}/${RED}$TOTAL_STEPS${RESET}]"

fi

if [ "$EXPORT_TO_PATH" == "true" ]; then
    env_path="/usr/share/smart"
    if [ ! -d "$env_path" ]; then
        echo " ${RLS} Creating Usr Share Folder"
        mkdir /usr/share/smart
        echo " ${RLS} Copying \.desktop Extension"
        cp /root/smart/preprods/startsmart.desktop /usr/share/smart
        echo " ${RLS} Copying Binary file to usr/bin"
        cp /root/smart/preprods/smart /usr/bin/
        echo " ${RLS} Copying Motion Configuration"
        cp /root/smart/motion.conf /etc/motion/
    else
        echo " ${GLS} Which Eve is functional ! Skipping ..."
    fi

```

```

fi

if [ "$RUN_CONFIG" == "false" ];
    then
    echo "Would you like to run tis initial setup next time ? ${GREEN}(T)${RESET}rue/${RED}(F)${RESET}alse ?"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read dont_ask
    case $dont_ask in
        f|F) sed -i ’/dont_ask/c\dont_ask="true"’ $config_file ;;
        t|T) sed -i ’/dont_ask/c\dont_ask="false"’ $config_file ;;
        *)
            echo "${RES}Not Valid Input Terminating !"
            exit 1
    esac
elif [[ ! -d "$DEFAULT_PATH/smart/" ]]; then
    echo "Valid Directory passed!"
fi




(( STEPS++ ))
sync_key="$(genpasswd)"
echo "${GLS} Sync Key = $sync_key"
xfce4-terminal --geometry 80x64+1111+0 --hide-menubar --hide-borders --hide-scrollbar --title=[S.M.A.R.T] --icon=/root/smart/icon.png -e /root/smart/smart.sh 2>/dev/null &
exit 0
```

Code 3.9: codes/genesis.sh

In the genesis.sh code snippet, global constants and declarations are made prior to the script. This script checks for required application and drivers and configures them if needed. Checklist which is provided in Appendix 5 either be local copy or can be updated over the web. After successful installation and fully met requirements SMART daemon is spawned to conduct its defined features.

CHAPTER 4

EVALUATION

In this section, various test environments and scenarios are explained. Both physical and computational limitations for the expected outcomes are observed and logged. Since there are countless scenarios that can be performed, must have USM features are monitored for any anomalies or malfunctioning.

Optimal network grade physical environmental necessities like a closed and a constantly cooled room is used to verify basic networking and computational performance.

Until must have functionalities are confirmed to be working, tests are deployed in a monitor mode, Where the network flows over the board are not modified yet only logged as a transparent proxy.

By utilizing multiple network interface cards, 4 different WAN access with different global IP addresses are connected to the board to identify chaotic traffic distribution what is also known as load balancing.

4.1 TEST ENVIRONMENT

Device is installed in the 42U network grade cabinet with 6x12cm fan array in a constantly cooled environment. Wall panels that are distributed inside the test building are carried with CAT6 infrastructure and terminated in the RJ-45 Patch Panel inside the network cabin. Additional Layer 2 transparent Switches are also used to multiplex the network access layer in order to reach out more than single node. (Figure 4.2)

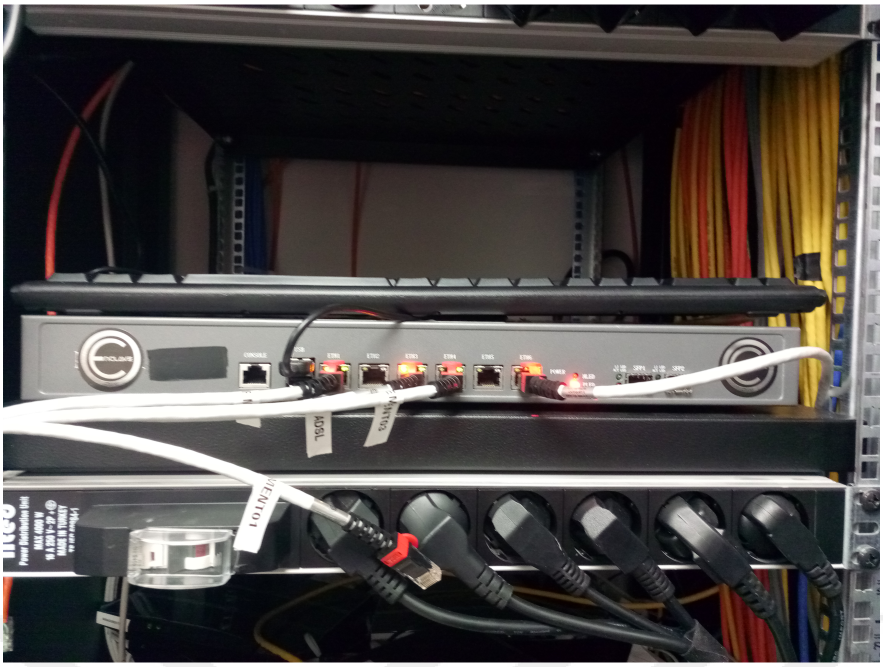


Figure 4.1: Network Cabin Installation of the Platform

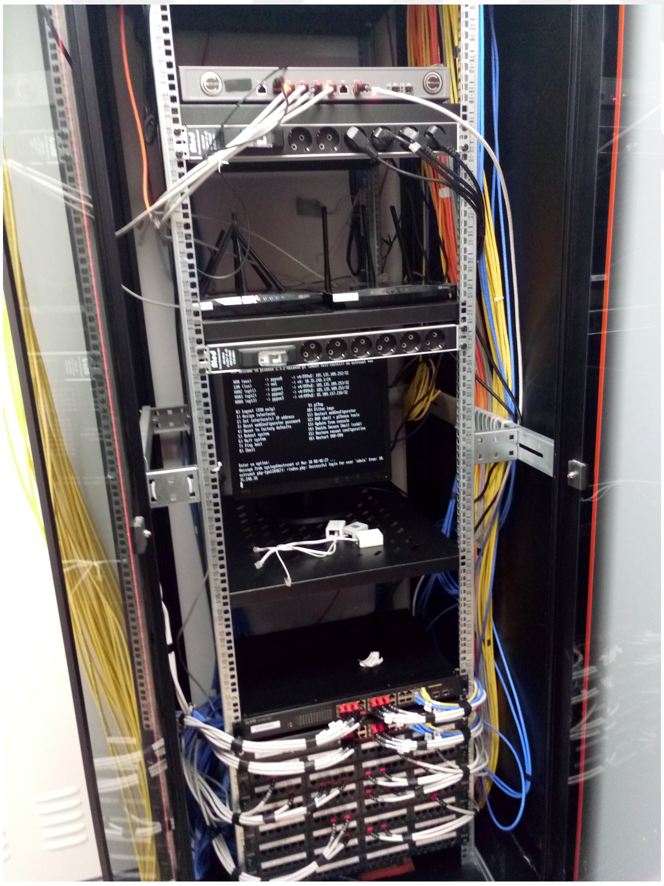


Figure 4.2: Overall Network Cabin Setup

As a WAN access 4 different 8 Mbps Download and 2 Mbps Upload speed ADSL connections. These WAN accesses later inter-connected by the help of SDN Load-Balancing implementation resulting theoretical 32Mbps/8Mbps bandwidth array. In Figure 4.1, left foremost 4 network interface cards are allocated as a Load balancing WAN interface where the Right foremost port is distributed to local network. Local network access is then distributed with the Layer 2 Switches and transmitted to the wall patch panel that is shown in the lowest mounting points of the network cabin.

4.2 STRESS TESTING

Since digital devices such as network interface cards may loose their performance over a saturated usage, necessity for constant traffic generation on the peripheral ports is emerged.

In order to improvise a flexible but not commercial traffic generator, a cluster of SoC nodes, known as Raspberry Pi 3s are connected as a single interface resulting over 10Gbps throughput by using tools like `iperf` (Figure 4.3). Specifically located Raspberry PI [1] embedded boards that are running a *NIX based operating systems are triggered to run the following application that is used to enforce flood like traffic to the target IP. This allows to conduct denial of service attacks and is helpful to collect healthy network throughput analysis that is described in Chapter 2.

4.3 MOBILE SMART TESTER

Specifically crafted embedded design, allows to perform on-site tests and pushes the collected data to S.M.A.R.T for further analysis. Open-source application known as `kismet` is used to capture monitor mode WIFI traffic and can also use

[1] https://www.raspberrypi.org/

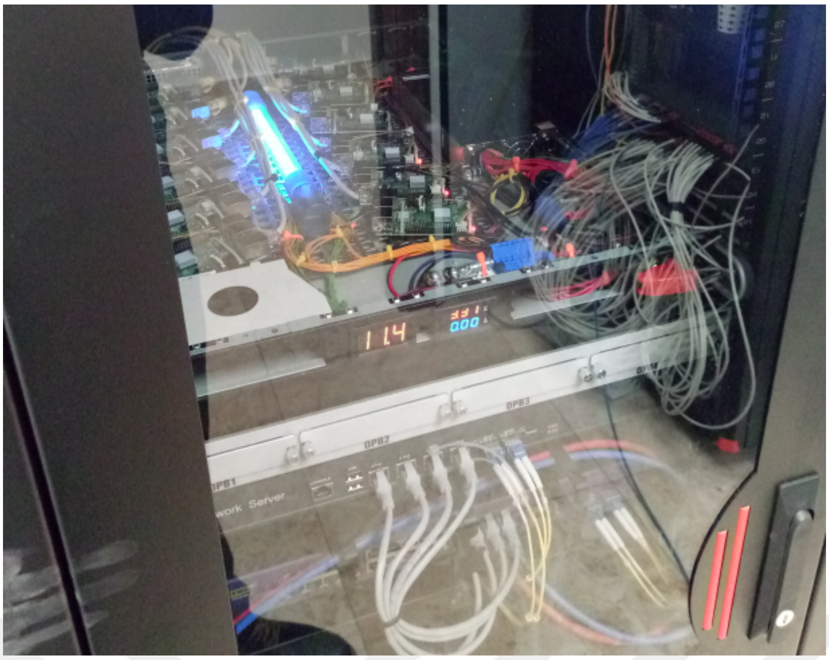


Figure 4.3: 12x Raspberry Pi 2/3 Traffic Generator Nodes

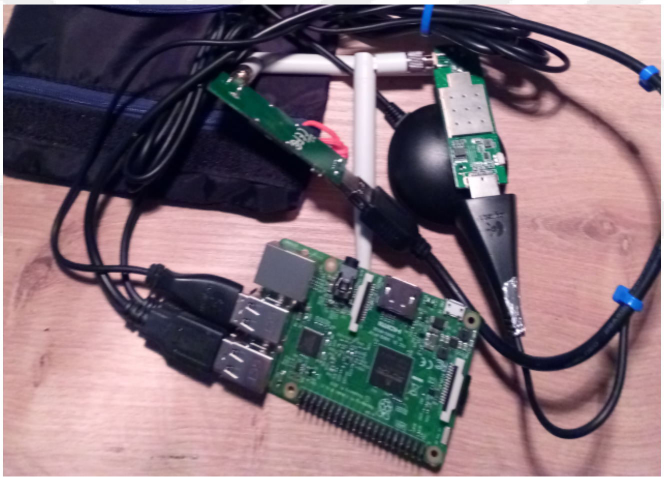


Figure 4.4: Raspberry Pi 3 with External Wireless Adapters and a GPS module

the GPS module as an external data source regarding of the captured WIFI node. Using more than one wireless adapter also provides more accurate results since wireless signal can be unstable because of the environment and the transmitters locations. (Figure 4.4)

4.4 COMPARISON

Despite the fact that there are many UTM devices available, most devices are based on defensive perspective, waiting for a breach attempt or an occurrence to detect possible threats. Where as the solution that is being presented here offers more offensive and aggressive approach towards common security problems.

Products that are available on the market are a capable of performing a predefined set of features. But in this research, a flexible platform with limited computational power can be allocated freely and/or can be stripped down from any unwanted features. This elasticity of the hardware platform also defines the efficiency on performing more specific actions such as utilizing the device just for Deep Package inspection or Cache server. On the other hand, by the advancements in Software Defined Networks and virtualization, global scale leading companies announced and released their operational functionalities as a software bundle where they let their customers decide which platform or server they want their software to be deployed. Despite the fact that they can be deployed on any virtual environment, physical access and management of the peripheral devices by these software bundles are not fully compatible with many hardware platforms.

4.5 FEATURE COMPARISON

In Table 4.1, cost/performance comparison is shown. Despite the fact that large scale organizations will require more concurrent network connections as well as higher firewall throughput. Regulatory logging requires considerable amounts of data storage area, as can be seen in table 4.1 sandbox hardware platform allows S.M.A.R.T host device to be vendor independent. Therefore end customer can easily extend the storage area of the SDN device.

Long-Term support for the commercial products is not only a free-support, rather it is a trust indicator when it comes to network equipment. Hiring different technical staff for every network equipment that are being used within the corporation, is considered to be unfeasible. Therefore, corporations which does

Product	S.M.A.R.T	FG-100D	TZ600
New Session/Sec	13700	22000	12000
Concurrent Sessions	150000	2500000	96000
Firewall Throughput	664Mbps	2500Mbps	500Mbps
Memory	4096 MB	2048 MB	1024 MB
O/S Storage	16 GB	32 GB	1 GB
Secondary Storage	256 GB	N/A	N/A
Power Usage - PEAK	90 Watts/hr	210 Watts/hr	80 Watts/hr
Power Usage - AVRG.	51 Watts/hr	63.1 Watts/hr	40 Watts/hr
Approximate Cost	450 USD (Est.)	2400 USD [Appendix5]	2100 USD [Appendix5]

Table 4.1: Performance and Feature Table.

not have it's individual IT teams, will often prefer to get their support and 7/24 services by the major manufacturers.

CHAPTER 5

CONCLUSION

In this chapter, implementation outcomes and future work that can be done is discussed. The final conclusions are derived in this chapter. There are no boundaries that can be implemented to extend functionalities, there is a computational limitation regarding the combination of these functionalities can be used simultaneously. Despite the fact that improving hardware specifications to a higher tier may seem a straight forward solution, functionalities and limitations need to be designed thoroughly before the implementation.

In Software Defined Network architectures boundaries are quite wide. Possible future implementation may include Application sandbox features. By the help of this feature any incoming executable or suspicious file can be run automatically in multiple Operating system platforms in rapid and automated fashion before allowing it to recipient. By the help of Sandbox possible internal security flaws can be prevented drastically.

Scanning available search engines such as Shodan.io it's possible to implement nature language processing features may improve detection rates for current or future data leakages.

Under the circumstance where an array of S.M.A.R.T devices distributed over the different WANs, can be utilized for conducting Distributed Denial of Service attacks as a network stress test. These test can be performed to the corresponding Local Networks in a queue fashion. By the help of this approach, magnitude of the denial of service attack can be adjusted to certain thresholds. Since the magnitude of the DDoS attacks are quite unpredictable, limitations and the behavior of the networks under these attacks are also quite unpredictable. But as a future work, software implementation and orchestration of such a test can also be included.

Physical environment sensors are also quite beneficial for monitoring physical security as well as climate of the data-center These physical sensors can be hooked up to board's general purpose input/output ports for serial communications.

Incident reporting and crisis handling functionalities can be implemented by establishing a real life Network Operation Center and monitoring the logged outputs on the live network. These logs can be analyzed within the S.M.A.R.T during low power consumption periods of work hours. System intensive analysis and possible actions can be derived during non-working hours.

In Conclusion it's certain that human-driven approach in penetration testing especially in offensive security perspective, there are always some unturned stones in an organization. These unchecked items may cause serious implications in overall security integrity. Yet most organizations limit their penetration testing procedures to once a year. There could be hundreds of newly discovered 0-Day attacks during that period of time. Therefore automating such a process with daily, weekly and monthly runs are indeed going to harden overall integrity.

By the help of honeypots not only allowing IT administrators to secure their network also deployment of these honeypots became more easy to activate.

It is clear that some conventional functionalities are under performed by their SDN implemented clones but optimization and more sophisticated implementations will close that gap in foreseeable future. Despite the fact that hardware dependency and the cost of that hardware is reduced, commercial product quality tests are reliable and trustworthy in regarding the customers perspective.

References

Agham, V. (2016). Unified threat management.

Alazab, M., Venkatraman, S., Watters, P., & Alazab, M. (2011). Zero-day malware detection based on supervised learning algorithms of api call signatures. In *Proceedings of the ninth australasian data mining conference-volume 121* (pp. 171–182).

Baker, M., Asami, S., Deprit, E., Ousterhout, J., & Seltzer, M. (1992). Non-volatile memory for fast, reliable file systems. In *Proceedings of the 5 th international conference on architectural support for programming languages and operating systems.*

Batsell, S. G., Rao, N. S., & Shankar, M. (2005). *Distributed intrusion detection and attack containment for organizational cyber security.*

Bellovin, S. M. (1989). Security problems in the tcp/ip protocol suite. *ACM SIGCOMM Computer Communication Review*, *19*(2), 32–48.

Benedikt, M. (1991). Cyberspace. In M. Benedikt (Ed.), (pp. 119–224). Cambridge, MA, USA: MIT Press. Retrieved from `http://dl.acm.org/citation.cfm?id=114772.114787`

Bovet, D. P., & Cesati, M. (2005). *Understanding the linux kernel: from i/o ports to process management.* " O'Reilly Media, Inc.".

Brown, A. L. (2004). The state of acpi in the linux kernel. In *Linux symposium* (p. 121).

Bugnion, E., Devine, S., Rosenblum, M., Sugerman, J., & Wang, E. Y. (2012). Bringing virtualization to the x86 architecture with the original vmware workstation. *ACM Transactions on Computer Systems (TOCS)*, *30*(4), 12.

Cooper, M. (2010). *Advanced bash scripting guide 5.3 volume 1* (Vol. 1). Lulu. com.

Einspruch, N. (2012). *Application specific integrated circuit (asic) technology* (Vol. 23). Academic Press.

Ericsson, G. N. (2010). Cyber security and power system communication—essential parts of a smart grid infrastructure. *IEEE Transactions on Power Delivery*, *25*(3), 1501–1507.

Fainelli, F. (2008). The openwrt embedded development framework. In *Proceedings of the free and open source software developers european meeting.*

Han, B., Gopalakrishnan, V., Ji, L., & Lee, S. (2015). Network function virtualization: Challenges and opportunities for innovations. *IEEE Communications Magazine*, *53*(2), 90–97.

Henkel, J. (2006). Selective revealing in open innovation processes: The case of embedded linux. *Research policy*, *35*(7), 953–969.

Hollabaugh, C. (2002). *Embedded linux: hardware, software, and interfacing.* Addison-Wesley Professional.

Jibao, L., Huiqiang, W., & Liang, Z. (2006). Study of network security situation awareness model based on simple additive weight and grey theory. In *Computational intelligence and security, 2006 international conference on* (Vol. 2, pp. 1545–1548).

Joshi, K., & Benson, T. (2016, Nov). Network function virtualization. *IEEE Internet Computing*, *20*(6), 7-9. doi: 10.1109/MIC.2016.112

Krishnaprasad, P. (2017). *Capturing attacks on iot devices with a multi-purpose iot honeypot* (Unpublished doctoral dissertation). INDIAN INSTITUTE OF TECHNOLOGY KANPUR.

Lee, J. W., Lim, D., Gassend, B., Suh, G. E., Van Dijk, M., & Devadas, S. (2004). A technique to build a secret key in integrated circuits for identification and authentication applications. In *Vlsi circuits, 2004. digest of technical papers. 2004 symposium on* (pp. 176–179).

Liu, J., Xiao, Y., Li, S., Liang, W., & Chen, C. P. (2012). Cyber security and privacy issues in smart grids. *IEEE Communications Surveys & Tutorials*, *14*(4), 981–997.

Love, R. (2005). *Linux kernel development (novell press).* Novell Press.

Lukasik, S. J. (2010). Why the arpanet was built. *IEEE Annals of the History of Computing*, *33*, 4-21. doi: doi.ieeecomputersociety.org/10.1109/MAHC.2010.11

Mairh, A., Barik, D., Verma, K., & Jena, D. (2011). Honeypot in network security: a survey. In *Proceedings of the 2011 international conference on communication, computing & security* (pp. 600–605).

Martin, R. A. (2008). Making security measurable and manageable. In *Military communications conference, 2008. milcom 2008. ieee* (pp. 1–9).

Martins, J., Ahmed, M., Raiciu, C., Olteanu, V., Honda, M., Bifulco, R., & Huici, F. (2014). Clickos and the art of network function virtualization. In *Proceedings of the 11th usenix conference on networked systems design and implementation* (pp. 459–473).

Mazurak, K., & Zdancewic, S. (2007). A bash: finding bugs in bash scripts. In *Proceedings of the 2007 workshop on programming languages and analysis for security* (pp. 105–114).

Mijumbi, R., Serrat, J., Gorricho, J.-L., Bouten, N., De Turck, F., & Boutaba, R. (2016). Network function virtualization: State-of-the-art and research challenges. *IEEE Communications Surveys & Tutorials*, *18*(1), 236–262.

Neuman, C. (2009). Challenges in security for cyber-physical systems. In *Dhs workshop on future directions in cyber-physical systems security* (pp. 22–24).

Petullo, M. (2010). Building custom firmware with openwrt. *Linux journal*, *2010*(196), 3.

Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., & Shenker, S. (2009). Extending networking into the virtualization layer. In *Hotnets.*

Portokalidis, G., Slowinska, A., & Bos, H. (2006). Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. In *Acm sigops operating systems review* (Vol. 40, pp.

15–27).

Provos, N., et al. (2004). A virtual honeypot framework. In *Usenix security symposium* (Vol. 173, pp. 1–14).

Ramim, M., & Levy, Y. (2006). Securing e-learning systems: A case of insider cyber attacks and novice it management in a small university. *Journal of Cases on Information Technology (JCIT)*, *8*(4), 24–34.

Senie, D., & Ferguson, P. (1998). Network ingress filtering: Defeating denial of service attacks which employ ip source address spoofing. *Network*.

Solomon, A. (2007). Linuxgym: software to automate formative assessment of unix command-line and scripting skills. *ACM SIGCSE Bulletin*, *39*(3), 353–353.

Syversen, J. (2006, March 30). *Method and apparatus for defending against zero-day worm-based attacks.* Google Patents. (US Patent App. 11/632,669)

Teodorczyk, M. (2013). Wi-fi mini honeypot. *Linux Journal*, *2013*(225), 3.

Wang, P., Wu, L., Cunningham, R., & Zou, C. C. (2010). Honeypot detection in advanced botnet attacks. *International Journal of Information and Computer Security*, *4*(1), 30–51.

Weiler, N. (2002). Honeypots for distributed denial-of-service attacks. In *Enabling technologies: Infrastructure for collaborative enterprises, 2002. wet ice 2002. proceedings. eleventh ieee international workshops on* (pp. 109–114).

Williams, J. A., & Bergmann, N. W. (2004). Embedded linux as a platform for dynamically self-reconfiguring systems-on-chip. In *Ersa'04: the 2004 international conference on engineering of reconfigurable systems and algorithms* (pp. 163–169).

Winter, J. (2008). Trusted computing building blocks for embedded linux-based arm trustzone platforms. In *Proceedings of the 3rd acm workshop on scalable trusted computing* (pp. 21–30).

Xi, R., Jin, S., Yun, X., & Zhang, Y. (2011). Cnssa: a comprehensive network security situation awareness system. In *Trust, security and privacy in computing and communications (trustcom), 2011 ieee 10th international conference on* (pp. 482–487).

Yin, X., Yurcik, W., & Slagell, A. (2005). The design of visflowconnect-ip: a link analysis system for ip security situational awareness. In *Information assurance, 2005. proceedings. third ieee international workshop on* (pp. 141–153).

Zhang, F., Zhou, S., Qin, Z., & Liu, J. (2003). Honeypot: a supplemented active defense system for network security. In *Parallel and distributed computing, applications and technologies, 2003. pdcat'2003. proceedings of the fourth international conference on* (pp. 231–235).

Zou, C. C., & Cunningham, R. (2006). Honeypot-aware advanced botnet construction and maintenance. In *Dependable systems and networks, 2006. dsn 2006. international conference on* (pp. 199–208).

APPENDIX A

This section is to provide detailed overview and technical contents of the research that is presented. Key scripts are provided in the codes section as well as necessary dependency and constants are also provided within.

CODES

