

Application of Intrusion Detection System in Automatic Evidence Collection using Digital Forensics

*A thesis submitted in
in partial fulfillment of the requirements
for the degree of*

Master of Technology

by

Ankit Kumar Jain

(Roll: 212cs2106)



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India

Application of Intrusion Detection System in Automatic Evidence Collection using Digital Forensics

*A thesis submitted in
in partial fulfillment of the requirements
for the degree of*

Master of Technology

in

Computer Science and Engineering

by

Ankit Kumar Jain

(Roll: 212cs2106)

under the supervision of

Prof. Sanjay Kumar Jena



Department of Computer Science and Engineering
National Institute of Technology Rourkela
Rourkela – 769 008, India



Department of Computer Science and Engineering
National Institute of Technology Rourkela

Rourkela-769 008, India. www.nitrkl.ac.in

Prof. Sanjay Kumar Jena

Professor

Certificate

This is to certify that the work in the thesis entitled *Application of Intrusion Detection System in automatic evidence collection using Digital Forensics* by *Ankit Kumar Jain*, bearing roll number 212cs2106, is a record of an original research work carried out by him under my supervision and guidance in partial fulfillment of the requirements for the award of the degree of *Master of Technology in Computer Science and Engineering*. Neither this thesis nor any part of it has been submitted for any degree or academic award elsewhere.

Sanjay Kumar Jena

Acknowledgment

Foremost, I would like to express my sincere gratitude to my advisor *Prof. Sanjay Kumar Jena* for the continuous support of my M.Tech study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my M.Tech study.

Besides my advisor, I extend my thanks to our HOD, *Prof. S. K. Rath*, *Prof. A. K. Turuk*, *Prof. B. D. Sahoo* and *Prof. K. S. Babu* for their valuable advices and encouragement.

My sincere thanks also goes to Ph.D. Scholar *Mr. Asish Dalai* for constantly guiding me throughout the work.

Last but not the least, I would like to thank my family: my mother Smt. Sunita Jain and my cousin Mr. Manish Jain for constantly supporting me throughout my life. Words fail me to express my gratitude to my beloved mother, who sacrificed her comfort for my betterment.

Ankit Kumar Jain

Roll: 212CS2106

Department of Computer Science

Author's Declaration

I hereby declare that all work contained in this report is my own work unless otherwise acknowledged. Also, all of my work has not been submitted for any academic degree. All sources of quoted information has been acknowledged by means of appropriate reference.

Ankit Kumar Jain

Roll: 212CS2106

Department of Computer Science

Abstract

Internet has changed and improved the way of working in organizations and businesses, at the same time this large network also opened doors for attackers as new attacks are emerging day by day. To protect the organizations and systems from these attacks, network security comes into action. Network security is the policy made by the administrator by running security software or installing appliances in the network in order to protect the network and its resources from security threats.

In network security, Intrusion Detection System (IDS) is one of the popular and effective mechanism to secure the network. The aim of IDS is to offer a layer of protection against unauthorized (or malicious) uses of systems by sensing the vulnerability in the system or misuse of a security policy, and alerts system administrator to an ongoing (or recent) attack. IDSs function is limited to detect the intrusion and respond to administrator about the intrusion by monitoring the system continuously. IDS is not able to preserve evidence about the intrusion, which makes it difficult to see the damage in the system and gather information about the attack and hence make it impossible to catch the intruder. Although evidence can be collected from IDS's and system log files, but integrity, reliability, and completeness of such evidence are doubtful as log files can also be altered by an intruder.

In order to preserve evidence in its original form, we have proposed Application of Intrusion Detection System in automatic Evidence Collection using Digital Forensics. In our model whenever an intrusion is detected, IDS notifies the administrator by sending an alert as well as activate the digital forensic tool to capture the current state of the system. This captured system image contains all the information about the system of the time when the attack was taking place. Hence such image can be used as evidence in legal proceedings. We used both signature based IDS and anomaly based IDS in the work and observe that signature based IDS is not able to detect novel threats while anomaly based IDS is able to detect such threats.

Keywords: Intrusion detection system, digital forensics, digital evidence, signature, anomaly, forensic analysis.

Contents

Certificate	ii
Acknowledgment	iii
Author’s Declaration	iv
Abstract	v
List of Figures	vii
List of Tables	ix
1 Introduction	2
1.1 Network Security	2
1.1.1 Necessity of Security	3
1.1.2 Threats to Network Security	4
1.1.3 Solutions to achieve Network Security	5
1.1.4 Difference between IDS and Firewall	6
1.2 Introduction to Intrusion Detection System (IDS)	7
1.2.1 Architecture of Intrusion Detection System	8
1.2.2 Classification of IDS	9
1.2.3 Placement of IDS in network	18
1.2.4 Characteristics of IDS	19
1.3 Introduction to Digital forensics (DF)	19
1.3.1 Fundamental principle in Digital forensic investigation	20
1.3.2 Digital Forensic Model	21
1.3.3 Relevant Forensic Techniques:	23
1.3.4 Limitation of Digital Forensics	24
1.3.5 Digital evidence and its admissibility	25
1.4 Combination of IDS and Digital Forensics	26
2 Literature review	27
3 Proposed Model for Automatic Evidence Collection	42
3.1 Model with Signature based IDS	43
3.1.1 System model and Assumptions	43
3.1.2 Methodology	45
3.1.3 Tools	48
3.2 Model with Anomaly based IDS	50
3.2.1 System model and Assumptions	50
3.2.2 Methodology	51
3.2.3 Tools	53
3.2.4 Port scan attack	56

4 Results and Discussions	57
4.1 Results for Model with Signature based IDS	57
4.2 Results for Model with Anomaly based IDS	66
4.3 Comparison of Models	74
5 Conclusion	76
Bibliography	78

List of Figures

1.1	Generalized architecture of IDS.	8
1.2	Classification of Intrusion detection system.	10
1.3	Signature based detection system.	12
1.4	Anomaly based detection system.	13
1.5	Placement of Intrusion Detection System in Network.	18
1.6	Area of application of digital forensics.	20
1.7	Digital Forensic Investigation Model.	21
3.1	System model with Signature Based Intrusion Detection System.	43
3.2	Flowchart for model with Signature based IDS.	46
3.3	System model with Anomaly Based Intrusion Detection System.	51
3.4	Flowchart for model with Anomaly based IDS.	52
4.1	Alert message on console.	59
4.2	DumpIt capturing the memory image.	59
4.3	Selecting plugins for analysis.	60
4.4	Analyzing memory image using memgator.	60
4.5	Homepage of report.	61
4.6	List of avtive processes.	62
4.7	Connections to the system.	62
4.8	List of open files.	63
4.9	List of loaded modules.	63
4.10	Number of alerts generated.	64
4.11	Types of alerts generated.	65

4.12	Alert files generated by Bro.	69
4.13	Entries in notice.log file.	70
4.14	Capturing memory image.	71
4.15	List of all active processes.	71
4.16	List of all open files.	72
4.17	Entries in System call table.	72
4.18	Alert generated in notice.log.	73
4.19	Connections in conn.log.	74
4.20	Information retrieved from both models.	75

List of Tables

4.1 Comparison of Models	75
------------------------------------	----

Chapter 1

Introduction

1.1 Network Security

The web has enormously enhanced and converted the way of working of businesses, in the meantime, this huge network and technologies associated with it expanded the amount of security dangers from which organizations must be protected. Security has developed in two different directions one is data security and other is network security [1]. Although network threats are more serious for the organizations that store delicate information, for example, money related or medical records, individual data, the outcomes of dangers on any element differ from a bit awkward to completely incapacitating. Consequences of the attack can be loss of important data, privacy violation, and hours or even days of network or system downtime. Because of more network vulnerabilities and equipped hackers the risk of network related attacks are growing and becoming a serious concern [2]. Considering the importance of security, Internet is a dangerous means of vulnerability so the risk involve in it should be looked carefully and security measures should be taken. Network security is a very challenging area in the modern world, consists of policies and provisions [3] implemented by the administrator to stop unwanted access, disavowal, and alteration or misuse of the available network assets. Attackers do various malicious activities

to break the security protocol of the network. Security for a network has a number of solutions to protect the network.

- Anti-spyware and anti-virus.
- Firewall- to protect the network from unauthorized access.
- Intrusion Detection System (IDS) - to detect suspicious activities.
- Virtual Private Networks (VPN) - to give safe access in a client-server environment.

1.1.1 Necessity of Security

Security gives a secure platform for users, programs and computers to perform their noteworthy work in a secured environment. A protected system secures its clients from worms, Trojan Horses, and viruses as they are sufficiently dangerous to take all the stored data from the system or occupy the system asset. Security is necessary to achieve following goals.

Authentication: Authentication enables the user to ensure the identity of the other user to which it is communicating [4]. A simple way to conduct authentication is by using user login passwords, biometrics and digital certificates and single sign-on (SSO) systems [5].

Confidentiality: Confidentiality makes the data distant to unauthorized user by guaranteeing that the data send by confirmed gathering does not get spilled or revealed to any unauthorized person or system [4]. A condential message means does not reveal its meaning to an eavesdropper [6].

Data integrity: Data integrity ensures that the received data is not modified or altered by the adversary during transmission [4]. Integrity can be ensured by

controlling the physical environment, maintaining precise authentication policies, and restricting access to data [6].

Non-repudiation Non-repudiation guarantees that a party cannot deny the information provided on a document or legitimacy of their signature or sending of a message that is actually originated by them [4].

1.1.2 Threats to Network Security

New threats to security are emerging consistently. These threats can be slightly harmful or can be fully devastating. Major security threats are:

Buffer overflow A buffer overflow happens when more data is supplied to buffer than it can hold by a process. Buffer overflow may happen accidentally through programming error or intentionally to launch an attack. It is an attack on integrity constraint of security. In these attacks, the additional data can be the code intended to do particular noxious activities.

Backdoor trojan It creates a backdoor in the system in order to get in to it. The backdoor Trojans are too dangerous as they have the ability to access machines remotely and may gain administrative privileges. Backdoor Trojan allows hackers to virtually sit at the victims keyboards.

Denial of Service attack A Denial of Service (DoS) attack, prevent legitimate users from accessing desired resources which were intended for them such as network bandwidth and web services. DoS attack is done by disrupting a legitimate users connectivity or services by exhausting bandwidth, router processing capacity, sockets, memory, CPU, I/O bandwidth, and database/disk bandwidth [7]. Flooding is a very common way to do this is.

Arbitrary code execution It is a vulnerability in the system that can be used by an attacker to run any script or program on a target machine. The attacker can do it by injecting malicious code into the process, which when run code get executed automatically.

Privilege escalation It is the process of gaining access to protected resources by misusing operating system bugs. Privilege escalation enables an application to perform more unauthorized actions with more privileges.

SQL injection SQL injection is an important vulnerability in web applications, which is present in applications that suffer from insufficient input validation. To exploit this vulnerability the attacker injects a piece of malicious SQL query into the application through the variables used in the application. Then web applications will send these queries to the database server where they get executed along with the legitimate query. A successful SQL injection attack can enable the attacker to read, insert, update, execute and delete SQL queries on the database [8].

1.1.3 Solutions to achieve Network Security

In order to deal with the security threats number of solutions have also emerged and still emerging day by day. Some popular security solutions are:

Intrusion detection system

It is a hardware appliance or a software designed to monitor the traffic in the network to find unusual or unauthorized activities and informs the system administrator. IDS have turned into a vital addition to the security for most of the organizations. Apart from detection many IDS can also respond to an attack to stop it. A number of techniques can be used such as IDS preventing itself from attack, change the security rules, or change the content of attack data [9].

Firewalls

A firewall is a set of devices or programs designed for filtering the traffic flowing through it. In firewall, set of rules are written to filter the traffic to pass or deny. It is located at a network gateway server to protect the resources of the network from the outside world. A firewall protects networks from unauthorized access by denying the malicious traffic while permitting authentic traffic to pass. A network firewall makes an overpass between the network that is being protected and an outside network, such as the public Internet [5].

Honeypot

A honeypot is a computer system or device on the Internet that is specifically set up to attract and trap the people who try to penetrate others systems or network. A honeypot allows evil patterns to enter into the network so that their activities can be examined to further enhance the quality of the of the system. The prime function of a honeypot is to analyze the attacker's behavior to conduct and launch the attack. Unlike firewalls that uses defensive security mechanism, honeypot work on offensive security mechanism [10].

1.1.4 Difference between IDS and Firewall

In the beginning firewalls gave good solution to secure the network. Limitation of the firewall was found later that a firewall makes traffic flow control decision by analyzing data packet header, but do not analyze the entire data of the packet. This limitation makes firewalls insufficient to protect the network on their own, although they are essential components of network security. These inefficiencies can be overcome by deploying the IDS either inside or outside the firewall.

- Firewall analyzes packet headers and applies protocol type based policy, addresses and ports while IDS analyzes entire packets including header and data, searching for malicious events.

- An Intrusion Detection System (IDS) detects unwanted attempts, whereas Firewall blocks unauthorized connections.
- IDS uses Signature and behaviour matching to detect the attacks, whereas Firewall scan the packets and block or allow the traffic based on the source and destination addresses.
- The firewall can be thought as a security guard and IDS as a security camera at the entrance.

1.2 Introduction to Intrusion Detection System (IDS)

When a hacker or system user exploits or breaks into the system by taking illegal action, is called intrusion. The intruder may be an insider or an outsider, who does malicious activities or actions in an unauthorized manner. Intrusion detection is a process of finding unauthorized activities (intrusion) by inspecting the inbound traffic. Intrusion detection involves observation and analysis of user and system activities, auditing of vulnerabilities and configurations of the system, evaluating the integrity of data files and critical system, statistical analysis of activity patterns, analysis of abnormal activities, and audit of operating system [11].

First concepts about Intrusion Detection System (IDS) was given in the early 80s by James P. Anderson. Intrusion detection system can be a physical appliance or security software to monitor the network traffic in order to detect suspicious activity. Many IDS keep information about the detected intrusions in a log file for further analysis or to combine these logs with other data to make policy and decisions. Most of the intrusion detection system detects suspected intrusion and then informs to the system administrator by sending an alert. Originally IDS was thought as a single, stand-alone system based on audit records processing based detection. Today IDS is a distributed system which contains multiple systems combined together [12].

1.2.1 Architecture of Intrusion Detection System

The generalized architecture of IDS is shown in Figure 1.1 [13]. A general IDS consists of three basic components.

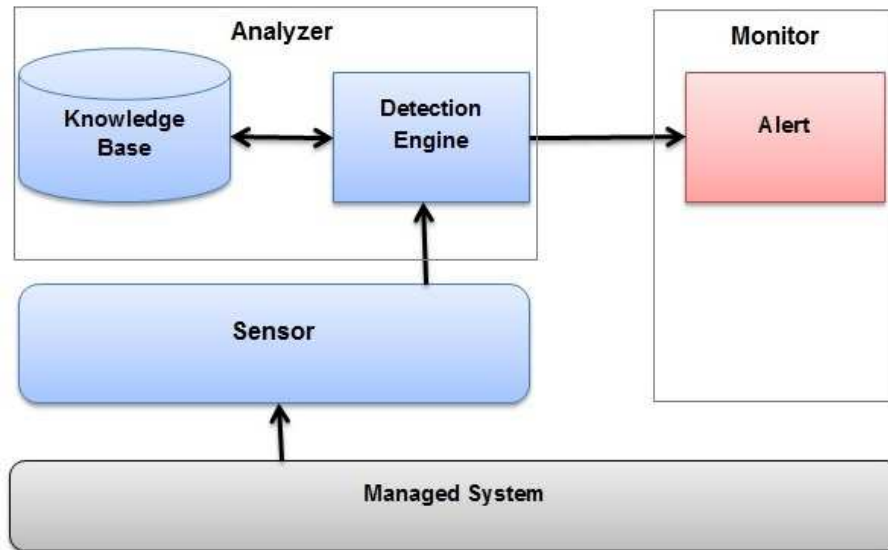


Figure 1.1: Generalized architecture of IDS.

Sensors

The sensor or an event generator is the component responsible for data collection. The event generator is responsible to take packets from large outside network and provide them to the IDS in form of events [14]. For example, event generators might be simple filters that take audit trails, observe a network and create relevant events for the packets in the network, or application code in a database which generates events describing database transactions.

Analyzers

The raw data collected by the sensors is submitted to the analyzers to classify either as an attack or dismissed as safe. Analyzer contains two components, detection

engine and knowledge base. Detection engine filters information and throw-outs irrelevant data found by event generator. The analyzer has a knowledge base that comprises the information such as profiles of normal system behavior, attack signatures, essential attributes (i.e. thresholds). Initial analysis determines severity of the attack, then further analysis determines scope, intention, or frequency of the attack. A significant amount of historical data storage may require for accurate analysis. Good analyzer can correlate two attacks or determine the inappropriate correlation [15].

Monitor

It is an interface to send a notification to the administrator about the threat detected by IDS. Adjustment of the monitor must be done according to the network traffic.

1.2.2 Classification of IDS

Intrusion detection elements can be categorized according to its primary components and assumptions [9]. Intrusion detection systems can be categorized based on five parameters:

- (i) Intruder Type
- (ii) Detection methodology
- (iii) Deployment based
- (iv) Detection Behavior
- (v) Processing of collected data

Classification of IDS based on above five parameters is shown below in Figure 1.2 [9].

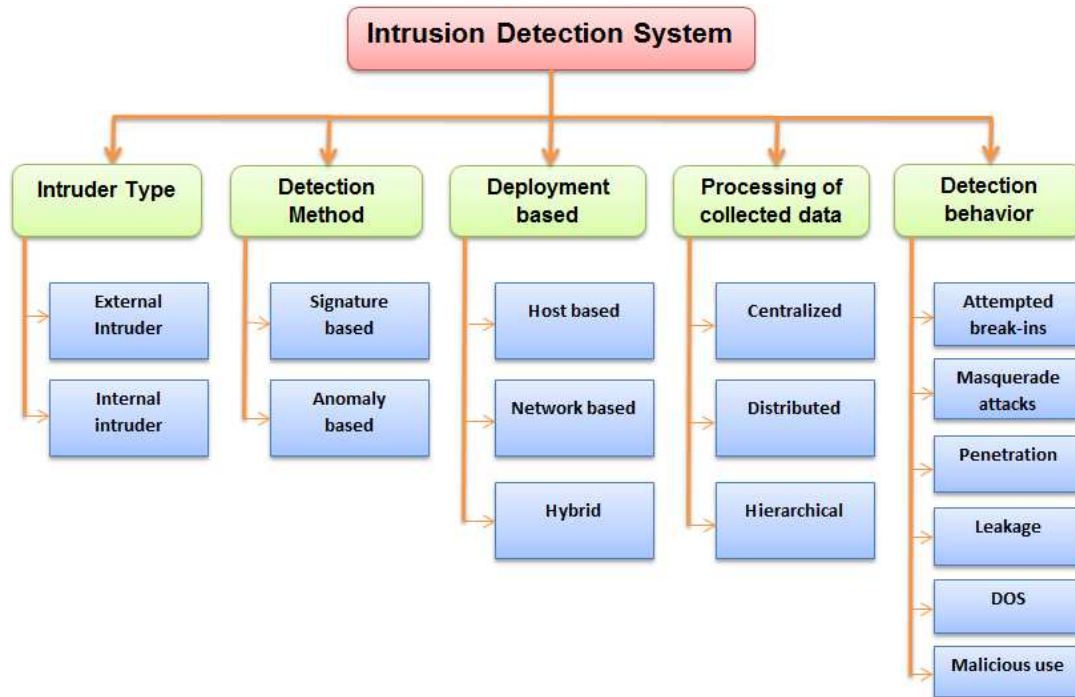


Figure 1.2: Classification of Intrusion detection system.

