Web-based Network Intrusion Detection Expert System

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Vamshi K Kankanala
Spring 2006

Committee Members

Dr. Mario Al. Garcia
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
Web-based Network Intrusion Detection Expert System

GRADUATE PROJECT TECHNICAL REPORT

Submitted to the Faculty of
Department of Computing and Mathematical Sciences
Texas A&M University-Corpus Christi
Corpus Christi, Texas

in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

by

Vamshi K Kankanala
Spring 2006

Committee Members

Dr. Mario Al. Garcia
Committee Chairperson

Dr. David Thomas
Committee Member

Dr. Long-zhuang Li
Committee Member
ABSTRACT

Intrusion Detection (ID) is a type of security management system for computers and networks. An ID system gathers and analyzes information from various areas within a computer or a network to identify possible security breaches, which include both intrusions (attacks from outside the organization) and misuse (attacks from within the organization). These systems are an important component of defensive measures to protect computer systems and networks from potential threats and vulnerabilities. Web-based Network Intrusion Detection Expert System (WNIDES) is implemented using Java Expert Shell System (JESS). The WNIDES expert system uses Snort to capture network packets; Snort is an open source network packet sniffer program. This technical report presents a literature survey of different types of intrusion detection systems and the various methods in developing them. It presents the design and implementation of a WNIDES using JESS.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>iii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>vii</td>
</tr>
<tr>
<td>1. INTRODUCTION AND BACKGROUND</td>
<td>1</td>
</tr>
<tr>
<td>1.1 Defining Intrusion Detection</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Importance of Intrusion detection System</td>
<td>1</td>
</tr>
<tr>
<td>1.3 Components of Intrusion Detection System</td>
<td>4</td>
</tr>
<tr>
<td>1.4 Previous Expert Systems</td>
<td>4</td>
</tr>
<tr>
<td>2. NARRATIVE</td>
<td>5</td>
</tr>
<tr>
<td>2.1 A Simple Expert System</td>
<td>5</td>
</tr>
<tr>
<td>2.1.1 Knowledge Base</td>
<td>6</td>
</tr>
<tr>
<td>2.1.2 Inference Engine</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3 User Interface</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Rule Based Expert System</td>
<td>7</td>
</tr>
<tr>
<td>2.2.1 Knowledge base</td>
<td>8</td>
</tr>
<tr>
<td>2.2.2 Working Memory</td>
<td>8</td>
</tr>
<tr>
<td>2.2.3 Inference Engine</td>
<td>9</td>
</tr>
<tr>
<td>2.2.4 Explanation Facility</td>
<td>9</td>
</tr>
<tr>
<td>2.2.5 User Interface</td>
<td>9</td>
</tr>
<tr>
<td>2.3 WNIDES as a Rule Based Expert System</td>
<td>9</td>
</tr>
<tr>
<td>2.3.1 Knowledge Engineering</td>
<td>11</td>
</tr>
<tr>
<td>2.3.2 Structuring Data</td>
<td>11</td>
</tr>
</tbody>
</table>
2.3.3 Developing Test Cases: ................................................................. 12
2.3.4 Interface Building: ................................................................. 12
2.3.5 Writing the rules: .............................................................. 12
2.3.6 Testing: .......................................................................... 13
2.3.7 Iterative Development: .................................................. 13

3. SYSTEM DESIGN .............................................................................. 15

3.1 System Requirements: ............................................................ 15
3.2 Java Expert System Shell (JESS) ................................................. 15
3.3 Visual Basic .NET 2003: .......................................................... 16
3.4 Snort: ................................................................................ 16
  3.4.1 Snort Rules: ................................................................. 16
3.5 Ethereal: ............................................................................. 17
3.6 WNIDES Architecture: .......................................................... 17
  3.6.1 Packet Capturing: ........................................................... 18
  3.6.2 Writing Rules: ............................................................... 20
  3.6.3 Writing Facts: ............................................................... 21
  3.6.4 Matching Rules and Facts: ............................................ 21
3.7 WNIDES User Interface ........................................................... 22
  3.7.1 WNIDES Main Page ......................................................... 22
  3.7.2 WNIDES JESS Documentation Page ................................. 23
  3.7.3 WNIDES Feedback Page ................................................ 24
  3.7.4 WNIDES Help Page ........................................................ 25
3.8 WNIDES Administrator Component ......................................... 26
LIST OF FIGURES

