Final Year Project

Academic Year: 2020-2021

Faculty: Science

Major: Computer Science

Subject: Intrusion Detection Systems; Overview and Implementation

Supervising Professor: Dr. Carol Bassil

Students: Michel Sleiman Rahme 56312

Michael Saba 54851

Abstract: Intrusion detection systems (IDS) help detect unauthorized activities or intrusions that may compromise the confidentiality, integrity or availability of a resource. This final year project represents a general view over the field of IDSs; the solutions it has to the world of security and the problems that it faces.

Keywords: IDS, HIDS, NIDS, Bayes, inline, IPS, anomaly, signature, Agent, hybrid intrusion detection system

---

1. INTRODUCTION

Intrusion detection is defined as identifying unauthorized use, misuse and abuse of computer systems by both inside and outside intruders. The main task of an intrusion detection system (IDS) is to defend a computer system or computer network by detecting hostile attacks on a network system or host device, monitoring the events occurring in a computer system or network and analyzing them for signs of intrusions. Security incidents resulting from attempted attacks violate the CIA triad of computer security; Confidentiality, Integrity and Availability. The National Institute of Standards and Technology (NIST) defines intrusion as an attempt to compromise CIA or to bypass the security mechanisms of a computer network.
An IDS identifies, logs and reports possibly security incidents. It is the software or hardware system to automate the intrusion detection process and is typically placed inline, at a spanning port of a switch, or on a hub in place of a switch.

2. IDEAL INTRUSION DETECTION SYSTEM

a. Materials and Procedures

An ideal IDS should address the issues below, no matter how effective, to prevent attacks and intruders. Some of these are described below:

i. The system must be able to proceed continuously without human supervision. It should be reliable enough to be operated after the visual system.

ii. It should not be a "black box", which means that internal performance should be checked from the outside.

iii. It has to tolerate errors, which means it has to survive a system crash and not have its database rebuilt when it restarts.

iv. It must withstand destruction. The system is able to monitor itself to ensure that it is not compromised.

v. It should look for and record the deviation from normal behavior.

vi. It should be easily built into the system. The whole system has a different application pattern, and the defense method should easily adapt to these patterns.

vii. It has to deal with changes in system behavior over time as new systems are added.

viii. The system must have a very low level of false and misleading quality.

Since the standard IDS generates a large amount of traffic and events on its logs, the important thing is that the system only generates alerts about events of interest. A valid IDS has a low level of positive and negative benefits. A good adjustment of the IDS, therefore, enhances the conditions of really good and true objections.

Table 1 - True positives and negatives

	POSITIVE	NEGATIVE
TRUE	Alerts when there is malicious traffic	Silent when traffic is benign
FALSE	Alerts when traffic is benign	Silent when malicious traffic occurs

3. CLASSIFICATION OF INRUSION DETECTION SYSTEMS

There is a lot of ways to classify the various types of IDS in a production network. These classifications are not mutually exclusive; for instance, a network-based IDS may be using the signature-based approach to detection. The following diagram describes the most common methodologies of IDS classifications, although the list is certainly not exhaustive.

a. Host-based IDS

An HIDS is an IDS that generally operates within a computer, node or device. Its main function is internal monitoring, although many variants of HIDS have been developed that can be used to monitor networks. Primarily, it monitors and analyzes the internals of a computer, node or device. An HIDS determines if a system has been compromised and warns administrators accordingly. For example, it can detect a rogue program that accesses a system's resources in a suspicious manner, or discover that a program has modified the registry in a harmful way.

HIDSs were the first types of intrusion detection software to have been designed. Unlike network-based IDSs, an HIDS can inspect the full communications stream. NIDS evasion techniques, such as fragmentation attacks or session splicing, do not apply because the HIDS is able to inspect the fully recombined session as it is presented to the operating system. Encrypted communications can be monitored because an HIDS inspection can look at the traffic before it is encrypted. This means that HIDS signatures will still be able to match against common attacks and not be blinded by encryption.

An HIDS is also capable of performing additional system level checks that only IDS software installed on a host machine can do, such as file integrity checking, registry monitoring, log analysis, rootkit detection, and active response.

b. Network-based IDS

NIDS differs from an HIDS in that it is usually placed along a LAN wire. It attempts to discover unauthorized and malicious access to a LAN by analyzing traffic that traverses the wire to multiple hosts. There are many algorithms for detecting malicious traffic, but they generally read inbound and outgoing packets and searches for any suspicious patterns. Any alert generated by an NIDS allows it to notify administrators or take active actions such as blocking the source IP address. Three of the most common placements of NIDS are directly connecting it to a switch spanning port, using a network tap, and connected inline.

Figure 1 shows the IDS connected to a switch that has SPAN port configuration capability. On some managed switches, a SPAN port can be configured to send all packets on the network to that port as well as their ultimate destination. In this configuration, the switch copies all traffic it receives to the IDS interface being used to monitor traffic. The major downside of this method is increased bandwidth and resource usage, since the switch must work twice as hard to deliver traffic.
Very few modern LANs use hubs due to the lack of security. Hubs allow systems to intercept traffic not intentionally sent to them. When using either a hub or switch with SPAN port capabilities, the systems on the internal network are not at the mercy of the IDS having a system failure brining the network down. Making use of a switch SPAN port is a common method of connecting sensors.

Figure 2 shows an IDS using a network tap, which essentially replicates data passed through the wire. Network taps are not commonly found in typical computer networks but may be purchased. Taps are handy when a network administrator needs to setup a hasty monitoring solution, perhaps to troubleshoot a problem or temporarily deploy an IDS. Overall, a network tap is needed when the network does not have managed switches, is not using hubs, or when putting an IDS inline is impractical.

Figure 3 illustrates an IDS connected inline. This instance includes two connections, shown in red, with one connected to the uplink port of the switch, and the second connected to the external network. In most cases, this is not the best method to use because system failure of the IDS will prevent systems on the internal network from communicating with external systems.
However, the benefit of the inline configuration is a guarantee all packets will be seen by the IDS. Packets are subject to being missed when an IDS is connected to a switch SPAN port, especially when that switch is busy processing a large burst of traffic. Missed traffic may be lost forever if they were not captured by a network sniffing protocol. Depending on the capability of an inline IDS, a similar burst may lead to congestion of network performance.
Although an NIDS is a powerful monitoring system for network traffic, there are several disadvantages. Common NIDS evasion techniques such as fragmentation attacks, session splicing, and even denial-of-service (DoS) attacks can be used to bypass an NIDS, rendering it useless. If the communications between hosts are encrypted, a passive NIDS does not have the ability to unencrypt a message in transition.

Figure 1 - IDS using a switch spanning port

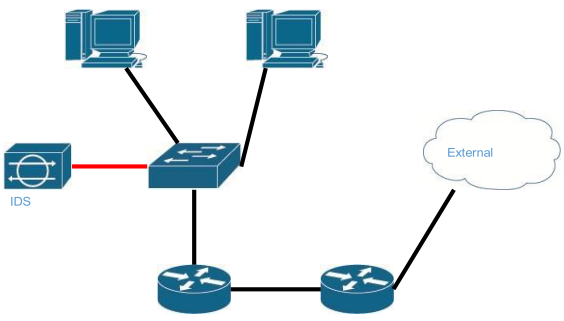


Figure 2 - IDS using network tap

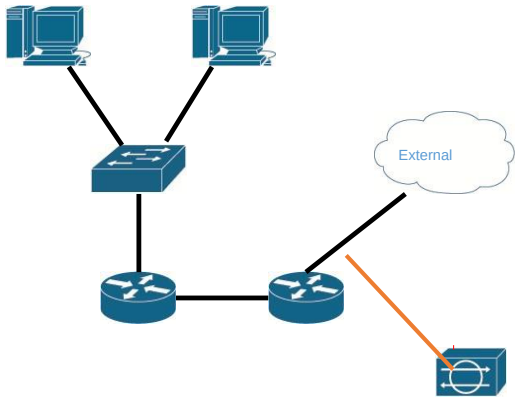
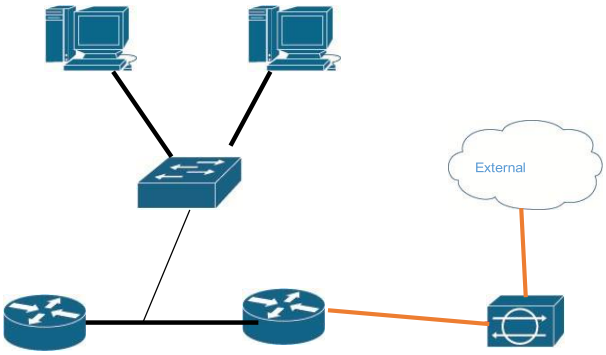


Figure 3 - IDS connected inline



c. Signature-based Detection

Signature-based approach, IDS looks for packets and compares them with predefined rules or patterns known as database signatures. These attack signatures override specific traffic or activity based on a known intervention function. The main advantage of this process is a simple and effective analysis of the audit data. Signature-based methods have a very low level of false positives. On the other hand, the very nature of the signature means that the same method does not apply to zero-day attacks in which there may have been no law enforcement or current method of attack. With the level of new attacks and malicious activity every hour, the signature-based IDS is similar to the dignity of its database and signatures.

d. Anomaly-based Detection

Anomaly based approach works by identifying patterns from users or groups of users already defined. This approach looks for differences and deviations from the basic values that can reflect evil. It includes an increase in the amount of processing used by an uncontrolled detector to study system performance in its research. The first foundation must be made of a program, network or program function. This base is a profile of how a general condition, usage, bandwidth or behavior will look in a particular area of the network. Thereafter, any activity that deviates from the foundation is treated as a possible intervention and notice will be given. The main advantage of an anomaly-based method is its ability to detect zero-day attacks, as it does not depend on the established signature database, but only deviations from the established base. The behavior of each target system or network is different, so methods based on malware use custom profiles that make it difficult for the attacker to know for sure what task they can do without turning off the alarm. On the other hand, anomaly-based IDS have a high level of false positives. It also takes time to establish basic functionality when it is initially installed in a new network or host device. These systems are also very complex and difficult to integrate an alarm with a specific event that created that alarm.

e. Passive and Active IDS

When classifying IDSs, we can also categorize them by the way IDSs respond during an attack. A passive IDS merely records, analyzes, logs and alerts an administrator about the possibility of an attack. In terms of placement, passive IDSs are generally placed off to the side in a network.

An active IDS can take actions when it detects a possible intrusion, such as blocking further traffic from a specific network source or locking down the system with safe mode. In modern security systems, an active IDS is also known as an Intrusion Prevention System (IPS). IPSs are placed inline in a network.

i. Intrusion Prevention Systems (IPS)

The main function of an IPS is to intervene in cases of suspected attacks. Generally, an IPS is essentially a combination of access control devices – such as firewalls and routers – and IDSs. In other words, an IPS is an IDS with access control capabilities or active response methods.

