

A Multi-Layered Approach to the Design of Intelligent Intrusion Detection and Prevention System (IIDPS)

*Oludele Awodele, Sunday Idowu,
Omotola Anjorin, and Vincent J. Joshua
Babcock University, Ilishan-Remo, Ogun State, Nigeria*

delealways@yahoo.com; saidowu06@hotmail.com;
anjorin_jnr@hotmail.com

Abstract

Ignoring security threats can have serious consequences; therefore host machines in network must continually be monitored for intrusions since they are the final endpoint of any network.

As a result, this paper presents an Intelligent Intrusion Detection and Prevention System (IIDPS), which monitors a single host system from three different layers; files analyzer, system resource and connection layers. The approach introduced, a multi-layered approach, in which each layer harnesses both aspects of existing approach, signature and anomaly approaches, to achieve a better detection and prevention capabilities. The design of IIDPS consist of three basic components; the iExecutive which is an agent that runs in the background, iBaseline which is a database that stores the signatures of intrusions and the iManager which is a user Interface that serves as an intermediary between the IIDPS and the user.

This work serves as a foundation upon which interested researchers can further build on to achieve better detection and prevention capabilities.

Keywords: Intrusion, anomaly, signature, TCP/IP, ICMP

Introduction

Considering the reliance of humans on computers and network infrastructures to perform virtually every aspect of day to day activity, there is a critical need for ensuring the reliability and integrity of these infrastructures. According to the National Institute of Standards and Technology, Intrusion is an attempt to compromise the confidentiality, integrity, availability or an attempt to bypass the security mechanisms of a computer or network (Jones & Sielkens, 2000). The reasons for

Material published as part of this publication, either on-line or in print, is copyrighted by the Informing Science Institute. Permission to make digital or paper copy of part or all of these works for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage AND that copies 1) bear this notice in full and 2) give the full citation on the first page. It is permissible to abstract these works so long as credit is given. To copy in all other cases or to republish or to post on a server or to redistribute to lists requires specific permission and payment of a fee. Contact Publisher@InformingScience.org to request redistribution permission.

these intrusions could be attempts to steal a company's most valuable information, personal employee and customer information or to use the company's computer resources, etc. For example, the 2003 CSI/FBI (Computer Security Institute/ Federal Bureau of Investigation) Computer Crime and Security Survey reported that participants in the survey lost about \$135 million from the theft of proprietary information

and denial of service attacks (Cisco Systems, 2004). Recently, Intrusion Detection Systems (IDS) have been used in monitoring attempts to break security, which provides important information for timely countermeasures (Chen, Abraham, & Yang, 2007). Intrusion Detection System (IDS) implements application monitors in the form of a software program to learn and monitor the behavior of system programs in order to detect attacks against computer hosts. Existing IDSs are built with either signature-based or anomaly-based system,

Signature matching is based on a misuse model, this intrusion detection system detects intrusions by looking for activities that corresponds to known intrusion techniques(signatures)or system vulnerabilities while anomaly detection is based on a normal use model (Hwang, Cai, Chen, & Qin, 2007), they detect intrusion by looking for activities that is different from a user's or systems normal behavior. They may be classified into Host-based and Network-based according to the information used by each IDS. A Host-based IDS refers to the class of intrusion that resides on the monitor and the individual host machine, while A Network-based IDS monitors the packets that traverse a given network link (Jones & Sielken, 2000). The system proposed here is a type Host-based intrusion detection systems (HIDSs),these type of systems rely on events collected by the host they monitor .HIDSs can be classified based on the type of audit data they analyze or based on the techniques used to analyze their input. Common classes:

- operating system-level intrusion detection systems
- application-level intrusion detection systems

The system proposed here is an operating system-level intrusion system, because the OS is a trusted entity and it controls access to resources, such as memory and files.

Overview of Existing Systems

From the literature, there are various works in the field of intrusion detection system. This paper reviews those that are closely related to the proposed work based on the Anomaly and Signature detection approaches and the combination of both. In order to ascertain the efficiency of the new approach, a comparison is drawn between the existing work that have used the combination of both approaches and the new system which shows further improvement over the existing ones.

Adaptable Real-time Misuse Detection System (ARMD) (1998) is a host-based misuse detection system. Its pattern of signatures is over a sequence of abstract events and this is tagged 'MuSig'. This describes conditions that the abstract event attributes must satisfy. Based on the signatures (MuSigs), the available audit trail, and the strategy costs, ARMD uses a strategy generator to automatically generate monitoring strategies to govern the misuse detection process.

It employs database query optimization techniques to speed up the processing of audit events. One advantage of ARMD is that knowing the characteristics of the audit trail helps estimate the cost of performing misuse detection and this gives the security officers the opportunity to tune the misuse detection system.

A limitation of ARMD is that it requires users to have a precise understanding of the attacks and to make careful plans for the abstraction of events. This planning is not an easy job, especially when a user does not know how his/her MuSigs may be used. In particular, unforeseen attacks may invalidate previously defined abstract events and MuSigs, thus forcing the redevelopment of some/all of the MuSigs (Peng & Sushil, 2003).

Primary Response 2.2 (2004) takes an innovative approach to application security, learning normal code paths taken during the execution of system calls, including local file access, and stepping in when it detects variations to prevent attacks. It can be installed and configured quickly and can be managed centrally via a web browser. Primary response consists of a management

server and “adaptive profiling” agents. The agents run on Windows or Solaris hosts, monitoring those servers and reporting back to the management box. It requires several days of “learning” before the agent can establish a baseline of normal usage (Givnens & Herring, 2004). Protection against buffer overflow attacks, however, is provided right out of the box without any need for tuning.