Figure 2.1 A Simple Expert System .................................................................5
Figure 2.2 Knowledge Acquisition .................................................................6
Figure 2.3 Inference Mechanism .................................................................7
Figure 2.4 A Typical Rule Based System .....................................................8
Figure 2.5 Development of Rule Based System .........................................10
Figure 3.1 Architecture of WNIDES .......................................................18
Figure 3.2 WNIDES Main Page .................................................................23
Figure 3.3 WNIDES JESS Documentation Page .........................................24
Figure 3.4 WNIDES Feedback Page ...........................................................25
Figure 3.5 WNIDES Help Page .................................................................26
Figure 3.6 WNIDES Rule Adder from Directory ........................................27
Figure 3.7 WNIDES Rule Adder from File ................................................27
Figure 3.8 WNIDES Single Rule Adder .......................................................28
Figure 4.1 Single Rule Test Result ...............................................................31
Figure 4.2 Single Domain Rule Test Result .............................................32
Figure 4.3 WNIDES Rule Database Test Result .......................................33
Figure 4.4 Stress Test result ..............................................................34
1. INTRODUCTION AND BACKGROUND

1.1 Defining Intrusion Detection

Intrusion can be defined as an action whose target is to compromise the availability, utility, integrity, authenticity, confidentiality and/or possession of an information system. It is an attempt to gain unauthorized access to a system with the purpose of either testing the security of the network or using the facility as a launching pad for further attacks on other systems. An Intrusion Detection System (IDS) is a tool designed to recognize unauthorized and malicious entry into a network or host, including monitoring for suspicious packet traffic, tracking intruders and identifying where the security hole is. [Sans 2004]

1.2 Importance of Intrusion detection System:

Intrusion Detection Systems (IDS) are very important in today’s computing environment because it is difficult to keep pace with the current and potential threats and vulnerabilities in our computing systems. The explosive growth of the Internet, and the pressure of more companies to be part of this development, opened a whole new aspect of network security. As the Internet grows exponentially, the potential damage for people who are not security conscious grows. Solutions to overcome such problems include firewall components such as packet filters and proxy firewalls. [Nibali 2004]

However, such solutions are no longer enough as firewalls cannot detect unintentional backdoors around the firewall. Statistics suggest that more than 50% of all recorded breaches today originate from someone legitimately behind the firewall. [Nibali
Therefore, people started deploying IDS as an additional part of a network's security architecture.

An intrusion has different goals some of them are listed below:

- Read protected information such as internal company data, credit cards numbers, financial records or password files;
- Change protected information such as changing file contents, deleting files.
- Use protected computer resources, CPU time, disk space, printer, audio-video tools.
- Denial of service.
- Use a computer’s identity. Use the computer as an agent to attack other systems behind a firewall.

The basic purpose of network based IDS is to sniff all traffic on a network and to compare the network packets with certain patterns [Cawsey 2005]. If a pattern matches, an alarm is raised.

The need for IDS depends on many aspects. Networks are big and complex and difficult to monitor. IDS can help reveal potential security problems of a network and helps in documenting it. IDS add the capability to verify the firewall configurations. In case of an incident, the probability of tracking down the attacker is increased by the chronological record of events generated by the IDS. With an intelligent IDS configuration, it is possible to trace strange occurrences by reconstructing the chain of events. This can be used as a proof of concept or to explain malicious traffic. IDS in contrast to a firewall, is a passive system which does not influence the network traffic. [Nibali 2004]
Some of the common intrusion methods are as follows [Network 2006]:

- **Crack**: This type of intrusions attempts to break a password file using brute force methods.

- **Packet Sniffers**: These are programs written by hackers. These programs sniff the packets that are transferred in an organization network. They are usually hard to detect. One detection procedure is to scan local machine for suspicious acts.

- **Machine and Services Discovery Utilities**: These tools are to discover all the running services on the network. This is usually done through sending ping packets asynchronously to multiple hosts and scanning ports of known services.