Like IDSs, an IPS can be host-based or network-based, and uses anomaly-based detection (prevention) or a signature- or ruleset-based approach.

Common countermeasures implemented by IPSs:

- Denying the traffic. This is the simplest method, where the intrusion system blocks. the IP addresses and ports involved – both source and destination. The downside to this method is that many devices on the global network are hidden behind a global address. Blocking that address will also block other legitimate traffic that may be located behind that address.
- Active logging. Although logging is a feature shared by IDSs, an IPS can increase the usability of a log by, for example, automatically exporting traffic logs that meet certain criteria to external network analysis software such as Wireshark.
- Communicating with a separate device with access control capabilities. Many modern IDSs and IPSs also complement the operations of a LAN by communicating with an external, or separate, firewall or router, which have access control capabilities. In the event of an intrusion, an IDS/IPS can send an alert or request to a firewall or router. The firewall or router will then take the necessary actions to deal with the intrusion, such as dropping the packets or blocking further traffic from that source.
- Sending a TCP reset. If an attack is a TCP-based attack, an IPS can send a reset signal back to the attacker's protocol stack, which would close the current session, and can be repeated as frequently as needed.
- Setting an SNMP trap. When an alarm is triggered, the intrusion system will send an SNMP trap to indicate to an SNMP management system that a network or device is under attack. The management system can choose to take an action based on the event, such as polling the agent directly, or polling other associated device agent to get a better understanding of the event.

ii. IPS or IDS?

Although the market trend is focusing on IPS rather than IDS with the advancement of DDoS attacks, there are reasons to choose between an IPS and an IDS. IDS rarely cause latency in network traffic, because it is generally off to the side (unless placed inline) and all traffic are merely copied to the sensor. An IPS, on the other hand, may cause small delays in traffic because its inline placement means that every packet has to be inspected and analyzed before being forwarded to its destination.
If an intrusion system is disabled, for example by accident or power outage, an IDS will not cause denial of service due to its positioning. An IPS, which would have to be connected inline, would negate the availability of network resources. Modern IPSs have preventive mechanisms such as fault tolerance or backup power to minimize disruption to network activities.
If cost is an issue, many routers such as Cisco® routers allow specific modules or firmware to be installed on top of the existing routers to provide intrusion detection and prevention capabilities.

Table 2 - Comparison between different IDS classifications

Type	Advantages	Disadvantages
HIDS	1. More accurate in intrusion detection 2. Able to detect encrypted attacks. 3. Does not require additional hardware	1. Higher cost. 2. May cause performance issues or resource hogging.
NIDS	1. Low cost. 2. Detect network- based attacks such as	1. High fluctuations in network traffic cause packets to be lost.
	denial-of- service attacks.	2. Requires more CPU power and resources in a large-scale LAN. 3. Unable to analyze encrypted packets.
Anomaly- based	1. Ability to detect zero- day attack attempts. 2. Low false negative rate.	1. Slow to work when placed in a new environment. 2. High false positive rate. 3. Low detection rate for known attacks.
Signature- based	1. High response time for known attacks. 2. Low false positive rate.	1. Limited capability to detect zero- day attacks. 2. Signature database must be updated frequently.

4. DETECTION TECHNIQUES

From a variety of sources, systems such as expert legal systems, state reform analysis, and genetic algorithms are specific and effective ways to initiate signature acquisition. Subsequent incoming patterns, artificial neural networks, statistical analysis and data mining methods have been used for incorrect detection. There are various types of framework used for anomaly-based detection.
This section presents an in-depth study of various intrusion techniques and hybrid detection techniques. A few suggested methods can be described as follows.

a. Bayesian Networks
Bayesian networks are probabilistic graphical models that represent sets of variables and their probabilistic independencies. Bayesian theory is named after Thomas Bayes. His theory can be explained as follows:
If the events A1, A2, … and An constitute a partition of the sample space S such that P(Ak) ≠ 0 for k = 1,2 , … , n, then for any event B such that P(B) ≠ 0:

$$P(A_i|B) = \frac{P(A_i \cap B)}{P(B)} = \frac{P(A_i)P(B|A_i)}{\sum_{k=1}^{n} P(A_k)P(B|A_k)} = \frac{P(A_i)P(B|A_i)}{P(B)}$$

Darwiche, et. al (2010) stipulated that Bayesian networks have been used in many computer science fields, such as email spam filters, speech recognition and pattern recognition, because of their ability to build coherent results by using probabilistic information about a specific situation.
Bayesian networks are directed acyclic graphs where the nodes represent variables and whose edges encode conditional dependencies between those variables. These are applied to anomaly detection in so many ways; for example, Valdes et al. has developed an anomaly detection system that employed naive Bayes, which is a two-layer Bayesian network that assumes complete independency between the nodes.

b. Genetic Algorithm (GA)
GA is a search technique that is used to find an appropriate solution to search problems. Genetic algorithms have been applied in anomaly detection in many ways, as they are flexible and a powerful search method. Some network intrusion detection approaches have used genetic algorithms for the classification of instances, while others like fuzzy data mining approach have applied this technique for feature selection. To list out an advantage of GA, it selects the best feature and has better efficiency but its method is complex.

c. Inductive Rule Generation Algorithms
These algorithms are one of the most famous techniques used. In this technique, we have a predictive model decision tree that maps observations of an item to conclusions about the item's target value.
The decision tree (DT) is very powerful and popular data mining algorithm for decision-making and classification problems. It is also used in many real-life applications like medical diagnosis, radar signal classification, weather prediction, credit approval, and fraud detection. This decision tree can be constructed from large volume of dataset with many attributes, because the tree size is independent of the dataset size. It can process both numerical and categorical data but trees created from numeric datasets can be complex. Construction of inductive rule

generation algorithms may not require any domain knowledge. It can handle high dimensional data and the representation is easy to understand. However, it is limited to one output attribute. Decision tree algorithms are unstable and most decision tree construction methods are non-backtracking.

d. Outlier Detection

Outlier detection approach is based on the idea of semi-supervised learning in which the system would learn a baseline data, and consider any instances that do not fit in the normal data profile as an anomaly.

Most of the anomaly detection algorithms require a set of baseline data to train the model, and they assume that anomalies can be treated as patterns never observed before. Since an outlier is defined as a data point which is very different from the rest of the data, so based on some measure, we employ several outlier detection schemes to see how efficiently these schemes may deal with the problems of anomaly detection. In statistics-based outlier detection techniques, the data points are modeled using a stochastic distribution and these points are determined to be outliers depending upon their relationship with this model.

e. Clustering

This technique is based on two important assumptions. First, majority of the network connections represent normal traffic and only a very small percentage of that traffic is malicious. And second, malicious traffic is statistically different from normal traffic. Anomalies will be detected based on their cluster size, i.e., large clusters are meant to be baseline data, and the rest correspond to malicious attacks.

Clustering is unsupervised learning. Labeling the data is not necessary and natural patterns in the data are extracted. It does not require the use of a labeled data set for training.

f. Neural Networks

Neural networks are networks of computational units that jointly implement complex mapping functions. First, the networks are trained with a labeled data set. Testing instances are then fed into the network to be classified as either normal or anomalous. An example of the neural network technique which is widely used in anomaly detection is the Support Vector Machines (SVM).

This method would be effective if the exact characteristics of the attack are already known. However, these intrusions are constantly changing because of the individual approaches taken by the attackers and regular changes done in the software and hardware of the targeted systems. Because of the wide variety of attacks and attackers despite their dedicated effort to constantly update the rule base of an expert system can never hope to accurately identify the variety of intrusions.

For these constantly changing natures of these network attacks, we require a flexible defensive system that can analyze these enormous amounts of network traffic in a manner, which is less structured than rule-based systems. For example, a neural network-based signature detection system could potentially address many of the problems that are found in rule-based systems. The inherent speed of neural networks is another benefit of this approach, as it requires a timely identification of attacks, and the processing speed of the neural networks could enable intrusion responses to be conducted before irreversible damage could be done to the system. It has a high signal-to-noise ratio and requires more time and more sample-training phase.

g. Fuzzy Logic

Fuzzy logic starts and builds on a set of user-supplied human language rules. The fuzzy systems convert these rules into their mathematical equivalents. This simplifies the job of the system designer and the computer, and results in a much more accurate representation in the way systems behave in the real world. Fuzzy logic is also simple and flexible. It can handle problems arising from imprecise and incomplete data, and model nonlinear functions of arbitrary complexities. Fuzzy logic techniques have been employed in the computer security field since the early 90's. Its ability to model complex systems makes it a valid alternative in the computer security field to analyze continuous sources of data and even unknown or imprecise processes.

Fuzzy logic has the potential in the intrusion detection field when compared to those systems using strict signature based matching or classic pattern deviation detection. Bridges and Vaughn state that the concept of security itself is fuzzy. And in other words, the concept of fuzziness helps to smooth out the abrupt separation of normal behavior from abnormal behavior. This means a given data point falling outside a defined baseline interval, will be considered anomalous to the same degree regardless of its distance from the interval. Fuzzy logic has a capability to represent imprecise forms of reasoning in areas where firm decisions must be made in indefinite environments like intrusion detection.

Dokas et al. suggested a model that works by building rare class prediction models for identifying known intrusions and their variations and anomaly/outlier detection schemes for detecting novel attacks whose nature is unknown.

Researchers propose techniques to generate fuzzy classifiers using genetic algorithms that can detect anomalies and some specific intrusions. The main idea was to evolve two rules; one for the normal class and other for the abnormal class using a baseline profile data set.

5. DESIGN OF DISTRIBUTED IDS

a. Design of a distributed NIDS model of system

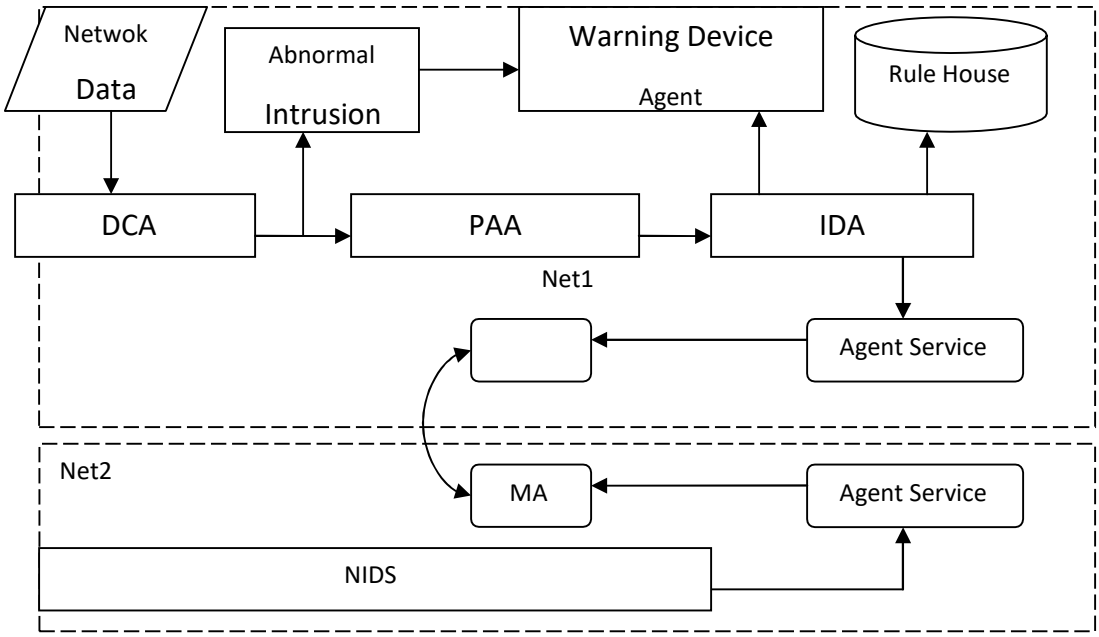


Fig.1 Network-based intrusion detection system

Introducing Agent in the network-based intrusion detection system, it can carry data and protocols directly move, improve network security, and so on.