It is easy to manage because it has granular options for blocking file access during an anomalous event and it has the ability to learn a server’s behavior on an incremental basis and to “readapt” after an OS is patched. Its downside is that it can only detect intrusions that are anomalous in nature since it was not integrated with a signature-based detection system (Garza & Roth, 2004).

The Computer Misuse Detection System (CMDS) (1998), CyberCop Monitor (CCM) (1999) and the new system, Intelligent Intrusion Detection and Prevention System (IIDPS) are all host based system. CMDS and CCM were implemented using both signature and anomaly detection methodologies. IIDPS is a new system introduced in this paper which also uses both aspects of misuse and anomaly detection approach but the edge it has over these two is in the use of a new approach, a multi layered approach. IIDPS uses multi layered approach to create a new design such that the each layer harnesses both aspects of anomaly and misuse detection approaches. The layers involved in this design are the File-Analyzer, the Connection layer and the System Resources Layer. The approach ensures detection and prevention at each layer and this result into lower false positives and detection of new attacks.

Comparisons between CMDS, CCM and IIDPS

This section discusses the comparison between the proposed system (IIDPS) and existing works that are closely related to the proposed system.

Protection

Both **CMDS** and **IIDPS** detect DOS attacks **CMDS** requires user ID and Password to start the console while **IIDPS** needs the Operating System UserName and Password to start the console.

Components

CMDS uses a GUI console, a manager and an agent, while the **CCM** uses a command console, a centralized database and an agent and the **IIDPS** uses a Window GUI, an agent and a Database.

Responses

Options available in **CMDS** are ‘ignore’, ‘increase observation’, ‘deny’, access to the perpetrator and ‘emergency shutdown of the target’, in **CCM** options available are terminate’, ‘shut down’, and ‘block’, while in **IIDPS** options available are ‘Allow’, ‘Block’, and ‘Ignore’. **CMDS** does not respond to pings while **IIDPS** responds to pings. Both **CMDS** and **IIDPS** raise alarms and warnings to the GUI in real time.

Reports

CMDS uses graphical trending reports from profiles, warning and alerts, **CCM** uses Text based and Graphical reports through SMI (Security Management Interface), while **IIDPS** uses text-based for all the layers and graphical trending reports for only the System resource analyzer.

Detection Mechanism

Both systems use both the Anomaly Detection (Statistical Deviation) and the Misuse Detection (Signature Recognition).

Mode of Operation

CMDS supports Real Time, Batch and On-Demand while **IIDPS** supports both real time and batch. Both **CMDS** and **CCM** support a distributed architecture while **IIDPS** runs on a single host.

Customization

CMDS requires the developers' professional services while **IIDPS** and **CCM** allows for custom settings and thresholds.

Functionality

CMDS agents collect, compress, encrypt, and transfer entire audit logs to the manager, **CCM** command console and agent collect, and analyze security status from each work station, while **IIDPS** collects, analyze and transfer entire processed data in real time to the database. **CMDS** Manager consolidates and analyzes this information for misuse, **CCM** database stores data from remote machines, while in **IIDPS** the database stores the information for the windows GUI.

CMDS GUI with ad hoc data analysis, reporting, and charting tools, and can correlate event from a number of managers, **CCM** Network Associate SMI (Security Management Interface) allows for the viewing of the data, while **IIDPS** GUI picks the information in real time form the Database.

Architecture of the Proposed System

Figure 1 explicitly shows the layers involved in the design of intelligent intrusion detection prevention system (IIDPS). Each of these layers is described briefly with the corresponding data flow for each layer.

Layers of the Proposed System

File-Analyzer

Most operating systems provide a file system, as a file system is an integral part of any modern operating system. The interface provided by an operating system between the user and the file system can be textual (such as provided by a command line interface, such as the Unix shell) or graphical (such as provided by a graphical user interface in Windows, such as file browsers). The graphical interface includes folders, containing documents and other files, and nested folders (Microsoft Encarta Encyclopedia, 2007). This layer, the File-Analyzer monitor is directed towards monitoring the files and folders on a host server that could be of interest to any intruder and this is determined to a large extent by the administrator. The administrator decides on the important files or directories to monitor so that the IIDPS does not monitor all the files and folders on the machine as this would cause a large overhead on the system resources.