```
#!/bin/bash
### Needs to have an installation script.
#Default directory to store everything in one place.
source ∼/smart/toolset/randomize_password.sh
source ∼/smart/config/coloring_scheme.conf

#Introduce Config file

default_path=∼/smart/config
default_config=$default_path/smart_def.conf
config_file=$default_path/smart.conf


#Introduce Config Params
STEPS=0
TOTAL_STEPS=‘cat ∼/smart/config/lilith.list | wc -l ‘

VPN_DIREC=‘cat $config_file | grep vpn_dir | cut -d"=" -f2 | tr -d ’",’ ‘
RUN_CONFIG=‘cat $config_file | grep dont_ask | cut -d"=" -f2 | tr -d ’",’ ‘
OUTPUTS_DIR=‘cat $config_file | grep outputs_dir | cut -d"=" -f2 | tr -d ’",’ ‘
DEFAULT_PATH=‘cat $config_file | grep default_path | cut -d"=" -f2 | tr -d ’",’ ‘
STARTUP_CHECK=‘cat $config_file | grep startup_check | cut -d"=" -f2 | tr -d ’",’ ‘
EXPORT_TO_PATH=‘cat $config_file | grep export_to_path | cut -d"=" -f2 | tr -d ’",’ ‘
REQUIREMENTS_MET=‘cat $config_file | grep requirements_met | cut -d"=" -f2 | tr -d ’",’ ‘


###For debuging Purposes
#echo "$TOTAL_STEPS"
#echo "This is run_config $RUN_CONFIG"
#echo "This is OUTPUTS_DIR $OUTPUTS_DIR"
#echo "This is DEFAULT_PATH $DEFAULT_PATH"
#echo "This is REQUIREMENTS_MET $REQUIREMENTS_MET"
clear
if [[ ! $(id -u) == 0 ]]; then
    echo -e "${RLS} This script must be run as root"
    exit 1
fi

if [ "$STARTUP_CHECK" == "true" ]; then
    for pc in $(cat ∼/smart/config/lilith.list);do
        (( STEPS++ ))
        bin=‘echo "$pc" | cut -d "#" -f2‘
        echo -ne "${RLS} ${darkgray} Checking ${lightyellow}$bin${RESET} ${darkgray}[${GREEN}$STEPS${RESET}${darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}\r"
        sleep 0.1
        echo -ne "                                                   \r"
        if [[ -z $(which $bin) ]]; then
            echo "${RES} ${RED}Required files are not fully provided SMART will start Installing Dependencies${RESET}"
            program_tbi=‘echo "$pc" | cut -d "#" -f1‘
            #echo "${RQS} ${darkgray}Checking ${lightyellow}$program_tbi${RESET}${darkgray} with ${lightyellow}$bin${RESET} ${darkgray}binary name.${RESET}"
            #sed -i ’/dont_ask/c\dont_ask="false"’ $config_file
            #sed -i ’/startup_check/c\startup_check="false"’ $config_file
            #sed -i ’/requirements_met/c\requirements_met="false"’ $config_file
            echo -e "${RLS} ${RED}Unable to find ${lightyellow}$program_tbi${RESET}. ${RED}Installing it !${RESET} "
            sleep 5
            apt-get -y -q=2 install $program_tbi 2>/dev/null
            echo -e "${GLS}${lightgreen}Successfully Installed${RESET} ${darkgray}[${GREEN}$STEPS${RESET}${darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}${RESET}"
            (( STEPS++ ))
            #read -p "Press any key"
            #exit 0
        fi
        #(( STEPS++ ))
        echo "${GCS} ${GREEN}Located ${lightyellow}$bin${RESET} ${GREEN}[1]${RESET}${darkgray}[${GREEN}$STEPS${RESET}${darkgray}/${RED}$TOTAL_STEPS${darkgray}]${RESET}"
    done
    echo "${GLS} Succesfully Provided dependencies for SMART !"
    sed -i ’/requirements_met/c\requirements_met="true"’ $config_file
    #sed -i ’/startup_check/c\startup_check="true"’ $config_file
    echo ""
    #echo "${GCS} Startup Check is completed !"
    #sed -i ’/requirements_met/c\requirements_met="true"’ $config_file
    #STEPS=0
fi

if [ ! -d "$OUTPUTS_DIR" ]; then
    echo " ${RLS}Necessary log directories doesnt exist, Creating them."
    echo " ${GLS}Creating $OUTPUTS_DIR"
    mkdir /root/smart_db/
    echo " ${GLS}Creating $OUTPUTS_DIR/fmr"
    mkdir /root/smart__db/fmr
    echo " ${GLS}Creating $OUTPUTS_DIR/motion"
    mkdir /root/smart_db/motion
    echo " ${GLS}Creating $OUTPUTS_DIR/bugin"
    mkdir /root/smart_db/bugin
    touch /root/smart_db/motion.log
    echo " ${GLS}Directory Creation Accomplished [${GREEN}$STEPS${RESET}/${RED}$TOTAL_STEPS${RESET}]"

fi

if [ "$EXPORT_TO_PATH" == "true" ]; then
    env_path="/usr/share/smart"
    if [ ! -d "$env_path" ]; then
        echo " ${RLS} Creating Usr Share Folder"
        mkdir /usr/share/smart
        echo " ${RLS} Copying \.desktop Extension"
        cp /root/smart/preprods/startsmart.desktop /usr/share/smart
        echo " ${RLS} Copying Binary file to usr/bin"
        cp /root/smart/preprods/smart /usr/bin/
        echo " ${RLS} Copying Motion Configuration"
        cp /root/smart/motion.conf /etc/motion/
    else
        echo " ${GLS} Which Eve is functional ! Skipping ..."
    fi


fi

if [ "$RUN_CONFIG" == "false" ];
    then
    echo "Would you like to run tis initial setup next time ? ${GREEN}(T)${RESET}rue/${RED}(F)${RESET}alse ?"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read dont_ask
    case $dont_ask in
        f|F) sed -i ’/dont_ask/c\dont_ask="true"’ $config_file ;;
        t|T) sed -i ’/dont_ask/c\dont_ask="false"’ $config_file ;;
        *)
            echo "${RES}Not Valid Input Terminating !"
            exit 1
    esac
elif [[ ! -d "$DEFAULT_PATH/smart/" ]]; then
    echo "Valid Directory passed!"
fi




