Extensible Intrusion Detection System

1Manish Kumar*, 2Dr. M. Hanumanthappa

1Assistant Professor, Dept. of Master of Computer Applications, M. S. Ramaiah Institute of Technology, Bangalore and Research Scholar, Department of Computers Science and Applications, Bangalore University, Bangalore, India
2Professor, Dept. of Computer Science and Applications, Jnana Bharathi Campus, Bangalore University, Bangalore, India

Abstract: - Nowadays, lots of commercial and open source Intrusion Detection Systems (IDS) implementations have emerged and has been widely used in practice for identifying malicious behaviours against protected hosts or network environments. With the broad range of systems and networks in use today, many systems use multiple IDS which brings with it the associated challenge of achieving a consistent approach to IDs management and analysis of different types of IDS alerts. In this paper we are discussing the technique for alert correlation of heterogeneous IDS. We proposed and implemented an extensible IDS architecture, which consists of several sensors and a central management unit. A prototype implementation of the proposed architecture is presented in this paper as a proof-of-concept.

Keywords— Intrusion Detection Message Exchange Format (IDMEF), Intrusion Detection System (IDS), Network Based Intrusion Detection System (NIDS), Host Based Intrusion Detection System (HIDS)

I. INTRODUCTION

Interoperability of different type of Intrusion Detection System and alerts correlation has become one of the important issues. The benefits of a collaborative approach to intrusion detection are threefold: (1) greater clarity about attacker intent, (2) precise models of adversarial behaviour, (3) and a better view of global network attack activity [5].

The Internet Engineering Task Force (IETF) has been working on an intrusion alert data model and accompanying message format standard called the Intrusion Detection Message Exchange Format (IDMEF). One of the main design goals of the IDMEF data model is to be able to express relationships between alerts.

II. INTRUSION DETECTION MESSAGE EXCHANGE FORMAT (IDMEF)

The purpose of the IDMEF [7] is to define data formats and exchange procedures for sharing information of interest to intrusion detection and response systems and to the management systems that may need to interact with them. The purpose of the IDMEF is to define data formats and exchange procedures for sharing information of interest to Intrusion Detection and Response Systems (IDRS), and to the management systems that may need to interact with them [6]. The implementation of IDMEF as mentioned in [7] could be useful and beneficial to:

a) A single Database Management System (DBMS) that could store the results from a variety of sensors would make it possible for data analysis and reporting activities to be performed on ‘the whole picture’ instead of just a part of it.

b) An Alert Correlation System (ACS) that could accept alerts from a variety of sensors would be capable of performing correlation and calculations.

c) A Graphical User Interface (GUI) that could display alerts from a variety of sensors would enable the user to monitor all of the sensors from a single screen, and require the user to learn only one interface, instead of several.

d) A Common Data Exchange (CDE) format would make it easier for different organizations (users, vendors, response teams, law enforcement) to not only exchange data, but also communicate about it.

III. EXTENSIBLE IDS Prototype Implementation

Goals of our research is to implement a prototype of Extensible IDS that works upon standardized communication and alerting formats. Our approach is to collect the alerts from HIDS, NIDS or through other sensors installed in the environment in IDMEF format for alert correlation at the central system. If the HIDS, NIDS or the sensors do not support the IDMEF format, then the native alerts format are converted in IDMEF format, store these IDMEF alerts in a central database, and perform analysis on the data both with native SQL queries and custom algorithms. We feel this commodity IDS solution suits the 'real world' concerns of many IDS users [1][2][3].

To implement the prototype of our Extensible IDS we thought to design it for Grid Computing environment. Grid computing make use of several computer connected in some way, to solve a large problem. The nodes on the grid are heterogeneous. The computers that are part of a grid can run different operating systems and have different hardware. This heterogeneous computing environment gives the most realistic situation for our Extensible IDS prototype testing and implementation. The individual nodes running with different types of OS and IDS generates different type of alerts.
in native format. These alerts need to be converted in IDMEF format and store these IDMEF alerts in a central database to perform analysis and alert correlation. Alert correlation is important to detect intrusion in grid environment as sometime alert form single node may or may not be real attack. The details of prototype implementation for Grid Computing Environment is discussed in next section [10][11].