(i). Intruder type

IDS classified based on the users (intruders) are of two types, internal and external. External intruders don't have any authority to use the machine they attack. On the other hand internal intruders have limited authority to the machine, but try to perform additional activities (i.e. Administrative activities) for which they are not authorized. Internal intruders are of three type masqueraders, misfeasor and clandestine [16].

- Masquerader is a person who enters into a system's access control to exploit the account of a legitimate user.
- Misfeasor is a genuine user who accesses resources, programs, or data for which such access is not authorized, or misuses the privileges of the authorized access.
- Clandestine user is a person who takes hold of system's supervisory control and uses it to avoid the auditing and access controls or to destroy audit collection.

Clandestine intruders are most dangerous one.

(ii). Detection method based

An IDS that detects intrusion according to security policies, are based on one of the detection methodologies. Based on detection methodology IDS are either Signature-based or anomaly-based. More than one detection methods are used, in maximum intrusion detection techniques for more accurate and broad detection.

Signature based: Signature based intrusion detection systems use known attacks signature to detect intrusion. Signature based detection is also known as misuse detection. In this approach, at first abnormal activities (signature) are defined, and any other undefined is treated as normal activity. A signature is a pattern related to some known attack. This detection methodology matches the signatures of observed events to known attack signatures to identify attack [17]. This method of intrusion detection is the simplest one. Examples of signatures are as follows:

- An attempt of Telnet login with root username, which violates the security policy of any organization.
- An e-mail with freepics.exe file attached, which contains malware characteristics.
- An OS log file entry having status code 645, which indicates disabled host's auditing.

Procedure for signature based detection is shown in Figure 1.3 [18].

Signature based IDS can be classified into further categories based on the techniques used such as string matching, simple rule based, state modeling based, expert system based [19]. Signature based detection is quick and effective for identifying known attacks yet profoundly inadequate in detection unknown attacks, different variants of known attacks, and attacks masked by some evasion techniques. For instance, if an attacker changed the malware in above mentioned example to use

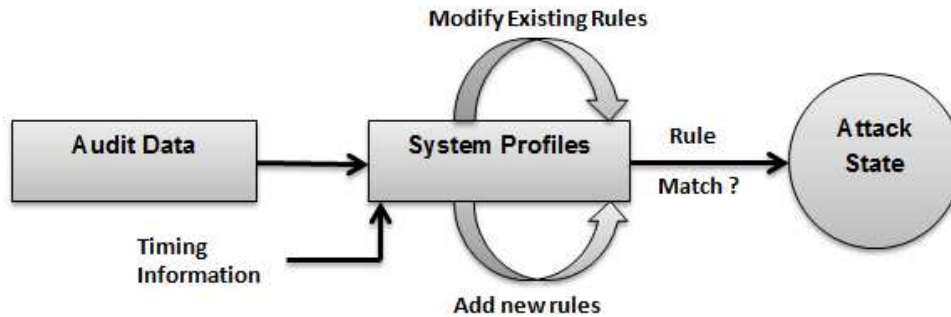


Figure 1.3: Signature based detection system.

a filename of "my_freepics.exe" rather than "freepics.exe" than a signature searching for "freepics.exe" will treat it as different and hence will not match. It fails to detect truly new attacks, also known as novel attacks.

Anomaly based: Anomaly based detection is the methodology of collating the definitions of a new activity against the normal activity to find the deviations. This is based on statistical behavior modeling. Normal operations of the members are profiled and a certain amount of deviation from the normal behaviour is hailed as an anomaly. An IDS using anomaly based detection represents the typical behaviour of things, for example, hosts, users, applications, or network connections by utilizing the profiles. The profiles are created by observing the normal activities over a certain period of time. Case in point, a profile for a system may indicate that 17% of network bandwidth is used by the internet border for the Web activity on an average during a normal day. The IDS then compare the incident activity's attributes to thresholds defined in the profile. For example, when Web activity consumes bandwidth more than 17%, then IDS treats it as intrusion sent an alert to the administrator for the anomaly. Profiles can be generated for numerous behaviour, for example, the number of failed login attempts for a host, the number of e-mails sent by a user, and usage of processor for a host in a given duration.

Profiles are developed by monitoring the system and network for a certain period of time (typically days, sometimes weeks). Anomaly based IDS can have either static

profile or dynamic profile. Static profile does not change while a dynamic profile updates regularly when new events occur. Procedure for anomaly based detection is shown in Figure 1.4 [18].

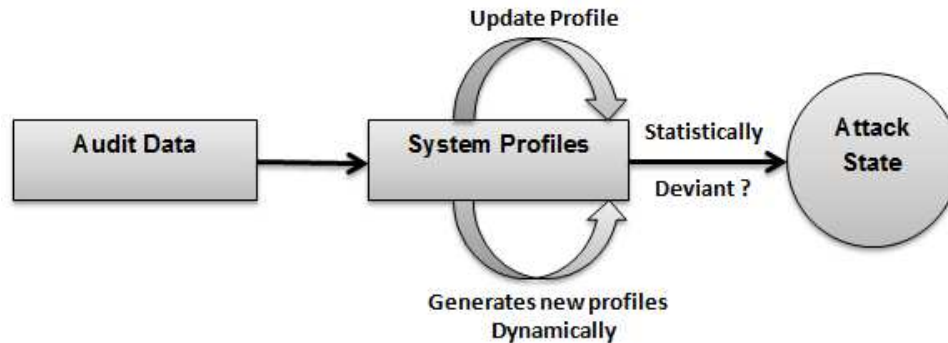


Figure 1.4: Anomaly based detection system.

The advantage of anomaly based detection is that it is suitable to detect novel attacks. Case in point, assume that a computer gets contaminated with a malware. The malware could devour the processing resources of computer, initiate large number of connections, send e-mails or perform other activities that are different from normal activities of the computer.

Some common examples of attack that can be detected by anomaly based IDS but not by signature based IDS are Probe attack, DoS (Denial of Service) attack, R2L (Remote to Local) attack, U2R (User to Root) attack etc.

- *Probe attack*: The probe attacks are intended to acquire the information about the target network. Examples of probe attack are port-scan, ping-sweep, etc [20].
- *DoS (Denial of Service) attack*: It is a sort of attack on a system that is planned to flood the system or network and force the target system to stop the service(s) by flooding it with futile data. Examples of DoS attack are ping-of-death, teardrop, smurf, SYN flood, etc [21].
- *R2L (Remote to Local) attack*: Unauthorized access from a remote machine.

The R2L attacks are most troublesome to identify as they contain the network level characteristics (duration of connection and service requested) and the host level characteristics (number of failed login attempts). Example of R2L attack is guessing passwords [20].

- *U2R (User to Root) attack*: Unauthorized access to local superuser privileges by a local unprivileged user. The U2r attacks contains the semantic points of interest that are extremely testing to find at a beginning stage. Such attacks target an application and are often content based. Examples of U2R attacks are various buffer overflow attacks [20].

Main problem with an anomaly based IDS products is unintentionally inclusion of malicious activity in the profile. Another problem is generating accurate profiles because of complex computing activity. Anomaly based IDS frequently generates more false positives simply on the grounds that the benign activities show huge deviation from profiles, particularly in more dynamic or different situations. Another significant issue with the anomaly based detection system is that, it is troublesome for analyst to find the cause of particular alert, to correctly validate an alert, because of the amount of events and complex nature of events that may have caused the alert. [17].

Anomaly based IDSs are further divided into three categories are as follows:

Statistical based: In statistical based anomaly intrusion detection systems, the network traffic is captured and after that a profile of its stochastic behaviour is produced. As the network operates in normal conditions (without any attack), a reference profile is created. After that, the network is monitored and profiles are generated and an anomaly score is generated by collating it to the reference profile. If the score passes a certain threshold, the IDS will flag an occurrence of the anomaly [22]. Any one of three statistical methods, univariate, multivariate, or time series model can be used.

Knowledge based: Knowledge based anomaly IDSs rely on the availability of the prior knowledge (data) of the network parameters in normal operating condition as well as the one under certain attacks [22]. Knowledge base used in IDS may be expert systems, description languages(UML), finite state machines, data clustering.

Machine learning based: In machine learning based anomaly IDSs, an explicit or implicit model of the analyzed patterns is generated. These models are updated periodically, in order to improve the intrusion detection performance on the premise of the past results. Machine learning techniques such as Bayesian networks, Markov model, fuzzy logic, genetic algorithm, neural networks, or principle component analysis can be used [22].

(iii). Deployment based

IDS can be deployed on a individual host or in a segment of the network. Based on the deployment, IDS can be of following types:

Host based IDS (HIDS): Host-based IDS observes the characteristics of an individual host and the events happening on that host for malicious actions. The HIDS is generally deployed to the critical hosts to provide protection by monitoring the system entities and their attributes. Such systems typically make use of the information specific to operating systems of the target machines. HIDS monitors incoming and outgoing wired and wireless network traffic, system logs, running processes, file access and modification, system and application configuration changes. HIDS reports intrusion by writing logs, sending e-mails, etc. HIDS use database to store objects and attributes. Host-Based IDS can be divided into four types [23]:

- **File system monitors:** Systems checking the integrity of files and directories.
- **Log file analysers:** Systems analyzing log files for patterns showing malicious activity.

- **Connection analysers:** Systems that monitor connection attempts from and to a host.
- **Kernel based IDSs:** Systems that detect kernel level suspicious activity.

HIDS has an advantage that it provides detailed information about the attack and is also known as System Integrity Verifier. HIDS is also have some limitations such as HIDS itself can be attacked (If an attacked host is down, then HIDS is also down), local installation on each host, and much host resources are needed [24]. Despite the fact that HIDS is much superior to NIDS in detecting malicious activities for an individual host yet they have restricted perspective of network topology and henceforth can't detect attacks that is focused for another host in a network where HIDS installed [25]. HIDS products such as Emerald eXpert-BSM, Dragon Squire, Intruder Alert, NFR HID, and Snort are popular.

Network Based IDS (NIDS): A network based IDS generally contains network device with a Network Interface Card (NIC) that operates in promiscuous mode. The IDS is placed on the boundary or along a network segment to monitor most of the traffic on that network segment. NIDS can be passively deployed, without many changes to networks or systems. NIDS are very effective for monitoring both outbound and inbound traffic. Load balancing in NIDS allows it to efficiently utilize the processing power of the nodes in a distributed system for scalability [26]. A single NIDS is able to protect the entire network and turning off a target system will not affect the NIDS. But NIDS cannot monitor the network if the system bandwidth is overloaded [24].

Hybrid IDS: Hybrid IDS improves HIDS technology by making it capable to monitor the network traffic flowing through the host, either into or out from individual host. Hybrid agents unite the functionality of HIDS with network based sensor technology that is just to analyze the network traffic focused to the particular

host where the hybrid agent is installed. The processor utilization in a hybrid agent is much more than a host-based agent [27].

(iv). Processing of collected data

Data are collected from one or more sources in intrusion detection systems. Collected data is stored and processed to detect intrusion. Based on storage and processing IDS can be classified as follows:

Centralized IDS: A centralized IDS is a system where the data analysis is performed in a fixed number of places, that does not depend on the number hosts are being monitored. Location is considered of only analysis components. Some examples of centralized IDS are: IDES, IDIOT, NADIR, and NSM [28].

Distributed IDS: A distributed IDS is the system where the data analysis is done at numerous locations, may be equivalent to the number of hosts that are being monitored. Locations and number are considered of just the analysis components, not the collection components. Some examples of distributed IDS are: DIDS, GrIDS, EMERALD, and AAFID [28].

Hierarchical IDS: This is proposed for multi-layer (clustering) network infrastructures. Cluster heads (CHs) are responsible for monitoring their member nodes, as well as participating in the global intrusion detection decisions.

(v). Detection behavior

Based on detection behavior IDS can be classified as follows [9]:

Attempted break-ins: Detected by common behaviour profiles or by security constraint violations.

Masquerade attacks: Also detected by common behaviour profiles or by security constraint violations.

Penetration of the security control system: Detected by searching for particular activity patterns.

Leakage: Detected by common utilization of system resources.

Malicious use: Detected by common behaviour profiles, utilization of special privileges, by security constraint violations.

1.2.3 Placement of IDS in network

Depending on the network topology, IDS can be placed at one or more places as shown in Figure 1.5 [29]. All traffic travels through an IDS sensor when the IDS is

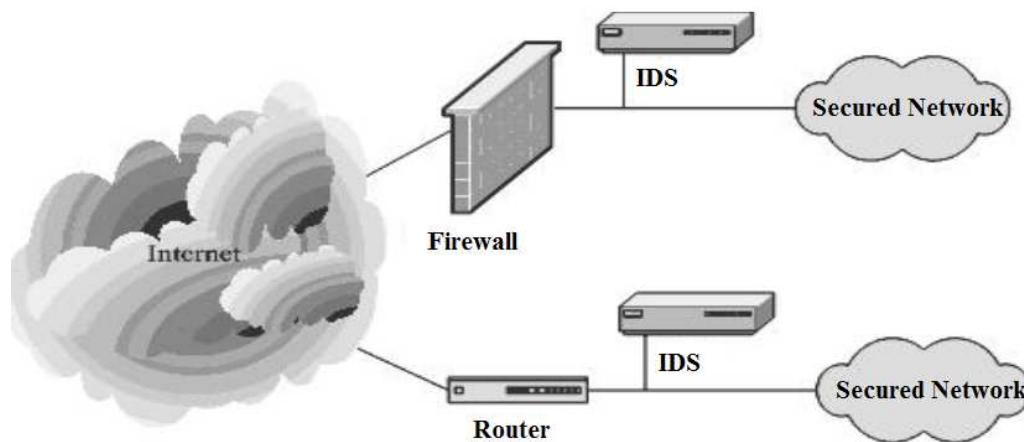


Figure 1.5: Placement of Intrusion Detection System in Network.

placed in an inline configuration. Effectiveness of an IDS can be improved if all traffic passes through the sensor. When IDS is not configured in an inline configuration, it sends a reset packet to the network to terminate the malicious session [29]. It likewise relies on what kind of intrusion activities are to be detected: outside, inside or both. If outside intrusion activities need to be detected, and only one router is connected with the web, the best place for an IDS is inside the router or a firewall. If there are different paths to the web, one IDS might be put at every passage point. Then again, if inner threats need to be detected, IDS can be placed at each network segment. More number of IDS mean more work and more maintenance expenses.

1.2.4 Characteristics of IDS

An IDS should have the following characteristics, irrespective of what mechanism, it is based on:

- It must run consistently without human intervention. The system must be capable to run in the background of the system that is being monitored.
- It should not work as a “black-box”. Its internal functionalities should be examinable for outsiders.
- It must be fault tolerant. It must cope even in system crash and its information base remake should not be required at restart.
- It must oppose subversion. They should have the capacity to monitor itself to guarantee that it has not been destabilized.
- It must impose less overhead on the system. It should not slow down the computer.
- It must be able to observe significant deviations from typical system behaviour.
- It must be effortlessly customized to the system. It should be able to adapt the diverse use of pattern and defense mechanism effectively as they are distinctive for each system.
- It must survive with change in system behaviour and the environment over the time as new applications and advance technologies are continuously included.

1.3 Introduction to Digital forensics (DF)

Digital forensics (also known as digital forensic science) is a branch of forensic science deals with the recovery and investigation of materials found in digital devices. Digital Forensics Investigation is a methodology of determining and relating

concentrated data and digital evidence to find accurate data for legal proceeding [30]. Forensic science is characterized as "the application of science and engineering in the law" [31]. Actually an investigation is partitioned into a few sub-extensions, based on the kind of digital devices included, network forensics, computer forensics, mobile device forensics, forensic data investigation. DFRWS (Digital Forensic Research Workshop) characterized Digital forensics as Use of scientifically derived and proven procedures for the collection, preservation, validation, analysis, identification, interpretation, and documentation of digital evidence found from digital sources for the purpose to facilitate the reconstruction of events found to be suspicious, or to help in anticipation of unauthorized actions shown to be disturbing to normal operations [32].

Increasingly the world is becoming dependent on the information derived from digital sources, computer systems, and networks involved in storing, processing, and transmitting data. This growing dependence drives development to advance required technology. Figure 1.6 shows the area where digital forensic comes into action [32].

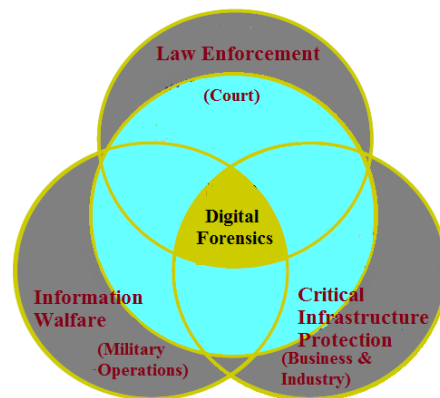


Figure 1.6: Area of application of digital forensics.

1.3.1 Fundamental principle in Digital forensic investigation

Core IT Security has fundamental principles as Confidentiality, Integrity and Availability. Similarly, digital forensics investigation also has core principles to view

the fundamental concept of different digital forensic investigation techniques. The core principles in digital forensic investigations are as follows:

- **Reconnaissance:** This is an observation of actions to be taken to secure and collect digital evidence in such a manner that they should not affect integrity of evidence. For this, forensics investigator uses different tools, methods, and practices specially developed for that and store on different storage media to make readable evidence [33].
- **Reliability:** Data extraction is not simply to save files to a disk or copying the data. Evidence-chain should be preserved during extraction, analysis, storage and transportation of data to make the forensic process reliable [33]. Trained persons should be appointed to conduct an examination of digital evidence.
- **Relevancy:** Relevancy of the evidence with the case may affect the usefulness and weight of the evidence, even though for admissible evidence [33]. For better and controlled investigation, advice from legal practitioner should be taken on what should be collected, time and cost spent during the process [34].