- **Packet Spoofing Utilities**: Again, these programs are also written by hackers. They change the content of the network packets. They are most effective on the network which they are executed. In order for them to work outside, some routing changes are needed for the local router.

- **Packet flooding Utilities**: These are programs written by hackers to exploit software bugs for denial of service attacks. This includes ping floods, port overloading and broadcast storms and depending on how, when and where they are applied, they can be difficult to detect.

Therefore, Intrusion Detection Systems are a must in today’s world to keep information on a system or network secure.
1.3 Components of Intrusion Detection System:

An IDS consists of a management console and sensors. Management console is the management and reporting module of the IDS. Sensors are agents that monitor hosts or networks on a real time basis for any network intrusions. IDS have a database of attack signatures. The attack signatures are patterns of different types of previously detected attacks.

If the sensors detect any malicious activity, it matches the malicious packet against the attack signature database. In case it finds a match, the sensor reports the malicious activity to the management console. The sensor can take different actions based on how they are configured. [SANS 2004]

1.4 Previous Expert Systems:

There are many expert systems available in today’s market. Most of the expert systems that are available are built using outdated software which cannot handle the growing requirements of the users. The main disadvantages of these systems are that they are on a standalone system. That is the expert system can be accessed only on a system on which it is installed. Hence a user or an organization needs to have the expert system on each system. This requires a lot of overhead on the organization.
2. NARRATIVE

The project deals with the implementation of a Web based expert system for Network Intrusion Detection. The project is implemented using the most suitable software available in today’s market to implement a Web based expert system called JESS.

2.1 A Simple Expert System:

An expert system is a computer program designed to simulate the problem-solving behavior of a human who is an expert in a narrow domain or discipline. An expert system is normally composed of a knowledge base (information, heuristics, etc.), inference engine (analyzes the knowledge base), and the end user interface (accepting inputs, generating outputs) as shown in figure 2.1 below [Generation5 2006]:

![Figure 2.1: A Simple Expert System](Generation5 2006)
2.1.1 Knowledge Base:

Building a knowledge base covers the process of collecting knowledge and transforming it into a form that can be processed by a computer as shown in figure 2.2 where the data collected is converted to the form that can be understandable by the expert system. Knowledge systems have traditionally been constructed by knowledge engineers, who are people interviewing domain experts and formalizing their knowledge. [Martin 1988]

![Knowledge Acquisition Diagram](image)

**Figure 2.2:** Knowledge Acquisition [Generation5 2006]

2.1.2 Inference Engine:

An Inference Engine is the transformation of the facts and rules in the knowledge base into a computer program that a common user can run to get certain results and recommendations. The inference mechanism is basically building of the expert system which understands the user input and matches it with the rules and facts of the expert system. Figure 2.3 describes the Inference Engine.
2.1.3 User Interface:

The User Interface is the component of the expert system where the user interacts with the expert system. The user inputs the expert system with his domain specific description of a new case and the expert system responds with the appropriate advice and explanation to the user. All the interaction between the user and the expert system are carried out in this component of the expert system.

2.2 Rule Based Expert System:

A rule based expert system is a typical expert system which operates on the basic rules provided in the knowledge base and the facts given as input from the user. The primary component of the expert system part of WNIDES is a rule based expert system. The architecture of the rule based expert system is shown in figure 2.4 [Expertise 2006].
2.2.1 Knowledge base:

The knowledge base of a typical rule based expert system comprises of the rules that operate the expert system. The rules are defined based on the particular domain on which the expert system operates. The rules may be stored in a database or as strings in a text file. The rules in the text file or database are taken as input by the inference engine to process the user request based on the facts available in the working memory [Expertise 2006].

2.2.2 Working Memory:

The working memory of the rule based expert system comprises of the user input stored in the form of facts or facts loaded from an external file. The facts in the working memory are the facts that determine the present state of the system. The facts in the working memory vary from time to time based on the user input and the operation of the expert system. The facts in the working memory are loaded into the inference engine.
where it matches the rules loaded into the expert system and fires the appropriate rule [Expertise 2006].

