Intrusion Detection Correlation in Computer Network Using Multi-Agent System

by Ayman Elsayed Elsayed Taha

Ain Shams University
Faculty of Engineering
Computer and Systems Engineering Department

Intrusion Detection Correlation in Computer Network Using Multi-Agent System

A Dissertation
Submitted in Partial Fulfillment of the Requirements of the
Degree of Doctor of Philosophy in Electrical Engineering
Computer and Systems Engineering Department

Submitted by
Ayman Elsayed Elsayed Taha

M. Sc., Electrical Engineering
(Computer and Systems Engineering)
Ain Shams University, 2002

Supervised by

Prof. Dr. Hani M. K. Mahdi

Prof. Dr. Ismail Abdel Ghafar Farag

Assoc. Prof. Dr. Ayman Mohamed Bahaa

Cairo, Egypt

July, 2011

Abstract

Ayman Elsayed Elsayed Taha

Intrusion Detection Correlation in Computer Network

Using Multi-Agent System

Doctor of Philosophy Dissertation

Ain Shams University, 2011

Alert and event correlation is a process in which the alerts produced by one or more intrusion detection systems and events generated from different systems and security tools are analyzed and correlated to provide a more succinct and high-level view of occurring or attempted intrusions. Current correlation techniques improve the intrusion detection results and reduce the huge number of alerts in a summarized report, but still have some limitations such as a high false detection rate; missing alerts in a multi-step attack correlation; alert verifications are still limited; Zero Day attacks still have low rates of detection; Low and Slow attacks and Advanced Persistent Threats (APTs) cannot be detected; and some attacks have evasion techniques against IDSs. Finally, current correlation systems do not enable the integration of correlations from multiple information sources and are limited to only operate in IDS alerts. Agents and multi-agent systems have been widely used in IDSs because of their advantages.

The thesis purpose is to prove the possibility of improving both IDS Accuracy and IDS Completeness through reducing either False Positive or False Negative alerts using correlation between different available information sources in the system and network environment. The dissertation presents a modular framework for a Distributed Agent Correlation Model (DACM) for intrusion detection alerts and events in computer networks. The framework supports the integration of multiple correlation techniques and enables easy implementation of new components.

The framework introduces a multi-agent distributed model in a hierarchical organization; correlates alerts from the IDS with attack signatures from information security tools and either system or application log files as other sources of information. Correlation between multiple sources of information reduces both false negative and false positive alerts, enhancing intrusion detection accuracy and completeness. Each local agent aggregates/correlates events from its source according to a specific pattern matching. The integration of these correlation agents together forms a complete integrated correlation system.

The model has been implemented and tested using a set of datasets. Agent's proposed models and algorithms have been implemented, analyzed, and evaluated to measure detection and correlation rates and reduction of false positive and false negative alerts.

In conclusion, DACM enhances both the accuracy and completeness of intrusion detection. DACM is flexible, upgradable, and platform independent. It decreases the audit load and the time cost required to obtain effective situational understanding; increases the coverage of the attack space and forensics; and improves the ability to distinguish the serious attack from the less important ones or identify the kind of needed reaction. DACM can also be used to enhance the early detection capability of APT. Finally, DACM can be used as a real time system with minor modifications. We think that this is a promising approach successfully combining correlation techniques with agent technology in intrusion detection systems in order to provide higher security for computer networks and internet services.

Keywords:

Intrusion Detection, Alert Correlation, Multi-Agent Systems, Learning Agent, Reduction Rate

Acknowledgements

First, thanks to Allah (God) who made me able to accomplish this work, I sincerely express my deepest gratitude to my thesis supervisors, **Dr. Hani Mahdi**, Professor of Computer Engineering, Faculty of Engineering, Ain Shams University, and **Dr. Ismail Abdel Ghafar**, Professor of Computer Engineering, Military Technical College, and **Dr. Ayman Bahaa**, Associate Professor of Computer Engineering, Faculty of Engineering, Ain Shams University. I was fortunate to have met such outstanding scholar supervisors. I like to express my thankfulness for their kind supervision and offering unfailing support, invaluable advices and comments and helpful and useful discussions in selecting the interesting point and during the preparation of this thesis. I owe a special acknowledgment to them for giving me a lot of their time during the years of preparing this thesis. I could never had done it without their support, technical advice and suggestions, thorough reading of all my work.

I would like to thank the Center for Education and Research of Information Assurance and Security (CERIAS), Purdue University, USA. I appreciate the valuable support of the CERIAS executive director **Prof. Eugene Spafford**, the generous effort of his staff especially Information Assurance Research Engineer **Keith Watson**, for their cooperation during my scholar visit to the Center. They provided me with great resources to capture and collect the data needed for this work. Special Thanks to my friend Glenn Glover who guided me to that center.

I appreciate the assistance and input from my colleagues in ORC and their support during this work, special thanks to **Ahmed Abdel Sabour** and **Galal Mohamed** for their help during implementing the proposed model.

I will never be able to thank my mother and my family enough for supporting me during my whole life. I tried to accomplish this work to make them proud of me. Finally, I am very grateful to my wife **Dalia**, and my lovely two kids, **Asser** and **Sama**, for their patient support especially during my scholar visit, sacrifices, sustained moral support, and encouragement. I always thank my God for blessing me with such a wonderful family. I would like to dedicate this work specifically to them and my mother and my whole family.

Statement

This dissertation is submitted to Ain Shams University for the degree of Doctor of Philosophy in Computer and Systems Engineering Department.

The work included in this thesis was carried out by the author at Computer and Systems Engineering Department, Faculty of Engineering, Ain Shams University.

No part of this thesis has been submitted for a degree or qualification at other university or institution.

Date : 07 / 07 / 2011

Signature :

Name : Ayman Elsayed Elsayed Taha

Table of Contents

List of Figures

LIST OF TABLES

List of Algorithms

LIST OF ABBREVIATION

ABCM	: Agent Based Correlation Model
ACCL	: Active Correlation Component List
AF	: Alert Fusion
APT	: Advanced Persistent Threat
ASR	: Attack Session Reconstruction
AV	: Alert Verification
CAM	: Comprehensive Approach Model
DACM	: Distributed Agent Correlation Model
DPCM	: Dynamic Parallel Correlation Model
FTP	: File Transfer Protocol
FR	: Focus Recognition
IDS	: Intrusion Detection system
LA	: Learning Agent
LSA	: Low and Slow Attack
MAS	: Multi-Agent System
MSA	: Multi Step Attack
RR	: Reduction Rate
TR	: Threat Reconstruction
SSH	: Secure Shell

CHAPTER 1

INRODUCTION

Chapter One: Introduction

Recently, computer networks have evolved into a ubiquitous infrastructure. High speed backbones and local area networks provide the end user with huge bandwidth compared with that available a few years ago. In addition, wireless technology is bringing connectivity to a number of devices, from laptops to cell phones and PDAs, creating a complex, highly dynamic network of systems. Most notably, the internet has become a mission-critical infrastructure for governments, companies, institutions, and millions of everyday users. Because of this increased reliance on networked computers, security has become a primary concern.

1.1 Intrusion Detection and Response Systems

Intrusion detection is the process of recognizing computer system misuse. Intrusion response is the process of responding to that misuse. They are essential techniques providing an extra layer of defense when other security mechanisms fail (e.g. identification, authentication, access controls, cryptography, firewalls, and VPNs). Intrusion Detection Systems (IDSs) are software and hardware systems that automate the process of monitoring the events occurring in a computer system or network, analyzing them for signs of security problems [1].

1.1.1 IDS Terminology and Parameters

An alert or an alarm is defined as a signal reporting that a system has been, or is being, attacked. A True Positive (TP) alert is defined as the case when a real attack triggers IDS to produce an alarm; this alarm is a correct alarm. A False Positive (FP) alert is defined as the case when an event triggers IDS to produce an alarm when no attack has actually taken place; this alarm is a false alarm. A False Negative (FN) is defined as the case of IDS failing to detect an actual attack. A True Negative (TN) alert is defined as the case when no attack has taken place and no alarm is raised [1].

Both false positive and false negative alerts are the main metrics of the IDS accuracy and completeness parameters, which are calculated as follows:

Accuracy = TP / (TP+ FP) (1.1)

Completeness = TP / (TP + FN) (1.2)

For example, IDS that produces 100 alerts for 80 real attacks where other 20 attacks were missed and 40 non-attack actions were detected as attacks then this situation can be expressed as follows:

IDS Alerts: 100, True Positive: 60, False Positive: 40

False Negative: 20 alerts

Accuracy = 60 / (60+40) = 60 %

Completeness = 60 / (60+20) =60/80 = 75 %

1.1.2 IDS Limitations

IDSs have some limitations affecting their performance. First, IDSs are prone to producing a large number of alerts. Second, false positives and false negative of IDSs are inevitable. Third, IDSs can only detect single attack but not multi-step attacks, which need network security experts to analyze manually. These limitations lead to the use of alert correlation techniques [2].

1.1.3 Intrusion Detection Alerts Correlation

Alert correlation [2] is a promising intrusion detection technique that significantly improves security effectiveness by analyzing alerts from one or more IDSs and providing a high level view of the attempted intrusions. Correlation components are procedures that aggregate alerts according to certain criteria; the aggregated alerts could have common features or could represent the steps of pre-defined scenario attacks. Correlation approaches are composed of a single component or a comprehensive set of components. The Correlation process is performed through several different stages including normalization, aggregation, verification, and correlation.

The Reduction Rate (RR) [10] is the ratio between the number of output alerts after correlation and the number of input alerts:

$$RR = (1- (\text{Output alerts}/\text{Input alerts}))*100 \tag{1.3}$$

The situation where IDS produces 100 alerts as an input for correlation system, the correlation system correlate those alerts together and summarizes them to 60 alerts, in this example the reduction rate of that correlation system can be calculated as follows:

Example: Input = 100 alerts, Output = 60 correlated alerts

RR = (1- 0.6) *100 = 40 %

1.1.4 Agents in IDS

An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through response system [3]. A software agent is a computer program which works toward goals in a dynamic environment on behalf of another entity, possibly over an extended period of time, without continuous direct supervision or control [3].

Agents have been widely used in IDSs [46-49] because they can be added and removed without having to restart the IDS, thereby providing flexible scalability. Agents are capable of performing simple functions on their own; a group of agents working together are able to derive complex results by exchanging information. Use of many agents reduces system overhead and avoids single point of failure. Finally, agents provide a multi-point detection and knowledge sharing capability.

1.2 IDS Correlation Problem Definition

The major problem with the existing correlation systems is that they do not provide a complete correlation solution. Instead, some of them address only a limited part of the correlation process. For instance, multi-step correlation does not solve the problem of the high false positive rates of intrusion detection sensors. Other comprehensive correlation techniques using an integrated set of correlation components still have performance problems; giving the fact that the effectiveness of a specific correlation technique is highly dependent on the properties of the network and attacks on it, some of the correlation components may not of use with a specific environment but they still consume valuable processing time. Moreover, current correlation systems do not enable the integration of correlations from multiple information sources and are limited to operate in IDSs alerts. Furthermore, existing systems are not easily extended, and most are platform dependent.

Finally, current correlation techniques improve the intrusion detection results and reduce the huge number of alerts in a summarized report, but still have some limitations as follows:

- High false detection rate;
- Missed alerts in a multi-step attack;
- Limited alert verifications;
- Low rates of detection for Zero Day Attacks;
- Failure to detect Low and Slow Attacks and Advanced Persistent Threats; and
- Ineffectiveness against IDS evasion techniques.

1.3 The Proposed Model

This dissertation presents a model to prove the possibility of enhancing both IDS Accuracy and IDS Completeness through reducing either False Positive or False Negative alerts using correlation between different available information sources in the system and network environment

The dissertation presents a modular framework for Distributed Agent Correlation Model (DACM) for intrusion detection alerts and events in computer networks. The framework supports the integration of multiple correlation techniques, and enables easy implementation of new components. The framework introduces a multi-agent distributed model in hierarchical organization and correlates alerts from the IDS with attack signatures from other source of information (e.g. firewalls, performance monitors, FTP logs, and other access/error log files).

Correlation between multiple sources of information reduces both false negative and false positive alerts, enhancing both intrusion detection accuracy and completeness. Each local agent aggregates/correlates events from its source according to specific pattern matching; the integration of the correlation agents forms a complete integrated correlation system.

1.4 Methodology

This research went through several steps, studying the existing correlation system for intrusion detection was the first step. This study allowed us to identify the missing part and the drawback of those systems. The next step was the presented idea to solve the current correlation system problems and improve its performance. After that we collected a datasets needed to test the proposed idea. This dataset was collected on networks with a variety of services and includes real networks, networks specifically constructed for dataset gathering, and simulated networks. The dataset included real and attempted attacks. Then we implement a prototype for the proposed model to prove the theory of the idea and to assure the success of the model. The proposed prototype was constructed from a set of individual agents for each task. The next step was integrating those agents together to build the whole model. Finally, we extract the model results and perform the needed analysis for these results for purpose of assessment and presentation

1.5 Contributions

This dissertation provides solutions for the problems outlined above, and provides the following contributions:

- Enhanced accuracy and completeness of intrusion detection;
- Improved flexibility, upgradability, and platform independence;

- Decreased audit load and time cost required to obtain effective situational understanding;
- Increased coverage of the attack space and forensics;
- Improved ability to distinguish serious attacks from less important ones;
- Distinguish between attacks where an immediate response is needed from others where an alternative is acceptable; and
- Enhanced early detection capability for recent cyber attacks such as Advanced Persistent Threats (APTs) and Low and Slow Attacks.

1.6 Dissertation Organization

The remainder of this dissertation is structured as follows. Chapter 2 presents a survey of intrusion detection and related work, describing and giving an introduction to current correlation systems and distributed correlation techniques. Chapter 3 introduces a description of our distributed agent correlation model and its components. Detailed implementation of this model components and algorithms is presented in Chapter 4. In Chapter 5, detailed experimental results of applying the proposed model on the gathered dataset are presented. Finally, Chapter 6 presents conclusions and outlines future work.

CHAPTER 2

Literature Survey and Related Work

Chapter Two: Literature Survey and Related work

In this chapter, a literature survey of security and intrusion detection, alert correlation, recent cyber attacks, and correlation related work will be presented.

2.1 The Importance of Security and Intrusion Detection

While computer systems in the past usually were not networked, or were connected to a small network spanning a company or a building, today almost every computer system is connected to the Internet. The main concern with this situation is that the number of potential attackers that can attack a given system has increased drastically. Whereas before an attacker had to be physically present at the console of the computer or be connected to the same local area network as the target computer, today's attacker can be located almost anywhere in the world.

Another reason for the increased importance of computer security is that today more sensitive data are stored on computers than before. For instance medical records and bank accounts are not paper based anymore. Another change that has happened lately is that many businesses rely on computer systems to perform their function. As a result, a computer system problem can shut down the whole operation. For instance, a web-based store would not get any customers if their network connection failed.

All these changes to the ways businesses handle data have increased the number of potential targets for attacks and the effect of successful attacks have become more serious. In addition, the attackers have improved their attack techniques. It is now common to see large scale coordinated attacks where the attacker utilizes multiple computers in order to attack a single target.

These kinds of attacks can be challenging to defend against, as it is not easy to identify the attacker when he is using multiple hosts. In addition, the attacker's computers are often located on different networks. This potentially makes the aggregate network throughput available to the attacker very large and can enable the attacker to flood the victim's network with traffic, creating a denial-of-service attack.

Computer security is an increasingly important topic. It is important to insure that the secrecy of sensitive data is protected, the integrity of important data is not violated, and the availability of critical systems is guaranteed. Computer security tries to achieve all these goals.

2.1.1 Security Mechanism

Computer security offers three types of security mechanisms [66, 67] to protect a system: authentication, authorization, and auditing. These three mechanisms are essential for securing a system against attacks. Authentication is the process of proving the user identity to a computer system. The most common form of authentication is to require the user to type in a user name and password before logging on to a computer.

The assumption is that only the real user knows the password. When the system is presented with a user name and a matching password, it grants the user access to the system and stores the user's identifier in protected system memory. At a later point, if the system needs to know the identity of the user, this information can be retrieved from the system memory.

In general, there are three possible ways of doing authentication. A user can be authenticated based on something he knows. An example of this is the password scenario mentioned above. Another way of doing authentication is based on something the user has. This can be a key, or some kind of security token like a smartcard. The third way of performing authentication is based on something the user is. In this case, the system scans one or more of the user biometric to identify that he is the person he claims to be. Fingerprint, eye iris, face shape, and voice recognition fall in this category. It is also common to combine several authentication schemes. For instance, a system might require both a password and a security card in order to complete the authentication, which is called two factor authentication, also there are three factor authentication is used.

Authorization is the process of checking that a logged in user is authorized to perform an operation or access a resource. If the user is not allowed to do so, access is blocked. Authorization requires authentication in order to function, otherwise a user might just lie to the system about who he is in order to gain access to protected resources. Authorization is performed by the operating system before a user is allowed to access the resource. For example, before a file is read the user's identity is fetched from protected storage and compared to the access control list associated with the file to be opened. If the user is allowed to read the file, a valid file handle is returned; otherwise, access is denied.

Auditing is the process of recording security relevant data about a user's activities. For instance, this can be information such as what time and from what IP a user has logged in, or information about attempted access to resources that were denied, such as a user trying to open a file he does not have access to. Some systems also log every system call invocation performed by the programs run by the users. The idea of auditing is to detect users trying to access resources they are not authorized to access, or users trying to circumvent security mechanisms.

Cryptography is also used to secure modern systems. For instance, it is common to encrypt sensitive traffic in transit in order to prevent outsiders from sniffing the data. This is especially true for authentication schemes. If, for instance, an authentication scheme relied on sending passwords in clear text across the network, the authentication could be easily broken by anyone able to read the communication between the server and a user logging in.

2.1.2 Intrusion Detection Systems

Since the security mechanisms presented can fail, additional protection is needed. In order to provide an extra layer of defense, intrusion detection systems (IDSs) have been proposed. IDSs scan through audit data or sniffs into network packets in order to find evidence of malicious behavior. When the manifestation of a possible security violation is found, a system administrator is alerted and presented with a report about the incident. The system administrator then chooses how to act on the report.

2.1.2.1 Classification of IDSs

Intrusion detection systems can be classified in several ways. It is common to classify any IDS by the detection mode, the audit source, the usage frequency, and the response mechanism [4].

Classification by the detection method is most common. There are two main types of detection methods: misuse detection and anomaly detection.

Misuse detection systems utilize a rule database that explicitly models what is not allowed. Everything that does not match any of the rules is allowed.

Anomaly-based systems, on the other hand, use a model of normal activity and anything that does not match the model of normality is considered an attack. An anomaly detector assumes that all anomalous events are signs of an attack and that all attacks produce anomalous events. Since an anomaly-based system does not model attacks specifically, it can detect previously unknown attacks. Misuse-based systems, on the other hand, can only detect attacks for which they have a rule, and cannot detect a novel attack unless it is some variation of one of the attacks already in the rule base.

Misuse detection systems are further divided into stateless and stateful systems. A stateless system only looks at the current audit event when determining if an event is malicious or not. In contrast, stateful systems store and use information about previous audit events. When an event is processed, both the current event and the sensors state are taken into account in the detection process. One of the main advantages of stateful systems is that they can support more complex rules than stateless systems. This complexity comes at a cost though, since stateful sensors consume memory in order to store the state. Stateful sensors also tend to require more processing power than a stateless system, since the rules are more complex.

Anomaly detectors can be either learning-based or specification-based. A learning-based anomaly detection system utilizes a training period, during which the system learns what the properties of normal traffic are. It is usually assumed that the training phase does not contain any attacks; otherwise, the system would include attacks in its normality model. When the training phase is completed, the system switches into detection mode. In this mode, the input data is compared to the model trained in the first step. Any audit event that does not match the learned model is reported as anomalous and logged.

Specification-based systems rely on a specification of what normal traffic should look like. This specification can either be introduced to the system manually or be automatically generated. Automatic generation of the specification can be helpful, since it is a difficult task to manually to specify what normal events look like. Automatic generation is also able to produce a specification with no errors, whereas a manual specification is more prone to include errors. Tools to automatically generate a specification have, for instance, been utilized by systems looking for illegal sequences of system calls. A list of all legal system call sequences can be automatically extracted from the source code of the protected program. During detection, the system call sequences generated by the program are compared to the list of legal sequences.

A different way of classifying intrusion detection systems is by the source of **audit data** processed. Three different categories of audit data sources are common, namely network-based audit data, host-based audit data, and application-based audit data.

Network-based sensors collect packets from the protected network in order to perform detection. Depending on the sensor, the network data processed can be complete packets, packet headers, or payload data. Some network-based systems use firewall logs as input. These firewall logs contain the headers of the network packets that have been blocked by the firewall. Host-based sensors process audit data generated by a host's operating system. It is very common for this type of sensor to perform detection on the log of system calls that have been executed. Other types of host-based systems process different types of system logs, such as UNIX system log data [4].

Some IDSs even use the content of all the files stored on the host's hard drive as input [5]. Application-based sensors process logs created by a user-space application. This kind of sensor is usually used to protect network daemons. For instance, several systems that process web logs [6] and FTP logs.

IDS systems can also be classified by their **usage frequency**. Inline systems operate in real-time and consume audit data as it is generated.. Other systems are run in offline mode, where the system is run periodically to look for signs of attack, this is the most common mode of operation.

It is also possible to classify intrusion detection systems by the **type of response the system** performs when an attack is detected. The most common type of response is passive where an attack occurrence is logged or the administrator is notified by other means (e.g., SMS or email), this is commonly called an alert. Active systems block an attack so it cannot succeed. These systems are usually referred to as intrusion prevention systems (IPSs). Depending on the implementation, an active system could, for instance, send a reset packet to tear down the attacker's connection or update the firewall rules so that the attacker IP is blocked.

2.1.2.2 Intrusion Detection Challenges

The main problem with most intrusion detection systems is that they generate an enormous amount of alerts that are not caused by real attacks. These false alerts are usually referred to as false positives. Many sites that deploy IDSs do not even look at the intrusion reports until after a break-in is detected by other means. The intrusion detection system in these cases becomes more of a tool to help the administrator perform forensic work after the break-in has occurred, than an actual security monitoring tool.

Another problem with current intrusion detection systems is that the intrusion reports they produce do not have enough information for the administrator to make an informed decision about how to handle the incident. One of the reasons for this lack of information is that sensors only operate in one domain. A network-based sensor only sees network-based attributes like IP addresses and port numbers. The network-based sensor has little idea of the security state of the host it is protecting. For instance, it does not know the process id or user id of the processes that are accepting the network connections it observes. Similarly, host-based sensors report little information about the network characteristics of an attack. A host-based sensor that reports a buffer overflow in a program usually does not include the IP address of the attacker simply because the sensor does not have access to this information.

A meaningful prioritization score is also missing from the alerts of most IDSs. Network-based sensors usually do not discriminate between attempted attacks and successful ones. This, in combination with the vast amount of alerts usually produced by sensors, makes it very hard to get a high-level picture of the security state of the protected network.

Non-contextual alerts are also a problem of existing systems. A non-contextual alert is an intrusion detection alert generated as a response to a real attack, but because of the configuration of the host, the attack cannot succeed. An example of this is an alert warning about a web-based attack that only works against Windows computers, while the target is a Linux box. The main cause of this problem is that intrusion detection sensors usually do not have enough information about the hosts they are protecting.

Finally, IDS can only detect single attack but not multi-step attacks, which need network security experts to analyze manually. These challenges may also produce undetected alerts which are real attack. These missing alerts are usually referred to as false negatives.

2.1.3 Intrusion Detection Correlation Systems

In order to alleviate some of the problems of intrusion detection systems, alert correlation systems have been proposed. Correlation systems collect the alerts from a number of sensors and process these alerts in order to generate a high-level view of the current security status of the protected system. The main goal of a correlation system is to reduce the number of alerts a system administrator has to manually process. The correlation system achieves this by identifying and suppressing false alerts, grouping alerts that refer to the same incident together, and prioritizing the alerts.

Three types of correlation techniques have been proposed: multi-step correlation, fusion-based correlation, and filter-based correlation. **Multi-step correlation** seeks to detect attacks that consist of multiple stages. These kinds of attacks are very common. For instance, an attacker might first scan ports in a host in order to identify possible vulnerabilities, before performing the actual break-in. This attack has two stages, the first being the scan, while the second stage is the break-in itself.

Fusion-based correlation systems utilize an alert similarity metric. Incoming alerts are compared to each other using this metric, and alerts that are found to be similar are grouped together. Different similarity metrics are utilized in order to perform different kinds of fusion. Usually systems like these perform multiple correlation steps, where a different similarity function is used for each step.

Correlation systems performing **filter-based** correlation seek to identify the most important alerts in the alert stream. These systems often perform prioritization, where each alert is given a score. This score can be utilized for ranking the alerts so the system administrator can easily get an overview of the most critical alerts. Filter-based correlators usually calculate the criticality of an alert by considering the importance of the assets under attack and the probability that the attack has succeeded.

2.1.4 Recent Cyber Security Attacks

Advanced Persistent Threats (APTs) [7, 8] are a cybercrime category directed at business and political targets. APTs require a high degree of disappearance over a prolonged duration of operation in order to be successful. The attack objectives therefore typically extend beyond immediate financial gain, and compromised systems continue to be of service even after key systems have been breached and initial goals reached. APT can be defined using its named requirement:

Advanced – Criminal operators behind the threat utilize the full spectrum of computer intrusion technologies and techniques. While individual components of the attack may not be classed as particularly “advanced” (e.g. malware components generated from commonly available Do It Yourself (DIY) construction kits, or the use of easily procured exploit materials), their operators can typically access and develop more advanced tools as required. They combine multiple attack methodologies and tools in order to reach and compromise their target.

Persistent – Criminal operators give priority to a specific task, rather than opportunistically seeking immediate financial gain. This distinction implies that the attackers are guided by external entities. The attack is conducted through continuous monitoring and interaction in order to achieve the defined objectives. It does not mean a barrage of constant attacks and malware updates. In fact, a “low-and-slow” approach is usually more successful.

Threat – means that there is a level of coordinated human involvement in the attack, rather than a mindless and automated piece of code. The criminal operators have a specific objective and are skilled, motivated, organized and well funded.

A key requirement for APTs (as opposed to an “every day” botnet, a botnet is a collection of infected computers or bots that have been taken over by hackers and are used to perform malicious tasks or functions) is to remain invisible for as long as possible. As such, the criminal operators of APT technologies tend to focus on “low and slow” attacks – stealthily moving from one compromised host to the next, without generating regular or predictable network traffic – to hunt for their specific data or system objectives. Tremendous effort is invested to ensure that malicious actions cannot be observed by legitimate operators of the systems.

APT Started with low-and-slow scanning, then get in network using malware such that it can creates backdoors and stay undetectable and remote controlled from outside the network, these condition enables it to get data and keep persistent. The APT Continues to Use a Repetitive and Identifiable Targeting and Exploitation Cycle through several steps:

1. Reconnaissance (data gathering): Attackers research and identify individuals they will target in the attacks, using public search or other methods, and get their email addresses or instant messaging handles (Face book, Twitter). Reconnaissance is a step which could not be detected using current security systems.

2. Probing the network: It all typically starts with spear-phishing emails, where the attacker targets specific users within the target company with spoofed emails that include malicious links or malicious PDF or Microsoft Office document attachments. That infects the employee’s

machine and gives the attacker a foot in the door. Detecting of this step depends on the applicability and awareness of specific security policy and the behavior of the employees, while some limitation of these policies prevents the detection of this phase.

3. Establishing a backdoor: The attackers try to get domain administrative credentials and extract them from the network. Since these credentials are typically encrypted, they then decrypt them using pass-the-hash [9] or other tools and gain elevated user privileges. From here, they move "laterally" within the victim's network, installing backdoors here and there. They typically install malware via process injection, registry modification, or scheduled services. The detection of this step could be achieved partially using the host based IDS, but it is still limited without suitable log files analysis.

4. Obtaining user credentials: Attackers get most of their access using valid user credentials, and they access an average of 40 systems [4] on the victim's network using the stolen credentials. The most common type: domain-administrator credentials. This step never been detected upon the success of previous step.

5. Installing multiple utilities: Utility programs are installed on the victim's network to conduct system administration, including installing backdoors, grabbing passwords, getting email, and listing running processes, for instance. The detection of this step could be achieved partially using the host based IDS, but it is still limited without suitable log files analysis.

6. Criminal Remote Control: APTs rely on the remote control functionality in order to navigate to specific hosts within target organizations, exploit and manipulate local systems, and gain continuous access to critical information. Detecting of this step could be achieved by monitoring continuous connection with external networks. While APT

malware can remain stealthy at the host level, the network activity associated with remote control is more easily identified.

7. Privilege escalation, lateral movement, and data exfiltration: Now the attackers start grabbing emails, attachments, and files from servers via the attacker’s computer and communication infrastructure. They typically funnel the stolen data to staging servers, where they encrypt and compress it, and then delete the compressed files from the staging server. Detection of this step depends on the early detection of step 5 and 6.

8. Maintaining persistence: If the attackers find they are being detected or remediated, then they use other methods to ensure they don’t lose their presence in the victim’s network, including revamping their malware. Success of APT attacks depends on their patience and resilience; they are very sophisticated, determined, and coordinated activities. The APT attackers are in there to stay for awhile, not to snatch and grab data.

APT can be detectable in some steps while it is hard in other ones as described above. It will be hard to detect APT in case of insider collusion (co-ordination) unless with robust access control policy. Some parameter needed to have threshold values to represent attacks added with probability factors. Network flow with the controller network needed to be addressed.

Table 2.1 show comparisons of current cyber attackers according to their environment: operating system and IPS they use for the attacks, how they select the attack destination, the attack parameters, the network status during the attack, purpose of the attacks, the investor, and the possibility of detection of these attacks. Attackers are classified into individual amateur, individual professional, group of professionals, and organized cybercrime.

Table 2.1 Cyber Attackers Comparison

	Amateur	**Professional**	**Group of Professional**	**Organized Cybercrime**
O/S	Single	Single / Multiple	Single/Multiple	Multiple
IP address	Single	Single / Multiple	Multiple	Multiple
Destination	Random	Random/ Selected	Selected	Selected
Attack types	Random Blindly	Both random or Predefined MSA	Both random or Predefined MSA	Predefined MSA
Persistent	NO	No	No	Yes
Frequency of Events	Fast	Fast	Fast, yet distributed	Slow, patient, and hidden
Network traffic	Very High	High	High-distributed	Low
Automation	Manuel	Manuel	Manuel/Automated	Automated
Funded	No	No	No	Yes
Purpose	Having fun being hacker	Personal reasons, gain money	Business reasons, gain money	Classified countries information, Big Companies losses
Detectable	Easy	Hardly detectable	Hardly detectable	Undetectable (APTs)

2.2 Basic Concepts of Data Correlation

Data correlation [10] is one of the intrusion detection analysis tools; it is similar but differs to other terms like data aggregation and event reconstruction. Data correlation means associating sets of events detected through various means and applying knowledge to determine whether they are related, and if so, in what manner and to what degree. Data aggregation refers to the process of acquiring more and more data. Event reconstruction means piecing data together to determine exactly what events occurred and in which order. Data sources can be intrusion detection sensors, logs, database, and so forth as shown in Table 2.2.

Previous work on intrusion correlation has mostly focused on IDSs alert correlation. There are three famous techniques [11, 12] for alert correlating which are Similarity-based, Pre-defined attack scenarios and Pre-requisites and consequences of individual attack. These techniques could be verified through correlation phases [13], and could be organized in different architectures and components. The correlation process may be organized from single component or comprehensive set of components. Alert correlation phases include: normalization, aggregation and fusion, verification, building attack scenarios techniques, and prioritization.

Normalization means that all alerts from different sensors should be described with similar attributes. Aggregation looks for alerts that have similar attributes between any two pairs of alerts, according to this similarity the two alerts could be correlated. Verification either identifies alerts that are irrelevant to the protected network or verify that if the alerts are successful in their attacks. The idea of building attack scenarios relies on the fact that complex attacks are usually executed in several r steps, where the first steps prepare for the attacks executed in the later steps.

Therefore, the multi-step correlation approach tries to link alerts that are part of different steps of the same complex attack scenario. Finally prioritization assigns a priority to each alert. Priorities are usually assigned to alerts depending on how important the attacked assets are.

Table 2.2 Sources of Intrusion Detection Data Correlation

Type of Data Source	Major Advantages	Major Disadvantages
System logging	Indicates what actually happened on targeted system	Can be tampered with or turned off altogether; difference in formats can be confusing
Firewall logging	Provide complete picture of inbound and outbound traffic at the point where the firewall is placed in the network	Overwhelming volume of data; difference in formats can be confusing; limitation in dealing with encrypted traffic; packet fragment reassembly issues
Packet dumps	Provide a detailed analysis of traffic going over the network	Overwhelming volume of data (unless dumps are for short time periods);tedious analysis
Network-monitoring tool output	Can provide comprehend picture of the state of the network; particularly valuable in spotting denial of service attacks	Financial expense, most of these tools are commercial
Target-monitoring output	Target-monitoring tools run in the background; changes in files and directories are often indication of attacks	False alarms; financial expense of commercial tools

Type of Data Source	Major Advantages	Major Disadvantages
SNMP traps	Easy to setup and administer; provides remote near-real-time alerting; usefulness of certain kinds of traps (failed logins)	Many versions of SNMP are riddled with vulnerabilities; can flood network
IDS output	Usually reasonably convenient to access and easy to understand	Quality of output (hit rate, false alarm rate) various from one IDS to another; limitation in dealing with encrypted traffic and evasion techniques; limited throughput rate.
Database containing data about attack	Can provide considerable amount of relevant data; allows data mining	Financial cost of setting up and maintaining database privacy issues
Web postings	Search engines can make a wide range of information about incident available; attackers who evade intrusion detection may reveal information about their attack on the web	The accuracy and validity of information posted on the web, especially information concerning attacks, is dubious, false information abounds.

There are two architectures for alert correlation system: centralized architecture [14] and distributed architecture [15, 16]. The key process unit of centralized architecture is Central IDS Correlation Node, which directly processes alerts from multiple IDS sensors. The correlation algorithm of this architecture is simple and can correlate overall alerts quickly. Distributed architecture composed of a set of correlation nodes and categorized as complete distributed architecture or hierarchical distributed architecture. The correlation phases and components are presented in more detail in the remainder of this chapter.

2.2.1 Alert normalization

In the correlation process alerts are received in different formats from different sensors. Intrusion Detection Message Exchange Format (IDMEF) [17] is an XML-based standard for intrusion detection alerts. This standard enables a correlation system to read alerts from different IDS sensors. IDMEF is a specification provided by the Intrusion Detection Working Group (IDWG).

The purpose of IDMEF is to define data formats and exchange procedures for sharing information of interest to security incident detection and response systems. In order to make Alerts compliant with the Intrusion Detection Message Exchange Format (IDMEF), it requires translating raw alerts into a standardized alert format, and assigning them with a standardized name. This format includes a comprehensive set of attributes that can be used by the IDS when reporting alerts. IDMEF is a very useful standard, but it has its problems. For instance, not many of the attributes are required, and for many of the attributes that are required the value \unknown" is accepted. As a result, many IDSs output alerts with most of the attributes set to \unknown".

Another problem is that the IDMEF standard only specifies the syntax of alerts; not the contents of these alerts. For instance, the attack type, which is one of the most important attribute, does not have any standard naming convention associated with it. As a result, different sensors call the same attack by different names. That is, one sensor can refer to a port scan as "port-scan" while another sensor can refer to the same attack as "scanning activity".

The attributes of raw alerts need to be copied to the appropriate fields of the alert as defined by the attribute mappings in the normalization database. The attributes of the standardized alert contain alerted type, analyzer time, attacker nodes, attack graph, consequence, name, priority, etc.

RFC4765 [18] describes a data model to represent information exported by security incident detection systems and explains the rationale for using this model. An implementation of the data model in the Extensible Markup Language (XML) is presented; an XML Document Type Definition is also developed.

One fact should be taken care is that time difference among each network sensors exists in an alert fusion system. All the clocks of the sensors used in the fusion system have to be synchronized. This can be achieved by using the Network Time Protocol (NTP). NTP is a protocol for synchronizing the clocks of computer client or server to another server or reference time source over packet-switched, variable-latency data networks.

2.2.2 Alert aggregation and Fusion

The goal of the alert aggregation is to aggregate large overlap alerts. Aggregation is the grouping of alerts that both are close in time and have similar features. It fuses together different "views" of the same event. Each alert usually has several attributes associated with it, for example, source and target IP addresses.

The similarity-based alert correlation approaches could provide a way to identify what sets of correlated alerts may be further integrated based on the similarity between their attributes. These approaches perform alert correlation through measuring the similarity between alerts attributes to discover the relationships among these alerts. For example, the model proposed in [19] presents a correlation process utilizing an alert similarity metric. The correlation process is carried out in three phases.

The first phase aggregates low-level events using the concept of attack threads. Alerts are clustered together if they are similar with respect to a similarity metric. The metric for the thread phase requires that the sensor field, attack class, attack name, source, and target in the alerts are the same. The idea is to cluster alerts that are part of the same ongoing attack. The next correlation step utilizes a different similarity metric. The requirement that the sensor field is the same is dropped; in addition, the requirement that the alert name is the same is relaxed. The idea of this step is that detection of the same attack by multiple sensors should be fused. The third and last correlation step utilizes another similarity function. This metric relaxes the requirement that the attack class should be the same. The idea of this correlation step is to merge alerts representing different attack steps, in an attempt to provide a higher-level view of the security state of the system.

In [20], a system that performs both aggregation and correlation of intrusion detection alerts produced by a number of different sensors has been proposed. A detailed semantic alert model is presented, and adapter modules are developed to map proprietary alert formats into this model. The pre-processed alerts are first correlated. Two different types of correlation are performed: duplicate removal, and consequence correlation. Duplicates are instances of the same attack as detected by two different sensors, and are detected utilizing rules read from a configuration file. Consequences are rules specifying that one event should be followed by another type of event. After correlation, aggregation is performed.

The aggregation phase clusters together alerts with similar attributes. Three different attributes are utilized in the aggregation phase: source, target, and attack class. The aggregation phase identifies hosts that are sources of attacks, hosts that are the target of attacks, and popular attack classes. For example, if alerts were generated for DDoS attack packets, they would be either similar in destination and attack class in destination, if two alerts are similar in source and target IP addresses, it may be possible that the corresponding attacks are launched by the same attacker.

In [21], a real time aggregation and correlation system named Alertclu is described. Using similarity-based alert clustering analyzing technology, Alertclu can improve the aggregation of intrusion detection system outputs and allow one to seamlessly incorporate additional information. In addition, Alertclu supports the operators by classifying alerts into true positives and false positives. The results of experiments show that the proposed system is able to reduce the numerous redundant alerts and effectively reduces the analyst operators' workload.

2.2.3 Alert verification and Prioritization

The purpose of the verification component is to take a single alert and determine the success of the attack that corresponds to this alert. The idea is that alerts that correspond to failed attacks should be appropriately tagged and their influence on the correlation process should be decreased. In general scenario, the alert corresponding to the worm attack is identified as unsuccessful attack action for a UNIX/Linux service, because it is an exploit for Microsoft Windows. Thus, the alert is tagged as non relevant and excluded from further correlation. Alert verification using vulnerability analysis information has been advocated as an important tool to reduce the noise in the alert stream produced by intrusion detection sensors in [22].

Alert verification can be performed using both passive alert verification and active alert verification techniques. Passive alert verification depends on a priori information gathered about the hosts, the network topology, and the installed services. This technique periodically performs vulnerability scans and updates a database of network assets. This database is then accessed by the correlation system when processing the alerts. If an alert is received and the database indicates that the attacked service is not vulnerable the alert is suppressed.

The advantage of passive techniques is not necessary to perform additional tests, and do not interfere with the normal operation of the network. The disadvantage of passive mechanisms is the potential difference between the status stored in the knowledge base and the actual security status of the network and does not support dynamic mechanisms for alert verification. Instead, they rely on information about the security configuration of the protected network that was collected at an earlier time using vulnerability scanning tools.

Active verification techniques need to look for evidence of the success of an attack by checking information at the victim machine and perform the vulnerability scans as the alerts arrive and do not rely on a database. Scanners are usually adopted in active verification techniques. For example, when a Windows DCOM RPC buffer overrun attack is detected by an detection system, a scanner will be activated. If the scanner script that checks for this particular vulnerability (“Microsoft RPC Interface Buffer Overrun KB824146”) reports that the host that was attacked does not run the Windows RPC service, this alert can be ignored.

Unfortunately, active actions are visible on the network and scanning could possibly have an adverse effect on one’s own machines. It is important to pay attention that scan test run by a vulnerability scanner could crash a service. Port scanning also consumes network bandwidth and resources at the scanned host. One also has to make sure that the alerts generated in response to the activity of the vulnerability scanner are excluded from the correlation process in order to avoid going in an infinite loop of alert detection - scanner execution.

The purpose of alert prioritization is to classify alerts based on their severity and take appropriate actions for dealing with each alert class. Alert prioritization component should take into account various domain information in addition to alert types. Security policy, network topology, vulnerability analysis of the network services and installed software, and asset profiles are some of factors affecting priority of alerts. The prioritization is performed by considering the importance of the asset under attack and the likelihood that the attack will succeed.

The model in [23] relies on a formal description of sensor capabilities in terms of scope and positioning to determine if an alert is a false positive. More precisely, the model is used to verify if all sensors that could have been able to detect an attack agreed during the detection process, assuming that inconsistent detections denote the presence of a false alarm. While this approach benefits from a sound formal basis, it suffers from the limitation that false alerts can only be detected for those cases in which multiple sensors are able to detect the same attack and can participate in the voting process.

Unfortunately, many real-world intrusion detection systems do not provide enough detection redundancy to make this process applicable. The model can be seen as a formal model for representing security related information including vulnerabilities, security tools, alerts and information system characteristics. Although it provides the formalism for modeling security related information, specific mechanisms are still required for prioritizing alerts.

2.3 Alerts Correlation Techniques

Alert correlation focuses on discovering the relationships between individual alerts raised by security incident detection systems and other security systems. Alert correlation has to do with the recognition of logically linked alerts, and is dedicated to disclose the logical association between network attack activities by analyzing their corresponding alerts.

Generally, the method of alert correlation deals with meta-alert which is generated by alert aggregation. The main approach of alert correlation can be divided into three classifications: correlation of attack scene, correlation of prerequisites and consequences, and causal analysis correlation based on a statistical technique.

2.3.1 Correlation of Attack Scene

The methods in [24] studies the relationship between contextual attack behaviors and use the method of correlation rule matching based on the causality relationship between two contextual attack steps to construct attack scenarios. This method is similar to the way of misuse detection. The predefined attack scenarios based approaches correlate alerts based on known scenario templates. The templates are patterns of known sequences of attacks consisting of individual attack steps. Then, they match agent alerts to attack steps in the scenario templates.

In [25], a new method of mining multi-stage attack behaviors pattern was proposed in order to recognize attacker's high level strategies and predict upcoming attack intentions. Authors applied a reformative algorithm to mine frequent attack sequence patterns from history alert data. They also used correlativity between two contextual elements in the attack sequence to correlate attack behaviors and identify potential attack intentions.

In [26] event correlation and attack scenario construction based on association with network attack graphs is proposed. It handles missed detections through the analysis of network vulnerability dependencies. The attack graph provides the necessary context for intrusion events, and provides the graph distances upon which the correlations are based. Online event processing depends on pre-computed attack graph distances only, and requires only a lookup and 4 arithmetic operations. To compute attack graph distances (offline), a model of attacker exploits and network vulnerabilities have been built.

The network vulnerability model has been created automatically from output of the Nessus [27] vulnerability scanner. The model then computes the distance of the shortest path between each pair of exploits in the attack graph. These distances are a concise measure of exploit relatedness, which could be used for subsequent online causal correlation of intrusion detection events. From the online stream of intrusion events, individual event paths have been built based on attack graph reachability. The inverse distance between each event in a path is a measure of correlation.

The approach proposed in [28] consists of a number of phases including alert clustering, alert merging, and intention recognition. In the first two phases, alerts are clustered and merged using a similarity function. The intention recognition phase is referenced in their model, but has not been implemented. An interesting aspect of this approach is the attempt to generate correlation rules automatically. While it may seem appealing, this technique could generate a number of spurious correlation rules that, instead of reducing the number of alerts and increasing the abstraction level of the reports, could introduce the correlation of alerts that are “close" or “similar" by pure chance, in this way increasing the noise in the alert stream.

Some approaches [29, 30] specify attack scenarios through attack languages. In [30] attack scenarios through chronicle language are modeled. A chronicle is a set of events that are connected by temporal constraints. The key of this method is how to construct the scenario templates by the patterns of correlated alerts. Several algorithms were developed for the mining of sequential patterns. It proposes a multi-alarm misuse correlation component based on the chronicles formalism.

In [31] State Transition Analysis Technique (STAT) has been provided to model and detect security incidents in large-scale, heterogeneous networks. In [32] propose a completely decentralized approach to solve the task of event correlation and information fusing of the data gathered from multiple points within the network is proposed. The system models an intrusion as a pattern of events that can occur at different hosts and consists of collaborating sensors deployed at various locations throughout the protected network installation. They present a specification language to define intrusions as distributed patterns and a mechanism to specify their simple building blocks.

The peer-to-peer algorithm to detect these patterns and its prototype implementation, called Quicksand, is developed. These methods can potentially uncover the causal relationship between alerts, but they need to define the specification of attacks and the results rely on the precision of correlation rules. Such limitations make the methods hard to implement.

2.3.2 Correlation of Pre and post conditions

Pre and post conditions (also called prerequisites and consequences) are defined for individual attacks. The prerequisites and consequences based approaches [33, 34] model each attack through describing its prerequisites and its consequences. Intuitively, the prerequisite of an attack is the necessary condition to launch an attack successfully, and the consequence of an attack is the possible outcome if an attack succeeds.

Alerts are connected (or correlated) when the post condition of one alert matches the precondition of a later one. This allows for the specification of complex chains of attacks without having to explicitly model complex scenarios. For an example of an attack that can be correlated using this technique, consider an attack where the intruder first breaks into a host in the Demilitarized Zone (DMZ) of a company. A demilitarized zone is a computer network that sits between the internal network and the Internet and acts as a security buffer. After breaking into this host, the attacker performs another attack starting from the compromised host. Both steps of the attack are detected by intrusion detection sensors and alerts are sent to the correlation system.

Upon receiving the first alert, the correlation system utilizes a rule that says no precondition is needed to attack a host in the DMZ and the postcondition is that the attacker has access to the attacked host. The second attack step triggers a rule that has a precondition that says that attacks originating from the DMZ require access to the DMZ host. The postcondition of this rule is that the attacker has access to an internal host. These two alerts will now be joined, since the postcondition of the first attack (access to a DMZ host) matches the precondition of the second attack.

Another example of a technique that uses pre and postconditions to identify causal relationships between alerts is presented in [35]. In this paper, attack conditions are expressed using capabilities and concepts. Capabilities are used to describe both information that the attacker must know to perform a certain attack (e.g., a user name and password for a valid account), or a condition that represents a necessary context for an attack (e.g., a particular configuration of the network). Concepts are used to model fragments of complex attacks (e.g., a denial-of-service attack against a specific host) and both their requisites and their impact on the security of the protected network are expressed in terms of capabilities.

By composing the capability provided by a concept with the capability required by another concept it is possible to recognize complex attack scenarios (e.g., a remote shell connection spoofing that relies on a denial-of-service attack).

The correlation method [36] uses logical formulas to represent the prerequisites and consequences of attacks. A logical formula is a logical combination of predicates. The prerequisites, consequences and attributes of attacks are formalized as meta-alert types (or alert types). A hyper-alert type (or alert type) is a triple (fact, prerequisite, consequence), where the fact is a set of alert attribute names associated with the corresponding domains, the prerequisite is a logical formula, and the consequence is a set of logical formulas.

The usefulness of the system has been demonstrated by showing how it could significantly reduce the number of false alarms reported by a detection system while negligibly reducing the valid alarms. Their tool is most logically used as an off-line forensic tool for mining old stored alerts after a new vulnerability is found.

2.3.3 Casual analysis Correlation based on Statistical Techniques

In [37] the proposed model focuses on discovering novel attack strategies via analysis of security alerts. In alert correlation, the developed correlation system was based on two hypotheses of attack step relationship. The first hypothesis is that some attack steps are directly related because an earlier attack enables or positively affects the later one. They developed a probabilistic-based correlation engine that incorporates domain knowledge to correlate alerts with direct causal relationship.

The second hypothesis is that some related attack steps, even though they do not have obvious or direct (or known) relationship in terms of security and performance measures, still exhibit statistical and temporal patterns. Two correlation engines have been developed to discover attack transition patterns based on statistical analysis and temporal pattern analysis, respectively. Based on the correlation results of these correlation engines, they construct attack scenarios and conduct attack path analysis. The security analysts are presented with aggregated information on attack strategies from the integrated correlation system.

Alert fusion is more complex when taking into account anomaly detection systems, because no information on the type or classification of the observed attack is available to the fusion algorithms. The model proposed in [38] generated high level correlated alerts from low level sensor data and then conducted causal analysis based on a statistical technique, known as the Granger Causality Test (GCT), to discover new patterns of attack relationships. They used time series analysis methods to find implicit relationships in alert data. They grouped alerts sharing all attributes together allowing a small time window in the order of few seconds. This grouped alerts issued on the same attack. In next step they grouped alerts with identical attribute values apart from the sensor. This step aggregated together alerts related to the same attack issued from heterogeneous sensors, again a small difference in time stamps is allowed.

2.3.4 Distributed Correlation

The model proposed in [39] describes a mission-impact-based approach to the analysis of security alerts produced by spatially distributed heterogeneous information security (INFOSEC) devices, such as firewalls, intrusion detection systems, authentication services, and antivirus software. The intent of their work is to deliver an automated capability to reduce the time and cost of managing multiple INFOSEC devices through a strategy of topology analysis, alert prioritization, and common attribute-based alert aggregation.

This approach relies on a knowledge base that describes the security-relevant characteristics of a protected network to prioritize the alerts through computing rank of the alerts and clustering them based on the ranks. The knowledge base is called Incident Handling Fact Base, which provides some critical information regarding alert codes, their descriptions, and dependencies of alert types to their required OS versions, hardware platforms, network services and vulnerabilities. Using this knowledge base a simple form of passive alert verification could be performed where alerts representing attacks against non-existent services are discarded.

The information about network assets is gathered using Nmap [40] and contains only information that is gathered by this specific tool (e.g., IP addresses, installed operating systems, and open ports). The prioritization is performed by considering the importance of the asset under attack and the likelihood that the attack will succeed. They developed a prototype system called the Mission Impact Intrusion Report Correlation System, or M-Correlator. M-Correlator is capable of receiving security alert reports from a variety of INFOSEC devices. It is intended to provide analysts (at all experience levels) a powerful capability to automatically fuse together and isolate those INFOSEC alerts that represent the greatest threat to the health and security of their networks.

In [41] a novel intrusion detection system for grid systems is presented. It is intended to identify potential attackers who try to modify or compromise the applications sent to execution by various users or target different resource groups within the Grid. The system makes use of a number of available local intrusion detection systems which send data to a grid-level intrusion detection system that takes decisions based on an overview of the entire Grid.