- Data Collection Agent (DCA）
  Data collection Agent is mainly responsible for simple filtering packets on the network, reducing
  the data of the follow module you want to analyze, and relieve pressure on the system, then capture packets that are not filtered, and packets. Implement two functions: data capture and data filtering.
- Protocol Analysis Agent (PAA）
  Protocol Analysis Agent is the whole core of the intrusion detection system, in accordance with
  protocol analysis algorithm for IP network-layer intrusion detection analysis of the attack, and in accordance with the protocol network packets for analysis of the results of the analysis, enabling packets to divert to the analysis of intrusion detection system based on protocol analysis to be dealt with.

- Intrusion Detection Agent (IDA）
  Primarily responsible for systems management, detection of an intrusion, write rules to update
  the data, and so on. Alarm Agent is detected after the intrusion and do various forms of reactions.
- DMobile Agent （MA）
  Collect data for intrusion detection system, when necessary, coordinate the network operation

and collaboration of the intrusion detection system, increase interface of Mobile Agent in intrusion detection subsystem so that it has the ability to interact with mobile code closely.

b. Design of a distributed HIDS model of system

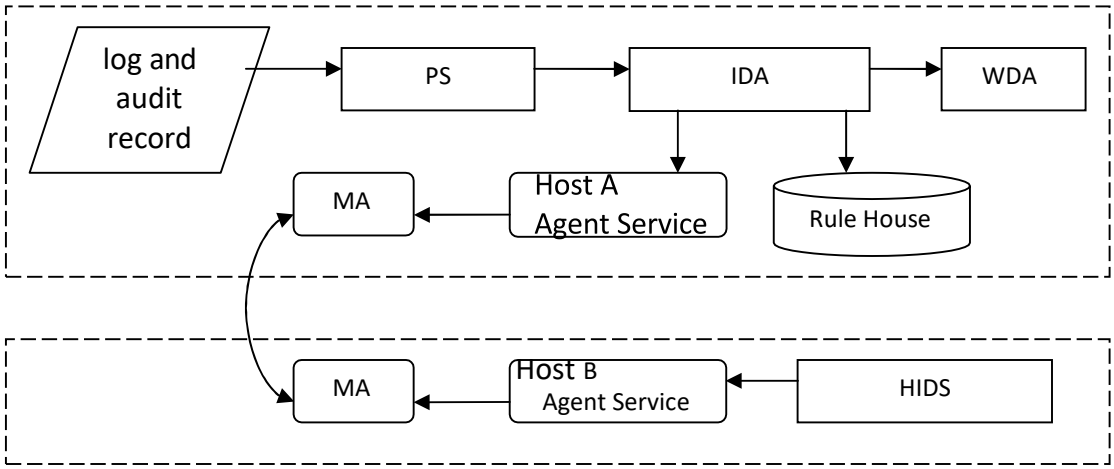


Fig.2 Host-based intrusion detection system

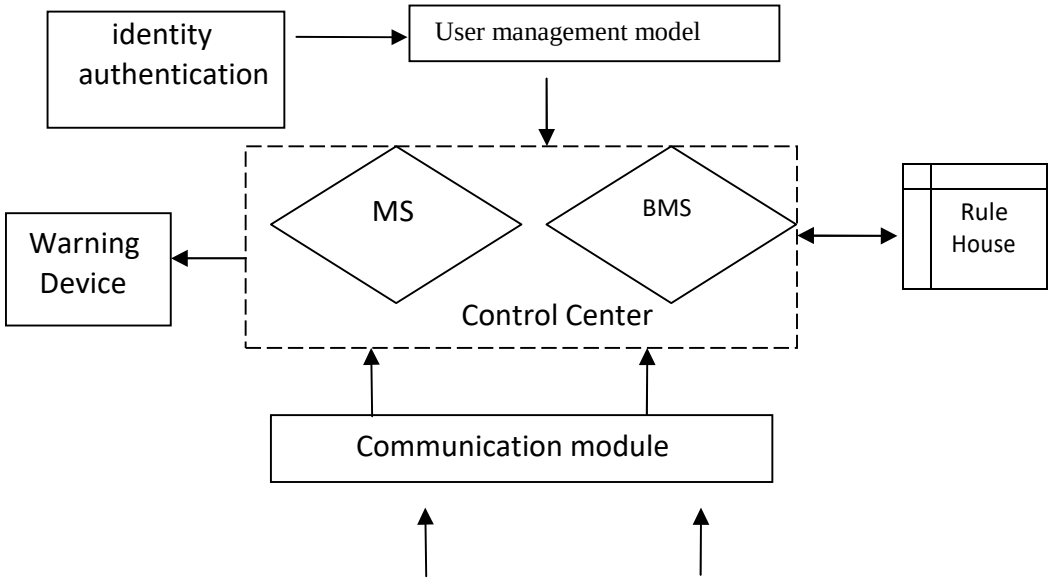
- Preprocessing Section (PS）
  The data of Host-based intrusion detection system mainly from log and audit records of system,
  preprocessing section is responsible for data normalization processing, extracting useful information, improve testing efficiency.
- Intrusion Detection Agent (IDA）
  Similar to the network-based IDS. Differences: when detecting abnormal behavior, it will produce alarm logs, and put it into the rule base for future query analysis.
- Mobile Agent （MA）
  Role of Mobile Agent: intrusion detection system detects an event, because of their limited,
  failed to make accurate judgments, then the assistance request to the Agent server on this computer, server send one or more Agent to the server which may accept the agent on other hosts, getting specific information back to the original host, providing appropriate treatment.

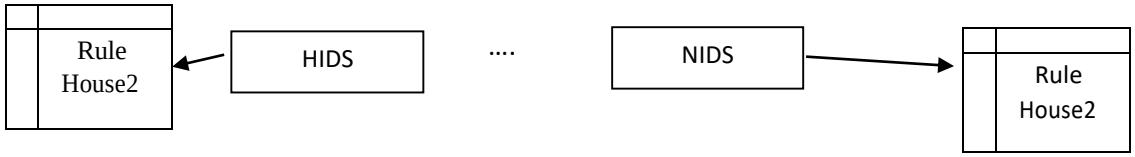
- Warning Device Agent (WDA）
  For information of intrusion detection Agent sent to, and then under response policy to make
  instant response, MAIL alarm, voice alarm, disconnected TCP session, break the process, and so

on.

c. Design A NIDS and HIDS hybrid intrusion detection system model

Many IDS tend to use detection methods of network-based or host-based detection methods to achieve security. However, the inadequate of HIDS and NIDS: in the face of complex network environments, NIDS shortcomings is that monitoring of the amount of data is too large and cannot be combined with the operating system features to accurately judge your network behavior. In addition, NIDS could not be collected in real time all data packets. HIDS is limited by its deployment environment, only support a single host security, lack of cross-platform, poor portability, so limited the scope of application and intrusions to the network protocol can do about it.





Control Center is mainly used to determine whether a sophisticated intrusion attacks, master system and other network communications to exchange information, update rules, assignment rules information database, make the appropriate alert. It includes a master system （MS) and based-on

the master system (BMS）, master is the core of the system, the second control system is a backup of

the master system, preventing the master control center attack when not working, replaced the main

control system completes the task.

Communication module is done through the SOCKET mechanism to control center and HIDS (NIDS) for information exchange. Communication components should have basic functionality: message delivery, receives message and message.

Rule: (1) if detected on packets with attack signatures, stored it in database, systems analysis, statistical sampling or statistical analysis modules; (2) storage rules of characteristics of intrusion or invasion as intrusion detection data analysis model if there are exceptions based on the package.

User management model is to pass identity authentication, and then manage the entire system, including rule base to upgrade and improve, the system log management, analysis and other functions.

d. Comprehensive analysis

Taking into account the increase of real-time intrusion detection, avoid overloading the network communication and other issues such as improving the efficiency of the system, introduced Agent technology in the network and host IDS, using static Agent for data collection, intrusion analysis, response, etc. Mobile Agent can be run across the network, cross-platform, it can be implemented using the Mobile Agent intrusion tracing of certain functions, search for the trace of the attackers, and forensics, make up for the traditional IDS systems platform of insurmountable and so on.

Intrusion detection system, based on "static Agent, Mobile Agent supplement" principle, that is, you can use the static Agent, Mobile Agent is not used as much as possible, to

reduce the complexity of the system.
Host-based IDS and network-based IDS, however there is still a lot of shortcomings. Such as HIDS apart from take up valuable resources, monitoring only the host itself outside of the host, there is no detection of conditions on the network; NIDS will appear in a switched Ethernet environment detection range limits, are not suitable for handling encrypted sessions, and poor accuracy of detecting intrusion, is difficult to configure in a switched network environment, prevention of intrusion deception is relatively poor.
Intrusion detection system using a hybrid-type system whose master of intrusion detection system in general further judgment, give full play to host based IDS and network-based IDS of both advantages, make up for the lack of both, to do both detection of network attack information, can also be found in log an exception from the system.
Comparison network-based intrusion detection system (NIDS) or host-based intrusion detection system (HIDS) with hybrid intrusion detection system (Mixed Intrusion Detection System, MIDS). Because of the MIDS data sources extensively, with good communication, and the control center can detect more complex attacks, so the MIDS testing more comprehensive. However, from the detection time, detection of MIDS need a longer time.