1.3.2 Digital Forensic Model

Several models for digital forensic investigation have been proposed. One most popularly used model of them is given in Figure 1.7 [35].

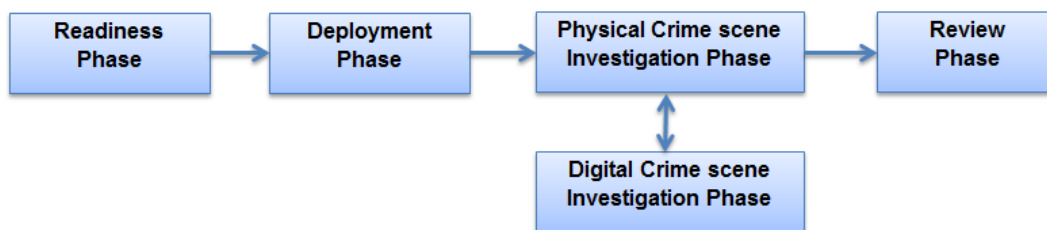


Figure 1.7: Digital Forensic Investigation Model.

Readiness phases

The concept of forensic readiness for a system describes the capability of the system to efficiently collect credible digital evidence that can then be used in legal proceedings [36]. Readiness phase is responsible to verify the ability of infrastructure and operations to support an investigation completely. It combines two phases:

- Operations Readiness phase, that ensures human capacity, whether it is capable or not to deal with incoming incidents.
- Infrastructure readiness phase, that ensures the infrastructure is sufficient or not to deal with incoming incidents.

Deployment phases

Deployment phase provides a mechanism and set up for detection and confirmation of an event. It also combines two phases:

- Detection and Notification phase, that detects the incident and notify investigator about it.
- Confirmation and Authorization phase, that verify the occurred incident and take authorized legal approval for the investigation.

Physical Crime Scene Investigation phases

Physical crime scene investigation phase is responsible for the collection and analysis of the physical evidence and reconstruction of the actions that took place during the occurrence of events. It contains six phases such as preservation phase, survey phase (taking photographs, sketches, and videos), documentation phase, search and collection phase, reconstruction phase, and presentation phase.

Digital Crime Scene Investigation phases

Digital Crime Scene Investigation phase is responsible for the collection and analysis of the digital evidence found either from the previous phase or from any other source of evidence collection. This phase is similar to physical crime scene investigation phases, but this phase mainly focuses on the digital evidence.

Review phase

This phase reviews the whole investigation process and identifies areas where improvement is needed.

1.3.3 Relevant Forensic Techniques:

Some popularly used forensic techniques are discussed below.

Imaging

A first technique used in a digital forensics investigation is to copy, or image the media that is to be examined. Once the image of media is captured then it can be used for further investigation to search the evidence. The problem with this technique is that media can be modified without care and such modification compromise the integrity of the evidence slightly. To deal with this problem a trained investigator is needed to have complete knowledge of behaviors of the different operating system so that stored data will not be overwritten accidentally [34].

Hashing

Hashing is a technique to provide authenticity that a file or image was not modified by identifying it quickly. Cryptographic hashing is mostly used by forensic community. Cryptographic functions such as MD5 or SHA-1 [4] are used. The forensic community uses MD5 in most tools because it was faster and produced a shorter hash review [34].

Carving

Carving is the process of scanning of disk blocks to find deleted data that do not belong to current files. Carvers are the tools used for it, combine inodes into the original files that were deleted by using known header and footer signatures [37]. Carving can only recover deleted file, but not overwritten or temporarily cached files on media. Recent advances in carving allows fragmented files to be recovered more accurately.

1.3.4 Limitation of Digital Forensics

Digital Forensics is becoming irrelevant as hard-won capabilities are in danger of being reduced or even lost as the result of advancement and fundamental changes in the computer industry [38]. Some limitations of digital forensics are as follows:

- The size of storage devices is increasing; this means that create a forensic image of devices, or process all of the data found in it is becoming frequently inefficient and highly time consuming.
- The increasing incidence of embedded flash storage and the rise in production of hardware interfaces indicates that storage devices can no longer be imaged or removed readily.
- Increase in production of OSs and file formats has also increased the need of complex data exploitation tools and hence the tool development cost.
- Now days, most of the cases require the analysis of multiple devices to gather evidence followed by the correlation of gathered evidence in various devices, instead of previous cases where analysis of a single device was sufficient.
- Pervasive encryption problem is there, that means data cannot be processed frequently, even when it can be recovered.

- Data or code cannot even be found frequently in the case of the cloud used for remote storage and processing.
- Expensive memory (RAM) forensic is needed for malwares that are not written to persistent storage.
- Legal challenges increasingly limit the scope of forensic investigations.

1.3.5 Digital evidence and its admissibility

Credible digital evidence is the data collected and preserved by a process that validate the legitimacy of that data. A number of factors are given below, which determine requisite scientific validity of digital evidence.

- Whether the concepts and techniques used by the scientific expert have been tested.
- Whether review and publication of digital evidence have been done.
- Whether the techniques used by the scientific expert have a known error rate.
- Whether they are up to standards of applications governed by them.
- Whether the concepts and techniques used by the scientific expert are widely accepted.

If digital evidence survives the above challenges, it may still have to get through several difficulties regarding the collection, processing, storage and presentation of the digital evidence. Digital evidence requires a proper foundation for introduction. Generally, digital evidence can be introduced as “party is offering the sufficient computer information in order to certify a finding that such information is truthful and the same opportunity is given to the opposite party to inquire the accuracy of the computer and input procedures used to collect information to validate the accuracy of on paper business records” [39].

1.4 Combination of IDS and Digital Forensics

Intrusion Detection System (IDS) and Digital Forensics (DF) are two broad areas in the computer world. Both of the above are used in computer and network security to protect and secure the system, network, or organizations. First one is used to detect the intrusion or attack in order to prevent the damage in the given system or network, whereas later one is used to analyze the attacked or compromised system in order to find the amount of damages occurred, cause of the attack and strategies used in the attack. Both intrusion detection system and digital forensics have an application in one another.

Digital forensics is a process of collecting, analyzing, and presenting digital evidences about the attack and intrusion detection system can be used to produce the evidences that can be used as evidence. Intrusion detection system is one of the most important and popular sources of digital evidence as it enables the administrator to collect the evidence about the attack by providing the information about the attack. The evidences collected with the help of intrusion detection system are the real time and practical one, as they are collected at the same time when an attack is detected. Such evidences contain the most of the important information about the attack such as network connection, running processes, open files, system calls, the amount of data flowing, memory use, etc. The evidences produced by intrusion detection systems directly are the documentary evidences and can be used in legal proceedings with a testimony of the person responsible to generate such evidence.

On the other hand digital forensic techniques can be used in intrusion detection process. This process is known as network forensics, which deals with volatile and dynamic information. Network forensics generally have two uses. The first related to security, in which network is monitored for anomalous traffic and intrusions are identified. The second use of network forensics is related to law enforcement. Network data is more unpredictable and volatile as compared to computer forensics, where data containing evidence is generally preserved on disk.

Chapter 2

Literature review

Brian Cusack and Muteb Alqahtani [40] have proposed “*Acquisition of evidence from Network Intrusion detection systems*”. This work evaluates the performance of NIDS in wired networks under different workloads to find the evidential value of the NIDS. The work shows that the performance of a network intrusion detection system (NIDS) varies very much for different workloads. Performance evaluation is done on the basis of few points given below:

- Number of packets lost due to high workload?
- Event data retention, how long event data can be retained.
- Complexity, NIDS can handle.
- The cost associated with the system.
- Feasibility of the evidence (generated by NIDS) in legal proceedings.

First four points relies on stress testing, whereas last point relies on empirical testing. A testbed has designed with numerous tests to check where NIDS peromed well in collecting digital evidence and where they need improvements. Work has tested three different NIDS Snort, Suricata, and Bro-IDS for two different types of attack named SQL Injection and Cross site scripting. Six different data rates (0.4 Mbps,

10 Mbps, 30 Mbps, 60 Mbps, 80 Mbps, and 100 Mbps) were used in the experiment. The results were the alarms on intrusion.

This work found that detection capability of NIDS decreases quickly on increasing workload. Also, when attack's complexity increased the detection capability of NIDS fell more quickly. Number of packet loss of NIDS increased with the workload. The greatest packet loss was found in snort up to 80%, whereas up to 10% in the Bro-IDS and Suricata. NIDSs show many variations in complexity test. Only Bro-IDS detects all cross scripting attacks at 10 Mbps and no NIDS could detect any cross scripting attack at 100 Mbps. Suricata also shows better performance in this test. Cost of CPU utilization is more in case of Suricata and Bro-IDS as compared to Snort. All the evidence generated by NIDSs are tested under Daubert criteria and accepted in the IT security community.

The main question asked in this work: What is the evidential value of NIDS? The answer can be given in two parts. First is that the evidence from NIDS are better prepared for forensics post event fashion and to prevent the intrusion occurrence as compared to the evidences from system log files. However NIDS failed to give the required performance when heavy workload and complex attacks are applied. The conclusion of the work is that the NIDS have weak evidential value for both system improvement and legal admissibility.

Jorge Herrerias and Roberto Gomez [41] have proposed “*A log correlation model to support the evidence search process in a forensic investigation*” to correlate the logs coming from diverse log files of various applications. Log files are the primary wellspring of information for forensic examination process in many systems. The information provided in the log files is used to find the information about the attack and such information can be used as evidence. Log files are written by some application (such as intrusion detection system) and contain the information about system activities. When a huge amount of logs is recorded in log files from one or more applications then analysis of such massive logs is very complicated task for system forensic investigators. Most of the logs in such huge log file collection are

not related to attack, that's why there are more burdens on the investigator to find the relevant logs from this big collection, which is very complicated. To overcome this problem, this work proposed a model to correlate the logs coming from various log files to assist the forensic investigator in evidence searching process.

The model proposed in this work contains a set of modules for collection, filtration, normalization and correlation of the logs from numerous log files. First module collects logs from various files and store in a container. After storing the log files at the same place events are filtered in order to reduce the number of events. After filtering, events are normalized to bring all the events in the same common format as logs from various applications may have different formats. Finally, the normalized events are used to reassemble the actions, though event correlation. In correlation, relationships are assigned between many related events. In the correlation process events are linked to recreate the attack sequence. By correlation process, attacker's actions can be identified and hence attackers can be traced. This work used three types of correlation approaches, rule based, pre-condition, and post-condition. In rule based correlation, scenarios of known attack are applied to detect a specified attack. Another approach correlates alerts, if the pre-conditions of a later alerts are fulfilled by the post-conditions of some prior alerts. A specialized language LAMBDA is used to define these pre and post conditions. By using this model now forensic investigator can recreate the attack scenario, and only needs to focus on the effective part of the system instead of looking in all the systems.

This work has an advantage that forensic investigator can search the evidence efficiently without doing a lot of searches by using the proposed model. The forensic analysis of a large network can be done easily by using this model. However this paper doesn't provide the log translation process, used in normalization. Also specialized language LAMBDA need an expertise to define pre and post condition properly.

Ali Reza Arasteh, Mourad Debbabi, Assaad Sakha, and Mohamed Saleh [42] have proposed "*Analyzing multiple logs for forensic evidence*". This work proposed

a model for formalization of forensic analysis of logs collected from multiple applications. Proposed model intended for proper automatic analysis of multiple logs in order to provide fast and efficient forensic analysis. The model uses two approaches, first approach is for representation of log events and properties logically, and the other is for proper automatic analysis of logs to search a specific event pattern. The approach used here is based on the computational logics and formal verification.

In the proposed model a tree is created from set of logs to be analyzed where the labels of the tree are the events extracted from logs where every event is a conceptual perspective of the applicable information present in the logs. Nodes of the tree brings into a common structure by expressing the events in terms of the multi-sorted term algebra, whose signature is chosen in such a way that it includes all relevant information necessary for forensic analysis. This approach is very useful when logs are collected from different system because logs from the different systems may have different syntaxes. Using this approach all logs are modeled in common structured events and then such events can be analyzed easily. Properties of this model were expressed by a specialized formal language ADM. The properties of logs are shown through the utilization of a logic containing, modal, temporal, computational and dynamic characteristics. Proposed model has tested for the denial of service attack (SYN attack). They have used the logs generated by the snort and find the SYN attack entry in the log file and then analyze by using the proposed model and successfully reconstruct the attack.

The model proposed in the work is very useful in automatic analysis and verification of the multiple logs. Work enables the forensic investigator to efficiently and quickly analyze the logs in order to find the evidence about the attack or reconstruct the attack. Work can be improved by providing security to log files as log files can be subject to change.

Ya-Ting Fan and Shih-Jeng Wang [43] have proposed “*Intrusion Investigations with Data-hiding for Computer Log-file Forensics*” to protect the log files from

alteration. This work proposed the concept of steganography to forensic analysis of logs for preservation of reliability and completeness of the digital evidence. As common security mechanisms, like an intrusion detection system back up the logs, but doesn't provide the security to the logs. An attacker who intrudes into the system can also modify or destroy the logs and making it impossible to preserve the evidence about the intrusion. This proposed technique uses steganography to secure the logs from modification or deletion.

Generally logs generated in organizations are transferred to the secured centralized log server for better security, but intruders can damage the logs during the transfer from systems to log server. To avoid this, work proposed a model in which instead of sending the log files to centralized server every working system itself keeps its log files. To preserve the log files in individual computers, log file's text is hidden into an image by using steganography. The LSB hidden technique was used in this work to hide the log files into the image. This image is now stored in the individual computer, but attacker will not be able to find the original file in the system so the log files can safely remain in the computers. After intrusion when an intruder tries to alter the log file records, he will search for the IP where logs are stored (i.e. Trace logs). In the proposed model a copy of the hidden file is sent to trace log and allow an intruder to alter or delete the log file. After completion of intruder's action incompleteness will lead in log files due to alteration by an intruder. Altered log file is compared with the hidden secured log file and differences are found, such differences are probably the evidences about the attack.

The proposed model consists of four modules generate log, log backup, log view, and deliver log. Generate log hides information into 512x512 gray scale image where every 8 image bit hide 3 information bits. Log backup, back-up the log files from computers to protect them. Log view displays the hidden files. Deliver log send the log file to log trace, send alerts to the administrator on alteration, and compare the altered file with original one to find the evidence.

The model proposed in the work is very useful because it preserves the completeness

and reliability of the evidence by protecting the log files from modification. The model also reduce the uses of the network resources as it stores the log files in the computer itself instead of transferring to log server hence reduce the transfer cost. The model can be used for real time detection. Although work has given a good model to preserve the evidence, but it has some limitation such as if log files are stored in individual computer then memory in every individual computer should be large. This work has also not given the detailed procedure of steganography technique to hide the information in the image.

Fahmid Imtiaz [44] has studied the “*Intrusion Detection System Logs as Evidence and Legal Aspects*”. This study focuses on the potential of using an intrusion detection system generated logs in legal as evidence in legal proceedings. As IDS can be used to detect the attack by tracking the trails left by an intruder during the reconnaissance. In reconnaissance step attack tries to probe and gather the information about the available ports or services on the network in order to launch the attack. However, government laws many times act as a hindrance in observing the private network due to some conflicting issues discussed in the Commonwealth Telecommunication Interception Act. Government legislators don't consider the intrusion detection system as a security tool and IDS logs as valid evidence. This study discusses the legislation rules in Australia, the United States, and the United Kingdom for using intrusion detection system logs as evidence.

In this study dimensions for conducting forensic investigation process from the intrusion detection system logs are given. According to which computer security community (i.e. IDS) lacks in awareness of two well-known principle of forensic process, principles of continuity and the principles of audibility. The study also focuses on the some tests such as admissibility, validity, and weight, that an evidence should pass to be used in legal proceedings. One more issue is discussed in the study is ‘acontextual’ presentation of log entries which may lead to misinterpretation of significance of log file entries or whole log file. Notable strengths and limitations of IDS as evidence are also given in the study. Strengths are, IDS logs are directly

potential documentary evidence, IDS performs real time analysis, and provides fault tolerance to the secured organizations. One limitation is that IDS are also susceptible to attacks and IDS logs can also be modified and hence completeness and reliability of log evidences is not trustworthy. Another limitation is the capability of IDS to survive with huge amount and high speed data. The study also discussed about the challenges such as false positive, false negative, handling encrypted data, OS specific application protocols, and periodically policy or rules update in IDS.

Finally study comes at the conclusion that the legislation limits the use of IDS logs as forensic evidence and the legislations need to change in order to use the logs as evidence. The legal aspect and value of using IDS logs as evidence will change if the legislature permits the companies to intercept communications legally.

Leonard Kwan, Pradeep Ray and Greg Stephens [45] have proposed “*Towards a Methodology for Profiling Cyber Criminals*”. This work proposed an approach using honeynets to collect legally valid evidences from cybercrime. Due to lack of effective methodologies of collecting evidence and prosecute the attacker, many cybercrimes go unpunished. Proposed approach addresses this problem by using generation-III honeynet. This study predominantly focuses on the data analysis phase of the digital forensic model with the notion of criminal profiling, cybercrime profiling and cybercrime-cybercriminal correlation.

Criminal profiling is an investigative and behavioral tool that is proposed to help forensic investigators to predict and profile the characteristics of unknown criminal subjects accurately. Two types of criminal profiling are discussed in the study. This work uses behavioral evidence analysis (BEA) profiling technique. Cybercrime profiling is discussed in four dimensions breadth, depth, vulnerabilities, and tools. Breadth is a metric used to measure the range or scope of the intrusion. Depth is same as breadth, but it focuses measure the degree of intrusion of the network intrusion. Vulnerabilities are the flaws and weaknesses in a system that allows an attacker to compromise it. Tools are simply the software that is utilized to exploit vulnerabilities. Proposed approach gives a conceptualization of a cybercriminal as a

container for holding cybercrimes. Cybercriminal-cybercrime correlation is a method to associate cybercrimes with their perpetrator to add precision and detail to the cybercriminal's profile. Each cybercrime profiles from cybercriminal profiles can be compared with a test cybercrime profile and similarity index can be calculated. Similarity index lies between 0 to 1, with 0 as greater difference and 1 is no difference. Snort IDS is for cybercrime profiles and criminal profiling is simply done in relation databases.

The given approach is very useful to correlate the detected cybercrime with the cybercriminals that has already profiled in the database. This correlation helps the investigator or security person to take action on the criminals. However approach proposed in the work is very time and resource consuming as the process repeats for each entry of a criminal profile to compare a single crime. A knowledge-base or expert system is needed to correlate the profiles with an attack signature, which need both cost and efforts.