These IDS can correlate the information received from the local systems, as well as monitoring data from the Grid System, using statistical methods, to identify attacks that cannot be detected at a local level. Another contribution of this paper is the classification of threats based on the intent of the attacker. This paper also demonstrates that these types of attacks can be detected using the proposed complex intrusion detection system.

The model in [42] is proposed to achieve alert correlation which supplies information about the vulnerabilities. They used a hyper-alert type to encode their knowledge about each type of attacks. The proposal has a relational database that implements parts and the corresponding tables are automatically generated from data sources. IDS and vulnerability scanner fill the database with events.

In [43] it is analyzed how the control and estimation methods can be applied to correlate distributed events for network security. Based on those methods, a Process Query System has been implemented which can scan and correlate distributed network events according to users' high-level description of dynamic processes.

2.4 Alert Correlation Limitations

Most of the approaches based on pre and post conditions focus on the modeling and detection of multi-step attacks to provide a high-level view of the “attack history" associated with a security compromise. It is assumed that the analyzed event stream is composed only of well-defined, relevant alerts, and that real attacks trigger more than a single alert. As a result, these systems can focus on clusters of related alerts and discard all alerts that have not been correlated.

Unfortunately, this assumption has not been substantiated by experimental data or supported by a rigorous analysis. In practice, it is often necessary to filter out irrelevant alerts that may generate spurious attack histories. This view is supported by [44] on alert correlation, which states that false alerts generated by IDSs have a negative impact [on correlation].

A limitation of approaches that are based on pre and postconditions is the need to manually define these conditions for all alerts. In addition, when only dependencies between alerts are modeled (as opposed to complete scenarios), it is not possible to monitor the evolution of a particular scenario instance from state to state in real-time, possibly anticipating the further progress of an intrusion. In addition, it requires that all the relevant preconditions and postconditions are modeled. If a relevant precondition or postcondition is not modeled, some causal relationships between alerts could go undetected.

Given the large number of attacks and the platform-specific nature of pre and postconditions, effective alert reduction would require a substantial modeling effort, similar to the effort required to develop complete attack rule sets for misuse-based detection systems. Another problem is the assumption that only attacks that are carried out in multiple steps are important. While it is reasonable to give high priority to alerts that have been detected as part of a multi-step attack, it is not wise to disregard all alerts that are not part of a multi-step attack.

2.5 Agents in IDS and Correlation

Agent is as a distinct software process being able to accomplish some work without manual intervention and supervision in certain condition [45]. It is self-adaptable, intelligent and collaborative. An Agent not only works independently, but also can accomplish some missions and cooperate with other Agents. Further, an Agent can be controlled to perceive the change of environment and act to the environment back. Agents are autonomous that can act independent from other agents and perform different tasks. They are also robust and fault-tolerant to changing environments.

There are two kinds of agents: static agents and mobile agents. A static agent was the first proposed agent technology which is applied in the area of intrusion detection. A static agent, that is to say, the agent that resides in a fixed position or some fixed platforms. A mobile agent is an entity capable to move from a node to another over the network in order to perform the work locally. It permits to spread dynamically the server interfaces managed on the different sites. It guarantees a big resistance to network breakdowns; it also permits savings of bandwidth since negotiations between mobile agent and server consist in local message exchanges that don’t pass by the network [46].

In an agent based IDS idea, there is no central node, therefore no central point of failure. Overcoming the deficiency of centralized structure is the major reason for using agents in the intrusions detection field. The agents usefulness includes also reduction of the network load, overcoming of network latency and support for disconnected operations.

In [47], a lightweight and adaptive mobile agent-based intrusion detection system (LAMAIDS) is presented. The presented model detects intrusion from outside the network as well as from inside. A main machine, being a typical intrusion detection system residing at a secure location, creates mobile IDS agents and dispatches them into the network. The mobile IDS agents are equipped with lightweight IDS capabilities and decision-making. On each hop, the agents sniff the network traffic and look for abnormal activities using a set of rules supplied by the main machine. Simulation results based on real-world scenarios demonstrate significant improvements in terms of detection rate, network overhead, and adaptability, scalability, and fault tolerance.

In [48], a novel hybrid model for Mobile Agent based Distributed Intrusion Detection System was proposed. The proposed model has new features such as robustness, capability of detecting intrusion against the IDS itself and capability of updating itself to detect new pattern of intrusions. In addition, the proposed model is also capable of tackling some of the weaknesses of centralized Intrusion Detection System models.

In [49] a distributed intrusion detection system model based on agents is proposed. This system adopts the way which combines static agent and mobile agents, Host-based Intrusion Detection System (IDS) and Network-based Intrusion Detection System. The system uses mobile agent for decentralized data collection, data analysis and response, and has certain dynamic learning capability.

In [50, 51], an autonomous agent has been trained to observe system behavior and flag any anomalous activity. In this prototype, agent monitors the network traffic and been subjected to training phase to detect the malicious behavior in the network by human operator. In [52], distributed agent architecture have been used for intrusion detection, the model proposed a mobile agent based model for intrusion detection system, called MAFIDS, including new metrics issued from emergent indicators of the agent synergy and a proposed event correlation engine. The model implementation showed its capabilities to detect the SYN flooding attack in a short time and lower false alarm rate by comparing it to SNORT [53]. The idea was to take advantage from agent technology to overcome two major problems of other IDS: a longer detection, higher false alarm rate.

In [54, 55], distributed agent approach for alarm correlation was proposed to identify the root causes of network failures and fault identification. The proposed model presented a new distributed alarm correlation approach that effectively tackles the aforementioned data deficiencies. According to the proposed approach, the managed network is first divided into a disjoint set of management domains and each domain is assigned an intelligent agent, the intelligent agent perceives each network entity in its domain as a source of information and assigns weights for emitted alarms by these entities. Based on their weights, the observed alarms are then correlated by their respective agent into a single local fuzzy composite alarm. Since local composite alarms constitute only partial views of the managed network, they are correlated, by a higher management entity, into a global alarm that accurately reflects a comprehensive view of the managed network.

2.6 Comprehensive Approach Model for IDS Alert Correlation

Comprehensive approach model for real-time alert correlation [56 - 58] has been produced as integrated solution. It consists of a set of correlation components which cover different correlation techniques as shown in Figure 2.1. The alert correlation module is composed of a set of procedures which can be arranged in different ways. Some procedures process data of an alert and the others implement correlation methods by combining alerts using individual filters.

Six main components have been implemented depending on five types of filters: Fusion, One2One, Network-Host, One2Many, and Many2One. The correlation components which effectively reduce alerts are: Alert Fusion (AF) which combines duplicate alerts that represent the independent detection of the same attack by different IDS. Alert Verification (AV) which takes a single alert and determines the success of the attack corresponding to that alert. Thread Reconstruction (TR) which combines a series of alerts that refer to attacks launched by a single attacker against a single target. Attack Session Reconstruction (ASR) associates network-based alerts with host-based alerts that are related to the same attack.

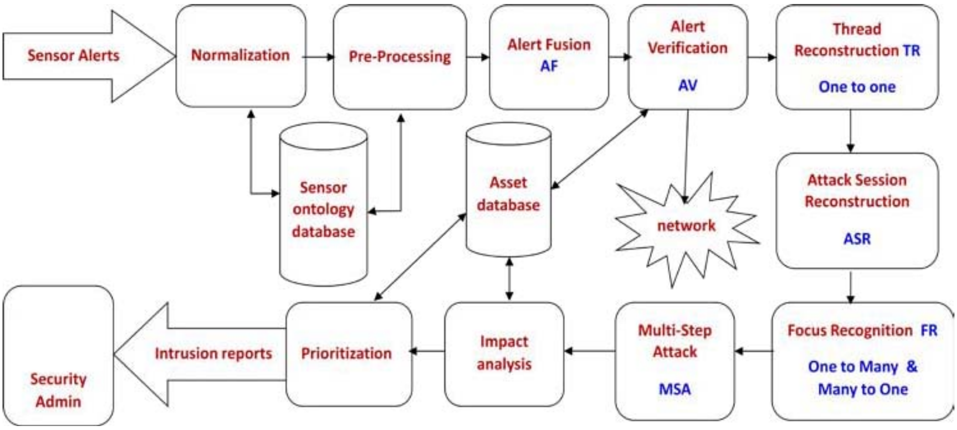
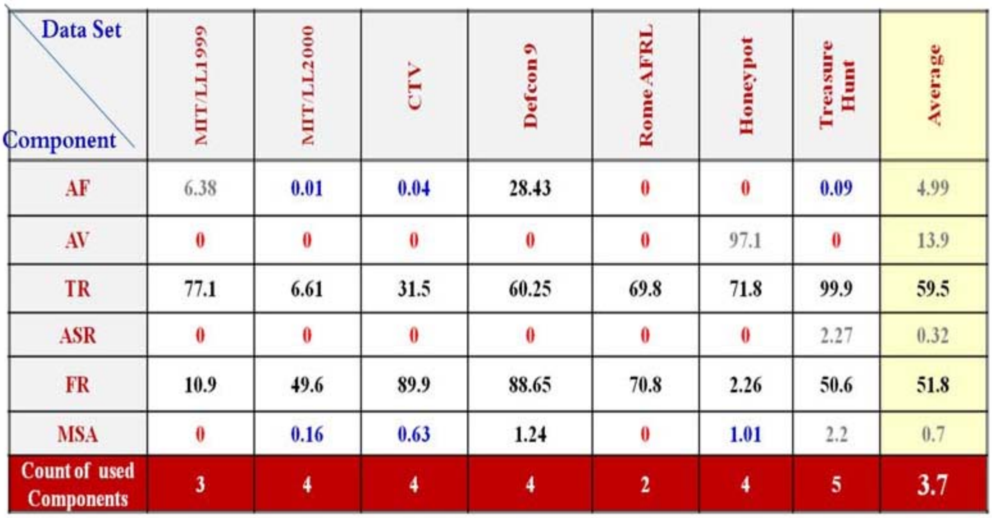


Figure 2.1 comperhensive approach model for IDS alert correlation

Focus Recognition (FR) which identifies the hosts that could be the source or the target of a substantial number of attacks. More specifically, this component aggregates the alerts associated with a single host attacking multiple victims (called a one2many scenario), and a single victim that is targeted by multiple attackers (called a many2one scenario). Multi-Step Attack (MSA) which identifies common attack patterns such as recon-breakin-escalate or island-hopping attacks {attacker breaks into a host and uses it as a launch for more attacks}. The victim in one alert becomes the attacker in the following one. There are more additional two components: impact analysis, and prioritization, that depend on the nature and the policy of the protected network. However, both of them are not evaluated in this approach.

The study and analysis of components reduction rate of the model is shown in Table 2.3. The rows represent different CAM components reduction rate values, while the columns represent the different used datasets. The table shows that TR and FR components have the highest Reduction Rate (RR) percentage, are considered the most effective components used for all datasets. Both AF and MSA have lower RR values, yet they are still used for the most of datasets. Each of AV and ASR does not have any effect except on one dataset only. It is concluded that the affected correlation components are six (AF, AV, TR, ASR, FR, MSA), but not all of such components are used for all different dataset (The average is 3.7 component).

Table 2.3 CAM Components Reduction Rate for Different Datasets



Data Set / Component	MIT/LL1999	MIT/LL2000	CTV	Defcon 9	RomeAFRL	Honeypot	Treasure Hunt	Average
AF	6.38	0.01	0.04	28.43	0	0	0.09	4.99
AV	0	0	0	0	0	97.1	0	13.9
TR	77.1	6.61	31.5	60.25	69.8	71.8	99.9	59.5
ASR	0	0	0	0	0	0	2.27	0.32
FR	10.9	49.6	89.9	88.65	70.8	2.26	50.6	51.8
MSA	0	0.16	0.63	1.24	0	1.01	2.2	0.7
Count of used Components	3	4	4	4	2	4	5	3.7

The performance of IDS correlation is measured by reduction rate and correlation time. The correlation time for each component is calculated by the count of input alerts and the correlation time for each alert. The sequence order of correlation components affects the correlation process performance; the total time needed for the whole process depends on the number of processed alerts in each component. Table 2.3 shows analysis result of the effectiveness of each component on the different analyzed datasets. The last row shows the total count of effective components whose reduction rate is more than zero value. Such count differs according to the dataset, and varies from minimum two components in the case of “Rome AFRL” dataset to five in the case of” Treasure hunt”. The RR for each component varies from 0 to 99.91 % depending on the component algorithm and selected dataset.

Different reduction rates of each component simply affect the following component input of alert stream, i.e. the arrangement order of the components is a primary concern for each dataset to obtain faster correlation process. Different RR of a single component in different dataset is varying because of the difference of attack scenario, and target networks used for each dataset.

CHAPTER 3

Distributed Agent Correlation Model

Chapter Three: Distributed Agent Correlation Model

As stated earlier, alert correlation is a major required mechanism to provide useful and comprehensive output of IDS. Although several techniques have been proposed to carry such a mechanism there is still a lot of work to enhance, both the performance and quality of such techniques. Whereas having an additional source of information about a particular event is useful when the uncertainty of the source, the accuracy, and the scope of that event is considered. In this chapter a novel model is proposed to enhance both the performance and quality of the correlation task. Agents, learning, and multi sources output correlation are used to achieve this task. In the rest of this chapter a description, component, and features of the proposed model will be illustrated.

3.1 Distributed Agent Correlation Model Description

Distributed Agent Correlation Model (DACM) is a multi-agent distributed correlation model in a hierarchical organization. It correlates alerts from IDS's and from other sources of information. Data sources for correlation are IDSs, system and application log files for different services provided by the system, and security tools. Examples of security tools are *firewalls*, *vulnerability scanners*, and *performance monitors*, while examples of application and system log files are audit system logs, FTP logs, SSH logs, http/https logs, and OS log files.

Figure 3.1 shows the block diagram of DACM. The figure shows that DACM has its inputs from different information sources. DACM core correlates these input data using a set of local and central agents depending on the learning capability as well as knowledge base and security policy. Finally, DACM produces the output as a report for security administrator or automated response capability. DACM phases include: *collection*, *storage*, *analysis*, *presentation*, *sharing*, and *reaction*. The details of each agent will be described in the rest of this chapter while the detailed implementation will be described in the next chapter.

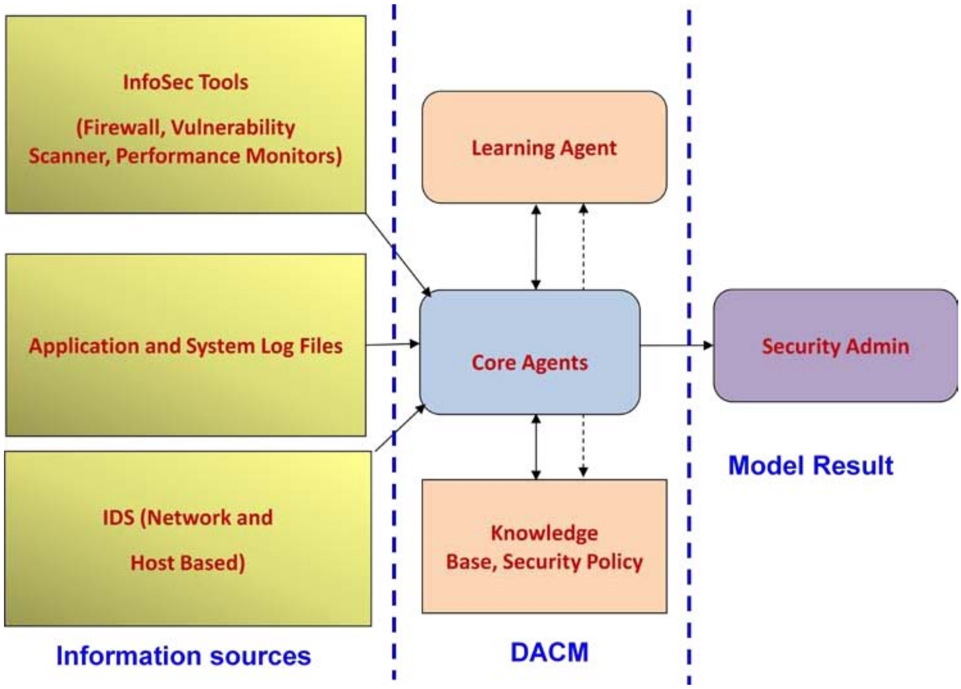

InfoSec Tools
(Firewall, Vulnerability Scanner, Performance Monitors)
Application and System Log Files
IDS (Network and Host Based)
Information sources
Learning Agent
Core Agents
Knowledge Base, Security Policy
DACM
Security Admin
Model Result

Figure 3.1DACM Block Diagram

DACM core agents consist of a set of correlation agents for different sources of information. These agents are grouped into three main classifications: IDS's correlation agents for both network based and host based IDSs, INFOSEC tool agents for different available security tools in the system, and system and application log agents for different auditing logs of operating system and available serves and application.

3.1.1 IDSs Correlation Agents

IDS alert correlation consists of a set of correlation components within a certain structure. IDS sensors could be network based IDS or host based IDS, each of them produces its own alerts. Network based IDS has local correlation agents to correlate its alerts, on the other hand host based IDS has its own correlation agent. The main IDS correlation agent correlates the output of network based correlation agent and host based correlation agent together and produce IDS’s correlated alerts. Figure 3.2 shows a block diagram of IDS's correlation agents for IDS sensors of network based and host based IDS's.

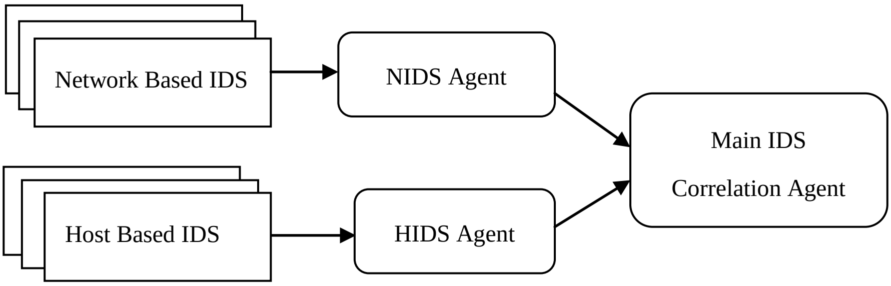
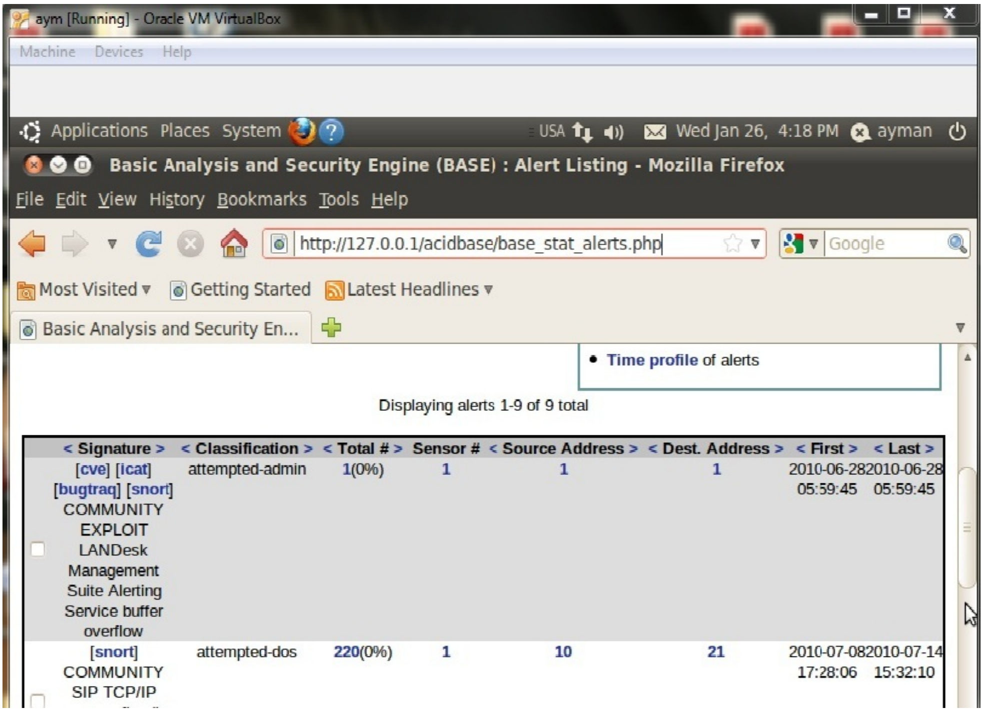


Figure 3.2 IDS Correlation Agents

Figure 3.3 shows an example of the output of snort IDS as network based IDS using Basic Analysis and Security Engine (BASE) tools [59] interface, the output shows alert attributes discovered by snort for intrusion attempts.


aym [Running] - Oracle VM VirtualBox
Machine Devices Help
Applications Places System
USA
Wed Jan 26, 4:18 PM
ayman
Basic Analysis and Security Engine (BASE) : Alert Listing - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://127.0.0.1/acidbase/base_stat_alerts.php
Google
Most Visited
Getting Started
Latest Headlines
Basic Analysis and Security En...

- Time profile of alerts

Displaying alerts 1-9 of 9 total

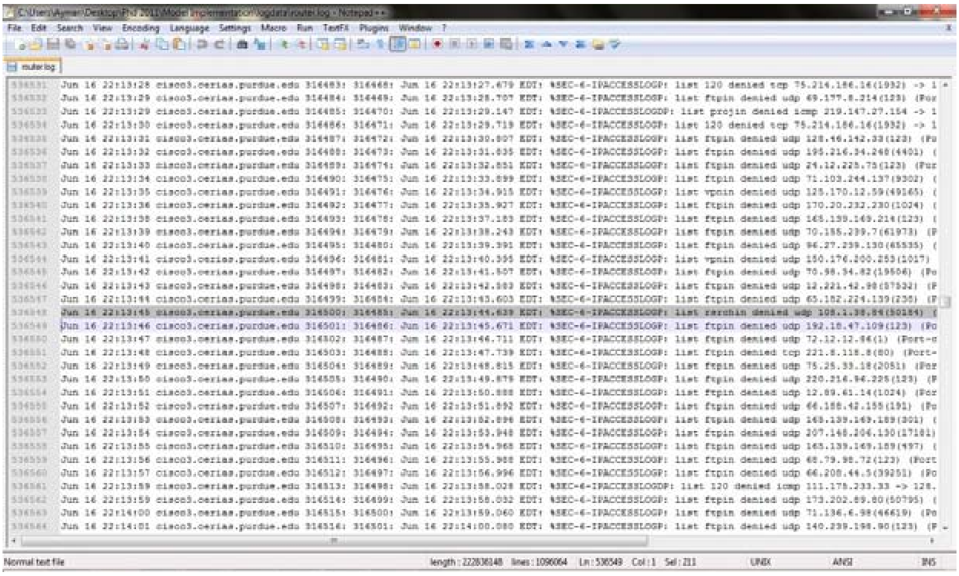
	< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
□	[cve] [icat] [bugtraq] [snort] COMMUNITY EXPLOIT LANDesk Management Suite Alerting Service buffer overflow	attempted-admin	1(0%)	1	1	1	2010-06-28 05:59:45	2010-06-28 05:59:45
□	[snort] COMMUNITY SIP TCP/IP	attempted-dos	220(0%)	1	10	21	2010-07-08 17:28:06	2010-07-14 15:32:10

Figure 3.3 IDS alerts Output using BASE for Snort

Details of the used correlation techniques for IDSs correlation agents are presented in section 4.2 in next chapter.

3.1.2 INFOSEC Tools Agents

Information Security tools are software which concern and analyze the information exchange within network traffic to determine which of this traffic trying to access resources as illegitimate behavior. A firewall is one of the most famous tools which consists of software and/or hardware devices. A firewall [60] is a secure internet gateway that is used to interconnect a private network to the Internet. It is used as a technological barrier designed to prevent unauthorized or unwanted communications between computer networks or hosts, depending on a set of access control lists within the network hosts and resources. Figure 3.4 shows the content of firewall router log files which shows the blocked list of IPs trying to access the website without authorized privileges or access control policy.


C:\Users\Ayman\Desktop\PhD 2011\Model implementation\logdata\router.log - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
router.log

```
Jun 16 22:13:28 cisco3.cerias.purdue.edu 316483: 316468: Jun 16 22:13:27.679 EDT: %SEC-6-IPACCESSLOGP: list 120 denied tcp 75.214.186.16(1932) -> 1
Jun 16 22:13:29 cisco3.cerias.purdue.edu 316484: 316469: Jun 16 22:13:28.707 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 69.177.8.214(123) (Por
Jun 16 22:13:29 cisco3.cerias.purdue.edu 316485: 316470: Jun 16 22:13:29.147 EDT: %SEC-6-IPACCESSLOGDP: list projin denied icmp 219.147.27.154 -> 1
Jun 16 22:13:30 cisco3.cerias.purdue.edu 316486: 316471: Jun 16 22:13:29.719 EDT: %SEC-6-IPACCESSLOGP: list 120 denied tcp 75.214.186.16(1932) -> 1
Jun 16 22:13:31 cisco3.cerias.purdue.edu 316487: 316472: Jun 16 22:13:30.807 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 128.46.142.33(123) (Po
Jun 16 22:13:32 cisco3.cerias.purdue.edu 316488: 316473: Jun 16 22:13:31.835 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 195.216.34.248(4401) (
Jun 16 22:13:33 cisco3.cerias.purdue.edu 316489: 316474: Jun 16 22:13:32.851 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 24.12.225.75(123) (Por
Jun 16 22:13:34 cisco3.cerias.purdue.edu 316490: 316475: Jun 16 22:13:33.899 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 71.103.244.137(9302) (
Jun 16 22:13:35 cisco3.cerias.purdue.edu 316491: 316476: Jun 16 22:13:34.915 EDT: %SEC-6-IPACCESSLOGP: list vpnin denied udp 125.170.12.59(49165) (
Jun 16 22:13:36 cisco3.cerias.purdue.edu 316492: 316477: Jun 16 22:13:35.927 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 170.20.232.230(1024) (
Jun 16 22:13:38 cisco3.cerias.purdue.edu 316493: 316478: Jun 16 22:13:37.183 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.214(123) (
Jun 16 22:13:39 cisco3.cerias.purdue.edu 316494: 316479: Jun 16 22:13:38.243 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 70.155.239.7(61973) (P
Jun 16 22:13:40 cisco3.cerias.purdue.edu 316495: 316480: Jun 16 22:13:39.391 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 96.27.239.130(65535) (
Jun 16 22:13:41 cisco3.cerias.purdue.edu 316496: 316481: Jun 16 22:13:40.395 EDT: %SEC-6-IPACCESSLOGP: list vpnin denied udp 150.176.200.253(1017)
Jun 16 22:13:42 cisco3.cerias.purdue.edu 316497: 316482: Jun 16 22:13:41.507 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 70.98.34.82(19506) (Po
Jun 16 22:13:43 cisco3.cerias.purdue.edu 316498: 316483: Jun 16 22:13:42.583 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 12.221.42.98(57532) (P
Jun 16 22:13:44 cisco3.cerias.purdue.edu 316499: 316484: Jun 16 22:13:43.603 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 65.182.224.139(238) (P
Jun 16 22:13:45 cisco3.cerias.purdue.edu 316500: 316485: Jun 16 22:13:44.629 EDT: %SEC-6-IPACCESSLOGP: list rerchin denied udp 108.1.38.84(50184) (
Jun 16 22:13:46 cisco3.cerias.purdue.edu 316501: 316486: Jun 16 22:13:45.671 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 192.18.47.109(123) (Po
Jun 16 22:13:47 cisco3.cerias.purdue.edu 316502: 316487: Jun 16 22:13:46.711 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 72.12.12.86(1) (Port-c
Jun 16 22:13:48 cisco3.cerias.purdue.edu 316503: 316488: Jun 16 22:13:47.739 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied tcp 221.8.118.8(80) (Port-
Jun 16 22:13:49 cisco3.cerias.purdue.edu 316504: 316489: Jun 16 22:13:48.815 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 75.25.33.18(2051) (Por
Jun 16 22:13:50 cisco3.cerias.purdue.edu 316505: 316490: Jun 16 22:13:49.879 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 220.216.96.225(123) (P
Jun 16 22:13:51 cisco3.cerias.purdue.edu 316506: 316491: Jun 16 22:13:50.888 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 12.89.61.14(1024) (Por
Jun 16 22:13:52 cisco3.cerias.purdue.edu 316507: 316492: Jun 16 22:13:51.892 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 66.188.42.155(191) (Po
Jun 16 22:13:53 cisco3.cerias.purdue.edu 316508: 316493: Jun 16 22:13:52.896 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.189(301) (
Jun 16 22:13:54 cisco3.cerias.purdue.edu 316509: 316494: Jun 16 22:13:53.948 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 207.148.206.130(17181)
Jun 16 22:13:55 cisco3.cerias.purdue.edu 316510: 316495: Jun 16 22:13:54.968 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.189(497) (
Jun 16 22:13:56 cisco3.cerias.purdue.edu 316511: 316496: Jun 16 22:13:55.988 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 68.79.98.72(123) (Port
Jun 16 22:13:57 cisco3.cerias.purdue.edu 316512: 316497: Jun 16 22:13:56.996 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 66.208.44.5(39251) (Po
Jun 16 22:13:59 cisco3.cerias.purdue.edu 316513: 316498: Jun 16 22:13:58.028 EDT: %SEC-6-IPACCESSLOGDP: list 120 denied icmp 111.175.233.33 -> 128.
Jun 16 22:13:59 cisco3.cerias.purdue.edu 316514: 316499: Jun 16 22:13:58.032 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 173.202.89.80(50795) (
Jun 16 22:14:00 cisco3.cerias.purdue.edu 316515: 316500: Jun 16 22:13:59.060 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 71.136.6.98(46619) (Po
Jun 16 22:14:01 cisco3.cerias.purdue.edu 316516: 316501: Jun 16 22:14:00.080 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 140.239.198.90(123) (P
```

Normal text file length : 222838148 lines : 1096064 Ln : 536549 Col : 1 Sel : 211 UNIX ANSI INS

Figure 3.4 Firewall router log file[1]

[1] Larger image of Figure 3.4is included in Appendix A

Local Agent to correlate (aggregate) firewall router log files entries and group these entries for each blocked IP per day. Correlation is based on grouping the same blocked IP into a single record, a number of attempts field is added as an aggregation attribute.

A vulnerability scanner [61] is a computer program designed to assess computers, computer systems, networks or applications for weaknesses. There are a number of types of vulnerability scanners available today, distinguished from one another by a focus on particular targets. Types of vulnerability scanners could be Port Scanner, Network Enumerator, Network Vulnerability Scanner, Web Application Security Scanner, Database Security Scanner , and Computer Worm. A system monitor is hardware and/or software based system used to monitor resources and performance in a computer system. DACM has INFSEC agent to correlate the output of these tools according to specific behavior.

Other local agents correlate vulnerability scanner outputs according to scanner type. Mainly the vulnerability that is related to the same target port and grouped together, and the IP/Port combination are used to identify this alert.

Performance monitor tools provide and view information about the use of hardware (CPU, memory, disk, and network) and software (file handles and modules) resources in real time. It displays basic system resource usage information, and displays column lists of processes, services, associated handles and associated modules; charts of CPU usage. It also displays overall physical memory consumption and separate consumption of every process; charts of used physical memory. It views disk usage through displaying processes with disk activity, storage, charts of disk usage (KB/sec), and disk queue length. Finally, it displays processes with network activity, TCP connections, and listening ports.

Performance monitor tools allow network administrators to proactively discover and address end-to-end network performance issues, measure the amount and type of traffic on a particular network, and locate and diagnose congestion and latency problems for network troubleshooting by using real-time and historical reports

Local agent correlates/aggregates performance monitor outputs according to specific use of profile matching. This agent type analyzes the network performance monitoring tool outputs. Normal network performance thresholds values are defined in different intervals during the day resulting in a performance profile. This profile is constructed by supervised learning and stored in the knowledge base. If the monitored performance exceeds the threshold, an alert is generated. Multiple alerts for the same performance metric and period are grouped together; performance metric is used for traffic rate, usage ratio, congestion rate and so on. Figure 3.5 show block diagram of INFOSEC tools agents.

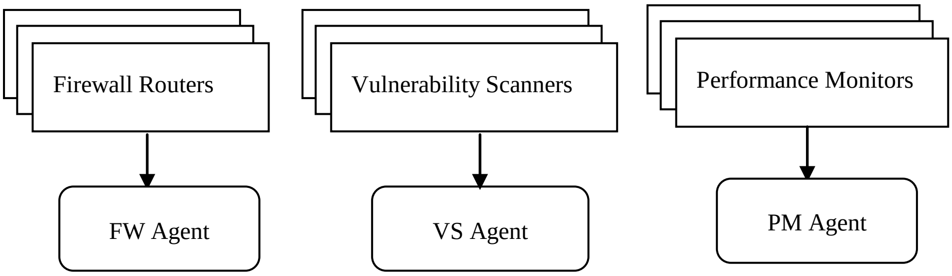
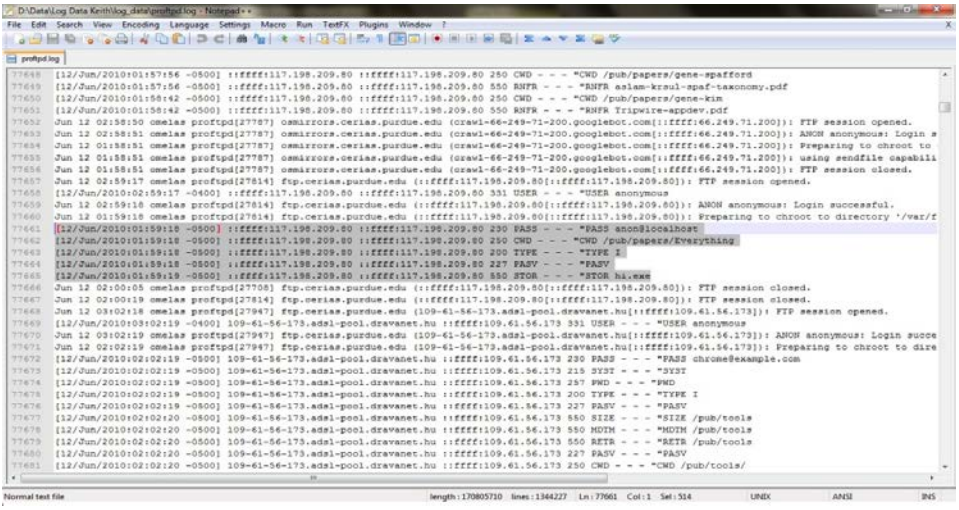


Figure 3.5 INFOSEC Tools Correlation Agents

3.1.3 System and Application Logs Agents

System logs consist of audit log files and application log files. Audit logs contain activities within each system user about his/her transaction with system files, where application log files contain the entries associated with specific services and hosts within the computer system or network. These logs may be either access log or error log within each running application or service. DACM includes local agents to correlate each system log file contents according to specific pattern matches and comparing attack profiles which were previously generated during learning period. These patterns and profiles are generated from a supervised learning process where normal and abnormal log patterns are identified by an operator or by the learning agent.

Figure 3.6 shows the contents of FTP log file as one of possible application logs which can be used within a network. Log file shows the complete session instruction for a specific user and his action since the FTP session opened and his transaction till the end of the session. Detailed description of the log content and whether these contents represent malicious or normal behavior will be described in chapter 4. Figure 3.7 block diagram of different agents for system and application logs files.


D:\Data\Log Data Keith\log_data\proftpd.log - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
proftpd.log

```
[12/Jun/2010:01:57:56 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/gene-spafford
[12/Jun/2010:01:57:56 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 RNFR - - - "RNFR aslam-krsul-spaf-taxonomy.pdf
[12/Jun/2010:01:58:42 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/gene-kim
[12/Jun/2010:01:58:42 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 RNFR - - - "RNFR Tripwire-appdev.pdf
Jun 12 02:58:50 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): FTP session opened.
Jun 12 02:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): ANON anonymous: Login s
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): Preparing to chroot to
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): using sendfile capabili
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): FTP session closed.
Jun 12 02:59:17 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session opened.
[12/Jun/2010:02:59:17 -0400] ::ffff:117.198.209.80 ::ffff:117.198.209.80 331 USER - - - "USER anonymous
Jun 12 02:59:18 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): ANON anonymous: Login successful.
Jun 12 01:59:18 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): Preparing to chroot to directory '/var/f
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 230 PASS - - - "PASS anon@localhost
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/Everything
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 200 TYPE - - - "TYPE I
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 227 PASV - - - "PASV
[12/Jun/2010:01:59:19 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 STOR - - - "STOR hi.exe
Jun 12 02:00:05 omelas proftpd[27708] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session closed.
Jun 12 02:00:19 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session closed.
Jun 12 03:02:18 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): FTP session opened.
[12/Jun/2010:03:02:19 -0400] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 331 USER - - - "USER anonymous
Jun 12 03:02:19 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): ANON anonymous: Login succe
Jun 12 02:02:19 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): Preparing to chroot to dire
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 230 PASS - - - "PASS chrome@example.com
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 215 SYST - - - "SYST
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 257 PWD - - - "PWD
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 200 TYPE - - - "TYPE I
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 227 PASV - - - "PASV
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 SIZE - - - "SIZE /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 MDTM - - - "MDTM /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 RETR - - - "RETR /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 227 PASV - - - "PASV
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 250 CWD - - - "CWD /pub/tools/
```

Figure 3.6 FTP Log Files[2]

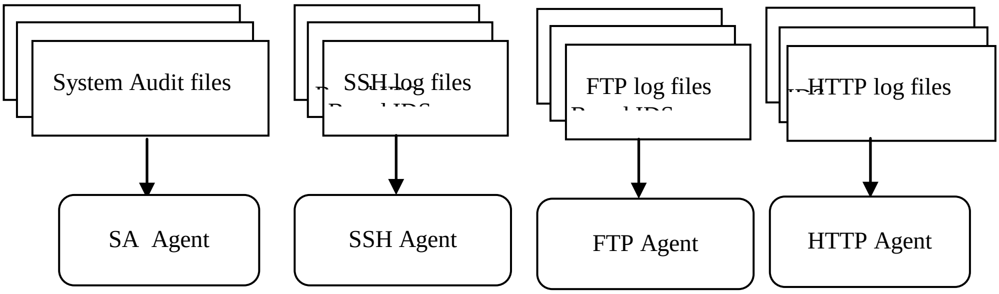


Figure 3.7 System and Application Logs Correlation Agents

[2] Larger image of Figure 3.6 is included in Appendix A

Service and application logs have formal description in which they represent mathematical relation to determine the attack signature in their log files, FTP Agent[3] as an example of these agents can be formally described as follows***:***

```
$A_{FTP} \in$ FTP alerts
FTP $_{Entry}$ (IP, Date, Time, Command, User) $\in$ FTP Log
{S}: set of unauthorized FTP commands; {U}: set of unauthorized users
$\forall$ Entry $\in$ FTP Log
If command (FTP $_{Entry}$) $\in \{S\}$ or user (FTP $_{Entry}$) $\in \{U\}$
Then FTP Entry is malicious, Produces AFTP
FTP $_{Entry}$ (IP, Date, Time, Command, User) $\longrightarrow$ FTP Attack Table
Else
Read next FTP $_{Entry}$
```

3.1.4 DACM Central Agent

Main central agent correlates alerts from IDS's with outputs from other local agents from other information sources. This agent is the heart of the model it provides better understanding of the network. Each local agent aggregates/correlates events from its source and modifies it to standard alert format and stores these results in its own table for main agent. Each agent has specific function and data to extract depending on its source of information and taking into account the network nature like impact analysis and prioritization. The output correlation of the central agent represents the final intrusion reports provided to security admin. These reports include even summary results or detailed intrusion attempts.

Standard alerts representation is used where the alert name and attributes are stored in a table as illustrated in Figure 4.8. Correlation between alerts from different sources is done based on a similarity function for the source of attack, attack type, and near time stamps.

[3] Formal description other individuals agents and central agent are presented in Appendix B

sid	cid	Sig_id	sig_name	timestamp	ip_src	ip_dst	proto	sport	dport

Figure 3.8 Standard Alert Attributes

3.1.5 Formal Description for Central Agent

Formal description is a method of presenting software systems in a way to facilitate further analysis for several metrics as completeness and correctness. In this section, a formal description for the central agent[4] is given as an example to show the mathematical formula used in the agent.

$A_i \in IDS\ alerts$, $A_f \in Firewall\ alerts$, $A_L \in log\ alerts$;

A_i (source, time, destination, type) $\in IDS\ alerts$
A_F (source, time, destination) $\in Firewall\ \ alerts$
A_L (source, time, destination, type) $\in Logs\ alerts$
$\forall\ alert\ \ A_i$
A_i Is verified alerts w.r.t. A_F
If source (A_i) = source (A_F) And Destination (A_i) = Destination (A_F)
And |Time (A_i) – Time (A_F)| <= $T_{threshold}$
Where $T_{threshold}$ is the minimum allowed difference time

A_i Is verified alerts w.r.t. A_L
If source (A_i) = source (A_L) And Destination (A_i) = Destination (A_L)
And |Time (A_i) – Time (A_L)| <= $T_{threshold}$
Where $T_{threshold}$ is the minimum allowed difference time

A_i Is IDS only
If attributes (A_i) < > attributes (A_L) OR
Attributes (A_i) < > attributes (A_F)

$A_{i\ Is}$ Low and Slow attack

[4] Formal description of Other individuals agents and central agent are presented in Appendix B

A_i is IDS only and Count (source [A_i]) = 1 per day

And days (source [A_i]) > 3

$\forall$ *alert* $A_{L,}$

***A_L* is negative** alert w.r.t. A_I

If source (A_i) = source (A_L) and

Time (Ai) < > Time (A_L)

Or attributes (A_L) < > attributes (A_i)

***A_L* is reconnaissance**

If count (A_L) > A_{Th}

Where Al is access count of specific IP / day and

A_{TH} : allowed threshold access per day

3.1.6 Response Agent

The response agent is responsible for the suitable action against the attacker. The response agent interacts with the main central agent to respond depending on the final report. The final report contains summary of correlated attacks and the response agent suggests suitable response against these attacks. The attack response matches are included in specific tables according to the knowledge base in the system depending on historical behavior or learning systems. The implementation of the response agent is not included in this thesis and could be considered as important topic for future work.

3.1.7 Learning Agent

The proposed model has learning capability through learning agent which learns the precondition and post condition of new attacks as well as needed learning from other sources, in log files which of the log contents could be considered as attack signatures and which is considered normal signature. The model support adaptive learning by providing the contents which has not been previously indicated as either an attack or a normal signature. These contents are classified into three different types; similar to attack, similar to normal and unknown. Later, the system administrator can convert any of these types to either a normal or an attack signature.

To enhance learning capability and trace attacker behavior, honey pot agent could be used to learn new attacks and build attack profiles for more accurate knowledge base. A honey pot [62] is a trap set to detect or deflect attempts at unauthorized use of information systems. It consists of a computer, data, or a network site that appears to be part of a network but which is actually isolated and protected, and which seems to contain information that would be of value to attackers. In addition, learning capability could be extended to include learned attacks through sharing information with other external knowledge bases of similar systems.

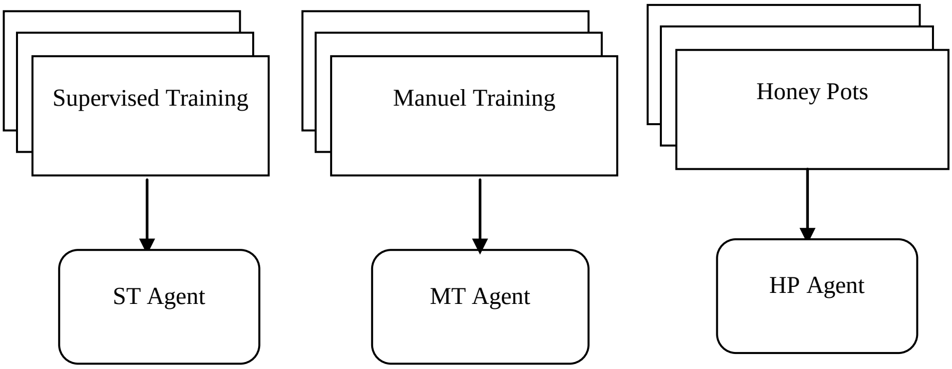


Figure 3.9 Learning Agents Block Diagram

3.1.8 The Knowledge Base and Security Policy

The knowledge base and security policy information represents the network nature and the needed authorization and behavioral profiles information. This information could be used by the individual local agents and the central agent to discover the related attacks. These information are saved in database tables which include preconditions and post conditions for multi step attacks, specific learning parameters, normal and attack profiles, attacks response matching, and access control lists which mention system users and their privileges. Some threshold values for profile matching, such as performance measure, time of use, and network reconnaissance measures, are also saved in the knowledge base.

3.2 DACM Components

DACM components structure is shown in Figure 3.10, it consists of two levels. In the first level a set of agents is presented. This set represents model components, some of which represent local correlation components within IDS, other INFOSEC tools, or system log files. And the learning agent represents the learning capability in the model. Each correlation agent reads data from its source and matches it according to a specific template. A template is a particular pattern used in pattern recognition; it could be a characteristic pattern of attack by an individual or group of attackers. DACM agent's algorithms are smart to avoid correlating important alerts; the new unknown alerts will be moved to second phase for further correlation and more analysis.

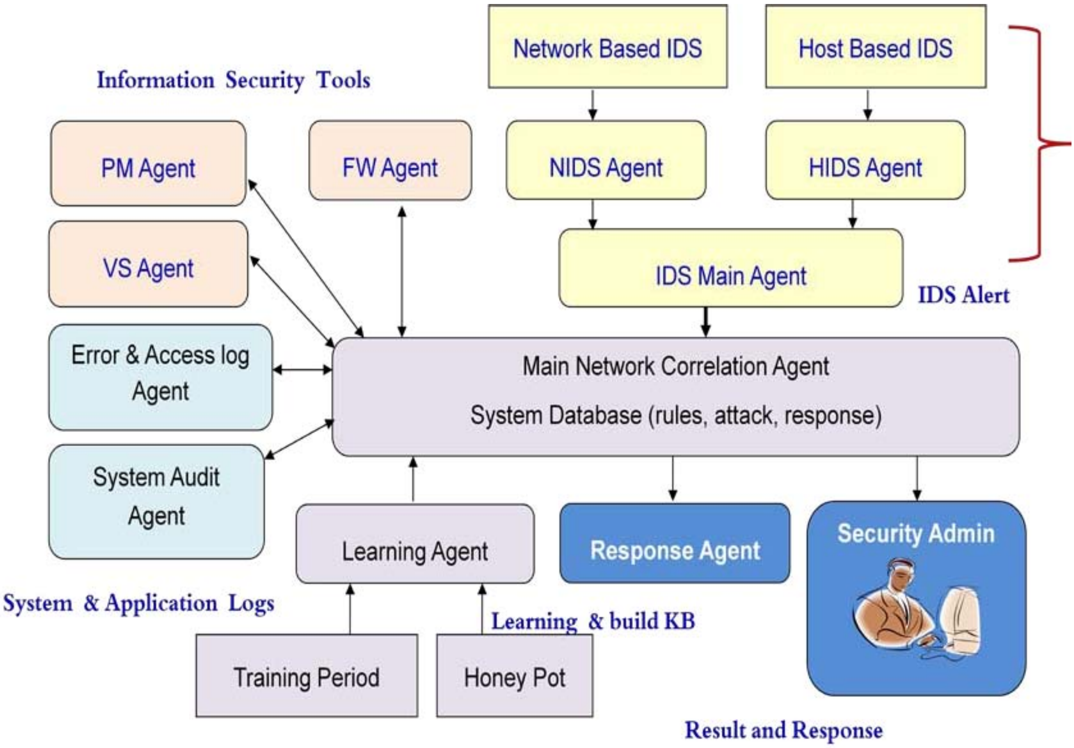


Figure 3.10. DACM Components Structure

In the second level, the main correlation agent is considered as the central agent of the model. This agent correlates the outputs of other agents to provide the whole picture of the network to the security administrator. It can also provide the response agent with the suitable automated response action against the detected attacks according to predefined rules.

3.3 DACM Knowledge Base

The model has a central database which consists of set of tables representing knowledge base and alerts results from different detectors and finally output tables for the security administrator. The output tables include final correlated alerts results and other decision tables for more learning and enhance the knowledge base tables.

3.3.1 System Parameters and Role Base Tables

The system parameters and role base tables include the required information for agent to distinguish between the attack signature in related log files and information sources. For example if we have FTP service within a network , it is needed to determine which permission is allowed for FTP users, could they read files or also they can add, store, modify, and delete files to FTP directory. The FTP write files command will be considered signature for attack if the users are not allowed to write files and could be considered normal behaviors if they are allowed to write files to FTP directory.

Threshold values for system parameters determine which cases could be malicious and which could be normal, network performance monitor measure could be 80 % or more during daily work hours, but if this measure is the same during the weekend or after midnight at 2:00 AM, so it is a signature that something wrong in network traffic is happening. Network asset tables include the network assets, operating systems and ports.

3.3.2 Alerts Table

Alerts tables are divided into two kinds: one for individual alerts which will be kept for long time for low and slow attack detection and another table for correlated alerts, both tables have the same structures as shown in Figure 3.11, the attributes are: Sensor ID, Alert id, attack type, Timestamp, Source IP, Destination IP, and Correlation type

Sensor ID	Alert ID	Attack Class	Timestamp	Source IP	Destination IP	Correlation Type

Figure 3.11 IDSs Correlated Alerts Table Attributes

3.3.3 Vulnerability Scanner

Port scan output reports determine which port could be vulnerable to attack as a destination port in alerts, scanning for vulnerable ports within the network will be executed periodically. Result will be saved to vulnerable ports table as shown in Figure 3.12, it includes: Date, Time, Port ID, protocol, Port status (open, warning, closed), Port service.

Date	Time	Port ID	Protocol	Port Status

Figure 3.12 Vulnerability Scanner Alert Attributes

The integration between this table and alerts tables verify the vulnerable ports with destination ports in alerts and triage between the false positive and accurate alerts. Figure 3.13 shows an example of Nessus tool output

```
poly-app-1.cerias.purdue.edu general/icmp Security notes found
Nmap    scan    report    for    kargad.cerias.purdue.edu
(128.10.252.9)
Host is up (0.0010s latency).
Not shown: 978 filtered ports
PORT      STATE  SERVICE
22/tcp    open   ssh
80/tcp    open   http
113/tcp   closed auth
443/tcp   open   https
```

Figure 3.13 Nessus Output for vulnerability scanner

3.3.4 Performance Monitors Tables

Asset performance increase or decrease compared with threshold values (performance-time) compared with normal system behavior or even by some special cases of system stress without attacks. Figure 3.14 shows performance monitor alert attributes: Asset ID (CPU, Memory, DISK IO, and Network IO), Date and time, Performance (%), Performance Type (high-low-normal) compared with threshold values for normal behavior.