6. OPEN SOURCE INTRUSION DETECTION TOOLS

There are many open source IDS tools are available in open space, but in this paper our analysis is restricted to two popular NIDS tools Snort and Bro & four HIDS tools OSSEC, Tripwire, AID and Samhain.

a. SNORT

Snort is an open and secure network security system (IDS / IPS) that incorporates signature benefits, compliance, and anomaly-based testing. Snort was originally written by Martin Roesch's integrated business models with the aim of building the resources and trading services offered by Sourcefire which was acquired by Cisco. Snort can be configured in three different ways namely inline, tap (passive) and inline-test. When Snort is in Inline mode, it acts as an IPS that allows drag rules to create, in Passive mode, it works as IDS i.e. Drop rules are not loaded and in Inline-Test mode mimics the inline line mode, allowing inline behavior testing without affecting traffic. Pull rules will be loaded and will be considered as a Wdrop warning (Drop Down). Snort is able to perform real-time traffic analysis, packet infiltration, alert and block in IP networks. Perform protocol analysis, content search, and content matching. Snort can be used to detect probes or attacks, attempts to install system fingerprints, standard gate attacks (CGI), buffer overload, server message block (SMB) probes, and port stealth scans.
Snort Architecture
Snort architecture consists of mainly 7 modules

i. Packet Capture Module: This module gathers packets from network adapter. It is based on the libpcap library for Unix like systems and for windows systems WinPacap is used.

ii. Decoder: Decoder fits the captured packets into data structures and identifies link level protocols. Then, it takes the next level, decodes IP, and then TCP or UDP in order to get useful information like ports and addresses. Snort will alert if it finds malformed headers (unusual length TCP options, etc.)

iii. Preprocessors: Preprocessors can be treated as filters, which identifies things such as suspicious connection attempts to some TCP/UDP ports or too many TCP SYN packets sent in a short period of time (port scan). Preprocessors function is to take packets potentially dangerous for the detection engine to try to find known patterns. Preprocessors can alert on, classify, or drop a packet before sending it to detection engine

iv. Detection Engine: Detection Engine making use of the detection plug-ins, it matches packets against rules loaded into memory during Snort initialization.

v. Rules Files: Rules are plain text files which contain a list of rules with a syntax. This syntax includes protocols, addresses, output plug-ins associated and some other things.

vi. Detection Plug-ins: These are modules referenced from its definition in the rules files. They are used to identify patterns whenever a rule is evaluated.

vii. Output Plug-ins: These are the modules which allow formatting the notifications (alerts, logs) for the user to access them in many ways (console, extern files, databases, etc.).

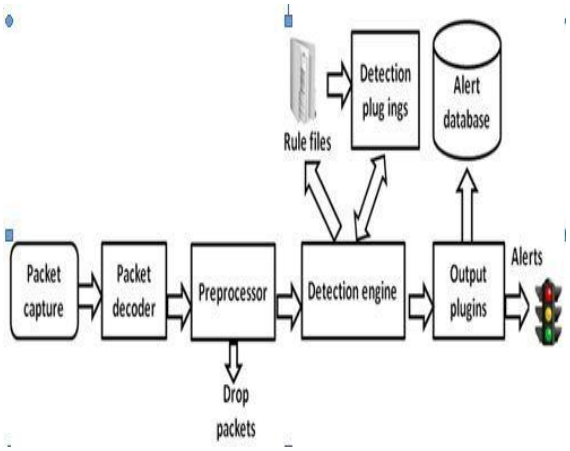


Fig. 2. Architecture of Snort

Some of the advantages of Snort are:

- Snort can easily be deployed on any node of a network, with minimal disruption to operations
- Snort provides well-documented and tested set of signatures. There are 22059 signatures available under Snort registered users on “snortrules-snapshot-2953.tar.gz “as on 26- Sep-2013.
- Portable (Linux, Windows, MacOS X, Solaris, BSD, IRIX, Tru64, HP-UX, etc.).

- Snort can act as IPS by configuring inline mode thus tends to drop packets whenever the rules trigger.

Even though snort is most popular open source IDS/IPS it has the following disadvantages

- Information overload, rules database is very large. For example, http packet in Snort having more than 1000 signatures, thus enormous processing is required to match the packets.
- Monitoring packets in large network is an expensive task.
- Fails to detect fragmented packets at high speed networks (> 5Gbps)

b. BRO.

Bro was originally written by Vern Paxson at Lawrence Berkeley National Lab and the International Computer Science Institute. Bro is a passive, open-source and unix based Network Intrusion Detection System (NIDS) that monitors network traffic looking for suspicious activity. Bro detects intrusions by first parsing network traffic to extract is application-level semantics and then executing event oriented analyzers that compare the activity with patterns deemed troublesome. Bro has gained its reputation due to its Stateful Protocol Analysis capabilities. Bro has its own specialized policy language and if Bro detects something of interest, it can be instructed to either generate a log entry, alert the operator in real-time, execute an operating system command (e.g., to terminate a connection or block a malicious host on-the-fly). In addition, Bro's detailed log files can be particularly useful for forensics. Bro is aimed to target high- speed (Gbps), high- volume intrusion detection. Making use of packet-filtering techniques, Bro is able to achieve the necessary performance while running on commercially available PC hardware, and thus can serve as a cost-effective means of monitoring a site's Internet connection.

Bro-ids Architecture
Bro IDS architecture consists of mainly 5 modules.

i. Packet Capture: Bro captures traffic using libpcap. Packets filtered by Bro-IDS are based on ports and bits in IP or TCP headers. For examples, 13th bit of TCP header indicates whether it is set with SYN, FIN, RST or nothing. This information is important to keep the status of TCP connections states.

ii. Event Engine: This layer performs several integrity checks to assure that the packet headers are well-formed. For example, it verifies the IP header checksum is correct. At this point Bro reassembles IP fragments so that network layer analyzer can accent to complete IP datagrams. It sends events to the Policy layer.

iii. Signature Engine: Signature Engine inspects the packet stream, and generates an

event each time a signature is matched. Those events can then be analyzed by a policy script.

iv. Policy Layer: The policy script interpreter executes scripts written in a specialized Bro language. These scripts specify event handlers the happenings received for the Event

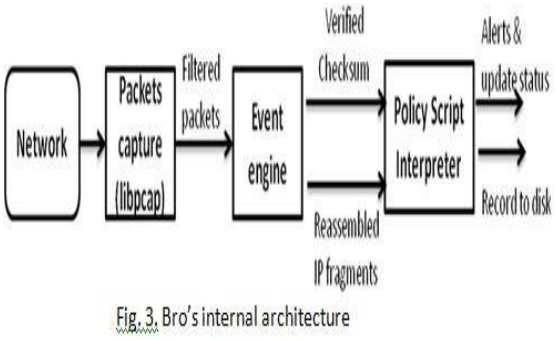


Fig. 3. Bro's internal architecture

Advantages of Bro Network Intrusion Detection System includes following:

- Bro reassembles the packet stream prior to reaching the event engine. Reassembling at this level implies that Bro can detect, not only attacks hidden by natural TCP segmentation, but also an important type of subterfuge attacks.

- Bro-ids is capable to perform application level deep packet inspection. Bro-ids analysis file contents exchanged over application-layer protocols including MD5/SHA1 computation for fingerprinting

- Bro is capable in doing Tunnel detection and analysis (including Ayiya, Teredo, GTPv1). Bro decapsulates the tunnels and then proceeds to analyze their content as if no tunnel was in place.

- Improved forensic capabilities with the support of Time Machine, a high-performance packet bulk recorder with a Bro interface.

Some of the drawbacks of Bro-IDS are as follows:

- Bro requires a UNIX platform. Bro-ids support only Linux, FreeBSD, and Mac OS

- Bro-ids only reports information to log files and do not have a graphical user interface (GUI) supported by Bro Project. (Brownian is a web interface for viewing and interacting with Bro IDS logs provided by GitHub Enterprises).

c. OSSEC

OSSEC is an Open Source Host-based Intrusion Detection System that performs log analysis, file integrity checking, Windows registry monitoring, unix-based rootkit detection, real-time alerting and active response. It runs on most operating systems, including Linux, MacOS, Solaris, OpenBSD, FreeBSD, HP-UX, AIX and Windows

The OSSEC project was founded by Daniel Cid, it was made public in 2004. In 2008, 'Third Brigade' acquired the OSSEC project, which was then acquired by 'Trend Micro' in 2009 continuing OSSEC as an open source and free.

The OSSEC HIDS can be installed as a stand-alone tool to monitor one host or can be deployed in a multi-host scenario, one installation being the server and the others as agents. The server and agents communicate securely using encryption. OSSEC also has intrusion prevention features, being able to react to specific events or set of events by using commands and active responses. Communication occurs on UDP port 1514 and messages are compressed using zlib and are encrypted using the symmetric key Blowfish algorithm.

OSSEC consists of a main application, a Windows agent, and a web interface. Main Application is required for distributed network or stand-alone installations. It is supported by Linux, Solaris, BSD, and Mac environments. Windows Agent is provided for Microsoft Windows environments. The main application needs to be installed and configured for server mode to support the Windows Agent. Web Interface provides a graphical user interface.

OSSEC Architecture

OSSEC is composed of multiple sections. It has a central manager for monitoring and receiving information from agents, syslog, databases and from agentless devices. It stores the file integrity checking databases, the logs, events and system auditing entries. OSSEC Agent is a small program or collection of programs installed on the systems which are need to be monitor. The agent will collect information in real time and forward it to the manager for analysis and correlation.

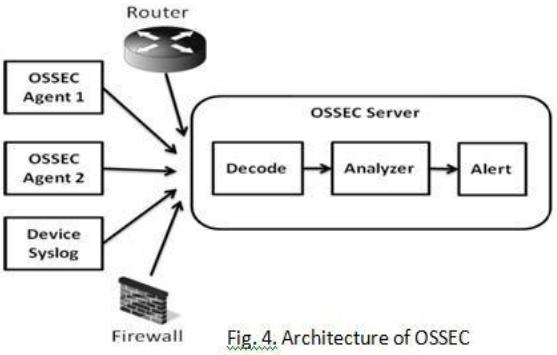


Fig. 4. Architecture of OSSEC

• Analyze logs from multiple devices and formats. The devices can be Agents, Syslog devices, Routers, Switches, Printers, etc.,

• An active response system. This means OSSEC will not only monitor, but also respond to threats (ex. black list naughty IP addresses)

Some of the disadvantages of OSSEC:

• Difficulty in upgrades between versions. OSSEC comes with default rules and they are overwritten on every upgrade.

• Coordinating pre-shared keys can be problematic. In OSSEC architecture Client and server communicate through encrypted channel using blowfish algorithm. Here pre-sharing keys before the communication establishment is a challenging issue.

d. TRIPWIRE.

Open Source Tripwire is a Host Based Intrusion Detection System for monitoring and alerting on specific file change(s) on a range of systems. The project is based on code originally contributed by Tripwire, Inc. in 2000. The first version of Tripwire was written by Gene Kim and Dr. Eugene Spafford at Purdue University in 1992 and released to the open source community Tripwire monitors Linux system to detect and report any unauthorized changes to the files and directories. Once a baseline is created, tripwire monitors and detects, which file is added/modified, what are the changes and access/modified timestamp details. Cryptographic hashes are employed to detect changes in a file without storing the entire contents of the file in the database. While

useful for detecting intrusions after the event, it can also serve many other purposes, such as integrity assurance, change management, and policy compliance.