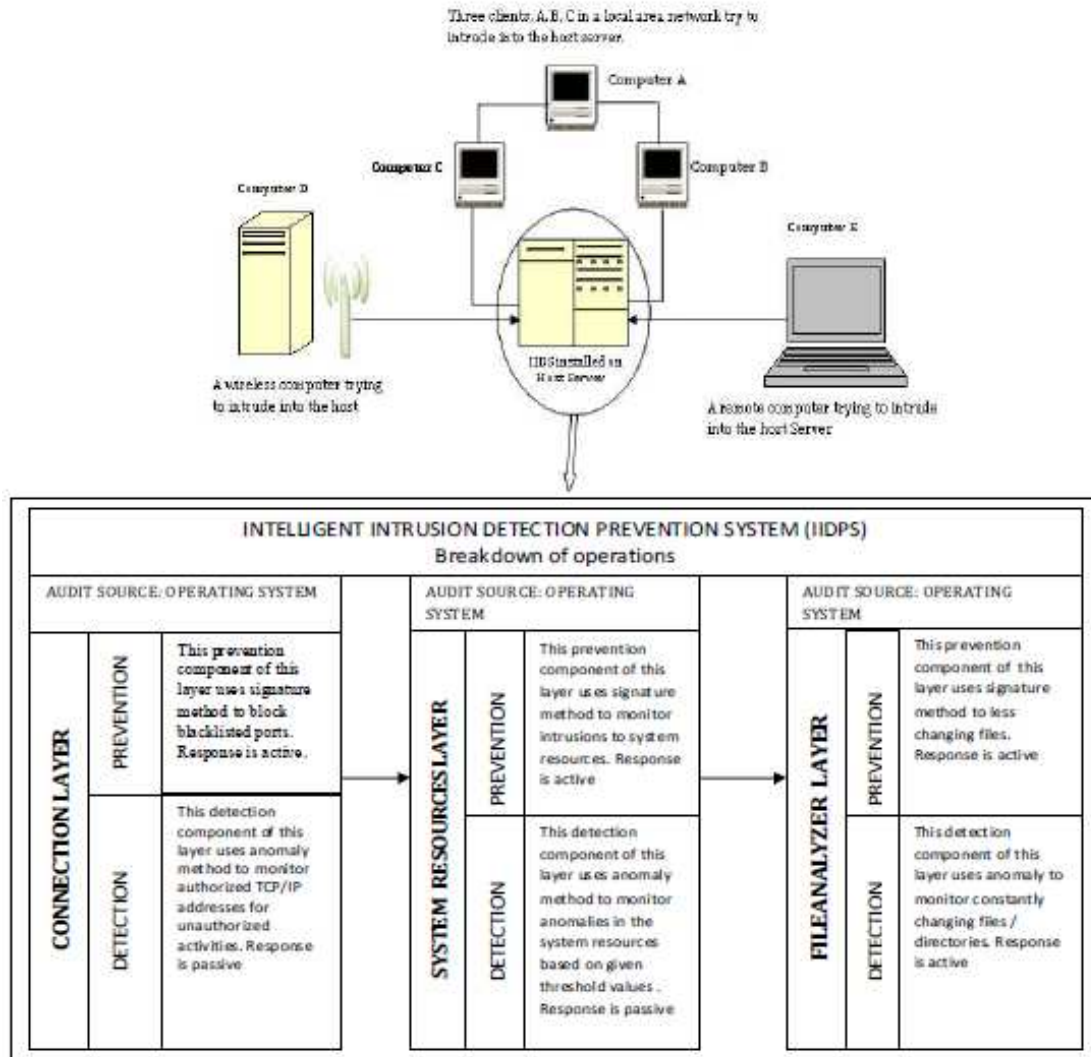


Figure 1. Architectural Design of Intelligent Intrusion Detection Prevention System

Figure 2 represents the flow of data from an external entity (User) at one end to another external entity (User) at the other end. Data flows from the user who relates with the system by specifying checks and threshold values on different files to be monitored to the file-analyzer layer of the system. The layer creates signatures for the checks specified by the user and stores both signatures and threshold values in the database. The agent collects this information for analysis to determine if the signatures match the incoming events or if the incoming events have exceeded the defined threshold values. If the agent flags an alerts to the threshold values being exceeded, the system respond by allowing the user to choose from an option of either allowing further modification or reject further modifications to the files. Otherwise, if the agent flags an alert to the matching of signatures, then the system respond automatically by restoring the file to its initial state. The intrusions flagged are stored in the database and a text based report is sent to the user through the user interface (iManager).

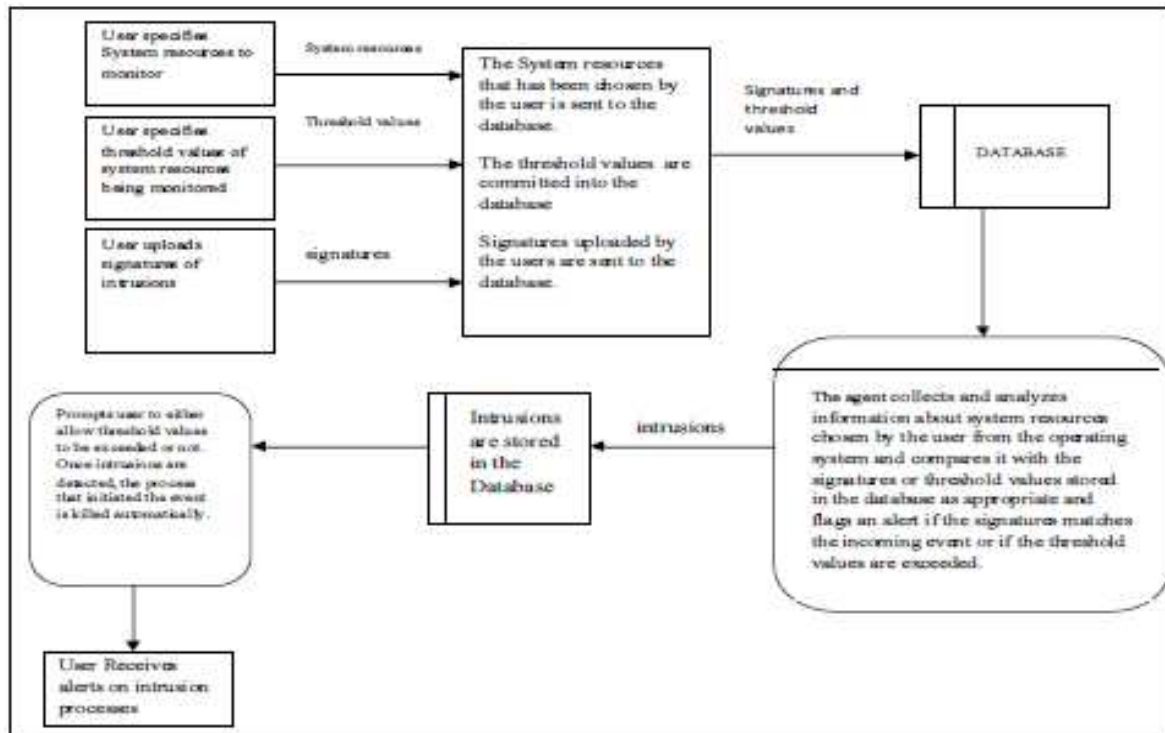


Figure 2. Data Flow Diagram for Layer 1: FILE – ANALYZER

System Resources

Programs often need certain system resources such as memory or disk space, and they request them as needed from the operating system (Microsoft Encarta Encyclopedia, 2007).