Collie and Byron [46] have studied "*Intrusion Investigation and Post-Intrusion Computer Forensic Analysis*". This paper gives an overview of the investigation and handling of security incidents, from the perspective of a legal administrative investigator. This work proposes a prologue to the investigation of intrusions and likewise portrays various post-intrusion computer forensics methodology and tools. This includes incident response, intrusion investigation, real-time intrusion investigation, post-intrusion forensics, and intrusion reconstruction. In incident response network security framework requires successful identification of attackers, as well as suitable response to them to with. Incident response activities may comprise access denial to an attacker, reporting of incident to the appropriate person or team, restraining the activities of an attacker, proceeding operation to accumulate extra information and restoring the influenced system. Next is intrusion investigation, which includes efficient accumulation of information produced by application, system, network, and client activities that is vital in intrusion detection, leading investigations to preserve evidence so that it can be used in court. Real-time

intrusion investigation reveals the intent of intruder, which may help with assessing and restricting the dispersal of the bargained information. Post-Intrusion Computer Forensics is a process of forensic analysis when evidences are collected. It comprises various methods, the first one is chronological event and evidence registers which documents the attacker's suspected actions and the site's responses enumerating what is suspected, what the site did and who was reached accordingly. Second is frosty the Scene in which the compromised system is turn off and restart from backups. Last is intrusion reconstruction, which involves correlation and analysis of all recovered data regarding the intrusion.

This paper is useful because it describes the procedure of intrusion detection and then use of the detected event in forensic analysis, provides the different methods of intrusion detection such as log file analysis, File integrity assessment, intruder artifacts, network transaction auditing which helps to system administrator in the complete investigation process. The limitation of this work is that it fails to give the algorithms and tools needed for effective handling of an intrusion. Details about tools and algorithms for doing intrusion investigation and implementation should be given in order to improve it.

Stephenson and Peter [47] have studied “*The application of intrusion detection systems in a forensic environment*”. This paper describes the framework to explore the applicability of IDS in process of collection and management of digital. This work reviews the performance and forensic appropriateness of numerous types of IDSs. During the research on ‘application of IDS in digital forensics’ two points of view have emerged. One viewpoint says that collection and preservation of forensic evidence in the case of a computer and network security incident to be unsuitable for an IDS. Another viewpoint says that an IDS is the most possible candidate for collecting original evidentiary data for forensic analysis in real or near real time. Use of IDS with digital forensics begs a number of questions and these questions are discussed in this paper are:

- Can IDS perform effectively if it also has to manage evidential data properly

to meet legal standards?

- What is required for automatic management of data from an evidential perspective?
- What features need to be added to IDS to ensure that it can perform as an IDS as well as it can manage evidence properly?

This work proposed an approach consists of three phases. First is the Current state in which detailed research and cataloging of prior formal work in forensics and intrusion detection is done and it also governs the general impact of management of forensic evidence on an IDS. The second phase is Theoretical model in which analysis and modeling of forensic process and IDS is done, a combined theoretical model for intrusion detection in forensic environment is created, and an appropriate architecture is designed using SABSA method. The final phase is laboratory demonstration in which the test bed and combined model generated in previous phases are used to measure the impact of forensic analysis on the optimized IDS.

This study is very useful to understand the use of IDS in forensics because this paper gives the detailed description about how an IDS is applicable in digital forensics also discusses on the issues that arise when use IDS for collecting the evidences about the intrusion and the legacy of the evidences generated by IDS. This study provides a theoretical model for using an intrusion detection system in computer forensics. It has some limitation such as this study only provides the theoretical model, but fails to the detail about the implementation of that and it also doesn't give the criteria about which type of evidence generated by IDS are legal for investigation. This can be improved by describing the tools and/or algorithm for implementing the model.

Kymie, Thompson, and Ruighaver [48] have studied "*Intrusion detection systems and a view to its forensic applications*". This study gives a view of forensic application within the Intrusion Detection framework, and anomaly based IDS was used for most of the work. As indicated by this paper machine security could be divided into the methods that endeavour to anticipate security breaches, and

those that endeavour to identify security violations. Both areas are extremely essential in the model of system security. The ultimate goal of forensic application is to get sufficient evidence to catch the criminal. In a computer system, cover of secrecy energize offenders by making ruinous conduct easy while making it amazingly troublesome for law authorities to demonstrate the identity of the criminal. Accordingly the ability to get a fingerprint of system users and their common conduct is definitive with a specific end goal to get some hang on recognizing the criminal.

The information available in log files can be used as evidence. In any case, often at a higher level, it is desired to have a more in-depth ability to restrict the field or even generate a list of conceivable suspects. At this level, it is not sufficient to have logging activities to find evidential data, but also some artificially intelligence techniques are required to assemble and make profiles of system users. This paper demonstrates that it is conceivable to accomplish effective results by utilizing simple variations of the feed forward network with available information of the user's behaviour. Thus, numerous customary issues in-built to Intrusion Detection and to neural networks have been overcome.

This work suggests an important application of IDS in digital forensics due to some reasons such as evidence is collected from the system log file which is comparatively easy. Work also gives an idea to use complex neural networks that will also be able to identify the perpetrator. An automated anomalous user behavior detection system will help to relieve a burden (sometimes unrealistic) from systems administrators. Automated systems are helpful during a violation as well as turn to be a very helpful forensic tool. At the same time it has some limitations such as evidence collected from the log files can only be used to learn the behavior of user, but not able to identify the intruder and paper doesn't have any detail about the neural networks used. Instead of using the log files for evidence collection system image (memory image and running processes) can be used to improve the system.

Sommer [49] has studied "*Intrusion detection systems as evidence*". This paper

discussed that how IDS could be utilized as evidence in criminal and civil lawful processes. This study explains two determinants for acceptance of evidence in legal process and determinants are weight and admissibility. Discussion about the issues of courts in managing novel scientific evidence and the difference in the legal and scientific proof is given. Evaluation criteria of IDSs as wellsprings of legal evidence are given, including protection and continuity of evidence and the transparency and trust of forensic technique. According to the procedure, known as adversarial, Evidence needs to pass two tests: first is admissibility (it must follow certain legitimate rules which are utilized by a judge) and second is weight (it must be justifiable and must be an attractive persuading to the court, it may be a jury or a judge).

Numerous sorts of evidences that may be exhibited in court is real, documentary, testimonial derived and expert are discussed in the paper. The evidence produced by an IDS are the logs which come under the category of documentary evidence, such evidence requires the testimony of responsible person for setting up the system and for gathering the evidential logs which are brought before a court as evidence. The evidence must have certain attributes such as Authenticity, Accurateness and Completeness. Some bullet points are given about the IDSs as sources of Evidence for Digital forensics:

- The quality of IDS relies on upon the degree to which it gives convenient and precise data.
- Evidence acquisition is a different yet related activity.
- If logs are generated by IDS then a prosecutor must be ready with complete details about the tool, its configuration and working.
- Logging evidence or anything produced by computer will need to formally prepare the testimonial by the individuals personally included, to the court.
- Logging evidence should always be the best that is immediate from the system

where it was found.

IDS can help this prosecution just cover the initial two points or in the greater part of the cases just second point.

This work is very useful because this gives the evaluation criteria of IDSs as sources of legal evidence, discussed about the legal acceptability of evidence by providing two determinants (admissibility and weight) of legacy and clearly describes, types of evidences are acceptable in court and how IDS fits in this. However, this research fails to give an explanation about the collection of such evidence those are accepted by the courts and also, does not gives the solution for generating reports of the legal evidences from the raw data captured by IDS. This can be improved by giving the description about the collection of acceptable evidences and the report generation and appropriate methods or tools for doing so.

Barhate and Jaidhar [50] have proposed “*Automated digital forensic technique with intrusion detection systems*”. In this paper, automated Digital Forensic Technique with Intrusion Detection System is proposed according to which technique once IDS detects an intrusion, it sends an alert message to administrator followed by invoke the digital forensic tool to capture the state of the system. The captured image can be presented as evidence in the court to prove the damage. IDS is used to detect the intrusion But IDS function is limited to detection as well as response. The IDS is unable to capture the state of the system when an intrusion is detected. Hence, it fails to preserve the evidences in original form against the attack. New security strategy is very much needed to preserve the reliability and completeness of evidence for later investigation. The limitations of previous log file approaches are that it is unable to capture the evidence of the attack. So it is not possible to preserve the log file damage for forensic analysis and evidence cannot be collected immediately against the attack to prove in the court of law. To overcome this limitation, in this work automated Digital Forensic Technique with Intrusion Detection System was proposed. Digital forensics play an important role by giving experimentally demonstrated techniques to collect, process, translate and use digital evidence to

bring a definitive depiction of the attack.

In this technique IDS is deployed on the target host which has crucial data. IDS continuously monitor incoming traffic and analyze it to classify whether it is malicious or not. If traffic is not malicious then IDS allows it and let the activity to take place, but if the traffic is malicious then it generates alert message to system administrator as well as the log server (a system where a periodic back up of log files are stored) and at the same time IDS activate the digital forensic tool to capture the current state (RAM and log file image) of the system. Then this image is compared with previously stored image and if both are same then it ensures that an intrusion has taken place. Hence the captured image is analyzed with to find the evidence against the attack and corresponding report is generated.

This work is very useful to understand how IDS is used to collect the evidence about the attack because it provides the process of digital forensics report generation with the help of Intrusion detection system. This work provides the complete flowchart for generating report of the system when an intrusion takes place and the process of report generation is automatic. As the report is generated by analyzing the current system image (memory, processes) so it gives the better then log file analysis. However work doesn't give the way to use the report as evidence in the digital forensic investigation. This work uses signature based IDS so it is only able to detect the known attacks, but not novel one.

Challenges

- **Evidence preservation:** Intrusion Detection System function is limited to detection as well as response but IDS is unable to preserve the evidences against the attack in original form.
- **Integrity:** Current IDSs are not able to protect the integrity of the information which is needed for finding digital evidence.

- **Completeness:** It is difficult to maintain the completeness of an evidence as computer files (log files) can be altered.
- **Novel attack detection:** Model with signature based IDS is only able to detect known attacks but not new attacks.

Chapter 3

Proposed Model for Automatic Evidence Collection

Proposed model named “Application of IDS in automatic evidence collection using digital forensics” can be used to automatically collect the information about the attack detected by anomaly based intrusion detection system whenever an attack occurs and then analyze the collected information to get evidence. The fundamental work of an IDS is to detect the attack and respond to the system administrator. This limits IDS to preserve the detailed information about the attack and hence about the attacker. IDS alone is unable to give any information about the techniques and tools use in the attack by the attacker and also not able to trace the attacker. The intrusion detection system is also not able to give any information about what and how much amount of damage to the system is done by particular attack. To overcome this problem we have proposed a model in which digital forensic tool is used along with intrusion detection system. This digital forensic tool will capture the memory image of the system in order to preserve the information about the intrusion. Proposed model gives the way to automatically trigger the digital forensic tool whenever the intrusion detection system alerts an intrusion. Two models are given below first by using signature based intrusion detection system and other is by

using anomaly based intrusion detection system. In the following section we have discussed system model, flowchart, tools used, and type of attack used for both the models.

3.1 Model with Signature based IDS

3.1.1 System model and Assumptions

System model with signature based IDS is shown in Figure 3.1 [50]. The function of each component is described below. Target host is a system that is to be protected from the attacker or intruder. This system consist of valuable assets such as data files, credential information, databases, system files, logs and many more important things that have to be protected from the attackers.

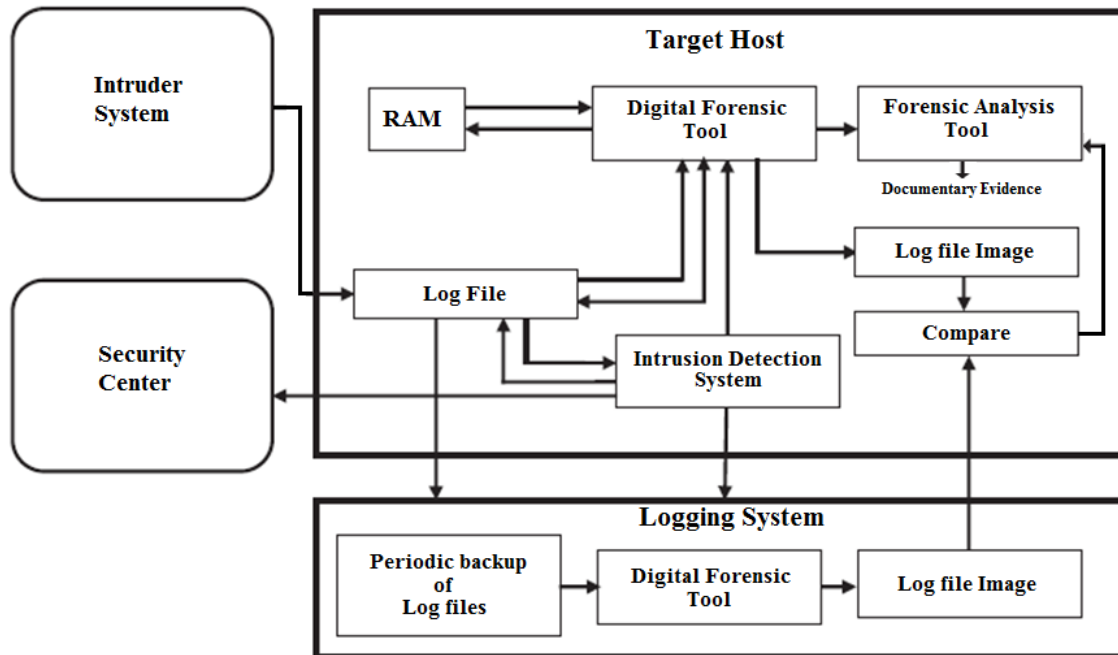


Figure 3.1: System model with Signature Based Intrusion Detection System.

Target host: This is the system that has to be protected from the attacks as it has many valuable assets. Target host consists intrusion detection system, digital forensic tool (memory acquisition tool), and forensic analysis tool (memory analysis tool). This system runs under the supervision of the system administrator.

Intruder system: This is the system from where the intruder launches the attack on the target host by sending the malicious packets. The intruder system consists techniques and tools required to launch the attack.

Security center: This is the system from where the security administrator monitors the target host. On intrusion occurrence an alert is sent to the security center to notify administrator about the intrusion. An alert can be sent in various ways, such as audio message, email, and message on console.

Intrusion detection system: To protect the target host some security mechanism is needed that monitor the system continuously and detect the intrusion whenever occurs. In this work IDS is used to secure the system, which inform the security administrator about the attack and automatically activate the digital forensic tool to capture the memory image of the system.

Signature based intrusion detection system is used in this model. Signature based intrusion detection system is based on signature matching. Signature (or rules) of known attacks written in the configuration file of the intrusion detection system are matched with the new incidents to classify new event as an attack or normal traffic. Signature based intrusion detection system has three main component, packet capture unit, preprocessor unit, detection engine. The detection engine reads the rules for known attack from rule file (or configuration file) using appropriate plugins.

Logging system: Logging system the system used to store a copy of the log files of the target host. Back up of target hosts log files is taken periodically (i.e. 12

hours, 3 days, 2 weeks), and stored on this system to protect them from intrusion.

Digital forensic tool: Digital forensic tool is a framework to collect efficient and forensically sound data from the attacked system. Digital forensic tools may be of many types such as memory acquisition tool, disk capture tool. In our proposed model we have used memory acquisition tool to capture the physical memory image of the target host whenever an attack is detected on the system.

Forensic analysis tool: Forensic analysis tool is a set of utilities used to analyze the information collected from the digital forensic tool, in order to find the digital evidence. Forensic analysis tool generate the evidence from the collected information which gives much information about the attack in an understandable format, and such information is acceptable by the court in legal proceedings. There are numerous types of forensic analysis tools such as, memory analysis tool, disk analysis tool, email analysis tool, file and data analysis tool, registry analysis tool, application analysis tool, etc. In our proposed model we have used a memory analysis tool which will analyze the previously captured image.

3.1.2 Methodology

The methodology of the system model with signature based intrusion detection system is given in following steps and pictorial representation of the same in the form of flowchart is shown in Figure 3.2.

Steps

1. Signature based intrusion detection system is deployed on the target host which has to be protected from intrusions or malicious activities. Target host contains intrusion detection system, digital forensic tool, and forensic analysis tool.
2. IDS is deployed in such a way that all the traffic coming to the host must pass through IDS. IDS monitors the incoming traffic continuously to detect

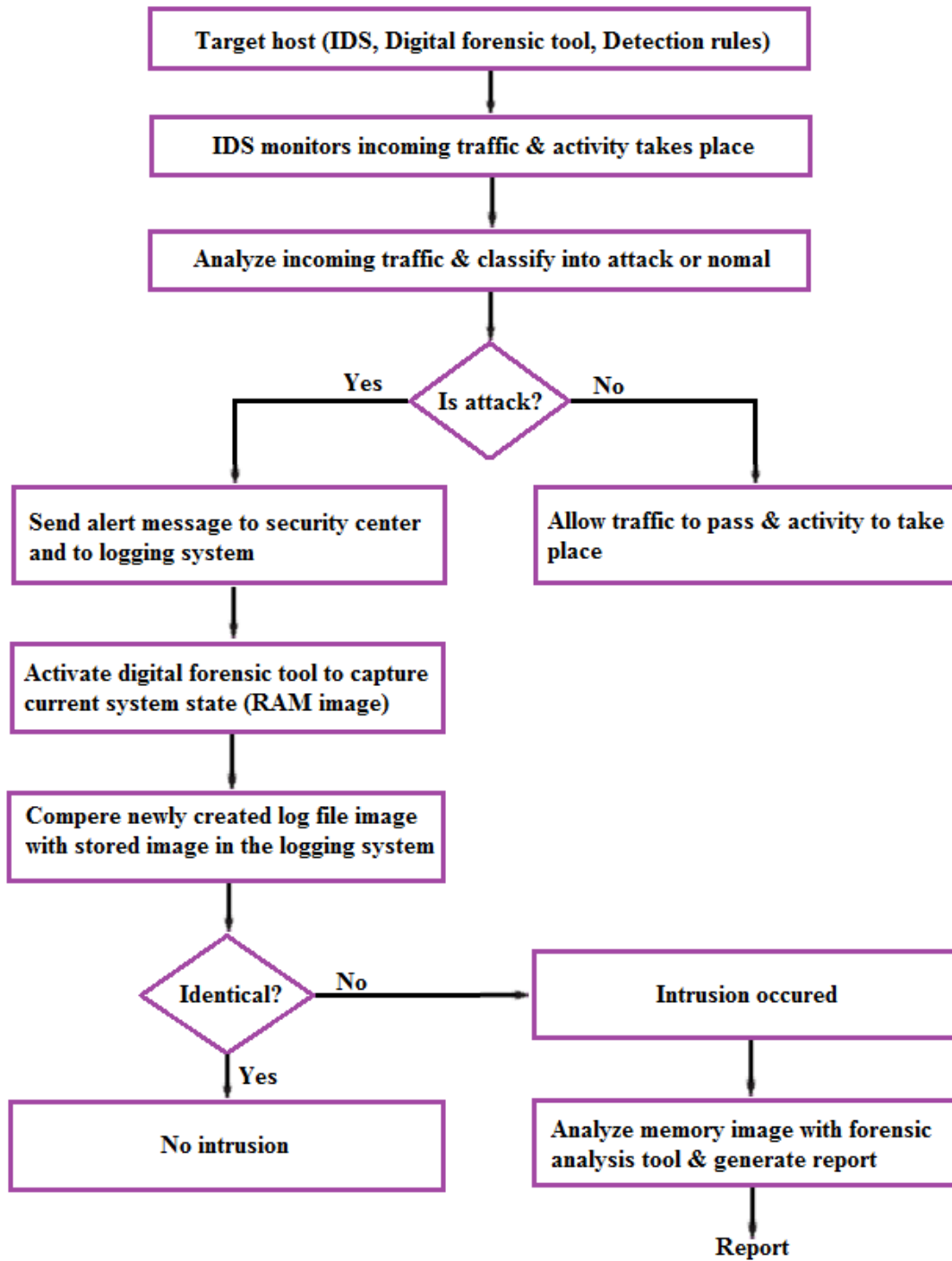


Figure 3.2: Flowchart for model with Signature based IDS.

unwanted traffic or malicious activity.