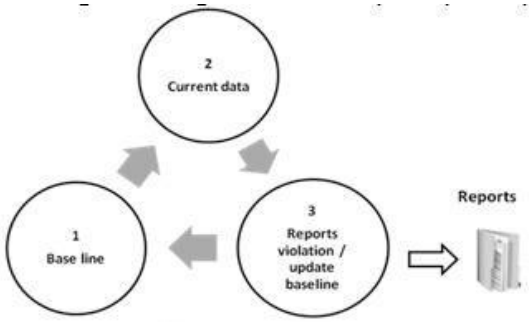


Fig. 5 Framework of Tripwire

Advantage of Tripwire:

• Advantage of tripwire is that it encrypts its database and config file.

Some of the disadvantages of Tripwire:

• Tripwire does not generate real-time alerts upon an intrusion.

• Tripwire will not detect any bugs that were already exists in the system. Tripwire should be installed right after the OS installation, and before system connected to a network for better performance.

• Open source Tripwire is suitable for monitoring a small number of Linux servers, where centralized control and reporting is not essential.

e. AIDE.

AIDE is a host-based IDS, which scans the file system and logs the attributes of important files, directories, and devices. Each time it runs, it compares its findings against the previous, "known good" data, and alerts you if something has changes. AIDE was originally written by Rami Lehti and Pablo Virolainen in 1999. Between 2003 and 2010 it was maintained by Richard van den Berg. In October 2010 Hannes von Haugwitz took over the project.

IDE supports multiple hash algorithms with which it can generate checksums for each file. AIDE takes a "snapshot" of the state of the system, register hashes, modification times, and other data regarding the files defined by the administrator. This "snapshot" is used to build a database that is saved and may be stored on an external device for safekeeping. AIDE is used on many Unix-like systems for file integrity checking and rootkit detection.

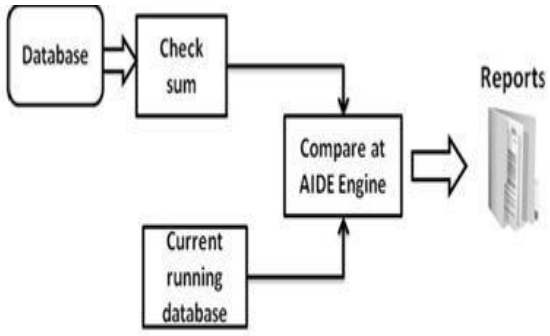


Fig. 6. Framework of AIDE

Advantages of AIDE:

• Constructs a database of directories specified in configuration file to verify the integrity of the files

• Creates a cryptographic checksum of each file and supports several message digest algorithms By default, the list includes MD5, SHA-1, SHA-256, SHA-512, WHIRLPOOL RMD-160, Tiger, HAVAL, and CRC-32

• Automatically generate a daily report. The report is mailed to root and is in /var/mail/root

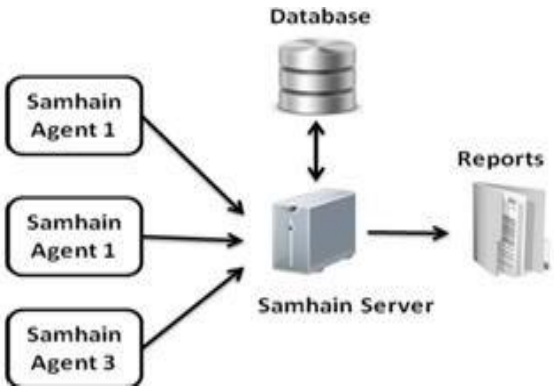
Disadvantages of AIDE:

• AIDE report changes after the incident. It does not prevent file from being altered.

• AIDE does not encrypt and sign the baseline database by default. Hence one has to sign the database for greater security or the newly created database should be moved to a secure location such as read-only media, otherwise an attacker could read and modify the configuration file.

f. SAMHAIN.

The Samhain host-based intrusion detection system (HIDS) provides file integrity checking and log file monitoring/ analysis, rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes. Samhain been designed to monitor multiple hosts with potentially different operating systems, providing centralized logging and maintenance, although it can also be used as standalone application on a single host. Samhain can run on platforms like Unix, Linux, Cygwin/Windows (crywin/ windows supports Samhain monitoring agent only as on Oct 2013). Samhain uses cryptographic checksums of files to detect modifications. Samhain can run continuously as a daemon (background process), and any stop/restart process will leave a recognizable mark. Thus it is capable to find rogue SUID executables anywhere on disk as long as the daemon is running. Samhain can also monitor which ports are open on the local host, and compare against a list of allowed or required port/services. Samhain equipped with a central log server. Messages are sent via encrypted TCP connections. Clients need to authenticate to the server. Database and configuration files can be signed, log file entries and e-mail reports are signed and support for stealth operation.



Advantages of Samhain:

• Samhain support for logging to a central server via encrypted and authenticated connections, signing of database and configuration files is an added advantage.

• Samhain can perform incremental checks on growing log files i.e. verify at each check that the data present at the preceding check have not been modified, which is one of the main requirement by Sect. 10.5.5 of the PCI DSS. (Payment Card Industry (PCI) Data Security Standard (DSS).

7. COMPARISON OF OPEN SOURCE INTRUSION DETECTION TOOLS

This Project discussed about Network Intrusion Detection System Tools Snort and Bro & Host Intrusion Detection System tools OSSEC, Tripwire, AIDE and Samhain. While choosing a Network Intrusion Detection System, Snort is one of the best lightweight IDS/IPS which can run on many operating systems. Snort can easily be deployed on any node of a network, with minimal disruption to operations. It has very high speed networks. On the other side Bro is suitable for those who are working with high speed network. Bro is flexible and highly customizable, but Bro-ids is not suitable for those who are working with windows environment. With the help of the script, snort2bro, Snort signatures can be converted automatically into Bro's signature syntax. However, one can't benefit from the additional capabilities that Bro provides as the approaches of the two systems are just too different. Bro organization is now stopped maintaining the snort2bro script, and there are now many newer Snort options which it doesn't support and now the snort2bro script is now no longer part of the Bro distribution. The comparison of these two Network Intrusion Detection tools is shown at Table I. Open source HIDS tool Samhain can perform incremental checks on growing log files, this feature is not available on OSSEC, AIDE and Tripwire, and moreover Samhain can also monitor which ports are open on a particular localhost. Tripwire and Samhain are able to encrypt and sign the database whereas AIDE cannot. OSSEC performs analysis on the server side, which means that the server can become a performance bottleneck. Hence OSSEC might show degrade performance when numbers of agents increases. Samhain does this analysis on the client side, and agents forward reports based on policy violations to the server. This minimizes both the network load and the computational load on the server. Table II gives the brief comparisons of Host based Intrusion Detection tools.

TABLE I

COMPARISON OF OPEN SOURCE NIDS TOOLS

Tools / Features	SNORT	BRO
Supported Platforms	Unix like systems , Windows, MacOS X,etc	Unix like systems, and Mac OS
License	GNU GPL v.2	BSD license
PGP Signed	●	●
IPS feature	●	x
Support to High speed Network	Medium	High
Probe attacks	●	●
buffer overflows	●	●
SQL injection	●	●
Web application attacks	●	●
DOS attacks	●	●

TABLE II

COMPARISON OF OPEN SOURCE HIDS TOOLS

Tools / Features	OSSEC	Tripwire	AIDE	Samhain
Supported Platforms	Unix like systems and Windows	Linux, all POSIX/UNIX Sys.	Unix like systems	Unix like systems and Windows
License	GNU GPL v.2	GNU GPL	GNU GPL	GNU GPL
PGP Signed	●	x	●	●
IPS feature	●	x	x	x
File integrity checking	●	●	●	●
Windows registry monitoring	●	x	x	●
Rootkit detection	●	●	●	●

8. A Denial of Service Resistant Intrusion Detection Architecture

While not yet seen in the wild, intrusion detection systems (IDSs) may become a primary target for attackers. As IDSs become more widely deployed, and as their automated response capability increases, attackers may find it necessary to disable an organization's IDS before carrying out their nefarious activity. Furthermore, such attacks may be easy to launch against the new breed of distributed IDSs with hierarchical, interdependent components. In such systems, IDSs often have command and control or analysis components that, if disabled, render the IDS useless over a large portion of the protected network. We call these critical components and the host on which they reside critical hosts.

To counter the threat of attackers finding and disabling critical IDS components, IDS vendors devote many resources towards designing intrusion detection systems such that they cannot be penetrated. For the most part these efforts have been successful and stand-alone intrusion detection hosts have shown a strong resistance to penetration attacks. Less effort has been spent securing IDSs from flooding denial of service (DOS) attacks. In part, this is because it is widely thought that nothing can be done about this type of attack.

However, it is a common misconception that systems in general cannot avoid the consequences of a flooding DOS attack. To demonstrate this, our paper describes an IDS architecture that enables IDSs to become resistant to flooding DOS attacks. Furthermore, the model does not require IDSs to operate in an ineffective or obscure manner, but rather builds upon the traditional distributed hierarchical model used by the majority of IDSs today.

Our IDS architecture resists flooding DOS attacks using a combination of techniques. First, critical IDS components are made invisible to an attacker's normal means of "seeing" in a network: active network scanning and passive packet monitoring. Second, critical IDS

components are made adaptive to flooding DOS attacks in that they can automatically relocate to another host in the event of an attack. The relocation is invisible to the attacker who then cannot persist in the attack. Our model does not prevent an attacker from launching attacks but instead makes the important targets invisible which forces the attacker to fire blindly. Random attacks may actually hit a critical IDS host, but the ability of IDS components to move between hosts in the event of an attack mitigates the damage.

a. Vulnerable IDS Architectures

We envision distributed IDSs of the future consisting of thousands of small event gathering agents reporting to hundreds of event analyzers that are all part of a unified