Asset ID	Date	Time	Performance %	

Figure 3.14 Performance monitor Alert Attributes

3.3.5 Firewall Log Files Tables

Firewall log file contains blocked IP packets according to specific access control lists, depending on network security policy. Firewall log agent read blocked IP packets within a specific time compared with other source of information; like IDS or other log files. Figure 3.15 shows an example of log entry of firewall log file.

```
May  9  00:02:21  cisco3.cerias.purdue.edu  592997:
592983:   .May   9   00:02:20.515   EDT:   %SEC-6-
IPACCESSLOGP:     list     120     denied     tcp
94.125.182.255(6665)  ->  128.10.244.160(1094),  1
packet
```

Figure 3.15 Firewall Output log file

The log agent stores its result in a table with attributes shown in Figure 3.16 which includes: *date*, *time*, *service*, *source IP*, *source port*, *target IP*, and *target port*.

Date	Time	Service	Source IP	SPort	Target IP	TPort

Figure 3.16 Firewall Alert Attributes

3.3.6 System Audit Files Tables

These tables include data about; write files, copy files, and move files to external network. These data are compared with specific profiles depending on access control policy and time of events. This comparison is helpful against indoor and outdoor attacks. The audit table could include a set of attributes for such activities as shown in Figure 3.17. These attributes are: *the date, action type (write-copy-move), file type, file name, file location, user ID, user password,* and *user IP.*

Date	Command type	File			User		
		Type	Name	Location	Name	Password	IP

Figure 3.17 System Audit Alert Attributes

3.3.7 Services Log Files Tables

Internet web sites provide different services for their users. FTP and SSH are examples of these services. Web site users explore and use different pages and web forms, users' activities through websites are saved in error log files and access log files. Users' activities could be normal behavior or malicious behavior. Depending on learning period, a set of attack profiles and normal profiles have been implemented. These service and applications log agents read these logs and compare them with the related profiles. When it matches any attack profiles, it stores the related attack into its result tables as shown in Figure 3.18. The alert attributes includes: *date, time, service, type of attack,* and *attacker IP address.*

Date	Time	Service	Attack Type	IP Address

Figure 3.18 Services Log Alerts Attributes

3.3.8 Output Tables:

Output tables summarize correlated alerts from different sources to address attacks from different IPs within a period. It includes the attacker's IP, date, and the detailed attacks from this IP. These alerts could be received from different agents such as an IDS agent, a firewall agent, SSH attack, an error log attack, and its scanning activity for the network. Output tables also include summarized information about false negative alerts and verified alerts.

Supervised learning decisions table contains unknown behavior with related log from IDS and other logs for the same IP to support adaptive learning capability.

3.4 DACM Features

The main purpose of the proposed model is to enhance the IDS accuracy and completeness by reducing both false positive and false negative alerts. It gives better situation understanding within the protected network. For example, in such attacks which are composed of many steps such as Multi-Step Attacks (MSA) or Attack Session Reconstruction (ASR); if any alert of the attack steps was missed it will be just partially matched not fully matched. As a result, the detection of the actual attack scenario or type will be limited. DACM could be used to detect missed alerts to reduce false negative alerts (missed alarms).

The learning agent learns the precondition and post condition of multi-step attacks. For example, alerts Al1, Al2 and Al3 represent three steps of multistep attack. Alert 1 has both precondition and post condition pairs AL1 (P1, S1), Alert 2 Al2 (P2, S2) where p2=s1, and Alert3 Al3 (P3, S3) where p3=s2. IDS may miss critical events that prevent matching the proper attack scenario which produces false negative alert. In case of detection alerts Al1, Al3 provides us with only a partial match of the multistep attack not the complete detection of MSA attack. Using DACM will help detect Al2 from other sources (firewall, Log file, etc…). DACM will enhance the partial match correlation assurance and help in detecting related attacks which did not have explicit relations.

The model may detect zero day attacks when detecting anomalous behavior compared with specific profiles using different sources of information. DACM enhances Alert Verification (AV) through correlation of alerts from IDS’s and other tools which reduce false positive alerts. It also determines potentially malicious sources of traffic compared with legitimate ones.

DACM has a chance of early discovery of the Advanced Persistent Threat (APT). APT has the same sequence of normal attacks but should be delayed and coordinated within a long period of time. Detecting the gathering of data is the initial in APT steps, through the access log files. DACM will keep individual alerts within a long period of time in special tables for detecting low and slow attacks. Keeping these individual alerts for a specific period depends on the attacked service. For example, for port 80 we would have these alerts running for three months instead of a year if the SSH service was the one being attacked. Later, correlating individual events and alerts from different resources occurring over a long period of time and comparing it with normal behavior should detect low and slow attacks. DACM may detect suspicious behavior with less precision than in specific attack detection (often in the grey area between attacks), network problems, and user misconduct.

Using multiple sources of information will help in detection of an unknown worm that generates abnormal traffic and a number of atypical connections to formerly unused ports and destinations. In addition, it enables detection of suspicious access, such as a user making a persistent connection to an administrative port for the first time. Excessive traffic with anomalous destinations and uses, and unknown attacks can be identified according to anomalous activity generated by the attacker. That of which can be detected by monitoring indicators of general user activity such as ports, services, traffic, times, etc. for all users (indoor and outdoor).

DACM can detect malicious connections in comparison to legitimate ones. The existence of the same source IP address in firewall log file, IDS, and other log files within the same time period indicates that it is a malicious connection. We now have to block this connection as a source. DACM can determine a potentially malicious source of traffic compared with legitimate ones. Repeated port scans and network traffic in different times for a long period, and sources IPs or output packets for specific IPs without host names may indicates these IPs as malicious sources of traffic.

DACM decreases the time cost required to obtain effective situational understanding. It also increases the coverage of the attack space and improves the ability to distinguish the serious attacks from the less important ones. It also distinguishes between the ones that require immediate reaction and others where an alternative is acceptable.

3.5 Implementation Scope and Performance Enhancement

The implemented model does not include all the previous described agents; it includes a set of agents representing the different types of correlation because of the nature of collected data and the scope of this research. The implemented model includes the required component to prove the research concept. Network based IDS correlation agents have been implemented as an example of an IDS correlation agent while a host based IDS correlation agent was not implemented. A firewall agent has been implemented as an example of INFOSEC tools agent while vulnerability scanner and performance monitor agents were not implemented.

Correlation agents for error and access log files for different services within the network have been implemented as an example for system log files, while the system audit files correlation agent was not implemented. Supervised training with support of system admin has been implemented as an example of learning capability, while learning using honey pot was not implemented. Finally response agent was considered out of the current scope for this research. It will be an interesting research topic for future work. Analysis of packet dump of the network during the period of collecting data was performed manually with Wire Shark tool [63]; automation of capture data packets correlation was considered out of scope for current model implementation. Figure 3.19 shows the highlighted implemented components in this work from entire DACM components.

Several algorithms, parallelization, and enhancement are presented in detail in the next chapter for the sake of performance enhancement.

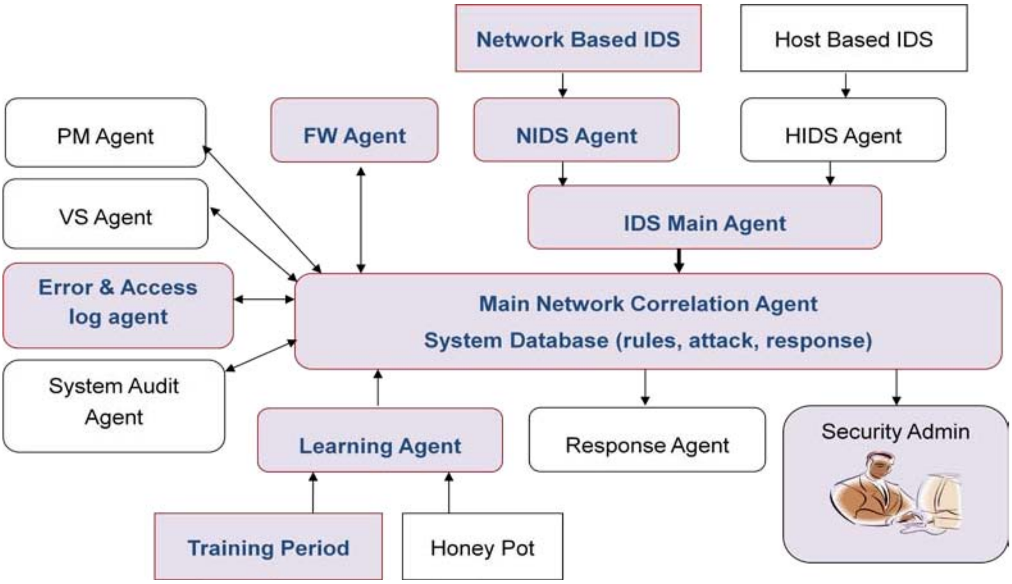


Figure 3.19 Implementation Scope of DACM Components

CHAPTER 4
DACM Design
and Algorithms

Chapter Four: DACM Design and Algorithms

In this chapter different individual agents and central agent implementation will be demonstrated. Different agent's algorithms for alerts and events correlation are presented.

4.1 IDS Alert Correlation

In this section, two IDS alerts correlation techniques are presented to enhance the correlation process presented in Comprehensive Approach Model (CAM) [56 - 58]. CAM results showed that the average time used to process one alert by different components varies depending on used dataset. Some components need more time to process one alert in a dataset while it needs shorter time to process one alert in another different dataset.

4.1.1 IDS Alert Correlation Performance Analysis

Comprehensive approach model [56 - 58] for IDS alert correlation was produced as integrated correlation components which include different sequential correlation components. Figure 2.1 showed CAM correlation components. Results of the reduction rate for each component against different datasets in CAM are presented in Table 2.3. This analysis showed that the sequence order of the correlation is not ideal and many components have not been used for most of the datasets which increases the correlation time needed to obtain effective correlation report for security administrator.

The correlation performance is measured by reduction rate and correlation time, the optimum correlation process has highest reduction rate in lowest correlation time.

Consider a N input alerts and O output alerts as a result of the correlation process, the reduction rate is defined as:

$$\text{Reduction Rate (RR)} = 1\text{-}\ \frac{\text{O}}{\text{N}}$$

For component (i), RR_i: Reduction rate by Component i is defined as:

RR_i = 1- (O_i/N_i)

The Total Reduction Rate:

$$\mathrm{RR} = \prod_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{RR}_{\mathrm{i}} \qquad (4.1)$$

Equation 4.1 represents the total reduction rate of the model components. For the ith component, T_i: is the total time taken by component i to perform correlation and is a function of the count of input alert and time taken to analyze each alert.

T_i = f (c_i,N_i)

$$\text{Total correlation time: } \mathrm{T} = \sum_{\mathrm{i}=1}^{\mathrm{n}} \mathrm{T}_{\mathrm{i}} \qquad (4.2)$$

The correlation time used in CAM model is represented in equation 4.2 which represent the sum of correlation time of all components even if they do not have effective reduction rate value.

4.2 Modified CAM Time

To eliminate the use of components with zero reduction rate affect, and to have optimum order of correlation components such that the components with higher reduction rates can be used before other components with lower reduction rate, we will assume the activity variable Xi is a Boolean variable that could be zero or one as follows

$$\mathrm{X}_{\mathrm{i}} = \begin{cases} 0, & \mathrm{RR}_{\mathrm{i}} = 0 \\ 1, & \mathrm{RR}_{\mathrm{i}} > 0 \end{cases}$$

Giving the condition that RR_i > $RR_{i+1,}$

The above condition determines the sequence of correlation components, with the minimum total correlation time. This sequence requires the components which have higher reduction rate to be used first before the components with lower reduction rate values. Modifying each component time Ti by its activity variable Xi eliminates component with zero reduction rate.

$$T_{opt} = \sum_{i=1}^{n} T_i X_i \tag{4.3}$$

Equation 4.3 represents optimum total correlation time depending on the used components and datasets. The correlation time will be calculated for effective components which have reduction rate greater than zero value. The enhancing of the correlation process can be obtained by calculating the reduced time. It can be represented by the difference in time between calculated T in equation 4.2 and calculated T_{opt} in equation 4.3 as follows:

$$T_{diff}=T-T_{opt} \tag{4.4}$$

4.2.1 Agent Based Correlation Model

Figure 4.1 shows the proposed model which presents an Agent Based Correlation Model (ABCM) for Intrusion Detection Alerts. In this model Learning Agent (LA) learns the nature and characteristics of normalized alerts produced by different IDSs within a network, and then it selects the suitable correlation components that can be used and their proper order.

The model provides minimum correlation time for all datasets whatever their nature. ABCM consists of two phases, learning phase and correlation phase. The input of ABCM is normalized and pre-processed alerts while the output goes to a set of selected correlation components called Active Correlation Components List (ACCL).

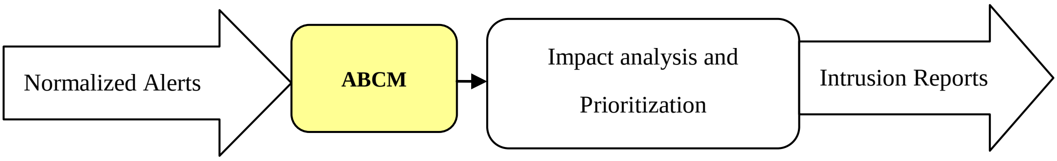


Figure 4.1 ABCM correlation model block diagram

The selection of added components in ACCL depends on agent learning, the output alerts correlated by ACCL is directed to the last two components of correlation process. This model is based on the real-time CAM [56 - 58]. However, instead of using sequence of all correlation components, it uses an optimal ordered set of specific effective correlation components depending on agent learning.

4.2.1.1 Learning Phase

During the learning phase, LA learns the output of each component and the dataset nature. Based on this learning beside a set of rules and knowledge base as well, LA can determine the active correlation components and their proper order. Each correlation component has specific criteria to aggregate and correlate alerts. The knowledge base for learning is formed by the criteria for each component in addition to the RR obtained by each component

The learning phase starts through the execution of initial correlation process as sequential learning as shown in Figure 4.2. In sequential learning, the initial components sequence order could be as described in Comprehensive Approach Model [56 - 58], or it could be random sequence order. Figure 4.2 shows that LA learning depends on initial inputs. These inputs are: the learning parameters which could be a period of time (t) or specific number of alerts (N); the normalized pre-processed alerts; and a pre generated knowledge base. Each component aggregates and merges its input alerts according to component algorithm and criteria. Alerts attributes (source, attack type, destination) are used as the basis of merging alerts.

RR can be calculated through comparing output alerts with input alerts. Depending on the value of RR for each component, LA builds ACCL which contains the components with RR higher than zero value. The learning phase could be processed in parallel learning as a separate process; parallel learning is shown in Figure 4.3.

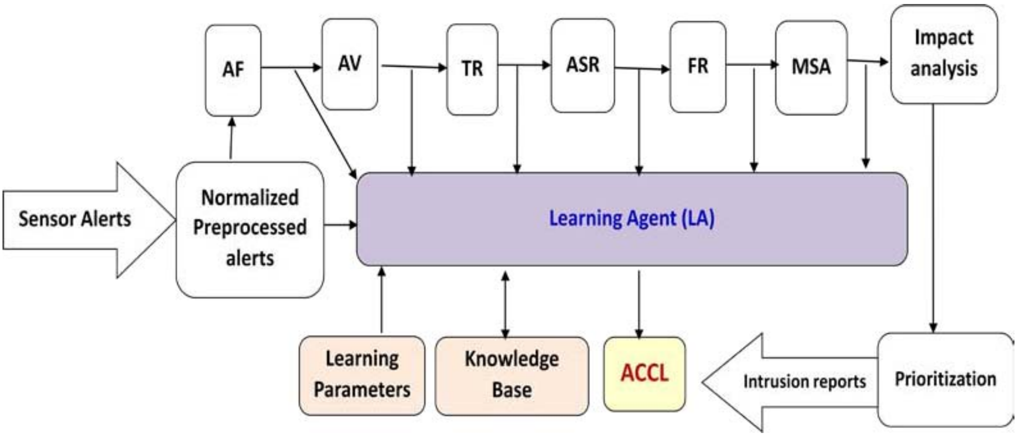


Figure 4.2 ABCM sequential learning Phase

The correlation of all components has been done in parallel and learning agent to get the correlation result of each component separately. The learning parameters determine learning period or count of learned alerts.

Learned alerts have been selected randomly during learning phase, and they are excluded in correlation process.

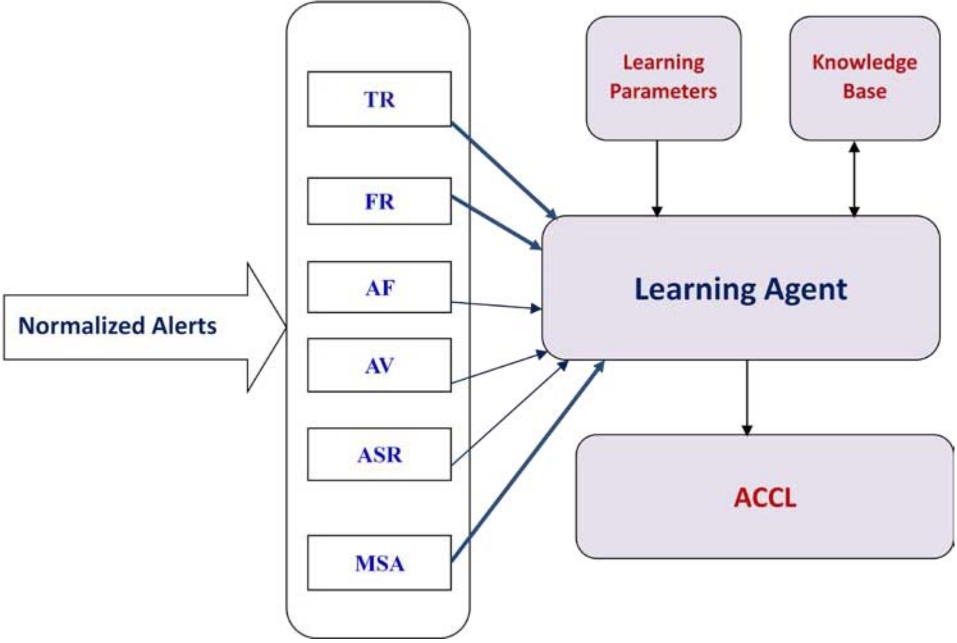


Figure 4.3 ABCM Parallel learning Phase

Figure 4.4 shows the learning result of 2866 alerts which represent a ratio of 10 % of collected alerts in one day (28866) of CERIAS dataset described in chapter 5. Learning results show that ACCL contains components: FR with RR of 92.87 %, TR with RR of 91.82 %, and MSA with RR of 2.17 % while component AF has zero reduction rates.

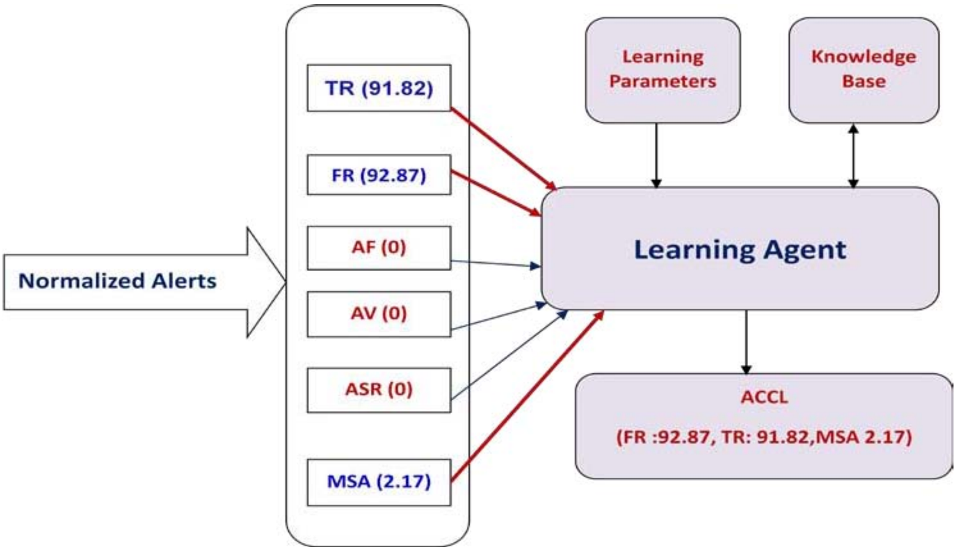


Figure 4.4 CERIAS ABCM Parallel learning Result

Algorithm 4.1 describes the learning phase process; LA builds ACCL in descending order of the component reduction rate. The normalized pre-processed alerts go through their basic correlation path. By the end of the correlation process of each component, LA reads RR of each component. If the RR is higher than zero, the component data will be added to the ACCL which includes serial, component name, and RR value. By the end of correlation of the last component, ACCL will be having a specific set of components with different RR values. The agent sorts these components in descending order of their RR, and it also disables components with zero RR values.

Moreover, LA updates the knowledge base using the new criteria of merging alerts in each correlation component. Learning phase should be enough for studying the nature of alerts in the network. Such phase continues depending on the learning parameter (t or N) and/or assuring no changes of the alerts nature.

Algorithm 4-1 Learning Phase

```
Algorithm 4.1 Learning Phase
Inputs: (IS) normalized and pre-processed stream of alerts, IP: number of input alerts,
learning parameter (Alerts number N)
Output: (ACCL), set of active correlation components (RRc > 0)
Initialization: Empty ACCL (Ser, CC, RR) ACCL (0,  ,0), k=6 (maximum number of
correlation components), m=0
For alerts in N
    While k > 1 do  // For each component do
        OSc  ←  CORRc(IS);
        OPc  ←  no of alerts in OSc
        RRc  ←  (1-OPc/IP)*100
        if RRc > 0 then
                 begin
                ACCl(Ser) ← ser+1;
                    ACCL(RR) ← RRc;
                ACCL(CC) ← CC;
                 end
        Else
        // For each component with RRc=0
           begin
             Disable component;
             m  ←  m+1;
                end
        end if      / / end if RR>0
      k ← k-1;
      loop
      end while
    sort ACCL(RR, descending);
end for
return ACCL;
```

LA could be a part of the correlation process by eliminating some alerts depending on the network nature (alerts against windows server while maintaining UNIX server).

4.2.1.2 Correlation Phase

By the end of learning phase, ACCL contains only effective correlation components in descending order of their RR. In the correlation phase, the flow of normalized alerts stream will be controlled by the agent. Alerts are directed to the first component in ACCL which has the highest RR during the learning phase. The output of the first component will be the input of the second one which has the second highest RR, and so on till they reach the last component in ACCL.

Figure 4.5 describes correlation phase of the normalized pre-processed alerts. Alerts are correlated using one path of many alternative paths. These alternative paths represent different suggested ACCLs which have been implemented previously during the learning phase. For example, the analysis of sample of CERIAS dataset correlation shows that ACCL has only FR and TR (Highlighted Boxes in Figure 4.5) with RR values (FR=92.87, TR=91.82, and MSA=2.17), While the other three components AF, AV, , and ASR have no effect on that dataset.

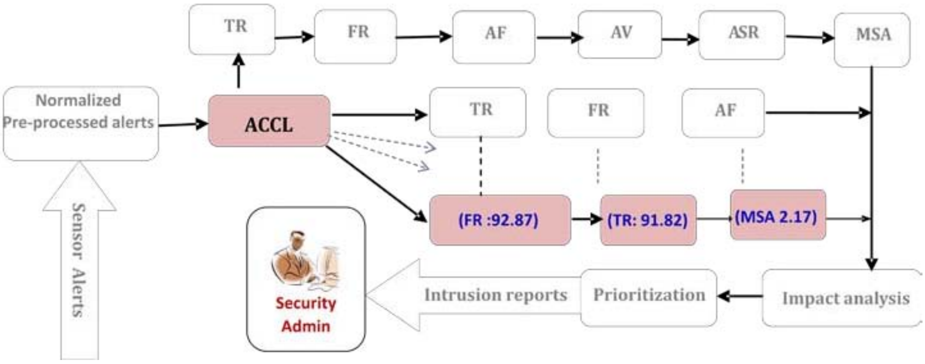


Figure 4.5 ABCM Correlation Phase

Algorithm 4.2 shows the correlation phase process; the inputs of the algorithm are: ACCL, and array of alerts which represent the remainder alerts after excluding the learned alerts in the learning phase.

The correlated alerts OCc are considered to be the output of the algorithm. The agent uses the first component in ACCL to correlate the input alerts, and then it moves the pointer of ACCL to the next component, the next components correlates the output alerts of first component, the loop continues till using all components in ACCL.

Algorithm 4-2 ABCM Correlation Phase

```
Algorithm 4.2 Correlation Phase
Inputs: (IS) normalized and preprocessed stream of alerts, IP: number of input alerts,
ACCL (Ser, CC, RR)
Output: (OS) correlated stream of alerts,
Begin
            While ACCL (ser) > 0 (is not empty) do
       // loop for all components in ACCL
       Begin
              CORRc(IS) using ACCL(CC);
              OSc $\leftarrow$ CORRc(IS);
              OPc $\leftarrow$ no of alerts in OSc
              RRc $\leftarrow$ (1-IP/OPc)*100
              ACCL(CC) $\leftarrow$ next ACCL(CC);
              // next lowest RR component in ACCL
            Loop // all components in ACCL have been used
   end while
    OS $\leftarrow$ OSc of last component in ACCL
end
return OS;
```

The total correlation time by ABCM is calculated as follows:

$T_{ABCM} = T_{learning} + T_{correlation}$ where $T_{correlation}$ of ACCL components as optimal serial sequence without unneeded components and in proper order is as follow:

$$T_{correlation} = \sum_{j=1}^{n} t_j$$

Where n is the count of ACCL components.

Total T_{ABCM} is much lower than total correlation time by CAM.

4.2.2 Dynamic Parallel Correlation Model

This novel model presents a Dynamic Parallel Correlation Model (DPCM) for Intrusion Detection Alerts; the model dynamically selects optimum correlation components arrangement order and provides minimum correlation (for all datasets, whatever their nature is). DPCM is a part of the entire correlation process as shown in Figure 4.6. The input of DPCM is a stream of normalized alerts while the output of DPCM will be the input of the rest of correlation components process.

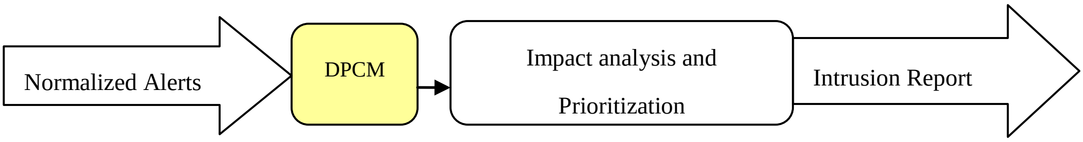


Figure 4.6 DPCM Block Diagram

The input of DPCM is normalized and pre-processed alerts. Figure 4.7 shows that DPCM is composed of a set of correlation stages, each stage contains k parallel correlation components (k=6) (AF-AV-TR-ASR-FR-MSA), the input of every stage is directed to all active components in this stage simultaneously.

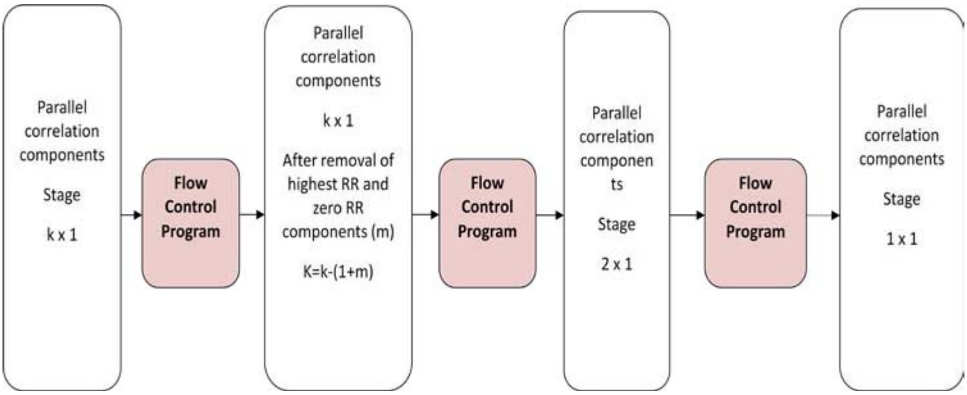

Parallel correlation components
Stage
k x 1
Flow Control Program
Parallel correlation components
k x 1
After removal of highest RR and zero RR components (m)
K=k-(1+m)
Flow Control Program
Parallel correlation componen ts
Stage
2 x 1
Flow Control Program
Parallel correlation components
Stage
1 x 1

Figure 4.7 DPCM Correlation Stages

The model assures that alerts go through only effective correlation components. The correlation criteria are different for each correlation component, since all components have their independent correlation and they can work in parallel independent of each other.

The components arrangement will be dynamically changed in descending order depending on the RR of each component. This model is based on the real-time correlation model. However, instead of using sequence of all correlation components, a set of correlation stage will be used. Each stage contains all effective correlation components in parallel manner.

DPCM creates a thread for each correlation component to perform synchronous correlation; all threads access alerts data at the same time, each thread process its dedicated correlation method and creates a list of correlated alerts. The counts of threads depend on the count of active component (k) in each stage. Using threads optimize processor and memory usage, all threads on the same process can access list of variables in memory at the same time, no need for semaphores to read variable.

Algorithm 4.3 shows the contents of DPCM algorithm which describes how it works, the input is a stream of normalized alerts (IS) and the output is a correlated stream of alerts (OS). In the program initialization, all components have been set to active state (k=6) and set zero value for count of components with zero RR values (m=0). The program reads the inputs stream alerts in the first correlation stage and reads the RR ratio of alerts obtained by each component. Depending on these RR results the program decides which components will be used in the next stage. The output of component with highest RR will be the input of the next correlation stage. The component with higher RR and components with zero values RR (m) will be disabled in the next stage.

The active components in next correlation stage will be calculated again by k=k-(1+m). In next stage the active components will reduce the input alerts stream each with a specific RR ratio. The loop continue till k=1 where all correlation stages used either by going through all six stages or specific set of them depending on values of (m) during the flow of alerts.

Each component in the correlation stages aggregates and merges its input alerts according to component algorithm and criteria. Alerts attributes (source, attack type, destination) are used as the basis of merging alerts. RR can be calculated through comparing output alerts with input alerts.

Figure 4.8 shows CERIAS alert dataset sample correlation using DPCM. All active components in first stage simultaneously correlate the input of normalized alerts stream. The results of first stage shows that three components (AV, ASR, and AF) have zero RRc values (m=3). Component TR have highest RR value (FR=92 %) and RR values of (TR=90 %, MSA=0.38 %). With k=6, and m= 3 number of active components in next stage is k=6-(1+3) = 2. In next stage the algorithm disables highest RR (FR) and zeros RR components (AV, ASR, and AF). The active correlation components in second correlation stage are TR and MSA.

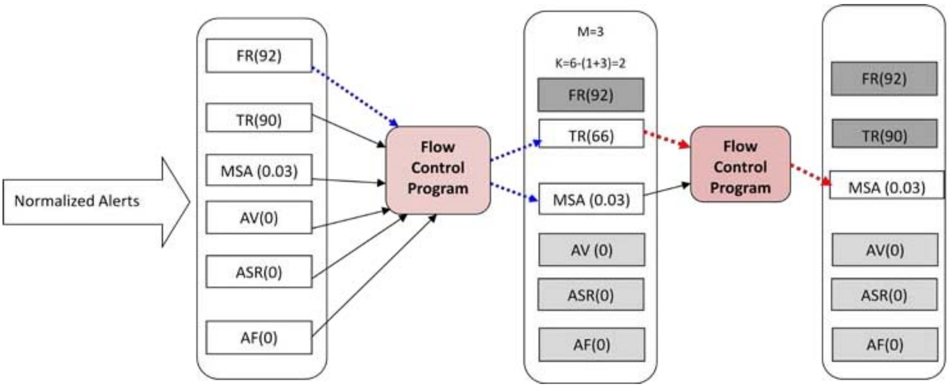


Figure 4.8 CERIAS DPCM Correlation Example

Both components will correlate the output of TR component from first stage. The RR of active components in second stage will be calculated again with values (TR=66 %, MSA=0.3%). The output of this stage is the correlated alerts by TR component {higher RR than MSA}. The program passes the output correlated alerts from TR to next stage and disable FR in next sage and recalculate k=2-(1+0)=1. The third stage has only MSA active component with RR = 0.03%. It correlates its input alerts and recalculates k = 1- (1+0) = 0.

Algorithm 4-3 DPCM Algorithm

```
Algorithm 4.3 DPCM Algorithm
Inputs: (IS) normalized and preprocessed stream of alerts,  IP:  input alerts
Output: (OS) correlated stream of alerts, OP: number of output correlated alerts
Initialization: k=6 (maximum number of correlation stages), m=0 (component with zero
RR), all components in active state.
Begin
While $k > 1$ do
        Begin       // For each component in active list
            Begin
                OSc $\leftarrow$ CORRc(IS);
                OPc $\leftarrow$ no of alerts in OSc
                RRc $\leftarrow$ 100 (1-IP/OPC)
            End
      if RRc = 0 then
                Begin
                        disable component;
                        m $\leftarrow$ m+1;
                End
            Else
         End if
         OS $\leftarrow$ output of component with max RRc
         k $\leftarrow$ k-(1+m)
         disable component with max RRc
end loop
end
```

This means there are no more active components or correlation stages anymore. The correlated alerts produced by the third stage are the final output of DPCM process. DPCM uses just three components instead of six. The optimum components order was FR, TR then MSA and was dynamically selected in descending order depending on their RR. The total correlation time by DPCM is calculated as follows:

$$T_{DPCM} = \sum_{i=1}^{n} T_i X_i$$

Where X_i represents the active correlation stages, these stages contain only effective correlation components and have dynamic descending order of its reduction rates. Total T_{DPM} correlation time is optimum compared with total correlation time by CAM.

4.3 DACM Individual Agents

In this section DACM correlation agents design and algorithm will be presented, block diagram of DACM components shown in Figure 4.9.

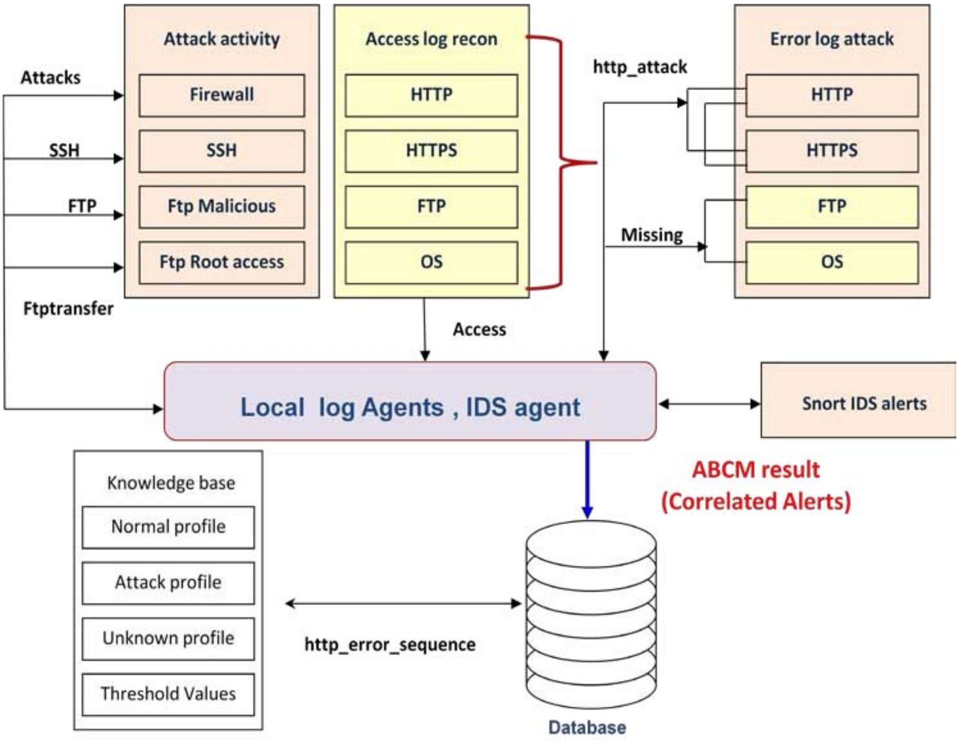


Figure 4.9 DACM Individual Agents

DACM is composed from a set of correlation agents; each agent correlates alerts or events from its information source. Different agent's sources of information including IDS alerts, firewall log file, other services log files. Log files may be error log files or access log files, and finally a set of knowledge base which include network security policy and needed threshold values to determine behavioral profiles and attacks signatures.

4.3.1 IP Address Normalization

Different source of information have been used for retrieving attack signatures, the detected IP address has different format in each source. IDS alerts include decimal format for source and destination IPs, while INFOSEC tools and other application log files include standard 32 bit representation “Standard IPs”. Normalizing IPs together is a necessary process for correlation such information together, this process indicates that every IP address has a unique ID in the system. Algorithm 4.4 performs save IP function to create IPs table, this algorithm transfers any used decimal or standard IP to a unique ID which represents it and could be used within the system. IPs table consists of three fields which are: ID, Decimal IP, and Standard (256 base) IP. ID is a unique id for each IP; Decimal IP is the IP address in decimal format retrieved from IDS alerts where Standard IP is the IP address in standard 32 bit format retrieved from INFOSEC tools and System Log files.

The algorithm read IP address from information sources and creates a record with a unique ID for this IP. If the IP address in decimal format, a convert process could be used to convert it to standard IP and insert a record of this IP in IPS table. If the IP address in standard format, it check if this IP is already has a unique ID, if so it return this ID, if not it creates a new record to assign new ID for this sting IP.

Convert IP function convert the decimal format detected from IDS to a standard format (base 256), the standard IP is composed of 4 parts from left to right. If we have a decimal IP address = “1812014676”, we need to convert it to standard IP address; convert function starts by initiating a loop from 1 to 4, and dividing the decimal format number to (256) ^ *(4 - i)* which produce 108 as first part of standard IP address, looping to i =2 and repeating steps from 2 to 9 till i=4 which produce IP in sting format (base 256) = “108.1.38.84”.

Algorithm 4-4 Save IP Function

```
Algorithm 4.4 Save IP function
Input:  IP standard variable or decimal ip
Output: IP record number
Begin
If format (IP) =decimal Then
        IP_standard = Convert (IP_decimal, IP_standard);
Else
        IP_decimal = null;
End if   // Check if the ip existing in IP table
Result = Select standard IP from `IPS` table where IP (base256) =Standard IP
        If result is true then
                return ID;
                return ;
        Else // result is false
          ID= max (ID) + 1;
          Insert into IPS values (ID, IP_decimal, IP_standard);
          Return ID;
     End if
End;
Return ID.
```

```
Convert (IP_decimal, IP_standard)
        For i = 1 To 4
          num = Int( IP_decimal / 256 ^ (4 - i))
          IP_decimal= IP_decimal - ( num * 256 ^ (4 - i))
          If i = 1 Then
               IP_Standard = num
          Else
               IP_Standard = IP_Standard & "." & num
          End If
         Next I
        End for
        Return IP_Standard
```

4.3.2 Firewall Agent

Firewall agent read the contents of router log file, this log file contains list of blocked IP which tried to attack the network, it extracts the data indicating the attack such as date and time of attack trial, the destination protocol, source IP and source port, and destination IP and port. Figure 4.10 shows an example of router log blocked IP entry, the entry include the blocked trial attributes and other detailed information about the protected server and router.

```
Jun 16 22:13:45 cisco3.cerias.purdue.edu 316500: 316485: Jun 16
22:13:44.639 EDT: %SEC-6-IPACCESSLOGP: list rsrchin denied
udp 108.1.38.84(50184) (Port-channel1 001f.9ed2.ba40) ->
128.10.247.62(54045), 1 packet
```

Figure 4.10 Firewall router log contents

Algorithm 4.5 shows the agent process to read the log contents and convert it to a record in the attack table. The input is the router log file, and the output is the attack table record.

The algorithm has some initial variables to be used as static split variables; these variables are static contents in the router log entry. The algorithm starts reading the file contents and checks each line contents; it splits with server name as first static variable then it reads the date and time from the first part of the splitting and continues splitting the second part with other splitting variables to get the other required attributes of attacks such as source IP and Port, protocol, destination IP, and Port.

The algorithm reaches the end of the line and stores the extracted attributes into the related record in the firewall attack table as shown in Table 4.1.

Algorithm 4-5 Firewall Agent

```
Algorithm 4.5 Firewall Agent
Input: router log file.
Output: fill data to database table 'attacks'
Initialization:
                    set server name = "cisco3.cerias.purdue.edu"
            Set SP1 = " %SEC-6-IPACCESSLOGP: " ,
            Set SP2 = " -> ""
Begin
     While not EOF
          For each line
            Read line contents;
                    Split line with static variables name;
                    read date and time;
                    read source IP and port;
                    read protocol type;
                    read destination IP and port;
                    ignore unwanted variables;
                    Store to attack table (date, time, source IP, source Port, protocol,
                    destination IP , destination Port );.
          End for;
     End While;
Return 'attack' table.
```

Table 4.1 Firewall attack table

Date	Time	Protocol	Source IP	SPort	Destination IP	DPort
Jun 16	**22:13:45**	**udp**	**108.1.38.84**	**50184**	**128.10.247.62**	**54045**

4.3.3 FTP local Agents

The first FTP agent algorithm reads the contents of the log file which contains ftp service logs and error messages and requests associated with FTP commands as shown in Figure 4.11. Local FTP agent reads complete session for the user activity in the log file to check the command type tried by the user.

```
Jun 12 02:59:17 omelas proftpd[27814] ftp.cerias.purdue.edu
(::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session opened.
[12/Jun/2010:02:59:17 -0400] ::ffff:117.198.209.80 ::ffff:117.198.209.80 331 USER - - -
"USER anonymousJun 12 02:59:18 omelas proftpd[27814] ftp.cerias.purdue.edu
(::ffff:117.198.209.80[::ffff:117.198.209.80]): ANON anonymous: Login successful.
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 230 PASS - - -
"PASS anon@localhost
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - -
"CWD /pub/papers/Everything[
[12/Jun/2010:01:59:18 -0500]::ffff:117.198.209.80::ffff:117.198.209.80 200 TYPE - - -
"TYPE I
[12/Jun/2010:01:59:19 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 STOR -
"STOR hi.exe
Jun 12 02:00:05 omelas proftpd[27708] ftp.cerias.purdue.edu
(::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session closed.
```

Figure 4.11 FTP log file example

Algorithm 4.6 shows the FTP agent process to read the log contents and check if it contains any FTP command which violates the network security policy. In case of finding an evidence of that violation, it extracts this log entry and inserts it to a record in FTP attack table. The input is the log file which contains user’s commands in FTP server, and the output is the FTP attack table.

Algorithm 4-6 FTP Agent

```
Algorithm 4.6 FTP agent
Input: FTP ( proftpd )  log
Output: fill data to database table ' Ftp'
Initialization: User select not allow events (DELE, MKD, STOR, STOU, RMD, ALLO,
                APPE);
                User type ftp server (ftp.cerias.purdue.edu )
While not EOF
        For each line
          Read line contents ;
                Read command type
                    If command type in list
                    Then
                          Store to ftp table (date, time, source IP, event , Description  ) ;
                    Else
                End if ;
                loop;
        End for;
End While;
Return ‘ftp’ table.
```

The algorithm has initial list which contains the list of prohibited commands in the FTP server. The algorithm starts reading the file contents and checks each line contents; it splits line contents and read the date, time, source, and command type. After reading command type the algorithm check if that the command is included in the not allowed command list; if yes then it stores a record in the ftp attack table. This record includes the date, time, source IP, command, and description of this command. In case of the command is an allowed command, the algorithm continue reading the next line till the end of the user session and the file contents.

Table 4.2 shows the related record indicating that attack and its detailed information as well as command description.

Table 4.2 FTP attack table

Date	Time	Protocol	Source IP	Event	Description
Jun 16	**22:13:45**	**FTP**	117.198.209.80	**STOR**	STOR hi.exe

FTP transfer agent algorithm detects malicious behavior during transfer FTP files. The log file contains a listing of files transferred over FTP, normally the third column should say ftp for all users which indicates FTP user trying to transfer file, while if we have a "root" username and the commands do not appear to be standard as shown in Figure 4.12, it indicates a malicious trial to FTP transfer using root access.

```
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  POST
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root:    USER-AGENT:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  HOST:     -
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  ACCEPT:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root      REFERER:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  PROXY-CONNECTION:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  COOKIE:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  CONTENT-LENGTH:
[11/Jun/2010:01:02:50 -0400]::ffff:66.199.234.66     root  CONTENT-TYPE:
```

Figure 4.12 FTP Transfer log file example

Algorithm 4.7 shows FTP transfer algorithm, it reads the log file as an input while it stores its output in FTP transfer attack table. The algorithm starts by reading the log contents line by line and it splits the line contents to get the required attributes and ignore unwanted characters. In each line it checks the username and file transferred format.

Algorithm 4-7 FTP Transfer Agent

```
Algorithm 4.7 FTP Transfer Agent
Input: xfer log file.
Output: fill data to database table ' Ftp'
Initialization : set  username = " root "
While EOF
        For each line N
          Read line contents;
                Search for username in the line in third column
                If username = “root”
                then
                        split line contents and ignore unwanted characters
                        read date;
                        read time;
                        read IP;
                        return process name;
                        Store to database (date , time , Ip , event , Process name  );
          Else
          End if;
        End for
End while
Return ‘FTP’ Table
```

If the username is FTP and transferred file has the required standard format then it is legitimate user. If the username is root, on ftp transfer file is not standard or trying to get proxy-connection as shown in Figure 4.12, so it is an evidence of malicious FTP transfer and stores related record in FTP transfer attack table. This record includes date, time, source IP, event, and process name as shown in Table 4.3

Table 4.3 FTP Transfer Attack Table

Date	Time	Source IP	Event	Description
Jun 11	**01:02:50**	*66.199.234.66*	Transfer	PROXY-CONNECTION

4.3.4 SSH Agent

SSH (secure shell) agent is an example of service agents. SSH agent reads the contents of SSH service log file that records service usage and error messages from the service and child processes. It shows the attackers' attempts to access SSH service with root user or invalid username or password.

The log includes error messages associated with SSH which identifies IPs and hostnames of people asking to guess users passwords. SSH agent reads the log contents and checks the error messages indicating illegal trials to access SSH through guessing user names and passwords or trying to hide the user browser identification. Figure 4.13 shows an example of some error messages in the log file which shows trial of IP "*222.186.24.122*" to guess user root password or guessing a user name and password for user "oracle".

```
Jun 12 18:23:00 basm.cerias.purdue.edu sshd[16678]: Failed password for root from
222.186.24.122 port 34507 ssh2
Jun 12 18:23:09 basm.cerias.purdue.edu sshd[16686]: Invalid user oracle from
222.186.24.122
Jun 12 18:23:09 basm.cerias.purdue.edu sshd[16686]: error: Could not get shadow
information for NOUSER
Jun 12 18:23:09 basm.cerias.purdue.edu sshd[16686]: Failed password for invalid user
oracle from 222.186.24.122 port 34748 ssh2
Jun 12 18:23:12 basm.cerias.purdue.edu sshd[16688]: Failed password for invalid user
test from 222.186.24.122 port 34800 ssh2
```

Figure 4.13 SSH log file "Inetdlog" example

The SSH agent reads those messages and store their attributes to SSH attack table which contains the date, time, source IP, source Port, and description. Algorithm 4.8 shows the SSH agent process to read the log contents and check if it contains any error messages which violates the network security policy and extracts this log entry and inserts it to a record in SSH attack table.

Algorithm 4-8 SSH Agent

```
Algorithm 4.8 SSH agent
Input: SSH (BASMSSH-inetd) log
Output: fill data to database table ' SSH'
Initialization: user type server name “basm.cerias.purdue.edu”
    user type unwanted string " Invalid , Failed password,  Did not receive identification"
While not EOF
          For each line
            Read line contents; Read message string
                  If message string in list
                      Then
                            split line contents and ignore unwanted characters
                            read date; read time;
                            read IP; read sport;
                            read error message;
                            Store to SSH table (date, time, source IP, Sport , error message ) ;
                      Else
                  End if ;
                  loop;
          End for;
End While;
Return ‘SSH’ table .
```

The input is the log file which contains users messages in SSH server, and the output is the SSH attack table. The algorithm has initial list which contains error messages which indicate malicious behavior within SSH. The algorithm starts reading the file contents and checks each line contents; it splits the line contents and reads the date, time, source, and error message. After reading the error message the algorithm checks if that message is included in the not allowed command list; if yes then it stores a record in SSH attack table. This record includes the date, time, source IP, source port, and the error message as shown in Table 4.4. In case the error message indicates allowed access, the algorithm continues reading the next line till the end of the file contents.

TABLE 4.4 SSH Transfer attack table

Date	Time	Source IP	SPort	Description
Jun 16	**22:13:45**	**222.186.24.122**	**34442**	**Failed password for root**

4.3.5 Error Log Agent

Error log agent reads the contents of the file associated with http and https services. The purpose of the error log agent is to identify attack signatures stored in the http or https error log files by reading these files and comparing their contents with either attack profile or normal profile. These profiles were previously created during the supervised learning period to distinguish which of these log entries was produced by attack and which was produced by normal usage.

Figure 4.14 shows an example of attack profile for IP address “108.1.38.84” during learning period. In case of detecting a new profile in the log files, error log agent checks the similarity of this profile to one of known profiles and identify the new profile as similar to attack or similar to normal.

Later the user administrator can assure this similarity and change the type of profile to attack profile or normal profile.

```
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP   1. {main}()
/var/www/www.cerias.purdue.edu/htdocs/education/k-12/shared/submit_link.php:0,
referer: http://www.cerias.purdue.edu/education/k-12/K-5_Resources/
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP   2. Form->outputForm()
/var/www/www.cerias.purdue.edu/htdocs/education/k-12/shared/submit_link.php:98,
referer: http://www.cerias.purdue.edu/education/k-12/K-5_Resources/
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP 3.ListField->pHtmlField()
/var/www/shared/cerias/class.formdata.php:60,referer:     http://www.cerias.purdue.edu
/education/k-12/K-5_Resources/
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP   4. ListField-
>pScrollingList()          /var/www/www.cerias.purdue.edu/htdocs/education/k-
12/lib/class.formdata.scrollinglist-k12.php:222,
referer:http://www.cerias.purdue.edu/education/k-12/K-5_Resources/
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP   5. renderGroupList()
/var/www/www.cerias.purdue.edu/htdocs/education/k-12/lib/class.formdata.scrollinglist-
k12.php:299,referer: http://www.cerias.purdue.edu/education/k-12/K-5_Resources/
[Tue Jun 22 22:45:24 2010] [error] [client 108.1.38.84] PHP   6. renderOptions()
/var/www/www.cerias.purdue.edu/htdocs/education/k-12/lib/submit_link.func.
php:159,referer:http://www.cerias.purdue.edu/education/k-12/K-5_Resources/
```

Figure 4.14 Error Log Contents

Algorithm 4.9 shows the error agent process to read the log contents and check if it contains any attack signatures which violates the network security policy and extracts this log entry and inserts it to a record in http attack table. The input is the “errorlog” log file which contains user’s messages in http and https server, and the output is the http attack table.

The algorithm starts reading the file contents and checks each user session and its error sequence contents; it splits lines contents and read the date, time, source, and error sequence.