### 2.2.3 Inference Engine:

Inference Engine is like the central processing unit of the expert system. The matching of the rules and facts are carried out in this phase. Inference engine gets the rules from the knowledge base and the facts from the working memory and pattern matches the facts to the appropriate rule. The rules matched are passed on to the agenda where it matches the most appropriate rule and fires the rule [Expertise 2006].

### 2.2.4 Explanation Facility:

This is the facility where the rules fired by the inference engine are converted into a form which can be understandable by the user. Based on the output of the explanation facility the user gets the solution or provides with more facts to get a better explanation of the problem [Expertise 2006].

### 2.2.5 User Interface:

User Interface is the component of the rule based expert system where the user interacts with the expert system by providing the user input and getting the appropriate output [Expertise 2006].

### 2.3 WNIDES as a Rule Based Expert System:

WNIDES is developed based on the basic principles for the design of a general rule based expert system as shown in figure 2.5. The development of WNIDES consisted of following steps
- Knowledge Engineering
- Structuring data
- Developing Test Cases
- Interface building
- Writing the rules
- Testing
- Iterative Development

**Figure 2.5:** Development of Rule Based System [Generation5 2005]
2.3.1 Knowledge Engineering:

Knowledge engineering is the initial and most important phase of any rule based expert system design. This phase consists of the knowledge being collected from the expert. The expert from which the data is to be acquired for the system can be from any source like human expert, books, journals and articles relevant to the domain of the expert system etc. In knowledge engineering for WNIDES all the data necessary for the successful implementation of WNIDES is collected from the experts. The patterns and signatures that correspond to different types of network intrusions are collected in the knowledge engineering phase from the knowledge base. One such knowledge base is Snort IDS [Beale 2003]. This is an open source IDS that is based on rules. The rules present in snort are based on the signature of the network packet. The signature is parsed to determine whether it a network intrusion or an authenticated packet. The signature is present in the header part or payload of the network packet. A rule is fired corresponding to the signature of the incoming packet. If the WNIDES detects any intrusion or non-authenticated packet format it fires the corresponding rule based on the signature of the packet [Generation5 2006].

2.3.2 Structuring Data:

Structuring data is the second phase of the rule based expert system WNIDES. In this phase the data collected in the knowledge engineering phase of WNIDES is to be converted into rules and facts that can be used for the development of the rule based expert system. This phase includes the structuring of the data which makes the rules and facts to be implemented easily and directly in the computer system. This process resembles the analysis phase of the software engineering process. The characteristics or
sub-domains for which the WNIDES is being created are identified and based upon each characteristic or sub-domain the concepts are listed to make it easier in the design process for WNIDES [Generation5 2006].

2.3.3 Developing Test Cases:

In order for the expert system to work as determined some test cases were written for the testing of the expert system. Test cases were written to test each and every rule, test cases for rules of each sub-domain, test cases for the whole rule database and test cases for the whole integrated system [Generation5 2006].

2.3.4 Interface Building:

Interface development is one of the major processes in the building of a rule based expert system. The interface building of the WNIDES was about how the expert system will interact with the environment on which it is being built. The interface is between the network packets and the expert system component of WNIDES. It describes how it detects and monitors the network packets and analyses the signatures and compares it with the rules and facts of the WNIDES expert system [Generation5 2006].

2.3.5 Writing the rules:

Once the data structure was defined, the interfaces were specified, and the test cases were in place, rules were written. The rules of WNIDES were developed based on the Snort rules. The rule base in the Snort system was identified and was converted to strings of text for the implementation of WNIDES using the administrative component of WNIDES [Generation5 2006].
2.3.6 Testing:

Testing is done on the structured rules of the WNIDES to help in the implementation of the bug free rule based expert system. A product is more reliable and robust when testing is carried out in each phase of the Expert System development process. Testing is an important phase of an Expert System development. Products that are tested continuously from the initial stages tend to be more robust, modular and understandable. In the development of WNIDES tests were conducted in three stages [Generation5 2006].

Knowledge Acquisition Test: In this stage, testing on the accuracy of the knowledge was done to make sure all the knowledge and rules in the system are totally accurate with the knowledge that collected from Snort.