3. For every incoming packet, IDS generates and matches its signature (some specific pattern that is defined as an attack in the rule file by the system administrator) to the previously stored attack signatures to classify incoming traffic as attack or normal. If the signature of incident traffic matches with any of the stored signature, then that incoming traffic is classified as an attack.
4. If incoming traffic is classified as normal traffic then IDS simply allow it to pass and corresponding activity to take place.
5. But if incoming traffic is classified as an attack that means an intrusion is detected by IDS. Hence an alert message is sent to the security center to inform the security administrator about the intrusion and notification also send to the logging system to create an image of stored log files.
6. Simultaneously, digital forensic tool is activated to capture the current system state in order to preserve the information about the attack for future analysis. RAM (and log file) image is captured to get current system state which contains all running processes, connections established, files altered or created and much more information about the system when an attack is taking place.
7. To verify the attack newly captured log file image is compared with the image created from the stored log file image. If both are identical, then there is no intrusion means false alarm was given by IDS.
8. But if both images are not same then definitely an intrusion has occurred.
9. As intrusion has occurred, captured RAM image is now analyzed by using forensic analysis (Memory analysis) tool to get useful information from RAM image.
10. The information found from analysis can be used to create a report about the intrusion. This report can be used as documentary evidence in court in legal

proceeding. By default an HTML report is generated by the tool used here for analysis.

3.1.3 Tools

Various tools have been used in model with signature based intrusion detection system for different purposes. Name and description of the tools are as follows:

Snort: Snort is an open source, lightweight intrusion detection system that can be used to monitor network to detect a variety of suspicious traffic coming to the system. Lightweight means it can be easily deployed on any host. Snort is a rule or signature based IDS which detects network attacks by performing content matching and protocol analysis [51]. Snort architecture is divided into five modules are as follows:

- *The packet capture unit* captures data packets from various network interfaces and then transferred them to the upper layers of snort IDS. Packet capturing is based on popular packet programming libpcap to provide a high level interface to packet capture [52].
- *Packet decoder* is composed around the layers of the protocol stack introduce in the upheld data link and TCP/IP protocol definitions. It fits data packet into data structures to find the data link layer protocols, decodes IP, and TCP/UDP to find relevant information (source and destination addresses and port) from it. Detection speed of snort depends on this phase [52].
- *Preprocessing unit* modifies or arrange data packets before sending them to the detection engine by normalizing protocol headers, detecting anomalies, and reassembling packets. This unit works as a filter for identifying the things to be checked later in the detection engine (i.e. too many TCP SYN, or suspicious connection on TCP/UDP ports).

- *Detection engine*, the heart of snort IDS is responsible for detecting intrusion activities in a packet if exist. Snort employed rules for detection purpose, maintained in a 2-D linked list as rule Headers and rule Options [51]. In a snort rule, rule header includes list of common attributes (i.e. action to take, IP and port addresses, direction of flow) and rule option includes detection modifier options (i.e. pattern to match, flags, size, message to display). Rules are searched and matched recursively against all packets in both directions for the rule options which has been set in the rule file. If a packet matches any rule defined in rule file, suitable action is taken; or else packet is treated as normal data, and hence dropped by IDS [52]. Rule file and plugins are used by snort to detect the intrusion. Rule file is a text file contains sets of rules written with specific known syntax and plugins are the module used to identify patterns in rule evaluation [53].
- *Logging and alerting unit* generates log messages and alerts depends upon the rule header. Finally output module format notifications (log and alert) and generate final output to user. The user can access the output from various sources like databases, external files, syslog, WinPopup message, server message block (SMB), alert file, console, and log files [52].

DumpIt: It is a memory acquisition tool to dump physical memory of windows system in raw format. It is a combination of win32dd and win64dd in one executable to capture memory image of both 32-bit and 64-bit system [54]. It can be easily used with either user of administrator privileges by executing dumpIt.exe. DumpIt generates a raw memory image with hostname and timestamp in the same directory from where dumpIt is running.

Memgator: It is a memory analysis tool to get relevant information about the system activities from captured memory image and also generates a report containing this information. Memgator is a combination of various tools such as volatility,

AESKeyFinder, and scalple file carver [55]. It supports almost all commands of very popular memory analysis tool volatility such as, pslit to list number of processes, and connections to get all network connections established by system at that time. Memgator uses functionality of scalple carver that can carve for usernames and passwords for Yahoo, Gmail, Facebook, Hotmail, and auto fill form entries for web browser [37]. It can also extract encryption keys present in the memory file. Finally memgator generate an HTML report which contains information about most of the activities of system of investigator's interest.

3.2 Model with Anomaly based IDS

3.2.1 System model and Assumptions

System model with anomaly based IDS for proposed approach is shown in Figure 3.3. Most of the components like intruder system, target host, security center, digital forensic tool, forensic analysis are already discussed under section system model with signature based IDS. Intrusion detection system used in this model is different from previous one.

Intrusion detection system: In this model anomaly based intrusion detection system is used in which definitions of normal data behavior are stored and compared with incident to categorize the incident as an intrusion or normal.

Anomaly based Intrusion detection system has three main components, packet capture unit, Event engine, and policy layer mechanism. Intrusion detection system is deployed on the target host so that all the traffic coming to that host will first go to intrusion detection system. Intrusion detection system then decides whether the traffic is normal or suspicious. If traffic is normal then the intrusion detection system notifies the system administrator and activate digital forensic tool, otherwise allow the traffic to simply pass and corresponding activity to take place.

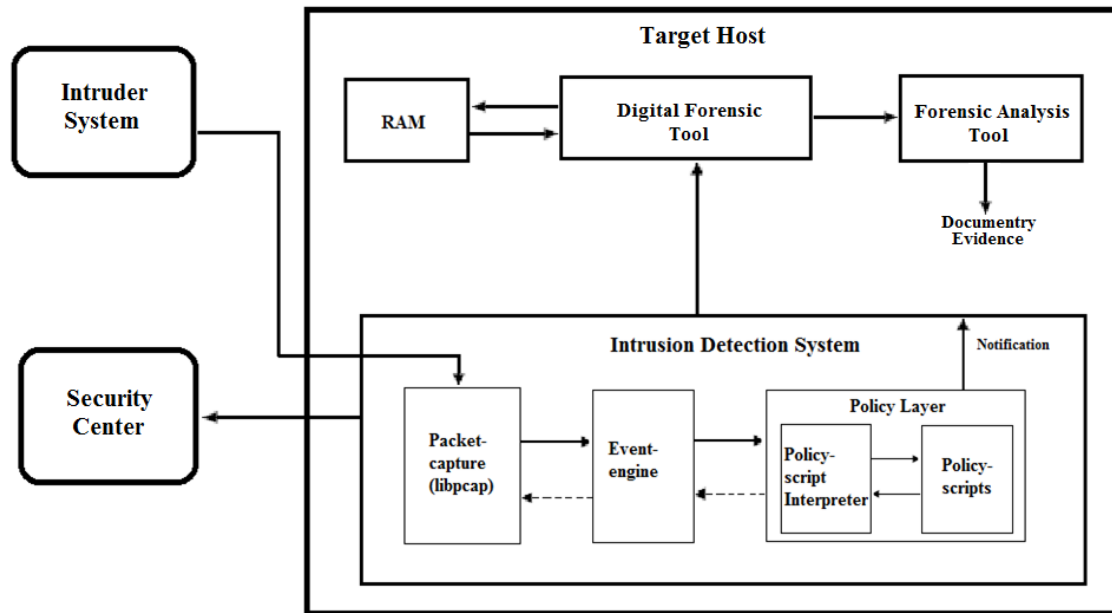


Figure 3.3: System model with Anomaly Based Intrusion Detection System.

3.2.2 Methodology

The following steps shows the working of the model with anomaly based IDS and pictorial representation of the same is shown in the flowchart in Figure 3.4.

Steps

1. Intrusion detection system is deployed on the target host monitors the network traffic continuously to protect the system from intrusions.
2. Packet capture unit catches data packets flowing through the channel and filter them based on the filter written such as a filter to allow only tcp packets, filter to allow only udp packets, filter to allow packets on or from any specific port such as port 79 for finger, 21 for ftp, and 23 for telnet.
3. Packet captured by the packet capture unit are filtered and then sent to the event engine to generate relevant events from the data streams. The packet capture unit performs low level processing of the data coming from lower

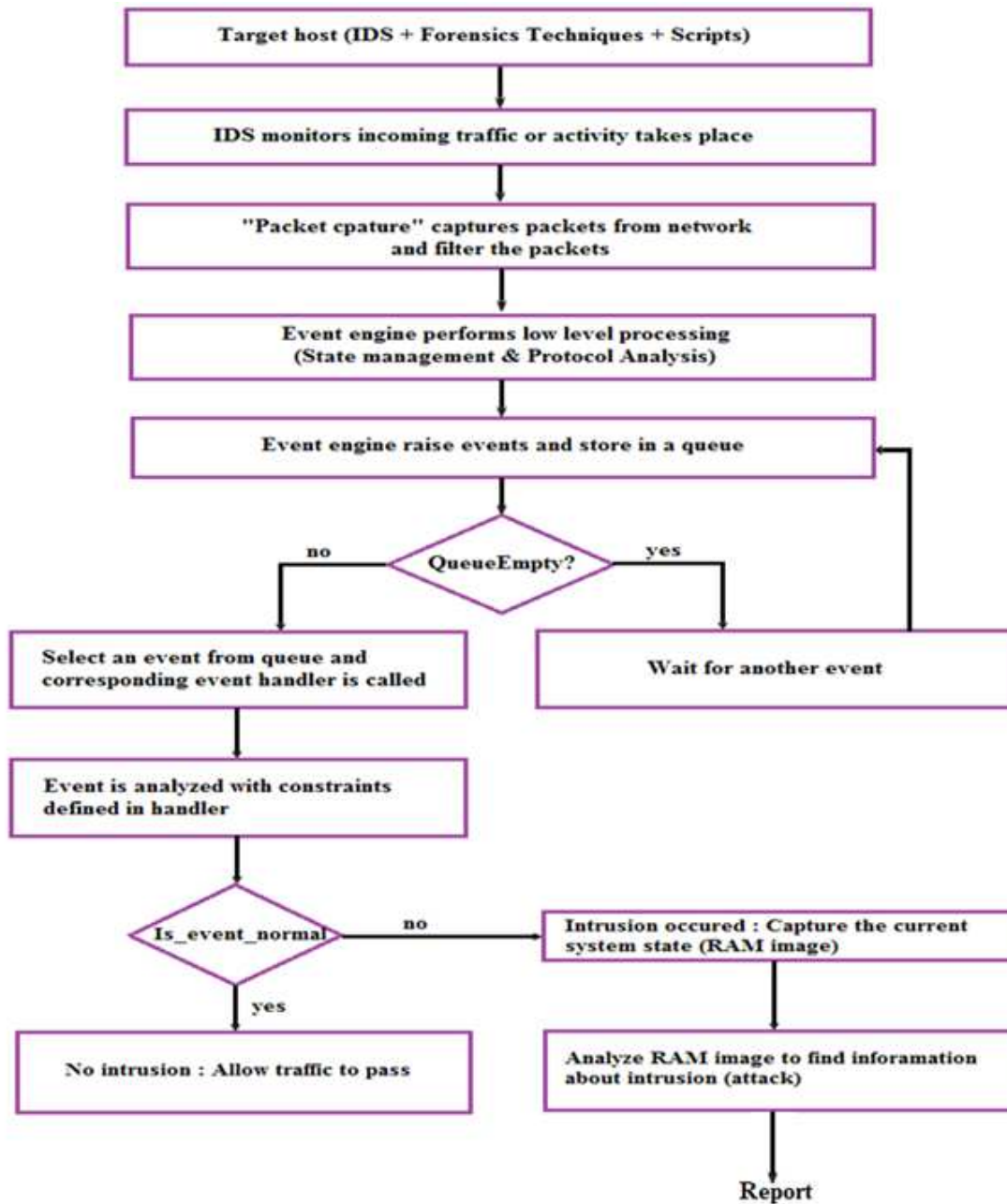


Figure 3.4: Flowchart for model with Anomaly based IDS.

layer. Low level processing involves management of connection states, various application analysis, management of timers, data structure, scripts etc.

4. After analyzing the data packets Event engine raise an event and place it in

a queue. Events are some relevant activities in the system or network such as `connection_established` is an event.

5. At the policy layer event handlers are written for almost every type of events. Every time one event from the queue is taken and the corresponding event handler from policy layer is called. This process continues until the queue becomes empty, when the queue becomes empty interpreter waits for new events to be placed in the queue.
6. Every time an event is analyzed with corresponding event handler and if it violates the normal data behavior policies, then classified as an intrusion otherwise simply allows to pass.
7. If an event is classified as an intrusion means intrusion has taken place in the system and hence alert is sent to security center and digital forensic tool is activated to dump the physical memory (RAM) of the system which will contain useful information about the attack.
8. Once the physical memory image is captured, it can be analyzed with forensic analysis tool to get the useful information such as number and name of the processes running at the time of the intrusion, connections established, files open or modified, system tables altered, etc.
9. A report can be generated from the information found in memory image analysis. This report can be used as evidence (documentary type) in a legal proceeding in order to punish the attacker.

3.2.3 Tools

Various tools have been used in our model with anomaly based intrusion detection system for different purposes. Name and description of the tools are as follows:

Bro-IDS: Bro is an open source anomaly based intrusion detection system provides high speed monitoring, real time notification, clear separation between mechanism and policy, and extensibility [56]. Bro's architecture is mainly divided into two parts one is an event engine which converts lower level data streams into higher level network events, and other is an interpreter which analyzes the network events based on the system security policies. Security policies are written and stored in the top layer of the Bro in a specialized scripting language bro. User (normally system administrator) can define their own security policy by writing a script in bro language to detect a particular attack. Clear separation between mechanism and policy is the main power of Bro, and hence it is also called policy neutral. This separation can be achieved in bro because event generation and policy definition are done independently on two different layers.

Event engine of Bro performs various low level task to generate relevant higher level network events from data streams [57]. Low level processing includes following:

- State management of network connections based on the SYN, FYN, and RST packets.
- Transport layer analyzer, analyzes packets such as TCP, UDP, and ICMP packets.
- Application layer analyzer, analyses packets, such as HTTP, SMTP, DNS, etc.
- Infrastructure which includes management of events and timer, data structure, script interpreter etc.

Some common types of network events generated by the Bro's event engine are `connection_attempt`, `connection_established`, `connection_rejected`, `connection_finished`, `http_request`, `login_success`, `login_failure`, `net_weird` [56] [57], etc.

LiME: LiME is used as a digital forensic tool in our model. LiME stands for Linux Memory Extractor. LiME is a Loadable Kernel Module (LKM) that allows

the acquisition of volatile memory from Linux and Linux-based devices [58]. LiME supports memory acquisition to the disk or on the network by specifying the port. In LiME, interaction between processes of kernel and user space is minimized during memory acquisition. Due to less interaction forensically better memory dump is produced. Multiple output formats are supported by LiME such as, raw (concatenates all system ranges), lime (each range is prepended with a fixed size header to contain address space information), and padded (pad all non-system RAM ranges with 0).

Volatility: Volatility is used as a forensic analysis tool in our model. The Volatility Framework is a set of tools designed for the extraction of digital information from memory (RAM) dump. Volatility is implemented in Python. The techniques performed for extraction are fully independent of the device or system being enquired, but give much visibility of the state of the system at runtime [59]. Volatility is very popular because of some characteristics such as open source, run on most of the operating systems, extensible and scriptable, coverage of many file formats, fast and efficient algorithms, and designed with forensic focus. Volatility contains a number of plugins for image identification, process memory, networking, processes, DLLs, kernel memory, crash dumps, file system, and registry. For example `linux_pslist` shows list of all processes running in the system and `linux_connections` shows all the connection established in the system.

Nmap: It is used on the intruder system to port scan the target host. Nmap stands for Network Mapper is an open source utility for network discovery and security auditing. Many network and system administrators also use Nmap for managing upgradation schedules of services, inventory in network, and observing service or host uptime. Nmap utilizes raw IP packets in novel ways to figure out what hosts are accessible on the network, services (application name and version) offered by those hosts, operating systems they are using, firewalls/packet filters they are using, and much more information of the host [60]. Nmap is designed to run on

all major OS.

3.2.4 Port scan attack

Port scanning is a technique to enquire the network or system for open ports and different services. The system administrator uses port scanning, for network exploration to manage the network and also to recognize the precursors to serious attacks. But for attackers, port scanning is helpful for finding information required to launch an attack successfully [61]. Port scanning is scanning done by malicious users searching for network vulnerabilities to discover the shortcomings of a host by sending probes for the port. Scanning is an important phase in launching an attack, and port scanning is the most popular one for it. Port scanning is of four types, vertical scanning, horizontal scanning, strobe scanning, and block scanning. In vertical scanning some or all ports on a single host are scanned, in horizontal scanning single port (i.e. port 80) of multiple hosts is scanned, in strobe scanning multiple ports of multiple hosts are scanned, and in block scanning all ports of multiple hosts are scanned. Port scan detection is also divided into five categories, algorithmic, soft computing based, threshold based, visual, and rule based. We have used vertical port scanning and threshold based detection approach in our work.