Algorithm 4-9 HTTP Agent

```
Algorithm 4.9 http Agent
Input: http or https error log file, error sequence table.
Output: fill data to database table ' http attack'
Initialization: user select type of file type :     1 – http , 2 – https
While not EOF
            For each line
            Read line contents;
            Get first error line
            Repeat until  error sequence end;
            Read date;
            Read time;
            Read ip address;
            Read error sequence;
            Check the error sequence in  http_error_sequence table ;
            If match
            then
                  get  error sequence code ;
                  get sequence type;
            else
              check similarity;
            end if;
End for;
End While ;
Return  ' http_attack ' table
```

After reading the whole error sequence for one user the algorithm gets the error sequence code and retrieve its type from error sequence table. Finally the algorithm stores a record indicating the user error sequence in the http attack table. This record includes the date, time, source IP, error sequence code, and profile type as shown in Table 4.5.

The following is the algorithm for the check similarity function in algorithm 4.9

```
Check similarity
If error sequence subset of other known error sequence
Then
        Error type is similar to type;
        Insert to http_error_sequence table ;
Else
        Error type is unknown;
        Insert to http_error_sequence table ;
End if;
example:        A - " {main}(),include() "
                B-" {main}(),include(),PageDef->show(),CPL:: checkInternalIP() "
                If error B stored as normal behaviour the error A  is part of B then A
                will have type "similar to normal "
```

Table 4.5 shows the result of malicious behavior by IP 108.1.38.84. Sequence “6” is the error sequence code stored in http error sequence profile and type “1” represent that the type of this profile is an attack profile.

Table 4.5 HTTP Attack Table Record

Date	Time	Source IP	Sequence Code	Type
Jun 16	**22:13:45**	108.1.38.84	**6**	1

4.3.6 Access log Agent

On the contrary of other individual agents, access log agent does not indicate attack signatures or malicious behavior by external users, it indicates users who are trying to gather information and check the website contents or the operating environment of the network. Access log agent reads the contents of “access log” files about access messages associated with http and https services.

The purpose of the access log agent is to identify reconnaissance activities against the network which allows early detection of expected attacks. Figure 4.15 shows an example of access log http service log file; it contains historical access of the user through the system.

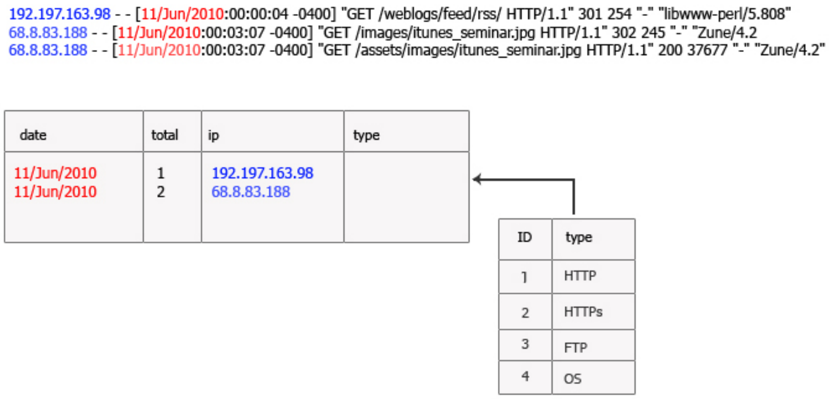
```
202.251.144.65 - - [11/Jun/2010:00:00:38 -0400] "GET /images/body_bg.png HTTP/1.1"
302 208 "http://www.cerias.purdue.edu/site/search/site? q=microsphere+"
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR
2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
OfficeLiveConnector.1.5; OfficeLivePatch.1.3)"
202.251.144.65 - - [11/Jun/2010:00:00:38 -0400] "GET /images/feed-icon16x16.png
HTTP/1.1" 302 214 "http://www.cerias.purdue.edu/site/search/site? q=microsphere+"
"Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR
2.0.50727; Media Center PC 5.0; .NET CLR 3.5.30729; .NET CLR 3.0.30729;
OfficeLiveConnector.1.5; OfficeLivePatch.1.3)"
```

Figure 4.15 http access log file

Algorithm 4.10 shows the access agent process to read the log contents and aggregates different user access within the website link. The input is the “access log” files which contain user’s access in http, https, OS, and FTP services, and the output is the access table.

The algorithm starts reading the file contents and checks each line which contains user session and its access messages which include date, IP, and the accessed resource. For each user IP, the algorithm checks if there is a record for this IP in the same day, if yes it increases the count of the access for the user IP by one, if no it creates a record in the access table per day.

Figure 4.16 shows an example of the access table records created from access log entries. Access table record include date, source IP, count of access per day and type of access.



```
192.197.163.98 - - [11/Jun/2010:00:00:04 -0400] "GET /weblogs/feed/rss/ HTTP/1.1" 301 254 "-" "libwww-perl/5.808"
68.8.83.188 - - [11/Jun/2010:00:03:07 -0400] "GET /images/itunes_seminar.jpg HTTP/1.1" 302 245 "-" "Zune/4.2
68.8.83.188 - - [11/Jun/2010:00:03:07 -0400] "GET /assets/images/itunes_seminar.jpg HTTP/1.1" 200 37677 "-" "Zune/4.2"
```

date	total	ip	type
11/Jun/2010 11/Jun/2010	1 2	192.197.163.98 68.8.83.188	

ID	type
1	HTTP
2	HTTPs
3	FTP
4	OS

Figure 4.16 Access Log Table

While access log agent indicates the external user access to the contents of the web site, Missing log agent indicates the external user scan of the network files which represent their trial to identify the server Operating system or looking for the availability of specific services. Missing log agent reads the contents of the OS error log and FTP error log files. The purpose of missing log agent is to identify scan activities against the network which allows early detection of expected attacks. Figure 4.17 shows an example of an OS http error log http service log file; it contains user trials to look for specific files within the system

```
[Fri Jun 11 00:01:16 2010] [error] [client 66.249.71.230] File does not exist: /var/ftp/osmirrors/pub/FreeBSD/ports/sun4v/packages-stable/python/pyne-1.1.0_6.tbz
[Fri Jun 11 00:01:31 2010] [error] [client 66.249.71.230] File does not exist: /var/ftp/osmirrors/pub/FreeBSD/ports/sun4v/packages/python/trac-TracGantt-0.3.2a_4.tbz
[Fri Jun 11 00:01:34 2010] [error] [client 67.218.116.168] File does not exist: /var/ftp/osmirrors/pub/debian/pool
```

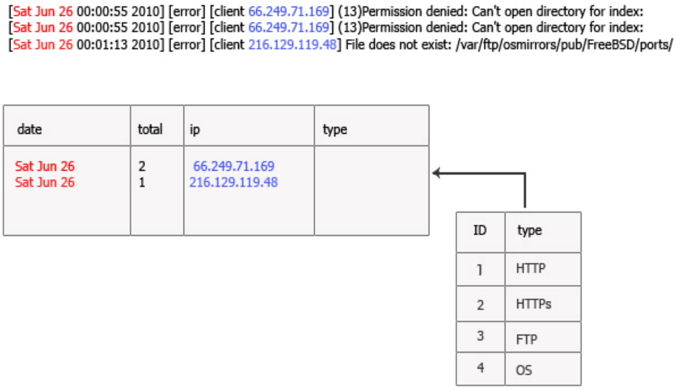
Figure 4.17 OSHTTP error log file

Algorithm 4-10 Access Log Agent

```
Algorithm 4.10 Access  agent
Input: Access log files .
Output: fill data to database table 'Access’
Initialization: User select file type
        1 – http , 2 – https , 3 – os mirror , 4 – ftp
While Not EOF
        For each line
          Read line contents;
          Read date;
          Read source IP;
                check ` access ` table  if date and IP is inserted
              if true then
                        update total = total + 1;
              else
                        Store to database (date , Ip , 1,type  ) .
                   End if;
        End for
End while
Return ' Access'  table .
```

Algorithm 4.11 shows the missing agent process to read the log contents and aggregates different user scans within the website files. The input is the “error log” files which contain user’s access in http, https, OS, and FTP services, and the output is the missing table. Initial scan messages list include the messages indicating that the users are trying to access unauthorized files or looking to specific files names which show the used operating system

The algorithm starts reading the file contents and checks each line which contains user session and its error messages which include the date, IP, and error. For each user IP, the algorithm check if this message is in scan messages list, if yes it will check if there is a record for this IP in the same day. In case if the user has previous record with the same day, the algorithm increases the count of the scan for the user IP by one, if no it creates a record in the missing table per day. Figure 4.18 shows an example of the missing table records created from missing log entries. Missing table record include the date, source IP, count of scan per day and type of scanned services.



```
[Sat Jun 26 00:00:55 2010] [error] [client 66.249.71.169] (13)Permission denied: Can't open directory for index:
[Sat Jun 26 00:00:55 2010] [error] [client 66.249.71.169] (13)Permission denied: Can't open directory for index:
[Sat Jun 26 00:01:13 2010] [error] [client 216.129.119.48] File does not exist: /var/ftp/osmirrors/pub/FreeBSD/ports/
```

date	total	ip	type
Sat Jun 26	2	66.249.71.169	
Sat Jun 26	1	216.129.119.48	

ID	type
1	HTTP
2	HTTPs
3	FTP
4	OS

Figure 4.18 Missing example

Threshold values have been set for the normal count of access links within the web site and the count of missing files in knowledge base tables. We can use these threshold values to distinguish between the normal accesses or scan files and the reconnaissance activity for the attack purpose in both access missing tables.

Algorithm 4-11 Missing Log Agent

```
Algorithm 4.11 Missing Log Agent
Input: FTP and OS mirror Error log files.
Output: fill data to database table ' missing
Initialization: Set scan message list = “permission denied, file does not exist”
While not EOF
        For each line
          Read line contents;
          Read date;
                Read source IP;
                Read message;
          If message in scan message list
                check ` access ` table  if date and IP is inserted
               if true then
                        update total = total + 1;
               else
                        Store to database (date , Ip , 1,type  ) .
                    End if;
                Else
                End if
        End for
End while
Return ‘missing’ table.
```

4.4 DACM Central Agent

DACM central agent has access to the result tables of different individual agents; it aggregates these results together into unified table which includes those results together in relation with the attacker IP. Figure 4.19 show that different individual agents stored their results in central database tables. Central agent gets that database to produce a set of useful reports which summarize different attacks against the network together.

Those results shown in the figure includes daily report, IP report, severity alerts, single alerts, false negative alerts, reconnaissance alerts, summary date report, and sever IPs report.

By the end of the individual agent results, table IPs contains all different IPs which has been stored in attack tables or access or missing tables with a unique ID representing that IP address.

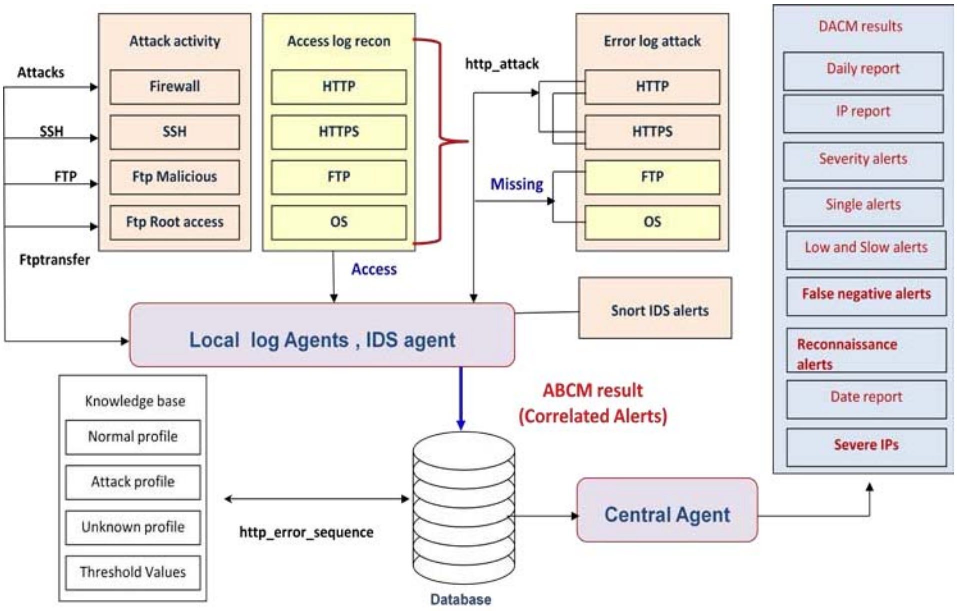


Figure 4.19 DACM Central Agent results

DACM central agent performs its function through two steps, the first step is to aggregate all different results from individual agent results into daily table which includes different attacks and activities for different IPs. The second step is the analysis of these attacks together to represent the whole picture of the situation in the network and improve the detection rate and to produce the correlation results of such different agents together.

Algorithm 4.12 show the first step process, the agent loops through IPS table, for each record, it selects the ID for the IP address and select related attacks and activity for that IP from different result tables and insert that record in new table called daily.

Algorithm 4-12 DACM Central Agent

```
Algorithm 4.12 Daily agent
Input: All other agents' results tables.
Output: fill data to database table ' daily' and ' daily_res'
Step 1
Loop through IPS table
                Insert new record in  ' daily' table with
                Select    IP from IPS, // where IP = IP in related tables
                          Date as current system data, // where date = date in related
                                                       tables
                   Count `abcm_res` , // IDS correlated alerts
                   Count  `attacks_res`, // firewall contents for IP
                   Count  `ftp`, // FTP attack table
                   count `ftptransfer`, // FTP transfer attack table
                   count `http_attack`,// http attack table
                          Count  `ssh`, // ssh attack table
                   Sum `access`, // acces count for that IP
                   Where count > access threshold value
                          sum `missing`, // scan count for that IP
                          Where count > scan threshold value
End loop
Step 2 : For IP in daily table select IP,
                   Date, Count `abcm_res` ,   Sum `access`,
                   Count  `attacks_res`, Count `ftp`,
                   Count `ftptransfer`, Count `http_attack`,
                   Sum `missing`, Count `ssh`, Alert type
```

Table 4.6 shows the daily table fields. These fields are Date, IP, count of alerts for this IP from IDS correlated alerts in that day, count of appeals of this IP in firewall blocked IPs log, related FTP and FTP transfer attacks, related http attacks, related SSH attacks, and related access or scan activities count for that IP which exceed the allowed threshold values for normal access or scan activities. In addition an analysis result field called type to show the IP behavior and conclusion in the system.

Table 4.6 Daily Report Table Attributes

IP	Date	Type	ABCM	Firewall	FTP	FTP root	Http attack	SSH	Access	Scan

Daily report algorithm aggregates that related attacks together for better understanding of current situation for different attacks, and summarizes wide range of activity for each attacker IP.

Algorithm 4.12 performs the second step to show the conclusion and better analysis of that aggregated alerts in step1. Step 2 determines the behavior type for each IP. Step 2 loops through daily table to determine the alert type by comparing the count of alerts from different sources to indicate the alert type according to a set of rules shown in the following pseudo code for each set of types.

The alert type for each IP will be determined according to different alert and attacks from that IP.

False negative alerts with respect to IDS are real alerts while it was not detected by these IDS. The alert type could be false negative in case that the IP attacks are not detected in the IDS alerts while they were detected from other sources of attacks such as FTP, SSH, and HTTP attacks. It could be considered false negative alerts where the IP attack is detected in IDS alert and detected in other sources but in different times.

False negative and verified alerts pseudo code

```
        For each IP
         If Count `abcm_res`  =  0
              and  {       OR   Count  `attacks_res`   >= 1
              OR   Count  `ftp`              >= 1
              OR   count `ftptransfer`      >= 1
              OR   count `http_attack`    >= 1
                    OR      Count  `ssh`             >= 1 }
    Then
        Alert type = False Negative
        Else
        For each alerts
        If time (alert) <> time (alerts from other attacks)
          Then
              Alert type = False Negative
          Else
          Alert type= Verified alerts
              End if
        End if;
```

False negative and verified alerts conditions are shown in false negative and verified alerts pseudo code; if the alerts were detected in IDS correlated alerts and detected in the same window time from other sources of attacks such as FTP, SSH, and HTTP attacks, then the alerts are considered verified alerts or severe alerts. Verified alerts assure that the alerts were detected using different sources of information which reduce false positive alerts.

Alert type is a single IDS alert if only one alert was detected from IDS and have not been detected by other sources of information in that window time. Such kind of an alert could be stored to indicate the probability of low and slow attack.

```
Case : single alert // single alerts for Low and Slow attacks
              If Count `abcm_res` = 1   and
                       Count  `attacks_res`  = 0  and
                              Count  `ftp`         = 0 and
                       count `ftptransfer`     = 0   and
                       count `http_attack`     = 0   and
                              Count  `ssh`                = 0  ;
              Then
                        Alert type = Single alert;
                       End if;
```

Alert type is Reconnaissance alert if there is no alerts detected either from IDS or from other sources of information in that time, but there is system access and/or scan which exceed the threshold allowed values. Such kinds of reconnaissance indicate it is gathering data about the system before being attacked. Such kinds of alert type enable the early detection of those trials of attacks.

```
Case : Reconnaissance // no attacks, gathering data or systemscan
                  If Count `abcm_res` = 0   and  {
                           Count  `attacks_res`   = 0
                       and   Count  `ftp`              = 0
                       and   count `ftptransfer`      = 0
                       and   count `http_attack`      = 0
                              and     Count  `ssh`              = 0   }
              and {       sum `missing`    >= minimum value
                       OR Sum `access`>= minimum value }
                       then
                  Alert type = Reconnaissance alert.
              Else
                       End if
```

Alert type is IDS only when many alerts were only detected by IDS and have not been detected by other sources of information in that time. Such kinds of alerts indicate that no other sources were able to detect that attack which may help in improving the logging capability.

```
Case: IDS alerts only
                If Count of `abcm_res` > 1   and
                        {     Count  `attacks_res`  = 0   and
                         Count  `ftp`            =      and
                        count `ftptransfer`     = 0   and
                        count `http_attack`     = 0   and
                                Count  `ssh`              = 0
              } and
                        {
                        OR   sum `missing`    <= minimum value
                        OR   Sum `access`     <= minimum value
                        }
                        Then
                                Alert type = IDS only alert;
                        Else
                        End if ;
        End loop
Return ' daily' and ' daily_res' tables
```

4.5 Implementation Environment

DACM have been implemented using Mysql database for storing alerts tables and correlated alerts results as well as result tables of different agents, it also used to store knowledge base and learning criteria. Borland6 C++ programming language have been used to implement IDS alert correlation agent, while Microsoft Visual Basic 6 used to implement central agent and other correlation agents for security tools and log files. Finally, we used Windows7 Ultimate 64-bit operating system over Dell studio laptop with Intel Core2Duo CPU-9300-2.5GHz – 6MB cache processor, and 2GB RAM for testing the implemented model.

CHAPTER 5
DACM Results and Analysis

Chapter Five: DACM Results and Analysis

This chapter presents detailed DACM results for individual agents and the central agent. The chapter is organized into six sections. Section one provides a description of the CERIAS dataset that was gathered and tested to implement DACM. Section 2 provides detailed results for the CAM, ABCM, and DPCM IDS alert correlation techniques and their performance/time metrics. Section 3 provides results for the individual agents from other sources of information. Section 4 summarizes DACM central agent results. Section five evaluates DACM performance measure and assessment. Finally, section six addresses some DACM implementation consideration.

5.1 CRIAS Data Set

The dataset used to implement DACM was collected by the author during a visiting scholar trip to the Center for Education and Research in Information Assurance and Security (CERIAS) [64] from April - July 2010. CERIAS, part of Purdue University, is considered to be one of the leading information security research centers in the world.

This section describes the data collection environment, the CERIAS network from border router to the web server, and the contents of the collected datasets; it also includes a description of simulated attacks and attack scenarios.

5.1.1 CERIAS Network Description

The CERIAS network shown in Figure 5.1 is described as follow:

1. CERIAS is connected to the Purdue Data Network and the rest of the world through an ITaP router.

2. The ITaP router connects to the CERIAS network bridging firewall. This firewall blocks IP addresses of known attack sites, compromised machines, and sources of disruption. The firewall also blocks non-routable IP addresses as defined in RFC 5735 [65] (Special-Use IPv4 Addresses).

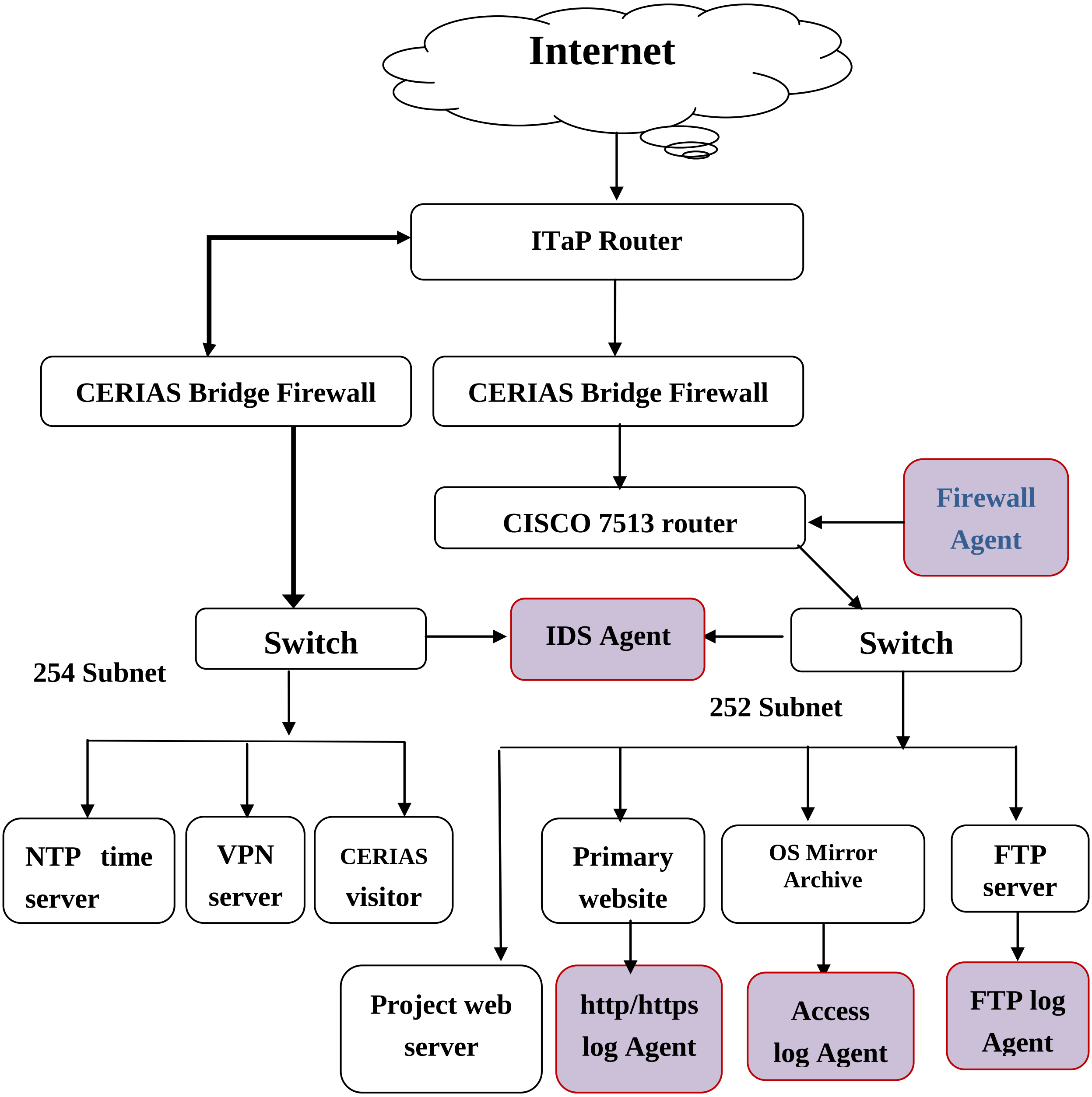


Figure 5.1 CERIAS Network Block Diagram

3. The bridging firewall is connected to a Cisco 7513 router. This router uses a variety of ACLs to protect the CERIAS subnets. The web server lives on the 129.10.252.0/24 subnet which is used as an internet DMZ network. Basically, the DMZ has limited network access both internally and externally. From outside the CERIAS network, only network connections for NTP, HTTP, and HTTPS are allowed. From inside the DMZ, the web server is not allowed access to the internal subnets.

4. The web server itself has been configured to resist attacks. It uses a recent supported version of Apache 2, uses mod_security2 for application-level firewall protection, and each component has been configured to security best practices for protection from attacks.

5.1.2 Data Description

Data was collected 11-28 June 2010 and consisted of three main sources: Snort [49] alerts, network packet data, and application and system log messages.

The snort sensors monitored network traffic on the 128.10.254.0/24 and 128.10.252.0/24 subnets. The 254 subnet is a network for CERIAS visitors, the CERIAS VPN server, and two public NTP timeservers; there is limited firewall protection for this subnet. The 252 subnet has the CERIAS primary web site (kargad), as well as the project web server (blackmesa), and the CERIAS Security Tool Archive and the OS Mirror Web and FTP site (omelas). Snort captured almost 800,000 alerts in this period. The alerts were stored in database tables within a MySQL database and are accessed through the ACIDBASE [55] web interface. The environment was Ubuntu 10.04 Linux-based OS.

We used Wireshark [59] to capture network packets traffic for CERIAS website (kargad) which contains detailed packet information. We also retrieved the output of Nessus [23], a network vulnerability scanner, and Nmap [36], a network mapping utility, to check for known network vulnerabilities and network port status.

Finally, we collected log files for CERIAS services as follows:

Archive FTP: FTP log messages from the ftp.cerias.purdue.edu and osmirror.cerias.purdue.edu sites.

Proftpd: error messages and requests associated with FTP.

Xferlog: a listing of files transferred over FTP.

Archive HTTP: HTTP log messages from the ftp.cerias.purdue.edu and osmirror.cerias.purdue.edu sites.

FTP HTTP: http requests and error messages for ftp.cerias.purdue.edu.

OSMIRROR HTTP: http requests and error messages for osmirror.cerias.purdue.edu.

BASM SSH
Inetdlog: Service and error messages from the inetd and child processes; this log includes error messages associated with SSH. IPs and hostnames of people asking to guess users passwords.

CERIAS HTTP:
access_log: HTTP requests.
error_log: Errors associated with HTTP requests.
https_access_log: HTTPS (SSL) requests.
https_error_log: HTTPS (SSL) errors associated with HTTPS.
Firewall Router: Log messages containing a list of blocked network packets from the outside world. We can identify the blocked IPs and compare it with the IPs listed in other resources during the same time period to assure their behavior.
While our model includes performance data, they were not collected during the data collection process (experiment) as necessary performance monitoring tools were not available.

5.1.3 Attacks

Some attacks were conducted during the capture data period to add to the normal attack behavior. The simulated attacks were conducted to assure that the captured data contained LOW and Slow attacks. The simulated attack data is as follows:

Nmap: port scan of web server; we used the version check option to determine the name and version of the service “nmap –sV”; we looked to port 80 and 443 (http and https) service.

Nikto: Configuration scan of the web server, we attempted to evade IDS detection by slowing the scan speed down, “nikto.pl -Tuning 3b -Pause 5 –evasion”.

Using a Firefox plug-in called tamper data, we attempted to send bad data to a form in the CERIAS web site in order to exploit vulnerabilities in the form processing script.

5.1.4 Attack scenarios

In collecting the above data, we attempted several attack scenarios. While we collected some signatures from these trials, the scenarios were not completed successfully because of CERIAS security.

Scenario 1

- An attacker uses a regular web browser to browse a web site for forms which he/she can use to attack the system. (One of these forms may have vulnerabilities that could be exploited by the attacker).
- The attacker uses a variety of techniques to determine the vulnerability of the forms:
  - Putting too much data in the form to check how the script responds
  - Altering the URL components to see how the script responds.
- If the script has a vulnerability, the attacker will attempt to:

    - Corrupt the database and disrupt the website;
    - Extract information from the database;
    - Alter information in the database;
    - Run a command in the system to gain illegal access or disrupt the system behavior.
- The attacker tries to determine what other systems are accessible and attempts to do reconnaissance to determine which of these systems could be compromised.
- The attacker will bring over utilities (attack tools to current compromised machine) to compromise other accessible machines.
- Attack accessible machines and compromise them.

Scenario 2

- The attacker probes the CERIAS FTP server looking for vulnerabilities and configuration errors.
    - Assumption: FTP server has a buffer overflow problem.
- The attacker uses buffer overflow to gain access to the FTP server.
- The attacker uses that access to bring over attack tools.
- Those tools are then used to gain higher level access to the FTP server operating system:
    - The attacker could delete the FTP archive; corrupt/modify the contents; or use the ftp as a distribution point for illegal software.
    - Determine other accessible systems to attack them.

Scenario 3

- The attacker identifies an available CERIAS SSH servers.
- He/she discovers the user id for a CERIAS employee.
- He/she uses a SSH brute force tool to guess the password for the user id identified in step2.
    - Assumption: the attacker gets a correct password.
- The attacker uses the account and password to get illegal access to other CERIAS systems.

- The attacker brings over attack tools to attack other systems.

Scenario 4

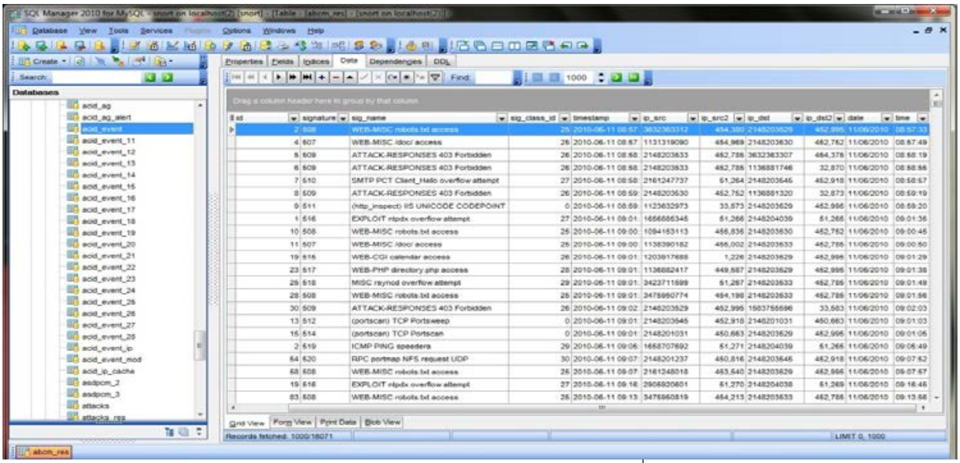
- Assumption. One of the projects runs student code that has a vulnerability.
- An attacker discovers the vulnerability and exploits the code in the project server and uses it to gain access on the system.

Scenario 5

- Attacker sends Phishing Email to the user.
- The user accesses the phishing site and enters his/her identity.
- The attacker uses the account and password to get illegal access to other CERIAS systems.
- The attacker brings over attack tools to attack other systems.

5.2 IDS Alerts Correlation Results

Over a period of 18 days, Snort collected 858,000 alerts in the CERIAS dataset; alerts were divided to be correlated through those days. The alerts were correlated using CAM [56–58], ABCM, and DPCM. Figure 5.2 shows the total number of alerts and the 18 tables which represent daily alerts.



id	signature	sig_name	sig_class_id	timestamp	ip_src	ip_src2	ip_dst	ip_dst2	date	time
2	508	WEB-MISC robots.txt access	25	2010-06-11 08:57	3632363312	454,380	2148203629	452,995	11/06/2010	08:57:33
4	507	WEB-MISC /doc/ access	26	2010-06-11 08:57	1131319090	454,969	2148203630	452,762	11/06/2010	08:57:49
5	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:58	2148203633	452,786	3632363307	454,376	11/06/2010	08:58:19
6	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:58	2148203633	452,786	1136881746	32,870	11/06/2010	08:58:56
7	510	SMTP PCT Client_Hello overflow attempt	27	2010-06-11 08:58	2161247737	51,264	2148203646	452,918	11/06/2010	08:58:57
8	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:59	2148203630	452,762	1136881320	32,873	11/06/2010	08:59:19
9	511	(http_inspect) IIS UNICODE CODEPOINT	0	2010-06-11 08:59	1123632973	33,673	2148203629	452,995	11/06/2010	08:59:20
1	516	EXPLOIT ntpdx overflow attempt	27	2010-06-11 09:01	1666686346	51,266	2148204039	51,266	11/06/2010	09:01:36
10	508	WEB-MISC robots.txt access	25	2010-06-11 09:00	1094153113	456,836	2148203630	452,762	11/06/2010	09:00:46
11	507	WEB-MISC /doc/ access	25	2010-06-11 09:00	1138390182	456,002	2148203633	452,786	11/06/2010	09:00:50
19	515	WEB-CGI calendar access	26	2010-06-11 09:01	1203917688	1,226	2148203629	452,995	11/06/2010	09:01:29
23	517	WEB-PHP directory.php access	28	2010-06-11 09:01	1136882417	449,587	2148203629	452,995	11/06/2010	09:01:38
25	518	MISC rsyncd overflow attempt	29	2010-06-11 09:01	3423711699	51,267	2148203633	452,786	11/06/2010	09:01:49
28	508	WEB-MISC robots.txt access	25	2010-06-11 09:01	3475950774	454,198	2148203633	452,786	11/06/2010	09:01:56
30	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 09:02	2148203629	452,995	1583755696	33,683	11/06/2010	09:02:03
13	512	(portscan) TCP Portsweep	0	2010-06-11 09:01	2148203645	452,918	2148201031	450,683	11/06/2010	09:01:03
15	514	(portscan) TCP Portscan	0	2010-06-11 09:01	2148201031	450,683	2148203629	452,995	11/06/2010	09:01:06
2	519	ICMP PING speedera	29	2010-06-11 09:06	1668707692	51,271	2148204039	51,266	11/06/2010	09:06:49
54	520	RPC portmap NFS request UDP	30	2010-06-11 09:07	2148201237	450,816	2148203645	452,918	11/06/2010	09:07:52
58	508	WEB-MISC robots.txt access	25	2010-06-11 09:07	2161248018	453,540	2148203629	452,995	11/06/2010	09:07:57
19	516	EXPLOIT ntpdx overflow attempt	27	2010-06-11 09:16	2905920601	51,270	2148204038	51,269	11/06/2010	09:16:46
83	508	WEB-MISC robots.txt access	25	2010-06-11 09:13	3475960819	454,213	2148203633	452,786	11/06/2010	09:13:56

Records fetched: 1000/18071

LIMIT 0, 1000

Figure 5.2 Snort IDS alerts[5]

[5] larger image of Figure 5.2 is included in Appendix A

Table 5.1 presents sample alerts. The collected alert data included sensor id, alert id, signature, timestamp, source IP, destination IP, protocol, source port, and destination port.

Table 5.1 SNORT IDS alert Attributes

Sid	Cid	Sig_Name	Timestamp	IP_src	IP_dst	Proto	Sport	Dport
6	16	ICMP PING speedera	6/11/2010 9:14	3460811837	2148204039	1		
6	20	EXPLOIT ntpdx overflow attempt	6/11/2010 9:21	1656885345	2148204039	17	123	123
7	28	WEB-MISC /doc/ access	6/11/2010 21:42	1131319090	2148203530	6	59285	80
7	30	WEB-MISC robots.txt access	6/11/2010 21:47	3475949512	2148203529	6	19427	80

5.2.1 IDS correlation Model

We implemented an integrated interface to correlate the alerts using the three different techniques and compare their reduction rate and correlation time. Figure 5.3 shows the IDS correlation models' interface. We ran the three models against the alerts from the 18 days and got the reduction rate and correlation time for each model.

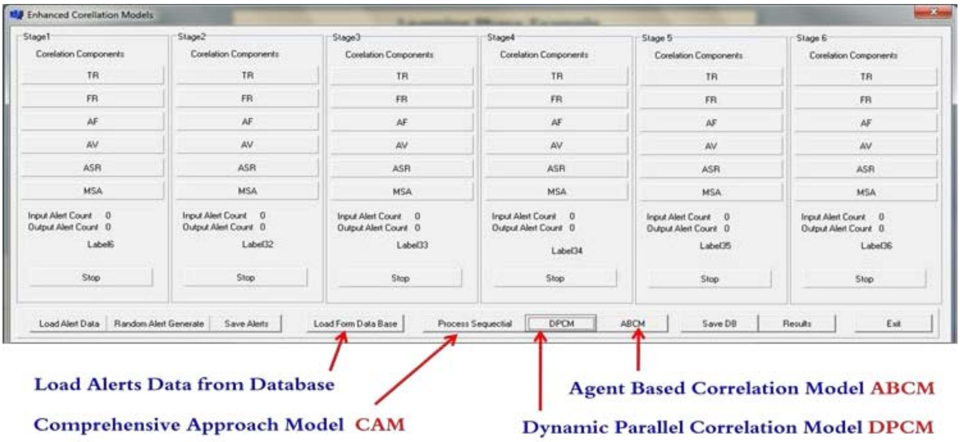
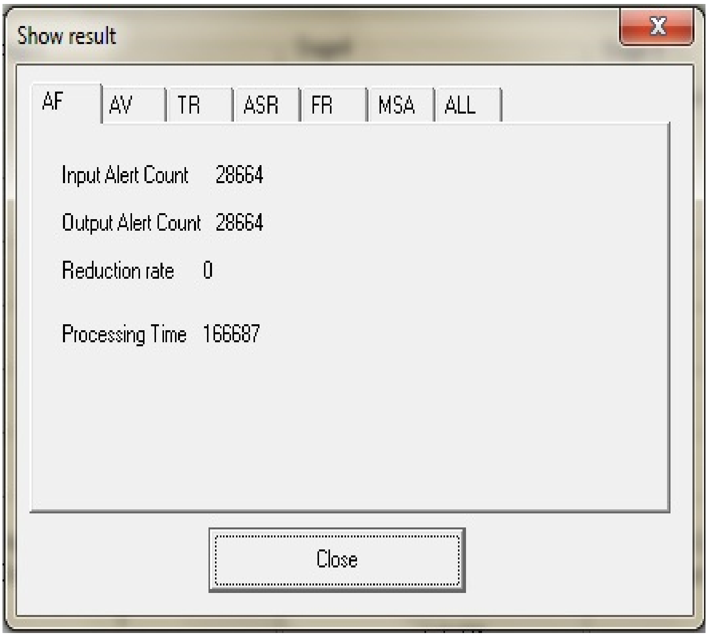


Figure 5.3 IDS alert correlation Interface[6]

[6] Larger image of Figure 5.3 is included in Appendix A

5.2.2 CAM Results

As an example, the 28,664 alerts from day 11 were tested, and we ran CAM to correlate those alerts. Figure 5.4 shows the results of correlation with component alert fusion (AF), producing a 0 % reduction rate in 166 seconds of correlation time. The AF component has no affect in reducing the number of alerts while still consuming high processing time.



Input Alert Count 28664

Output Alert Count 28664

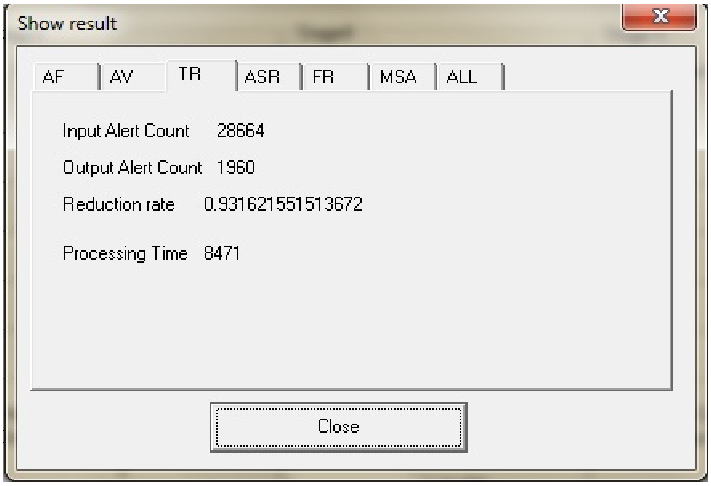
Reduction rate 0

Processing Time 166687

Figure 5.4 AF correlation result[7]

Figure 5.5 shows that the Threat Reconstruction correlation component produced 1,960 alerts compared with 28,664 input alerts. This is a 93.16% reduction in the output alert rate with 8.5 seconds of processing time.

[7] Larger image of Figure 5.4 is included in Appendix A



Input Alert Count 28664

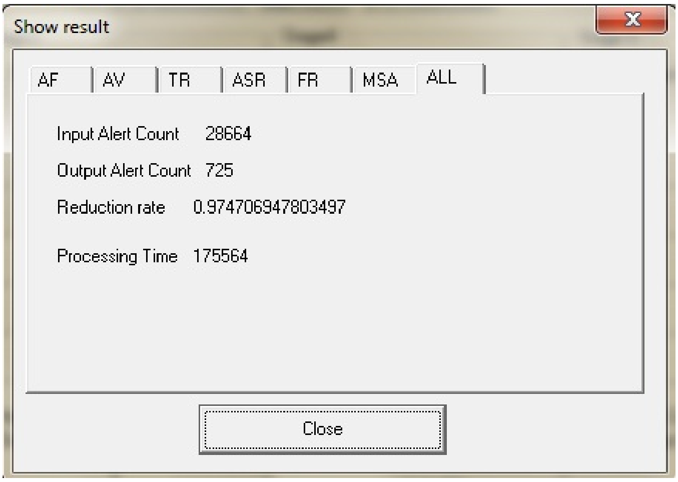
Output Alert Count 1960

Reduction rate 0.931621551513672

Processing Time 8471

Figure 5.5 TR Correlation Result

The final correlation result for the 28,644 alerts in our sample test was obtained using the FR and MSA correlation components. As show in Figure 5.6, the final reduction rate using CAM is 97.4 % with a total processing time of 175 seconds. The sequence components results indicate that only the TR, FR, and MSA components are effective. The effective correlation components have a total correlation time of approximately 11 seconds, while AF has a correlation time 166 seconds without any alert reduction. CAM thus has 166 seconds of wasted time.



Input Alert Count 28664

Output Alert Count 725

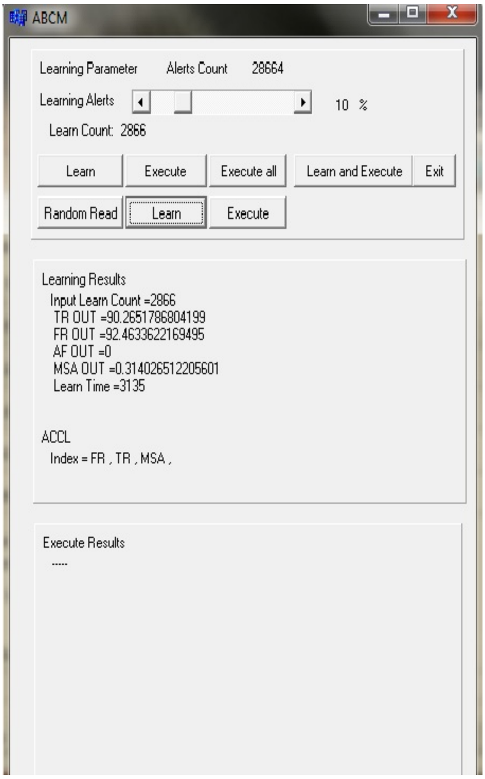
Reduction rate 0.974706947803497

Processing Time 175564

Figure 5.6 Final CAM Correlation Result

5.2.3 ABCM results

Agent based correlation Model runs through Learning and Correlation Phases. The Learning Phase creates ACCL to determine which component can be used and in which order. The Learning Phase results presented in Figure 5.7 show that ACCL will be composed of FR, TR, and MSA components. The learning time was 3 seconds for 2866 alerts (10 % of the total number). The order of ACCL depends on the reduction rate of each component in ACCL. FR has the highest reduction rate followed by TR; MSA was the lowest reduction rate.



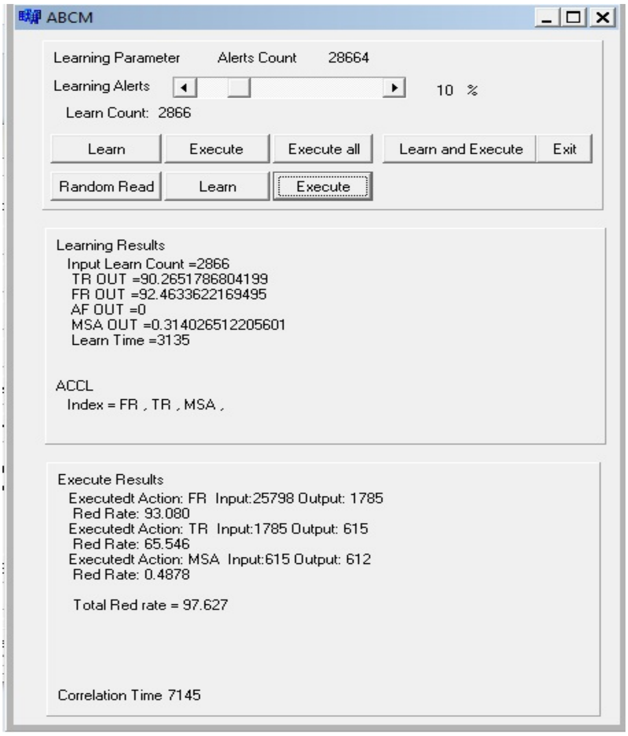
```
Learning Parameter      Alerts Count     28664
Learning Alerts                          10  %
  Learn Count:  2866

Learning Results
  Input Learn Count =2866
   TR OUT =90.2651786804199
   FR OUT =92.4633622169495
   AF OUT =0
   MSA OUT =0.314026512205601
   Learn Time =3135

ACCL
  Index = FR , TR , MSA ,

Execute Results
  -----
```

Figure 5.7 ABCM Learning Phase[8]

In the Correlation Phase, only the effective correlation components in ACCL will be used to correlate the alerts. The input to the first correlation component in ACCL will be the rest of the alerts after removal of the learned alerts. As shown in Figure 5.8, the FR component has a reduction rate of 93% for the 25,798 input alerts and produces 1,785 correlated alerts for the second component in ACCL. The TR component has a reduction rate of 65% for the 1,785 input alerts and produces 615 correlated alerts for the third component in ACCL. The MSA component has a reduction rate of 0.4% for those 615 input alerts and produces 612 correlated alerts as the final correlated alerts. The 612 final correlated alerts represent a reduction rate of 97% for the 25,798 input alerts. The total correlation time for the sequence of correlation using ACCL

[8] Larger image of Figure 5.7 is included in Appendix A

components is 7 seconds, which produces total learning and correlation time of 10 seconds. Thus the Learning Phase enhanced the correlation process compared with CAM by eliminating the time consumed for the AF component.



```
Learning Parameter      Alerts Count     28664
Learning Alerts                                     10  %
  Learn Count:  2866
```

```
Learning Results
  Input Learn Count =2866
   TR OUT =90.2651786804199
   FR OUT =92.4633622169495
   AF OUT =0
   MSA OUT =0.314026512205601
   Learn Time =3135


ACCL
  Index = FR , TR , MSA ,
```

```
Execute Results
  Executedt Action: FR  Input:25798 Output: 1785
   Red Rate: 93.080
  Executedt Action: TR  Input:1785 Output: 615
   Red Rate: 65.546
  Executedt Action: MSA  Input:615 Output: 612
   Red Rate: 0.4878

   Total Red rate = 97.627



Correlation Time 7145
```

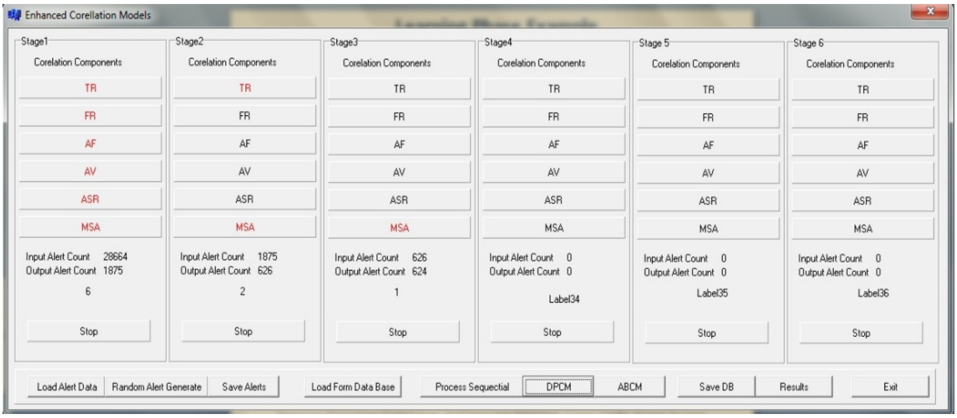
Figure 5.8 ABCM’s Correlation Phase Results[9]

5.2.4 DPCM Results

In DPCM, correlation is done in correlation stages, with each stage including a set of correlation components instead of an individual component. Figure 5.9 shows the DPCM correlation for 11 June of 28,664 alerts as same example correlated by CAM and ABCM. All correlation components in the first stage have been used and produced different reduction rates for each of them. Components AV, ASR, and AF have 0% reduction rates and will be disabled in the next stage. Since the

[9] Larger image of Figure 5.8 is included in Appendix A

FR Component has the highest reduction rate in the first correlation stage, its output will be the input to the next stage and it will be disabled in the next stage. In the second correlation stage, TR and MSA correlate the output of the FR component from the first stage. Since TR has a higher reduction rate than MSA, the correlated alerts output of the TR component will be the input to the third stage. In the third stage the TR component will be disabled, and only the MSA component is active.



Stage1
Input Alert Count 28664
Output Alert Count 1875
6

Stage2
Input Alert Count 1875
Output Alert Count 626
2

Stage3
Input Alert Count 626
Output Alert Count 624
1

Stage4
Input Alert Count 0
Output Alert Count 0
Label34

Stage 5
Input Alert Count 0
Output Alert Count 0
Label35

Stage 6
Input Alert Count 0
Output Alert Count 0
Label36

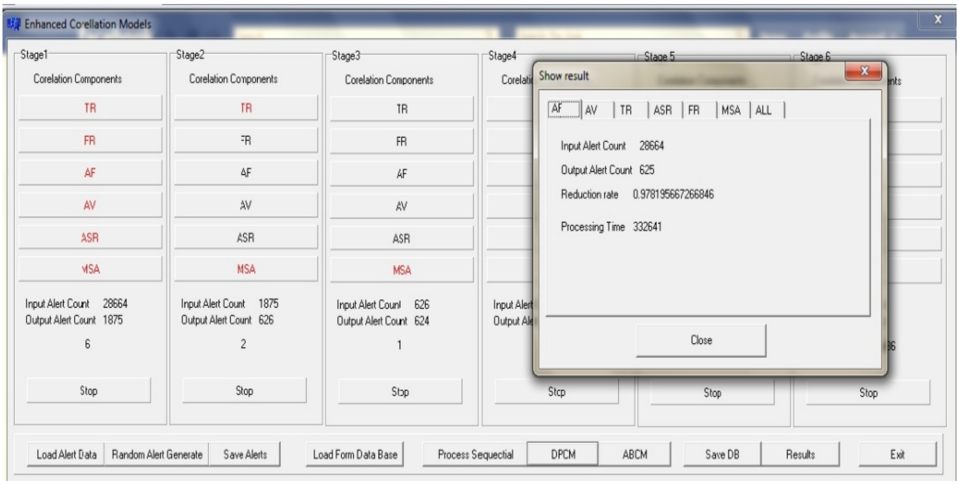
Figure 5.9 DPCM Correlation Stages Result[10]

The output of the third stage, represented by the MSA output, is the output correlated alerts done by DPCM. This is 625 alerts out of the total input of 28,664 alerts. Figure 5.9 shows the total result of the DPCM correlation process. The DPCM reduction rate is 97.8% while consuming 336 seconds of processing time for the total correlation time by the different correlation stages.