Structure and Design Test: In this stage, the structure of the knowledge and the user interface has been tested to ensure the ability of WNIDES on solving the intrusion detection issues. Besides that, the test on the system interface also done in this stage, it is to ensure that the users are comfortable with the design.

Full Prototype Test: In this stage, testing of the whole system has been done to ensure it reach the goals, where it will provide solution and advice to the users so that the users can use WNIDES to detect any intrusions.

2.3.7 Iterative Development:

Once the testing of the WNIDES Expert System was complete, the WNIDES expert system was done except the fact that the new rules or intrusion detection techniques that will be figured out in the future should be taken care of. When such a rule or intrusion detection technique is figured out in the future the development process of
WNIDES should be iterated from the knowledge engineering phase for that specific rule or intrusion detection technique. The rule based expert system WNIDES will make it possible for the new rules or intrusion detection techniques to be accepted from the administrative component of WNIDES as shown in figure 2.5 [Generation5 2006].
3. SYSTEM DESIGN

3.1 System Requirements:

The project was implemented using JESS an expert system building tool using Snort Rules as the knowledge base for building the expert system. Rules for WNIDES were developed understanding those available in Snort and scripting them in the JESS environment.

Java 2 Runtime Environment was used to run the JESS Applets and frames. Jakarta TOMCAT Web server which supports the Java applets was downloaded and installed. Java Applet pages were run on TOMCAT Server, IIS Server, and University’s Web server.

3.2 Java Expert System Shell (JESS)

JESS is a rule engine and scripting environment written entirely in Sun's Java language by Ernest Friedman-Hill at Sandia National Laboratories in Livermore, CA. JESS can be used to build java software that has the capacity to reason using knowledge supplied in the form of declarative rules. JESS is small, light, and one of the fastest rule engines available. [JESS 2004]

JESS has many features including backward chaining, working memory queries, and the ability to manipulate and directly reason about Java objects. JESS is also a powerful Java scripting environment, from which Java objects can be called directly and Java methods can be created without compiling any Java code. More about jess is described in Appendix A [JESS 2004].
3.3 Visual Basic .NET 2003:

Visual Basic .NET 2003 provides an easy way to develop applications, and it is a productive language tool for rapidly building applications for Microsoft Windows. Ideal environment for existing Visual Basic developers as well as new developers in the Microsoft .NET development environment, Visual Basic .NET 2003 delivers enhanced visual designers, increased application performance, and a powerful integrated development environment (IDE) to get one on the fast track to application development. VB.NET 2003 was used to develop the administrative component of WNIDES [MSDN 2006].

3.4 Snort:

Snort is an open source network intrusion prevention and detection system utilizing a rule-driven language, which combines the benefits of signature, protocol and anomaly based inspection methods. In the network intrusion detection phase Snort screens the packets received during the packet sniffering mode and fires the snort rules that trigger an alert whenever an unauthorized or a malicious packet is detected in the network. This snort rule database is converted into the WNIDES rule database using the administrative component of WNIDES [Snort 2006].

3.4.1 Snort Rules:

Most Snort rules are written in a single line. Snort rules are divided into two logical sections, the rule header and the rule options. The rule header contains the rule's action, protocol, source and destination IP addresses and netmasks, and the source and destination ports information. The rule option section contains alert messages and
information on which parts of the packet should be inspected to determine if the rule action should be taken. A sample snort rule is shown below

```
alert tcp any any -> 192.168.1.0/24 111 (content:"|00 01 86 a5|"; msg:"mountd access");
```

These Snort rules are based on intruder signatures. Snort has different set of rules for each set of specified signatures. Snort signatures are based on specified types of attacks. The Snort rule database available from snort was converted into WNIDES rule database using the administrative component of WNIDES. A set of different types of snort rules are described in Appendix B [Snort 2006].

### 3.5 Ethereal:

Ethereal is a network analyzer or packet sniffer. It works by reading network packets, decoding them, and present them in an easy to understand format. Ethereal is open source software that can be downloaded and used on many popular platforms such as Windows, UNIX, and Linux. Ethereal is a powerful networking tool that can be used by network professionals for troubleshooting, analysis, and protocol development. Each packet captured by ethereal contains Protocol, source_ipaddr, source_port, direction, destination_ipaddr, destination_port and content. These packets captured by ethereal are converted to facts that can be understandable to the expert system component of WNIDES [Ethereal 2006].