This layer is designed to periodically scan through the system log for latest entries and compare with the threshold value that must have been defined by the user. This enables the system resources layer to detect intrusions on system resources.

Figure 3 represent the flow of data from an external entity (User) at one end to another external entity (User) at the other end. Data flows from the user who relates with the system by specifying the system resource to be monitored, specify the threshold values and upload signatures for intrusions into the system. This layer sends this information (the threshold values and the signatures) to the database. The agent collects this information from the database for analysis to determine if the signatures in the database match the incoming events or if the incoming events have exceeded the defined threshold values for the system resource in question. If the agent flags an alerts to the threshold values being exceeded, the system responds by alerting the user through text – based messages. Otherwise, if the agent flags an alert to the matching of signatures, then the system respond automatically by killing the process that initiated the incoming event. The intrusions flagged are stored in the database and a text based and graphical based report is sent to the user through the user interface (iManager).

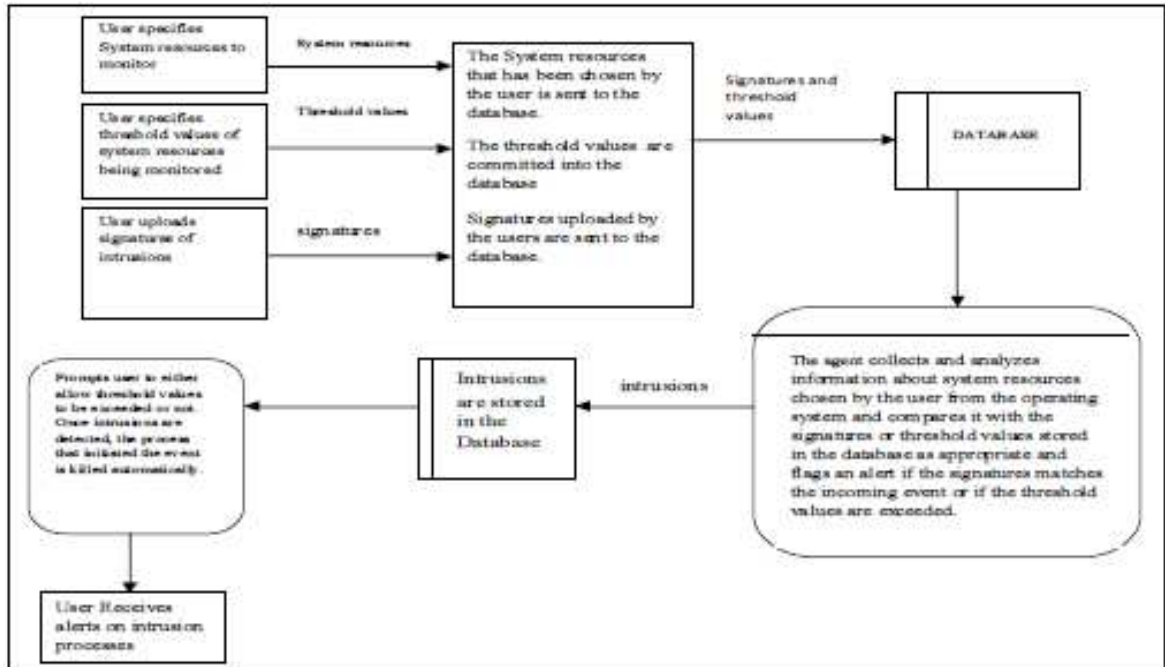


Figure 3. Data Flow Diagram for Layer 2: SYSTEM RESOURCES

Connection Analyzer

Connection can be described as a link between two entities which could either be physical or logical; physical connections let computers directly transmit and receive signals while logical or virtual connections allow computer applications, such as e-mail programs and the browsers explore the World Wide Web, to exchange information. Physical connections are defined by the medium used to carry the signal, the geometric arrangement of the computers (topology), and the method used to share information. Logical connections are created by network protocols and allow data sharing between applications on different types of computers. Some logical connections use client-server application software and are primarily for file and printer sharing. The Transmission Control Protocol/Internet Protocol (TCP/IP) suite, originally developed by the United States Department of Defense, is the set of logical connections used by the Internet, the world wide consortium of computer networks (Microsoft Encarta Encyclopedia, 2007). Connection layer monitors all of such connections made to a host machine.

Figure 4 represents the flow of data from an external entity (User) at one end to another external entity (User) at the other end. Data flows from the user who relates with the system by specifying a blacklist of TCP/IP addresses (i.e. TCP/IP addresses that are not allowed to connect to the host at all) and a list of authorized TCP/IP addresses. The layer creates signatures for blocking all the blacklisted computers specified by the user such that connection from these computers cannot be established. The layer also creates a baseline for the normal activities of the permitted TCP/IP addresses. The agent collects this information and analyzes all connection attempting to connect to the host. If the agent matches the signatures with a TCP/IP address, such a computer is blocked from establishing a connection. If the agent flags an alert to the deviation of the normal baseline of authorized TCP/IP addresses, the system flags an alert and sends a text based message to the user who will take further actions. The intrusions flagged are stored in the database and a text based report is sent to the user through the user interface (iManager).