A. Extensible IDS Implementation for Grid Computing Environment

The prototype of Extensible IDS for Grid Computing (Figure 1) is composed of a small cluster, Host Based Intrusion Detection System (HIDS) configured on each nodes, Network Based Intrusion Detection System (NIDS), the Sun Grid Engine (SGE), Globus Toolkit, The Prelude IDS and auxiliary software such as the Syslogger. The purpose of our prototype is to correlate the alert generated by the different IDS on different host or systems and detect the intrusion. As it’s briefly discussed in the above section that IDMEF is the standard format that enable interoperability among different types of IDS, we have implemented the prototype using the IDS and log analyzer which support IDMEF. The detail descriptions of these software are as follows:

1) The Cluster Environment

The testing environment is implemented in a small cluster, which consists of the cluster front-end and computational nodes. The front-end is a sort of gateway to the computing resources. All interaction between the user and the cluster occurs at the front-end. The programs running at the front-end receive the tasks or jobs from the user, put them in a queue, and match them with the most suitable available computational resources, i.e. the nodes. The front-end runs the server packets, while the nodes are equipped with the client-side packets. At the front-end, the configuration of nodes can be specified and installed automatically upon booting the system. Within the cluster, the home directories are shared over the Network File System (NFS). This set-up enables the nodes to access information physically located on the front-end. We configured the cluster environment for our prototype using Globus Toolkit and SGE. The cluster front-end runs the Sun Grid engine, responsible for scheduling of submitted jobs and matching them with available resources [4].

2) Host Based Intrusion Detection System (HIDS)

The host-based IDS looks for signs of intrusion on the local host system. These frequently use the host system’s audit and logging mechanism as a source of information for analysis. They look for unusual activity that is confined to the local host such as logins, improper file access, unapproved privilege escalation, or alterations on system privileges. In our prototype, we are having three nodes and each nodes are configured with different types of HIDS. The detail of these HIDS are as follows:

i. **OSSEC**: It’s an Open Source Host-based Intrusion Detection System that performs log analysis, file integrity checking, policy monitoring, rootkit detection, real-time alerting and active response. OSSEC is installed on Node 1 and Node 2.

ii. **SamHain**: The SamHain host-based intrusion detection system (HIDS) provides file integrity checking and log file monitoring/analysis, as well as rootkit detection, port monitoring, detection of rogue SUID executables, and hidden processes. SamHain is configured on Node 3.

3) **SNORT- Network Based Intrusion Detection System (NIDS)**

Snort’s open source network-based intrusion detection system (NIDS) has the ability to perform real-time traffic analysis and packet logging on Internet Protocol (IP) networks. Snort can be configured in three main modes: sniffer, packet logger, and network intrusion detection. In sniffer mode, the program will read network packets and display them on the console. In packet logger mode, the program will log packets to the disk. In intrusion detection mode, the program will monitor network traffic and analyse it against a ruleset defined by the user. The program will then perform a specific action based on what has been identified. The Snort sensor is set to monitor network traffic at the interface of the node [8][9].

4) **Sun Grid Engine**

SGE is typically used on a computer farm or high-performance computing (HPC) cluster and is responsible for accepting, scheduling, dispatching, and managing the remote and distributed execution of large numbers of standalone, parallel or interactive user jobs. It also manages and schedules the allocation of distributed resources such as processors, memory, disk space, and software licenses. The benefits of the SGE are effective utilization of computing resources and granting access to large computing resources by combining them in an efficient way. The SGE is installed on the front-end of the cluster in the set-up.

5) **Syslogger**

Syslogger is a standard for computer data logging. It allows separation of the software that generates messages from the system that stores them and the software that reports and analyzes them. It also provides devices which would otherwise be unable to communicate a means to notify administrators of problems or performance. Syslogger is installed on the cluster front-end. The Syslogger is capable of parsing data from log files. The configuration file of Syslogger can be modified to match software-specific log format, in order to extract data of interest. Syslogger parses data from log files of HIDS, NIDS and the SGE.
6) Prelude IDS

Prelude collects, archives, normalizes, sorts, aggregates, correlates and reports all security-related events independently of the product brand or license giving rise to such events. Prelude IDS uses IDMEF format. It's a good tool to get familiar with IDMEF format as all attribute values are visible from the Graphical User Interface. Prelude is capable of handling large number of connections, and processing large amounts of alerts. It uses per client scheduling queues in order to process alerts by severity fairly across clients. The Prelude Manager comes with multiple plugins like filtering plugins (idmef-criteria, thresholding, etc.) or reporting plugins like the SMTP plugin which automatically sends emails containing a textual description of alerts to a configured list of recipients. In our prototype we are using prelude to collect the alerts logs from all the HIDS and NIDS for correlation and generating the aggregate alert log.