(( STEPS++ ))
sync_key="$(genpasswd)"
echo "${GLS} Sync Key = $sync_key"
xfce4-terminal --geometry 80x64+1111+0 --hide-menubar --hide-borders --hide-scrollbar --title=[S.M.A.R.T] --
      icon=/root/smart/icon.png -e /root/smart/smart.sh 2>/dev/null &
exit 0
```

Code 5.1: codes/genesis.sh

```
#!/bin/bash -
#title          :smart.sh
#description    :SMART NFV APP
#author         :Mert Kilic
#date           :17-08-17
#version        :v0.98d beta(Non-Release)(PoC)(UNDER DEV)
#usage          :./smart.sh
#notes          :
#bash_version   :4.4.0(1)-release
#=================================================================================#
### | Declerations       | ===================================================#
LSB=/usr/bin/lsb_release
zenity="/usr/bin/zenity"
# Color Decleratins.

declare -r RED=$ESC"31m"
declare -r BLUE=$ESC"34m"
declare -r RESET=$ESC"39m"
declare -r GREEN=$ESC"32m"
declare -r LBLUE=$ESC"36m"

declare -r RES=${RED}"[!]"${RESET}
declare -r RLS=${RED}"[*]"${RESET}
declare -r RQS=${RED}"[?]"${RESET}
declare -r BES=${BLUE}"[!]"${RESET}
declare -r BLS=${BLUE}"[*]"${RESET}
declare -r BQS=${BLUE}"[?]"${RESET}
declare -r GES=${GREEN}"[!]"${RESET}
declare -r GLS=${GREEN}"[*]"${RESET}
declare -r GQS=${GREEN}"[?]"${RESET}

declare -r dim=‘echo -en "\e[2m"‘
declare -r bold=‘echo -en "\e[1m"‘
declare -r blink=‘echo -en "\e[5m"‘
declare -r normal=‘echo -en "\e[0m"‘
declare -r hidden=‘echo -en "\e[8m"‘
declare -r rsmartrse=‘echo -en "\e[7m"‘
declare -r underline=‘echo -en "\e[4m"‘
declare -r strickthru=‘echo -en "\e[9m"‘
```

```

AQUA=‘echo -en "\e[46m"‘
aqua=‘echo -en "\e[36m"‘
GRAY=‘echo -en "\e[47m"‘
gray=‘echo -en "\e[37m"‘
BLACK=‘echo -en "\e[40m"‘
black=‘echo -en "\e[30m"‘
WHITE=‘echo -en "\e[107m"‘
white=‘echo -en "\e[97m"‘
ORANGE=‘echo -en "\e[43m"‘
orange=‘echo -en "\e[33m"‘
PURPLE=‘echo -en "\e[45m"‘
purple=‘echo -en "\e[35m"‘
DEFAULT=‘echo -en "\e[49m"‘
default=‘echo -en "\e[39m"‘
DARKGRAY=‘echo -en "\e[100m"‘
darkgray=‘echo -en "\e[90m"‘
LIGHTRED=‘echo -en "\e[101m"‘
lightred=‘echo -en "\e[91m"‘
LIGHTBLUE=‘echo -en "\e[104m"‘
lightblue=‘echo -en "\e[94m"‘
LIGHTAQUA=‘echo -en "\e[106m"‘
lightaqua=‘echo -en "\e[96m"‘
LIGHTGREEN=‘echo -en "\e[102m"‘
lightgreen=‘echo -en "\e[92m"‘
LIGHTYELLOW=‘echo -en "\e[103m"‘
lightyellow=‘echo -en "\e[93m"‘
LIGHTPURPLE=‘echo -en "\e[105m"‘
lightpurple=‘echo -en "\e[95m"‘


source /root/smart/toolset/exifhelper.sh
source /root/smart/toolset/randomize_password.sh
source /root/smart/toolset/autogen.sh
source /root/smart/toolset/4CA.sh

### | Functions           | =================================================#

# {*} Function status = Finished
# {*} Function Desc = Timestamp File Name Compatible
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = timestamp function can be called as is.
function timestamp() {
    date +’%D_%T’| tr :/ _
}

# {*} Function status = Finished
# {*} Function Desc = Changes Terminal Title
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function set_ttl() {
    echo -ne ’\033]2;’$1’\007’
}

# {*} Function status = Finished
# {*} Function Desc = Fluid Menu Animation
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function clear_screen() {
    printf "\033c"
}

# {*} Function status = Finished
# {*} Function Desc = $1-> Message (optional)
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function pause(){

    local message="$@"
    [ -z $message ] && message="Press ${lightyellow}[Enter]${normal} key to continue..."
    read -p "$message" readEnterKey
    clear_screen

}

# {*} Function status = Primitive
# {*} Function Desc = Progress indicator of any given job
# {*} Function To do = Control-Flow Mechanism
# {*} Priority Stat = Least[ ]Avg[X]Medium[ ]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = Usage progress_indicator <task> read -p "# " proc_name ; progress_indicator "
      $proc_name"
function progress_indicator(){

    if [ -z "$1" ]                         # Is parameter #1 zero length?
    then
      echo "-Parameter #1 is zero length.-" # Or no parameter passed.
    else
      echo "-Parameter #1 is \"$1\".-"
    fi
    echo
    pid_pless="$1" # Process Id of the previous running command
    pid=$(pidof $1)
    echo -e "pid is $pid"
    spin=’-\|/’
    i=0
    while kill -0 $pid 2>/dev/null
    do
        i=$(( (i+1) %4 ))
        printf "\r[${spin:$i:1}] Still Runnin’"
        sleep .1
    done
    pause

}


# [-----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-----------------------------------------------------------------------------------------]
function loading_indicator(){

    MAX=${1:-11}
    TIME="${2:-0.08}"
    TL="${3:-[}"
    S="${4:-#####}"
    TR="${5:-]}"
    while true; do
        R=0
        while [ $R -lt $MAX ]; do
            RSP=$(($MAX - $R ))
            if [ $RSP -gt $MAX ]; then RSP=$MAX ; fi
            LSP=$(($MAX - ${RSP}))
            echo -n "$TL"
            for l in $(seq 1 $LSP); do
                echo -n " "
            done
            echo -n $S
            for r in $(seq 1 $RSP); do
                echo -n " "
            done; echo -ne "$TR\r"
            sleep $TIME ; ((R++))
        done
        while [ $R -ne 0 ]; do
            RSP=$(($MAX - $R ))
            if [ $RSP -ge $MAX ]; then RSP=$MAX ; fi
            LSP=$(($R + 0 ))
            if [ $LSP -lt 0 ]; then LSP=0 ; fi
            echo -n "$TL"
            for l in $(seq 1 $R); do
                echo -n " "
            done
            echo -n $S
            for r in $(seq 1 $RSP); do
                echo -n " "
            done; echo -ne "$TR\r"
            sleep $TIME; ((R--))
        done
    done

}


# {*} Function status = Finished
# {*} Function Desc = Progress indicator for given sleep time
# {*} Function To do = Control-Flow Mechanism
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = Usage sleep_indicator <seconds>"
function sleep_indicator(){

    i=0
    spin=’-\|/’
    secs=$(($1 * 1))
    custom_msg=$2
    while [ $secs -gt 0 ]; do
            i=$(( (i+1) %4 ))
            printf " \r%s[${spin:$i:1}]%s $custom_msg’" "${RED}" "${RESET}"
            sleep .1
            sleep 1
        : $((secs--))
    done

}

# {*} Function status = Finished
# {*} Function Desc = sha256 Checksum and hash anythin’
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_sha256() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        sha256sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

```

```
}

# {*} Function status = Finished
# {*} Function Desc = sha512 Checksum and hash anythin'
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_sha512() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        sha512sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

}

# {*} Function status = Finished
# {*} Function Desc = sha384 Checksum and hash anythin'
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_sha384() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        sha384sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

}

# {*} Function status = Finished
# {*} Function Desc = sha224 Checksum and hash anythin'
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_sha224() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        sha224sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

}

# {*} Function status = Finished
# {*} Function Desc = sha1 Checksum and hash anythin'
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_sha160() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        sha1sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

}

# {*} Function status = Finished
# {*} Function Desc = md5 Checksum and hash anythin'
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_md5sam() {

    file_to_hash=$1
    if [ ! -z "$file_to_hash" ];
        then
        md5sum $file_to_hash | cut -d " " -f1
    else
        echo "Nothing Selected !"
    fi

}

# {*} Function status = Finished
# {*} Function Desc = Summary Of All Hash Functions
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_em_all() {

    auxillary=$1
    custom_data=$2
    if [ "$auxillary" == "all" ];
        then
        file_to_hash=$(pick_single_file)
        if [ ! -z "$file_to_hash" ];
            then

            md5sam=‘hash_md5sam "$file_to_hash"‘
            write_header "This is MD5Sum"
            printf "\v%s\n" "$md5sam"
            sha160=‘hash_sha160 "$file_to_hash"‘
            write_header "This is SHA160"
            printf "\v%s\n" "$sha160"
            sha224=‘hash_sha224 "$file_to_hash"‘
            write_header "This is SHA224"
            printf "\v%s\n" "$sha224"
            sha256=‘hash_sha256 "$file_to_hash"‘
            write_header "This is SHA256"
            printf "\v%s\n" "$sha256"
            sha384=‘hash_sha384 "$file_to_hash"‘
            write_header "This is SHA384"
            printf "\v%s\n" "$sha384"
            sha512=‘hash_sha512 "$file_to_hash"‘
            write_header "This is SHA512"
            printf "\v%s\n" "$sha512"

        else
            echo "Nothing Selected !"
        fi
    fi
    if [ "$auxillary" == "manual" ];
        then
            file_to_hash=$custom_data
            if [ ! -z "$file_to_hash" ];
                then
                    md5sam=‘hash_md5sam "$file_to_hash"‘
                    write_header "This is MD5Sum"
                    printf "\v%s\n" "$md5sam"
                    sha160=‘hash_sha160 "$file_to_hash"‘
                    write_header "This is SHA160"
                    printf "\v%s\n" "$sha160"
                    sha224=‘hash_sha224 "$file_to_hash"‘
                    write_header "This is SHA224"
                    printf "\v%s\n" "$sha224"
                    sha256=‘hash_sha256 "$file_to_hash"‘
                    write_header "This is SHA256"
                    printf "\v%s\n" "$sha256"
                    sha384=‘hash_sha384 "$file_to_hash"‘
                    write_header "This is SHA384"
                    printf "\v%s\n" "$sha384"
                    sha512=‘hash_sha512 "$file_to_hash"‘
                    write_header "This is SHA512"
                    printf "\v%s\n" "$sha512"
                else
                    echo "Nothing Selected !"
            fi
    fi
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Hash Oil’
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function hash_oil() {

    echo "+-----------------------------------------------------------------------------+"
    echo "#           [1) Checksum File | 2) Manual Input | 3) Terminate]      |"
    echo "+-----------------------------------------------------------------------------+"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read t
        case $t in
        1)  hash_em_all "all" ;;
        2)  temp_file=$(make_temp_file); read -p "Input Text=" custom_data ; echo "$custom_data" >>
             $temp_file ; hash_em_all "manual" "$temp_file" ;;
        3)  echo " Terminated" ; pause ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
            pause
    esac

}

# {*} Function status = Finished
# {*} Function Desc = Generic mktemp func
# {*} Function To do =
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function make_temp_file() {
    #TFILE="/tmp/$(basename $0).$$.tmp"
    #$TFILE
    #echo "$TFILE"
    mktemp
}


# {*} Function status = Finished
# {*} Function Desc = Check all rules in IPTABLES
# {*} Function To do = Generic set of rules to control iptables
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function iptables_check_rules() {
    iptables -L -v
    pause
}

# {*} Function status = Not Started
# {*} Function Desc = Check any given port activity within iptables
# {*} Function To do = Generic set of rules to control iptables
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function iptables_check_port() {
    echo "iptables_check_port"
}

# {*} Function status = Not Started
# {*} Function Desc = Check any given ip activity within iptables
# {*} Function To do = None
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[X]
# {*} Note/Bugs/Usg = None
function iptables_check_ip() {
    echo "iptables_check_ip"
}

# {*} Function status = Not Started
# {*} Function Desc = Ban Single IP adress for any egress/ingress comm.
# {*} Function To do = iptables mambojambo
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function ip_ban() {
    echo " ban ip adress "
}

# {*} Function status = Not Started
# {*} Function Desc =
# {*} Function To do = None
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function port_forward() {
    echo "1" > /proc/sys/net/ipv4/ip_forward
}


# {*} Function status = Beta Usage
# {*} Function Desc = Clear exif data off of .jpg and .jpeg files
# {*} Function To do = Functionality to add entire folder and detect all files that matches the extension || has exif data
# {*} Priority Stat = Least[ ]Avg[ ]Medium[ ]Ab.Avg[X]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function exif_tools(){

    local t
    echo "+----------------------------------------------------------------------------+"
    echo "#          [1) View Exif Data | 2) Clear Exif Data | 3)Terminate]   |"
    echo "+----------------------------------------------------------------------------+"
    #echo -e "What to do now; 1) Check Browser | 2) New Identity | 3) Kill Tor "
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read t
        case $t in
        1)  view_exif ;;
        2)  clear_exif ;;
        3)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
            pause
    esac

}

# [-----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-----------------------------------------------------------------------------------------]
function view_exif(){
    file_to_exif="$(pick_single_file)"
    exif_view "$file_to_exif"
    pause
}

# [-----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-----------------------------------------------------------------------------------------]
function clear_exif(){
    file_to_clear="$(pick_single_file)"
    exif_clean "$file_to_clear"
    pause
}
```

```


# [----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [----------------------------------------------------------------------------------------]
function one_time_pad(){
    xfce4-terminal --geometry 56x24+530+0 --hide-menubar --zoom=0.80 -x ∼/smart/toolset/otp.sh 2>/dev/null &
    pause
}

# {*} Function Stat = Finished
# {*} Function Desc = Regex for correct IPv4 definition before passing to funcs.
# {*} Function ToDo = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function validate_ip(){

    local ip=$1
    local stat=1
    if [[ $ip =∼ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]]; then
        OIFS=$IFS
        IFS='.'
        ip=($ip)
        IFS=$OIFS
        [[ ${ip[0]} -le 255 && ${ip[1]} -le 255 \
            && ${ip[2]} -le 255 && ${ip[3]} -le 255 ]]
        stat=$?
    fi
    return $stat

}

# {*} Function status = Finished
# {*} Function Desc = Peeling off the Onion
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function start_tor(){


    local t
    if [ -z ‘pidof tor‘ ];
        then
            #tor 2>/dev/null & ### -> For Debugging TOR Connection
            tor --quiet 2>/dev/null &
            echo "${GLS} Tor Has started !"
            echo "${GLS} Waiting for link!"
            sleep_indicator "23" "Handshaking With The Onion Routing"
            echo " ${GREEN}Completed ! ${RESET} "
            pause
        else
            echo "Tor is running Please, Proceed.."
            tor_ops
    fi