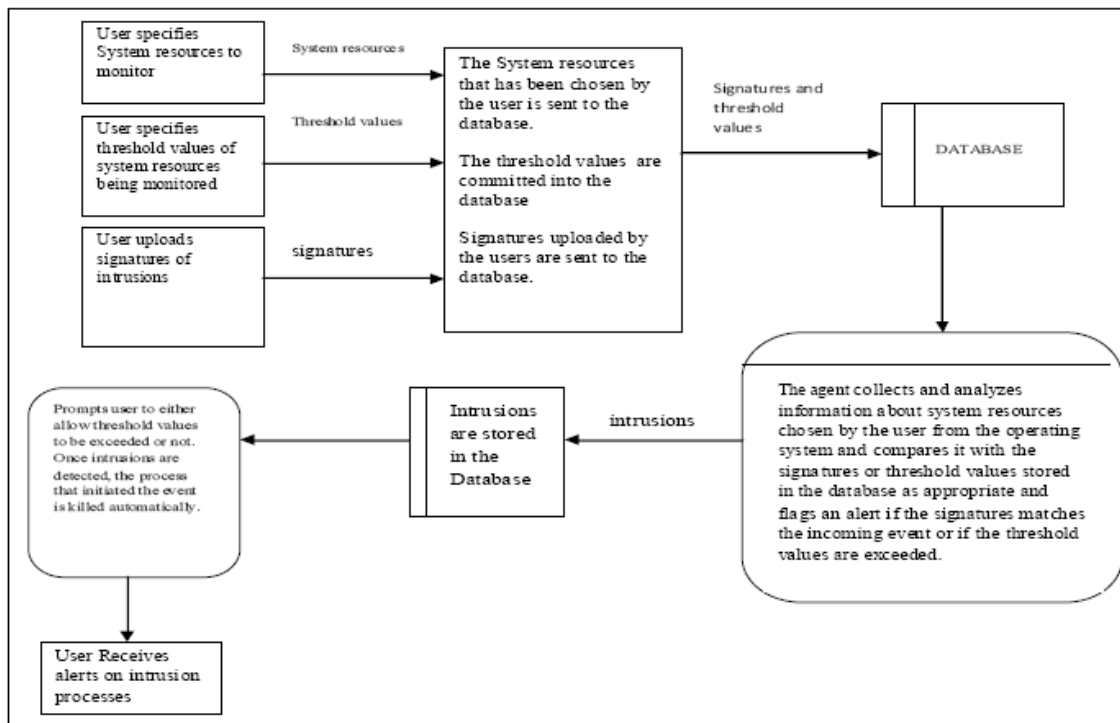


Figure 4. Data Flow Diagram for Layer 3: CONNECTION ANALYZER

System Components

In order to adequately implement and harness these layers as described in the previous section, three basic components are needed in the implementation of these layers. These components are: a window service known as “iExecutive”, a database called the “iBaseline” and a user interface called the “iManager”. Each of these components is further analyzed below:

iExecutive

iExecutive is the core component of IIDPS. It is an agent which is a long-running executable that runs at the background without a users’ intervention. It monitors the files at the file-analyzer layer, system resources at the system resources layer and connections to the host at the connection layer. Usually, iExecutive starts when the operating system is booted and runs in the background as long as the OS is running, though it can be configured to run manually. Once the iExecutive is installed, it can be managed by the user through the iManager (the user Interface).

iManager

iManager is the Graphical User Interface (GUI) that aids interaction between the computer and the user. The input and the output designs for each layer are discussed below.

Input Design

In this section, the input designs of each layer are discussed.

File Analyzer

The monitor definition interface for the File Analyzer consists of three panes as seen in Figure 5; the first pane displays the contents of all the drives such that the user can specify which files or directories to be monitored for protection, the second pane consists of options of several checks that can be performed on the selected file or directory (Signature) and the third pane consists of options for the user to set the maximum and minimum size and the units in which the file/folder is not meant to exceed (Anomaly).

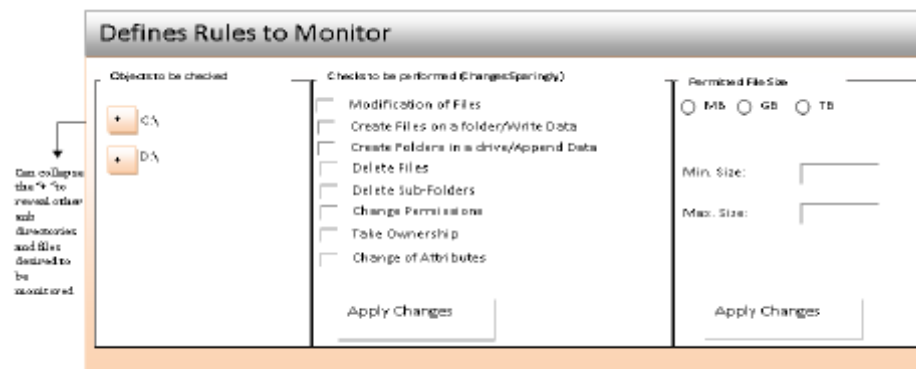


Figure 5. Monitor Definitions for File Analyzer

System Resource Analyzer

Figure 6 shows the design for the monitor definition of the System Resource Analyzer. This enables the user to choose the kind of system property (i.e. CPU usage, memory usage and disk usage) to monitor such that the specified resource does not fall below the threshold value defined by the user and also the opportunity to upload new signature for the System Resource Layer.