### 3.6 WNIDES Architecture:

The Architecture of WNIDES consists of activities like catching a packet from the network using Ethereal. Then the packet captured by Ethereal is sent to the expert system
component of WNIDES where it is converted to facts by constructs provided in JESS. These facts are compared with the rules in WNIDES and the matching rules are then fired and the corresponding action is taken.

![Figure 3.1: Architecture of WNIDES](image)

3.6.1 Packet Capturing:

Packet Capturing is the first phase of WNIDES expert system. Each network packet captured by Ethereal has the following format:

\[
\text{\{date\} - \{time\} \{source-ip-address:port\} \rightarrow \{destination-ip-address:port\} \{protocol\} \{TTL\} \{TOS\} \{ID\} \{IP-length\} \{datagram-length\} \{payload-length\} \{hex-dump\} \{ASCII-dump\}
\]
The data and time fields represent the date and time at which the packet is received. Source IP and port are the IP address and port number of the system from which the packet originated. Destination IP and port are the IP address and port number of the system to which the packet is to be delivered. Protocol is the type of protocol being used like TCP, UDP protocols. TTL is the time to live field of the packet. TOS determines the type of service of the packet. ID is the fragment identification number of the packet. IP-length gives the length of the packet. Datagram-length gives the length of the datagram. Payload-length gives the length of the payload in the packet. Hex-dump and ASCII-dump gives information about the content of the packet.

A sample output of two packets captured using Ethereal running on a host computer is as follows:

```
Protocol source_ipaddr source_port direction destination_ipaddr destination_port content
DNS 192.168.1.107 1489 ⇒ domain Standard query A www.google.com
Frame 1 (74 bytes on wire, 74 bytes captured)
User Datagram Protocol, Src Port: 1489 (1489), Dst Port: domain (53)
Domain Name System (query)

Protocol source_ipaddr source_port direction destination_ipaddr destination_port content
DNS ns5.lsn.net domain 192.168.1.107 1489 Standard query response CNAME
Frame 2 (318 bytes on wire, 318 bytes captured)
User Datagram Protocol, Src Port: domain (53), Dst Port: 1489 (1489)
Domain Name System (response)
```

19
The network packet format captured using Ethereal is implemented as a template for the WNIDES expert system using JESS. This is done using the `deftemplate` construct.

It is implemented as shown below:

```
(deftemplate NetworkPacket "A network packet"
(slot source_ipaddr) ; IP Address of the source
(slot source_port) ; Port Number of the source
(slot direction) ; Direction of the flow of the packet
(slot destination_ipaddr); IP Address of the destination
(slot destination_port) ; Port Number of the destination
(slot protocol) ; Protocol followed by the packet
(slot TTL) ; Time To Live value of the packet
(slot TOS) ; Type Of Service value of the packet
(slot ID); Fragment Identification Number of the packet
(slot IpLen) ; Length of the IP field
(slot DgmLen) ; Length of the datagram
(slot Seq) ; Value of Sequence Number of the packet
(slot Ack); Value of Acknowledgement Number of the packet
(slot content) ; ASCII or Binary data in the packet)
```

The Keyword `deftemplate` is used to define a template in Jess. Here the template is given the name “NetworkPacket” with description a network packet and different parts of the packet as slots.

### 3.6.2 Writing Rules:

The rules for WNIDES were written using the `defrule` construct of JESS. The content for the rule for WNIDES was obtained from Snort. There are about three thousand rules for the WNIDES expert system. These rules are subdivided into different subsections as described in Appendix A. A simple rule construct in jess can be described as follows.

```
(defrule rule4 "bad traffic rule"
(packet (protocol tcp)
(source_ipaddr $HOME_NET)
(source_port 0)
(direction <>)
```
This rule is fired when a bad traffic fact corresponding to this rule is loaded. The fact should be of TCP protocol from sender $HOME_NET from port number 0 to $EXTERNAL_NET with any port number.