The correlation time for each stage depends on the longest correlation component in that stage. The DPCM reduction rate is more accurate than ABCM with no need for a learning process, while having a longer correlation time in comparison with CAM and ABCM.

[10] Larger image of Figure 5.9 is included in Appendix A

DPCM is also expected to have a lower time than CAM exactly as ABCM if a fully parallel architecture is used to implement it. The time produced here is because a single processor with multi-threading is used to implement DPCM.



Stage1: Input Alert Count 28664
Output Alert Count 1875
6

Stage2: Input Alert Count 1875
Output Alert Count 626
2

Stage3: Input Alert Count 626
Output Alert Count 624
1

Show result:
Input Alert Count 28664
Output Alert Count 625
Reduction rate 0.978195667266846
Processing Time 332641

Figure 5.10 DPCM Final Correlation Result[11]

5.2.5 IDS Alert Correlation Techniques Performance

This section compares the performance of three different IDS correlation techniques against the 18 days of alerts gathered in the CERIAS dataset. Table 5.2 summarizes the reduction rates for CAM, DPCM, and ABCM. There were 838,348 total alerts collected over 18 days. There was an average of 46,574 daily alerts, with a minimum of 11,788 alerts (20 June), and maximum of 152,240 alerts collected (18 June).

The total output of correlated alerts by CAM was 17,741 alerts out of 838,348 input alerts, representing a 97.88% total reduction rate. The total output of correlated alerts by ABCM was 16,218 alerts out of 754,505 alerts after excluding learned alerts from correlation, representing a total reduction rate of 97.88%. The total output correlated alerts by DPCM was 17,004 of 838,348 and a total reduction rate of 97.97%.

[11] Larger image of Figure 5.10 is included in Appendix A

Table 5.2 Alert Correlation Reduction Rates Comparison

Day	I/P alerts	CAM		DPCM		ABCM		
		O/P	RR	O/P	RR	alert learned	O/P	RR
11	28664	725	97.47	625	97.82	2866	619	97.60
12	46703	1076	97.70	979	97.90	4670	961	97.71
13	54759	1060	98.06	935	98.29	5475	909	98.16
14	34303	1184	96.55	1178	96.57	3430	1050	96.60
15	51823	944	98.18	870	98.32	5182	832	98.22
16	49609	1095	97.79	1010	97.96	4960	997	97.77
17	34879	982	97.18	905	97.41	3487	881	97.19
18	152240	1083	99.29	1077	99.29	15224	1044	99.24
19	15175	531	96.50	513	96.62	1517	497	96.36
20	11788	354	97.00	346	97.06	1178	312	97.06
21	49336	1164	97.64	1143	97.68	4933	1118	97.48
22	39786	1146	97.12	1139	97.14	3978	1035	97.11
23	27753	1214	95.63	1171	95.78	2775	1067	95.73
24	70686	1192	98.31	1174	98.34	7086	1140	98.21
25	36072	1117	96.90	1106	96.93	3607	1003	96.91
26	57598	1055	98.17	1036	98.20	5759	1018	98.04
27	52035	1083	97.92	1075	97.93	5203	1039	97.78
28	25139	736	97.07	722	97.13	2513	696	96.92

Figure 5.11 shows a graph of the daily reduction rate for CAM, DPCM, and ABCM. The x-axis represents the daily alert count, and the Y-axis represents the reduction rate percentage of each model for those daily alerts. The results showed that the reduction rates of the three models are very close to each other with minor differences. While they are almost equal, DPCM has the highest reduction rate, followed by CAM, and then ABCM.

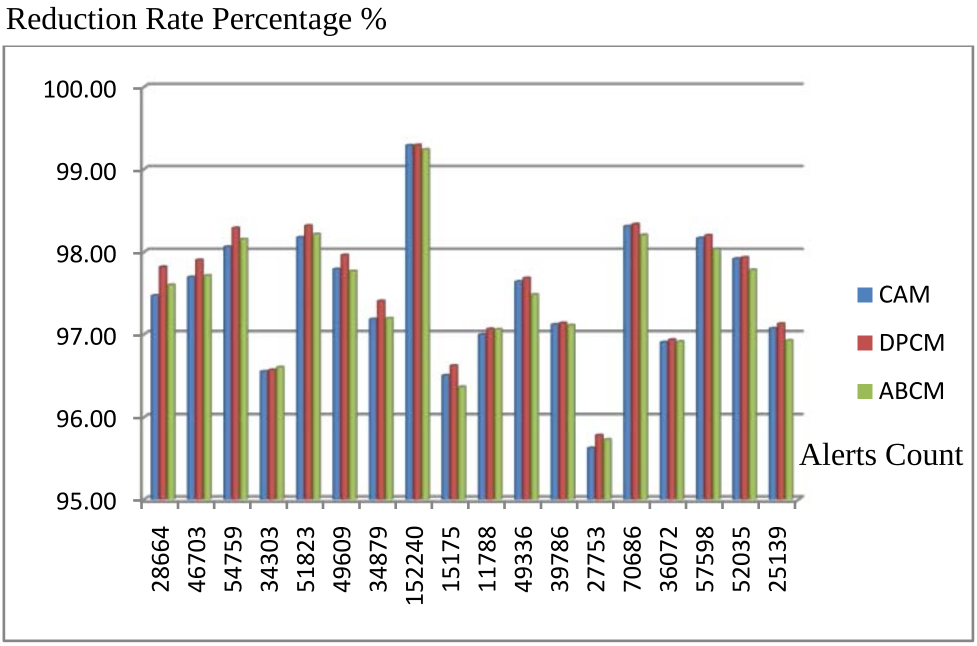


Figure 5.11 Reduction Rates Comparison of IDS Correlation Techniques

Table 5.3 summarizes the correlation times for CAM, DPCM, and ABCM. It shows the daily alerts ordered by the date of the alerts, while Table 5.4 shows the same results ordered by the alert count and correlation time for each technique.

Table 5.3 Correlation Time Comparison for IDS Alert Correlation Models ordered by date of alerts

Day	Alerts Count	CAM Time	ABCM Time	DPCM Time
11	28664	195	10	367
12	46703	436	28	842
13	54759	620	35	1159
14	34303	233	23	434
15	51823	447	35	982
16	49609	504	39	1130
17	34879	325	24	526
18	152240	6048	103	10082
19	15175	52	9	102
20	11788	27	3	50
21	49336	457	32	889
22	39786	375	34	738
23	27753	171	18	281
24	70686	940	44	1863
25	36072	278	20	543
26	57598	652	39	1260
27	52035	564	27	1120
28	25139	141	15	272

Table 5.4 Correlation Time Comparison for IDS Alert Correlation Models ordered by alerts count

Day	Alerts Count	CAM Time	ABCM Time	DPCM Time
20	11788	27	3	50
19	15175	52	9	102
28	25139	141	15	272
23	27753	171	18	281
11	28664	195	10	367
14	34303	233	23	434
17	34879	325	24	526
25	36072	278	20	543
22	39786	375	34	738
12	46703	436	28	842
21	49336	457	32	889
16	49609	504	39	1130
15	51823	447	35	982
27	52035	564	27	1120
13	54759	620	35	1159
26	57598	652	39	1260
24	70686	940	44	1863
18	152240	6048	103	10082

Figure 5.12 shows a chart of the daily correlation time for CAM, DPCM, and ABCM. The X-axis shows the daily alerts count; the Y-axis represents the correlation time in seconds. The results show that the correlation time increased linearly with increasing alerts count until the range of 70,000 alerts, while it increased exponentially in case of 152,240 alerts. The chart shows that ABCM has the lowest correlation time compared with CAM and DPCM techniques. DPCM has the highest correlation time compared with CAM and ABCM. DPCM timing depends on the correlation time of each stage, while CAM and ABCM timing depends on the correlation time of each component. The correlation time of each stage depends on the longest correlation time of the active components in that stage. The chart shows results varying from 3 seconds in the case of ABCM for minimum alerts count, to more than 10,000 seconds for DPCM technique in the case of 152,240 as maximum alert count. Detailed correlation times for alert counts less than 70,000 alerts (dashed red rectangular in Figure 5.11) will be shown in a separate chart.

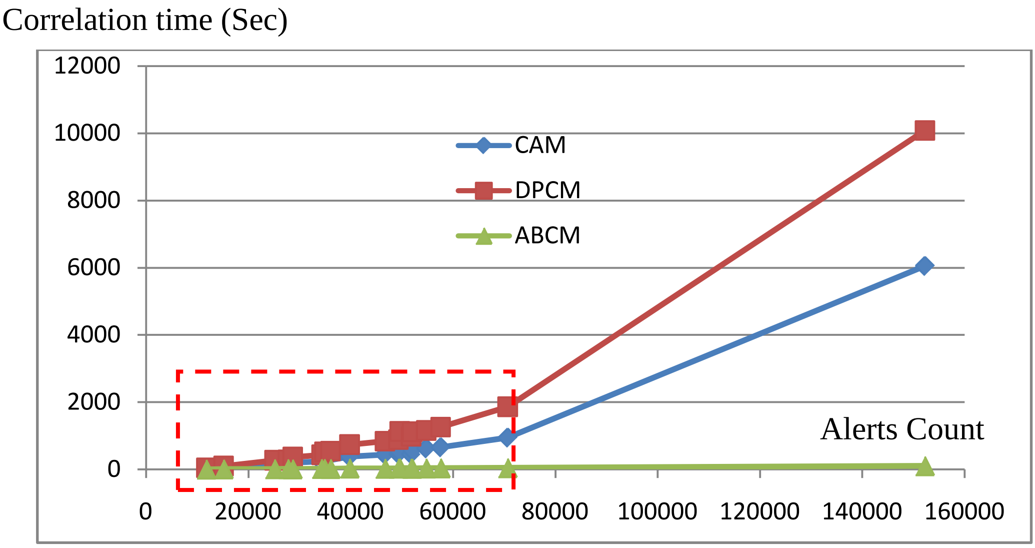


Figure 5.12 Correlation Times Comparison of IDS Correlation Techniques

Figure 5.13 presents a graph of correlation times for different IDS correlation techniques with excluding of maximum number of alerts of 152,240 alerts to show the detailed differences in correlation time for each technique for alerts less than 70,000 alerts.

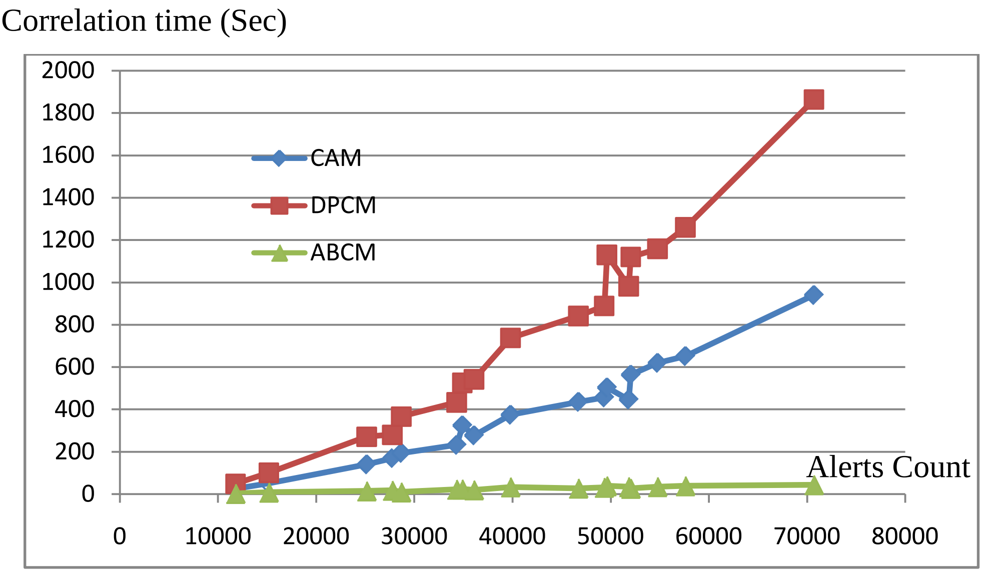


Figure 5.13 Correlation Times Comparison of IDS Correlation Techniques

ABCM has the lowest correlation time because it uses only effective components in ACCL, while CAM has a higher correlation time because it consumes time in ineffective components; DPCM has a higher correlation time because the use of stages increased every stage time compared with the use of components in the single processor environment used.

5.3 DACM Components Results

This section presents the result of different individual agents. These results depend on some parameters like previous period or certain IP address. Figure 5.14 shows SSH agent result which includes detected malicious activity within SSH service. The detected attacks described by ID, source IP, Date, Time, process ID, user, and description.

The SSH agent detects users who are trying to guess the username or password for SSH services or try to hide their browser information to prevent the system from identifying them.

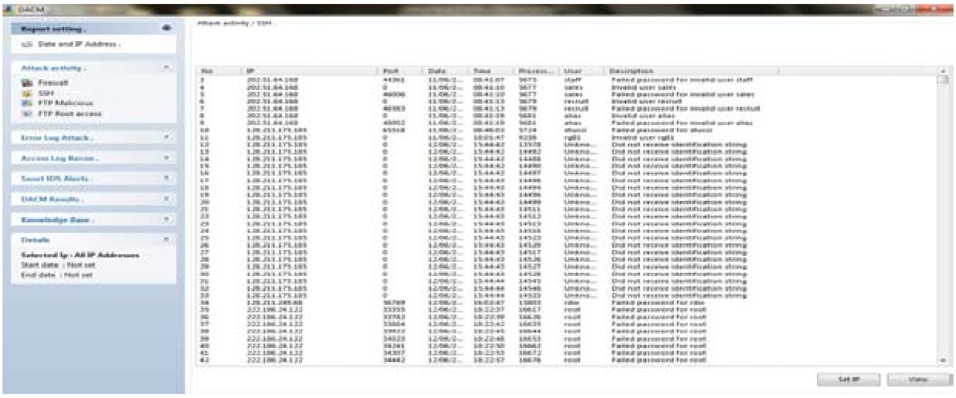


Figure 5.14 SSH Agent Result[12]

Other similar agent’s results for different services such as Firewall, FTP, FTP transfer, error log attack, access log, and system scan could be presented in such way. ABCM correlation for IDS alerts is included as IDS agent result to integrate with other agents results. Figure 5.15 shows ABCM results as a module in DACM including correlated alerts for specific IP address (98.194.16.97).

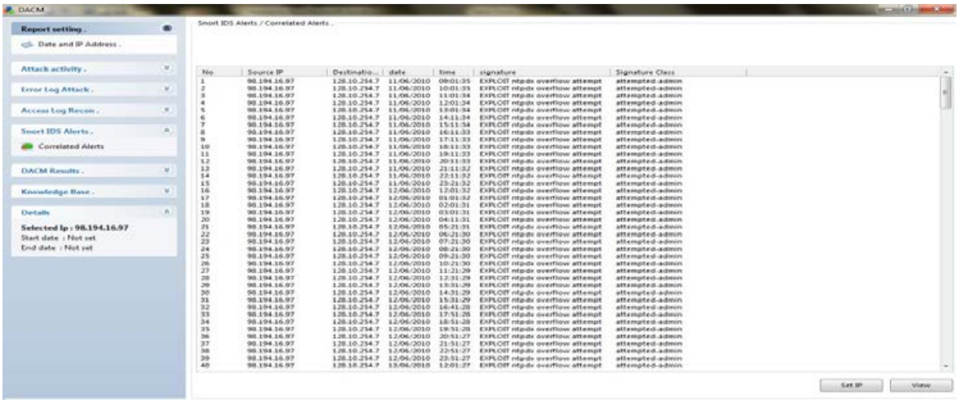


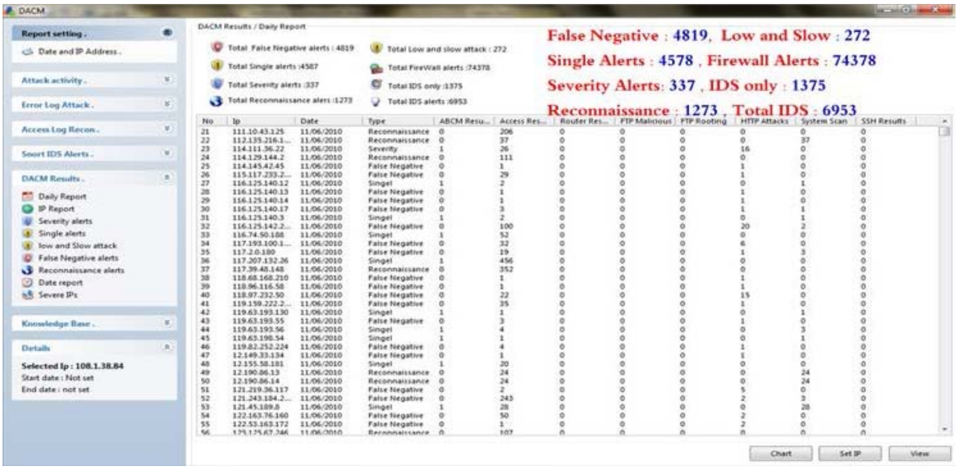
Figure 5.15 ABCM Result for Specific IP as part of DACM[13]

[12] Larger images of Figures 5.14 and 5.15 are included in Appendix A

ABCM correlated alerts include source IP, date, time, alert signature, and destination. Each agent result includes ID, Date, and Time, Source IP, and attack description. Three common attributes of all alerts from different agent result date, time, and attacker or source IP, these attributes could be used to integrate attacks done by same IP in same time. Integrating such kind of alerts together with IDS correlated alerts conclude the current situation of attempted intrusion to the system

5.4 DACM Central Agent Results

DACM central agent has rich valuable information from different individual agent's result. Using this information together, the central agent provides valuable reports summarizing the improvement in IDS capability. Figure 5.16 shows the daily report of DACM which concludes alerts and show their total classification. Daily alerts report is described by alert ID, source IP, Date, alert type, count of alerts from IDS correlated alerts for source IP. It also include other alerts from other source SSH, Firewall, for that source IP , and finally it conclude access log and system scan alerts which exceeds the allowed threshold value.



Selected Ip : 108.1.38.84
Start date : Not set
End date : not set

DACM Results / Daily Report

Total False Negative alerts : 4819
Total Single alerts :4587
Total Severity alerts :337
Total Reconnaissance alerts :1273
Total Low and slow attack : 272
Total FireWall alerts :74378
Total IDS only :1375
Total IDS alerts :6953

False Negative : 4819, Low and Slow : 272
Single Alerts : 4578 , Firewall Alerts : 74378
Severity Alerts: 337 , IDS only : 1375
Reconnaissance : 1273 , Total IDS : 6953

No	Ip	Date	Type	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
21	111.10.43.125	11/06/2010	Reconnaissance	0	206	0	0	0	0	0	0
22	112.135.216.1...	11/06/2010	Reconnaissance	0	37	0	0	0	0	37	0
23	114.111.36.22	11/06/2010	Severity	1	26	0	0	0	16	0	0
24	114.129.144.2	11/06/2010	Reconnaissance	0	111	0	0	0	0	0	0
25	114.145.42.45	11/06/2010	False Negative	0	1	0	0	0	1	0	0
26	115.117.233.2...	11/06/2010	False Negative	0	29	0	0	0	1	0	0
27	116.125.140.12	11/06/2010	Singel	1	2	0	0	0	0	1	0
28	116.125.140.13	11/06/2010	False Negative	0	1	0	0	0	1	0	0
29	116.125.140.14	11/06/2010	False Negative	0	1	0	0	0	1	0	0
30	116.125.140.17	11/06/2010	False Negative	0	3	0	0	0	1	1	0
31	116.125.140.3	11/06/2010	Singel	1	2	0	0	0	0	1	0
32	116.125.142.2...	11/06/2010	False Negative	0	100	0	0	0	20	2	0
33	116.74.50.188	11/06/2010	Singel	1	52	0	0	0	0	0	0
34	117.193.100.1...	11/06/2010	False Negative	0	32	0	0	0	6	0	0
35	117.2.0.180	11/06/2010	False Negative	0	19	0	0	0	1	3	0
36	117.207.132.26	11/06/2010	Singel	1	456	0	0	0	0	0	0
37	117.39.48.148	11/06/2010	Reconnaissance	0	352	0	0	0	0	0	0
38	118.68.168.210	11/06/2010	False Negative	0	1	0	0	0	1	0	0
39	118.96.116.58	11/06/2010	False Negative	0	1	0	0	0	1	0	0
40	118.97.232.50	11/06/2010	False Negative	0	22	0	0	0	15	0	0
41	119.159.222.2...	11/06/2010	False Negative	0	35	0	0	0	1	0	0
42	119.63.193.130	11/06/2010	Singel	1	1	0	0	0	0	1	0
43	119.63.193.55	11/06/2010	False Negative	0	3	0	0	0	1	0	0
44	119.63.193.56	11/06/2010	Singel	1	4	0	0	0	0	3	0
45	119.63.198.54	11/06/2010	Singel	1	1	0	0	0	0	1	0
46	119.82.252.224	11/06/2010	False Negative	0	4	0	0	0	1	0	0
47	12.149.33.134	11/06/2010	False Negative	0	1	0	0	0	1	0	0
48	12.155.58.181	11/06/2010	Singel	1	20	0	0	0	0	0	0
49	12.190.86.13	11/06/2010	Reconnaissance	0	24	0	0	0	0	24	0
50	12.190.86.14	11/06/2010	Reconnaissance	0	24	0	0	0	0	24	0
51	121.219.36.117	11/06/2010	False Negative	0	2	0	0	0	5	0	0
52	121.243.184.2...	11/06/2010	False Negative	0	243	0	0	0	2	3	0
53	121.45.189.8	11/06/2010	Singel	1	28	0	0	0	0	28	0
54	122.163.76.160	11/06/2010	False Negative	0	50	0	0	0	2	0	0
55	122.53.163.172	11/06/2010	False Negative	0	1	0	0	0	2	0	0
56	125.125.67.246	11/06/2010	Reconnaissance	0	107	0	0	0	0	0	0

Figure 5.16 DACM Daily Results[13]

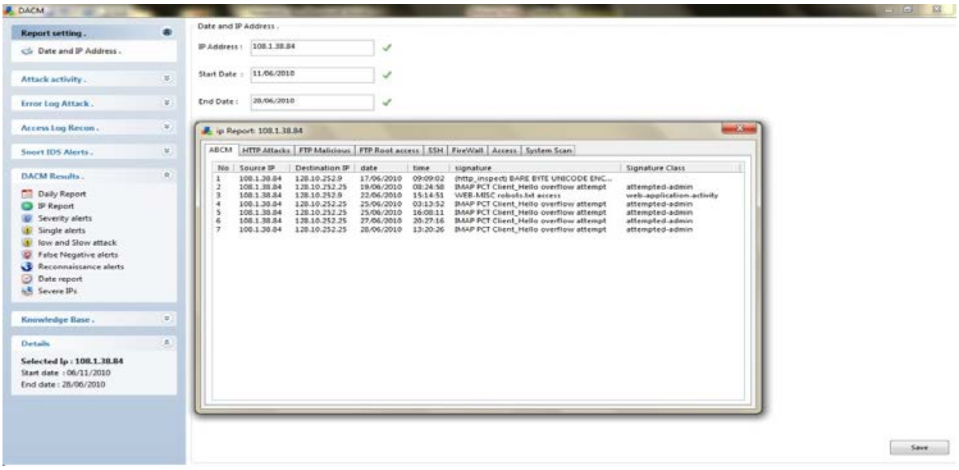
[13] Larger Image of Figure 5.16 is included in Appendix A

The alert type is driven from the integration of different agent's result for the same IP. False Negative alerts during the whole period of test were 4819 alerts. The alerts were not detected by IDS through its correlated alerts and discovered by other agent's results. The false negative alerts show which agent detected it from FTP, SSH, and http attack. Severity alerts during the whole period of the test were 337 alerts. The alerts were detected by IDS through its correlated alerts and also discovered by other agents result. The severity alerts show which other agent detects it from firewall, FTP, SSH, and http attack added to IDS alerts.

IDS alerts during the whole period of the test were 1375 alerts. The alerts were detected more than one time for the same IP by only IDS through its correlated alerts and not discovered by any of the other agents' results Single alerts during the whole period of test were 4578 alerts. The alerts were detected by only IDS through its correlated alerts just one time during the whole period of test and never repeated and not discovered by any of the other agents' results. Single alerts are stored for a while to be analyzed for detection of low and slow attack. Total IDS alerts were 6953 which is equal to IDS only alerts added to single alerts.

Firewall alerts represent summarized information in daily result report. The count of firewall alerts during the whole period of test was 74378 alerts. The alerts were detected only by firewall locked address within the test period. Reconnaissance alerts conclude the trial of gathering data about the network through unhallowed access of system or trying to scan the system to discover the operating system. The count of reconnaissance alerts during the whole period of test was 1273 alerts. Different detailed reports for each alert type could be displayed within specific period or IP address as a report parameter.

DACM agent has different reports to trace a specific IP address to conclude the trial and attack signature for this IP in different source of information. Figures 5.17 and 5.18 show an example of IP report which show that IP “108.1.38.84” was detected by ABCM correlation as IDS alerts and the same IP appeared in the firewall agent result as blocked IP, also it was detected by http attack in the same time window in one case.



Date and IP Address .

IP Address : 108.1.38.84

Start Date : 11/06/2010

End Date : 28/06/2010

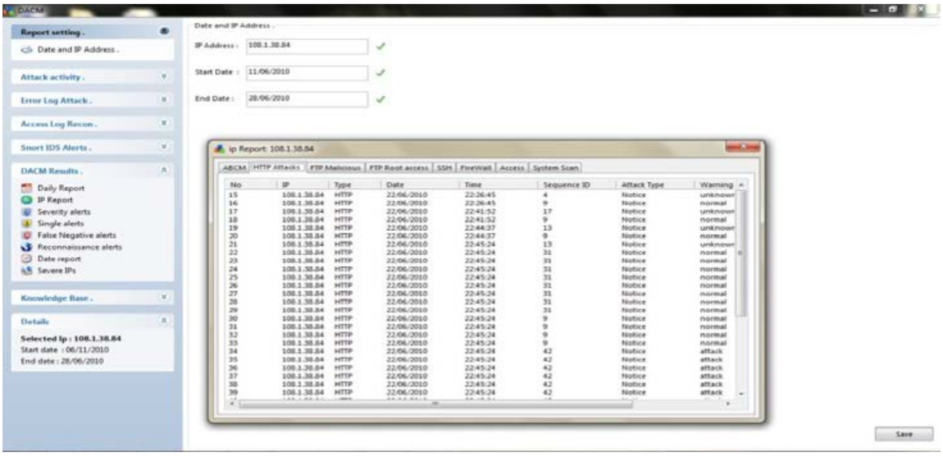
Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

ip Report: 108.1.38.84

No	Source IP	Destination IP	date	time	signature	Signature Class
1	108.1.38.84	128.10.252.9	17/06/2010	09:09:02	(http_inspect) BARE BYTE UNICODE ENC...	
2	108.1.38.84	128.10.252.25	19/06/2010	08:24:58	IMAP PCT Client_Hello overflow attempt	attempted-admin
3	108.1.38.84	128.10.252.9	22/06/2010	15:14:51	WEB-MISC robots.txt access	web-application-activity
4	108.1.38.84	128.10.252.25	25/06/2010	03:13:52	IMAP PCT Client_Hello overflow attempt	attempted-admin
5	108.1.38.84	128.10.252.25	25/06/2010	16:08:11	IMAP PCT Client_Hello overflow attempt	attempted-admin
6	108.1.38.84	128.10.252.25	27/06/2010	20:27:16	IMAP PCT Client_Hello overflow attempt	attempted-admin
7	108.1.38.84	128.10.252.25	28/06/2010	13:20:26	IMAP PCT Client_Hello overflow attempt	attempted-admin

Figure 5.17 DACM IP Report form ABCM IDS correlated alerts[14]

In other cases it was detected in http attack while not detected in IDS because of the use of IDS evasion technique. In other cases it was detected only in IDS and was not detected by http attack because of the attack nature.



Date and IP Address .

IP Address : 108.1.38.84

Start Date : 11/06/2010

End Date : 28/06/2010

Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

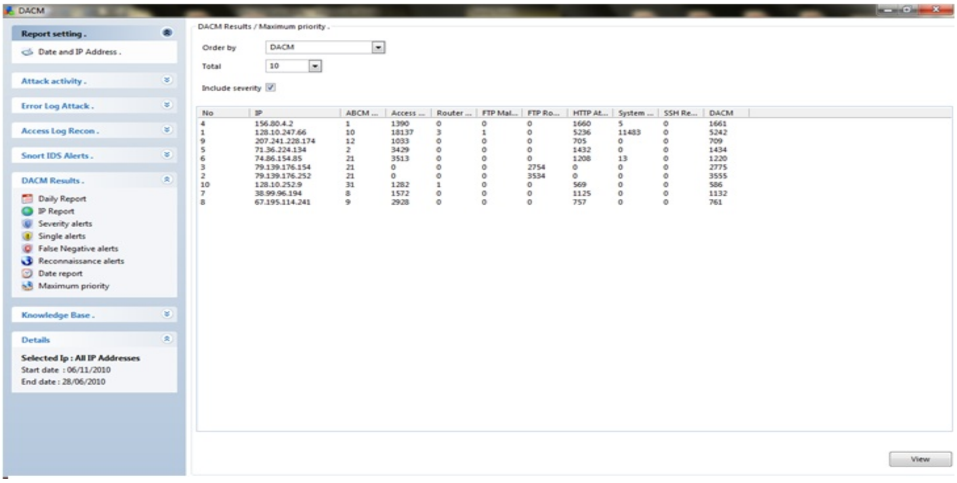
ip Report: 108.1.38.84

No	IP	Type	Date	Time	Sequence ID	Attack Type	Warning
15	108.1.38.84	HTTP	22/06/2010	22:26:45	4	Notice	unknown
16	108.1.38.84	HTTP	22/06/2010	22:26:45	9	Notice	normal
17	108.1.38.84	HTTP	22/06/2010	22:41:52	17	Notice	unknown
18	108.1.38.84	HTTP	22/06/2010	22:41:52	9	Notice	normal
19	108.1.38.84	HTTP	22/06/2010	22:44:37	13	Notice	unknown
20	108.1.38.84	HTTP	22/06/2010	22:44:37	9	Notice	normal
21	108.1.38.84	HTTP	22/06/2010	22:45:24	13	Notice	unknown
22	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
23	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
24	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
25	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
26	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
27	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
28	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
29	108.1.38.84	HTTP	22/06/2010	22:45:24	31	Notice	normal
30	108.1.38.84	HTTP	22/06/2010	22:45:24	9	Notice	normal
31	108.1.38.84	HTTP	22/06/2010	22:45:24	9	Notice	normal
32	108.1.38.84	HTTP	22/06/2010	22:45:24	9	Notice	normal
33	108.1.38.84	HTTP	22/06/2010	22:45:24	9	Notice	normal
34	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack
35	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack
36	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack
37	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack
38	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack
39	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack

Figure 5.18 DACM IP Report form HTTP attack[15]

[14] Larger images of Figures 5.17 and 5.18 are included in Appendix A

DACM decreases the audit load and the time cost required to obtain effective situational understanding of the network through displaying the most repeated IPs as sources for different attacks. Figure 5.19 shows maximum priority report, which display the most common IPs which were detected by different agent's results or by DACM central agent. The reports shows the top 10 IPs which have the highest count of repeated alerts for different alerts type and/or reconnaissance activity.



Selected Ip : All IP Addresses
Start date : 06/11/2010
End date : 28/06/2010

DACM Results / Maximum priority .

Order by: DACM
Total: 10
Include severity ☑

No	IP	ABCM ...	Access ...	Router ...	FTP Mal...	FTP Ro...	HTTP At...	System ...	SSH Re...	DACM
4	156.80.4.2	1	1390	0	0	0	1660	5	0	1661
1	128.10.247.66	10	18137	3	1	0	5236	11483	0	5242
9	207.241.228.174	12	1033	0	0	0	705	0	0	709
5	71.36.224.134	2	3429	0	0	0	1432	0	0	1434
6	74.86.154.85	21	3513	0	0	0	1208	13	0	1220
3	79.139.176.154	21	0	0	0	2754	0	0	0	2775
2	79.139.176.252	21	0	0	0	3534	0	0	0	3555
10	128.10.252.9	31	1282	1	0	0	569	0	0	586
7	38.99.96.194	8	1572	0	0	0	1125	0	0	1132
8	67.195.114.241	9	2928	0	0	0	757	0	0	761

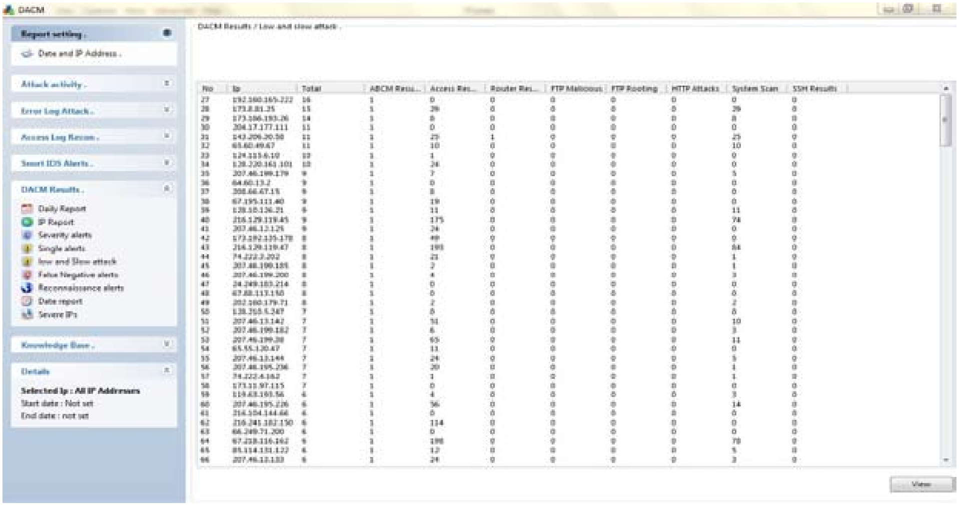
Figure 5.19 DACM Maximum Priority Report[15]

DACM detects low and slow attacks which occur over several days, and classify them depending on their source of detection. Figure 5.20 shows Low and Slow summary report within the test period, it includes source IP of low and slow attack and count of daily detected single alerts for that IP.

Figure 5.21 shows the detailed low and slow attacks for IP 192.160.165.222. DACM detected 16 single alerts for this IP in 16 different days. Figure 5.22 shows other low and slow attacks and other reconnaissance activity for same IP. The IP 216.129.119.45 has 9

[15] Larger Image of Figure 5.19 is included in Appendix A

individual single alerts and system access and scan greater than threshold values.

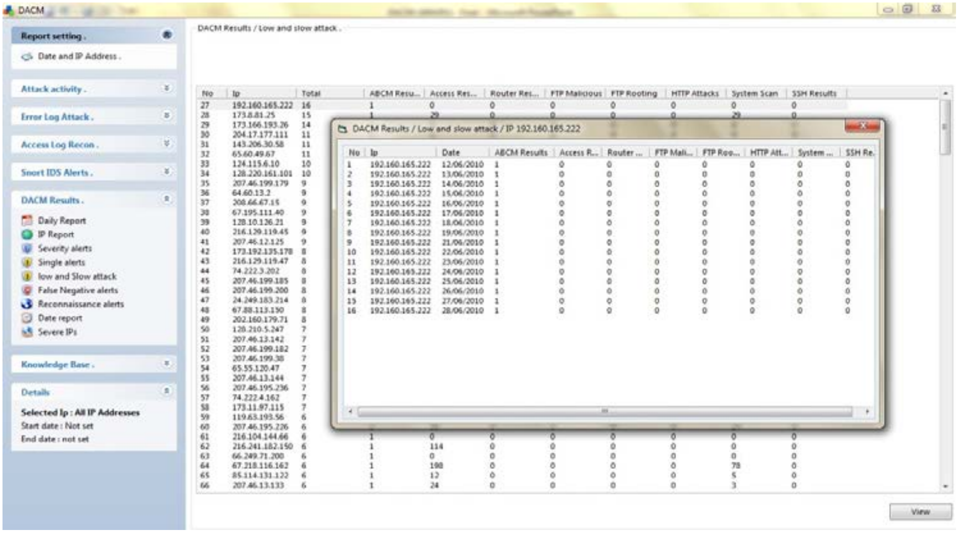


Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Low and slow attack .

No	Ip	Total	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
27	192.160.165.222	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14	1	8	0	0	0	0	8	0
30	204.17.177.111	11	1	0	0	0	0	0	0	0
31	143.206.30.58	11	1	25	1	0	0	0	25	0
32	65.60.49.67	11	1	10	0	0	0	0	10	0
33	124.115.6.10	10	1	1	0	0	0	0	0	0
34	128.220.161.101	10	1	24	0	0	0	0	0	0
35	207.46.199.179	9	1	7	0	0	0	0	5	0
36	64.60.13.2	9	1	0	0	0	0	0	0	0
37	208.66.67.15	9	1	8	0	0	0	0	0	0
38	67.195.111.40	9	1	19	0	0	0	0	0	0
39	128.10.126.21	9	1	11	0	0	0	0	11	0
40	216.129.119.45	9	1	175	0	0	0	0	78	0
41	207.46.12.125	9	1	24	0	0	0	0	0	0
42	173.192.135.178	8	1	49	0	0	0	0	0	0
43	216.129.119.47	8	1	193	0	0	0	0	84	0
44	74.222.3.202	8	1	21	0	0	0	0	1	0
45	207.46.199.185	8	1	2	0	0	0	0	1	0
46	207.46.199.200	8	1	4	0	0	0	0	3	0
47	24.249.183.214	8	1	0	0	0	0	0	0	0
48	67.88.113.150	8	1	0	0	0	0	0	0	0
49	202.160.179.71	8	1	2	0	0	0	0	2	0
50	128.210.5.247	7	1	8	0	0	0	0	8	0
51	207.46.13.142	7	1	51	0	0	0	0	10	0
52	207.46.199.182	7	1	6	0	0	0	0	3	0
53	207.46.199.38	7	1	65	0	0	0	0	11	0
54	65.55.120.47	7	1	11	0	0	0	0	0	0
55	207.46.13.144	7	1	24	0	0	0	0	5	0
56	207.46.195.236	7	1	20	0	0	0	0	1	0
57	74.222.4.162	7	1	1	0	0	0	0	1	0
58	173.11.97.115	7	1	0	0	0	0	0	0	0
59	119.63.193.56	6	1	4	0	0	0	0	3	0
60	207.46.195.226	6	1	56	0	0	0	0	14	0
61	216.104.144.66	6	1	0	0	0	0	0	0	0
62	216.241.182.150	6	1	114	0	0	0	0	0	0
63	66.249.71.200	6	1	0	0	0	0	0	0	0
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	85.114.131.122	6	1	12	0	0	0	0	5	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0

Figure 5.20 Low and Slow Attack Summary[16]



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Low and slow attack .

No	Ip	Total	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
27	192.160.165.222	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14								
30	204.17.177.111	11								
31	143.206.30.58	11								
32	65.60.49.67	11								
33	124.115.6.10	10								
34	128.220.161.101	10								
35	207.46.199.179	9								
36	64.60.13.2	9								
37	208.66.67.15	9								
38	67.195.111.40	9								
39	128.10.126.21	9								
40	216.129.119.45	9								
41	207.46.12.125	9								
42	173.192.135.178	8								
43	216.129.119.47	8								
44	74.222.3.202	8								
45	207.46.199.185	8								
46	207.46.199.200	8								
47	24.249.183.214	8								
48	67.88.113.150	8								
49	202.160.179.71	8								
50	128.210.5.247	7								
51	207.46.13.142	7								
52	207.46.199.182	7								
53	207.46.199.38	7								
54	65.55.120.47	7								
55	207.46.13.144	7								
56	207.46.195.236	7								
57	74.222.4.162	7								
58	173.11.97.115	7								
59	119.63.193.56	6								
60	207.46.195.226	6								
61	216.104.144.66	6	1	0	0	0	0	0	0	0
62	216.241.182.150	6	1	114	0	0	0	0	0	0
63	66.249.71.200	6	1	0	0	0	0	0	0	0
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	85.114.131.122	6	1	12	0	0	0	0	5	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0

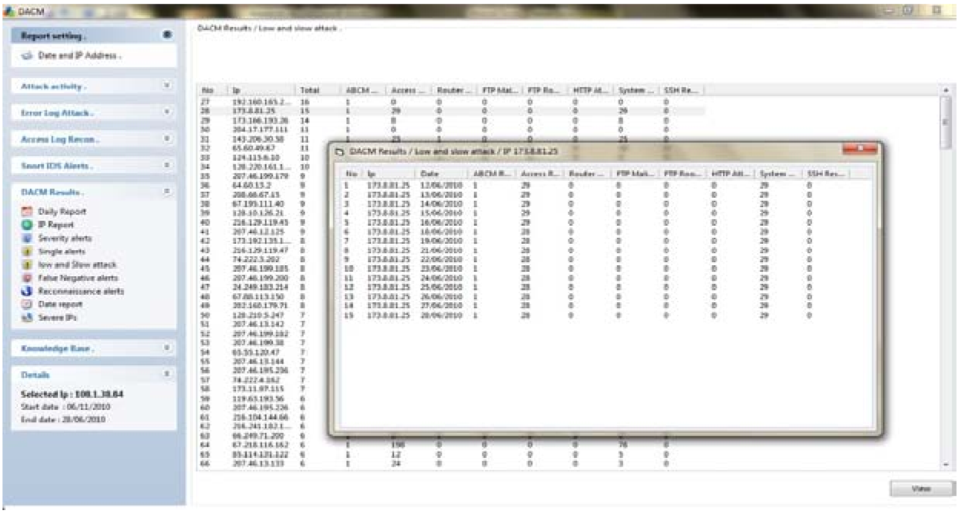
DACM Results / Low and slow attack / IP 192.160.165.222

No	Ip	Date	ABCM Results	Access R...	Router ...	FTP Mali...	FTP Roo...	HTTP Att...	System ...	SSH Re...
1	192.160.165.222	12/06/2010	1	0	0	0	0	0	0	0
2	192.160.165.222	13/06/2010	1	0	0	0	0	0	0	0
3	192.160.165.222	14/06/2010	1	0	0	0	0	0	0	0
4	192.160.165.222	15/06/2010	1	0	0	0	0	0	0	0
5	192.160.165.222	16/06/2010	1	0	0	0	0	0	0	0
6	192.160.165.222	17/06/2010	1	0	0	0	0	0	0	0
7	192.160.165.222	18/06/2010	1	0	0	0	0	0	0	0
8	192.160.165.222	19/06/2010	1	0	0	0	0	0	0	0
9	192.160.165.222	21/06/2010	1	0	0	0	0	0	0	0
10	192.160.165.222	22/06/2010	1	0	0	0	0	0	0	0
11	192.160.165.222	23/06/2010	1	0	0	0	0	0	0	0
12	192.160.165.222	24/06/2010	1	0	0	0	0	0	0	0
13	192.160.165.222	25/06/2010	1	0	0	0	0	0	0	0
14	192.160.165.222	26/06/2010	1	0	0	0	0	0	0	0
15	192.160.165.222	27/06/2010	1	0	0	0	0	0	0	0
16	192.160.165.222	28/06/2010	1	0	0	0	0	0	0	0

Figure 5.21 Low and Slow Attack for 192.160.165.222[17]

[16] Larger Images of Figures 5.20 and 5.21 are included in Appendix A

DACM summarizes the total daily alerts and classify them depending on their source of detection. Figure 5.23 shows DACM summary report, it includes daily alerts count of the test period, count of alerts detected by IDS which are correlated by ABCM and other alerts which were missed from IDS and detected by other agents such SSH, FTP, and http attacks.



Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

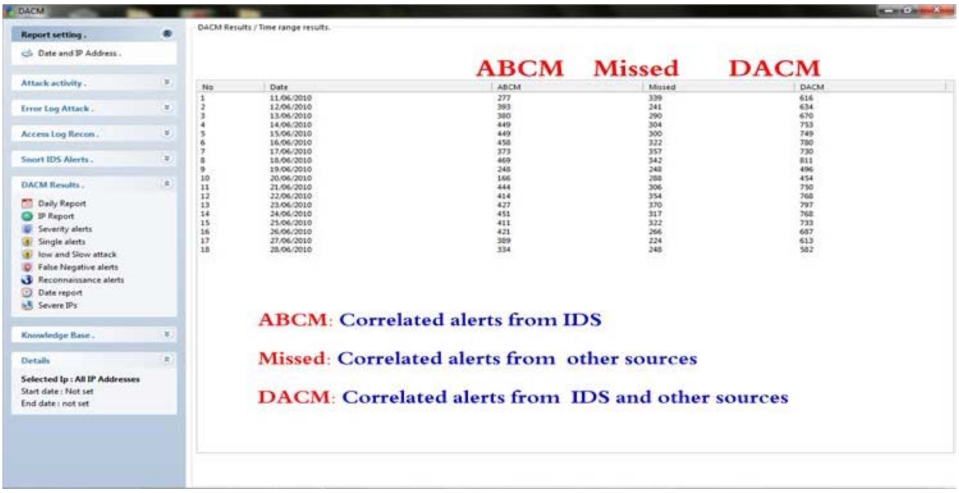
DACM Results / Low and slow attack .

No	Ip	Total	ABCM ...	Access ...	Router ...	FTP Mal...	FTP Ro...	HTTP At...	System ...	SSH Re...
27	192.160.165.2...	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14	1	8	0	0	0	0	8	0
30	204.17.177.111	11	1	0	0	0	0	0	0	0
31	143.206.30.58	11	1	25	1	0	0	0	25	0
32	65.60.49.67	11								
33	124.115.6.10	10								
34	128.220.161.1...	10								
35	207.46.199.179	9								
36	64.60.13.2	9								
37	208.68.67.15	9								
38	67.195.111.40	9								
39	128.10.126.21	9								
40	216.129.119.45	9								
41	207.46.12.125	9								
42	173.192.135.1...	8								
43	216.129.119.47	8								
44	74.222.3.202	8								
45	207.46.199.185	8								
46	207.46.199.200	8								
47	24.249.183.214	8								
48	67.88.113.150	8								
49	202.160.179.71	8								
50	128.210.5.247	7								
51	207.46.13.142	7								
52	207.46.199.182	7								
53	207.46.199.38	7								
54	65.55.120.47	7								
55	207.46.13.144	7								
56	207.46.195.236	7								
57	74.222.4.162	7								
58	173.11.87.115	7								
59	129.63.193.56	6								
60	207.46.195.226	6								
61	216.104.144.66	6								
62	216.241.182.1...	6								
63	66.249.71.209	6								
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	83.114.121.122	6	1	12	0	0	0	0	3	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0

DACM Results / Low and slow attack / IP 173.8.81.25

No	Ip	Date	ABCM R...	Access R...	Router ...	FTP Mal...	FTP Rea...	HTTP Att...	System ...	SSH Res...
1	173.8.81.25	12/06/2010	1	29	0	0	0	0	29	0
2	173.8.81.25	13/06/2010	1	29	0	0	0	0	29	0
3	173.8.81.25	14/06/2010	1	29	0	0	0	0	29	0
4	173.8.81.25	15/06/2010	1	29	0	0	0	0	29	0
5	173.8.81.25	16/06/2010	1	29	0	0	0	0	29	0
6	173.8.81.25	18/06/2010	1	28	0	0	0	0	29	0
7	173.8.81.25	19/06/2010	1	28	0	0	0	0	29	0
8	173.8.81.25	21/06/2010	1	28	0	0	0	0	29	0
9	173.8.81.25	22/06/2010	1	28	0	0	0	0	29	0
10	173.8.81.25	23/06/2010	1	28	0	0	0	0	29	0
11	173.8.81.25	24/06/2010	1	28	0	0	0	0	29	0
12	173.8.81.25	25/06/2010	1	28	0	0	0	0	29	0
13	173.8.81.25	26/06/2010	1	28	0	0	0	0	29	0
14	173.8.81.25	27/06/2010	1	28	0	0	0	0	29	0
15	173.8.81.25	28/06/2010	1	28	0	0	0	0	29	0

Figure 5.22 Low and Slow Attack for 216.129.119.45[17]



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Time range results.

No	Date	ABCM	Missed	DACM
1	11/06/2010	277	339	616
2	12/06/2010	393	241	634
3	13/06/2010	380	290	670
4	14/06/2010	449	304	753
5	15/06/2010	449	300	749
6	16/06/2010	458	322	780
7	17/06/2010	373	357	730
8	18/06/2010	469	342	811
9	19/06/2010	248	248	496
10	20/06/2010	166	288	454
11	21/06/2010	444	306	750
12	22/06/2010	414	354	768
13	23/06/2010	427	370	797
14	24/06/2010	451	317	768
15	25/06/2010	411	322	733
16	26/06/2010	421	266	687
17	27/06/2010	389	224	613
18	28/06/2010	334	248	582

ABCM: Correlated alerts from IDS

Missed: Correlated alerts from other sources

DACM: Correlated alerts from IDS and other sources

Figure 5.23 DACM Summary Report[18]

[17] Larger Images of Figures 5.22 and 5.23 are included in Appendix A

Third column shows the total alerts detected by DACM which is calculated from addition of IDS alerts to other agent’s alerts. DACM enhances IDS completeness through detecting the false negative alerts which were missed from the IDS alerts. Table 5.5 summarizes daily alerts count and number of IDS detection and missed alerts from IDS which have been detected by other log agents, and the total of them.

Table 5.5 DACM Summary Result

Day	Alerts	IDS Detection	Other logs Detection	DACM
11	28664	277	339	616
12	46703	393	241	634
13	54759	380	290	670
14	34303	449	304	753
15	51823	449	300	749
16	49609	458	322	780
17	34879	373	357	730
18	152240	469	342	811
19	15175	248	248	496
20	11788	166	288	454
21	49336	444	306	750
22	39786	414	354	768
23	27753	427	370	797
24	70686	451	317	768
25	36072	411	322	733
26	57598	421	266	687
27	52035	389	224	613
28	25139	334	248	582

The minimum alerts count was 11788 alerts in day 20; the detected correlated alerts from IDS were 166 alerts. DACM detected 288 alerts from other log agents in the same day. The maximum alerts count was 152240 alerts in day 18; the detected correlated alerts from IDS were 469 alerts. DACM detected 342 alerts from other log agents in the same day.

Table 5.6 DACM Percentage Summary Result

Day	**Alerts**	**IDS %**	**Other Logs %**	**DACM %**
11	28664	45	55	100
12	46703	62	38	100
13	54759	57	43	100
14	34303	60	40	100
15	51823	60	40	100
16	49609	59	41	100
17	34879	51	49	100
18	152240	58	42	100
19	15175	50	50	100
20	11788	37	63	100
21	49336	59	41	100
22	39786	54	46	100
23	27753	54	46	100
24	70686	59	41	100
25	36072	56	44	100
26	57598	61	39	100
27	52035	63	37	100
28	25139	57	43	100

Table 5.6 summarizes the percentage of detection of daily alerts count and number of IDS detection to the percentage of missed alerts from IDS which have been detected by other log agents, and the total of both of them. The average percentage alerts for DIS alerts percentage was 56% and was 44% percentages for missed alerts in case of average daily alerts of 46574 alerts.