Chapter 4

Results and Discussions

In this chapter implementation detail of both the models are given. Scripts and codes used in bro and snort to detect the intrusion are given in this chapter. Program and shell script, used to automate the digital forensic process when intrusion is detected by IDS are also given. Tools and plugins used for memory analysis are discussed. For both the models we have run system for 4 hours each and analyzed the real time traffic to test our models. The number and types of alerts generated by IDS used in both the models are given in bar charts. Some screenshots of results for both the models are given. Finally a comparison of both models is given in the form of table and graph.

4.1 Results for Model with Signature based IDS

Snort is an open source signature based intrusion detection system. In our experimental work we have used Snort-2.6 for detecting intrusion. Snort is installed on the target host “192.168.40.17”. Snort is configured by selecting a valid interface card (*i1*) in my system.

Snort rules are written in *snort.conf* file inside etc directory of snort. Priorities of alerts are defined in *classification.conf* and allowed signature IDs are configured in *threshold.conf*. After configuration Snort is started by running following commands:

```
snort -A console -i1 -c C:\snort\etc\snort.conf -l C:\snort\log
```

Above command capture packets by using *i1* interface, analyze packet with rules written in *snort.conf* and finally send the output on console as well as write alerts in a log file in the log directory of snort.

In our work snort rule is written for detecting ICMP ping from a specific system “192.168.40.18” (Intruder system) as an intrusion. Rule to detect ICMP ping as the intrusion is written as follows:

```
alert icmp 192.168.40.18 any ->192.168.40.17 any (msg “Detected ICMP ping on my system”; sid: 1000002)
```

Priority of alerts are defined by modifying *classification.conf* file. A sample entry in *classification.conf* is as follows:

```
config classification: icmp-event,Generic ICMP event,3 Threshold.conf is configured to suppress the events which we want to ignore. Sample entry in threshold.conf is as follows:
```

```
suppress gen_id 120, sig_id 3
```

Configuration file *Snort.conf* file is configured to set the mode in which output of snort to be sent. In our work we have sent output in three modes email, console and writing in the alert file. Screenshot of alert messages on the console is shown in Figure 4.1. Same alert entry will be written in alert file *alerts.ids* file by snort. When an entry containing “Detected ICMP ping on my system” is made in *alert.ids* means intrusion has detected by snort. A program was written to continuously monitor *alert.ids* searching for a pattern “Detected ICMP ping on my system”. As soon as this pattern is matched program code will execute memory acquisition tool to capture the RAM image. This tool will dump memory in .raw format in the current directory where it is stored. To capture memory image DumpIt-1.3.2 is installed and configured on the target system. This memory acquisition tool will be executed from the program automatically when alert containing intrusion will be written in *alert.ids* file. Screenshot of DumpIt capturing memory is given in Figure 4.2.

```

Command Prompt - snort -A console -i1 -c C:\snort\etc\snort.conf -l C:\snort\log
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_DMP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Commencing packet processing (pid=1932)
12/01-14:35:56.200256 [**] [1:1000002:0] Detected ICMP ping on my system [**] [
Priority: 01 <PROTO:058> fe80:0000:0000:0000:a441:793c:bbad:156a -> ff02:0000:00
00:0000:0000:0000:0000:0016
12/01-14:35:56.209992 [**] [1:1000002:0] Detected ICMP ping on my system [**] [
Priority: 01 <PROTO:058> fe80:0000:0000:0000:a441:793c:bbad:156a -> ff02:0000:00
00:0000:0000:0000:0000:0016
12/01-14:35:56.211107 [**] [1:1000002:0] Detected ICMP ping on my system [**] [
Priority: 01 <PROTO:058> fe80:0000:0000:0000:a441:793c:bbad:156a -> ff02:0000:00
00:0000:0000:0000:0000:0016
12/01-14:35:56.219775 [**] [1:1000002:0] Detected ICMP ping on my system [**] [
Priority: 01 <PROTO:058> fe80:0000:0000:0000:a441:793c:bbad:156a -> ff02:0000:00
00:0000:0000:0000:0000:0016
12/01-14:35:56.498065 [**] [1:1000002:0] Detected ICMP ping on my system [**] [
Priority: 01 <PROTO:058> fe80:0000:0000:0000:a441:793c:bbad:156a -> ff02:0000:00
00:0000:0000:0000:0000:0016

```

Figure 4.1: Alert message on console.

```

D:\Softwares\dumplt\DumpIt\DumpIt.exe
DumpIt - v1.3.2.20110401 - One click memory memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>

Address space size:      4821352448 bytes < 4598 Mb>
Free space size:        74491273216 bytes < 71040 Mb>

* Destination = \\?\D:\_Softwares\dumplt\DumpIt\ANKIT-PC-20140504-172753.raw

--> Are you sure you want to continue? [y/n] y
+ Processing...

```

Figure 4.2: DumpIt capturing the memory image.

Once the memory image was captured and stored in the specified directory, image was analyzed by memory analysis tool Memgator for getting relevant information. Some screenshots of results are given below. Figure 4.3 and Figure 4.4 shows a selection of tools in memgator and analysis of memory image respectively. Figure 4.5 shows the home page of generated report, which contains basic details about the system.

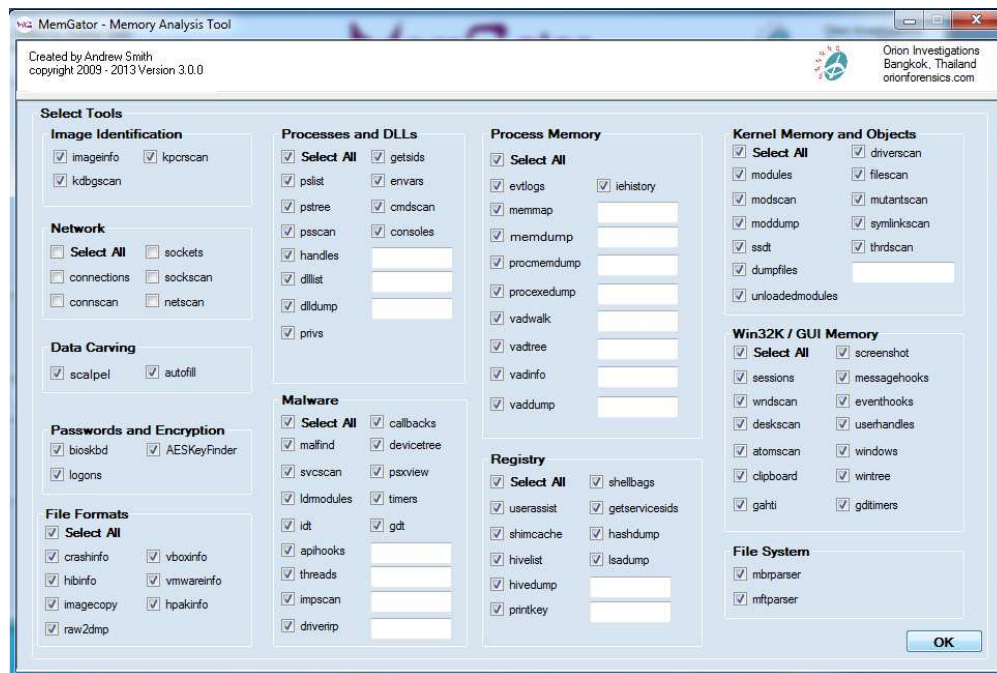


Figure 4.3: Selecting plugins for analysis.

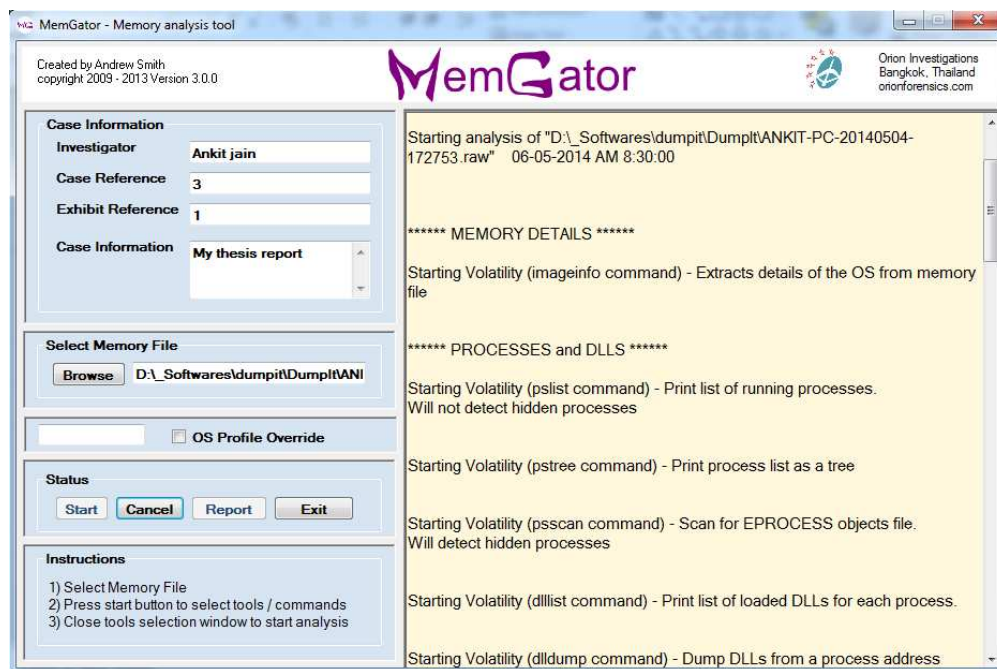


Figure 4.4: Analyzing memory image using memgator.

Orion Investigations Version 3.0.0 Andrew Smith © 2009 - 2013

MemGator Memory Analysis Report

MG

Results [Home](#)

Case Details

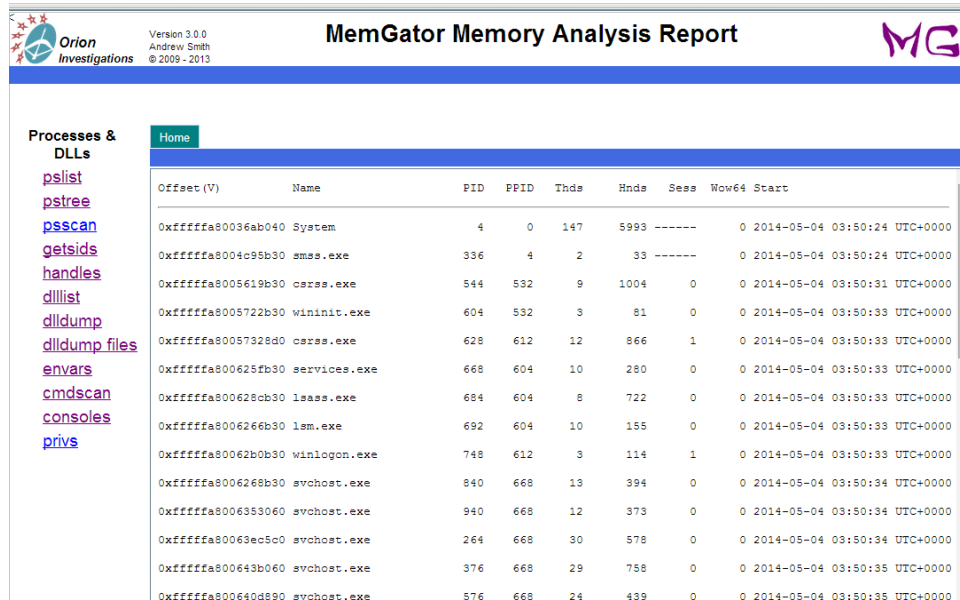
Name of Examiner: Ankit jain
Case Reference: 3
Exhibit Reference: 1
Case Information: My thesis report

Memory File Details

Determining profile based on KDBG search...
Suggested Profile(s) : Win2008R2SP0x64, Win7SP1x64, Win7SP0x64, Win2008R2SP1x64
AS Layer1 : AMD64PagedMemory (Kernel AS)
AS Layer2 : FileAddressSpace (D:_Softwares\dumpit\DumpIt\ANKIT-PC-20140504-082244.raw)
PAE type : No PAE
DTB : 0x187000L

Figure 4.5: Homepage of report.

Figure 4.6 shows active processes and Figure 4.7 shows all active TCP connections in the system at the time when intrusion was occurred. Figure 4.8 and Figure 4.9 shows all open files in the system and loaded modules respectively.



Orion Investigations Version 3.0.0 Andrew Smith © 2009 - 2013

MemGator Memory Analysis Report

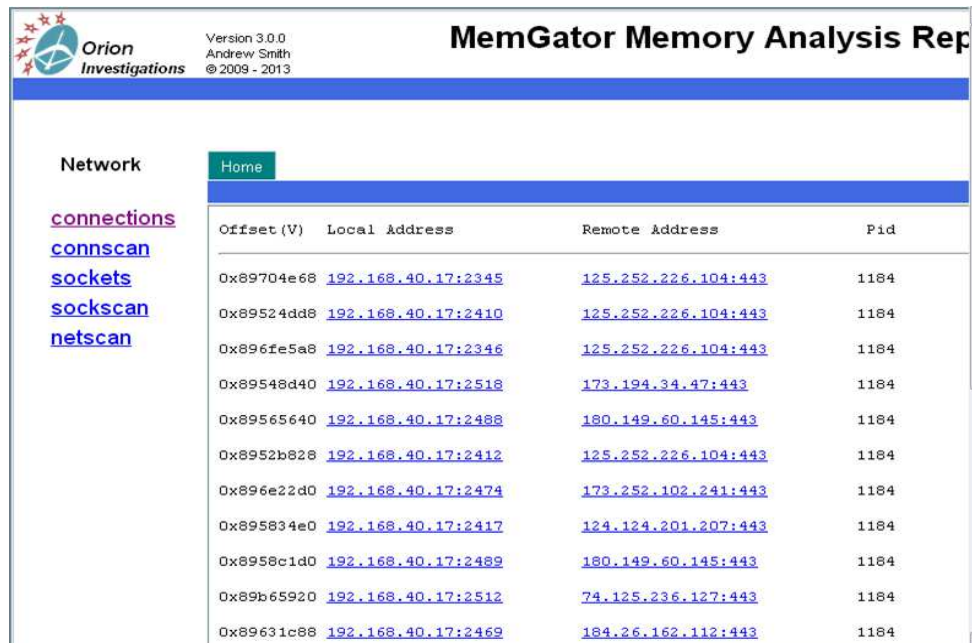
MG

Processes & DLLs

- [Home](#)
- [pslist](#)
- [pstree](#)
- [psscan](#)
- [getsids](#)
- [handles](#)
- [dlllist](#)
- [dlldump](#)
- [dlldump_files](#)
- [envars](#)
- [cmdscan](#)
- [consoles](#)
- [privs](#)

Offset (V)	Name	PID	PPID	Thds	Hnds	Sess	Wow64	Start
0xffffffff80036ab040	System	4	0	147	5993	-----	0	2014-05-04 03:50:24 UTC+0000
0xffffffff8004c95b30	smss.exe	336	4	2	33	-----	0	2014-05-04 03:50:24 UTC+0000
0xffffffff8005619b30	csrss.exe	544	532	9	1004	0	0	2014-05-04 03:50:31 UTC+0000
0xffffffff8005722b30	wininit.exe	604	532	3	81	0	0	2014-05-04 03:50:33 UTC+0000
0xffffffff80057328d0	csrss.exe	628	612	12	866	1	0	2014-05-04 03:50:33 UTC+0000
0xffffffff800625fb30	services.exe	668	604	10	280	0	0	2014-05-04 03:50:33 UTC+0000
0xffffffff800628cb30	lsass.exe	684	604	8	722	0	0	2014-05-04 03:50:33 UTC+0000
0xffffffff8006266b30	lsm.exe	692	604	10	155	0	0	2014-05-04 03:50:33 UTC+0000
0xffffffff80062b0b30	winlogon.exe	748	612	3	114	1	0	2014-05-04 03:50:33 UTC+0000
0xffffffff8006268b30	svchost.exe	840	668	13	394	0	0	2014-05-04 03:50:34 UTC+0000
0xffffffff8006353060	svchost.exe	940	668	12	373	0	0	2014-05-04 03:50:34 UTC+0000
0xffffffff80063ec5c0	svchost.exe	264	668	30	578	0	0	2014-05-04 03:50:34 UTC+0000
0xffffffff800643b060	svchost.exe	376	668	29	758	0	0	2014-05-04 03:50:35 UTC+0000
0xffffffff800640d890	svchost.exe	576	668	24	439	0	0	2014-05-04 03:50:35 UTC+0000

Figure 4.6: List of active processes.



Orion Investigations Version 3.0.0 Andrew Smith © 2009 - 2013

MemGator Memory Analysis Report

Network

- [Home](#)
- [connections](#)
- [connscan](#)
- [sockets](#)
- [sockscan](#)
- [netscan](#)

Offset (V)	Local Address	Remote Address	Pid
0x89704e68	192.168.40.17:2345	125.252.226.104:443	1184
0x89524dd8	192.168.40.17:2410	125.252.226.104:443	1184
0x896fe5a8	192.168.40.17:2346	125.252.226.104:443	1184
0x89548d40	192.168.40.17:2518	173.194.34.47:443	1184
0x89565640	192.168.40.17:2488	180.149.60.145:443	1184
0x8952b828	192.168.40.17:2412	125.252.226.104:443	1184
0x896e22d0	192.168.40.17:2474	173.252.102.241:443	1184
0x895834e0	192.168.40.17:2417	124.124.201.207:443	1184
0x8958c1d0	192.168.40.17:2489	180.149.60.145:443	1184
0x89b65920	192.168.40.17:2512	74.125.236.127:443	1184
0x89631c88	192.168.40.17:2469	184.26.162.112:443	1184

Figure 4.7: Connections to the system.

(F)	#Ptr	#Hnd	Access	Name
0000000432f20	16	0	RWrd	\Device\HarddiskVolume2\Users\Ankit\AppData\Local\Google\Chrome\User Data\Defau
000000169b1e0	16	0	R--rw-	\Device\HarddiskVolume2\Users\Ankit\Documents\Virtual Machines\Ubuntu\Ubuntu.vm
0000001b0b070	16	0	R--rwd	\Device\HarddiskVolume2\Windows\System32\FXSRESM.dll
0000001b0b250	16	0	R--rwd	\Device\HarddiskVolume2\Windows\System32\IPHLPAPI.DLL
0000001e46070	2	0	RW-rwd	\Device\HarddiskVolume2\Directory
00000033a81f0	16	0	R--r--	\Device\HarddiskVolume2\Windows\Fonts\HTOWERT.TTF
00000033b8070	16	0	R--r--	\Device\HarddiskVolume2\Windows\winsxs\FileMaps\program_files_x86_windows_sideb
00000033ba960	16	0	R--r-d	\Device\HarddiskVolume2\Windows\Microsoft.NET\Framework64\v4.0.30319\SetupCache
00000033c0dd0	16	0	R--rwd	\Device\HarddiskVolume2\Windows\System32\cryptbase.dll
00000033c0f20	16	0	RWrd	\Device\HarddiskVolume2\Users\Ankit\AppData\Local\Google\Chrome\User Data\Defau
00000033c2a20	6	0	R--r--	\Device\HarddiskVolume2\Windows\Prefetch\AgCk_SCI.db.trx
00000033f3070	16	0	R--r--	\Device\HarddiskVolume2\Windows\winsxs\FileMaps\program_files_x86_windows_sideb
00000033f73a0	16	0	R--r--	\Device\HarddiskVolume2\Windows\winsxs\FileMaps\\${postcat_zh-tw_d74f4ee430136b
0000003432340	1	1	R--rw-	\Device\HarddiskVolume2\Windows

Figure 4.8: List of open files.