}


# [----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [----------------------------------------------------------------------------------------]
function tor_ops(){

    echo "+---------------------------------------------------------------------------+"
    echo "#          [1) Check Browser | 2) New Identity | 3) Kill Tor]       |"
    echo "+---------------------------------------------------------------------------+"
    #echo -e "What to do now; 1) Check Browser | 2) New Identity | 3) Kill Tor "
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read t
        case $t in
        1)  check_browser ;;
        2)  new_identity ;;
        3)  kill_tor ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
            pause
    esac

}

# {*} Function status = Finished
# {*} Function Desc = Checks if Tor is running in bg
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_tor(){

    pidof tor > /dev/null && echo "Tor Is Running !"
    pidof tor > /dev/null || echo "Tor Is NOT Running !"
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Sends Interrupt to TOR service for a new identity
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function new_identity() {
    pidof tor | xargs sudo kill -HUP 2>/dev/null
    pause
}

# {*} Function status = Finished
# {*} Function Desc = Kills BG Tor Process
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function tor_killa(){
    kill -9 `pidof tor`
}

# {*} Function status = Finished
# {*} Function Desc = Kills Tor Running in the Background
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function kill_tor() {

    read -p "Are you sure want to viciously kill tor ? (y/n)" chc0
        case $chc0 in
            y|Y ) usg=1 ;;
            n|N ) usg=0 ;;
            * ) echo "No Input Provided";;
        esac
    if [ "$usg" == "1" ];
        then
        kill -9 `pidof tor` 2>/dev/null
        if [ -z `pidof tor` ];
            then
            echo "Tor Has been disconnected *(ENFORCED)*"
            hev "deactivated"
            notify-send "Tor Has been disconnected *(ENFORCED)*"
        else
            read -p "Process is still running ! Try Again ? (y/n)" yn
            echo "Process is still running "
                case "$yn" in
                    y|Y ) ;;
                    n|N ) clear_screen ; main_menu ;;
                    * ) echo "No Input Provided";;
                esac
        fi
    fi
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Display Public and Proxychained IP
# {*} Function To do = Maybe add michealbay effects
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = Could be better with a blocking operation
function check_GIP() {

    PUBLIC_IP=$(wget http://ipecho.net/plain -O - -q 2>/dev/null)
    echo -n "${RED}Public IP${RESET} = "
    echo $PUBLIC_IP | cut -d" " -f 3
    DNSLEAKTEST=$(curl https://www.dnsleaktest.com 2>/dev/null | grep "Hello" | cut -d">" -f 2 | cut -d " " -f2 | cut -d"<" -f1)
    echo -n "${RED}DNS Query IP${RESET} = "
    echo $DNSLEAKTEST
    Proxy_IP=$(proxychains wget http://ipecho.net/plain -O - -q 2>/dev/null)
    echo -n "${GREEN}Proxyd IP${RESET} = "
    echo $Proxy_IP | cut -d" " -f 3
    PROXYDNSLEAKTEST=$(proxychains curl https://www.dnsleaktest.com 2>/dev/null | grep "Hello" | cut -d">" -f 2 | cut -d " " -f2 | cut -d"<" -f1)
    echo -n "${GREEN}ProxyDNS IP${RESET} = "
    echo $PROXYDNSLEAKTEST
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Remotely Check proxy access
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_browser() {

    echo -ne " Waiting for ${lightyellow}ESTABLISHED${normal} signal ...\033[0K\r"
    reta=`curl --socks5 localhost:9050 --socks5-hostname localhost:9050 -s https://check.torproject.org/ | cat | grep -m 1 Congratulations | xargs`
    if [ ! -z "$reta" ];
        then
        echo " Successfully ${GREEN}ESTABLISHED${RESET} the Link with TOR"
    else
```

```
        echo " Connection was ${RED}NOT${RESET} made !"
    fi
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Checks if Tor Installed
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_tor_installed() {

  echo -e "Checking if Tor is installed...\n"
  TOR="/etc/init.d/tor"
  if [ -f $TOR ];
    then
    echo -e "Tor is Installed!\n"
    echo -e "Starting Tor :-)\n"
    systemctl start tor
  else
    echo -e "Tor is not installed! apt-get update and then apt-get install tor\n"
    exit
  fi

}

# {*} Function status = Finished
# {*} Function Desc = Check if Proxychains Installed
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_proxychains_installed() {

  echo -e "Checking if Proxychains is installed...\n"
  PC="/etc/proxychains.conf"
  if [ -f $PC ];
    then
    echo -e "Proxychains is Installed!\n"
  else
    echo -e "Proxychains is not installed! apt-get update and then apt-get install proxychains\n"
    exit
  fi

}

# {*} Function status = Beta
# {*} Function Desc = Cover Local Tracks a bit
# {*} Function To do = None
# {*} Priority Stat = Least[ ]Avg[X]Medium[ ]Ab.Avg[X]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function cover_tracks() {

  echo "Follow me =)" > /var/log/auth.log
  history -c
  history -w
  echo "I am sorry =)" > ~/.bash_history
  rm ~/.bash_history -rf
  history -c
  history -w

}

# {*} Function status = To be Tested
# {*} Function Desc = Man gotta protect himself right ?
# {*} Function To do = None
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function go_turtle() {
 # allow only 1.1.1.0/24 and ports 80,443 and log drops to /var/log/messages

  iptables -A INPUT -s 1.1.1.0/24 -m state --state RELATED,ESTABLISHED,NEW -p tcp -m multiport --dports
        80,443 -j ACCEPT
  iptables -A INPUT -i eth0 -m state --state RELATED,ESTABLISHED,NEW -j ACCEPT
  iptables -A INPUT DROP
  iptables -A OUTPUT -o eth0 -j ACCEPT
  iptables -A INPUT -i lo -j ACCEPT
  iptables -A OUTPUT -o lo -j ACCEPT
  iptables -N LOGGING
  iptables -A INPUT -j LOGGING
  iptables -A LOGGING -m limit --limit 4/min -J LOG --log_prefix "SMART_DROPPED_CONN "
  iptables -A LOGGING -j DROP

}

# {*} Function status = Finished
# {*} Function Desc = Spoof Mac Adress for any interface
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function change_mac() {

    echo ""
    echo "+----------------------------------------------------------------------------+"
    echo "|                    [Available Interfaces Below]                   |"
    echo "+----------------------------------------------------------------------------+"
    echo -ne "${GLS} "
    echo ‘ifconfig | grep flags | cut -d ":" -f1‘
    echo "+----------------------------------------------------------------------------+"
    read -p "Which Interface to change Mac ? ( eth0 | wlan0 | tap0 ) =" chcInf
    if [ -z "$chcInf" ];
        then
        echo "${RLS} No Interface Selected !"
    else
        echo -n "Current MAC address for that Device is = "
        curr_mac=‘ifconfig $chcInf | grep ether | cut -d " " -f 10‘
        echo "$curr_mac"
        echo "ALL YOUR CONNECTION WILL BE INTERRUPTED"
        read -p "(1)Randomize 2)Trusted OID 3)Back to Defaults F)Terminate? = " ce
        case "$ce" in
            1) sudo ifconfig $chcInf down; sudo macchanger -r $chcInf; sudo ifconfig $chcInf up;;
            2) sudo ifconfig $chcInf down; sudo macchanger -e $chcInf; sudo ifconfig $chcInf up;;
            3) sudo ifconfig $chcInf down; sudo macchanger -p $chcInf; sudo ifconfig $chcInf up;;
            f|F ) echo " Terminated !" ;;
            * ) echo "No Input Provided";;
        esac
    fi
    pause

}



# {*} Function status = Finished
# {*} Function Desc = SMB Null Session Checker
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function nullmein(){

    if [ -z "$1" ]; then
        echo "[*] Try SMB Null Session for specific ip or range"
        echo "[*] Usage : $0 <file_to_read>"
    exit 0
    fi

    source_file=$1;

    for ips in $(cat $source_file); do
        printf "Scanning for Null Session @ %s\n""$ips"
        output=‘bash -c "echo ’srvinfo’ | rpcclient $ips -U%"‘
        echo $output
    done

}

# {*} Function status = Finished
# {*} Function Desc = Remote Information Sub-Menu Items
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function remote_menu(){

    local c
    clear_screen
    figlet -ctf small "S.M.A.R.T"
    echo ""
    echo "+----------------------------------------------------------------------------+"
    echo "|                          [Auxillary Toolset]                       |"
    echo "+----------------------------------------------------------------------------+"
    echo "     [0x01] - Zenity Ping           |  [0x09] - [smarttool]Ping sweep  "
    echo "     [0x02] - Zenity Whois          |  [0x1A] - [smarttool]Exiftools   "
    echo "     [0x03] - xxxxxxxxxxxxxxxxxxxxx |  [0x1B] - [smarttool]Generate Bash "
    echo "     [0x04] - xxxxxxxxxxxxxxxxxxxxx |  [0x1C] - [smarttool]4CHA        "
    echo "     [0x05] - xxxxxxxxxxxxxxxxxxxxx |  [0x1D] - Extract File Archives  "
    echo "     [0x06] - xxxxxxxxxxxxxxxxxxxxx |  [0x1E] - TCPTRACK Interface     "
    echo "     [0x07] - xxxxxxxxxxxxxxxxxxxxx |  [0x1F] - SHA256 CheckSum Files  "
    echo "     [0x08] - xxxxxxxxxxxxxxxxxxxxx |  [0xFF] - Back To Main Menu      "
    echo "+----------------------------------------------------------------------------+"
    echo ""
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read c
    case $c in
        1)  zen_ping ;;
        2)  zen_whois ;;
        3)  make_temp_file ;;
        4)  echo "Custom Data" ;;
        5)  echo "Custom Data" ;;
        6)  echo "Custom Data" ;;
        7)  echo "Custom Data" ;;
        8)  echo "Custom Data" ;;
        9)  ping_sweep ;;
        1A) exif_tools ;;
        1a) exif_tools ;;
        1B) generate_shell ; pause ;;
        1b) generate_shell ; pause;;
        1C) 4cha_main ; pause ;;
        1c) 4cha_main ; pause ;;
        1D) extract_file ;;
        1d) extract_file ;;
        1E) display_tcptrack ;;
        1e) display_tcptrack ;;
        1f) hash_oil ;;
        1F) hash_oil ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
            pause
    esac

}

# {*} Function status = Finished
# {*} Function Desc = Information Sub-Menu Items
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function info_menu(){

    local c
  clear_screen
    figlet -ctf small "S.M.A.R.T"
    echo ""
    echo "+----------------------------------------------------------------------------+"
    echo "|                          [Information Menu]                        |"
    echo "+----------------------------------------------------------------------------+"
    echo "       [0x01] - Operating system info | [0x09] - Check Proxy GIP "
    echo "       [0x02] - Hostname and dns info | [0x1A] - Check USB Syslog "
    echo "       [0x03] - Network info        |   [0x1B] - Check NT OEM keys "
    echo "       [0x04] - Who is online       |   [0x1C] - Common Ports List "
    echo "       [0x05] - Last logged in users |  [0x1D] - Iptables Rules Sum "
    echo "       [0x06] - Free/used memory info |  [0x1E] - Iptables Check IPP "
    echo "       [0x07] - Watch Netstat Ops   |   [0x1F] - Iptables Check Port"
    echo "       [0x08] - Check TOR Status    |   [0xFF] - Back To Main Menu "
    echo "+----------------------------------------------------------------------------+"
    echo ""
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read c
    case $c in
        1)  os_info ;;
        2)  host_info ;;
        3)  net_info ;;
        4)  user_info "who" ;;
        5)  user_info "last" ;;
        6)  mem_info ;;
    7)  display_netstat ;;
        8)  check_tor ;;
        9)  check_GIP ;;
        1A) check_usb_log ;;
        1a) check_usb_log ;;
        1B) check_oem_key ;;
        1b) check_oem_key ;;
        1C) common_portlist ;;
        1c) common_portlist ;;
        1D) iptables_check_rules ;;
        1d) iptables_check_rules ;;
        1E) iptables_check_ip ;;
        1e) iptables_check_ip ;;
        1f) iptables_check_port ;;
        1F) iptables_check_port ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
            pause
    esac

}

# {*} Function status = Finished
# {*} Function Desc = Writes a Header with passed arg. $1 is the Message
# {*} Function To do = Change fckin echoes to printf
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = Usage : write_header " System information "
function write_header(){

    local h="$@"
    echo "+----------------------------------------------------------------------------+"
    echo "#[${h}]"
    echo "+----------------------------------------------------------------------------+"

}

# {*} Function status = Finished
# {*} Function Desc = Display a list of users currently/recently logged on
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function os_info(){

    write_header " System information "
    echo "Operating system : $(uname)"
    [ -x $LSB ] && $LSB -a || echo "$LSB command is not insalled (set \$LSB variable)"
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Get info about host such as dns, IP, and hostname
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function host_info(){

    local dnsips=$(sed -e '/^$/d' /etc/resolv.conf | awk '{if (tolower($1)=="nameserver") print $2}')
    write_header " Hostname and DNS information "
    echo "Hostname : $(hostname -s)"
    echo "DNS domain : $(hostname -d)"
    echo "Fully qualified domain name : $(hostname -f)"
    echo "Network address (IP) : $(hostname -i)"
    echo "DNS name servers (DNS IP) : ${dnsips}"
    pause

}

# {*} Function status = Finished
# {*} Function Desc = Network inferface and routing info
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function net_info(){

    devices=$(netstat -i | cut -d" " -f1 | egrep -v "^Kernel|Iface|lo")
    write_header " Network information "
    echo "Total network interfaces found : $(wc -w <<< ${devices})"
    echo "*** IP Addresses Information ***"
    ip -4 address show
    echo "+-----------------------------------------------------------------------------+"
    echo "|                          [Network Routing]                          |"
    echo "+-----------------------------------------------------------------------------+"
    netstat -nr
    echo "+-----------------------------------------------------------------------------+"
    echo "|                     [Interface traffic information]                |"
    echo "+-----------------------------------------------------------------------------+"
    netstat -i
    pause

}


# [---------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [---------------------------------------------------------------------------------------]
function check_inet_connectivity(){

    res=`ping -c 1 -w 1 8.8.8.8 | grep ttl`
    if [ -z "$res" ];
        then
        echo "Internet Connection is ${RED}DOWN${RESET}"
        exit 0
    fi

}


# [---------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [---------------------------------------------------------------------------------------]
function check_dependencies_file(){

    if [ ! -f "$dependencies" ];
        then
        check_inet_connectivity
        curl https://pastebin.com/raw/fk7JdRb4 > /home/pi/zulfikar/dependencies.list
        exit 0
    fi

}

#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] = !R
#[Function Note] = Code Needs an Improvement on Config builder as well.
function kill_app(){

    app_name=$1
    app_cmd="pidof $app_name"
    if [ ! -z `$app_cmd` ];
        then
        sudo kill -9 `$app_cmd` 2>/dev/null
    else
        echo "$app_name Is Not Running "
    fi

}
#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] = !R
#[Function Note] = Code Needs an Improvement on Config builder as well.
function start_app(){

    app_name=$1
    app_cmd="pidof $app_name"
    if [ -z ‘$app_cmd‘ ];
        then
        sudo $app_name 2>/dev/null &
    else
        echo " $app_name is Already Running "
    fi

}
#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] = !R
#[Function Note] = Code Needs an Improvement on Config builder as well.
function start_service(){

    service_name=$1
    service_cmd="service $service_name start"
    service_status=$(check_service $service_name)
    if [ "$service_status" == "inactive" ];
        then
        sudo $service_cmd 2>/dev/null
    else
        echo " $service_name is Currently up or Broken Package "
    fi