Figure 5.24 shows a graph chart of number of IDS detection as IDS and missed alerts from IDS which have been detected by other log agents as other log, and the total for both of them as DACM. The x-axis represents the daily IDS alerts count while the y-axis represents the count of detected alerts. The graph shows that the use of other agents in DACM enhances the detection rate of missed alerts.

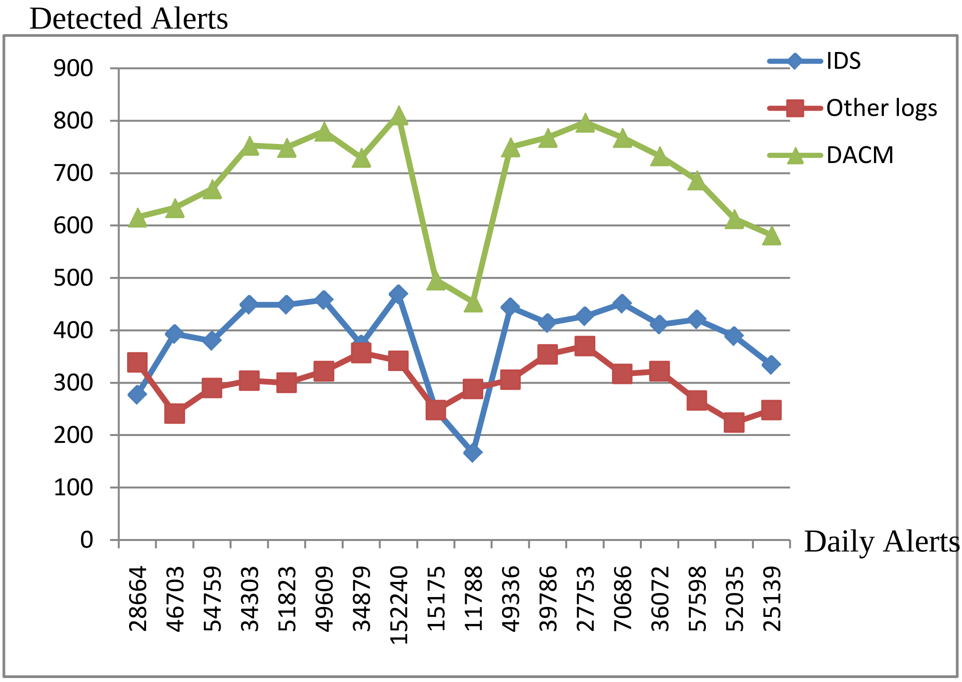


Figure 5.24 DACM Summary Results Chart

Figure 5.25 shows a graph chart of percentages of IDS detection in blue color and missed alerts from IDS which have been detected by other log agents in red color, and the total for both of them in green color. The x-axis represents the daily IDS alerts count while the y-axis represents the percentage of count of detected alerts to the total detected alerts by DACM. Total detected alerts by DACM represent the complete unit for both kinds of detection. The graph shows that the use of other agents in DACM enhances the detection rate of missed alerts by 44% compared with the case of using just IDS correlation.

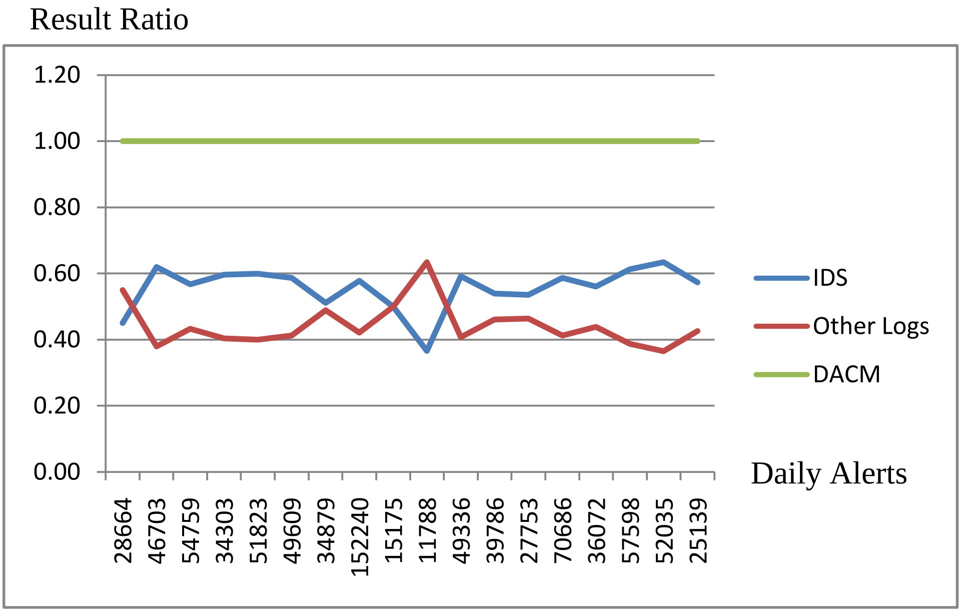


Figure 5.25 DACM Percentage Summary Results Chart

5.5 DACM Evaluation and Assessment

In this section we will present summary of the model assessment and implementation issues, DACM capabilities, DACM limitation, and needed consideration for implementation will be presented.

5.5.1 DACM Limitation

Additional DACM learning is needed to build more accurate behavioural profiles which determine attack signatures in system and application log files. Multi-step attack scenarios also need more learning to build pre-condition and post-condition tables for such attacks. DACM alerts are considered on an equal footing, and aren’t considered the influencing factors of different alerts on the same information system. Research is needed to distinguish between such alerts and assign weights for each alert type depending on its source of information and influence of the provided service. The assurance and quality of information from different agents is needed to avoid existing of fake agents and limiting the practical implementation.

The implemented model could be attacked by someone who knows the model idea by one of the following methods: Modification of log records, changing his behaviour to avoid learned profiles, sending malicious data, generating false alerts, or compromise a system to generate large amount of data to hide his activity.

5.5.2 DACM Assessment

DACM improves IDS capability through the use of different sources of information. False positive alerts are reduced because of the verification of alerts detected by IDS from other sources like firewalls, different log attacks, or http attacks. False negative alerts are reduced because missed alerts from IDSs are detected from other logs such as SSH, FTP, and http attacks. DACM enables early detection of trials to gather data which represent the first phase of advanced persistent threat and individual alerts for Low and Slow attack.

Previous correlation techniques were limited to the use of IDS alerts for correlation and enhancing correlation component performance. Few techniques [35, 36] used vulnerability scanners to assure alert verification.

Using the simplicity of the relationship between individual agents, it is an easy and simple task for each individual agent to correlate its alerts and shares its output with other agents. This approach reduces the overhead and enables the ideal use of system resources such as memory and CPU. DACM enables minimum correlation time of ABCM as IDS alerts correlation technique and allows continuous adaptive learning to update

ACCL, assuring the use of suitable correlation components for different datasets.

The DACM central agent accesses the results tables of other agents from central database, reducing network traffic compared with the case of accessing them from multiple machines or accessing the information source itself. DACM is ready for real time operation with minor modification in agent programs; the current proposed prototype was implemented after collecting the dataset, so it was not possible to run it as a real time model.

5.6 Practical Implementation Issues

The implementation of DACM model must reflect that the DACM is real time system; faster hardware produces better results in suitable time. The structure of the model can be unified hierarchical system or can be divided into group of smaller distributed systems. The structure nature must consider the communication overhead and the number of nodes and information source (500 IDS, 300 log file, 100 Firewall, etc).

The expected time to implement such system will depend on the availability of qualified engineers and programmers to build the needed learning systems, and the availability of the proper hardware. DACM is platform independent; it can be implemented using windows, Linux, or UNIX operating system. It also can be implemented on network of PCs or sun workstation.

DACM is scalable system, but more scalability analysis is needed to determine the maximum number of monitored nodes. The larger numbers of nodes will require complex communication design and high rate of sending data which may affect the system performance. The central agent performance may be affected with huge number of nodes, the idea of implementing multiple hierarchical small systems and exchange data between each subsystem central agent may be better for huge number of nodes.

CHAPTER 6

Conclusions and Future Work

Chapter Six: Conclusions and Future Work

6.1 Conclusions

This dissertation proved that it is possible to enhance both IDS Accuracy and IDS Completeness through reducing either False Positive or False Negative alerts using correlation between different available information sources in the system and network environment. The dissertation presented a Distributed Agent Correlation Model (DACM) providing a scalable alert correlation for large scale networks. The model utilizes multiple distributed agents to provide an integrated correlation solution. The model can be extended by creating new correlation agents, and can be tailored to a protected network by selecting what agents to use and configuring each individual agent's parameters. DACM correlates alerts from IDSs with other information source such as INFOSEC tools and system and application log files.

A collection of datasets was used to evaluate the correlation system. The datasets were collected on networks with a variety of services and includes real networks, networks specifically constructed for dataset gathering, and simulated networks. The collected datasets are real-world datasets, with real attacks in addition to some simulated attacks to build behavioral profiles, since no cooperation from the attacker can be assumed. The intentions of the attackers were deduced from the gathered datasets.

Agent's proposed models and algorithms have been implemented, analyzed, and evaluated to measure detection and correlation rate and reduction of false positive and false negative alerts.

This dissertation proposed two alternative models to enhance the IDS alert correlation process: an Agent Based Correlation Model (ABCM) and a Dynamic Parallel Correlation Model (DPCM).

The ABCM works through a learning phase and correlation phase. During the learning phase, Learning Agent (LA) learns the nature of the alert datasets and effective correlation components and their Reduction Rate (RR) and builds an Active Correlation Component List (ACCL). The ACCL contains the effective correlation components in descending order of their RR. Depending on the learning phase, the agent controls the correlation process during the correlation phase using the implemented ACCL. The order of correlation starts with components with higher RRs in ACCL followed by lower RRs until correlation by the last component in ACCL.

DPCM has parallel processing correlation to assure using the suitable component and its order. It consists of correlation stages with each stage consisting of a set of correlation components. The proposed model dynamically selects the optimum order of the needed correlation components depending on the working environment. The input to each stage is the output of the correlation component with the highest RR in the previous stage. In the next stage, the higher RR component and components which have zero value RRs will be disabled. The optimal components order minimize the number of processed alerts in each stage by starting from higher to lower reduction rate components. ABCM is scalable regarding the number of correlation components in ACCL, while DPCM is scalable regarding the number of correlation components in each stage. Using threads in PDCM optimizes usage of memory and processor during correlation process.

The results showed that ABCM and DPCM have similar RRs as CAM, while ABCM has the lowest correlation time and DPCM has the highest correlation time. That means ABCM maintains the same correlation accuracy provided by CAM in less time and less number of components. While it has the longest time in our single processor implementation, DPCM is expected to have the lower time if fully parallel architecture is used. DPCM has the highest reduction rate with minor differences than ABCM and CAM.

Firewall log file was used as INFOSEC tools information example, firewall agent reads router log files and summarizes the blocked IP in firewall tables. SSH, FTP, and error logs were used as system and application log files information example, these logs agents read the related log files, and extract the attack signature in each file to its output tables. Each agent has a previously learned pattern to determine the normal versus attack behavior in the log file contents. Access log agents and OS log agents determine when other users are trying to gather information about the protected network contents or the services provided and the used operating systems.

The DACM central agent correlates the output of ABCM as IDS alert correlation with other agent's output. The results show that DACM enhances both the accuracy and the completeness of intrusion detections by reducing false positive and false negative alerts through the integration of these alerts from multiple information sources. DACM supports an adaptive continuous learning capability by providing profiles which have never been learned as normal or attack behavior to the system administrator to classify these profiles.

DACM decreases the audit load and the time cost required to obtain effective situational understanding of the network. DACM is scalable for large scale networks; many different agents can be added to expand the area of detection by different attacks. The results show that DACM provides 44% better intrusion detection than other IDS techniques through the detection of new attacks which were not detected by IDSs. It also showed that DACM detected low and slow attacks and reconnaissance trials by external users. These reconnaissance trials are a signature of early detection of Advanced Persistent Threats. DACM can be used to detect Zero Day Attacks through detection of any malicious behavior compared with normal network behavior.

Finally, DACM could be used as a real time system with minor modifications to the current implementation to allow continuous online correlation for individual and central agents. The model presented in this dissertation is a promising approach which combines use of correlation techniques and agent technology.

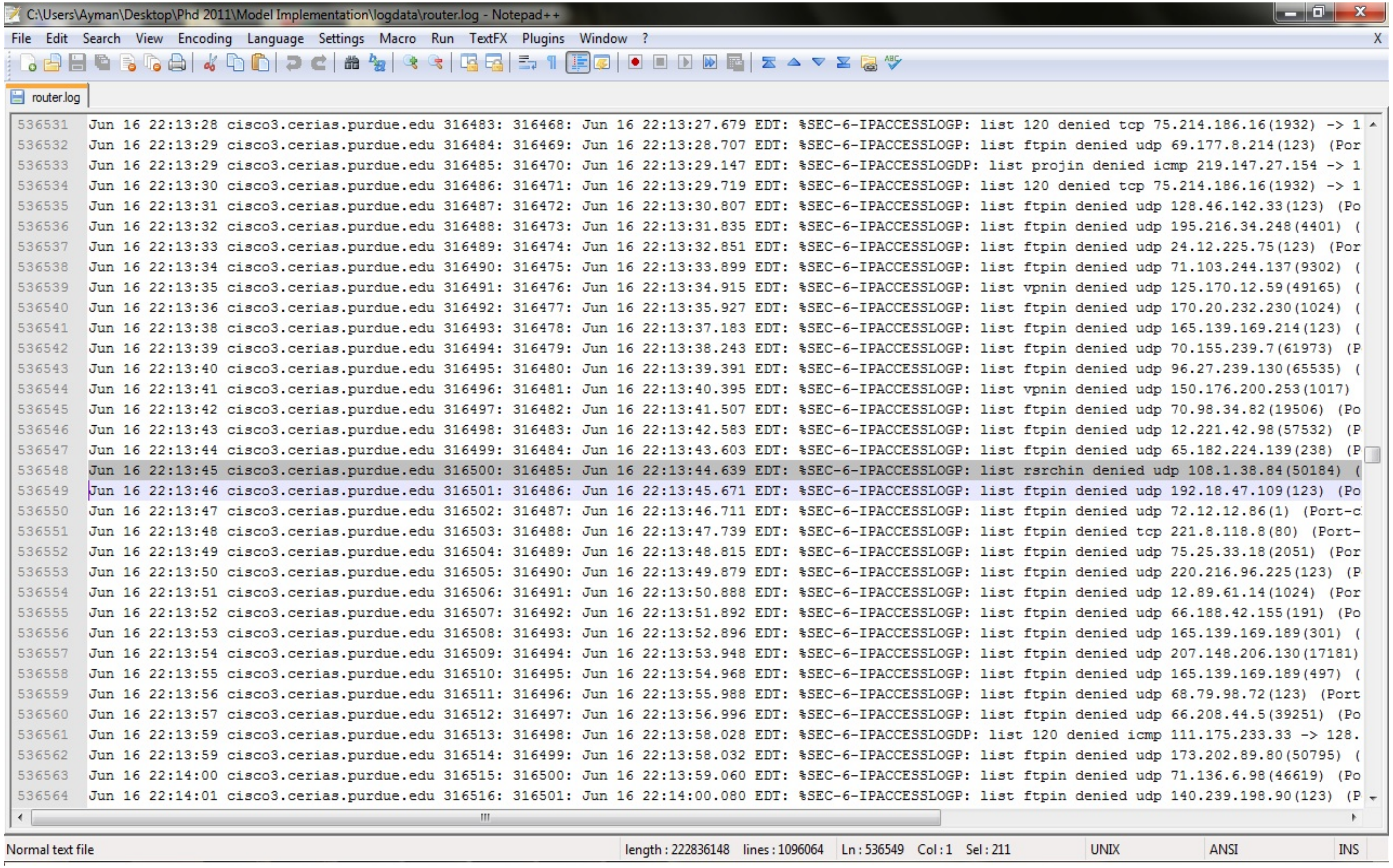
6.2 Future Work

This dissertation introduces several directions for future research including extending the model by implementing other agents for network security tools, system audit logs, and host based IDS; enhancing the learning capability with more accurate behavioral profiles for detecting coordinated attacks and multi-step attacks; and in preparing different scenarios to include those kinds of attacks and generating the datasets to be utilized in building the knowledge base for learning. Studying distributed wide area networks and worldwide correlation would improve the intrusion detection and early detection of new attacks.

Expanding the model to include automated responses would address the need for immediate responses to attacks; the automated response agent would depend on the correlated alert results and select the proper response from among the available network capabilities. While DACM depends on the knowledge base and network security policy, studying indoor risk analysis and security assurance appears as a critical point for future research. Finally, measuring the performance, trustworthiness, and assurance of distributed agents is a challenge to the problem of the probability of the existence of fake agents.

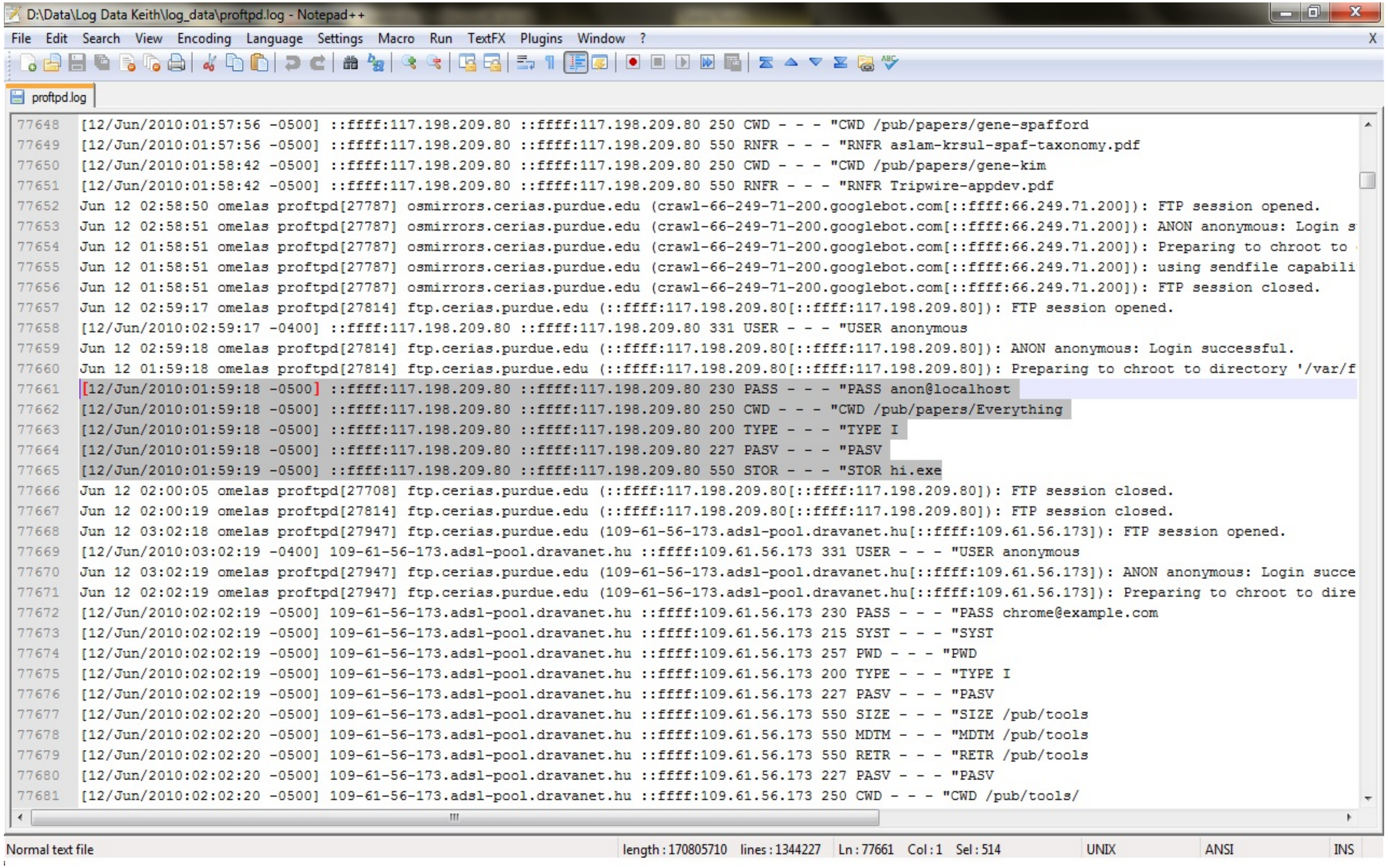
APPENDIX A
Larger Images of Results Figures

A-1 Larger Image of Figure 3.4 Firewall router log file



```
Jun 16 22:13:28 cisco3.cerias.purdue.edu 316483: 316468: Jun 16 22:13:27.679 EDT: %SEC-6-IPACCESSLOGP: list 120 denied tcp 75.214.186.16(1932) -> 1
Jun 16 22:13:29 cisco3.cerias.purdue.edu 316484: 316469: Jun 16 22:13:28.707 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 69.177.8.214(123) (Por
Jun 16 22:13:29 cisco3.cerias.purdue.edu 316485: 316470: Jun 16 22:13:29.147 EDT: %SEC-6-IPACCESSLOGDP: list projin denied icmp 219.147.27.154 -> 1
Jun 16 22:13:30 cisco3.cerias.purdue.edu 316486: 316471: Jun 16 22:13:29.719 EDT: %SEC-6-IPACCESSLOGP: list 120 denied tcp 75.214.186.16(1932) -> 1
Jun 16 22:13:31 cisco3.cerias.purdue.edu 316487: 316472: Jun 16 22:13:30.807 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 128.46.142.33(123) (Po
Jun 16 22:13:32 cisco3.cerias.purdue.edu 316488: 316473: Jun 16 22:13:31.835 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 195.216.34.248(4401) (
Jun 16 22:13:33 cisco3.cerias.purdue.edu 316489: 316474: Jun 16 22:13:32.851 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 24.12.225.75(123) (Por
Jun 16 22:13:34 cisco3.cerias.purdue.edu 316490: 316475: Jun 16 22:13:33.899 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 71.103.244.137(9302) (
Jun 16 22:13:35 cisco3.cerias.purdue.edu 316491: 316476: Jun 16 22:13:34.915 EDT: %SEC-6-IPACCESSLOGP: list vpnin denied udp 125.170.12.59(49165) (
Jun 16 22:13:36 cisco3.cerias.purdue.edu 316492: 316477: Jun 16 22:13:35.927 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 170.20.232.230(1024) (
Jun 16 22:13:38 cisco3.cerias.purdue.edu 316493: 316478: Jun 16 22:13:37.183 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.214(123) (
Jun 16 22:13:39 cisco3.cerias.purdue.edu 316494: 316479: Jun 16 22:13:38.243 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 70.155.239.7(61973) (P
Jun 16 22:13:40 cisco3.cerias.purdue.edu 316495: 316480: Jun 16 22:13:39.391 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 96.27.239.130(65535) (
Jun 16 22:13:41 cisco3.cerias.purdue.edu 316496: 316481: Jun 16 22:13:40.395 EDT: %SEC-6-IPACCESSLOGP: list vpnin denied udp 150.176.200.253(1017)
Jun 16 22:13:42 cisco3.cerias.purdue.edu 316497: 316482: Jun 16 22:13:41.507 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 70.98.34.82(19506) (Po
Jun 16 22:13:43 cisco3.cerias.purdue.edu 316498: 316483: Jun 16 22:13:42.583 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 12.221.42.98(57532) (P
Jun 16 22:13:44 cisco3.cerias.purdue.edu 316499: 316484: Jun 16 22:13:43.603 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 65.182.224.139(238) (P
Jun 16 22:13:45 cisco3.cerias.purdue.edu 316500: 316485: Jun 16 22:13:44.639 EDT: %SEC-6-IPACCESSLOGP: list rsrchin denied udp 108.1.38.84(50184) (
Jun 16 22:13:46 cisco3.cerias.purdue.edu 316501: 316486: Jun 16 22:13:45.671 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 192.18.47.109(123) (Po
Jun 16 22:13:47 cisco3.cerias.purdue.edu 316502: 316487: Jun 16 22:13:46.711 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 72.12.12.86(1) (Port-c
Jun 16 22:13:48 cisco3.cerias.purdue.edu 316503: 316488: Jun 16 22:13:47.739 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied tcp 221.8.118.8(80) (Port-
Jun 16 22:13:49 cisco3.cerias.purdue.edu 316504: 316489: Jun 16 22:13:48.815 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 75.25.33.18(2051) (Por
Jun 16 22:13:50 cisco3.cerias.purdue.edu 316505: 316490: Jun 16 22:13:49.879 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 220.216.96.225(123) (P
Jun 16 22:13:51 cisco3.cerias.purdue.edu 316506: 316491: Jun 16 22:13:50.888 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 12.89.61.14(1024) (Por
Jun 16 22:13:52 cisco3.cerias.purdue.edu 316507: 316492: Jun 16 22:13:51.892 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 66.188.42.155(191) (Po
Jun 16 22:13:53 cisco3.cerias.purdue.edu 316508: 316493: Jun 16 22:13:52.896 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.189(301) (
Jun 16 22:13:54 cisco3.cerias.purdue.edu 316509: 316494: Jun 16 22:13:53.948 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 207.148.206.130(17181)
Jun 16 22:13:55 cisco3.cerias.purdue.edu 316510: 316495: Jun 16 22:13:54.968 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 165.139.169.189(497) (
Jun 16 22:13:56 cisco3.cerias.purdue.edu 316511: 316496: Jun 16 22:13:55.988 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 68.79.98.72(123) (Port
Jun 16 22:13:57 cisco3.cerias.purdue.edu 316512: 316497: Jun 16 22:13:56.996 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 66.208.44.5(39251) (Po
Jun 16 22:13:59 cisco3.cerias.purdue.edu 316513: 316498: Jun 16 22:13:58.028 EDT: %SEC-6-IPACCESSLOGDP: list 120 denied icmp 111.175.233.33 -> 128.
Jun 16 22:13:59 cisco3.cerias.purdue.edu 316514: 316499: Jun 16 22:13:58.032 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 173.202.89.80(50795) (
Jun 16 22:14:00 cisco3.cerias.purdue.edu 316515: 316500: Jun 16 22:13:59.060 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 71.136.6.98(46619) (Po
Jun 16 22:14:01 cisco3.cerias.purdue.edu 316516: 316501: Jun 16 22:14:00.080 EDT: %SEC-6-IPACCESSLOGP: list ftpin denied udp 140.239.198.90(123) (P
```

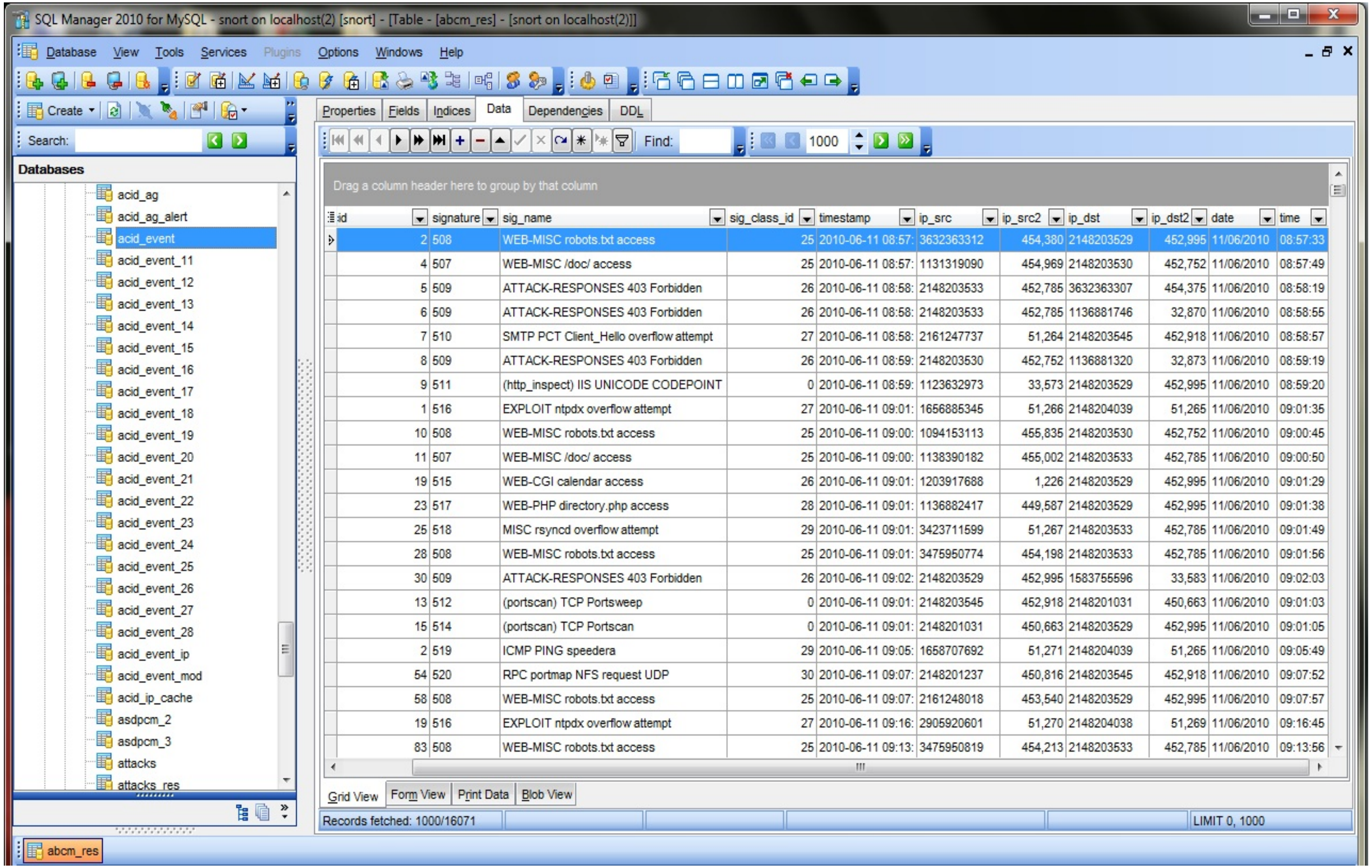
A-2 Larger Image of Figure 3.6 FTP Log File


D:\Data\Log Data Keith\log_data\proftpd.log - Notepad++
File Edit Search View Encoding Language Settings Macro Run TextFX Plugins Window ?
X
proftpd.log

```
[12/Jun/2010:01:57:56 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/gene-spafford
[12/Jun/2010:01:57:56 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 RNFR - - - "RNFR aslam-krsul-spaf-taxonomy.pdf
[12/Jun/2010:01:58:42 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/gene-kim
[12/Jun/2010:01:58:42 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 RNFR - - - "RNFR Tripwire-appdev.pdf
Jun 12 02:58:50 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): FTP session opened.
Jun 12 02:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): ANON anonymous: Login s
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): Preparing to chroot to
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): using sendfile capabili
Jun 12 01:58:51 omelas proftpd[27787] osmirrors.cerias.purdue.edu (crawl-66-249-71-200.googlebot.com[::ffff:66.249.71.200]): FTP session closed.
Jun 12 02:59:17 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session opened.
[12/Jun/2010:02:59:17 -0400] ::ffff:117.198.209.80 ::ffff:117.198.209.80 331 USER - - - "USER anonymous
Jun 12 02:59:18 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): ANON anonymous: Login successful.
Jun 12 01:59:18 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): Preparing to chroot to directory '/var/f
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 230 PASS - - - "PASS anon@localhost
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 250 CWD - - - "CWD /pub/papers/Everything
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 200 TYPE - - - "TYPE I
[12/Jun/2010:01:59:18 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 227 PASV - - - "PASV
[12/Jun/2010:01:59:19 -0500] ::ffff:117.198.209.80 ::ffff:117.198.209.80 550 STOR - - - "STOR hi.exe
Jun 12 02:00:05 omelas proftpd[27708] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session closed.
Jun 12 02:00:19 omelas proftpd[27814] ftp.cerias.purdue.edu (::ffff:117.198.209.80[::ffff:117.198.209.80]): FTP session closed.
Jun 12 03:02:18 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): FTP session opened.
[12/Jun/2010:03:02:19 -0400] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 331 USER - - - "USER anonymous
Jun 12 03:02:19 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): ANON anonymous: Login succe
Jun 12 02:02:19 omelas proftpd[27947] ftp.cerias.purdue.edu (109-61-56-173.adsl-pool.dravanet.hu[::ffff:109.61.56.173]): Preparing to chroot to dire
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 230 PASS - - - "PASS chrome@example.com
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 215 SYST - - - "SYST
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 257 PWD - - - "PWD
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 200 TYPE - - - "TYPE I
[12/Jun/2010:02:02:19 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 227 PASV - - - "PASV
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 SIZE - - - "SIZE /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 MDTM - - - "MDTM /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 550 RETR - - - "RETR /pub/tools
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 227 PASV - - - "PASV
[12/Jun/2010:02:02:20 -0500] 109-61-56-173.adsl-pool.dravanet.hu ::ffff:109.61.56.173 250 CWD - - - "CWD /pub/tools/
```

Normal text file   length : 170805710   lines : 1344227   Ln : 77661   Col : 1   Sel : 514   UNIX   ANSI   INS

A-3 Larger Image of Figure 5.2 Snort IDS Alerts

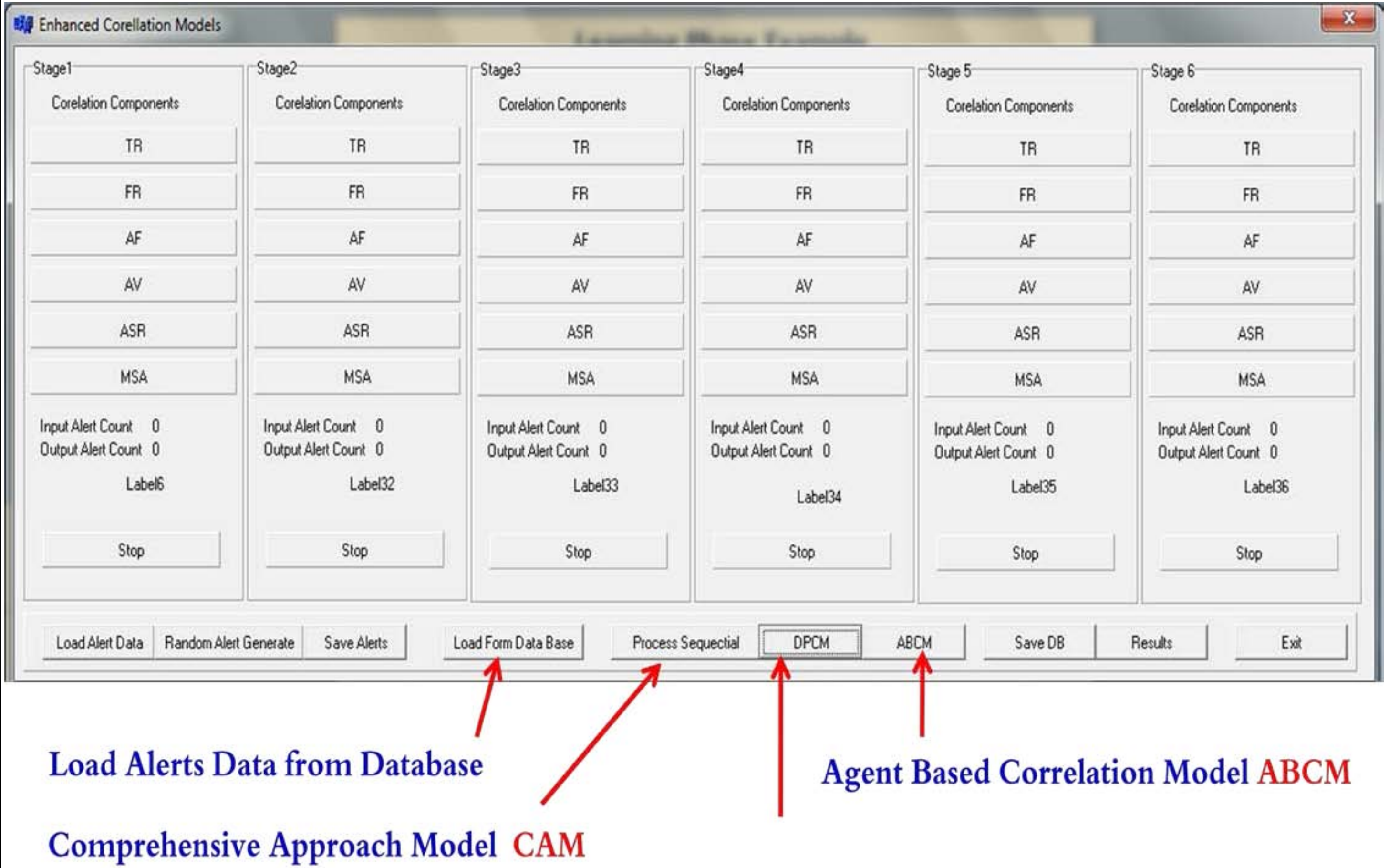


SQL Manager 2010 for MySQL - snort on localhost(2) [snort] - [Table - [abcm_res] - [snort on localhost(2)]]

id	signature	sig_name	sig_class_id	timestamp	ip_src	ip_src2	ip_dst	ip_dst2	date	time
2	508	WEB-MISC robots.txt access	25	2010-06-11 08:57:	3632363312	454,380	2148203529	452,995	11/06/2010	08:57:33
4	507	WEB-MISC /doc/ access	25	2010-06-11 08:57:	1131319090	454,969	2148203530	452,752	11/06/2010	08:57:49
5	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:58:	2148203533	452,785	3632363307	454,375	11/06/2010	08:58:19
6	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:58:	2148203533	452,785	1136881746	32,870	11/06/2010	08:58:55
7	510	SMTP PCT Client_Hello overflow attempt	27	2010-06-11 08:58:	2161247737	51,264	2148203545	452,918	11/06/2010	08:58:57
8	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 08:59:	2148203530	452,752	1136881320	32,873	11/06/2010	08:59:19
9	511	(http_inspect) IIS UNICODE CODEPOINT	0	2010-06-11 08:59:	1123632973	33,573	2148203529	452,995	11/06/2010	08:59:20
1	516	EXPLOIT ntpdx overflow attempt	27	2010-06-11 09:01:	1656885345	51,266	2148204039	51,265	11/06/2010	09:01:35
10	508	WEB-MISC robots.txt access	25	2010-06-11 09:00:	1094153113	455,835	2148203530	452,752	11/06/2010	09:00:45
11	507	WEB-MISC /doc/ access	25	2010-06-11 09:00:	1138390182	455,002	2148203533	452,785	11/06/2010	09:00:50
19	515	WEB-CGI calendar access	26	2010-06-11 09:01:	1203917688	1,226	2148203529	452,995	11/06/2010	09:01:29
23	517	WEB-PHP directory.php access	28	2010-06-11 09:01:	1136882417	449,587	2148203529	452,995	11/06/2010	09:01:38
25	518	MISC rsyncd overflow attempt	29	2010-06-11 09:01:	3423711599	51,267	2148203533	452,785	11/06/2010	09:01:49
28	508	WEB-MISC robots.txt access	25	2010-06-11 09:01:	3475950774	454,198	2148203533	452,785	11/06/2010	09:01:56
30	509	ATTACK-RESPONSES 403 Forbidden	26	2010-06-11 09:02:	2148203529	452,995	1583755596	33,583	11/06/2010	09:02:03
13	512	(portscan) TCP Portsweep	0	2010-06-11 09:01:	2148203545	452,918	2148201031	450,663	11/06/2010	09:01:03
15	514	(portscan) TCP Portscan	0	2010-06-11 09:01:	2148201031	450,663	2148203529	452,995	11/06/2010	09:01:05
2	519	ICMP PING speedera	29	2010-06-11 09:05:	1658707692	51,271	2148204039	51,265	11/06/2010	09:05:49
54	520	RPC portmap NFS request UDP	30	2010-06-11 09:07:	2148201237	450,816	2148203545	452,918	11/06/2010	09:07:52
58	508	WEB-MISC robots.txt access	25	2010-06-11 09:07:	2161248018	453,540	2148203529	452,995	11/06/2010	09:07:57
19	516	EXPLOIT ntpdx overflow attempt	27	2010-06-11 09:16:	2905920601	51,270	2148204038	51,269	11/06/2010	09:16:45
83	508	WEB-MISC robots.txt access	25	2010-06-11 09:13:	3475950819	454,213	2148203533	452,785	11/06/2010	09:13:56

Records fetched: 1000/16071

LIMIT 0, 1000

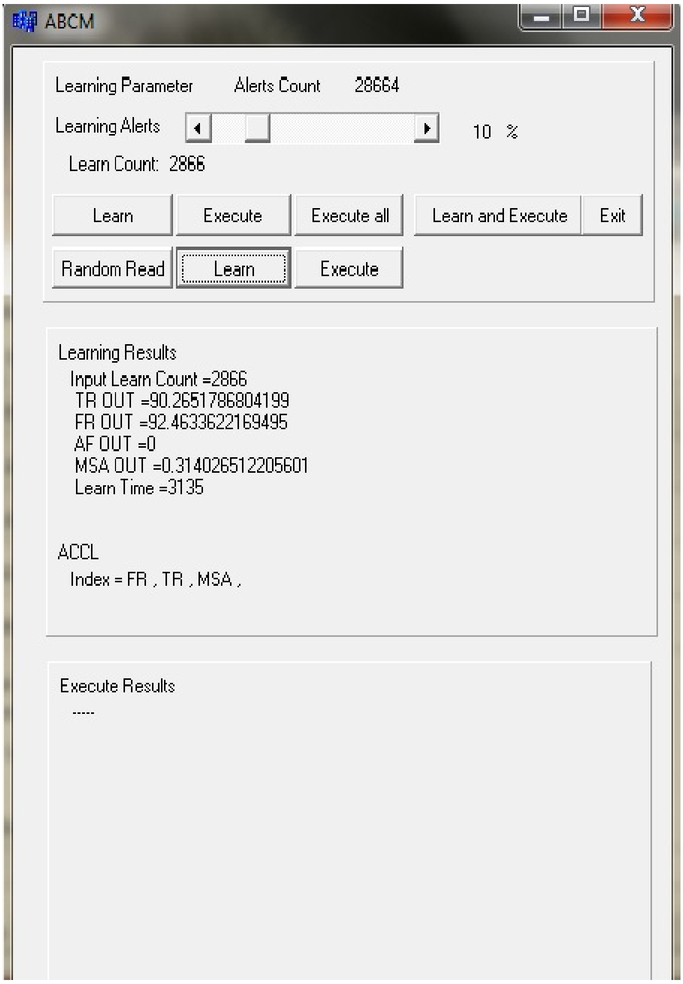


Load Alerts Data from Database

Agent Based Correlation Model ABCM

Comprehensive Approach Model CAM

A-4 Larger Image of Figure 5.3 IDS Alerts Correlation Interface



Learning Parameter Alerts Count 28664

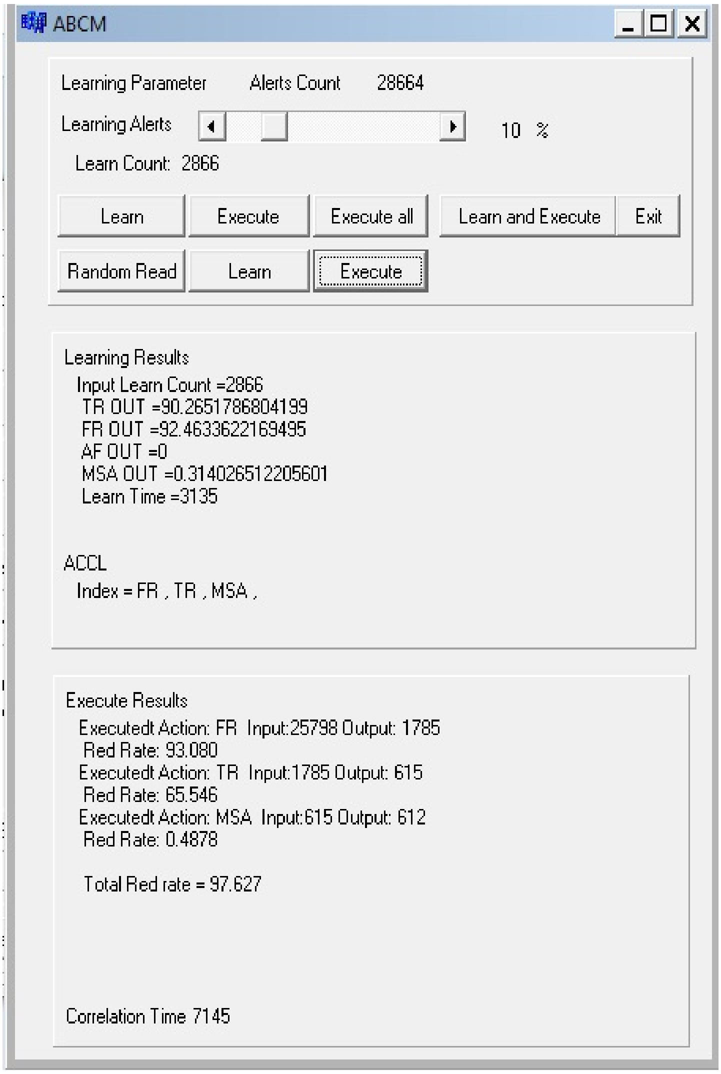
Learning Alerts 10 %

Learn Count: 2866

```
Learning Results
  Input Learn Count =2866
   TR OUT =90.2651786804199
   FR OUT =92.4633622169495
   AF OUT =0
   MSA OUT =0.314026512205601
   Learn Time =3135


ACCL
  Index = FR , TR , MSA ,
```

```
Execute Results
  -----
```

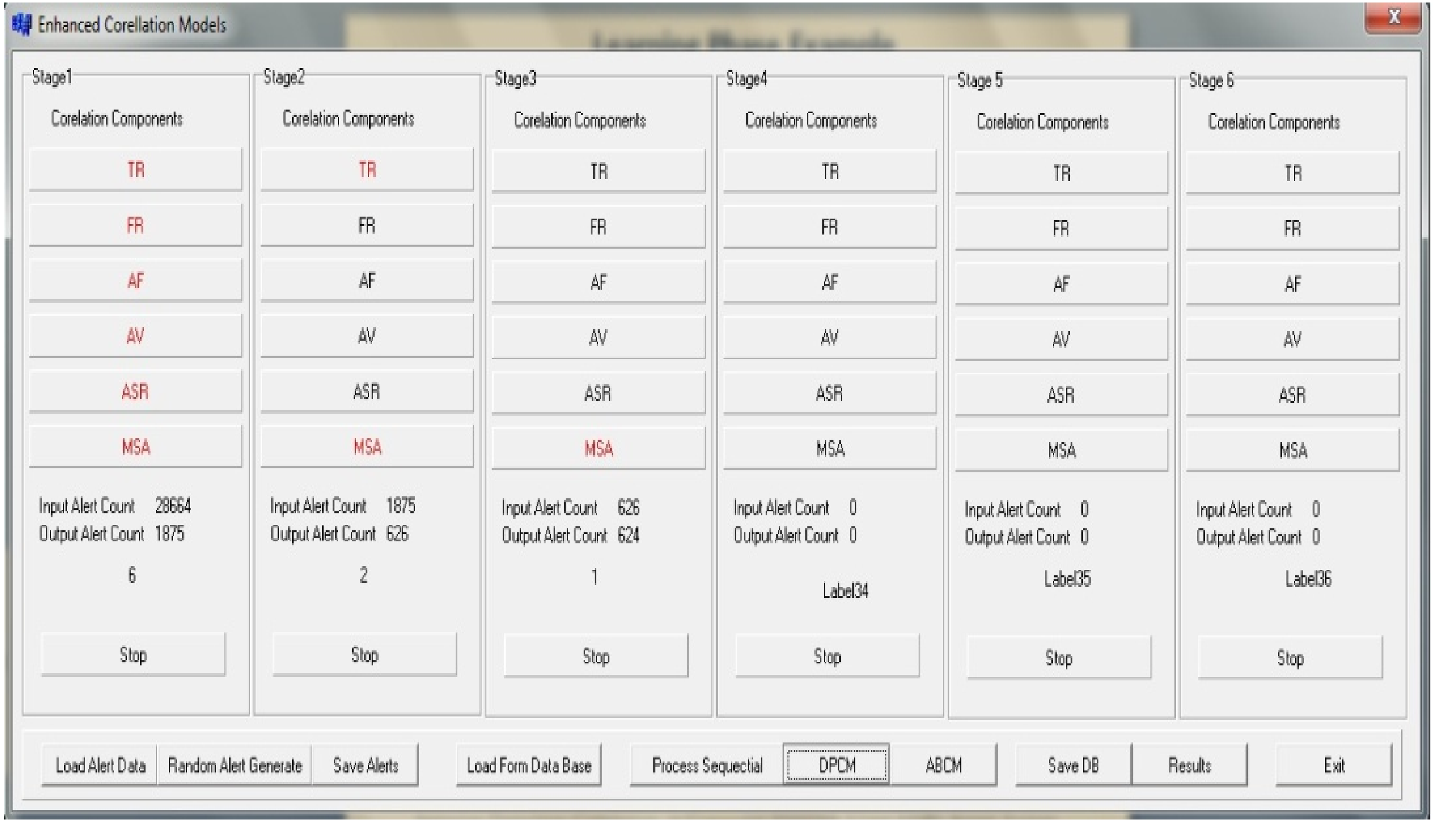
A- 5 Larger Image of Figure 5.7 ABCM Learning Phase Result



```
Learning Parameter      Alerts Count     28664
Learning Alerts                              10  %
  Learn Count:  2866

Learning Results
  Input Learn Count =2866
   TR OUT =90.2651786804199
   FR OUT =92.4633622169495
   AF OUT =0
   MSA OUT =0.314026512205601
   Learn Time =3135


ACCL
  Index = FR , TR , MSA ,

Execute Results
  Executedt Action: FR  Input:25798 Output: 1785
   Red Rate: 93.080
  Executedt Action: TR  Input:1785 Output: 615
   Red Rate: 65.546
  Executedt Action: MSA  Input:615 Output: 612
   Red Rate: 0.4878

   Total Red rate = 97.627



Correlation Time 7145
```