Address	Name	Pointer	Path
0xfffffa8003639c20	CLFS.SYS	0xfffff88000ce0000	0x5e000 \SystemRoot\system32\CLFS.SYS
0xfffffa8003639b30	CI.dll	0xfffff88000d3e000	0xc0000 \SystemRoot\system32\CI.dll
0xfffffa8003639a30	Wdf01000.sys	0xfffff88000eb6000	0xc2000 \SystemRoot\system32\drivers\Wdf010
0xfffffa8003639950	WDFLDR.SYS	0xfffff88000f78000	0x10000 \SystemRoot\system32\drivers\WDFLDR
0xfffffa8003639860	ACPI.sys	0xfffff88000f88000	0x57000 \SystemRoot\system32\drivers\ACPI.e
0xfffffa8003639780	WMILIB.SYS	0xfffff88000fd0000	0x9000 \SystemRoot\system32\drivers\WMILIE
0xfffffa8003639690	msisadrv.sys	0xfffff88000fe8000	0xa000 \SystemRoot\system32\drivers\msisad
0xfffffa80036395b0	pci.sys	0xfffff88000e00000	0x33000 \SystemRoot\system32\drivers\pci.sy
0xfffffa80036394c0	vdrvroot.sys	0xfffff88000e33000	0xd000 \SystemRoot\system32\drivers\vdrvrc
0xfffffa80036393d0	partmgr.sys	0xfffff88000e40000	0x15000 \SystemRoot\system32\drivers\partmg
0xfffffa80036392e0	compbatt.sys	0xfffff88000e55000	0x9000 \SystemRoot\system32\DRIVERS\compba
0xfffffa8003639200	BATTIC.SYS	0xfffff88000e5e000	0xc000 \SystemRoot\system32\DRIVERS\BATTIC.
0xfffffa8003639120	volmgr.sys	0xfffff88000e6a000	0x15000 \SystemRoot\system32\drivers\volmgr
0xfffffa8003640010	volmgrx.sys	0xfffff88000c00000	0x5c000 \SystemRoot\system32\drivers\volmgr
0xfffffa8003640f30	vmci.sys	0xfffff88000e7f000	0x19000 \SystemRoot\system32\DRIVERS\vmci.s

Figure 4.9: List of loaded modules.

By analyzing the log and alert files of snort, we have found information about the total number of alerts generated, and priority based division of alerts according to snort.

- Total alerts: 5710
- Priority-1 alerts: 1880
- Priority-2 alerts: 1510
- Priority-3 alerts: 2120

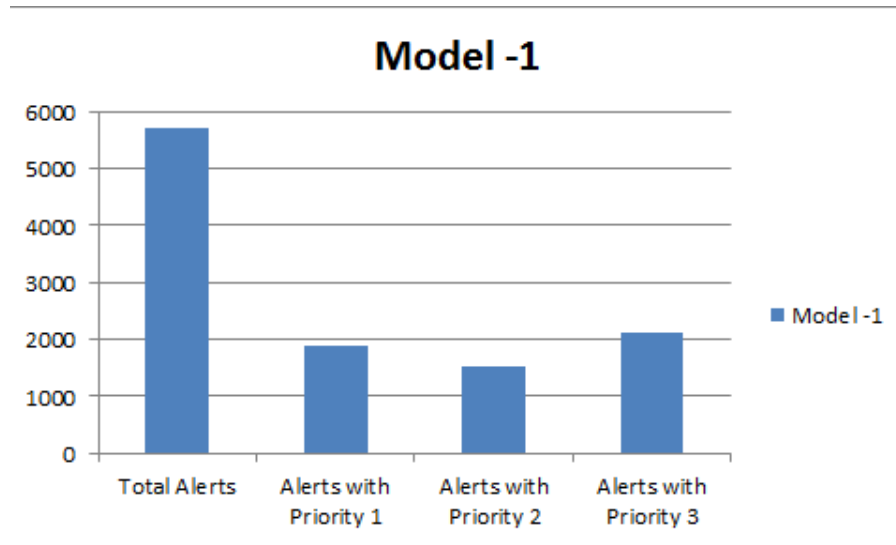


Figure 4.10: Number of alerts generated.

We have analyzed information about some popular alerts that occurred very often in Snort. ICMP ping, Bad segment, and Consecutive small segments alerts were generated by the snort more than others, 2115, 1420, and 1220 times respectively. Other alerts such as ICMP destination unreachable, http_inspects long header, and web client portable alerts were generated 224, 910, and 16 times respectively. ICMP ping alert indicates a ping request or reply from or to host. Bad Segment alert indicates an anomalous packet detection by stream5 preprocessor. An occurrence of

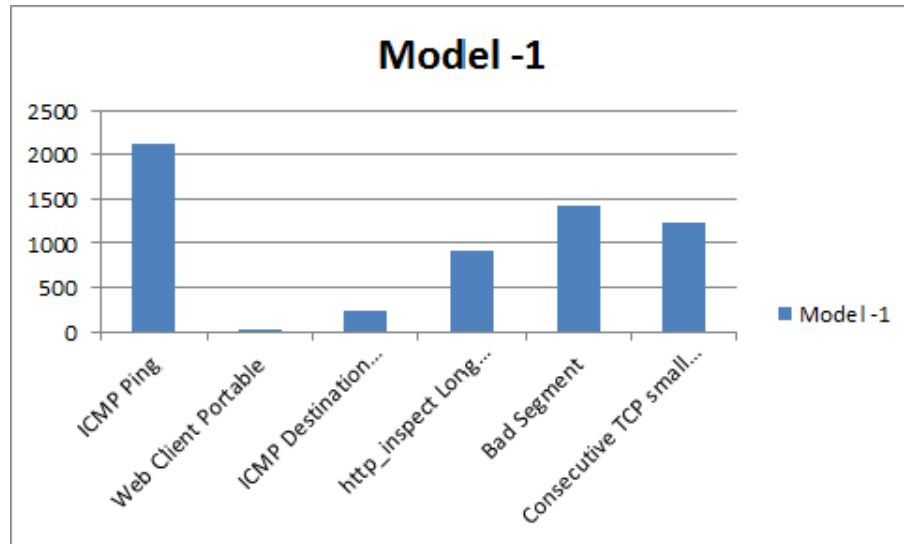


Figure 4.11: Types of alerts generated.

ICMP destination Unreachable alert means connection to a system was not available when tried to connect. Web Client Portable alert means downloading of a portable executable which may affect the system. Alert http_inspect Long Header occurs when http preprocessor detects an anomalous packet.

4.2 Results for Model with Anomaly based IDS

Bro is an open source anomaly based detection system. In our experimental work, we have used *Bro-IDS 2.2* on target host “192.168.40.18” Linux Ubuntu 13.10. Port scan from another system is considered as attack to test the proposed model. Bro is installed on the target system and configured by modifying *node.cfg*, *networks.cfg*, and *broctl.cfg* files in bro directory. After configuration, Bro is started by running following commands:

```
root@ubuntu:/# sudo broctl
Welcome to Brocontrol 1.2
Type help for help

[BroControl] > start
Starting bro . . . . .
```

After starting bro, it starts capturing packets by using the libpcap packet capture library. Bro script (scan.bro) is written in a bro scripting language to detect port scan attack. This script is kept in bro directory */usr/local/bro/share/bro/site/scan.bro*. Written script *scan.bro* is loaded from another script */usr/local/bro/share/bro/site/local.bro*. *local.bro* is the main entry point in the bro which is managed by BroControl. *Scan.bro* is loaded by adding the following lines into *local.bro*:

```
@load /usr/local/bro/share/bro/site/scan
```

Bro script for detecting port scan is written as follows:

```
export {
    redef enum Notice::Type +=
    {
        PortScan,
    };
}
```

```

const portScanInterval = 4min &redef;
const portScanThreshold = 25.0 &redef;
global Scan::portScanPolicy:
                                hook( attacker: addr,
                                target: addr,
                                scannedPort: port );
                                }
event bro_init() &priority=4
{
local r1:
    SumStats::Reducer =
        [ $data="scan_fail",
          $use=set(SumStats::UNIQUE),
          $fix_maximum=
            double_to_count(2+portScanThreshold) ];
SumStats::create([ $name="port_scanning",
                  $epoch=portScanInterval,
                  $reducers=set(r1),
                  $threshold_val(key: SumStats::Key,
                                result: SumStats::Result) =
                    {
                    return
                    result["scan_fail"]
                    0.0+$fix;
                    },
                  $threshold=portScanThreshold,
                  $thresholdCrossed(key:
                                SumStats::Key, result:
                                SumStats::Result) =

```

```

{
    local a = result ["scan_fail"];
    local x =
    Site::isLocalAddr(target$host) ?
    "local" : "remote";
    local dur =
    duration_to_mins_secs(a$end-a$begin);
    local print =fmt(" %s Port Scan Attack Detected..
    %d different ports of system %s has
    scanned in %s",target$host ,
    a$unique , target$str , dur);
    NOTICE([ $note=Port_Scan ,
            $source=target$host ,
            $destination=to_addr(target$str) ,
            $sub=x,
            $message=print ,
            $id=cat(target$host) ]]);
}

```

Bro generates a number of alert files in */usr/local/bro/logs* containing information about various protocols, connections, scanning, summaries, etc. After packet capturing and analyzing event against this script bro wrote alert message in *notice.log* file. Screenshot of various alert files generated by bro is given in Figure 4.12 and the *notice.log* alert file is given in Figure 4.13.

notice.log file contains a number of alerts with one alert showing “port scan attack detected”. When bro puts this alert in *notice.log* means intrusion has detected. A shell script *automatic_1.sh* was written to continuously monitor the *notice.log* file for automatic activation of digital forensic tool to capture the current system state. Code is written in shell script to search for pattern “Port Scan Attack Detected”

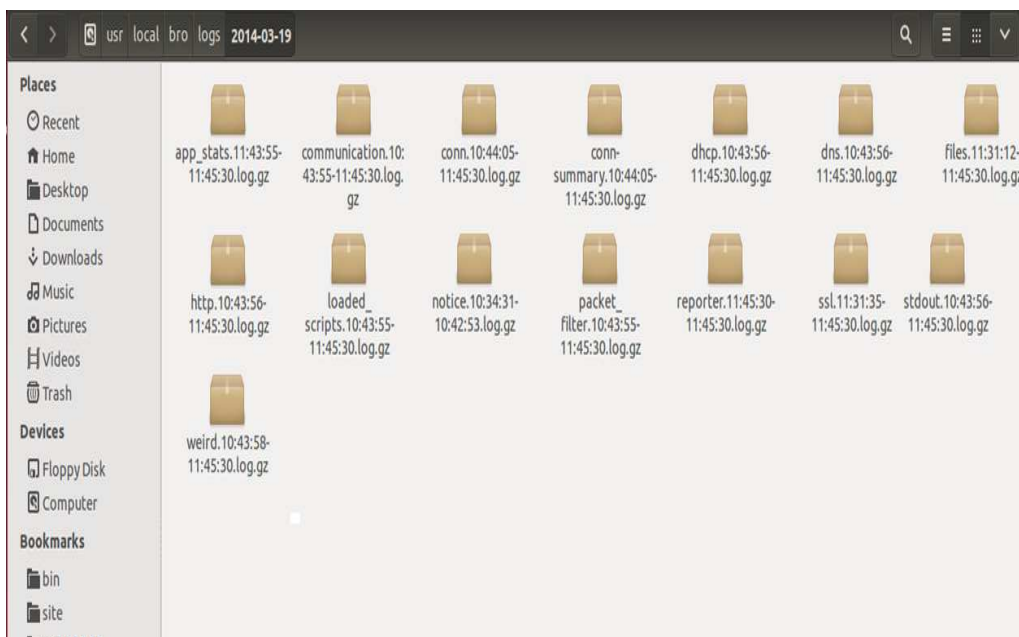


Figure 4.12: Alert files generated by Bro.

and whenever this pattern is matched code is written to run memory acquisition tool LiME to capture memory (RAM) image of the system.

Code used for pattern matching is as follow:

```
prev_count=0
count=$(grep -c -w "Port_Scan"
/usr/local/bro/spool/bro/notice.log)
if [ "$count" -ne "$prev_count" ]; then
echo "there is intrusion : Port Scan Attack Detected..."
```

To capture memory image, LiME 1.1 is installed and configured on the target system by compiling and writing a program to generate an object file to link with the kernel.

To run the LiME, following command is written in shell script:

```
insmod /usr/local/lime-forensics-1.1-r17/src/
lime-3.11.0-17-generic.ko
" path=/usr/local/ram.lime_$now format=lime"
```

```

*notice.10:34:31-10:42:53.log (~/cache/fr-BHU7NC) - gedit
#separator \x09
#set_separator ,
#empty_field (empty)
#unset_field -
#path notice
#open 2014-03-19-10-34-31
#fields ts uid id.orig_h id.orig_p id.resp_h
id.resp_p fuid file_mime_type file_desc proto note msg
sub_src dst p n peer_descr actions suppress_for
dropped remote_location.country_code remote_location.region
remote_location.city remote_location.latitude
remote_location.longitude
#types time string addr port addr port string string string
[enum] interval bool string string string string count string table
1395205471.466423 - - - - - - - - -
Scan::Port Scan Attack Detected.. scanned at least 25 unique ports of host
192.168.40.17 in 0m0s remote 192.168.40.19 192.168.40.17 - -
bro Notice::ACTION_LOG 3600.000000
F - - - - - - - - -
1395205593.841696
PacketFilter::Dropped_Packets 227 packets dropped after filtering, 43073
received, 43081 on link - - - - - bro
Notice::ACTION_LOG 3600.000000
F - - - - - - - - -
1394716850.167524 Cr70BZbvC4RhcY3tj 192.168.40.17 50180
100 50 148 10 442
Plain Text Tab Width: 8 Ln 9, Col 346 INS

```

Figure 4.13: Entries in notice.log file.

Screenshot of running script to capture a memory image is given in Figure 4.14. Once memory image is captured and stored in specified directory, it is analyzed by very popular memory analysis framework Volatility 2.3. Some screenshots of results are given below. Figure 4.15 shows all running processes, Figure 4.16 shows open files, and Figure 4.17 shows entries in the system call table.


```

root@ubuntu: /
root@ubuntu:/# ./automatic_1
grep: /usr/local/bro/spool/bro/notice.log: No such file or directory
./automatic_1: line 7: [: : integer expression expected
no intrusion
grep: /usr/local/bro/spool/bro/notice.log: No such file or directory
./automatic_1: line 7: [: : integer expression expected
no intrusion
there is intrusion : Port Scan Attack Detected...
Memory Image Capturing.....

```

Figure 4.14: Capturing memory image.

```

root@ubuntu: /usr/local/volatility-2.3.1
root@ubuntu: /usr/local/volatility-2.3.1# ./vol.py -f /usr/local/ram.lime_2014-03-19.10:34:57 --profile=LinuxUbuntu1310x86 plugin name linux pslist
Volatility Foundation Volatility Framework 2.3.1
Offset      Name                Pid      Uid      Gid      DTB
-----
Start Time
-----
0xf70f0000  init                1        0        0        0x368f300
0 2014-03-19 04:27:34 UTC+0000
0xf70f0ce0  kthreadd            2        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf70f19c0  ksoftirqd/0        3        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf70f3380  kworker/0:0H        5        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf70f4d40  migration/0        7        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf70f5a20  rcu_bh              8        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf70f6700  rcu_sched           9        0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf7120000  watchdog/0        10       0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf7126700  khelper            11       0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf7138000  kdevtmpfs          12       0        0        -----
- 2014-03-19 04:27:34 UTC+0000
0xf7138ce0  netns              13       0        0        -----
- 2014-03-19 04:27:34 UTC+0000

```

Figure 4.15: List of all active processes.

```

root@ubuntu: /usr/local/volatility-2.3.1
root@ubuntu: /usr/local/volatility-2.3.1# ./vol.py -f /usr/local/ram.lime_2014-03-19.10:
34:57 --profile=LinuxUbuntu1310x86 plugin_name linux_lsof
Volatility Foundation Volatility Framework 2.3.1
Pid      FD      Path
-----
1        0      /dev/null
1        1      /dev/null
1        2      /dev/null
1        3      pipe:[7413]
1        4      pipe:[7413]
1        5      /anon_inode:/inotify
1        6      /anon_inode:/inotify
1        7      socket:/UNI:[7414]
1        9      /var/log/upstart/systemd-logind.log
1        10     socket:/UNI:[7698]
1        11     socket:/UNI:[8622]
1        12     socket:/UNI:[8153]
1        15     socket:/UNI:[8189]
1        19     /dev/ptmx
1        20     /dev/ptmx
1        21     /var/log/upstart/modemmanager.log
1        22     /dev/ptmx
1        39     /dev/ptmx
324     0      /dev/null
324     1      /dev/null
324     2      /dev/null
324     3      socket:/UNI:[7689]
324     4      pipe:[7690]
324     5      pipe:[7690]
324     6      socket:/NETLINK:[7681]

```

Figure 4.16: List of all open files.

```

root@ubuntu: /usr/local/volatility-2.3.1
root@ubuntu: /usr/local/volatility-2.3.1# ./vol.py -f /usr/local/ram.lime_2014-03-19.10:
34:57 --profile=LinuxUbuntu1310x86 plugin_name linux_check_syscall
Volatility Foundation Volatility Framework 2.3.1
Table Name      Index Address      Symbol
-----
32bit           0x0 0xc1063090 sys_restart_syscall
32bit           0x1 0xc1054ee0 Sys_exit
32bit           0x2 0xc1051c80 sys_fork
32bit           0x3 0xc116ba80 Sys_read
32bit           0x4 0xc116bb10 sys_write
32bit           0x5 0xc116a7d0 sys_open
32bit           0x6 0xc116a860 Sys_close
32bit           0x7 0xc10551d0 sys_waitpid
32bit           0x8 0xc116a830 sys_creat
32bit           0x9 0xc11798a0 Sys_link
32bit           0xa 0xc11795a0 Sys_unlink
32bit           0xb 0xc1171e40 sys_execve
32bit           0xc 0xc1169ef0 sys_chdir
32bit           0xd 0xc1055f20 Sys_time
32bit           0xe 0xc11793f0 sys_mknod
32bit           0xf 0xc116a1c0 sys_chmod
32bit          0x10 0xc10afe60 Sys_lchown16
32bit          0x11 0xc1070da0 compat_sys_lookup_dcookie
32bit          0x12 0xc116fae0 Sys_stat
32bit          0x13 0xc116b050 sys_lseek
32bit          0x14 0xc1065420 sys_getpid
32bit          0x15 0xc11879c0 sys_mount
32bit          0x16 0xc1186cf0 sys_oldumount

```

Figure 4.17: Entries in System call table.