B. Operating Details of the Grid IDS

With the components and their role explained, now we will explain the function of the whole system. A user submits a job to the SGE, which then matches its requirements with the available resources, and hands over the job to a suitable node. If the job initiates activity that triggers an alert by HIDS or NIDS, an alarm is generated by the respective IDS. If the log generated by the IDS in not in IDMEF format it is converted in IDMEF format using some plugins. The log is then transferred to the Prelude IDS for alert correlation.

The Prelude collects, normalizes, sorts, aggregates, correlates and generate the log in IDMEF format. The log is then picked up by the Syslogger, which resides on Alert Analyzer system. The Syslogger is configured to parse the relevant information from the Prelude logs. At the same time, the Syslogger accesses the SGE logs for information on the identity of the user, whose job is being executed. The identity of the user is retrieved by comparing the IP of the source of attack, provided by HIDS or NIDS, and by looking for a user ID which is executing a job on a resource that has the same IP as that provided by SGE.

The alert contains information on the conditions that triggered the alert and the identity of the corresponding user. In a real-life Grid IDS, tracking the identity of the user responsible for suspicious activity is most important. When the user responsible for the suspicious activity is identified, suitable actions can be undertaken, such as imposing sanctions and restrictions on the user’s subsequent rights and access to the resources. This course of action is taken in order to restrain the threat from escalating any further.

C. Results

The prototype is tested with a misuse case scenario of a user submitting a malicious resource scanning job. All misuse case scenarios revolve around illicit scanning of cluster resources, as this is considered a first step in executing many different types of attacks.

Several misuse cases have been devised in order to test the performance of the prototype. The misuse cases are a part of a more general threat scenario, which is described as follows. A user submits a job to the cluster front-end. The job itself is a normal Bash shell script, testjobs.sh, containing executable instructions written by the user and several variables containing information on the location of files and directories of interest. The script is submitted to the cluster front-end.

After submission, the job is executed on suitable resources identified by the SGE. The submitted job in this case has malicious content, which consists of running the nmap program. This program is a popular network scanning tool, and can be used to obtain information such as the IP addresses of online hosts, their operating systems, the services they run, and open ports. Scanning of resources does not as such cause any harm with normal use of scanning tools, but the obtained information can be used to help discover vulnerabilities in the underlying system.
Once the resources are scanned, the results are written to a plain text file, which is accessible by the malicious user. The file contains results of the nmap resource scan, and is stored in the user’s account. The file contains data on the underlying resources of the cluster, in this case the IP addresses of the online hosts and the corresponding ports and services.

Gathering the type of information on computing resources, as discussed above, is a part of the so-called reconnaissance phase of a forthcoming attack, and thus scanning activity should not be ignored. Once basic information on resources is obtained by the attacker, it can be used to carry out a more detailed resource scan. The discovered vulnerabilities can then be used to help gain control of resources. Captured resources can, for instance, be used for launching a massive DoS attack against public or private organizations.

The threat scenario described above is subdivided into three distinctive cases, and these are discussed in more detail in the following text.

**Threat Scenario 1: Scanning Hostile Job**

In this scenario, a single user with authorized access to the resources submits a job. The content of the job is defined as malicious, with the HIDS, residing on one of the computing nodes, tuned to detect the scanning activity that takes place as the commands contained in the job script are executed. The goals, requirements and methods related to the detection of events of the first case scenario are presented in Table 1. It should be noted that the definition of the time window in the formal notation requires that the clocks of the subsequent systems are synchronized. If this is not the case, the length of the time window based on subtraction of the time of occurrence of events is of little utility.

<table>
<thead>
<tr>
<th>Goal</th>
<th>To detect a malicious scan and resolve the user identity behind this activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Capability to detect a scan activity and finding the user responsible for it.</td>
</tr>
<tr>
<td>Method</td>
<td>Detection using Prelude logs and SGE logs.</td>
</tr>
<tr>
<td>Alert Condition</td>
<td>If within a time window, an attack is detected from a IP address and this IP is also found in the SGE logs, an alert results.</td>
</tr>
</tbody>
</table>

**Threat Scenario 2: Scanning Non-Hostile Job**

The second case scenario is that of an administrator of the cluster (e.g. root user) scanning the cluster resources for a legitimate reason, such as a part of a normal maintenance task. In such case, no alert should result, it thus needs to be defined that scanning of the cluster resources is interpreted as a malicious action, unless this action is carried out by an instance possessing administrator privileges. The goals, requirements and methods related to the detection of events of the first case scenario are presented in Table 2.