}

#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] = !R
#[Function Note] = Code Needs an Improvement on Config builder as well.
function stop_service(){

    service_name=$1
    service_cmd="service $service_name stop"
    service_status=$(check_service $service_name)
    if [ "$service_status" == "active" ];
        then
        sudo $service_cmd 2>/dev/null
    else
        echo " $service_name is not Running || Installed "
    fi

}

#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] = !R
#[Function Note] = Code Needs an Improvement on Config builder as well.
function check_service(){
    service_name=$1
    service_stat=‘service $service_name status | grep Active | cut -d ":" -f2 | tr -d ’ ’ | cut -d "(" -f1‘
    echo "$service_stat"
}

#[Function Name] =
#[Function Desc] =
#[Function Prio] =
#[Function Stat] =
#[Function Note] = Code Needs an Improvement on Config builder as well.
function intfChg(){
    intfName=$1
    intfOpt=$2
    sudo ifconfig $intfName $intfOpt
}

# {*} Function status = Finished
# {*} Function Desc = Display a list of users currently/recently logged on
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function user_info(){

    local cmd="$1"
    case "$cmd" in
        who) write_header " Who is online "; who -H; pause ;;
        last) write_header " List of last logged in users "; last ; pause ;;
    esac

}

# {*} Function status = Finished
# {*} Function Desc = Free Used and Memory Usage
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function mem_info(){

    echo "+-----------------------------------------------------------------------------+"
    echo "|                         [Free and used memory]                     |"
    echo "+-----------------------------------------------------------------------------+"

    free -m

    echo "+-----------------------------------------------------------------------------+"
    echo "|                        [Virtual memory statistics]                  |"
    echo "+-----------------------------------------------------------------------------+"

    vmstat

    echo "+-----------------------------------------------------------------------------+"
    echo "|                       [Top 5 memory eating process]                 |"
    echo "+-----------------------------------------------------------------------------+"

    ps auxf | sort -nr -k 4 | head -5

    echo "+-----------------------------------------------------------------------------+"

    pause

}

# {*} Function status = Alpha
# {*} Function Desc = Displays Common Ports In a New Terminal
# {*} Function To do = Echo bunch of stuff and put it in a tidy terminal with --zoom=0.75, Make a config file to select personal favorites.
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function common_portlist() {
    xfce4-terminal --geometry 35x30+547+0 --hide-menubar --zoom=0.80 -x ~/smart/toolset/port_list.sh 2>/dev/null &
    pause
}

# {*} Function status = Primitive
# {*} Function Desc = Last USB Syslog activities.
# {*} Function To do = Compare Files
# {*} Priority Stat = Least[X]Avg[ ]Medium[ ]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = None
function check_usb_log(){
    cat /var/log/syslog | grep "USB" | grep "usb" | tail -5
    pause
}

# {*} Function status = Finished
# {*} Function Desc = Checks MSDM for OEM key embedded in the Chipset , ACPI
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_oem_key(){
    sudo xxd /sys/firmware/acpi/tables/MSDM 2>/dev/null
    pause
}

# {*} Function Stat = Finished
# {*} Function Desc = Duh !
# {*} Function ToDo = Proxychainableility!
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function bring_terminal(){
    proxychains xfce4-terminal --geometry 100x25+0+480 --hide-menubar 2>/dev/null &
    xfce4-terminal --geometry 100x25+0+0 --hide-menubar -e "bash -c \"proxychains wget http://ipecho.net/plain -O - -q 2>/dev/null ; exec bash\"" 2>/dev/null &
    pause
}

# {*} Function Stat = Finished
# {*} Function Desc = Whois for a given Domain
# {*} Function ToDo = Save to File option, Multiple search options
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function zen_whois(){

    _zenity="/usr/bin/zenity"
    #_out="/tmp/whois.output.$$"
    _out=$(make_temp_file)
    domain=$(${_zenity} --title "Enter domain" --entry --text "Enter the domain you would like to see whois info" 2>/dev/null )

    if [ $? -eq 0 ]
    then
     # Display a progress dialog while searching whois database
    whois $domain 2>/dev/null | tee 2>/dev/null >(${_zenity} 2>/dev/null --width=200 --height=100 --title="whois" --progress --pulsate --text="Searching domain info..." --auto-kill --auto-close --percentage=10) >${_out} 2>/dev/null

    # Display back output
    ${_zenity} --width=800 --height=600 --title "Whois info for $domain" --text-info 2>/dev/null --filename="${_out}"
    else
    ${_zenity} --error --text="No input provided" 2>/dev/null
    fi
    clear_screen

}

# {*} Function Stat = Finished
# {*} Function Desc = Ping Scan of ip/range
# {*} Function ToDo = Specify Options
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function zen_ping(){

    _zenity="/usr/bin/zenity"
    #_out="/tmp/ping.output.$$"
    _out=$(make_temp_file)
    echo " temp file is $_out"
    ip=$(${_zenity} --title "Enter IP to Ping" --entry --text "Enter the ip address you would like to ping" 2>/dev/null )

    if [ $? -eq 0 ]
    then
    ping -c 4 $ip 2>/dev/null | tee 2>/dev/null >(${_zenity} 2>/dev/null --width=200 --height=100 --title="Probing" --progress --pulsate --text="Ping Probing..." --auto-kill --auto-close --percentage=10) >> ${_out} 2>/dev/null

    ${_zenity} --width=400 --height=240 --title "Probing info for $ip" --text-info 2>/dev/null --filename="${_out}"
    else
    ${_zenity} --error --text="No input provided" 2>/dev/null
    fi
    clear_screen

}

# {*} Function Stat = Finished
# {*} Function Desc = Ping sweep a Network for up hosts.
# {*} Function ToDo = Format Output and Save/Log
# {*} Priority Stat = Least[ ]Avg[ ]Medium[X]Ab.Avg[ ]Highest[ ]Critical[ ]Extreme[ ]
# {*} Note/Bugs/Usg = Converted to understand any CIDR format within range. Can be used by selecting nic to scan.
function ping_sweep(){

    printf "%s" "${GLS}Enter Your Network Address in CIDR or Pick an Interface ie.{wlan0} = "${RESET} ; read -r ip
    sip_param="-i $ip"

    end_1=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 5 | cut -d"." -f 1‘
    end_2=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 5 | cut -d"." -f 2‘
    end_3=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 5 | cut -d"." -f 3‘
    end_4=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 5 | cut -d"." -f 4‘

    start_1=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 3 | cut -d"." -f 1‘
    start_2=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 3 | cut -d"." -f 2‘
    start_3=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 3 | cut -d"." -f 3‘
    start_4=‘sipcalc $sip_param | grep "Usable range" | cut -d " " -f 3 | cut -d"." -f 4‘

    for octet_1 in $(seq $start_1 $end_1);do
        for octet_2 in $(seq $start_2 $end_2);do
            for octet_3 in $(seq $start_3 $end_3);do
                for octet_4 in $(seq $start_4 $end_4);do
                    #echo " Trying ${octet_1}.${octet_2}.${octet_3}.${octet_4} "
                    ping -c 1 ${octet_1}.${octet_2}.${octet_3}.${octet_4} | grep "ttl=" | cut -d ’ ’ -f4 | cut -d ":" -f1 2>/dev/null &
                    #echo " Done Running Command ${octet_1}.${octet_2}.${octet_3}.${octet_4}"
                done
            done
        done
    done
wait
   # sleep 5
    echo "Waiting for Ping Probes to Finish"
    pidof_ping=‘pidof ping‘
    if [ -z "$pidof_ping" ];
        then
        pause
    else
        echo "Still Running Ping Probes in the Background"
        echo "Sleeping $(sleep_indicator "5") 5 Secs"
        pause
    fi

}

#pingres=$(ping -c 4 $ipa)
#grep rtt | cut -d "/" -f 5
#echo "$pingres" | grep rtt | cut -d "/" -f 5

# [-------------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-------------------------------------------------------------------------------------------]
function vpn_quality_checker(){

    vpn_list=$(make_temp_file)
    ls /root/smart_vpn/ > $vpn_list
    echo "$vpn_list"
    fastest="N/A"
    fastest_ttl="9999"
    protocol="N/A"
    for ovpn in $(cat $vpn_list);do

        echo "+------------------------------+"
        echo "Fastest VPN = ${fastest} ${protocol} "
        echo "TTL Rate   = ${fastest_ttl}  "
        echo "+------------------------------+"
        ipa=$(cat /root/smart_vpn/$ovpn | grep "remote" | head -1 | cut -d " " -f2 )
        proto=$(cat /root/smart_vpn/$ovpn | grep "proto" | head -1 | cut -d " " -f2 )
        pingres=$(ping -c 4 $ipa)
        avg_ttl=$(echo "$pingres" | grep rtt | cut -d "/" -f 5 | cut -d "." -f1 )
        #echo "$ovpn"
        base_vpn_name=$(echo "$ovpn" | cut -d"." -f1 )


        if [ $avg_ttl -ge $fastest_ttl ];

            then
            echo "${RLS}$base_vpn_name AVG TTL is higher Skipping..."
            #sleep .2
        else
            echo "${GLS}$base_vpn_name has Avg TTL time of $avg_ttl"
            echo "${GLS}Assigning New Fastest --> $base_vpn_name with $avg_ttl ms."
            fastest_ttl="$avg_ttl"
            fastest="$base_vpn_name"
            protocol="$proto"


        fi
        clear_screen
        main_menu
    done
    pause

}



# {*} Function Stat = Finished
# {*} Function Desc = Spawn Netstat -antp with watch command in different terminal
# {*} Function ToDo = Coloring Maybe , Kill remanining terminal after HALT
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function display_netstat(){
    xfce4-terminal --geometry 100x25+0+0 --hide-menubar --zoom=0.80 -e "bash -c \"watch netstat -antp; exec bash\"" 2>/dev/null &
    pause
}

# {*} Function Stat = Finished
# {*} Function Desc = Spawn TCPTRACK in different Window
# {*} Function ToDo = Coloring Maybe , Kill remanining terminal after HALT
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function display_tcptrack(){

    echo " List of Available Interfaces "
    devices=$(netstat -i | cut -d" " -f1 | egrep -v "^Kernel|Iface|lo")
    read -p "Which Interface to change Mac ? ( eth0 | wlan0 | tap0 ) =" chcInf
    if [ -z "$chcInf" ];
        then
        echo "${RLS} No Interface Selected !"
    else
        intf=$chcInf
        xfce4-terminal --geometry 80x25+0+0 --hide-menubar --zoom=0.80 -e "bash -c \"tcptrack -i $intf; exec bash\"" 2>/dev/null &
    fi
    pause

}

# {*} Function Stat = Finished
# {*} Function Desc = Run Proxychained Firefox with running tor backbone
# {*} Function ToDo = Check if Firefox Running, List Webbrowser and Apps
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = Still needs sanitization for outputs, Need to kill the terminal afterwards
function proxy_browse(){

    echo "+---------------------------------------------------------------------------+"
    echo "#            [1) Midori | 2) Firefox | 3) Lynx | 4)Terminate]        |"
    echo "+---------------------------------------------------------------------------+"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read pb
    case "$pb" in
        1) browser="midori";;
        2) browser="firefox --private";;
        3) browser="lynx";;
        4) browser="terminate" ;;
        * ) echo "No Input Provided";;
    esac

    if [ "$pb" == "4" ]; then
        echo "${RLS} Terminated !"
        pause
    else
        _zenity="/usr/bin/zenity"
        url_to_visit=$(${_zenity} --title "Enter domain" --entry --text "Enter an URL to VISIT with proxychains" 2>/dev/null )
        xfce4-terminal --geometry 64x16+310+720 --hide-menubar -e "bash -c \"proxychains $browser $url_to_visit ; exec bash\"" 2>/dev/null
        pause
    fi

}

# {*} Function status = Finished
# {*} Function Desc = Sends Interrupt to TOR service for a new identity
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function spawn_proxy_app(){
    _zenity="/usr/bin/zenity"
    app_param=$(${_zenity} --title "Enter Custom Command" --entry --text "Enter a command to use with proxychains" 2>/dev/null )
    xfce4-terminal --geometry 100x16+0+0 --hide-menubar -e "bash -c \"proxychains $app_param 2>/dev/null ; exec bash\"" 2>/dev/null
    pause
}

# {*} Function status = Finished
# {*} Function Desc = Pick Single File with Zenity pass the path
# {*} Function To do =
# {*} Priority Stat = @
# {*} Note/Bugs/Usg =
function pick_single_file() {

    OLDIFS="$IFS"
    IFS=’-’
    single_file=$(zenity --file-selection --multiple --separator=’-’ --title "Pick a file" 2>/dev/null)
    IFS="$OLDIFS"
    echo $single_file

}

# {*} Function status = Finished
# {*} Function Desc = Pick Multiple Files with Zenity pass the paths
# {*} Function To do =
# {*} Priority Stat = @
# {*} Note/Bugs/Usg =
function pick_multiple_file() {

    FILES=’-’
    OLDIFS="$IFS"
    IFS=’-’
    FILES=($(zenity --file-selection --multiple --separator=’-’ --title "Pick a file" 2>/dev/null))
    IFS="$OLDIFS"
    for multi_file in "${FILES[@]}"
    do
       echo $multi_file
    done

}

# {*} Function status = Finished
# {*} Function Desc = Extract Files
# {*} Function To do = Nothing to implement
# {*} Priority Stat = @
# {*} Note/Bugs/Usg =
function extract_file() {

    OLDIFS="$IFS"
    IFS=’-’
    single_file=$(zenity --file-selection --multiple --separator=’-’ --title "Pick a file" 2>/dev/null)
    IFS="$OLDIFS"
    echo $single_file
    if [[ -f "$single_file" ]]; then
      case "$single_file" in
        *.tar.bz2) tar xjf "$single_file" ;;
        *.tar.gz) tar xzf "$single_file" ;;
        *.bz2)    bunzip2 "$single_file" ;;
        *.rar)    rar x  "$single_file" ;;
        *.7z)     7z x   "$single_file" ;;
        *.gz)     gunzip "$single_file" ;;
        *.tar)    tar xf "$single_file" ;;
        *.tbz2)   tar xjf "$single_file" ;;
        *.tgz)    tar xzf "$single_file" ;;
        *.zip)    unzip  "$single_file" ;;
        *)        echo "$single_file cannot be extracted" ;;
      esac
    else
      echo "$single_file is not a valid file"
    fi
    pause

}


# {*} Function status = Beta
# {*} Function Desc = Do FoolProof (kinda..) Disk Imaging tool to gather all iso
# {*} Function To do = Need a control flow mechanism. 5 sec termination sequence does not listen.
# {*} Priority Stat =
# {*} Note/Bugs/Usg = There can also be another type of file picker option from smart_image
function create_disk(){

    echo "${DARKGRAY}"
    echo "+----------------------------------------------------------------------------+"
    echo "| ${LIGHTRED}Warning${DEFAULT}${DARKGRAY} !        Please Think Twice of Your Actions ! |"
    echo "+----------------------------------------------------------------------------+"
    echo "# [1) List Disks | 2) Format a Disk | 3) Privacy Cleanup | 4)Create Disk |"
    echo "+----------------------------------------------------------------------------+"
    echo "${DEFAULT}"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read t
```