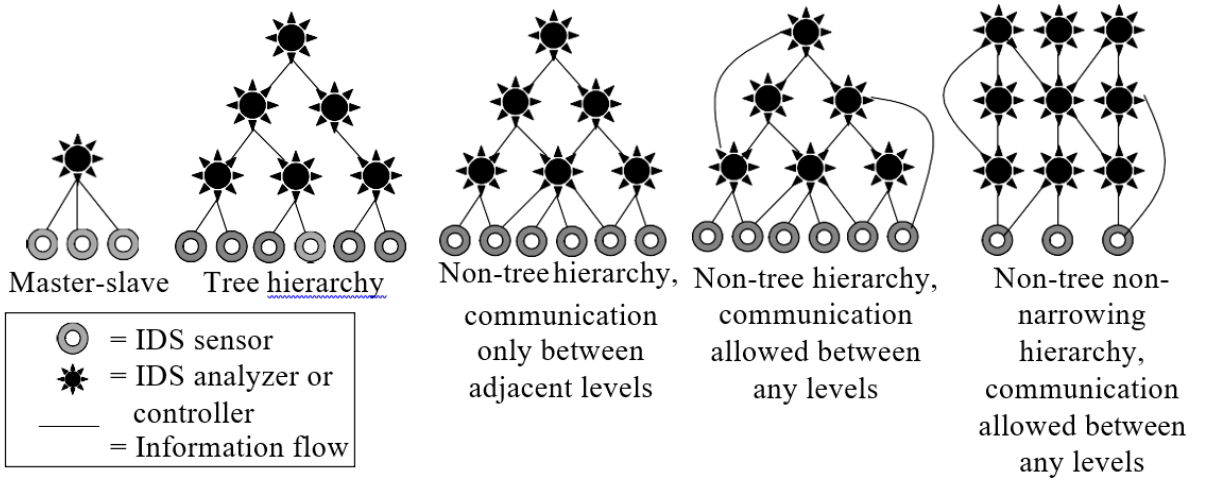


Figure 2.1: Various types of distributed hierarchical IDS architectures

system with centralized reporting and control. Each of these agents may be configured to detect a single event, or they may detect several events. Each network component may host one or many agents. Since there will be a large number of event generators, for performance reasons the events must be abstracted, analyzed, and condensed by a hierarchy of agents before reaching the centralized reporting and control facility. Various types of hierarchies are shown in Figure 2.1. While not shown in the figure, event information flow in the hierarchy is usually up while command and control flow is usually down. Also, the number of IDS components higher up the hierarchy is usually, but not always, smaller than the number of components at the bottom of a hierarchy since event information is abstracted and condensed as it moves higher. The physical location of event generators will be fixed since they monitor stationary resources. However, the internal nodes of the hierarchy may exist at many locations in the network since they receive their input and give their output via network connections.

A hierarchical architecture is already used today by many distributed IDSs that have reached the scalability limits of using a single analyzer and control component for all event generators in a network. Examples of IDSs using a hierarchical structure for their component communication model include: UC Davis's GrIDS, Lawrence Livermore's SPI-NET, Cisco's NetRanger, Axent's Intruder Alert, Internet Security System's RealSecure, Network Associates Incorporated's Active Security, and Purdue's AAFID.

While hierarchical IDS architectures provide many organizational and scalability benefits, they have a weakness that we must compensate for when designing them. The weakness is that

hierarchical IDSs are prone to have single points of failure that are easily discoverable by an attacker. If an attacker can disrupt such a failure point, a large portion of the networks IDS becomes inoperable and the attacker can then exploit weaknesses in an organization without fear of detection or retribution.

b. Methods to Target IDS Hosts

Before an attacker can disrupt an internal IDS component, he or she must find the host on which the component is residing. Unless the attacker is part of an organization's network management or security groups, the attacker has a limited ability to "see" hosts in a network. The two primary ways for a hacker to see are: sniffing and probing. Passive network sniffing is where an attacker listens to the network traffic passing by a

host on which the attacker has control. Active probing is where a hacker maps out the hosts in a network by sending out packets to the IP addresses owned by an organization. Active probing can reveal hosts that are alive, the operating systems they are running, the server applications running on those operating systems, and even the version numbers of server software. The most popular software package for active probing is Nmap.

c. The Susceptibility of Hosts to Denial of Service Attacks

While difficult, it is possible to secure a host against penetration attacks by carefully designing the software that runs on the host. Using the same technique, it is also possible to secure a host against DOS attacks that take advantage of a vulnerability in order to freeze, slow down, or shut down the host. However, it is impossible to secure a host (with no outside protections) against flooding DOS attacks. An attacker who can gain network connectivity to a target host can send it more information than it can process thereby overwhelming the host's ability to respond to legitimate requests. Countering this with a faster target system only requires that an attacker gather more network resources with which to flood. Given the number of poorly managed systems in the typical network, it is not difficult for an attacker to gain control of many "non-important" hosts and coordinate attacks with these hosts using tools like Tribe Flood Network and Stacheldraht. Attacks like smurf make the task of gaining more flooding resources easier, and allow an attacker to amplify his flooding resources a hundred or even a thousand-fold. It may not also be practical to defend all critical resources by installing extremely fast computers and even if an organization went to this extreme, a flooding DOS attack could fill the network pipe that is feeding traffic to the target. In this case, the target continues to function but cannot process any legitimate network requests because the attacker has consumed all of the network bandwidth.

d. Example Attack Scenario

We fear that hackers may discover the location of important IDS components and then launch flooding DOS attacks to freeze or shut them down. A hacker might distribute reconnaissance code throughout an organization using targeted viruses or worms as shown in Figure 5.1. This is simple to accomplish since virus detection systems only detect previously recorded viruses.

Alternately, if the hacker is an insider he may already have access to a large number of systems and can manually place the reconnaissance code.

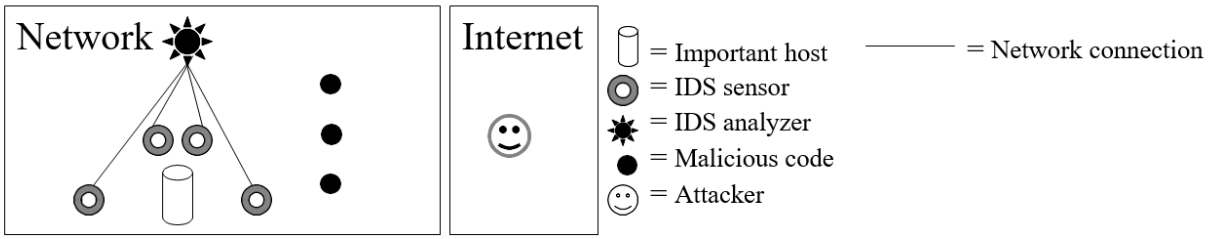


Figure 5.1: Attacker sneaks malicious code into the network

As shown in Figure 5.2, the malicious code can gather information about the network by actively scanning the internal hosts and by passively monitoring network traffic. Upon determining the location of the critical IDS components and suitable targets, the malicious code opens a covert, channel back to the attacker. Even if an organization became aware of the reconnaissance code, by the time a response was initiated the hacker would have gained a view of the organization's internal IDS topology. Someday there may exist a black market to sell such network topology information.

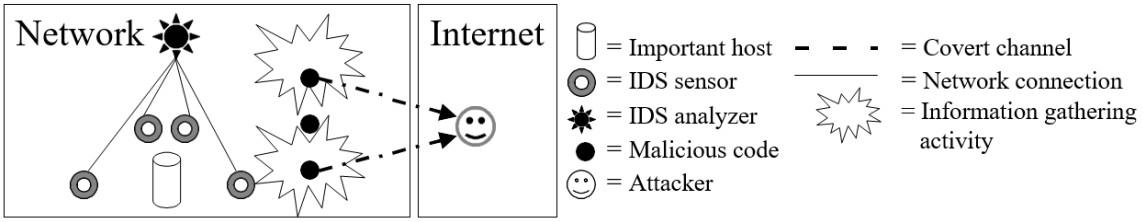


Figure 5.2: Malicious code gathers and reports information about the location of IDSs and important servers to the attacker

Upon discovery of this IDS topology, a hacker would like to penetrate and control the distributed IDS. However, key IDS components are likely to be well maintained and difficult to penetrate.

Instead, as shown in Figure 5.3, each instance of the malicious program can launch a flooding DOS attack against critical IDS components. An organization's IDS, without the critical aggregation, analysis, reporting, and command elements cannot effectively detect and respond to attacks. During this IDS downtime, the hacker, other instances of the malicious code, and possible others can launch attacks and probes that penetrate, steal information from, and install back doors in important systems.

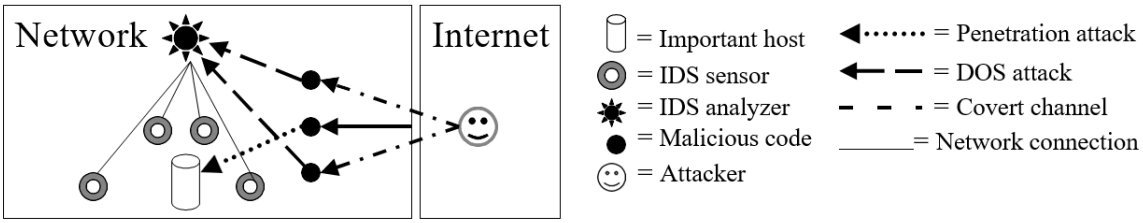


Figure 5.3: Attacker instructs the malicious code to disable the IDS and to penetrate an important host.

e. Existing Solutions

There are no known ways to prevent flooding DOS attacks against hosts that are visible on the Internet. One can purchase an ever faster set of servers and network connections, but an attacker with enough resources can flood those publicly accessible resources. Because of this, many assume that flooding DOS attacks in general cannot be countered. As a result, little research is done to explore the different ways these attacks can be mitigated or prevented.

However, applications that do not have to be publicly visible on the Internet, like IDSs, can defend themselves against flooding DOS attacks. There exist several solutions for IDSs that offer different levels of protection and require the use of different levels of resources.

Active IDS Response

Some IDSs have the ability to add or change filtering rules in routers and firewalls. With this capability an IDS sensor can detect a flooding DOS attack and quickly reconfigure the appropriate routers to stop the attack. Such an action can effectively stop a flooding DOS attack, however, the solution has some drawbacks:

i. The IDS must have control of most of the routers in an organization in order to initiate a response that can stop any possible attacking host from flooding any important IDS host.
ii. Initiating filtering on routers decreases their performance and many organizations will desire solutions that do not reduce their network bandwidth.
iii. Network operations personnel may be unwilling to relinquish control of all network routers to security operations personnel.
iv. The fear that such automatically generated filtering responses could harm legitimate traffic may make this solution unappealing.
v. Unless filtering is enabled on most routers prior to any attack, an attacker may be

able to launch a flooding DOS attack against the IDS response host, that contacts the routers to initiate filtering, thereby disabling the defensive system.

vi. IDSs often take time to detect and respond to attacks (sometimes as much as several minutes). Attackers can use this window to temporarily disable the IDS with flooding DOS attacks while they penetrate their real targets.

Separate Communication Channels

Another solution is to have separate physical wires on which IDSs can communicate. This solution is effective (assuming that IDS nodes are not penetrated), but it is too costly for most organizations to install a separate network just for IDS systems.

Decentralized Non-hierarchical IDSs

Some researchers have investigated creating IDSs that are not hierarchical and whose IDS components are not interdependent. While such an IDS could be very resistant to many DOS attacks, it has proven difficult to build such decentralized distributed IDSs. This area is actively researched, and viable prototypes may emerge.