After running the second model, Bro-IDS generated 15 log files in the log directory. Even some of the log files did not contain any information, but they were created by Bro-IDS. Some log files such as *notice.log*, *connection.log* contains information about alerts and connections. The total number of alerts and two most often alerts in *notice.log* is shown in Figure 4.18. The total number of alerts

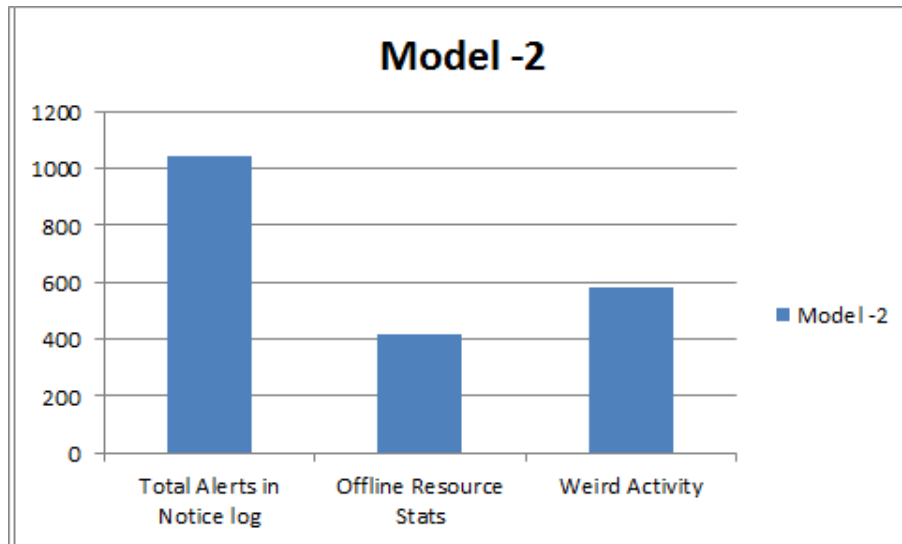


Figure 4.18: Alert generated in notice.log.

generated by Bro in *notice.log* were 1040, 420 of them were of type Offline Resource Stats, and 580 were of type Wierd Activity. Alert type Offline Resource Stats indicates the number of queued event and Weird Activity alert indicates packet in which IPv4 not included. Another important log file is *conn.log* which contains all connections from or to the target host. After analyzing *conn.log* file we found there are 8140 total number of connections, 6650 http connections, 635 ssh connections, and 210 SMTP connections.

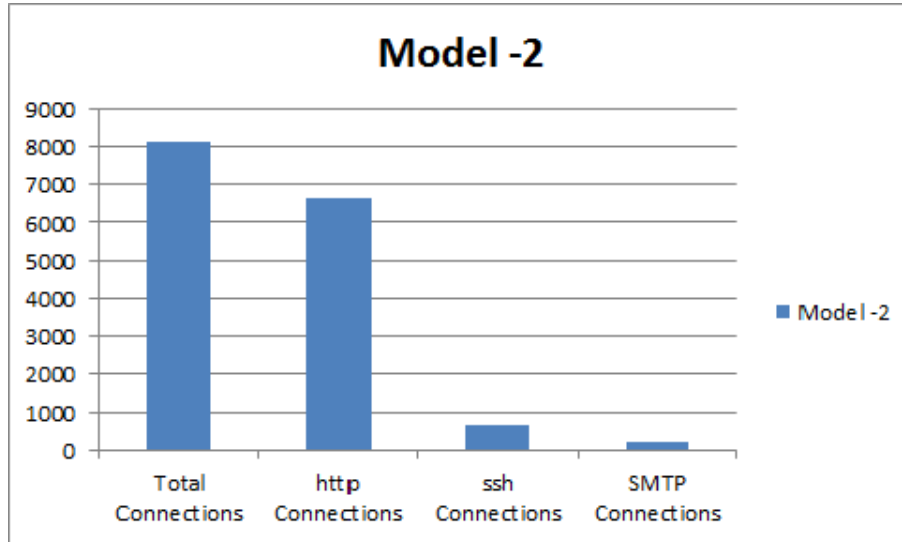


Figure 4.19: Connections in conn.log.

4.3 Comparison of Models

After analyzing the memory image of both the models, we found relevant information of the system of the time when the attack occurred. We have taken some parameters from information to draw a comparison of the models. Parameters are, number of active processes, number of active connections, number of open files, number of device driver detected, number of loaded modules, number of hidden processes, and the number of commands in command history. Number of processes, connections and so on is given in Table 4.1.

Table 4.1: Comparison of Models

Information /Model	Number of processes	Number of Connections	Number of open files	Number of device drivers	Number of Loaded Modules	Number of Hidden Processes	Command History
Model -1	128	53	1820	139	167	242	3
Model -2	203	68	1490	164	110	363	3

Graph in Figure 4.20 show comparison of information retrieved from both the models.

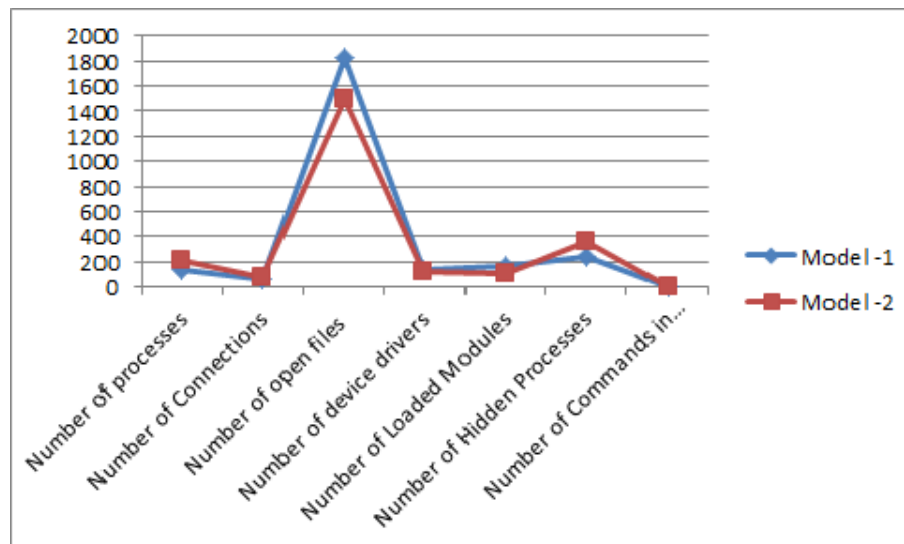


Figure 4.20: Information retrieved from both models.

Chapter 5

Conclusion

In this work “Application of Intrusion Detection System in Automatic Evidence Collection using Digital Forensics” is proposed by using both signature and anomaly based intrusion detection system. In this work intrusion detection system is used to detect intrusion and then current system image is captured by digital forensic tool. Further, this captured memory image is analyzed by using forensic analysis tool. In this work two models have been given, first model used signature based intrusion detection and other is using anomaly based intrusion detection. In our experimental study for first model, snort is used as intrusion detection system, DumpIt is used as a digital forensic tool which capture RAM image of the system, and Memgator is used as a forensic analysis tool which analyze image to find relevant information from it. To test our model, rules were written in snort to detect ICMP ping as an intrusion and Ping message from another system is sent. Our model has successfully detected the intrusion and captured the memory image which was further analyzed by forensic analysis tool and relevant information (i.e. running processes, connections) have been collected.

In experimental study of the second model, Bro-IDS is used as intrusion detection system, LiME is used as a digital forensic tool, and volatility is used as a forensic analysis tool. To test this model a bro script is written and configured on the target system, and port scanning is done from another system (intruder system) by using

Nmap. Our model successfully detected the port scan attack and captured memory image followed by the analysis of the image.

Finally a comparison of both models is given based on the alerts shown, connections established, and information retrieved from memory image. Model with Bro-IDS is better than the model with Snort because Bro-IDS is able to detect novel attacks which cannot be detected by Snort.

Bibliography

- [1] Wu Kehe, Zhang Tong, Li Wei, and Ma Gang. Security model based on network business security. In *Computer Technology and Development, 2009. ICCTD'09. International Conference on*, volume 1, pages 577–580. IEEE, 2009.
- [2] Mohammad Ashiqur Rahman and Ehab Al-Shaer. A formal framework for network security design synthesis. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 560–570. IEEE, 2013.
- [3] Chenghua Tang and Shunzheng Yu. Assessment of network security policy based on security capability. In *Communication Systems, 2008. ICCS 2008. 11th IEEE Singapore International Conference on*, pages 1204–1208. IEEE, 2008.
- [4] Behrouz A Forouzan. *Cryptography & Network Security*. McGraw-Hill, Inc., 2007.
- [5] Xin Yue, Wei Chen, and Yantao Wang. The research of firewall technology in computer network security. In *Computational Intelligence and Industrial Applications, 2009. PACIA 2009. Asia-Pacific Conference on*, volume 2, pages 421–424. IEEE, 2009.
- [6] Xiangqian Chen, Kia Makki, Kang Yen, and Niki Pissinou. Sensor network security: a survey. *Communications Surveys & Tutorials, IEEE*, 11(2):52–73, 2009.
- [7] S Zargar, James Joshi, and David Tipper. A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks. 2013.
- [8] Ke Wei, Muthusrinivasan Muthuprasanna, and Suraj Kothari. Preventing sql injection attacks in stored procedures. In *Software Engineering Conference, 2006. Australian*, pages 8–pp. IEEE, 2006.
- [9] Tarek S Sobh. Wired and wireless intrusion detection system: Classifications, good characteristics and state-of-the-art. *Computer Standards & Interfaces*, 28(6):670–694, 2006.
- [10] Abdallah Ghourabi, Tarek Abbes, and Adel Bouhoula. Honeypot router for routing protocols protection. In *Risks and Security of Internet and Systems (CRiSIS), 2009 Fourth International Conference on*, pages 127–130. IEEE, 2009.

- [11] Clive Grace. Understanding intrusion detection systems. *PC Network Advisor*, 122:11–15, 2000.
- [12] Debra Anderson, Teresa F Lunt, Harold Javitz, Ann Tamaru, Alfonso Valdes, et al. *Detecting unusual program behavior using the statistical component of the Next-generation Intrusion Detection Expert System (NIDES)*. SRI International, Computer Science Laboratory, 1995.
- [13] Ahmed Patel, Qais Qassim, and Christopher Wills. A survey of intrusion detection and prevention systems. *Information Management & Computer Security*, 18(4):277–290, 2010.
- [14] Phil Porras, Dan Schnackenberg, and Stuart Staniford-Chen. Common intrusion detection framework architecture. <http://gost.isi.edu/cidf/drafts/architecture.txt>, Accessed on February 3, 2014.
- [15] Glenn A Fink, BL Chappell, TG Turner, and KF O’Donoghue. A metrics-based approach to intrusion detection system evaluation for distributed real-time systems. Technical report, DTIC Document, 2002.
- [16] Anita K Jones and Robert S Sielken. Computer system intrusion detection: A survey. *Computer Science Technical Report*, pages 1–25, 2000.
- [17] Karen Scarfone and Peter Mell. Guide to intrusion detection and prevention systems (idps). *NIST special publication*, 800(2007):94, 2007.
- [18] Aurobindo Sundaram. An introduction to intrusion detection. *Crossroads*, 2(4):3–7, 1996.
- [19] Stefan Axelsson. Intrusion detection systems: A survey and taxonomy. Technical report, Technical report, 2000.
- [20] Sapna S Kaushik and PR Deshmukh. Detection of attacks in an intrusion detection system. *International Journal of Computer Science and Information Technologies (IJCSIT)*, 2(3):982–986, 2011.
- [21] Aleksandar Lazarevic, Levent Ertöz, Vipin Kumar, Aysel Ozgur, and Jaideep Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *SDM*. SIAM, 2003.
- [22] Ismail Butun, S Morgera, and Ravi Sankar. A survey of intrusion detection systems in wireless sensor networks. 2013.
- [23] Pieter de Boer and Martin Pels. Host-based intrusion detection systems. *Amsterdam University*, 2005.
- [24] Wikipedia. Intrusion detection system. http://de.wikipedia.org/wiki/Intrusion_Detection_System, Accessed on May 3, 2014.

- [25] Thomas D Garvey and Teresa F Lunt. Model-based intrusion detection. In *Proceedings of the 14th national computer security conference*, volume 17, 1991.
- [26] Mauro Andreolini, Sara Casolari, Michele Colajanni, and Mirco Marchetti. Dynamic load balancing for network intrusion detection systems based on distributed architectures. In *Network Computing and Applications, 2007. NCA 2007. Sixth IEEE International Symposium on*, pages 153–160. IEEE, 2007.
- [27] PMG. Maximizing the value of network intrusion detection. Technical report, A Corporate White Paper from the Product Management Group of Intrusion.com, 2001.
- [28] Eugene H Spafford and Diego Zamboni. Intrusion detection using autonomous agents. *Computer networks*, 34(4):547–570, 2000.
- [29] T Wu. Information assurance tools report–intrusion detection systems. Retrieved online from http://iac.dtic.mil/csiac/download/intrusion_detection.pdf, 2009.
- [30] Frank Y Shih. *Multimedia Security: Watermarking, Steganography, and Forensics*. CRC Press, 2012.
- [31] Eoghan Casey. *Digital evidence and computer crime: forensic science, computers and the internet*. Academic press, 2011.
- [32] Palmer Gary. A road map for digital forensic research. In *Digital Forensics Research Workshop*, 2001.
- [33] Ricci SC Jeong. Forza–digital forensics investigation framework that incorporate legal issues. *digital investigation*, 3:29–36, 2006.
- [34] Sara V Hart, John Ashcroft, and Deborah J Daniels. Forensic examination of digital evidence: a guide for law enforcement. *National Institute of Justice NIJ-US, Washington DC, USA, Tech. Rep. NCJ*, 199408, 2004.
- [35] Brian Carrier, Eugene H Spafford, et al. Getting physical with the digital investigation process. *International Journal of digital evidence*, 2(2):1–20, 2003.
- [36] Carol Taylor, Barbara Endicott-Popovsky, and Deborah A Frincke. Specifying digital forensics: A forensics policy approach. *digital investigation*, 4:101–104, 2007.
- [37] Golden G Richard III and Vassil Roussev. Scalpel: A frugal, high performance file carver. In *DFRWS*, 2005.
- [38] Simson L Garfinkel. Digital forensics research: The next 10 years. *Digital Investigation*, 7:S64–S73, 2010.

-
- [39] Daniel J Ryan and Gal Shpantzer. Legal aspects of digital forensics. In *Proceedings: Forensics Workshop*, 2002.
- [40] Brian Cusack and Muteb Alqahtani. Acquisition of evidence from network intrusion detection systems. 2013.
- [41] Jorge Herrerias and Roberto Gomez. A log correlation model to support the evidence search process in a forensic investigation. In *Systematic Approaches to Digital Forensic Engineering, 2007. SADFE 2007. Second International Workshop on*, pages 31–42. IEEE, 2007.
- [42] Ali Reza Arasteh, Mourad Debbabi, Assaad Sakha, and Mohamed Saleh. Analyzing multiple logs for forensic evidence. *digital investigation*, 4:82–91, 2007.
- [43] Ya-Ting Fan and Shiuh-Jeng Wang. Intrusion investigations with data-hiding for computer log-file forensics. In *Future Information Technology (FutureTech), 2010 5th International Conference on*, pages 1–6. IEEE, 2010.
- [44] Fahmid Imtiaz. Intrusion detection system logs as evidence and legal aspects. Technical report, School of Computer and Information Science Edith Cowan University, 2007.
- [45] Leonard Kwan, Pradeep Ray, and Greg Stephens. Towards a methodology for profiling cyber criminals. In *Hawaii International Conference on System Sciences, Proceedings of the 41st Annual*, pages 264–264. IEEE, 2008.
- [46] Federal Agent Byron S Collie, Headquarters Air Command, and Royal Australian Air Force. Intrusion investigation and post-intrusion computer forensic analysis. *Headquarter Air Command, Royal Australian Air Force, year=2006*.
- [47] Peter Stephenson. The application of intrusion detection systems in a forensic environment. In *The Third International Workshop on Recent Advances in Intrusion Detection (RAID)*, 2000.
- [48] Fahmid Imtiaz. Intrusion detection systems and a view to its forensic applications. Technical report, The University of Melbourne Department of Computer Science, 2000.
- [49] Peter Sommer. Intrusion detection systems as evidence. *Computer Networks*, 31(23):2477–2487, 1999.
- [50] Komal Barhate and CD Jaidhar. Automated digital forensic technique with intrusion detection systems. In *Advance Computing Conference (IACC), 2013 IEEE 3rd International*, pages 185–189. IEEE, 2013.
- [51] Martin Roesch et al. Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238, 1999.

- [52] MC Moya. Analysis and evaluation of the snort and bro network intrusion detection systems, 2008.
- [53] Pritika Mehra. A brief study and comparison of snort and bro open source network intrusion detection systems, 2012.
- [54] Russ McRee. Memory analysis with dumpit and volatility, 2011.
- [55] Andrew Smith. Memgator - memory analysis tool. <http://www.orionforensics.com/MemGator.php>, 2009(Accessed on May 3, 2014).
- [56] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer networks*, 31(23):2435–2463, 1999.
- [57] Robin Sommer. Bro: An open source network intrusion detection system. In *DFN-Arbeitstagung über Kommunikationsnetze*, pages 273–288, 2003.
- [58] Joe Sylve. Lime-linux memory extractor. *ShmooCon12*, 2012.
- [59] Michael Hale Ligh Mike Auty, Andrew Case. Volatility: An advanced memory forensics framework. <https://code.google.com/p/volatility/>, 2013(Accessed on May 3, 2014).
- [60] Gordon Lyon. Nmap: Network mapper. <http://nmap.org/docs.html>, 2014(Accessed on May 3, 2014).
- [61] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Surveying port scans and their detection methodologies. *The Computer Journal*, 54(10):1565–1581, 2011.