```
        case $t in
        1)  lsblk ;;
        2)  format_disk ;;
        3)  privacy_cleanup ;;
        4)  create_disk_image ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
        echo "Please select a valid option !"
    esac
    pause

}

# [----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [----------------------------------------------------------------------------------------]
function format_disk(){

    echo "${lightyellow}"
    lsblk | grep disk
    echo "${DEFAULT}"
    main_drive=$(lsblk | grep disk | cut -d " " -f1)
    echo "${RLS}You Should Not be Picking your Resident Drive ${RED}$main_drive${DEFAULT}"
    echo "${GLS}Please enter your device [NOT Partition if Image] {ie./dev/sdc} = "
    read padisk
    echo "${RLS}Are you sure ? Please enter again to confirm = "
    read pbdisk
    if [[ "$padisk" = "$pbdisk" ]];
        then
        echo "${GLS} Formatting is Commencing in 5, You can Still Unplug it !"
        sleep_indicator 5
        dd if=/dev/zero of=$padisk bs=1M status=progress && sync
    fi

}

# [----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [----------------------------------------------------------------------------------------]
function create_disk_image(){

    echo "${lightyellow}"
    lsblk | grep disk
    echo "${DEFAULT}"
    main_drive=$(lsblk | grep disk | cut -d " " -f1)
    echo "${RLS}You Should Not be Picking your Resident Drive ${RED}$main_drive${DEFAULT}"
    echo "${GLS}Please enter your device [NOT Partition if Image] {ie./dev/sdc} = "
    read iadisk
    echo "${RLS}Are you sure ? Please enter again to confirm = "
    read ibdisk
    echo "${GLS}Please pick a image file"
    sleep 1
    image_file=$(pick_single_file)
    if [ ! -z "$image_file" ];
            then
            if [[ "$iadisk" = "$ibdisk" ]];
                 then
                    read -r -p "Are you sure? [y/N] " response
                    case "$response" in
                        [yY][eE][sS]|[yY])
                            #echo "${GLS} "
                            sleep_indicator 5 "Creating the image Commencing in 5, You can Still Unplug it !"
                            echo -ne ""
                            dd if=$image_file of=$iadisk bs=1M status=progress && sync
                            ;;
                        *)
                            pause
                            ;;
                    esac

             fi
    fi

}

# [----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [----------------------------------------------------------------------------------------]
function are_you_sure(){

    read -r -p "Are you sure? [y/N] " response
    case "$response" in
        [yY][eE][sS]|[yY])
            do_something
            ;;
        *)
            do_something_else
            ;;
    esac

}

# {*} Function status = Finished
# {*} Function Desc = GPA and GPG options to use
# {*} Function To do = BACKUP PGP CONFIGS
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = timestamp, can be used as is.
function pgp_ops() {

    echo "+----------------------------------------------------------------------------+"
    echo "#     [1) List Keys | 2) Encrypt File | 3) Decrypt File | 4)Import Key] |"
    echo "+----------------------------------------------------------------------------+"
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read t
        case $t in
        1)  list-public-keys ;;
        2)  encrypt_gpg ;;
        3)  decrypt_gpg ;;
        4)  import_pubkey ;;
        FF)  clear_screen && return 0 ;;
        ff)  clear_screen && return 0 ;;
        *)
            echo "Please select a valid option !"
    esac
    pause

}

# {*} Function Stat = Finished
# {*} Function Desc = Import Public Keys Directly To Chain
# {*} Function ToDo = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function import_pubkey(){

    path_to_file="$(pick_single_file)"
    if [ -z "$path_to_file" ]; then
        echo "${RLS} Nothing Selected !"
        pause
    else
        echo "Path to file is $path_to_file"
        gpg --import $path_to_file
    fi

}

# {*} Function Stat = Finished
# {*} Function Desc = Encrypt with public
# {*} Function ToDo = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function encrypt_gpg(){

    #query_users=‘gpg --list-public-keys | grep "@" | cut -d"]" -f2 | cut -d" " -f2-3‘
    query_users=‘gpg --list-public-keys | grep "@" | cut -d"]" -f2 | cut -d" " -f2-3 | cut -d"<" -f2 | cut -d
            ">" -f1‘
    #echo "$query_users"
    publicArray=($query_users)
    #echo "$publicArray"
    total_users=${#publicArray[@]}
    echo "${GLS} $total_users keys found ! "
    diff_val=1
    for scp in $(seq 0 $(( total_users - diff_val )));do
        index=‘printf "[%02d]" $scp‘
        printf "${RES} $index ${publicArray[$scp]} \n"
    done
    #echo "${RLS} Would you like to perform a scan ?"
    read -p "${GLS} Pick user as recipient or q to quit " choice

    if [ "$choice" == "q" ]; then
        echo "${RLS} Terminated !"
        pause
    else
        recipient="${publicArray[$choice]}"
        #echo "${publicArray[$choice]}" | xclip -i # TR d is to cut return key problem with new line input
        path_to_file="$(pick_single_file)"
        if [ ! -z "$path_to_file" ];then
            echo "Path to file is $path_to_file"
            gpg --encrypt --recipient $recipient $path_to_file
        else
            echo "${RLS} Nothing Selected !"
            pause
        fi
    fi

}

# {*} Function Stat = Finished
# {*} Function Desc = !Encrypt
# {*} Function ToDo = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function decrypt_gpg(){

    path_to_file="$(pick_single_file)"
    if [ -z "$path_to_file" ];then
        echo "${RLS} Nothing Selected !"
        pause
    else
        echo "Path to file is $path_to_file"
        gpg --decrypt $path_to_file
    fi

}

# {*} Function Stat = Finished
# {*} Function Desc = List and Order Public keys
# {*} Function ToDo = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function list-public-keys(){

    #query_users=‘gpg --list-public-keys | grep "@" | cut -d"]" -f2 | cut -d" " -f2-3‘
    query_users=‘gpg --list-public-keys | grep "@" | cut -d"]" -f2 | cut -d" " -f2-3 | cut -d"<" -f2 | cut -d ">" -f1‘
    #echo "$query_users"
    publicArray=($query_users)
    #echo "$publicArray"
    total_users=${#publicArray[@]}
    echo "${GLS} $total_users keys found ! "
    diff_val=1
    for scp in $(seq 0 $(( total_users - diff_val )));do
        index=‘printf "[%02d]" $scp‘
        printf "${RES} $index ${publicArray[$scp]} \n"
    done

}

# {*} Function status = Finished
# {*} Function Desc = $1-> Message (optional)
# {*} Function To do = None
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function check_pastebin(){

#https://pastebin.com/NsnAJ8Ev

}

# [------------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [------------------------------------------------------------------------------------------]
function ffiat_curr(){

    echo "${GLS}Check Time @ $(timestamp)"

    temp_price=$(make_temp_file)
    curl_raw=‘curl https://api.coinmarketcap.com/v1/ticker/monero/ 2>/dev/null‘
    echo "$curl_raw" >> $temp_price
    symbol=‘cat $temp_price | grep "symbol" | cut -d ":" -f2 | cut -d "\"" -f2 ‘
    price_usd=‘cat $temp_price | grep "price_usd" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_1h=‘cat $temp_price | grep "percent_change_1h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_24h=‘cat $temp_price | grep "percent_change_24h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_7d=‘cat $temp_price | grep "percent_change_7d" | cut -d ":" -f2 | cut -d "\"" -f2‘
    echo -e "${BES}Currencysym = ${BLUE}$symbol${RESET}"
    echo -e "${BES}Current USD = $price_usd"
    echo -e "${BES}1 Hr Change = $percent_change_1h %"
    echo -e "${BES}1 Da Change = $percent_change_24h %"
    echo -e "${BES}1 We Change = $percent_change_7d %"
    echo -e ""
    #echo -e "$(timestamp)"

    temp_price=$(make_temp_file)
    curl_raw=‘curl https://api.coinmarketcap.com/v1/ticker/ethereum/ 2>/dev/null‘
    echo "$curl_raw" >> $temp_price
    symbol=‘cat $temp_price | grep "symbol" | cut -d ":" -f2 | cut -d "\"" -f2‘
    price_usd=‘cat $temp_price | grep "price_usd" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_1h=‘cat $temp_price | grep "percent_change_1h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_24h=‘cat $temp_price | grep "percent_change_24h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_7d=‘cat $temp_price | grep "percent_change_7d" | cut -d ":" -f2 | cut -d "\"" -f2‘
    echo -e "${RES}Currencysym = ${RED}$symbol${RESET}"
    echo -e "${RES}Current USD = $price_usd"
    echo -e "${RES}1 Hr Change = $percent_change_1h %"
    echo -e "${RES}1 Da Change = $percent_change_24h %"
    echo -e "${RES}1 We Change = $percent_change_7d %"
    echo -e ""
    #echo -e "$(timestamp)"

    temp_price=$(make_temp_file)
    curl_raw=‘curl https://api.coinmarketcap.com/v1/ticker/bitcoin/ 2>/dev/null‘
    echo "$curl_raw" >> $temp_price
    symbol=‘cat $temp_price | grep "symbol" | cut -d ":" -f2 | cut -d "\"" -f2‘
    price_usd=‘cat $temp_price | grep "price_usd" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_1h=‘cat $temp_price | grep "percent_change_1h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_24h=‘cat $temp_price | grep "percent_change_24h" | cut -d ":" -f2 | cut -d "\"" -f2‘
    percent_change_7d=‘cat $temp_price | grep "percent_change_7d" | cut -d ":" -f2 | cut -d "\"" -f2‘
    echo -e "${GES}Currencysym = ${GREEN}$symbol${RESET}"
    echo -e "${GES}Current USD = $price_usd"
    echo -e "${GES}1 Hr Change = $percent_change_1h %"
    echo -e "${GES}1 Da Change = $percent_change_24h %"
    echo -e "${GES}1 We Change = $percent_change_7d %"
    echo -e ""
    #echo -e "$(timestamp)"

    temp_price=$(make_temp_file)
    curl_raw=`curl https://api.coinmarketcap.com/v1/ticker/siacoin/ 2>/dev/null`
    echo "$curl_raw" >> $temp_price
    symbol=`cat $temp_price | grep "symbol" | cut -d ":" -f2 | cut -d "\"" -f2`
    price_usd=`cat $temp_price | grep "price_usd" | cut -d ":" -f2 | cut -d "\"" -f2`
    percent_change_1h=`cat $temp_price | grep "percent_change_1h" | cut -d ":" -f2 | cut -d "\"" -f2`
    percent_change_24h=`cat $temp_price | grep "percent_change_24h" | cut -d ":" -f2 | cut -d "\"" -f2`
    percent_change_7d=`cat $temp_price | grep "percent_change_7d" | cut -d ":" -f2 | cut -d "\"" -f2`
    echo -e "${GES}Currencysym = ${GREEN}$symbol${RESET}"
    echo -e "${GES}Current USD = $price_usd"
    echo -e "${GES}1 Hr Change = $percent_change_1h %"
    echo -e "${GES}1 Da Change = $percent_change_24h %"
    echo -e "${GES}1 We Change = $percent_change_7d %"
    echo -e ""
    pause

}

# [-----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-----------------------------------------------------------------------------------------]
function deploy_honeypot(){
  xfce4-terminal --geometry 56x24+530+0 --hide-menubar --zoom=0.80 -x ~/smart/toolset/pbox/smart_hp.rb 2>/dev/null &
  pause
}

# [-----------------------------------------------------------------------------------------]
# {*} Function status = Skeleton[ ]-Alpha[ ]-Beta[ ]-Functional[ ]-Finished[ ]-Perfections[ ]
# {*} Function Desc =
# {*} Function To do =
# {*} Priority Stat = Least[ ]-Avg[X]-Medium[ ]-Ab.Avg[ ]-Highest[ ]-Critical[ ]-Extreme[ ]
# {*} Note/Bugs/Usg =
# [-----------------------------------------------------------------------------------------]
function non_exists(){

echo -e "       \                        /"
echo -e "        \                      /"
echo -e "         \   This page does   /"
echo -e "         ]   not exist yet!  [    ,'|"
echo -e "         ]                   [   /  |"
echo -e "         ]___             ___[  ,'  |"
echo -e "         ]  ]\            /[ [ |:   |"
echo -e "         ]  ] \          / [ [ |:   |"
echo -e "         ]  ]  ]        [  [ [ |:   |"
echo -e "         ]  ]  ]__    __[  [ [ |:   |"
echo -e "         ]  ]  ] ]\ _ /[ [ [ [ |:   |"
echo -e "         ]  ]  ] ] (#) [ [ [ [ :===='"
echo -e "         ]  ]  ]_].nHn.[_[ [ ["
echo -e "         ]  ]  ]  HHHHH. [ [ ["
echo -e "         ]  ] /   'HH( N \ [ ["
echo -e "         ]__]/     HHH  * \[__["
echo -e "         ]         NNN       ["
echo -e "         ]         N/7       ["
echo -e "         ]         N H       ["
echo -e "        /          N          |"
echo -e "       /           q,          |"
echo -e "      /                         |"
pause

}

# {*} Function Stat = Finished
# {*} Function Desc = Main Menu Items
# {*} Function ToDo = Fill remaining place holders,
# {*} Priority Stat = @
# {*} Note/Bugs/Usg = None
function main_menu(){
    #echo -e "${DARKGRAY}${bold}"
    #figlet SMART\'s ARSENAL
    echo -e "+----------------------------------------------------------------------------+"
    figlet -ctf small "S.M.A.R.T"
    echo -e "+----------------------------------------------------------------------------+"
    echo -e "+---------------Security Metric Assesment And Reporting Tool-----------------+"
    #echo -e "${RESET}${normal}"
    echo -e "${DARKGRAY}"
    echo -e "${DARKGRAY}+-------------------------------------------------------------------------+${normal}"
    echo -e "${DARKGRAY}|                         ${BLUE}${bold}[MAIN Chest]${RESET}${DARKGRAY}                      |${normal}"
    echo -e "${DARKGRAY}+-------------------------------------+-----------------------------------+${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x01]${RESET}${DARKGRAY} - ${RED}The Onion || Identity${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x09]${RESET}${DARKGRAY} - ${purple}Physical Security ${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x02]${RESET}${DARKGRAY} - ${RED}Global PrxyChainedIP${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1A]${RESET}${DARKGRAY} - ${purple}PGP Operations
```

```
            Center${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x03]${RESET}${DARKGRAY} - ${RED}Change IntMAC Address${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1B]${RESET}${DARKGRAY} - ${purple}Create Disk Templates${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x04]${RESET}${DARKGRAY} - ${GREEN}ProxyChained Commands${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1C]${RESET}${DARKGRAY} - ${lightyellow}VPN Quality Tester ${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x05]${RESET}${DARKGRAY} - ${GREEN}ProxyChained WBrowser${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1D]${RESET}${DARKGRAY} - ${lightyellow}OneTimePad Generator ${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x06]${RESET}${DARKGRAY} - ${GREEN}ProxyChained Terminal${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1E]${RESET}${DARKGRAY} - ${lightyellow}Blockchain Currencies${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x07]${RESET}${DARKGRAY} - ${BLUE}Information Sub-Menu${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0x1F]${RESET}${DARKGRAY} - ${lightyellow}Deploy Quick Honeypot${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}|    ${bold}${BLACK}${RED}[0x08]${RESET}${DARKGRAY} - ${BLUE}Auxillary Sub-Menu${normal}${DARKGRAY} | ${bold}${BLACK}${RED}[0xFF]${RESET}${DARKGRAY} - ${bold}Terminate -[HALT]- ${RESET}${DARKGRAY} |${normal}"
    echo -e "${DARKGRAY}+--------------------------------------+------------------------------------+${normal}"
    echo ""
}
### | Main Flow          | =====================================================#