<table>
<thead>
<tr>
<th>Goal</th>
<th>To decrease the false positive alerts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Capability to ignore the scan activity when carried by the root</td>
</tr>
<tr>
<td>Method</td>
<td>Detection using Prelude logs and SGE logs.</td>
</tr>
<tr>
<td>Alert Condition</td>
<td>If within a time window, an attack is detected from a IP address and this IP correspond to root of the grid, do not make any alert.</td>
</tr>
</tbody>
</table>

**Threat Scenario 3: Scanning Mix Jobs**

The third case scenario is a generalization of the first case scenario, extending of the original problem to the case of multiple users with any number of them possibly having malicious intent. It is the most complex and at the same time the most realistic of the threat scenarios. In order to make a correct match between suspicious network traffic within the cluster, and the user ID behind this suspicious action, more sophisticated correlation rules need to be implemented. For such a match to be possible, additional information on Grid activity is needed. One approach would be to look for any connections between a job being executed, the related network activities, and any other related log files of the underlying system. Since a job and its results can be traced to a user once completed, it would be logical to infer that this is also true during the intermediate steps of the execution of the job. The concrete approach could thus be to inspect related program log files and system log files for useful information. Once such information is identified, it can then be used together with detected intrusion activity, in order to connect suspicious activity to its originator(s). The goals, requirements and methods related to the detection of events of the first case scenario are presented in Table 3.

<table>
<thead>
<tr>
<th>Goal</th>
<th>To detect a malicious activity among multiple jobs submitted by multiple users</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements</td>
<td>Capability to ignore the scan activity when carried by the root</td>
</tr>
</tbody>
</table>
Method Detection using Prelude logs and SGE logs.
Alert Condition Mix of alert condition given in above Table 1 and Table 2

The implemented Extensible IDS prototype was able to successfully detect the above mentioned threat scenario. The objective of our implementation was to collect the alerts from different types of IDS running on different nodes. Then the native alerts format are converted in IDMEF format and passed it to the central database for alert correlation, in our prototype the alert correlation is done by Prelude.

In our other experiment with the same prototype implementation we wanted to analyze that what’s the benefit of using IDS alert log in IDMEF format in terms of size of alerts log. It’s one of the important analysis as in distributed environment or in large network the IDS and sensors are installed on multiple hosts and the alerts log generated by these IDS and sensors are communicated to central database for alert correlation. If the size (storage size) of these alerts are big it will consume more bandwidth of the communication network. In our experiment we collected the 1000 alerts generated by OSSEC, SamHain and Snort in the native format as well as in IDMEF format. The details are shown in the Table 4. The experimental analysis shows that the size of alert logs generated in IDMEF format is on average 2.5 times (Figure 2) less than the logs generated in native format. This clearly shows that such kind of Extensible IDS architecture which uses the log’s generated in IDMEF format also save the bandwidth and feasible for realistic implementation.

| TABLE IV: Alert Log Size Generated by Different IDS |
|---------------------------------|--------|--------|--------|
|                                | OSSEC  | SamHain| SNORT  |
| Native Format                  | 5.5 MB | 6.32 MB| 4.88 MB|
| IDMEF Format                   | 1.99 MB| 2.42 MB| 1.95 MB|

Fig 2: - Analysis of Alert Log Size Generated by Different IDS

IV. CONCLUSION

Interoperability of different type of Intrusion Detection System and alerts correlation has become one of the important issues. In today’s complex scenario of Internet and networks, it’s practically impossible for the single IDS to fulfill the requirement. To cope up with such complex network environment, integration of multiple IDS is the only feasible solution. We have achieved successful development of such a prototype to support IDS interoperability using IDMEF.

REFERENCES


AUTHORS PROFILE

Manish Kumar is working as Assistant Professor in the Department of Computer Applications, M. S. Ramaiah Institute of Technology, Bangalore, India. He is pursuing his PhD from Bangalore University, Bangalore. His specialization is in Network and Information Security. He has worked on the R&D projects related on theoretical and practical issues about a conceptual framework for E-Mail, Web site and Cell Phone tracking, which could assist in curbing misuse of Information Technology and Cyber Crime. He has published many research papers in National, International Conferences and Journals. He is also the active member of various professional societies.

Dr. M Hanumanthappa is currently working as Professor in the Department of Computer Science and Applications, Bangalore University, Bangalore, India. He has over 15 years of teaching (Post Graduate) as well as Industry experience. He is member of Board of Studies /Board of Examiners for various Universities in Karnataka, India. He is actively involved in the funded research project and guiding research scholars in the field of Data Mining and Network Security. He has published many research papers in National, International Conferences and Journals. He is also the active member of various professional societies.