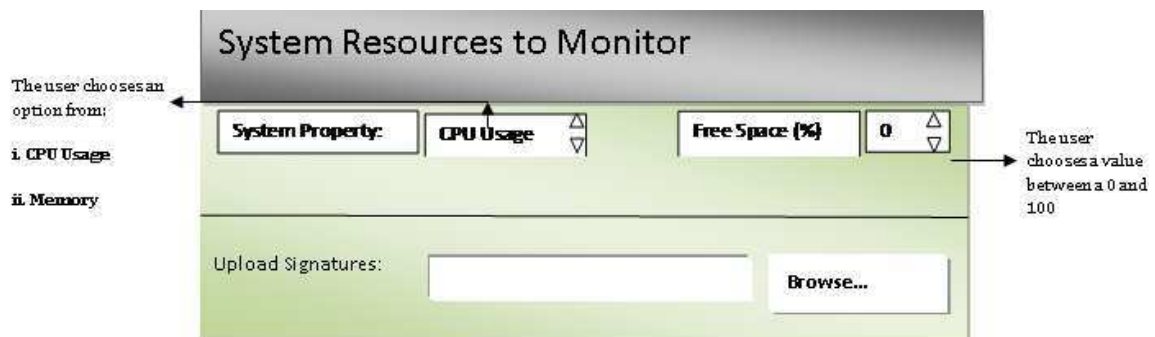


Figure 6. Monitor definitions for System Resource Analyzer

Connection Analyzer

This signature part of the layer has a user input interface where a list of black- listed IP addresses is given, once any of them is trying to connect the IIDPS blocks the IP address from connecting. The anomaly part of the layer have a user input interface where a list of authorized IP addresses is given, once an abnormality is detected from an authorized IP address it flags an alert. (See Figure 7.)

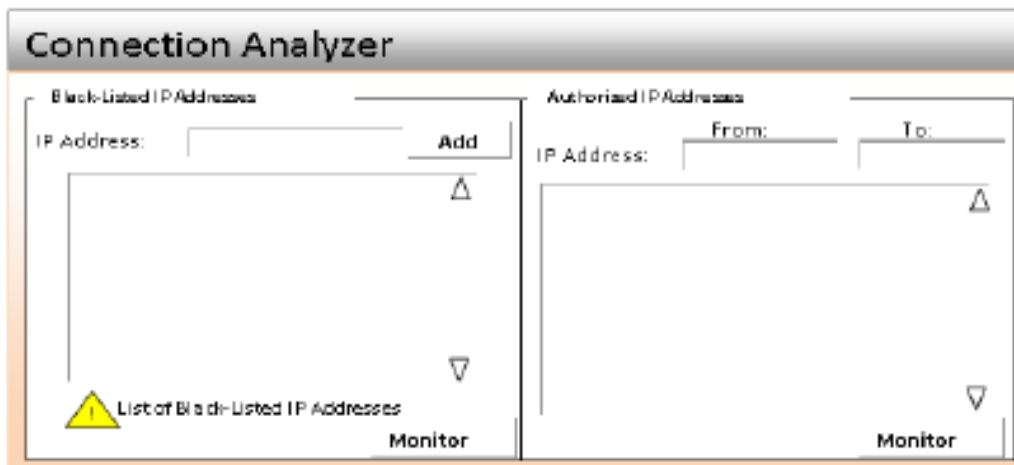


Figure 7. Monitor Definitions for Connection Analyzer

Output Design

The output basically consists of alert boxes, tabular reports on alerts, graphical reports and summaries of various alerts run on specific date range. The output designs are constructed per layer basis for clarity.

File Analyzer

Figure 8 represents the output design which enables the administrator receive the details of intrusions on the integrity of files and directories previously specified for monitoring.

Object Access Alert Report					
S/N	Message	Action Performed	Time	Day	User

Figure 8. A Tabular report for flagged alerts at the File Analyzer Layer

S/N: this represents the number of alerts that has been generated.

Message: This represents the message logged for any occurrence of an event and this is used by the file - Analyzer to notify the user of details of an alert.

Action Performed: this shows the actions taken by the user in regard to the alerts been made

Time: specifies the time when an event triggers an alert.

Day: specifies the day of the week when an event triggers an alert.

User: This specifies the user that initiated an event of a particular category which resulted in an intrusion.

System Resource analyzer

Figure 9 represents the output design which enables the administrator receive the details of anomaly behavior that deviates from the pre-defined threshold values specified on each system resource.



Figure 9. A Tabular report for flagged alerts at the System Resource Analyzer

Figure 10 represents the output design which enables the administrator to receive the details in graphical form of the system resources.

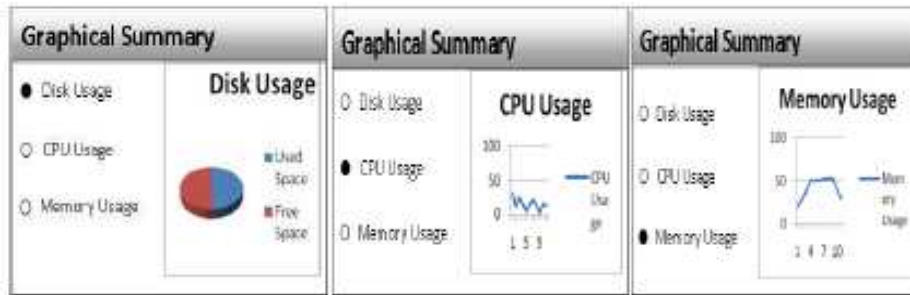


Figure 10. A Graphical report on Disk Usage, CPU Usage and Memory Usage

Connection analyzer

Figure 11 represents the output design which enables the administrator receive reports of pingging activities



Figure 11. A Tabular report for pingging activities at the Connection Analyzer Layer

Figure 12 represents an alert box which pops up whenever there is a port scan activity on the Host system.