A-6 Larger Image of Figure 5.8 ABCM Correlation Phase Result

A-7 Larger Image of Figure 5.9 DPCM Correlation Stages Result



Stage1
Input Alert Count 28664
Output Alert Count 1875
6

Stage2
Input Alert Count 1875
Output Alert Count 626
2

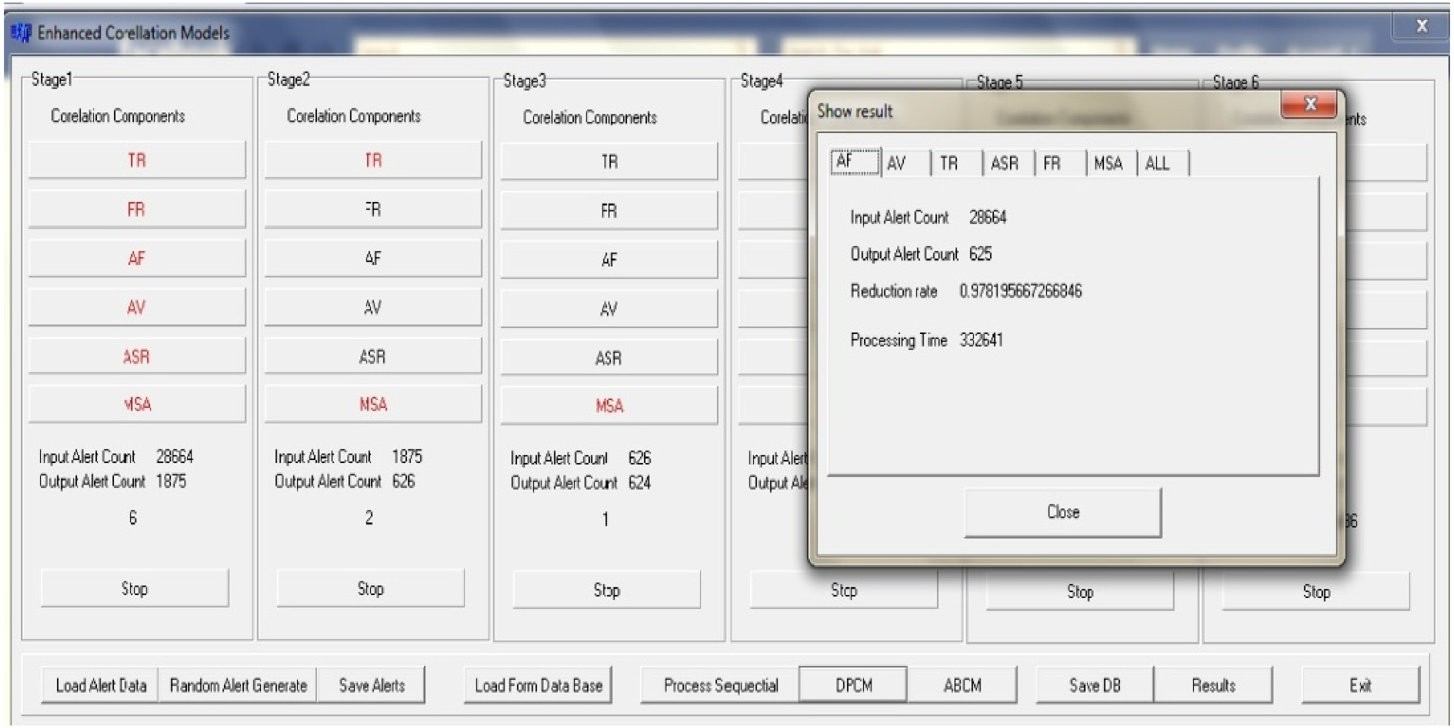
Stage3
Input Alert Count 626
Output Alert Count 624
1

Stage4
Input Alert Count 0
Output Alert Count 0
Label34

Stage 5
Input Alert Count 0
Output Alert Count 0
Label35

Stage 6
Input Alert Count 0
Output Alert Count 0
Label36

A-8 Larger Image of Figure 5.10 DPCM Final Correlation Result



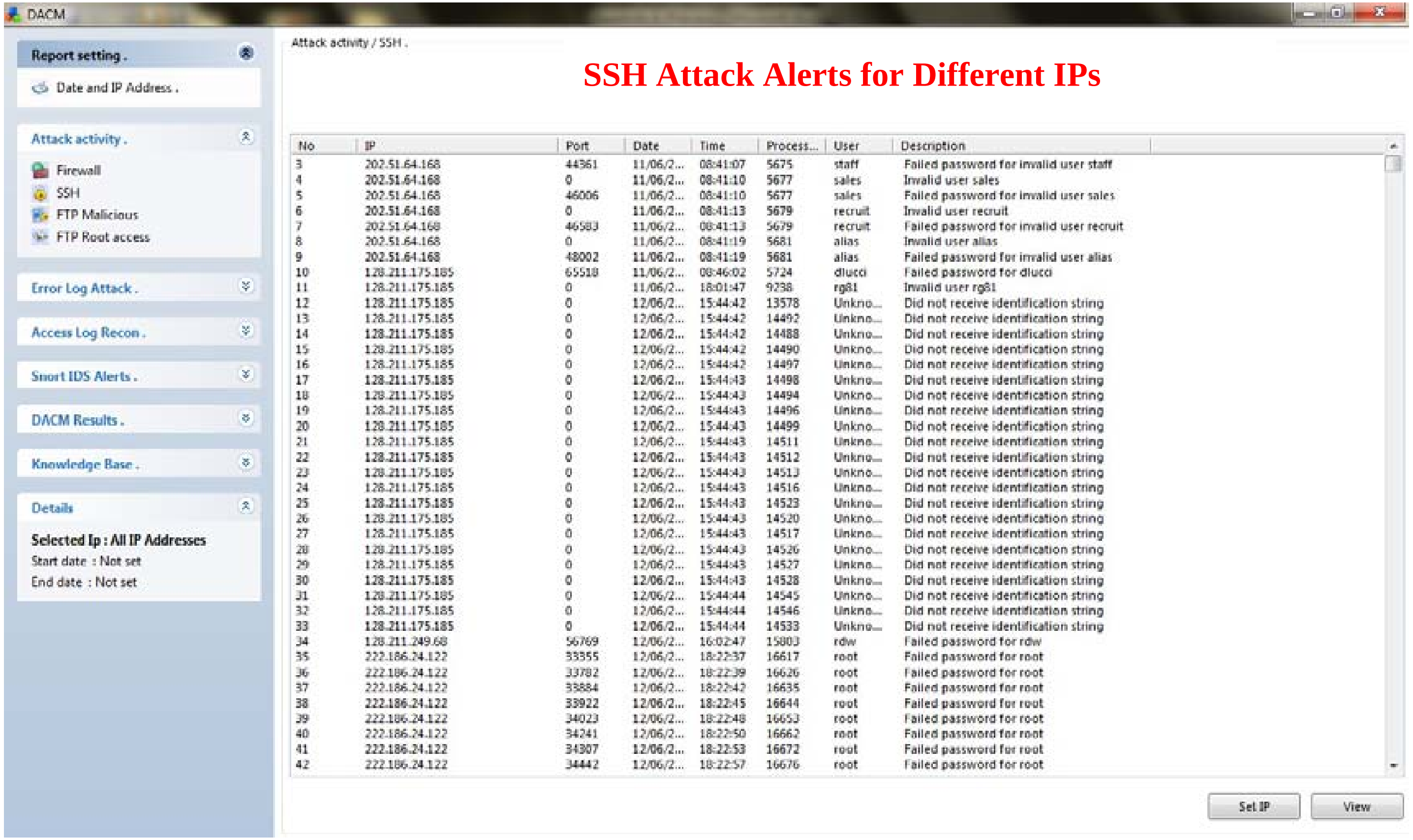
Stage1
Input Alert Count 28664
Output Alert Count 1875
6

Stage2
Input Alert Count 1875
Output Alert Count 626
2

Stage3
Input Alert Count 626
Output Alert Count 624
1

Show result
Input Alert Count 28664
Output Alert Count 625
Reduction rate 0.978195667266846
Processing Time 332641

A-9 Larger Image of Figure 5.14 SSH Agent Result

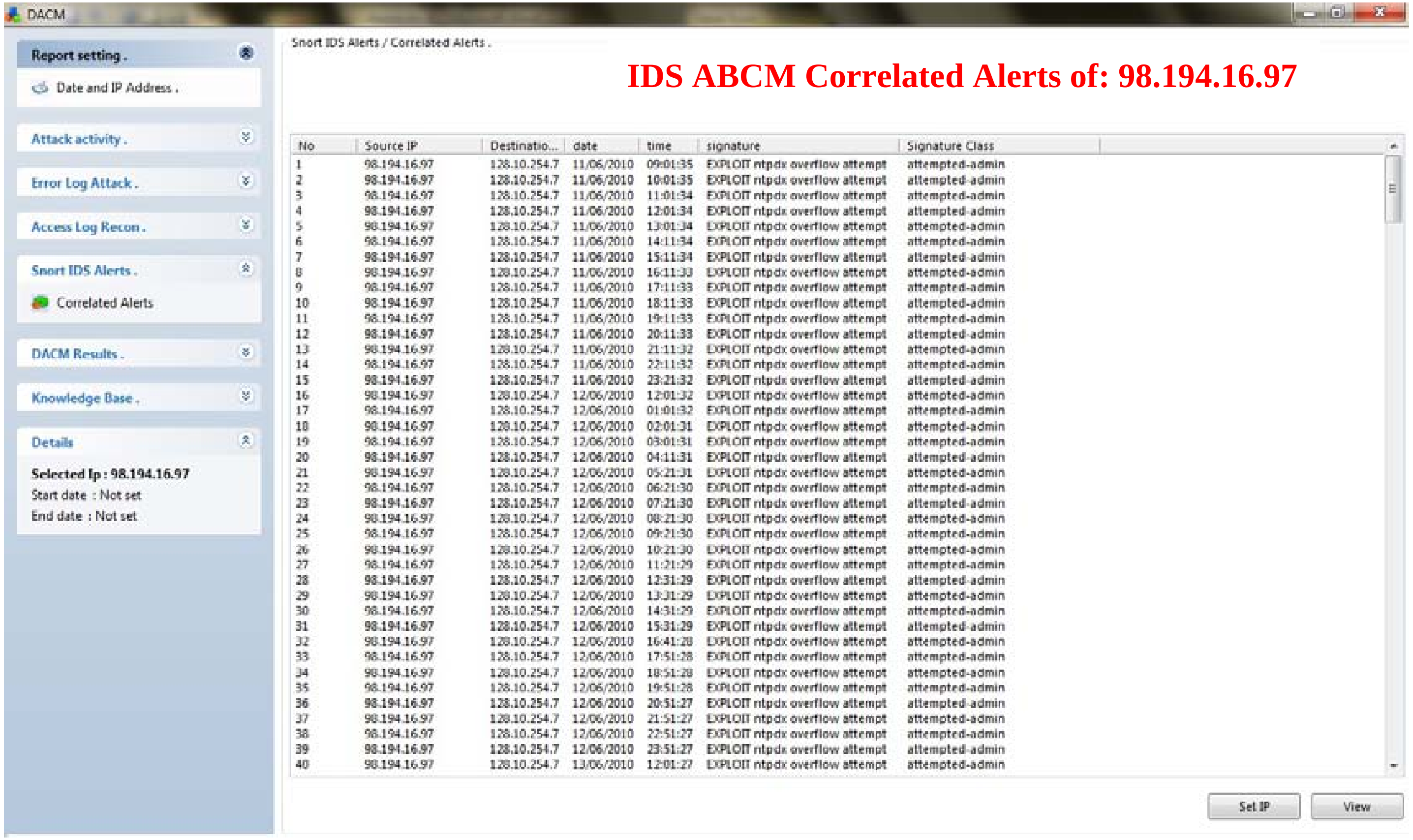


Selected Ip : All IP Addresses
Start date : Not set
End date : Not set

SSH Attack Alerts for Different IPs

No	IP	Port	Date	Time	Process...	User	Description
3	202.51.64.168	44361	11/06/2...	08:41:07	5675	staff	Failed password for invalid user staff
4	202.51.64.168	0	11/06/2...	08:41:10	5677	sales	Invalid user sales
5	202.51.64.168	46006	11/06/2...	08:41:10	5677	sales	Failed password for invalid user sales
6	202.51.64.168	0	11/06/2...	08:41:13	5679	recruit	Invalid user recruit
7	202.51.64.168	46583	11/06/2...	08:41:13	5679	recruit	Failed password for invalid user recruit
8	202.51.64.168	0	11/06/2...	08:41:19	5681	alias	Invalid user alias
9	202.51.64.168	48002	11/06/2...	08:41:19	5681	alias	Failed password for invalid user alias
10	128.211.175.185	65518	11/06/2...	08:46:02	5724	dlucci	Failed password for dlucci
11	128.211.175.185	0	11/06/2...	18:01:47	9238	rg81	Invalid user rg81
12	128.211.175.185	0	12/06/2...	15:44:42	13578	Unkno...	Did not receive identification string
13	128.211.175.185	0	12/06/2...	15:44:42	14492	Unkno...	Did not receive identification string
14	128.211.175.185	0	12/06/2...	15:44:42	14488	Unkno...	Did not receive identification string
15	128.211.175.185	0	12/06/2...	15:44:42	14490	Unkno...	Did not receive identification string
16	128.211.175.185	0	12/06/2...	15:44:42	14497	Unkno...	Did not receive identification string
17	128.211.175.185	0	12/06/2...	15:44:43	14498	Unkno...	Did not receive identification string
18	128.211.175.185	0	12/06/2...	15:44:43	14494	Unkno...	Did not receive identification string
19	128.211.175.185	0	12/06/2...	15:44:43	14496	Unkno...	Did not receive identification string
20	128.211.175.185	0	12/06/2...	15:44:43	14499	Unkno...	Did not receive identification string
21	128.211.175.185	0	12/06/2...	15:44:43	14511	Unkno...	Did not receive identification string
22	128.211.175.185	0	12/06/2...	15:44:43	14512	Unkno...	Did not receive identification string
23	128.211.175.185	0	12/06/2...	15:44:43	14513	Unkno...	Did not receive identification string
24	128.211.175.185	0	12/06/2...	15:44:43	14516	Unkno...	Did not receive identification string
25	128.211.175.185	0	12/06/2...	15:44:43	14523	Unkno...	Did not receive identification string
26	128.211.175.185	0	12/06/2...	15:44:43	14520	Unkno...	Did not receive identification string
27	128.211.175.185	0	12/06/2...	15:44:43	14517	Unkno...	Did not receive identification string
28	128.211.175.185	0	12/06/2...	15:44:43	14526	Unkno...	Did not receive identification string
29	128.211.175.185	0	12/06/2...	15:44:43	14527	Unkno...	Did not receive identification string
30	128.211.175.185	0	12/06/2...	15:44:43	14528	Unkno...	Did not receive identification string
31	128.211.175.185	0	12/06/2...	15:44:44	14545	Unkno...	Did not receive identification string
32	128.211.175.185	0	12/06/2...	15:44:44	14546	Unkno...	Did not receive identification string
33	128.211.175.185	0	12/06/2...	15:44:44	14533	Unkno...	Did not receive identification string
34	128.211.249.68	56769	12/06/2...	16:02:47	15803	rdw	Failed password for rdw
35	222.186.24.122	33355	12/06/2...	18:22:37	16617	root	Failed password for root
36	222.186.24.122	33782	12/06/2...	18:22:39	16626	root	Failed password for root
37	222.186.24.122	33884	12/06/2...	18:22:42	16635	root	Failed password for root
38	222.186.24.122	33922	12/06/2...	18:22:45	16644	root	Failed password for root
39	222.186.24.122	34023	12/06/2...	18:22:48	16653	root	Failed password for root
40	222.186.24.122	34241	12/06/2...	18:22:50	16662	root	Failed password for root
41	222.186.24.122	34307	12/06/2...	18:22:53	16672	root	Failed password for root
42	222.186.24.122	34442	12/06/2...	18:22:57	16676	root	Failed password for root

A-10 Larger Image of Figure 5.15 ABCM Result for Specific IP in DACM

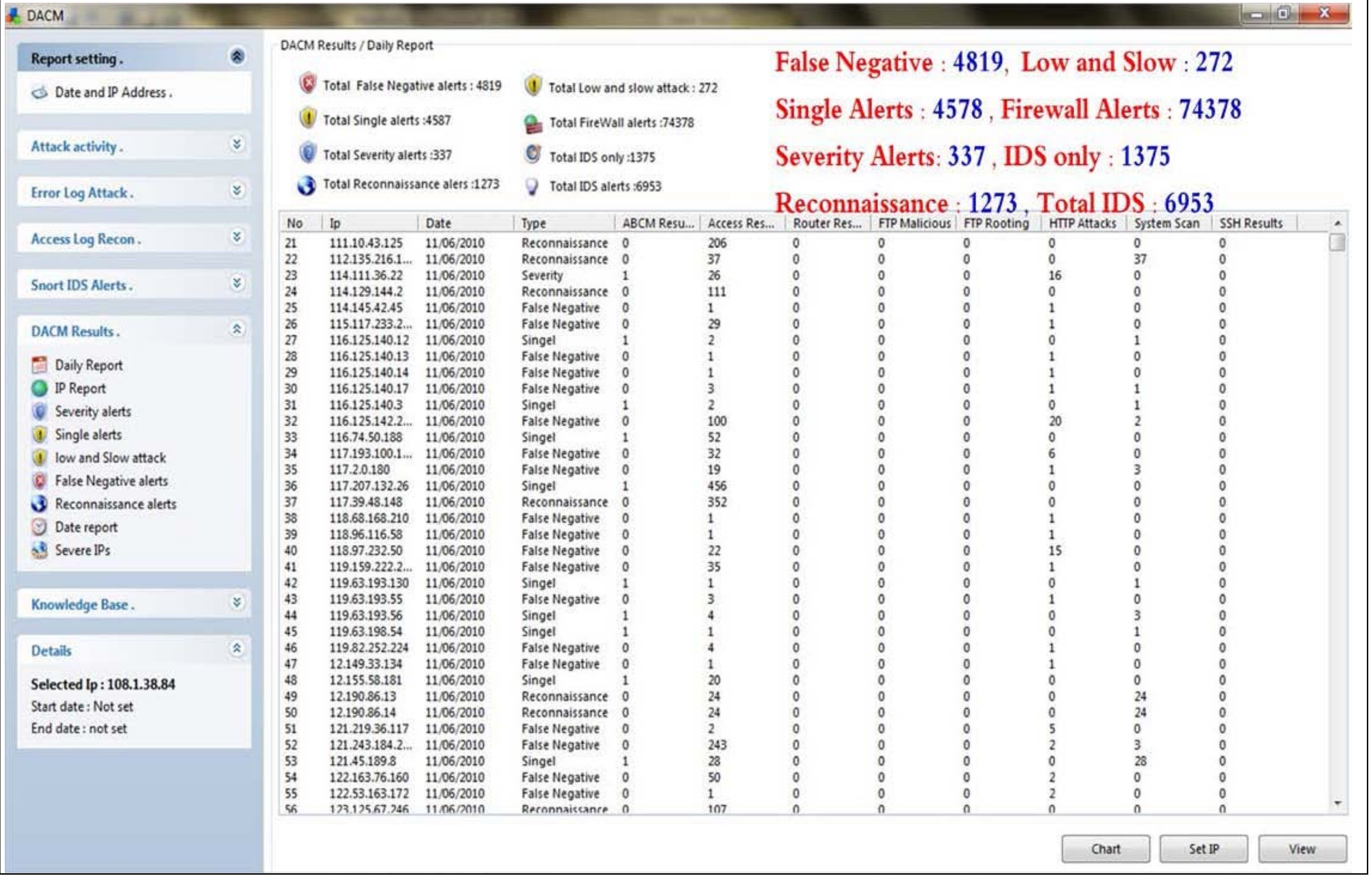


Selected Ip : 98.194.16.97
Start date : Not set
End date : Not set

Snort IDS Alerts / Correlated Alerts .

IDS ABCM Correlated Alerts of: 98.194.16.97

No	Source IP	Destinatio...	date	time	signature	Signature Class
1	98.194.16.97	128.10.254.7	11/06/2010	09:01:35	EXPLOIT ntpdx overflow attempt	attempted-admin
2	98.194.16.97	128.10.254.7	11/06/2010	10:01:35	EXPLOIT ntpdx overflow attempt	attempted-admin
3	98.194.16.97	128.10.254.7	11/06/2010	11:01:34	EXPLOIT ntpdx overflow attempt	attempted-admin
4	98.194.16.97	128.10.254.7	11/06/2010	12:01:34	EXPLOIT ntpdx overflow attempt	attempted-admin
5	98.194.16.97	128.10.254.7	11/06/2010	13:01:34	EXPLOIT ntpdx overflow attempt	attempted-admin
6	98.194.16.97	128.10.254.7	11/06/2010	14:11:34	EXPLOIT ntpdx overflow attempt	attempted-admin
7	98.194.16.97	128.10.254.7	11/06/2010	15:11:34	EXPLOIT ntpdx overflow attempt	attempted-admin
8	98.194.16.97	128.10.254.7	11/06/2010	16:11:33	EXPLOIT ntpdx overflow attempt	attempted-admin
9	98.194.16.97	128.10.254.7	11/06/2010	17:11:33	EXPLOIT ntpdx overflow attempt	attempted-admin
10	98.194.16.97	128.10.254.7	11/06/2010	18:11:33	EXPLOIT ntpdx overflow attempt	attempted-admin
11	98.194.16.97	128.10.254.7	11/06/2010	19:11:33	EXPLOIT ntpdx overflow attempt	attempted-admin
12	98.194.16.97	128.10.254.7	11/06/2010	20:11:33	EXPLOIT ntpdx overflow attempt	attempted-admin
13	98.194.16.97	128.10.254.7	11/06/2010	21:11:32	EXPLOIT ntpdx overflow attempt	attempted-admin
14	98.194.16.97	128.10.254.7	11/06/2010	22:11:32	EXPLOIT ntpdx overflow attempt	attempted-admin
15	98.194.16.97	128.10.254.7	11/06/2010	23:21:32	EXPLOIT ntpdx overflow attempt	attempted-admin
16	98.194.16.97	128.10.254.7	12/06/2010	12:01:32	EXPLOIT ntpdx overflow attempt	attempted-admin
17	98.194.16.97	128.10.254.7	12/06/2010	01:01:32	EXPLOIT ntpdx overflow attempt	attempted-admin
18	98.194.16.97	128.10.254.7	12/06/2010	02:01:31	EXPLOIT ntpdx overflow attempt	attempted-admin
19	98.194.16.97	128.10.254.7	12/06/2010	03:01:31	EXPLOIT ntpdx overflow attempt	attempted-admin
20	98.194.16.97	128.10.254.7	12/06/2010	04:11:31	EXPLOIT ntpdx overflow attempt	attempted-admin
21	98.194.16.97	128.10.254.7	12/06/2010	05:21:31	EXPLOIT ntpdx overflow attempt	attempted-admin
22	98.194.16.97	128.10.254.7	12/06/2010	06:21:30	EXPLOIT ntpdx overflow attempt	attempted-admin
23	98.194.16.97	128.10.254.7	12/06/2010	07:21:30	EXPLOIT ntpdx overflow attempt	attempted-admin
24	98.194.16.97	128.10.254.7	12/06/2010	08:21:30	EXPLOIT ntpdx overflow attempt	attempted-admin
25	98.194.16.97	128.10.254.7	12/06/2010	09:21:30	EXPLOIT ntpdx overflow attempt	attempted-admin
26	98.194.16.97	128.10.254.7	12/06/2010	10:21:30	EXPLOIT ntpdx overflow attempt	attempted-admin
27	98.194.16.97	128.10.254.7	12/06/2010	11:21:29	EXPLOIT ntpdx overflow attempt	attempted-admin
28	98.194.16.97	128.10.254.7	12/06/2010	12:31:29	EXPLOIT ntpdx overflow attempt	attempted-admin
29	98.194.16.97	128.10.254.7	12/06/2010	13:31:29	EXPLOIT ntpdx overflow attempt	attempted-admin
30	98.194.16.97	128.10.254.7	12/06/2010	14:31:29	EXPLOIT ntpdx overflow attempt	attempted-admin
31	98.194.16.97	128.10.254.7	12/06/2010	15:31:29	EXPLOIT ntpdx overflow attempt	attempted-admin
32	98.194.16.97	128.10.254.7	12/06/2010	16:41:28	EXPLOIT ntpdx overflow attempt	attempted-admin
33	98.194.16.97	128.10.254.7	12/06/2010	17:51:28	EXPLOIT ntpdx overflow attempt	attempted-admin
34	98.194.16.97	128.10.254.7	12/06/2010	18:51:28	EXPLOIT ntpdx overflow attempt	attempted-admin
35	98.194.16.97	128.10.254.7	12/06/2010	19:51:28	EXPLOIT ntpdx overflow attempt	attempted-admin
36	98.194.16.97	128.10.254.7	12/06/2010	20:51:27	EXPLOIT ntpdx overflow attempt	attempted-admin
37	98.194.16.97	128.10.254.7	12/06/2010	21:51:27	EXPLOIT ntpdx overflow attempt	attempted-admin
38	98.194.16.97	128.10.254.7	12/06/2010	22:51:27	EXPLOIT ntpdx overflow attempt	attempted-admin
39	98.194.16.97	128.10.254.7	12/06/2010	23:51:27	EXPLOIT ntpdx overflow attempt	attempted-admin
40	98.194.16.97	128.10.254.7	13/06/2010	12:01:27	EXPLOIT ntpdx overflow attempt	attempted-admin



Selected Ip : 108.1.38.84
Start date : Not set
End date : not set

DACM Results / Daily Report

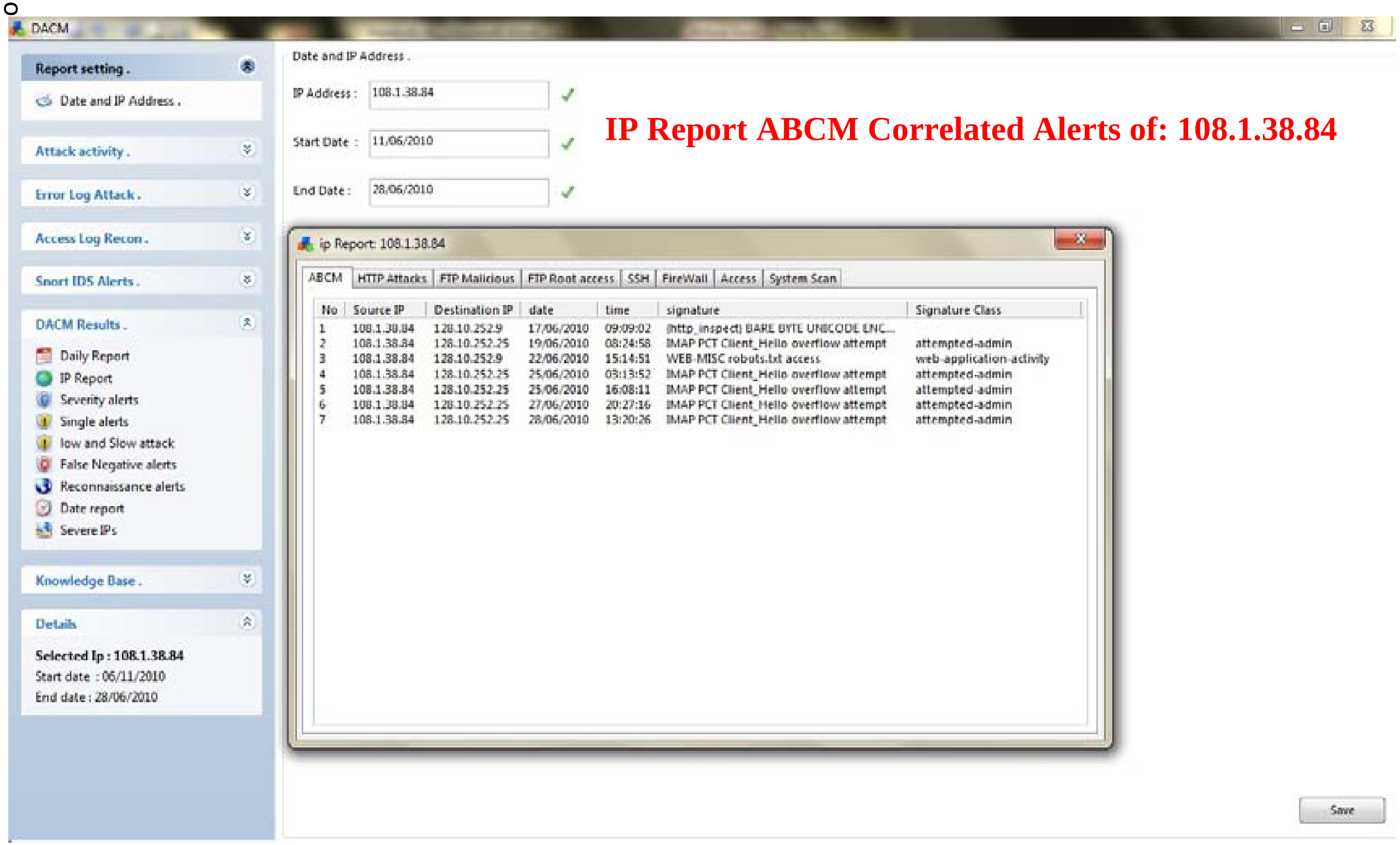
Total False Negative alerts : 4819
Total Low and slow attack : 272
Total Single alerts :4587
Total FireWall alerts :74378
Total Severity alerts :337
Total IDS only :1375
Total Reconnaissance alers :1273
Total IDS alerts :6953

False Negative : 4819, Low and Slow : 272
Single Alerts : 4578 , Firewall Alerts : 74378
Severity Alerts: 337 , IDS only : 1375
Reconnaissance : 1273 , Total IDS : 6953

No	Ip	Date	Type	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
21	111.10.43.125	11/06/2010	Reconnaissance	0	206	0	0	0	0	0	0
22	112.135.216.1...	11/06/2010	Reconnaissance	0	37	0	0	0	0	37	0
23	114.111.36.22	11/06/2010	Severity	1	26	0	0	0	16	0	0
24	114.129.144.2	11/06/2010	Reconnaissance	0	111	0	0	0	0	0	0
25	114.145.42.45	11/06/2010	False Negative	0	1	0	0	0	1	0	0
26	115.117.233.2...	11/06/2010	False Negative	0	29	0	0	0	1	0	0
27	116.125.140.12	11/06/2010	Singel	1	2	0	0	0	0	1	0
28	116.125.140.13	11/06/2010	False Negative	0	1	0	0	0	1	0	0
29	116.125.140.14	11/06/2010	False Negative	0	1	0	0	0	1	0	0
30	116.125.140.17	11/06/2010	False Negative	0	3	0	0	0	1	1	0
31	116.125.140.3	11/06/2010	Singel	1	2	0	0	0	0	1	0
32	116.125.142.2...	11/06/2010	False Negative	0	100	0	0	0	20	2	0
33	116.74.50.188	11/06/2010	Singel	1	52	0	0	0	0	0	0
34	117.193.100.1...	11/06/2010	False Negative	0	32	0	0	0	6	0	0
35	117.2.0.180	11/06/2010	False Negative	0	19	0	0	0	1	3	0
36	117.207.132.26	11/06/2010	Singel	1	456	0	0	0	0	0	0
37	117.39.48.148	11/06/2010	Reconnaissance	0	352	0	0	0	0	0	0
38	118.68.168.210	11/06/2010	False Negative	0	1	0	0	0	1	0	0
39	118.96.116.58	11/06/2010	False Negative	0	1	0	0	0	1	0	0
40	118.97.232.50	11/06/2010	False Negative	0	22	0	0	0	15	0	0
41	119.159.222.2...	11/06/2010	False Negative	0	35	0	0	0	1	0	0
42	119.63.193.130	11/06/2010	Singel	1	1	0	0	0	0	1	0
43	119.63.193.55	11/06/2010	False Negative	0	3	0	0	0	1	0	0
44	119.63.193.56	11/06/2010	Singel	1	4	0	0	0	0	3	0
45	119.63.198.54	11/06/2010	Singel	1	1	0	0	0	0	1	0
46	119.82.252.224	11/06/2010	False Negative	0	4	0	0	0	1	0	0
47	12.149.33.134	11/06/2010	False Negative	0	1	0	0	0	1	0	0
48	12.155.58.181	11/06/2010	Singel	1	20	0	0	0	0	0	0
49	12.190.86.13	11/06/2010	Reconnaissance	0	24	0	0	0	0	24	0
50	12.190.86.14	11/06/2010	Reconnaissance	0	24	0	0	0	0	24	0
51	121.219.36.117	11/06/2010	False Negative	0	2	0	0	0	5	0	0
52	121.243.184.2...	11/06/2010	False Negative	0	243	0	0	0	2	3	0
53	121.45.189.8	11/06/2010	Singel	1	28	0	0	0	0	28	0
54	122.163.76.160	11/06/2010	False Negative	0	50	0	0	0	2	0	0
55	122.53.163.172	11/06/2010	False Negative	0	1	0	0	0	2	0	0
56	123.125.67.246	11/06/2010	Reconnaissance	0	107	0	0	0	0	0	0

A-11 Larger Image of Figure 5.16 DACM Daily Report

A-12 Larger Image of Figure 5.17 DACM IP Report form ABCM IDS Correlated Alerts



IP Report ABCM Correlated Alerts of: 108.1.38.84

Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

Date and IP Address .

IP Address : 108.1.38.84 ✓

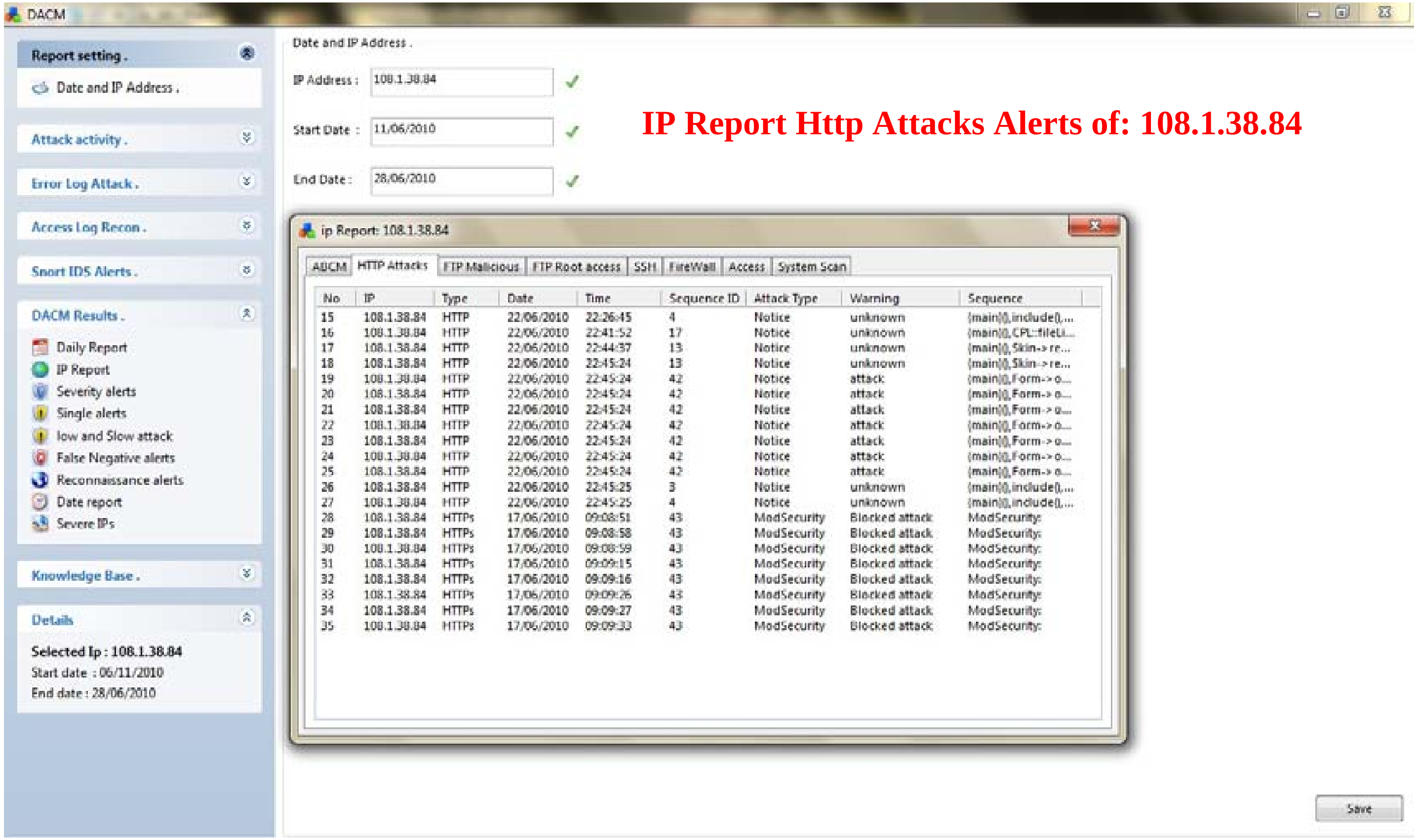
Start Date : 11/06/2010 ✓

End Date : 28/06/2010 ✓

ip Report: 108.1.38.84

No	Source IP	Destination IP	date	time	signature	Signature Class
1	108.1.38.84	128.10.252.9	17/06/2010	09:09:02	(http_inspect) BARE BYTE UNICODE ENC...	
2	108.1.38.84	128.10.252.25	19/06/2010	08:24:58	IMAP PCT Client_Hello overflow attempt	attempted-admin
3	108.1.38.84	128.10.252.9	22/06/2010	15:14:51	WEB-MISC robots.txt access	web-application-activity
4	108.1.38.84	128.10.252.25	25/06/2010	03:13:52	IMAP PCT Client_Hello overflow attempt	attempted-admin
5	108.1.38.84	128.10.252.25	25/06/2010	16:08:11	IMAP PCT Client_Hello overflow attempt	attempted-admin
6	108.1.38.84	128.10.252.25	27/06/2010	20:27:16	IMAP PCT Client_Hello overflow attempt	attempted-admin
7	108.1.38.84	128.10.252.25	28/06/2010	13:20:26	IMAP PCT Client_Hello overflow attempt	attempted-admin

A-13 Larger Image of Figure 5.18 DACM IP Report form HTTP Attack



IP Report Http Attacks Alerts of: 108.1.38.84

Date and IP Address .

IP Address : 108.1.38.84 ✓

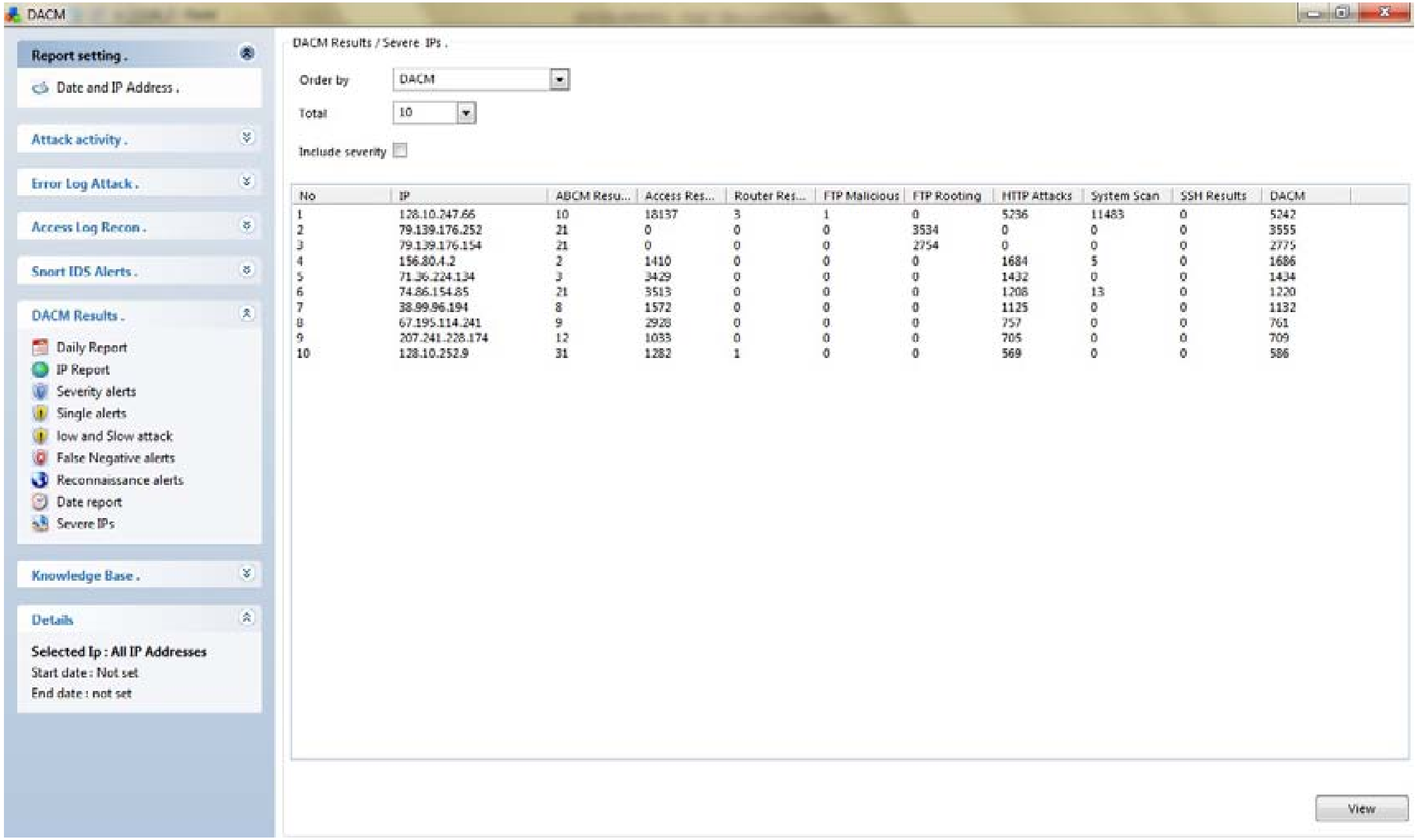
Start Date : 11/06/2010 ✓

End Date : 28/06/2010 ✓

No	IP	Type	Date	Time	Sequence ID	Attack Type	Warning	Sequence
15	108.1.38.84	HTTP	22/06/2010	22:26:45	4	Notice	unknown	{main}(),include(),...
16	108.1.38.84	HTTP	22/06/2010	22:41:52	17	Notice	unknown	{main}(),CPL::fileLi...
17	108.1.38.84	HTTP	22/06/2010	22:44:37	13	Notice	unknown	{main}(),Skin-> re...
18	108.1.38.84	HTTP	22/06/2010	22:45:24	13	Notice	unknown	{main}(),Skin-> re...
19	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
20	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
21	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
22	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
23	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
24	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
25	108.1.38.84	HTTP	22/06/2010	22:45:24	42	Notice	attack	{main}(),Form-> o...
26	108.1.38.84	HTTP	22/06/2010	22:45:25	3	Notice	unknown	{main}(),include(),...
27	108.1.38.84	HTTP	22/06/2010	22:45:25	4	Notice	unknown	{main}(),include(),...
28	108.1.38.84	HTTPs	17/06/2010	09:08:51	43	ModSecurity	Blocked attack	ModSecurity:
29	108.1.38.84	HTTPs	17/06/2010	09:08:58	43	ModSecurity	Blocked attack	ModSecurity:
30	108.1.38.84	HTTPs	17/06/2010	09:08:59	43	ModSecurity	Blocked attack	ModSecurity:
31	108.1.38.84	HTTPs	17/06/2010	09:09:15	43	ModSecurity	Blocked attack	ModSecurity:
32	108.1.38.84	HTTPs	17/06/2010	09:09:16	43	ModSecurity	Blocked attack	ModSecurity:
33	108.1.38.84	HTTPs	17/06/2010	09:09:26	43	ModSecurity	Blocked attack	ModSecurity:
34	108.1.38.84	HTTPs	17/06/2010	09:09:27	43	ModSecurity	Blocked attack	ModSecurity:
35	108.1.38.84	HTTPs	17/06/2010	09:09:33	43	ModSecurity	Blocked attack	ModSecurity:

Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

A-14 Larger Image of Figure 5.19 DACM Maximum Priority Report



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Severe IPs .

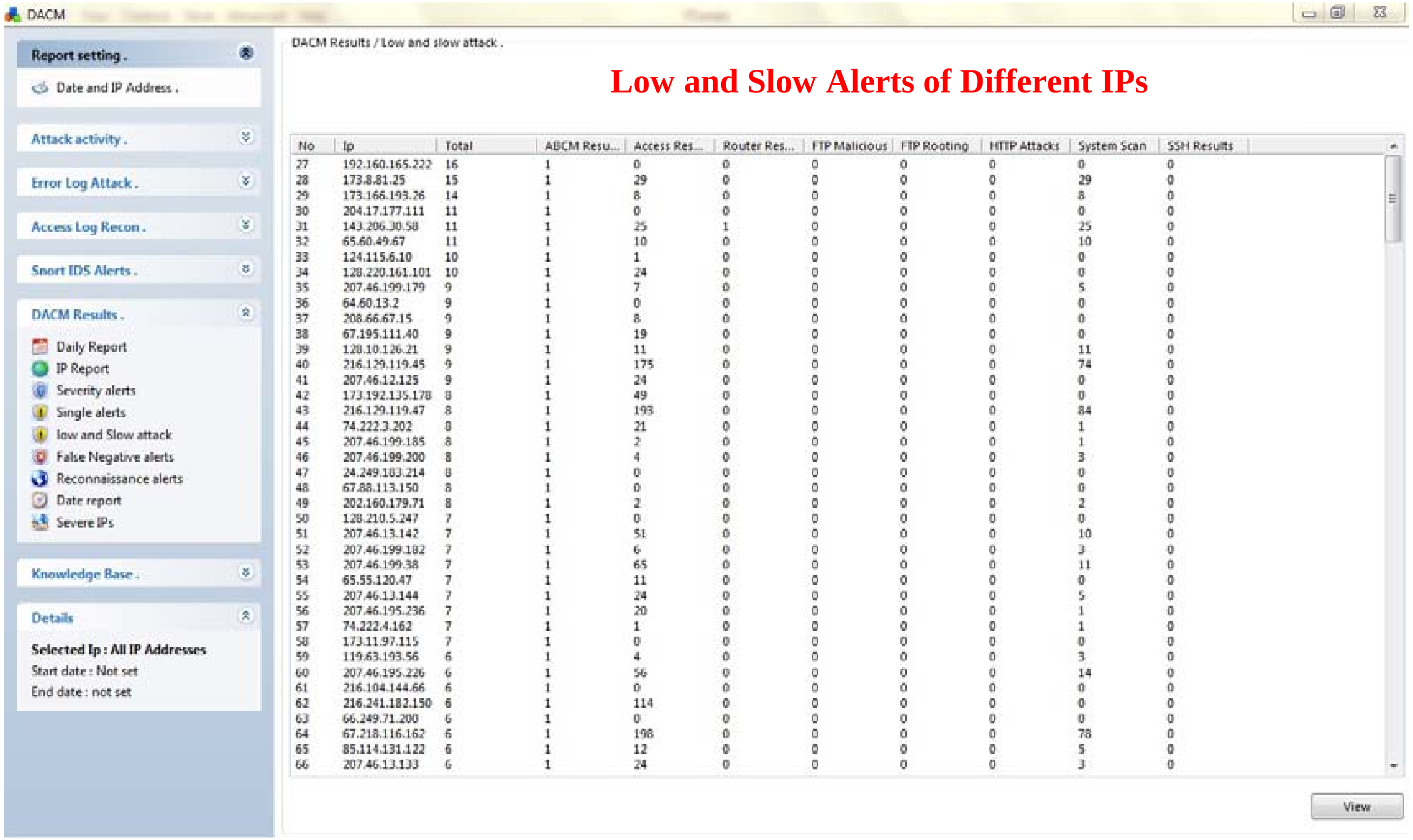
Order by DACM

Total 10

Include severity □

No	IP	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results	DACM
1	128.10.247.66	10	18137	3	1	0	5236	11483	0	5242
2	79.139.176.252	21	0	0	0	3534	0	0	0	3555
3	79.139.176.154	21	0	0	0	2754	0	0	0	2775
4	156.80.4.2	2	1410	0	0	0	1684	5	0	1686
5	71.36.224.134	3	3429	0	0	0	1432	0	0	1434
6	74.86.154.85	21	3513	0	0	0	1208	13	0	1220
7	38.99.96.194	8	1572	0	0	0	1125	0	0	1132
8	67.195.114.241	9	2928	0	0	0	757	0	0	761
9	207.241.228.174	12	1033	0	0	0	705	0	0	709
10	128.10.252.9	31	1282	1	0	0	569	0	0	586

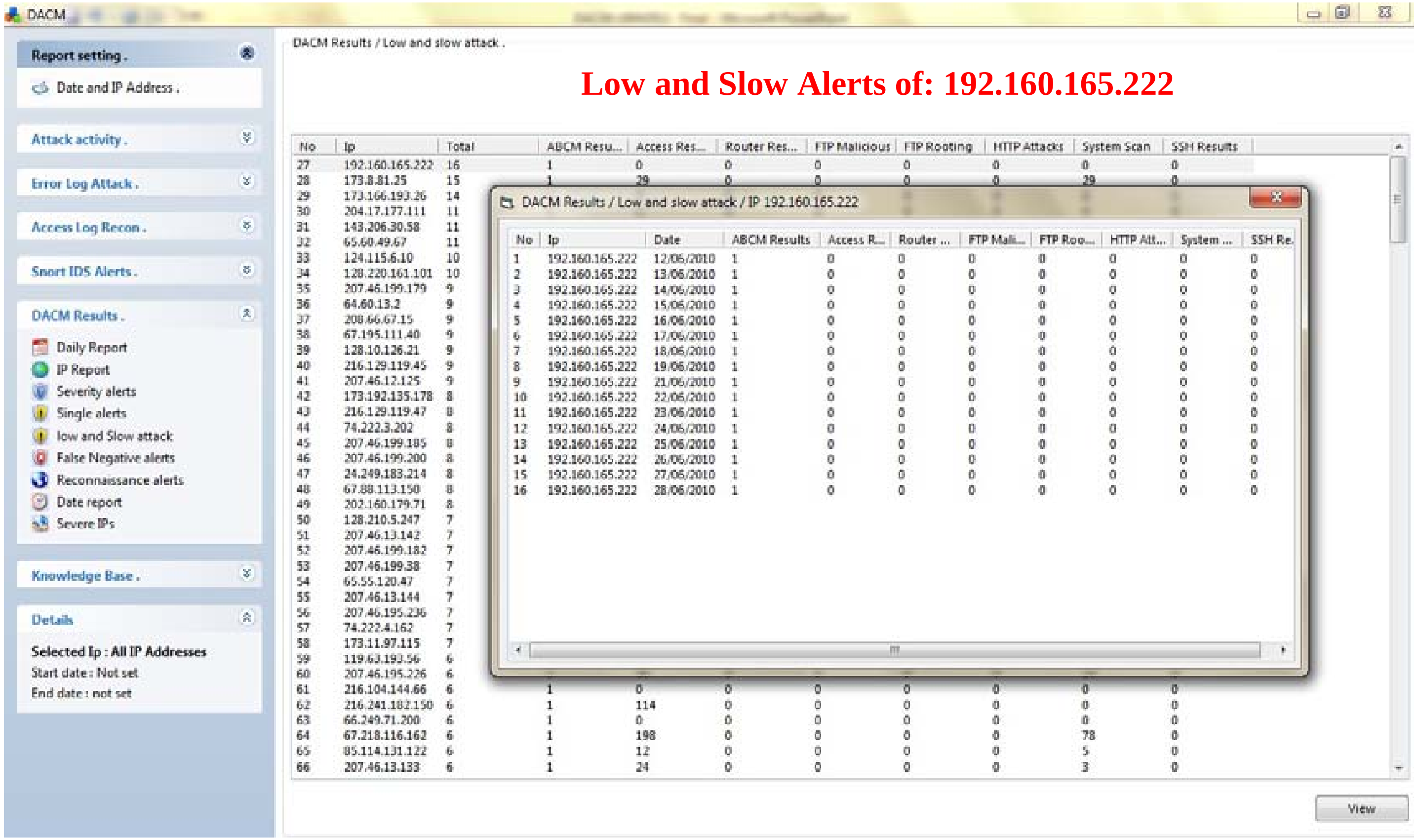
A-15 Larger Image of Figure 5.20 Low and Slow Attacks



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

Low and Slow Alerts of Different IPs

No	Ip	Total	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
27	192.160.165.222	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14	1	8	0	0	0	0	8	0
30	204.17.177.111	11	1	0	0	0	0	0	0	0
31	143.206.30.58	11	1	25	1	0	0	0	25	0
32	65.60.49.67	11	1	10	0	0	0	0	10	0
33	124.115.6.10	10	1	1	0	0	0	0	0	0
34	128.220.161.101	10	1	24	0	0	0	0	0	0
35	207.46.199.179	9	1	7	0	0	0	0	5	0
36	64.60.13.2	9	1	0	0	0	0	0	0	0
37	208.66.67.15	9	1	8	0	0	0	0	0	0
38	67.195.111.40	9	1	19	0	0	0	0	0	0
39	128.10.126.21	9	1	11	0	0	0	0	11	0
40	216.129.119.45	9	1	175	0	0	0	0	74	0
41	207.46.12.125	9	1	24	0	0	0	0	0	0
42	173.192.135.178	8	1	49	0	0	0	0	0	0
43	216.129.119.47	8	1	193	0	0	0	0	84	0
44	74.222.3.202	8	1	21	0	0	0	0	1	0
45	207.46.199.185	8	1	2	0	0	0	0	1	0
46	207.46.199.200	8	1	4	0	0	0	0	3	0
47	24.249.183.214	8	1	0	0	0	0	0	0	0
48	67.88.113.150	8	1	0	0	0	0	0	0	0
49	202.160.179.71	8	1	2	0	0	0	0	2	0
50	128.210.5.247	7	1	0	0	0	0	0	0	0
51	207.46.13.142	7	1	51	0	0	0	0	10	0
52	207.46.199.182	7	1	6	0	0	0	0	3	0
53	207.46.199.38	7	1	65	0	0	0	0	11	0
54	65.55.120.47	7	1	11	0	0	0	0	0	0
55	207.46.13.144	7	1	24	0	0	0	0	5	0
56	207.46.195.236	7	1	20	0	0	0	0	1	0
57	74.222.4.162	7	1	1	0	0	0	0	1	0
58	173.11.97.115	7	1	0	0	0	0	0	0	0
59	119.63.193.56	6	1	4	0	0	0	0	3	0
60	207.46.195.226	6	1	56	0	0	0	0	14	0
61	216.104.144.66	6	1	0	0	0	0	0	0	0
62	216.241.182.150	6	1	114	0	0	0	0	0	0
63	66.249.71.200	6	1	0	0	0	0	0	0	0
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	85.114.131.122	6	1	12	0	0	0	0	5	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Low and slow attack .