#Require Root Priv.
if [[ ! $(id -u) == 0 ]]; then
    echo -e "${RED}[!]${RESET} This script must be run as root"
    exit 1
fi
trap ’’ SIGINT SIGQUIT SIGTSTP      #Trap CTRL Z / X / C Interrupts
set_ttl "| < = ] --------{ SMART }--------- [ = > |"
selection=
until [ "$selection" = "0" ]; do
    main_menu
    echo -en "${bold}${RED}SMART${RESET}${normal}${BLUE}->${RESET} "
    read selection
    case $selection in
        1 ) start_tor ;;
        2 ) check_GIP ;;
        3 ) change_mac;;
        4 ) spawn_proxy_app ;;
        5 ) proxy_browse;;
        6 ) bring_terminal;;
        7 ) info_menu;;
        8 ) remote_menu;;
        9 ) afk_paranoia ;;
        1A ) pgp_ops;;
        1a ) pgp_ops;;
        1B ) create_disk;;
        1b ) create_disk;;
        1C ) vpn_quality_checker;;
        1c ) vpn_quality_checker;;
        1D ) one_time_pad;;
        1d ) one_time_pad;;
        1E ) ffiat_curr;;
        1e ) ffiat_curr;;
        1F ) deploy_honeypot;;
        1f ) deploy_honeypot;;
        fi ) non_exists;;
        FI ) non_exists;;
        FF ) exit ;;
        ff ) exit ;;
        backup ) echo "Backup Script" ;;
        * ) echo "Nothing Selected !" ; pause ;;
    esac
done
#
#================================================================================#
#>EOF<
#================================================================================#
```

Code 5.2: codes/smart.sh

```
#!/bin/bash -
#title          :iperf.sh
#description    :IPERF CHECKER
#author         :Mert Kilic
#date           :29-08-17
#version        :v0.98d beta(Non-Release)(PoC)(UNDER DEV)
#usage          :./iperf.sh
#notes          :
#bash_version   :4.4.0(1)-release
#================================================================================#
if ! [ -x "$(type -P iperf3)" ]; then
  echo "ERROR: script requires iperf"
  echo "For Debian and friends get it with ’apt-get install iperf’"
  echo "If you have it, perhaps you don’t have permissions to run it, try ’sudo $(basename $0)’"
  exit 1
fi

if [ "$#" -ne "2" ]; then
  echo "ERROR: script needs four arguments, where:"
  echo
  echo "1. Number of times to repeat test (e.g. 10)"
  echo "2. Host running ’iperf3 -s’ (e.g. somehost)"
  echo
  echo "Example:"
  echo " $(basename $0) 10 somehost"
  echo
  echo "The above will run 'iperf3 -c' 10 times on the client and report totals and average."
  exit 1
else
  runs=$1
  host=$2
fi

log=iperf.$host.log

if [ -f $log ]; then
  echo removing $log
  rm $log
fi

echo "=================================================================="
echo " Results"
echo "=================================================================="
echo " target host .... $host"
echo "------------------------------------------------------------------"

for run in $(seq 1 $runs); do
  iperf3 -c - R $host -f m >> $log
  echo -e " run $run: \t $(awk '/Bandwidth/ {getline}; END{print $7, $8}' $log)"
done

avg=$(awk -v runs=$runs '/Bandwidth/ {getline; sum+=$7; avg=sum/runs} END {print avg}' $log)


echo "------------------------------------------------------------------"
echo " average ....... $avg Mbits/sec"
echo
echo "see $log for details"
```

Code 5.3: codes/iperf.sh

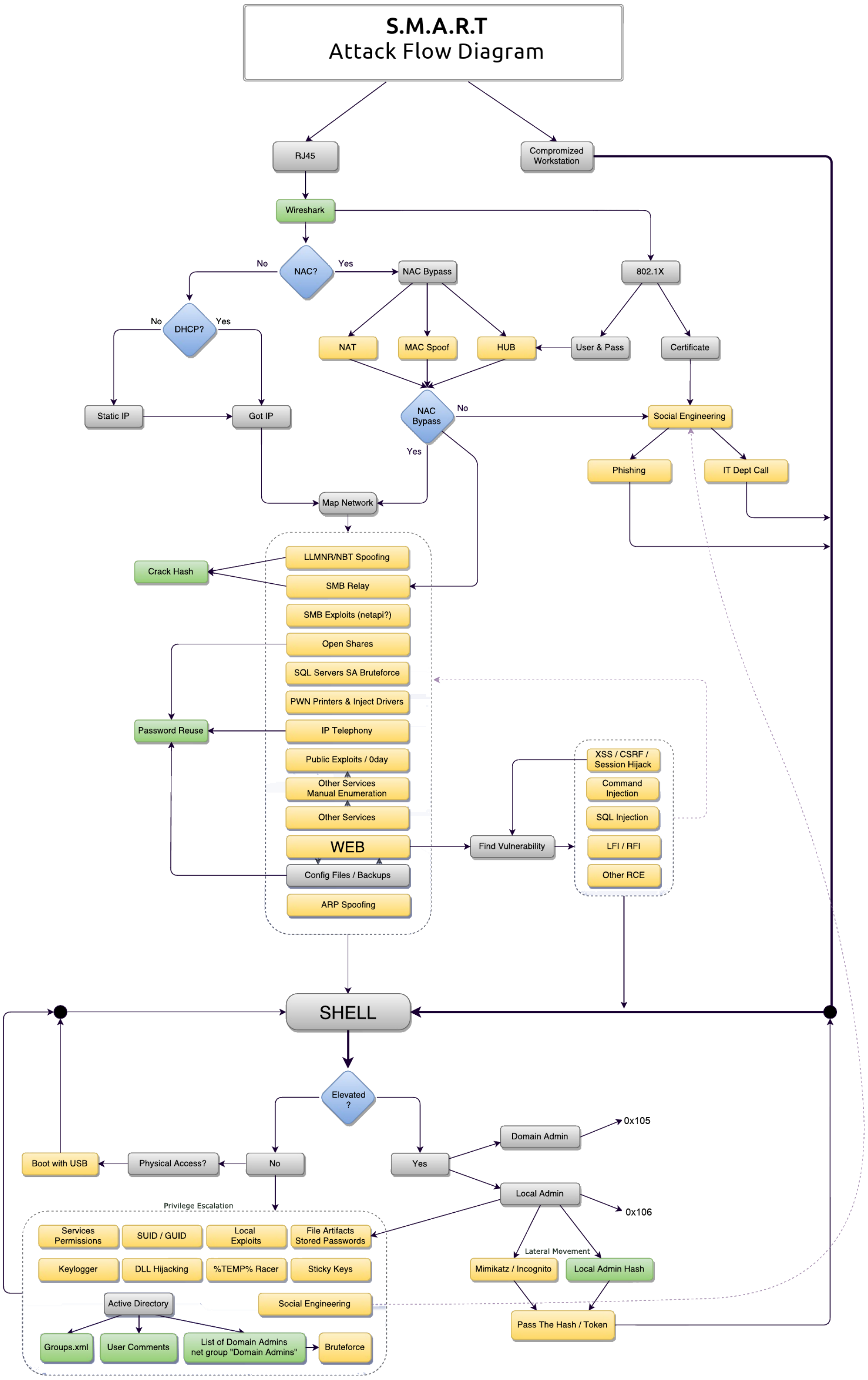
```
#!/bin/bash -
#title         :smart_fetch.sh
#description   :SMART Web Search
#author        :Mert Kilic
#date          :10-05-17
#version       :v0.98d beta(Non-Release)(PoC)(UNDER DEV)
#usage         :./smart_fetch.sh
#notes         :
#bash_version  :4.4.0(1)-release
#================================================================================#
clear
echo ""
echo ".========================================================."
echo "| S.M.A.R.T                                 |"
echo "| COMMAND LINE WWW SEARCH                          |"
echo "| -------------------------------------------------- |"
echo "|                                                  |"
echo "| Version: 1.0                                     |"
echo "| Security Metric Assesment And Reporting Tool     |"
echo "|                                                  |"
echo "|                                                  |"
echo "| Usage: ./smart_fetch.sh <search strings>         |"
echo "| Example: ./smart_fetch.sh New Java Vulnerabilities |"
echo "|                                                  |"
echo ".========================================================."
echo ""

if [ -z $1 ]
then
 echo "ERROR: No search string supplied."
 echo "USAGE: ./smart_fetch.sh <search string>"
 echo ""
 echo -n "Search: "
 read SEARCH
else
 SEARCH=$@
fi

URL="http://google.com/search?hl=en&safe=off&q="
STRING=`echo $SEARCH | sed 's/ /%20/g'`
URI="$URL%22$STRING%22"

lynx -dump $URI > gone.tmp
sed 's/http/\^http/g' gone.tmp | tr -s "^" "\n" | grep http| sed 's/\ .*//g' > gtwo.tmp
rm gone.tmp
sed '/google.com/d' gtwo.tmp > urls
rm gtwo.tmp

echo "SUCCESS: Extracted `wc -l urls` and listed them in '`pwd`/urls' file for reference."
echo ""
cat urls
echo ""
```

Code 5.4: codes/smart_fetch.sh

Reference Content

FGT100D - Fortigate 100D Firewall UTM - http://a.co/dkuy91t

TZ600 - SonicWall TZ600 SOHO Firewall UTM http://a.co/fTAsv4S



Dependency List

```
tcptrack#tcptrack
figlet#figlet
alsa-utils#arecord
locate#locate
zenity#zenity
motion#motion
streamer#streamer
libnotify-bin#notify-send
tor#tor
sipcalc#sipcalc
proxychains#proxychains
xxd#xxd
nmap#nmap
rar#rar
bunzip2#bunzip2
gunzip#gunzip
tar#tar
unzip#unzip
p7zip-full#7z
meld#meld
remmina#remmina
gpg#gpg
mktemp#mktemp
iptables#iptables
exiftool#exiftool
curl#curl
lynx#lynx
midori#midori
macchanger#macchanger
iperf3#iperf3
kleopatra#kleopatra
```

Code 5.5: codes/lilith.list

Global Constants and Declerations

Smart Configuration File

```
#!/bin/bash
#####DO NOT DELETE THE DEFAULT FILE !
dont_ask="true"
outputs_dir="/root/smart/"
default_path="/root"
vpn_dir="/root/smart/"
requirements_met="true"
startup_check="false"
export_to_path="true"
```

Code 5.6: codes/smart.conf

Default Configuration File

```
#!/bin/bash
dont_ask="false"
outputs_dir="/root"
default_path="/root"
requirements_met="false"
startup_check="false"
export_to_path="false"
```

Code 5.7: codes/smart_def.conf

Coloring definitions for the CLI Terminal

```
#!/bin/bash
declare -r RES=${RED}"[!]"${RESET}
declare -r RLS=${RED}"[*]"${RESET}
declare -r RQS=${RED}"[?]"${RESET}
declare -r BES=${BLUE}"[!]"${RESET}
declare -r BLS=${BLUE}"[*]"${RESET}
declare -r BQS=${BLUE}"[?]"${RESET}
declare -r GLS=${GREEN}"[*]"${RESET}
declare -r GES=${GREEN}"[!]"${RESET}
declare -r GQS=${GREEN}"[?]"${RESET}

declare -r dim=‘echo -en "\e[2m"‘
declare -r bold=‘echo -en "\e[1m"‘
declare -r blink=‘echo -en "\e[5m"‘
declare -r normal=‘echo -en "\e[0m"‘
declare -r hidden=‘echo -en "\e[8m"‘
declare -r reverse=‘echo -en "\e[7m"‘
declare -r underline=‘echo -en "\e[4m"‘
declare -r strickthru=‘echo -en "\e[9m"‘

AQUA=‘echo -en "\e[46m"‘
```

```
aqua=`echo -en "\e[36m"`
GRAY=`echo -en "\e[47m"`
gray=`echo -en "\e[37m"`
BLACK=`echo -en "\e[40m"`
black=`echo -en "\e[30m"`
WHITE=`echo -en "\e[107m"`
white=`echo -en "\e[97m"`
ORANGE=`echo -en "\e[43m"`
orange=`echo -en "\e[33m"`
PURPLE=`echo -en "\e[45m"`
purple=`echo -en "\e[35m"`
DEFAULT=`echo -en "\e[49m"`
default=`echo -en "\e[39m"`
DARKGRAY=`echo -en "\e[100m"`
darkgray=`echo -en "\e[90m"`
LIGHTRED=`echo -en "\e[101m"`
lightred=`echo -en "\e[91m"`
LIGHTBLUE=`echo -en "\e[104m"`
lightblue=`echo -en "\e[94m"`
LIGHTAQUA=`echo -en "\e[106m"`
lightaqua=`echo -en "\e[96m"`
LIGHTGREEN=`echo -en "\e[102m"`
lightgreen=`echo -en "\e[92m"`
LIGHTYELLOW=`echo -en "\e[103m"`
lightyellow=`echo -en "\e[93m"`
LIGHTPURPLE=`echo -en "\e[105m"`
lightpurple=`echo -en "\e[95m"`
```

Code 5.8: codes/coloring_scheme.conf

RPI Health Check

```
#!/bin/bash
# SMART RPI node, Temperature check
cpuTempC=$(($(cat /sys/class/thermal/thermal_zone0/temp)/1000))
cpuTempF=$(($cpuTempC*9/5+32))

gpuTempC=$(/opt/vc/bin/vcgencmd measure_temp)
gpuTempC=${gpuTempC:5:2}
gpuTempF=$(($gpuTempC*9/5+32))

echo "CPU Temp: $cpuTempC C or $cpuTempF F"
echo "GPU Temp: $gpuTempC C or $gpuTempF F"
```

Code 5.9: codes/temp.sh

OUTPUTS

Dmidecode

```
# dmidecode 3.0
Getting SMBIOS data from sysfs.
SMBIOS 2.8 present.
42 structures occupying 2980 bytes.
Table at 0x87EB0000.