Figure 12. A pop – up message to alert a user of a port scan activity

Figure 13 represents the output display for the user to alert him/her about the portscan activities on the machine.

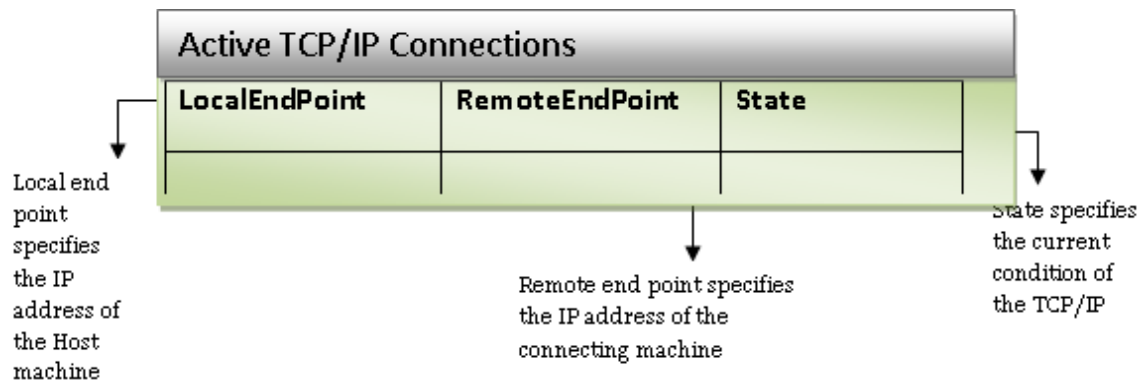


Figure 13. A Tabular report for port scan activities at the connection analyzer Layer

***i*BASELINE**

iBaseline is the component that collects data for the IIDPS (Intelligent Intrusion Detection Prevention System) and manages the data such that data can be easily retrieved and manipulated. It is a relational database in which data are stored in tables - rows and columns of data, where these tables are related by means of keys – primary key and foreign key. A primary key is a column or columns in a table whose values uniquely identify each row (record) in a table. A foreign key is a column or columns whose values are the same as the primary key of another table. The relationship is made between two tables by matching the values of the primary key in one table with the value of the primary key in the other.

The design of iBaseline consists of the specifications of the various tables, fields contained in the tables and their corresponding data types.

Tables specifying the data elements are described below :

File Analyzer Layer

a. Signature

Table 1 shows the various data elements used in the signature component of the layer. The functions are described below :

- **FileSystemID:** this is a particular ID assigned to a particular file.
- **FName:** This is the particular path name of the object.
- **Type:** This stores the type of intrusion that is occurring
- **Permissions:** This store the changes made in the permissions of a file or directory
- **Owner:** This stores change in the owner or group of a file or directory.
- **CreationTime:** This store the time when a new file or directory is created.
- **LastModifiedTime:** This stores the last time a particular file or directory was modified
- **LastAccessTime:** This stores the last time a particular file or directory was accessed
- **CheckSum:** This stores a change in the integrity of the file or directory.

FileSignatures
FileSystemID
FName
Type
Permissions
Owner
CreationTime
LastModifiedTime
LastAccessTime
Checksum

Table 1. Table for the Signature component of the File Analyzer

b. Anomaly

Table 2 shows the various data elements used in the anomaly component of this layer. The functions are described below :

- FileSizeID: this stores the unique ID of a particular file or directory
- FileName: this stores the file or directory address
- MinimumSize: this stores the minimum size in which the file or directory must not be lower than
- MaximumSize: this stores the maximum size in which the file or directory must not exceed.

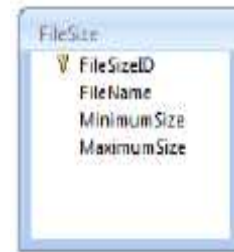


Table 2 Table for the File Analyzer (Signatures)

System Resource Analyzer

Anomaly

The database consists of various tables which stores user inputs and system output (intrusion) for the anomaly component of the system resource layer.

User Input Table

The tables in this layer are the Activities, Processor Usage, and Memory Usage table.

The relationship that exists between these tables is the one-to-many relationship as shown in Table 3.



Table 3. System Resource Analyzer input table

Activities

The Activities table consists of the system resources being monitored such as the processor usage, memory usage and disk usage:

- ActivityID (Primary Key): this stores the unique ID for each activity performed by the System Resource.
- Analyze Activity: this stores the specific system resource being monitored.

Processor Usage & Memory Usage

These tables stores threshold values input by the users for processor and memory usage.

- UsageID: this stores the unique ID
- PercentageUsed: this stores a percentage in which the system must not exceed.
- ActivityID (Foreign Key): this stores the ID of that activity

Output

The tables in this layer are the Memory Usage, Processor Usage, and Disk Usage as shown in Table 4. They store the intrusion details.

- Percent: stores the current percentage of the system after an intrusion has been discovered
- Time: stores the time in which the intrusion occur.
- Day: stores the date in which the intrusion occur

MemoryUsage	DiskUsage	ProcessorUsage
Percent	Percent	Percent
Time	Time	Time
Day	Day	Day

Table 4 System Resource Analyzer output table

Signature

Table 5 shows the various data elements used in the System Resource Analyzer (Signature). Their functions are described below:

- SystemSigID: this assigns and stores the unique ID for every new signature uploaded
- SystemSig: this holds the signature information.