Mobile Recoverable Components

Another solution is to make it difficult for an attacker to disable critical IDS components by wrapping the critical IDS components as mobile agents. In this scheme, the mobile agent critical components randomly move around a network so that an attacker does not know where they reside. If an attacker does manage to find and attack a critical component, other mobile agents immediately take over the functionality of the destroyed agent. Furthermore, the architecture is designed such that there exists no single points of failure that an attacker could use to take down the mobile agent system. While this scheme may be useful, it fails to take advantage of domain specific knowledge about IDSs and assumptions that can be made about network topologies. This paper builds upon the concepts presented in this work and attempts to build a stronger defensive mechanism by using IDS domain specific knowledge and assumptions about networking topologies.

9. Ossec Implementation:

Implementation requirement:

1. Virtual machine
2. Ubuntu Server
3. Libraries installation on Ubuntu
4. Ossec installation
5. Rules configuration

1. After installing the Ubuntu server, install Local Ossec.
    a. We will adjust the Ossec configuration file to check files in real time.
    b. Change the Ossec rules and the alerts level to send alert by email.
    c. Enable client syslog and sending syslog output to the server IP.
    d. Restarting the Osssec.
2. Configuring Ossec:

    Nano var/ossec/etc/ossec.conf:
    a. These changes are done to push to system to execute checks in real time and every 2 seconds
        i. Change in the system check the frequency and set it to 2
        ii. add to directories “realtime=”yes””
        iii. add to directories “report_changes=yes”
    b. after sys check we should configure the syslog output option to send syslog messages to a syslog server
        i. add: <syslog_output> <server> 127.0.0.1</server> </syslog_output>
        ii. /var/ossec/bin/ossec-control enable client-syslog
        iii. Tail –n 1000 /var/ossec/logs/ossec.log | grep csyslog
    c. To be able to show all the alerts levels
        i. Change in the <alert>: <email_alert_level>1</email_alert_level> /
        ii. do the same for <log_alert_level>
3. Adding new rule:
    a. We added a new rule to the var/ossec/rules/local_rules.xml file by overwriting

```
<rule id=”554” level=”7” overwrite=”yes”>
<category>ossec</category>
<decoded_as>syscheck_new_entry</decode_as>
<description>File added to system</description>
<group>syscheck,</group>
</rule>
```

4. Rules Configuration:
    a. We added email alerts to the rules that we want to detect and audit as email not in the general log file
        i. We add alerts to the var/ossec/rules/syslog_rules.xml file with the following xml block

```
<rule id = “2504” level =”3”>
<match> ILLEGAL ROOT LOGIN | ROOT LOGIN REFUSED</match>

<options> alert_by_email </option>

<description> ILLEGAL ROOT LOGIN</description>
```

<group>invalid_login,</group>

</rule>

We added alerts to many rules using the same steps

Rules (by id):
1002/2501/2502/2504/5103/5304/5305/5902/5903/5401/5402/5403/5404/5405/10100

5. Testing:
    a. User Authentication alerts:
        i. Login log;
        ii. Accessing server from an unknown user

    b. Accessing a restricted file alert:

    c. Accessing the root user alert

    d. Users alerts Creating and deleting a user

6. Logs After Triggering Alerts:

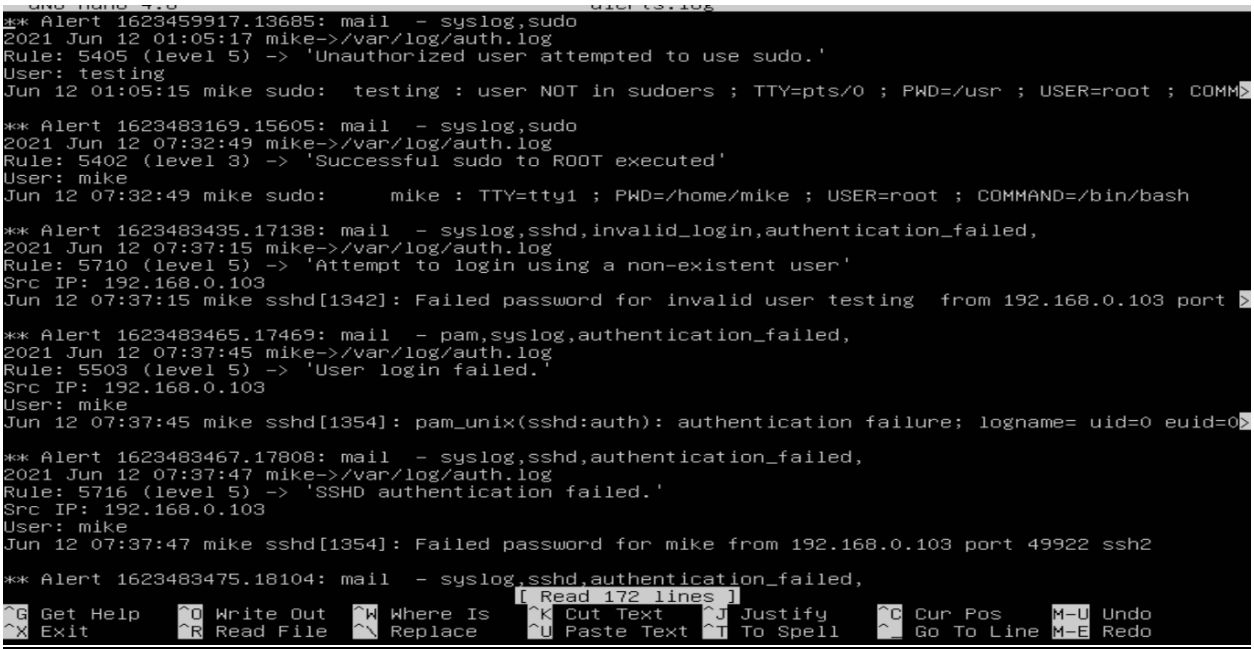
```
 --END OF NOTIFICATION


New mail has arrived.
?
Return-Path: <ossecm@mike>
Received: from notify.ossec.net (localhost [127.0.0.1])
        by mike (8.15.2/8.15.2/Debian-18) with SMTP id 15BLUQdb003707
        for <root@localhost>; Fri, 11 Jun 2021 21:30:26 GMT
Message-Id: <202106112130.15BLUQdb003707@mike>
To: <root@mike>
From: OSSEC HIDS <ossecm@mike>
Date: Fri, 11 Jun 2021 21:30:26 +0000
Subject: OSSEC Notification - mike - Alert level 3


OSSEC HIDS Notification.
2021 Jun 11 21:30:12

Received From: mike->/var/log/auth.log
Rule: 5501 fired (level 3) -> "Login session opened."
Portion of the log(s):

Jun 11 21:30:11 mike systemd: pam_unix(systemd-user:session): session opened for user rahme by (uid=
0)



 --END OF NOTIFICATION
```

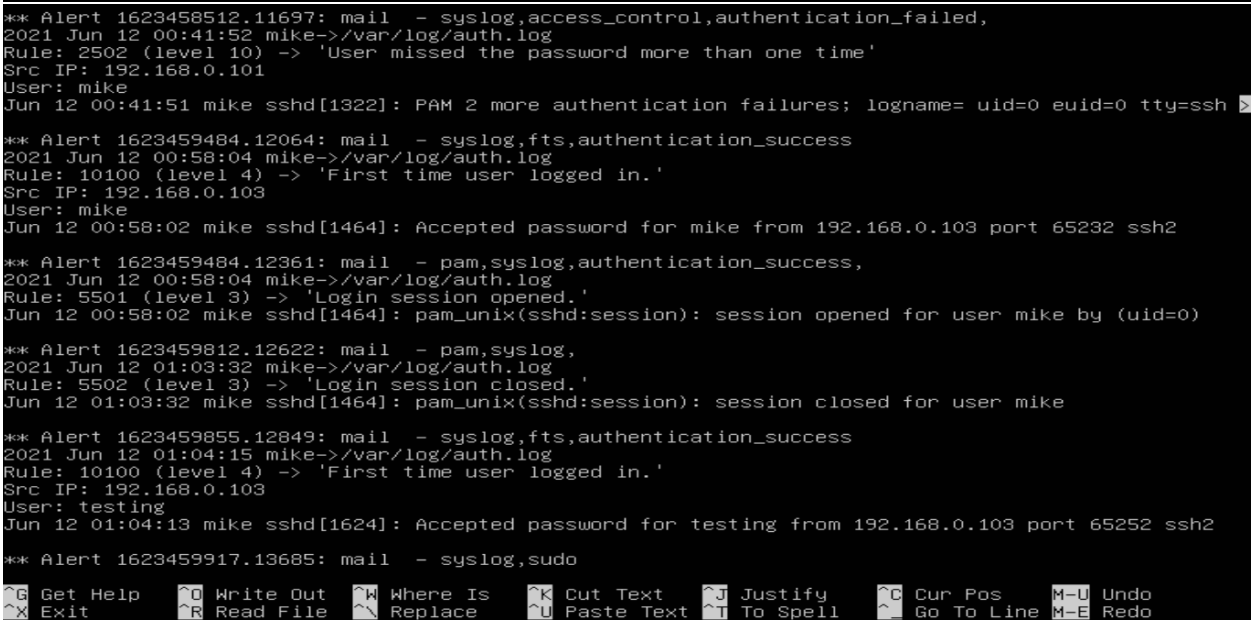
```
OSSEC HIDS Notification.
2021 Jun 12 01:05:17

Received From: mike->/var/log/auth.log
Rule: 5405 fired (level 5) -> "Unauthorized user attempted to use sudo."
User: testing
Portion of the log(s):

Jun 12 01:05:15 mike sudo:  testing : user NOT in sudoers ; TTY=pts/0 ; PWD=/usr ; USER=root ; COMMA
ND=/bin/bash



 --END OF NOTIFICATION

?
Return-Path: <testing@mike>
Received: from mike (localhost [127.0.0.1])
        by mike (8.15.2/8.15.2/Debian-18) with ESMTP id 15C16F0u002091
        for <root@mike>; Sat, 12 Jun 2021 01:06:15 GMT
Received: (from testing@localhost)
        by mike (8.15.2/8.15.2/Submit) id 15C16Fpr001733;
        Sat, 12 Jun 2021 01:06:15 GMT
Date: Sat, 12 Jun 2021 01:06:15 GMT
Message-Id: <202106120106.15C16Fpr001733@mike>
To: root@mike
From: testing@mike
Auto-Submitted: auto-generated
Subject: *** SECURITY information for mike ***
Status: O
X-UID: 11

mike : Jun 12 01:05:15 : testing : user NOT in sudoers ; TTY=pts/0 ; PWD=/usr ; USER=root ; COMMAND=
/bin/bash
?
```