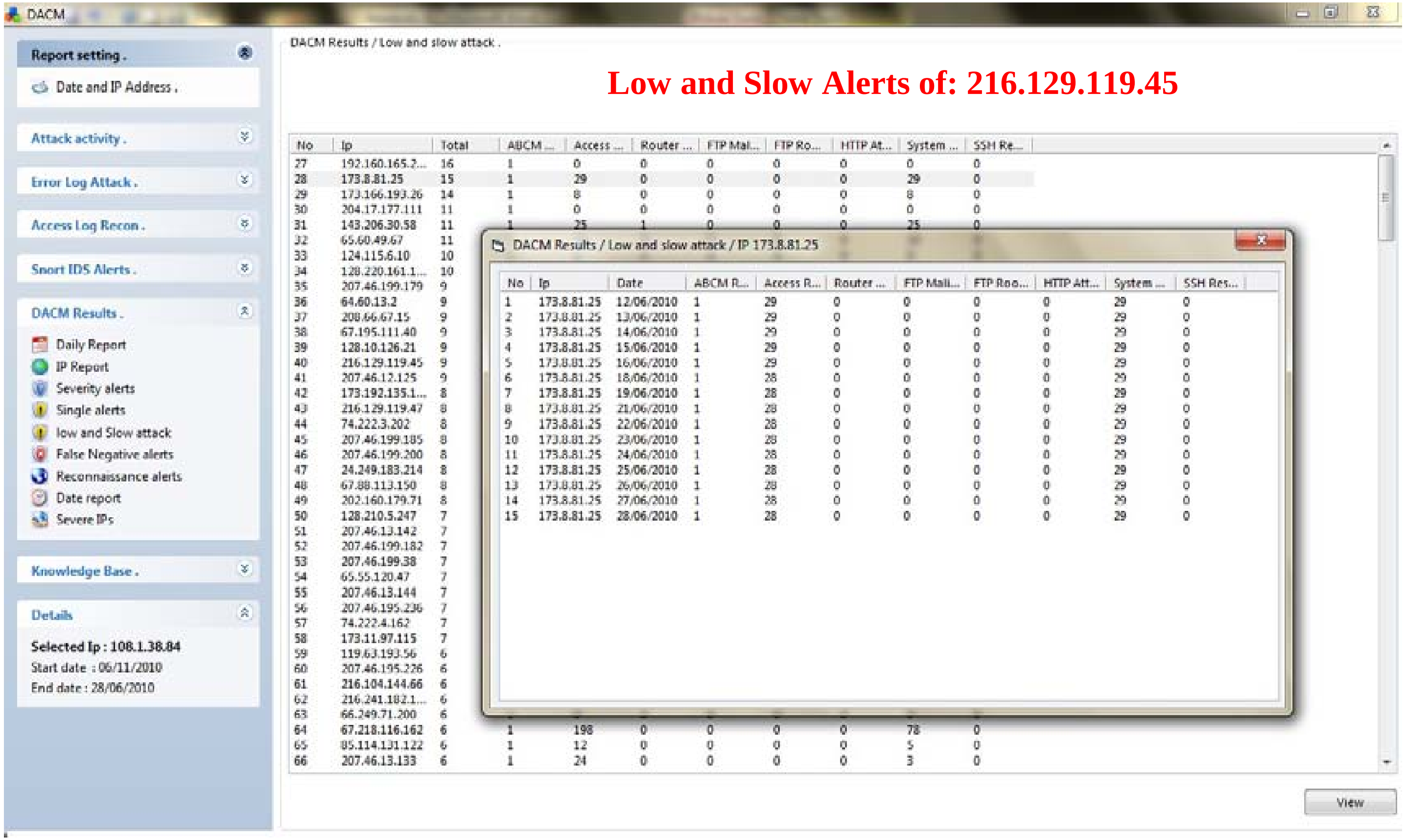
Low and Slow Alerts of: 192.160.165.222

No	Ip	Total	ABCM Resu...	Access Res...	Router Res...	FTP Malicious	FTP Rooting	HTTP Attacks	System Scan	SSH Results
27	192.160.165.222	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14								
30	204.17.177.111	11								
31	143.206.30.58	11								
32	65.60.49.67	11								
33	124.115.6.10	10								
34	128.220.161.101	10								
35	207.46.199.179	9								
36	64.60.13.2	9								
37	208.66.67.15	9								
38	67.195.111.40	9								
39	128.10.126.21	9								
40	216.129.119.45	9								
41	207.46.12.125	9								
42	173.192.135.178	8								
43	216.129.119.47	8								
44	74.222.3.202	8								
45	207.46.199.185	8								
46	207.46.199.200	8								
47	24.249.183.214	8								
48	67.88.113.150	8								
49	202.160.179.71	8								
50	128.210.5.247	7								
51	207.46.13.142	7								
52	207.46.199.182	7								
53	207.46.199.38	7								
54	65.55.120.47	7								
55	207.46.13.144	7								
56	207.46.195.236	7								
57	74.222.4.162	7								
58	173.11.97.115	7								
59	119.63.193.56	6								
60	207.46.195.226	6								
61	216.104.144.66	6	1	0	0	0	0	0	0	0
62	216.241.182.150	6	1	114	0	0	0	0	0	0
63	66.249.71.200	6	1	0	0	0	0	0	0	0
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	85.114.131.122	6	1	12	0	0	0	0	5	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0

DACM Results / Low and slow attack / IP 192.160.165.222

No	Ip	Date	ABCM Results	Access R...	Router ...	FTP Mali...	FTP Roo...	HTTP Att...	System ...	SSH Re.
1	192.160.165.222	12/06/2010	1	0	0	0	0	0	0	0
2	192.160.165.222	13/06/2010	1	0	0	0	0	0	0	0
3	192.160.165.222	14/06/2010	1	0	0	0	0	0	0	0
4	192.160.165.222	15/06/2010	1	0	0	0	0	0	0	0
5	192.160.165.222	16/06/2010	1	0	0	0	0	0	0	0
6	192.160.165.222	17/06/2010	1	0	0	0	0	0	0	0
7	192.160.165.222	18/06/2010	1	0	0	0	0	0	0	0
8	192.160.165.222	19/06/2010	1	0	0	0	0	0	0	0
9	192.160.165.222	21/06/2010	1	0	0	0	0	0	0	0
10	192.160.165.222	22/06/2010	1	0	0	0	0	0	0	0
11	192.160.165.222	23/06/2010	1	0	0	0	0	0	0	0
12	192.160.165.222	24/06/2010	1	0	0	0	0	0	0	0
13	192.160.165.222	25/06/2010	1	0	0	0	0	0	0	0
14	192.160.165.222	26/06/2010	1	0	0	0	0	0	0	0
15	192.160.165.222	27/06/2010	1	0	0	0	0	0	0	0
16	192.160.165.222	28/06/2010	1	0	0	0	0	0	0	0

A-16 Larger Image of Figure 5.21 Low and Slow Attack for 192.160.165.222



Selected Ip : 108.1.38.84
Start date : 06/11/2010
End date : 28/06/2010

DACM Results / Low and slow attack .

Low and Slow Alerts of: 216.129.119.45

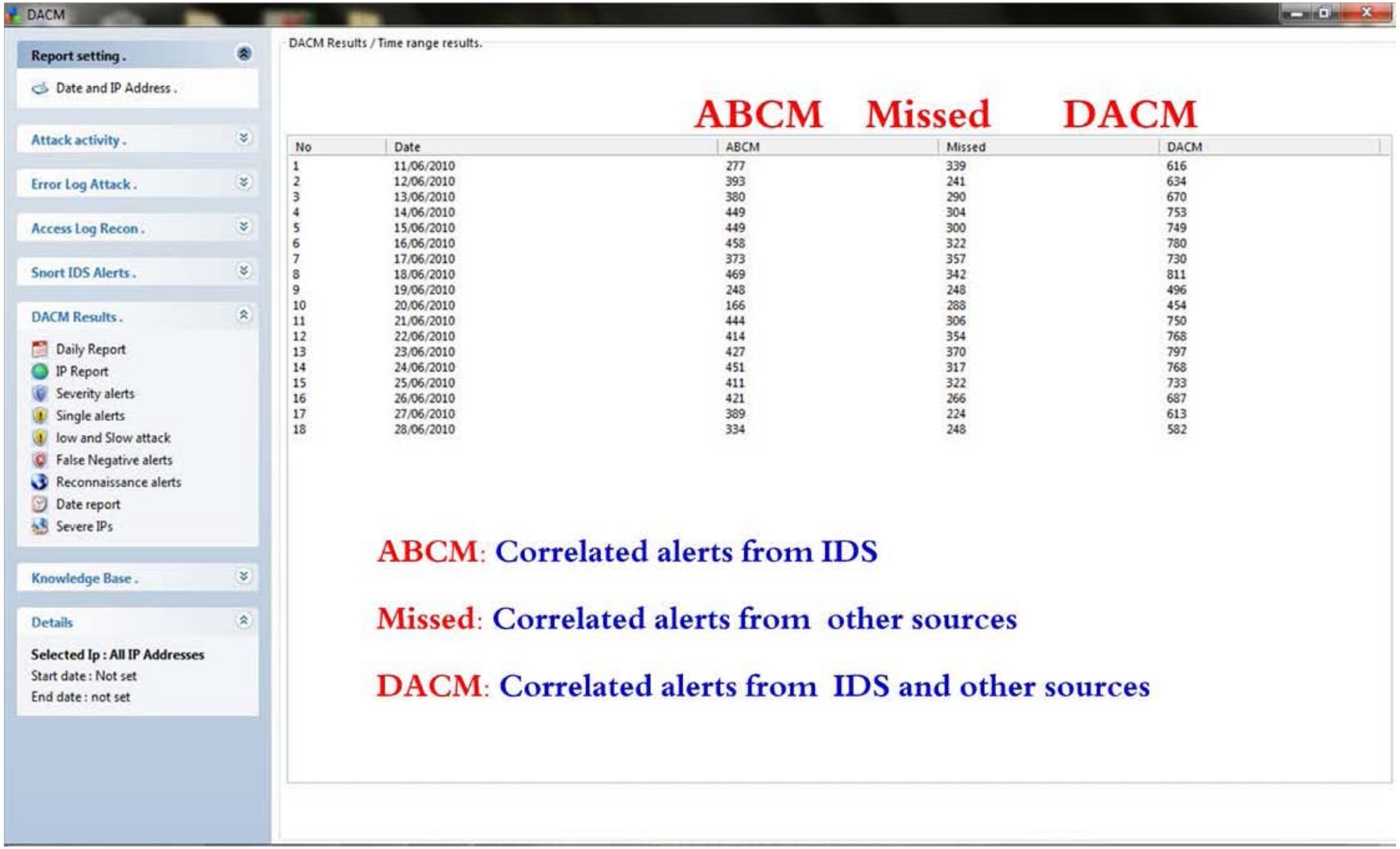
No	Ip	Total	ABCM ...	Access ...	Router ...	FTP Mai...	FTP Ro...	HTTP At...	System ...	SSH Re...
27	192.160.165.2...	16	1	0	0	0	0	0	0	0
28	173.8.81.25	15	1	29	0	0	0	0	29	0
29	173.166.193.26	14	1	8	0	0	0	0	8	0
30	204.17.177.111	11	1	0	0	0	0	0	0	0
31	143.206.30.58	11	1	25	1	0	0	0	25	0
32	65.60.49.67	11								
33	124.115.6.10	10								
34	128.220.161.1...	10								
35	207.46.199.179	9								
36	64.60.13.2	9								
37	208.66.67.15	9								
38	67.195.111.40	9								
39	128.10.126.21	9								
40	216.129.119.45	9								
41	207.46.12.125	9								
42	173.192.135.1...	8								
43	216.129.119.47	8								
44	74.222.3.202	8								
45	207.46.199.185	8								
46	207.46.199.200	8								
47	24.249.183.214	8								
48	67.88.113.150	8								
49	202.160.179.71	8								
50	128.210.5.247	7								
51	207.46.13.142	7								
52	207.46.199.182	7								
53	207.46.199.38	7								
54	65.55.120.47	7								
55	207.46.13.144	7								
56	207.46.195.236	7								
57	74.222.4.162	7								
58	173.11.97.115	7								
59	119.63.193.56	6								
60	207.46.195.226	6								
61	216.104.144.66	6								
62	216.241.182.1...	6								
63	66.249.71.200	6								
64	67.218.116.162	6	1	198	0	0	0	0	78	0
65	85.114.131.122	6	1	12	0	0	0	0	5	0
66	207.46.13.133	6	1	24	0	0	0	0	3	0

DACM Results / Low and slow attack / IP 173.8.81.25

No	Ip	Date	ABCM R...	Access R...	Router ...	FTP Mail...	FTP Roo...	HTTP Att...	System ...	SSH Res...
1	173.8.81.25	12/06/2010	1	29	0	0	0	0	29	0
2	173.8.81.25	13/06/2010	1	29	0	0	0	0	29	0
3	173.8.81.25	14/06/2010	1	29	0	0	0	0	29	0
4	173.8.81.25	15/06/2010	1	29	0	0	0	0	29	0
5	173.8.81.25	16/06/2010	1	29	0	0	0	0	29	0
6	173.8.81.25	18/06/2010	1	28	0	0	0	0	29	0
7	173.8.81.25	19/06/2010	1	28	0	0	0	0	29	0
8	173.8.81.25	21/06/2010	1	28	0	0	0	0	29	0
9	173.8.81.25	22/06/2010	1	28	0	0	0	0	29	0
10	173.8.81.25	23/06/2010	1	28	0	0	0	0	29	0
11	173.8.81.25	24/06/2010	1	28	0	0	0	0	29	0
12	173.8.81.25	25/06/2010	1	28	0	0	0	0	29	0
13	173.8.81.25	26/06/2010	1	28	0	0	0	0	29	0
14	173.8.81.25	27/06/2010	1	28	0	0	0	0	29	0
15	173.8.81.25	28/06/2010	1	28	0	0	0	0	29	0

A-17 Larger Image of Figure 5.22 Low and Slow Attack for 216.129.119.45

A-18 Larger Image of Figure 5.23 DACM Summary Report



Selected Ip : All IP Addresses
Start date : Not set
End date : not set

DACM Results / Time range results.

ABCM Missed DACM

No	Date	ABCM	Missed	DACM
1	11/06/2010	277	339	616
2	12/06/2010	393	241	634
3	13/06/2010	380	290	670
4	14/06/2010	449	304	753
5	15/06/2010	449	300	749
6	16/06/2010	458	322	780
7	17/06/2010	373	357	730
8	18/06/2010	469	342	811
9	19/06/2010	248	248	496
10	20/06/2010	166	288	454
11	21/06/2010	444	306	750
12	22/06/2010	414	354	768
13	23/06/2010	427	370	797
14	24/06/2010	451	317	768
15	25/06/2010	411	322	733
16	26/06/2010	421	266	687
17	27/06/2010	389	224	613
18	28/06/2010	334	248	582

ABCM: Correlated alerts from IDS

Missed: Correlated alerts from other sources

DACM: Correlated alerts from IDS and other sources

APPENDIX B

DACM Agents Formal Description

Appendix B: DACM Agents Formal Description

B-1: IDS Alert Correlation

A_i (sensor, ID, source, timestamp, destination, type) $\in IDS\ alerts$
A_{i+1} (sensor, ID, source, timestamp, destination, type) $\in IDS\ alerts$
A_i and A_{i+1} could be correlated using different correlation components, these components represents specific criteria in which they use to correlate the alerts, the criteria of each component will be described in mathematical relation of the alerts attributes. Correlation components used in [56 - 58] were formally described as follows:

Alert Fusion Correlation Component can be described as follows:

A_i, Ai+1 will be correlated together
If source (A_i) =source (A_{i+1}) and
Destination (A_i) =destination (A_{i+1}) and
Type (A_i) =type (A_{i+1}) and
|Time (A_i) – Time (A_F)| <= $T_{threshold}$ and
Sensor (A_i) <> sensor (A_{i+1})
Where $T_{threshold}$ is the minimum allowed difference time

Threat Reconstruction Correlation Component can be described as follows:

A_i , A_{i+1} will be correlated together
If source (A_i) =source (A_{i+1}) and
Destination (A_i) =destination (A_{i+1}) and
Type (A_i) =type (A_{i+1}) and
Sensor (A_i) = sensor (A_{i+1}) and
|Time (A_i) – Time (A_F)| <= T_{window}
Where T_{window} is the minimum allowed difference time to correlate two alerts from same source

Focus Recognition Correlation Component can be described as follows:

A_i , A_{i+1} will be correlated together
If (source (A_i) < > source (A_{i+1}) and
Destination (A_i) =destination (A_{i+1}) and
Type (A_i) =type (A_{i+1}) and
Sensor (A_i) = sensor (A_{i+1}) and
$|$Time (A_i) – Time (A_F)$|$ <= T_{window})
Or
If (source (A_i) = source (A_{i+1}) and
Destination (A_i) < > destination (A_{i+1}) and
Type (A_i) =type (A_{i+1}) and
Sensor (A_i) = sensor (A_{i+1}) and
$|$Time (A_i) – Time (A_F)$|$ <= T_{window})
Where T_{window} is the minimum allowed difference time to correlate two alerts from same source

Multi Step Attack Correlation Component can be described as follows:

A_i , A_{i+1} will be correlated together
If Destination (A_i) = source (A_{i+1}) and
Time (A_i) < Time (A_{i+1})

B-2: InfoSec Tools Agents

Firewall Agent can be described as follows:

$A_{FW} \in$ Firewall alerts
Firewall Entry (IP, Date, Time, Destination, Port) ∈ Firewall Log
∀ Entry ∈ Firewall Log
Firewall Entry (IP, Date, Time, Destination, Port) ⟶ Firewall Attack Table
Read next FTP Entry

Vulnerability Scanner Agent can be described as follows:

A_{VS} ∈ Vulnerability Scanner alerts

Vulnerability Scanner Entry (Date, Time, Port, service, Status) ∈ Vulnerability Scanner Log

∀ Entry ∈ Vulnerability Scanner Log

If status (Vulnerability Scanner Entry) is open

Then Port (Vulnerability Scanner Entry) is vulnerable, Produces A_{VS}

Vulnerability Scanner Entry (Date, Time, Port, Service, Status) → Vulnerability Scanner Alert Table

Else

Read Vulnerability Scanner Entry

B-3: Service and Application Logs Agents Formal Description

FTP Agent can be described as follows:

A_{FTP} ∈ FTP alerts

FTP Entry (IP, Date, Time, Command, User) ∈ FTP Log

{S}: set of unauthorized FTP commands; **{U}:** set of unauthorized users

∀ Entry ∈ FTP Log

If command (FTP Entry) ∈ {S} Or user (FTP Entry) ∈ {U}

Then FTP Entry is malicious, Produces AFTP

FTP Entry (IP, Date, Time, Command, User) → FTP Attack Table

Else

Read next FTP Entry

SSH Agent can be described as follows:

ASSH ∈ FTP alerts

SSH Entry (Date, Time, Source IP, Sport, error message) ∈ SSH Log

{S}: list of error messages associated with attack signatures

∀ Entry ∈ SSH Log

If error message (SSH Entry) ∈ {S}

Then SSH Entry is malicious, Produces ASSH

SSH Entry ((Date, Time, Source IP, Sport, error message) → SSH Attack Table

Else

Read next SSH Entry

HHTP and HTTPS Agents can be described as follows:

A_{HTTP} ∈ HTTP alerts , A_{HTTPS} ∈ HTTPS alerts

HTTP Entry (Date, Time, Source IP, error sequence messages) ∈ SSH Log

{S}: list of error sequence messages associated with attack profiles

{N}: list of error sequence messages associated with normal profiles

∀ Entry ∈ HTTP/HTTPS Log

If error sequence messages (HTTP Entry) ∈ {S}

Then HTTP Entry is malicious, Produces AHTTP/HTTPS

HTTP Entry ((Date, Time, Source IP, Sport, error sequence message) → HTTP/HTTPS Attack Tables

Else

If error sequence messages (HTTP Entry) ∈ {N}

Read next HTTP Entry

B-4: DACM Central Agent can be described as follows:

A_i ∈ IDS alerts, A_f ∈ Firewall alerts , A_L ∈ log alerts ;

A_i (source, time, destination, type) ∈ IDS alerts

AF (source, time, destination) ∈ Firewall alerts

AL (source, time, destination, type) ∈ Logs alerts

∀ alert Ai

A_i Is verified alerts w.r.t. A_F

If source (Ai) = source (AF) And Destination (Ai) = Destination (AF)

And $|\text{Time (Ai)} - \text{Time (AF)}| <= T_{threshold}$

Where $T_{threshold}$ is the minimum allowed difference time

Ai Is verified alerts w.r.t. A_L

If source (A_i) = source (A_L) And Destination (Ai) = Destination (AL)

And $|\text{Time } (A_i) - \text{Time } (A_L)| <= T_{threshold}$

Where $T_{threshold}$ is the minimum allowed difference time

A_i Is IDS only

If attributes (A_i) < > attributes (A_L) OR

Attributes (A_i) < > attributes (A_F)

A_i Is Low and Slow attack

A_i is IDS only and Count (source [A_i]) = 1 per day

And days (source [Ai]) > 3

∀ alert A_L,

AL is negative alert w.r.t. A_I

If source (A_i) = source (A_L) and

Time (A_i) < > Time (A_L)

Or attributes (A_L) < > attributes (A_i)

AL is reconnaissance

If count (A_L) > A_{Th}

Where A_l is access count of specific IP / day and

A_{TH} : allowed threshold access per day

REFERENCES

LIST OF PUBLICATIONS

1- Ayman E. Taha, Ismail Abdel Ghaffar, Ayman M. Bahaa Eldin, Hani M. K. Mahdi, “Agent Based Correlation Model for Intrusion Detection Alerts”, pp 89-94 proceeding of IEEE International Conference on Intelligence and Security Informatics (ISI 2010), May 2010, Vancouver, Canada.

2- Ismail Abdel Ghaffar, Ayman E. Taha, Ayman M. Bahaa Eldin , Hani M. K. Mahdi, “Towards Implementing Agent Based Correlation Model for Real-Time Intrusion Detection Alerts”, proceeding of 7th International Conference on Electrical Engineering, ICEENG 2010, May 2010, MTC, Cairo, Egypt.

3- Ayman M. Bahaa Eldin , Hani M. K. Mahdi, Ayman E. Taha, Ismail Abdel Ghaffar, “Dynamic Parallel correlation Model for intrusion detection alerts”, poster in Annual Information Security Symposium of Center of Education and Research of Information Assurance and Security (CERIAS), Purdue University, March 2010, west Lafayette, Indiana, USA.

4- Ayman E. Taha, Ayman M. Bahaa Eldin, Ismail Abdel Ghaffar, Hani M. K. Mahdi, “Distributed Agents Correlation Model for intrusion detection in computer network” , Computers & Security Journal , Elsevier, In Progress.

REFERENCES

[1] Karen Scarfone, Peter Mell , “*Guide to Intrusion Detection and Prevention Systems (IDPS),*” National Institute of Standards and Technology ,NIST Special Publication 800-94, Computer Security, February 2007.

[2] Tianning Zang, Xiaochun Yun, Yongzheng Zhang, “A Survey of Alert Fusion Techniques for Security Incident,” *Proceeding of the Ninth International Conference on Web-Age Information Management IEEE*, November, 2008.

[3] Agent Maíra Gatti, Arndt von Staa , “Testing & Debugging Multi-Agent Systems: A State of the Art Report,” ISSN: 0103-9741, February, 2006

[4] H. Debar, M. Dacier, and A. Wespi. “Towards taxonomy of intrusion detection systems,” *Computer Networks*,” vol.31, No.8, pp. 805-822, 1999

[5] Gene H. Kim and Eugene H. Spafford. “The Design and Implementation of Tripwire: A File System Integrity Checker,” *Technical report*, Purdue University, November, 1993.

[6] G.Vigna, W.Robertson, V.Kher, and R.A. Kemmerer. “A Stateful Intrusion Detection System for World-Wide Web Servers.” *In Proceedings of the Annual Computer Security Applications Conference (ACSAC 2003)*, pp. 34-43, Las Vegas, NV, December, 2003.

[7] Mandiant, “M-TRENDS, the advanced persistent threat”, http:-//www.princeton.edu/~yctwo-/files/readings/M-Trends.pdf, June, 2010.

[8] Advanced Persistent Threats (APTs), http://www.damballa.com/knowledge/advanced-persistent-threats.php, June, 2010.

[9] Bashar Ewaida , “Pass-the-hash attacks: Tools and Mitigation,” Technical paper, SANS Institute InfoSec Reading Room, January, 2010.

[10] Carl Endorf , Gene Schultz , Jim Mellander,”*Intrusion Detection and Prevention*,” McGraw-Hill, ISBN: 978-0072229547, December, 2003.

[11] Robiah Yusof, Siti Rahayu Selamat, and Shahrin Sahib “Intrusion alert correlation technique analysis for heterogeneous log,” *IJCSNS International Journal of Computer Science and Network Security*, vol.8, No.9, September 2008.

[12] Zhai, Y., Ning, P., & Xu, J. "Integrating IDS alert correlation and OS-level dependency tracking." North Carolina State University, North Carolina, 2005.

[13 Tianning Zang, Xiaochun Yun, Yongzheng Zhang, "A Survey of Alert Fusion Techniques for Security Incident", *The Ninth International Conference on Web-Age Information Management*, July 2008

[14] C. Mu, H. Huang, and S. Tian, "A survey of intrusion-detection alert aggregation and correlation techniques," *Journal of Computer Research and Development*, vol. 43, pp. 1-8, 2006.

[15] Donghai Tian, Hu Changzhen, Yang Qi, and Wang Jianqiao "Hierarchical Distributed alert correlation model," *2009 Fifth International Conference on Information Assurance and Security*, pp. 765-768, August, 2009.

[16] Chenfeng VincentZhou, ChristopherLeckie, Shanika Karunasekera, "Decentralized multi-dimensional alert correlation for collaborative intrusion detection," *scienceDirect, Journal of Network and Computer Applications*, vol. 32, pp. 1106–1123, February, 2009.

[17] D. Curry and H. Debar, "Intrusion Detection Message Exchange Format: Extensible Markup Language (XML) Document Type Definition," draft-ietf-idwg-idmef-xml-10.txt+, January, 2003.

[18] The Intrusion Detection Message Exchange Format [Online], Available: http://www.ietf.org/rfc/rfc4765.txt`, January, 2010.

[19] D. Andersson, M. Fong, and A. Valdes. "Heterogeneous Sensor Correlation: A Case Study of Live Traffic Analysis.", *In Proceedings of the 3rd Annual IEEE Information Assurance Workshop*, United States Military Academy West Point, New York, June 2002.

[20] H. Debar and A. Wespi, "Aggregation and correlation of intrusion detection alerts," *Proceeding of International Symposium. Recent Advances in Intrusion Detection*, pp. 85-103, October, 2001.

[21] Tian Zhihong, Qin Baoshan, Ye Jianwei, Zhang Hongli, "Alertclu: A Realtime Alert Aggregation and Correlation System," *International Conference on Cyber worlds 2008*, pp

778-781, September, 2008.

[22] R. Gula. Correlating IDS Alerts with Vulnerability Information. *Technical report*, Tenable Network Security, December 2002.

[23] B. Morin and H. Debar., "Correlation of Intrusion Symptoms: an Application of Chronicles," *In Proceedings of the International Symposium on Recent Advances in Intrusion Detection,* Pittsburgh, PA, September, 2003.

[24] D. Xu, and P. Ning, "Alert Correlation through Triggering Events and Common Resources", *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC '04),* December, 2004.

[25] Wang Li Li Zhi-tang Lei Jie, "Learning attack strategies through mining and correlation of security alarms", *Proceeding of 10th IFIP/IEEE International Symposium on Integrated Network Management*, pp. 713-717, May, 2007

[26] Steven Noel, Eric Robertson, Sushil Jajodia, "Correlating Intrusion Events and Building Attack Scenarios Through Attack Graph Distances", *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC'04),* December, 2004

[27] Nessus Vulnerabilty Scanner, http://www.nessus.org/, June, 2010

[28] F. Cuppens and A. Miege, Alert correlation in a cooperative intrusion detection framework, *In Proceedings of The 2002 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2002.

[29] S. T. Eckmann, G. Vigna, and R.A. Kemmere, "STATL: An Attack Language for State-based Intrusion Detection," *Journal of Computer Security*, 10(1/2), pp. 71–104, 2002.

[30] B. Morin and H. Debar, Correlation of intrusion symptoms: an application of chronicles, *In Proceedings of The 6th International Conference on Recent Advances in Intrusion Detection (RAID'03),* September 2003.

[31] R.A. Kemmer, G. Vigna, A Model-Based Real- Time Intrusion Detection System for Large Scale heterogeneous Networks, California Univeristy, Santa Barbara, Department of Computer Science, *Technical Report [Online*], August 2003, Available: http://www.stormingmedia.us/42/4280/A428024.html

[32] C. Krugel, T. Tuth, and C. Kerer, "Decenlralized event

correlation for intrusion detection," *In International Conference on Information Security and Cryptology (IUSC),* December, 2001.

[33] P. Ning, Y. Cui, D. S. Reeves, and D. Xu, "Techniques and tools for analyzing intrusion alerts," *ACM Transactions on Information and Systems Security*, vol. 7, pp. 274-318, 2004.

[34] P. Ning, Y. Cui, and D. S Reeves, Analyzing intensive intrusion alerts via correlation, *In Proceedings of The 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002),* pp. 74–94, Zurich, Switzerland, October, 2002.

[35] S. Templeton and K. Levitt, "A requires/provides model for computer attacks," *In Proceedings of New Security Paradigms Workshop,* pp. 31–38. ACM Press, September, 2000.

[36] P. Ning, Y. Cui, and D. S Reeves, "Constructing attack scenarios through correlation of intrusion alerts," *In Proceedings of The 9th ACM Conference on Computer and Communications Security,* pp. 245–254, Washington, D.C., November, 2002.

[37] Xinzhou Qin, "A Probabilistic-Based Framework for INFOSEC Alert Correlation", *Ph.D. Thesis,* College of Computing, Georgia Institute of Technology, Georgia, USA, August, 2005.

[38] X. Qin and W. Lee, "Statistical Causality Analysis of INFOSEC Alert Data. *In Proceedings of The 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003),* vol. 2820 of Lecture Notes in Computer Science, Springer–Verlag. Heidelberg, Germany, pp. 73–93, 2003.

[39] P. Porras, M. Fong, and A. Valdes, "A Mission-Impact-Based Approach to INFOSEC Alarm Correlation," *In Proceedings of the. International Symposium. The Recent Advances in Intrusion Detection,* pp. 95-114, Zurich, Switzerland, October 2002.

[40] Nmap- Network Mapper, Security Scanner For Network Exploration & Hacking. http://nmap.org/, June, 2010

[41] Catalin Leordeanu, Levni Arif and Valentin Cristea, "Correlation of Intrusion Detection Information in Grid Environments," *2010 International Conference on Complex, Intelligent and Software Intensive Systems*, pp. 463-468, February, 2010.

[42] Wen Long, Yang Xin, Yixian Yang "Vulnerabilities Analyzing

Model for Alert Correlation in Distributed Environment,*” 2009 IITA International Conference on Services Science, Management and Engineering,* pp. 408-411, November, 2009.

[43] Guofei Jiang, Member, George Cybenko, “Temporal and Spatial Distributed Event Correlation for Network Security,” *Proceedings of the American Control Conference*, 30 June-2 July, 2004

[44] P. Ning and D. Xu. “Learning Attack Strategies from Intrusion Alert,” *In Proceedings of the ACM Conference on Computer and Communications Security (CCS '03)*, Washington, DC, October 2003.

[45] V. Honavar and L. Miller and J. S. K. Wong, “Distributed knowledge networks,” *IEEE Information Technology Conference, Syracuse*, pp. 87-90, 1998.

[46] Dalila Boughaci, Habiba drias, Ahmed Bendib, “A Distributed Intrusion Detection Framework based on Autonomous and Mobile Agents,” *Proceedings of the International Conference on Dependability of Computer Systems IEEE*.

[47] Mohamad Eid, Hassan Artail, Ayman Kayssi, and Ali Chehab, “A Lightweight Adaptive Mobile Agent-based Intrusion Detection System LAMAIDS,” *International Journal of Network Security*, Vol.6, No.2, pp. 145–157, March, 2008

[48] Amir Vahid Dastjerdi, and Kamalrulnizam Abu Bakar, “A Novel Hybrid Mobile Agent Based Distributed Intrusion Detection System,” *Proceedings of world academy of science, engineering and technology,* vol. 35, ISSN 2070-3740, November, 2008.

[49] Jianxiao Liu , Lijuan Li , “A Distributed Intrusion Detection System Based on Agents,” *2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pp. 553-557, December, 2008.

[50] Mark Crosbie, Gene Spafford, “Active Defense of computer system using autonomous agent,” *Technical report no 95-008, COAST group,* computer science department, Purdue University, February, 1995.

[51] Jai Sundar Balasubramaniyan, Eugene Spafford, Diego Zamboniy, “An Architecture for Intrusion Detection using Autonomous Agents,” *COAST Technical Report 98/05*, COAST

Laboratory, Purdue University, June 11, 1998

[52] Farah Barika KTATA, Nabil EL KADHI, Khaled GHEDIRA, "Distributed agent architecture for intrusion detection based on new metrics," *Proceeding 2009 Third International Conference on Network and System Security*, pp. 321-327, October, 2009.

[53] Snort – the open source network intrusion prevention and detection system. http://www.snort.org, 2010.

[54] Abduljalil A. Mohamed, Otman Basir, "Fusion Based Approach for Distributed Alarm Correlation in Computer Networks," 2010 *Second International Conference on Communication Software and Networks*, pp. 318-324, February, 2010.

[55] A.A Mohamed and O. Basir, "An Adaptive Multi-Agent Approach for Distributed Alarm Correlation and Fault Identification," *Parallel and Distributed Computing and Networks*, February, 2010.

[56] F.Valeur, G.Vigna, C. Kruegel, and R.A.Kemmerer, "Comprehensive approach to intrusion detection alert correlation," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, pp. 146-69, July-September, 2004.

[57] F. Valeur, "Real-time Intrusion Detection Alert Correlation," *Ph.D. Thesis*, University of California Santa Barbara, Santa Barbara, California, USA, 2006.

[58] Christopher Kruegel, Fredrik Valeur, Giovanni Vigna, "*Intrusion Detection and Correlation Challenges and Solutions,*" ISBN: 0-387-23398-9, Springer, 2005.

[59] Basic Analysis and Security Engine (BASE), http://base.secureideas.net/about.php, June, 2010.

[60] David W Chadwick, "Network Firewall Technologies," *Technical Report*, IS Institute, University of Salford, Salford, M5 4WT, England.

[61] Avi Kak, "Port Scanning, Vulnerability Scanning, Packet Sniffing, and Intrusion Detection," *Lecture Notes on "Computer and Network Security,"* Purdue University, April, 2011.

[62] Franck.Veysset, Laurent.Butti, "Honey pot technologies," First Conference, France Télécom R&D, June, 2006.

[63] Wireshark, Network protocol analyzer, http://www.wireshark-.org, June, 2010.

[64] Center of Education and Research for Information Assurance and Security (CERIAS), http://www.cerias.purdue.edu, June, 2011.

[65] M. Cotton,L.Vegoda, "Special Use IPv4 Addresses," Internet Engineering Task Force (IETF),ISSN: 20701721, http://tools.ietf.org/html//rfc5735, June, 2010.

[66] Vijay Ahuja, “Network and internet security”, AP professional, 1996.

[67] William Stallings, “Network and internetworking security principles and practices”, Prentice Hall, New Jersy, 1995.

شـــكر

بداية أشكر الله سبحانه وتعالى لتوفيقه لى بما مكننى من اتمام هذا العمل.

اود ان اعبر بصدق عن شكرى العميق لمشرفى الرسالة أ.د **هانى كمال مهدى** استاذ هندسة الحاسبات ، قسم هندسة الحاسبات والنظم ، كليه الهندسة جامعة عين شمس، أ.د **اسماعيل عبد الغفار** استاذ هندسة الحاسبات بقسم هندسة الحاسبات وبحوث العمليات ومدير الكلية الفنية العسكرية و د **ايمن محمد بهاء الدين** استاذ مساعد هندسة الحاسبات ، قسم هندسة الحاسبات والنظم ، كليه الهندسة جامعة عين شمس . لقد كنت محظوظا باشراف هذه النخبة من الاساتذة ، لقد قاموا بتقديم نصائح وملاحظات لا تقدر بثمن ، مناقشات مفيدة ونافعة فى اختيار موضوع البحث واثناء تنفيذ هذا العمل. انا مدين لهم جميعا بشكر خاص فقد منحوني الكثير من وقتهم خلال سنوات إعداد هذه الرسالة. فلم أكن لأتم هذه الرسالة بدون دعمهم وتقديم المشورة العلمية والاقتراحات المفيدة، والمراجعة الدقيقة لجميع مراحل العمل.

اود تقديم الشكر لاعضاء لجنة المناقشة ومراجعة الرسالة استاذ دكتور يوجين سبافورد رئيس مركز تعليم وابحاث تامين المعلومات بجامعة بوردو بالولايات المتحدة الامريكية واستاذ دكتور ياسر هشام دكرورى الاستاذ بقسم هندسة الحاسبات والنظم بكلية الهندسة جامعة عين شمس ورئيس جامعة التعليم الالكترونى بجمهورية مصر العربية، فقد قامو بامدادى بالعديد من الملاحظات المدروسة والتوجيهات والارشادات القيمة.

اود تقديم الشكر لمركز تعليم وابحاث تامين المعلومات بجامعة بوردو بالولايات المتحدة الامريكية. اقدر الدعم القيم لمدير المركز الدكتور يوجين سبافورد والجهد السخى لطاقم العمل خاصة مهندس ابحاث تامين المعلومات كيث واتسون لتعاونهم المثمر خلال زيارتى العلمية للمركز وما وفروه من مصادر عظيمة لجمع البيانات والمواد العلمية اللازمة لاتمام هذا العمل.

اقدر مجهود ودعم زملائى بالعمل بمركز بحوث العمليات خلال هذا العمل وشكر خاص لزملائى احمد عبد الصبور وجلال محمد لمساعدتهم فى بناء النموذج المقترح.

لا استطيع تقديم الشكر المناسب لامى وعائلتى لدعمهم الدائم وتشجيعهم لى خلال حياتى كلها وانى احاول تقديم هذا العمل لاجعلهم فخورين بى، اخيرا انا فى منتهى الامتنان لزوجتى داليا واطفالى الاحباء آسر وسما لصبرهم وتضحيتهم خلال فترة الدراسة وخاصة خلال سفرى خارج البلاد لجمع المادة العلمية ، واشكرهم لتشجيعهم ودعمهم المعنوى الدائم ، وأشكر الله دائما على منحى هذه الاسرة الرائعة. و فى النهاية أود اهداء هذا العمل اليهم والى أمى وجميع افراد عائلتى.

متعددون موزعة فى تسلسل هرمى، حيث تقوم بترابط وتجميع التحذيرات الدالة على وجود اختراق من وسائل وانظمة اكتشاف الاختراق وايضا من ادوات تامين الشبكات وملفات الحفظ للتطبيقات والخدمات المختلفة. ترابط وتكامل التحذيرات من هذه المعلومات يحسن كفاء واكتمال اكتشاف الاختراق بالشبكة من خلال تقليل عدد التحذيرات الخاطئة السلبية او الايجابية. يقوم كل وكيل او عميل بتجميع دلائل اكتشاف الاختراق من خلال المعلومات المتاحة لديه باستخدام نماذج مطابقة تتيح له التعرف على بصمات الهجوم ويتيح تكامل هذه المعلومات تكوين تظام ترابط متكامل.

تناولت الرسالة بناء النماذج وتجربة النظام على قاعدة بيانات اختراقات تم تجميعها من شبكة مركز أبحاث تأمين المعلومات بجامعة بوردو بالولايات المتحدة الأمريكية. قام الطالب بتحليل االخوارزمات والنماذج المقترحة وتقييم الأداء لكل منها ودراستها تحليليا من حيث معدلات التعرف الصحيح على الاختراقات ونسبة الخطأ ومعدلات الترابط ومعدلات تقليل الإنذارات الكاذبة وكذلك أداء النظام.

خلاصة استخدام نموذج الارتباط الموزع باستخدام الوكلاء المتعددون يحسن كفاءة واكتمال نظام اكتشاف الاختراق، يتيح امكانية التطوير بمرونة ولا يعتمد على نظام تشغيل معين، كما انه يقلل الوقت والمجهود المطلوب لفهم الموقف المتكامل بالشبكة. استخدام النموذج يتيح امكانية اكتشاف اكبر من وسائل الهجوم ويتيح القدرة على الفرز بين التحذيرات الخطيرة او الاقل اهمية من خلال تنوع التحذيرات فى الوسائل المختلفة لنفس المصدر ويتيح فرز التحذيرات المطلوب رد فعل سريع لها او الاخرى التى لا تحتاج الى هذا.

يتيح النظام امكانية الاكتشاف المبكر لوسائل الهجوم الحديثة مثل الهجوم البطىء وتهديد الهجوم المتقدم المتواصل من خلال اكتشاف التحذيرات المنفصلة او محاولة بعض المستخدمين استطلاع الشبكة بشكل مبالغ فيه وغير مبرر لتحديد الثغرات الموجودة به قبل محاولة الهجوم. النظام معد للعمل فى التوقيت الحقيقى حيث يحتاج الى بعض التعديلات البسيطة لتنفيذ الترابط الفورى للتحذيرات.

نحن نعتقد أن الرسالة تقدم نموذج ونهج واعد الذى يجمع بين استخدام تقنيات ترابط التحزيرات والوكلاء المتعددون لاكتشاف الاختراق من أجل توفير خدمات تامين عالية للشبكات وخدمات الانترنت.

كلمات مفتاحية

اكتشاف الاختراق ، ترابط التحزيرات ، الأنظمة متعددة الوكلاء ، الوكيل المتعلم ، معدل التقليل

ملخص

أيمن السيد السيد طه

ترابط اساليب اكتشاف الاختراق فى شبكات الحواسب باستخدام نظام العملاء المتعددون

رسالة دكتوراه

جامعة عين شمس – كلية الهندسة ٢٠١١

ترابط تحذيرات او دلائل اكتشاف الاختراق هى الاداة التى تحلل التحذيرات من نظام او اكثر لانظمة اكتشاف الاختراق وتقدم تقرير مختصر يمثل الرؤية الشاملة لمحاولات الاختراق. التقنيات الحالية لترابط التحذيرات حسنت نتائج اكتشاف الاختراق من خلال تقليل التحذيرات الصادرة عنها وتقديمها فى تقارير مختصرة ولكن لا زال بها بعض اوجه القصور مثل المعدل العالى من التحذيرات الخاطئة و فقد احد التحذيرات التى تمثل الهجوم من خلال عدة خطوات متتالية والتاكد من بعض التحذيرات. كما يوجد بعض اساليب الهجوم الحديثة لا يتم اكتشافها مثل الهجوم البطىء وتهديد الهجوم المتقدم المتواصل وايضا استخدام بعض ادوات الهجوم لتقنيات يمكنها من الاختفاء والمناورة لانظمة اكتشاف الاختراق. اخيرا التقنيات الحالية تعتمد على ترابط تحذيرات أنظمة اكتشاف الاختراق فقط ولا تعتمد على التكامل بين جميع مصادر المعلومات المتاحة بالشبكة مثل وسائل التامين وملفات الحفظ للتطبيقات ونظم التشغيل.

تم استخدام انظمة العملاء او الوكلاء المتعددون بشكل موسع لانظمة اكتشاف الاختراق ، حيث يتم اضافة او حذف وكيل بدون الحاجة الى اعادة تشغيل النظام مما يتيح امكانية مرونة التوسع فى امكانيات النظام. يقوم كل وكيل بتنفيذ وظيفة بسيطة اعتمادا على امكانياته ولكنها تستنتج نتائج معقدة عند تبادل هذه المعلومات. استخدام انظمة العملاء او الوكلاء المتعددون يقلل استخدام امكانيات النظام ويجنب النظام التوقف عن العمل فى حالة استخدام وظيفة مركزية وفشلها، اخيرا تتيح الاكتشاف المتعدد للاختراق وتبادل المعلومات.

تهدف هذه الرسالة الى اثبات امكانية تحسين كفاءة واكتمال انظمة اكتشاف الاختراق من خلال تقليل معدل وعدد الانذارات الايجابية والسلبية وذلك باستخدام الترابط بين مختلف مصادر المعلومات المتاحة بالنظام او ببيئة التشغيل وشبكات الحواسب المستخدمة.

تقدم هذه الرسالة اطار نمطى لنموذج ترابط اكتشاف الاختراق الموزع باستخدام انظمة العملاء او الوكلاء المتعددون لتحذيرات ودلائل اكتشاف الاختراق فى شبكات الحواسب. هذا الاطار يدعم تكامل تقنيات ترابط متعددة ويمكن من بناء مكونات جديدة مستقبلاً بسهولة. يقدم الاطار وكلاء

جامعة عين شمس

كلية الهندسة

صفحة العنوان

اسم الباحث : أيمن السيد السيد طه

اسم الدرجة : دكتوراه الفلسفة فى الهندسة الكهربية

القسم التابع له : هندسة الحاسبات والنظم

اسم الكلية : كلية الهندسة – جامعة عين شمس

سنة التخرج : ١٩٩٢ - الكلية الفنية العسكرية.

سنة المنح : ٢٠١١

جامعة عين شمس

كلية الهندسة

تعريف بمقدم الرسالة

إسم الباحث : ايمن السيد طه

تاريـخ الميلاد : ٦ / ٨ / ١٩٧٠

محل الميلاد : القاهرة

المؤهل الدراسى : بكالوريوس الهندسة الكهربية – هندسة الحاسبات والنظم

الجهة المانحة لها : الكلية الفنية العسكرية

الدرجة العلمية الأولى : ماجستير الهندسة الكهربية – هندسة الحاسبات والنظم

الجهة المانحة لها : كلية الهندسة – جامعة عين شمس

تاريخ المنح : ٢٠٠٢

الوظيفة الحالية : عقيد مهندس بإدارة نظم المعلومات – القوات المسلحة

اسم مقدم البحث : ايمن السيد طه

التوقيع :

التاريخ : ٧ / ٧ / ٢٠١١

جامعة عين شمس

كلية الهندسة

قسم هندسة الحاسبات والنظم

رسالة دكتوراه

اسم الباحث : أيمن السيد السيد طه

عنوان الرسالة : ترابط اساليب اكتشاف الاختراق فى شبكات الحواسب باستخدام نظام العملاء المتعددون

الدرجة : دكتوراه الهندسة الكهربية

لجنة الإشراف

الاسم	الوظيفة
أ.د. هانى محمد كمال مهدى	أستاذ بقسم هندسة الحاسبات والنظم كلية الهندسة - جامعة عين شمس
أ.د. اسماعيل عبد الغفار فرج	أستاذ هندسة الحاسبات الكلية الفنية العسكرية
د. أيمن محمد بهاء الدين	أستاذ مساعد بقسم هندسة الحاسبات والنظم كلية الهندسة - جامعة عين شمس

تاريخ البحث: / /

الدراسات العليا

موافقة مجلس الكلية **موافقة مجلس الجامعة**

/ / **/ /**

جامعة عين شمس - كلية الهندسة

قسم هندسة الحاسبات والنظم

ترابط اساليب اكتشاف الاختراق فى شبكات الحواسب باستخدام نظام العملاء المتعددون

رسالة

مقدمة للحصول على درجة الدكتوراه فى الهندسة الكهربية

(هندسة الحاسبات والنظم)

مقدمة من

أيمن السيد السيد طه

ماجستير الهندسة الكهربية

(هندسة الحاسبات والنظم)

جامعة عين شمس – ٢٠٠٢

تحت اشراف

أ.د. هانى محمد كمال مهدى

أ.د. اسماعيل عبد الغفار فرج

د. أيمن محمد بهاء الدين

القاهرة - مصر

يوليو - ٢٠١١