SystemSignature
SystemSigID
SystemSig

Table 5 System Resource Analyzer (Signature) output table

Connection Analyzer

a. Signature

Table 6 shows the various data elements used in the signature component of the Connection layer. The functions are described below:

- BlackListedID: this assigns and stores the unique ID for each blacklisted IP Address
- IPAddress: stores blacklisted IP addresses for easier retrieval

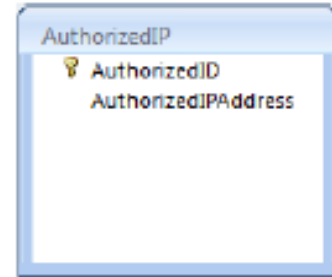
BlackListedIP
BlackListedID
IPAddress

Table 6 Connection Analyzer (Signature) table

b. Anomaly

Table 7 shows the various data elements used in the Anomaly component of the Connection layer. The functions are described below:

- Authorize id: this assigns and stores the unique ID for each Authorized IP Address
- Authorized IP Address: stores authorized IP addresses for easy retrieval



AuthorizedIP	
AuthorizedID	AuthorizedIPAddress

Table 7 Connection Analyzer (Signature) table

Mode of Operation

The iExecutive and the iManager will be installed on the host intended to be monitored while the iBaseline configured to the correct path. The user identifies the files that change sparingly and those that change constantly. He/she specifies these files on the iManager and choose appropriate checks to be performed if the file changes sparingly and specifies the threshold values for the those files that are constantly changing. The user can check the dashboard for summary report. The user can also specify the system resources to be monitored as well as updating the signatures of the system resources through the iManager. The user can also blacklist some IP addresses that will not be allowed to connect to the host while he/she also specifies the list of authorized IP addresses. All report can be viewed on the iManager.

Conclusion

The field of Intrusion Detection System is a vast field where lots of interest is being shown.

Existing works were reviewed which gave an insight to the functionality of a typical intrusion detection system. This work introduced a new approach, a multi – layered approach to the design of Intrusion Detection and Prevention System which comprised of three distinct layers; the file analyzer, system resources and connection layers. Each of these layers harnessed both aspects of the basic detection methodologies; anomaly and signature. The work incorporates prevention and detection capabilities into the design of each layer of the Intelligent Intrusion Detection and Prevention System. The IIDPS is software which can be installed on any single Host for detection of unwanted activities such as tampering with important files, unauthorized connection, unauthorized elevation of permissions etc.

Future Work

The present work was implemented on a single host; the implementation can be further extended on multiple hosts such that the agent (i.e. iExecutive) is installed on each host and the iManager is able to correlate all information from each of these agents while the iBaseline (the database) is maintained in such a way that it does not become full and ‘hang’. This system can however be further extended such that the iExecutive and the iManager can be installed on several hosts which then connects to system that contains this iBaseline. This would involve using a better database like MySQL, MS SQL Server and other database products that supports multiple users and concurrent connections. This way issues like the iBaseline hanging will be eradicated and the iExecutive and iManager can also be installed on other operating systems, as long as they can connect to the machine with the iBaseline.

References

- Chen, Y., Abraham, A., & Yang, B. (2007). Hybrid flexible neural-tree-based intrusion detection systems. *International Journal of Intelligent Systems*, 22.
- Cisco Systems. (2004). *Deploying host-based intrusion detection*.
- de Boer, P., & Pels, M. (2005). *Host-based intrusion detection systems*. Retrieved from <http://staff.science.uva.nl/~delaat/snb-2004-2005/p19/presentation.pdf>
- Garza, R. V., & Roth, L. J. (2004). *Intruders beware*. Info world.Com
- Hwang, K., Cai, M., Chen, Y., & Qin, M.. (2007). Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes. *IEEE Transactions on Dependable and Secure Computing*, 4(1).
- Jackson, A. K. (1999). Intrusion detection system (IDS) product survey. Distributed Knowledge Systems Team, New Mexico, USA.
- Jones, K. A., & Sielken, S. R. (2000). *Computer system intrusion detection: A survey*. Department of Computer Science, University of Virginia, Thornton Hall.
- Microsoft Encarta Encyclopedia*. (2007). Microsoft Corporation.
- Peng, N., & Sushil, J. (2003). Intrusion detection techniques. *The Internet Encyclopedia*. John Wiley & Sons.

Biographies



Oludele Awodele Ph.D is presently a lecturer in the department of computer science & mathematics, Babcock University, Ilishan-Remo, Ogun State, Nigeria. His research areas are Software Engineering, Data Communication and Artificial Intelligence. He has published works in several journals of international repute. He can be contacted at deleal-ways@yahoo.com.



Sunday Idowu is a senior lecturer in the department of computer science & mathematics, Babcock University, Ilishan-Remo, Ogun State, Nigeria. He holds a Masters degree in Software Engineering, and he is currently working on his Ph.D in the University of Ibadan, Oyo State, Nigeria. His research areas are Software Engineering, Web Application Development and Security. He has published works in several journals of international repute. He can be contacted at saidowu06@hotmail.com.



Omotola Anjorin is a final year undergraduate student of Computer Information Systems at Babcock University, Ilishan-Remo, Ogun State, Nigeria. He is a member of the Nigerian Computer Society, and the International Institute of Electrical and Electronic Engineers (IEEE). His areas of interest include Database Technologies, Software Development, Systems Administration and Networking. He is a Microsoft Certified Professional, a Microsoft Certified Systems Administrator and a Cisco Certified Network Associate. He can be contacted at anjorin_jnr@hotmail.com.

Vincent J. Joshua, is a lecturer in the department of computer science & mathematics, Babcock University, Ilishan-Remo, Ogun State, Nigeria. He is a software engineer with many years of technical experience. His areas of research work include database management and object oriented programming. He can be contacted at jvjoshua@hotmail.com