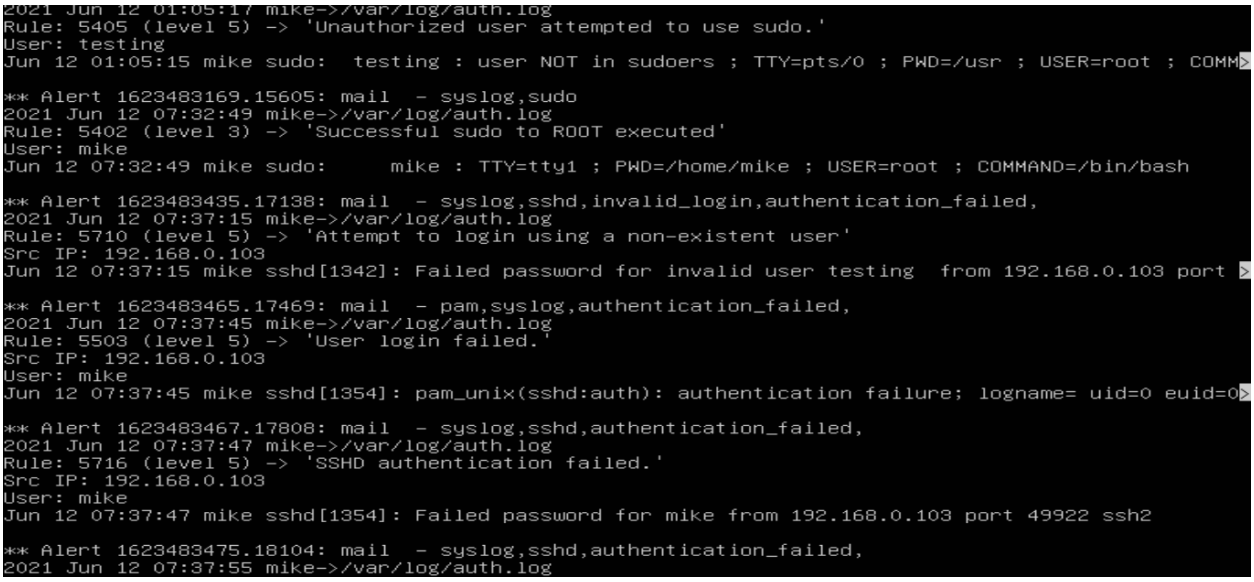
```
 U  11 OSSEC HIDS         Sat Jun 12 07:34  41/937   OSSEC Notification - mike - Alert level 7
 U  12 OSSEC HIDS         Sat Jun 12 07:37  57/1373  OSSEC Notification - mike - Alert level 5
 U  13 OSSEC HIDS         Sat Jun 12 07:37  45/1096  OSSEC Notification - mike - Alert level 5
 U  14 OSSEC HIDS         Sat Jun 12 07:38  28/727   OSSEC Notification - mike - Alert level 5
 U  15 OSSEC HIDS         Sat Jun 12 07:38  58/1327  OSSEC Notification - mike - Alert level 3
 U  16 OSSEC HIDS         Sat Jun 12 07:54  41/937   OSSEC Notification - mike - Alert level 7
? 4
Return-Path: <ossecm@mike>
Received: from notify.ossec.net (localhost [127.0.0.1])
        by mike (8.15.2/8.15.2/Debian-18) with SMTP id 15C0g4Tk001364
        for <root@localhost>; Sat, 12 Jun 2021 00:42:04 GMT
Message-Id: <202106120042.15C0g4Tk001364@mike>
To: <root@mike>
From: OSSEC HIDS <ossecm@mike>
Date: Sat, 12 Jun 2021 00:42:04 +0000
Subject: OSSEC Notification - mike - Alert level 10
Status: O
X-UID: 6


OSSEC HIDS Notification.
2021 Jun 12 00:41:52

Received From: mike->/var/log/auth.log
Rule: 2502 fired (level 10) -> "User missed the password more than one time"
Src IP: 192.168.0.101
User: mike
Portion of the log(s):

Jun 12 00:41:51 mike sshd[1322]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh r
user= rhost=192.168.0.101  user=mike



 --END OF NOTIFICATION
```


```
 U  11 OSSEC HIDS         Sat Jun 12 07:34  41/937   OSSEC Notification - mike - Alert level 7
 U  12 OSSEC HIDS         Sat Jun 12 07:37  57/1373  OSSEC Notification - mike - Alert level 5
 U  13 OSSEC HIDS         Sat Jun 12 07:37  45/1096  OSSEC Notification - mike - Alert level 5
 U  14 OSSEC HIDS         Sat Jun 12 07:38  28/727   OSSEC Notification - mike - Alert level 5
 U  15 OSSEC HIDS         Sat Jun 12 07:38  58/1327  OSSEC Notification - mike - Alert level 3
 U  16 OSSEC HIDS         Sat Jun 12 07:54  41/937   OSSEC Notification - mike - Alert level 7
? 4
Return-Path: <ossecm@mike>
Received: from notify.ossec.net (localhost [127.0.0.1])
        by mike (8.15.2/8.15.2/Debian-18) with SMTP id 15C0g4Tk001364
        for <root@localhost>; Sat, 12 Jun 2021 00:42:04 GMT
Message-Id: <202106120042.15C0g4Tk001364@mike>
To: <root@mike>
From: OSSEC HIDS <ossecm@mike>
Date: Sat, 12 Jun 2021 00:42:04 +0000
Subject: OSSEC Notification - mike - Alert level 10
Status: O
X-UID: 6


OSSEC HIDS Notification.
2021 Jun 12 00:41:52

Received From: mike->/var/log/auth.log
Rule: 2502 fired (level 10) -> "User missed the password more than one time"
Src IP: 192.168.0.101
User: mike
Portion of the log(s):

Jun 12 00:41:51 mike sshd[1322]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh r
user= rhost=192.168.0.101  user=mike



 --END OF NOTIFICATION
```


```
? ~
```


** Alert 1623459917.13685: mail - syslog,sudo
2021 Jun 12 01:05:17 mike->/var/log/auth.log
Rule: 5405 (level 5) -> 'Unauthorized user attempted to use sudo.'
User: testing
Jun 12 01:05:15 mike sudo: testing : user NOT in sudoers ; TTY=pts/0 ; PWD=/usr ; USER=root ; COMM
** Alert 1623483169.15605: mail - syslog,sudo
2021 Jun 12 07:32:49 mike->/var/log/auth.log
Rule: 5402 (level 3) -> 'Successful sudo to ROOT executed'
User: mike
Jun 12 07:32:49 mike sudo: mike : TTY=tty1 ; PWD=/home/mike ; USER=root ; COMMAND=/bin/bash
** Alert 1623483435.17138: mail - syslog,sshd,invalid_login,authentication_failed,
2021 Jun 12 07:37:15 mike->/var/log/auth.log
Rule: 5710 (level 5) -> 'Attempt to login using a non-existent user'
Src IP: 192.168.0.103
Jun 12 07:37:15 mike sshd[1342]: Failed password for invalid user testing from 192.168.0.103 port
** Alert 1623483465.17469: mail - pam,syslog,authentication_failed,
2021 Jun 12 07:37:45 mike->/var/log/auth.log
Rule: 5503 (level 5) -> 'User login failed.'
Src IP: 192.168.0.103
User: mike
Jun 12 07:37:45 mike sshd[1354]: pam_unix(sshd:auth): authentication failure; logname= uid=0 euid=0
** Alert 1623483467.17808: mail - syslog,sshd,authentication_failed,
2021 Jun 12 07:37:47 mike->/var/log/auth.log
Rule: 5716 (level 5) -> 'SSHD authentication failed.'
Src IP: 192.168.0.103
User: mike
Jun 12 07:37:47 mike sshd[1354]: Failed password for mike from 192.168.0.103 port 49922 ssh2
** Alert 1623483475.18104: mail - syslog,sshd,authentication_failed,
[Read 172 lines]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo


** Alert 1623458512.11697: mail - syslog,access_control,authentication_failed,
2021 Jun 12 00:41:52 mike->/var/log/auth.log
Rule: 2502 (level 10) -> 'User missed the password more than one time'
Src IP: 192.168.0.101
User: mike
Jun 12 00:41:51 mike sshd[1322]: PAM 2 more authentication failures; logname= uid=0 euid=0 tty=ssh
** Alert 1623459484.12064: mail - syslog,fts,authentication_success
2021 Jun 12 00:58:04 mike->/var/log/auth.log
Rule: 10100 (level 4) -> 'First time user logged in.'
Src IP: 192.168.0.103
User: mike
Jun 12 00:58:02 mike sshd[1464]: Accepted password for mike from 192.168.0.103 port 65232 ssh2
** Alert 1623459484.12361: mail - pam,syslog,authentication_success,
2021 Jun 12 00:58:04 mike->/var/log/auth.log
Rule: 5501 (level 3) -> 'Login session opened.'
Jun 12 00:58:02 mike sshd[1464]: pam_unix(sshd:session): session opened for user mike by (uid=0)
** Alert 1623459812.12622: mail - pam,syslog,
2021 Jun 12 01:03:32 mike->/var/log/auth.log
Rule: 5502 (level 3) -> 'Login session closed.'
Jun 12 01:03:32 mike sshd[1464]: pam_unix(sshd:session): session closed for user mike
** Alert 1623459855.12849: mail - syslog,fts,authentication_success
2021 Jun 12 01:04:15 mike->/var/log/auth.log
Rule: 10100 (level 4) -> 'First time user logged in.'
Src IP: 192.168.0.103
User: testing
Jun 12 01:04:13 mike sshd[1624]: Accepted password for testing from 192.168.0.103 port 65252 ssh2
** Alert 1623459917.13685: mail - syslog,sudo
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste Text ^T To Spell ^_ Go To Line M-E Redo



10.References:

- W. Li. Using genetic algorithm for network intrusion detection. In Proceedings of the United States Department of Energy Cyber Security Group Training Conference, pages 1– 9, 2004.
- Darwiche, A, et. al. (2010). Bayesian networks. Communications of the ACM, 53 (12), 2010
- Biermann E, Cloete E, Venter LM (2001). A comparison of intrusion detection systems. Computers & Security. 1;20(8):676-83
- Lin WC, Ke SW, Tsai CF. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. Knowledge-based systems. 2015 Apr 30;78:13-21.
- Wang G, Hao J, Ma J, Huang L. A new approach to intrusion detection using Artificial Neural Networks and fuzzy clustering. Expert systems with applications. 2010 Sep 30;37(9):6225-32.
- Dayou Liu, Yangkun, Jianzhong Chen. Research status and development trend on Agent. Journal of software, November 2000 pp ： 315-321
- Yongzhong Li, Yunsheng Luo,Sunyan.Research based on Mobile Agent for an intelligent intrusion detection system structure. Computer research and development. June 2006 pp:296-301.
- A Denial of Service Resistant Intrusion Detection Architecture-Peter Mel,

Donald Marks, Mark McLarnon - National Institute of Standards and Technology - Computer Security Division

- D. Mudzingwa and R. Agrawal, “A study of methodologies used in intrusion detection and prevention systems (idps),” in South eastcon,2012 Proceedings of IEEE. IEEE, 2012