Handle 0x0000, DMI type 0, 24 bytes
BIOS Information
  Vendor: American Megatrends Inc.
  Version: F.2F
  Release Date: 12/15/2015
  Address: 0xF0000
  Runtime Size: 64 kB
  ROM Size: 6144 kB
  Characteristics:
    PCI is supported
    BIOS is upgradeable
    BIOS shadowing is allowed
    Boot from CD is supported
    Selectable boot is supported
    EDD is supported
    5.25"/1.2 MB floppy services are supported (int 13h)
    3.5"/720 kB floppy services are supported (int 13h)
    3.5"/2.88 MB floppy services are supported (int 13h)
    Print screen service is supported (int 5h)
    8042 keyboard services are supported (int 9h)
    Serial services are supported (int 14h)
    Printer services are supported (int 17h)
    ACPI is supported
    USB legacy is supported
    Smart battery is supported
    BIOS boot specification is supported
    Function key-initiated network boot is supported
    Targeted content distribution is supported
    UEFI is supported
  BIOS Revision: 15.47
  Firmware Revision: 33.35

Handle 0x0001, DMI type 1, 27 bytes
System Information
  Manufacturer: HP
  Product Name: HP Spectre x360 Convertible
  Version:
  Serial Number: 5CD543BL6S
  UUID: 35444335-3334-4C42-3653-534C33344435
  Wake-up Type: Power Switch
  SKU Number: P5P85EA#AB8
  Family: 103C_5335KV G=N L=CON B=HP S=SPT

Handle 0x0002, DMI type 2, 15 bytes
Base Board Information
  Manufacturer: HP
  Product Name: 804E
  Version: 33.23
  Serial Number: PFLJH028J9M02I
  Asset Tag: Base Board Asset Tag
  Features:
    Board is a hosting board
    Board is replaceable
  Location In Chassis: Base Board Chassis Location
  Chassis Handle: 0x0003
  Type: Motherboard
  Contained Object Handles: 0

Handle 0x0003, DMI type 3, 25 bytes
Chassis Information
  Manufacturer: HP
  Type: Notebook
  Lock: Not Present
  Version: Chassis Version
  Serial Number: Chassis Serial Number
  Asset Tag: Not Specified
  Boot-up State: Safe
  Power Supply State: Safe
  Thermal State: Safe
  Security Status: None
  OEM Information: 0x00000000
  Height: Unspecified
  Number Of Power Cords: 1
  Contained Elements: 1
    <OUT OF SPEC> (0)
  SKU Number: Not Specified

Handle 0x0004, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J1A1
  Internal Connector Type: None
  External Reference Designator: PS2Mouse
  External Connector Type: PS/2
  Port Type: Mouse Port

Handle 0x0005, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J1A1
  Internal Connector Type: None
  External Reference Designator: Keyboard
  External Connector Type: PS/2
  Port Type: Keyboard Port

Handle 0x0006, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J2A1
  Internal Connector Type: None
  External Reference Designator: TV Out
  External Connector Type: Mini Centronics Type-14
  Port Type: Other

Handle 0x0007, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J2A2A
  Internal Connector Type: None
  External Reference Designator: COM A
  External Connector Type: DB-9 male
  Port Type: Serial Port 16550A Compatible

Handle 0x0008, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J2A2B
  Internal Connector Type: None
  External Reference Designator: Video
  External Connector Type: DB-15 female
  Port Type: Video Port

Handle 0x0009, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J3A1
  Internal Connector Type: None
  External Reference Designator: USB1
  External Connector Type: Access Bus (USB)
  Port Type: USB

Handle 0x000A, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: J3A1
  Internal Connector Type: None
  External Reference Designator: USB2
  External Connector Type: Access Bus (USB)
  Port Type: USB

```

```
Handle 0x000B, DMI type 9, 17 bytes
System Slot Information
  Designation: J6B2
  Type: x16 PCI Express
  Current Usage: In Use
  Length: Long
  ID: 0
  Characteristics:
    3.3 V is provided
    Opening is shared
    PME signal is supported
  Bus Address: 0000:00:01.0

Handle 0x000C, DMI type 11, 5 bytes
OEM Strings
  String 1: $HP$
  String 2: ABS 70/71 79 7A 7B 7C
  String 3: FBYTE#6b7N7R7W8AaBaHapaqarauawbVbhbnbzdUdXdpdq.fD;BUILDID#15WW3K
  String 4: PT603#SAB8#DAB8;
  String 5:
  String 6:
  String 7:
  String 8:
  String 9:
  String 10:
  String 11:
  String 12:
  String 13:

Handle 0x000D, DMI type 22, 26 bytes
Portable Battery
  Location: Primary
  Manufacturer: 3332C
  Name: PK03056XL
  Design Capacity: 56540 mWh
  Design Voltage: 11400 mV
  SBDS Version: 1.1
  Maximum Error: Unknown
  SBDS Serial Number: 063C
  SBDS Manufacture Date: 2015-09-15
  SBDS Chemistry: LION
  OEM-specific Information: 0x000A070C

Handle 0x000E, DMI type 32, 20 bytes
System Boot Information
  Status: No errors detected

Handle 0x000F, DMI type 41, 11 bytes
Onboard Device
  Reference Designation: Onboard IGD
  Type: Video
  Status: Enabled
  Type Instance: 1
  Bus Address: 0000:00:02.0

Handle 0x0010, DMI type 7, 19 bytes
Cache Information
  Socket Designation: L1 Cache
  Configuration: Enabled, Not Socketed, Level 1
  Operational Mode: Write Back
  Location: Internal
  Installed Size: 64 kB
  Maximum Size: 64 kB
  Supported SRAM Types:
    Synchronous
  Installed SRAM Type: Synchronous
  Speed: Unknown
  Error Correction Type: Parity
  System Type: Data
  Associativity: 8-way Set-associative

Handle 0x0011, DMI type 7, 19 bytes
Cache Information
  Socket Designation: L1 Cache
  Configuration: Enabled, Not Socketed, Level 1
  Operational Mode: Write Back
  Location: Internal
  Installed Size: 64 kB
  Maximum Size: 64 kB
  Supported SRAM Types:
    Synchronous
  Installed SRAM Type: Synchronous
  Speed: Unknown
  Error Correction Type: Parity
  System Type: Instruction
  Associativity: 8-way Set-associative

Handle 0x0012, DMI type 7, 19 bytes
Cache Information
  Socket Designation: L2 Cache
  Configuration: Enabled, Not Socketed, Level 2
  Operational Mode: Write Back
  Location: Internal
  Installed Size: 512 kB
  Maximum Size: 512 kB
  Supported SRAM Types:
    Synchronous
  Installed SRAM Type: Synchronous
  Speed: Unknown
  Error Correction Type: Single-bit ECC
  System Type: Unified
  Associativity: 4-way Set-associative

Handle 0x0013, DMI type 7, 19 bytes
Cache Information
  Socket Designation: L3 Cache
  Configuration: Enabled, Not Socketed, Level 3
  Operational Mode: Write Back
  Location: Internal
  Installed Size: 4096 kB
  Maximum Size: 4096 kB
  Supported SRAM Types:
    Synchronous
  Installed SRAM Type: Synchronous
  Speed: Unknown
  Error Correction Type: Multi-bit ECC
  System Type: Unified
  Associativity: 16-way Set-associative

Handle 0x0014, DMI type 4, 48 bytes
Processor Information
  Socket Designation: U3E1
  Type: Central Processor
  Family: Core i7
  Manufacturer: Intel(R) Corporation
  ID: E3 06 04 00 FF FB EB BF
  Signature: Type 0, Family 6, Model 78, Stepping 3
  Flags:
    FPU (Floating-point unit on-chip)
    VME (Virtual mode extension)
    DE (Debugging extension)
    PSE (Page size extension)
    TSC (Time stamp counter)
    MSR (Model specific registers)
    PAE (Physical address extension)
    MCE (Machine check exception)
    CX8 (CMPXCHG8 instruction supported)
    APIC (On-chip APIC hardware supported)
    SEP (Fast system call)
    MTRR (Memory type range registers)
    PGE (Page global enable)
    MCA (Machine check architecture)
    CMOV (Conditional move instruction supported)
    PAT (Page attribute table)
    PSE-36 (36-bit page size extension)
    CLFSH (CLFLUSH instruction supported)
    DS (Debug store)
    ACPI (ACPI supported)
    MMX (MMX technology supported)
    FXSR (FXSAVE and FXSTOR instructions supported)
    SSE (Streaming SIMD extensions)
    SSE2 (Streaming SIMD extensions 2)
    SS (Self-snoop)
    HTT (Multi-threading)
    TM (Thermal monitor supported)
    PBE (Pending break enabled)
  Version: Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz
  Voltage: 1.0 V
  External Clock: 100 MHz
  Max Speed: 3100 MHz
  Current Speed: 3100 MHz
  Status: Populated, Enabled
  Upgrade: Other
  L1 Cache Handle: 0x0011
  L2 Cache Handle: 0x0012
  L3 Cache Handle: 0x0013
  Serial Number: To Be Filled By O.E.M.
  Asset Tag: To Be Filled By O.E.M.
  Part Number: To Be Filled By O.E.M.
  Core Count: 2
  Core Enabled: 2
  Thread Count: 4
  Characteristics:
    64-bit capable
    Multi-Core
    Hardware Thread
    Execute Protection
    Enhanced Virtualization
    Power/Performance Control

Handle 0x0015, DMI type 16, 23 bytes
Physical Memory Array
  Location: System Board Or Motherboard
  Use: System Memory
  Error Correction Type: None
  Maximum Capacity: 16 GB
  Error Information Handle: Not Provided
  Number Of Devices: 2

Handle 0x0016, DMI type 17, 40 bytes
Memory Device
  Array Handle: 0x0015
  Error Information Handle: Not Provided
  Total Width: 64 bits
  Data Width: 64 bits
  Size: 4096 MB
  Form Factor: Row Of Chips
  Set: None
  Locator: Bottom - on board
  Bank Locator: BANK 0
  Type: LPDDR3
  Type Detail: Synchronous
  Speed: 1600 MHz
  Manufacturer: Elpida
  Serial Number: Not Available
  Asset Tag: 0
  Part Number: EDFB164A1MA-JD-F
  Rank: 2
  Configured Clock Speed: 1600 MHz
  Minimum Voltage: Unknown
  Maximum Voltage: Unknown
  Configured Voltage: 1.2 V

Handle 0x0017, DMI type 17, 40 bytes
Memory Device
  Array Handle: 0x0015
  Error Information Handle: Not Provided
  Total Width: 64 bits
  Data Width: 64 bits
  Size: 4096 MB
  Form Factor: Row Of Chips
  Set: None
  Locator: Bottom - on board
  Bank Locator: BANK 2
  Type: LPDDR3
  Type Detail: Synchronous
  Speed: 1600 MHz
  Manufacturer: Elpida
  Serial Number: Not Available
  Asset Tag: 0
  Part Number: EDFB164A1MA-JD-F
  Rank: 2
  Configured Clock Speed: 1600 MHz
  Minimum Voltage: Unknown
  Maximum Voltage: Unknown
  Configured Voltage: 1.2 V

Handle 0x0018, DMI type 19, 31 bytes
Memory Array Mapped Address
  Starting Address: 0x00000000000
  Ending Address: 0x001FFFFFFFF
  Range Size: 8 GB
  Physical Array Handle: 0x0015
  Partition Width: 2

Handle 0x0019, DMI type 221, 12 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:00.1, MAC address 00:01:06:00:00:00

Handle 0x001A, DMI type 20, 35 bytes
Memory Device Mapped Address
  Starting Address: 0x00000000000
  Ending Address: 0x000FFFFFFFF
  Range Size: 4 GB
  Physical Device Handle: 0x0016
  Memory Array Mapped Address Handle: 0x0018
  Partition Row Position: 1

Handle 0x001B, DMI type 20, 35 bytes
Memory Device Mapped Address
  Starting Address: 0x00100000000
  Ending Address: 0x001FFFFFFFF
  Range Size: 4 GB
  Physical Device Handle: 0x0017
  Memory Array Mapped Address Handle: 0x0018
  Partition Row Position: 1

Handle 0x001C, DMI type 221, 26 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:00.3, MAC address 00:01:06:00:00:00
  NIC 2: PCI device 00:00.2, MAC address 00:00:00:33:00:03

Handle 0x001D, DMI type 221, 26 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:00.3, MAC address 00:01:06:00:00:00
  NIC 2: PCI device 00:00.2, MAC address 0A:00:00:01:00:03

Handle 0x001E, DMI type 221, 68 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:01.1, MAC address 00:01:06:00:00:00
  NIC 2: PCI device 03:00.2, MAC address FF:FF:FF:FF:FF:04
  NIC 3: PCI device ff:00.0, MAC address FF:FF:21:00:05:00
  NIC 4: Not Installed
  NIC 5: Not Installed
  NIC 6: Disabled
  NIC 7: Disabled
  NIC 8: PCI device 0a:00.0, MAC address 00:34:00:00:00:00

Handle 0x001F, DMI type 221, 54 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:00.7, MAC address 00:01:06:00:00:00
  NIC 2: PCI device 00:00.2, MAC address 01:06:00:01:00:03
  NIC 3: PCI device 01:00.0, MAC address 06:00:00:00:04:05
  NIC 4: Not Installed
  NIC 5: Not Installed
  NIC 6: PCI device 08:1f.7, MAC address 00:08:00:FF:FF:FF

Handle 0x0020, DMI type 41, 11 bytes
Onboard Device
  Reference Designation: Intel Stone Peak 2 7265 Combo /NON-vPro NGFF Combo Wireless-AC 7265
  Type: Other
  Status: Enabled
  Type Instance: 1
  Bus Address: 0000:02:00.0

Handle 0x0021, DMI type 41, 11 bytes
Onboard Device
  Reference Designation: Realtek PCIE CardReader
  Type: Other
  Status: Enabled
  Type Instance: 1
  Bus Address: 0000:01:00.0

Handle 0x0022, DMI type 221, 96 bytes
HP BIOS iSCSI NIC PCI and MAC Information
  NIC 1: PCI device 01:01.5, MAC address 00:00:00:00:FF:00
  NIC 2: PCI device 00:00.2, MAC address FF:FF:FF:FF:FF:03
  NIC 3: PCI device ff:00.4, MAC address FF:FF:FF:FF:05:06
  NIC 4: Not Installed
  NIC 5: Not Installed
  NIC 6: Disabled
  NIC 7: Not Installed
  NIC 8: PCI device 0c:00.0, MAC address 00:FF:FF:FF:FF:FF
  NIC 9: PCI device 00:01.5, MAC address 02:00:00:00:00:0E
  NIC 10: PCI device ff:00.0, MAC address FF:FF:FF:FF:0F:00
  NIC 11: Not Installed

Handle 0x0023, DMI type 8, 9 bytes
Port Connector Information
  Internal Reference Designator: Ctrl0Port1
  Internal Connector Type: SAS/SATA Plug Receptacle
  External Reference Designator: Primary HDD Bay
  External Connector Type: SAS/SATA Plug Receptacle
  Port Type: SATA

Handle 0x0024, DMI type 136, 6 bytes
OEM-specific Type
  Header and Data:
    88 06 24 00 00 00

Handle 0x0025, DMI type 14, 23 bytes
Group Associations
  Name: Firmware Version Info
  Items: 6
    0x0019 (<OUT OF SPEC>)
    0x001C (<OUT OF SPEC>)
    0x001D (<OUT OF SPEC>)
    0x001E (<OUT OF SPEC>)
    0x001F (<OUT OF SPEC>)
    0x0022 (<OUT OF SPEC>)

Handle 0x0026, DMI type 14, 8 bytes
Group Associations
  Name: $MEI
  Items: 1
    0x0000 (<OUT OF SPEC>)

Handle 0x0027, DMI type 219, 81 bytes
HP ProLiant Information
  Power Features: 0x45010301
  Omega Features: 0x06900002
  Misc. Features: 0x00000000
    iCRU: No
    UEFI: No

Handle 0x0028, DMI type 13, 22 bytes
BIOS Language Information
  Language Description Format: Long
  Installable Languages: 5
    en|US|iso8859-1
    fr|FR|iso8859-1
    es|ES|iso8859-1
    zh|TW|unicode
    zh|CN|unicode
  Currently Installed Language: en|US|iso8859-1

Handle 0x0029, DMI type 127, 4 bytes
End Of Table
```

Code 5.10: logs/dmidecode.log