3.6.3 Writing Facts:

The facts of the WNIDES will determine which rules are to be fired and when. The facts of WNIDES corresponding to different attacks are stored in an external file which will be loaded during the execution of the WNIDES expert system. A simple fact construct of jess can be described as follows.

(deffacts (packet (protocol tcp)(source_ipaddr any) (source_port 110) (direction ->) (destination_ipaddr any) (destination_port any) (content "Suddlently")))

Here this fact corresponds to a Trojan virus incoming. The fact Packet is added to the fact list and when this fact is loaded the rule corresponding to the Trojan virus incoming is fired by the expert system component of WNIDES.

3.6.4 Matching Rules and Facts:

WNIDES has a fixed set of rules, representing different types of attacks, while the facts, representing the network traffic, keep changing frequently. These rules and facts are supplied to the JESS rule engine that performs pattern matching. When a pattern is
matched to a set of facts the rule corresponding to the facts is fired or executed. All the
Rules corresponding to WNIDES are described in Appendix C.

3.7 WNIDES User Interface

WNIDES is implemented on the web. The User interface of client component of
WNIDES mainly consists of four web pages Main Page, JESS Documentation, Feed-
back and Help pages.

3.7.1 WNIDES Main Page

The main page of WNIDES consists of the JESS console and buttons to execute
WNIDES on web. The JESS console of WNIDES provides a textbox where JESS
commands can be executed. This feature can be used as a learning tool for JESS without
any pre-installation of JESS. The WNIDES main page is shown in Figure 3.2 below:
The WNIDES Jess Documentation page leads to the website of Sandia laboratories where all the documentation of JESS is provided. This is a good resource to know everything about JESS. The WNIDES JESS Documentation page is shown in Figure 3.3 below.
3.7.3 WNIDES Feedback Page

The WNIDES feedback page accepts the user feedback and mails it to the administrator. This User feedback helps in the bug fixing of WNIDES and helps in the development of WNIDES future releases. The WNIDES Feedback page is shown in Figure 3.4 below.
3.7.4 WNIDES Help Page

The WNIDES help page gives the user a good understanding of WNIDES Expert system. It helps the user to execute WNIDES and also gives a description about each user control on the WNIDES main page. The WNIDES help page is shown in Figure 3.5 below.
3.8 WNIDES Administrator Component

The Administrator Component of WNIDES consists of three main components
Rule Adder from Directory, Rule Adder from File and Single Rule Adder

3.8.1 Rule Adder from Directory

The Rule adder from Directory is developed in VB.NET 2003. This component allows the administrator to create rules from a set of files. The created rules thus can be added to the rule base of WNIDES. The rule adder from directory component of WNIDES is shown in figure 3.6 below.
3.8.1 Rule Adder from a File

The Rule adder from a File is developed in VB.NET 2003. This component allows the administrator to create rules from a single file. The created rules thus can be added to the rule base of WNIDES. The rule adder from a file component of WNIDES is shown in figure 3.7 below.
3.8.3 Single Rule Adder

The Single Rule adder is a component of the WNIDES rule based expert system. This component adds new rules to the WNIDES rules knowledge base. This application is for the administrative use of the WNIDES expert system. When a new rule is detected in the future the new rule adder component of the WNIDES expert system makes it possible for WNIDES to add new rules to its rule database. The new rule adder component of WNIDES is shown in figure 3.8 below.

![WNIDES Rule Loader](image)

**Figure 3.8:** WNIDES Single Rule Adder
4. EVALUATION AND RESULTS

The Evaluation of the WNIDES expert system was done using testing at three different phases Knowledge acquisition test, Structure and design test and Full prototype test. Different test cases for these three phases were carried out and the test results were evaluated.

4.1 Executing WNIDES

Web-based Network Intrusion Detection Expert System using JESS (WNIDES) is implemented in a human friendly interface. Executing WNIDES needs a few clicks of buttons. The following are the steps that need to be carried out in order to execute WNIDES.

1. Reset the WNIDES System using "Reset" button. This button will clear the WNIDES Working set memory.
2. Load the WNIDES JESS file using "WNIDES" button. This button loads the Wnides.clp file where the WNIDES rule database is stored.
3. Load the facts using "Load Facts" button. This button loads an external facts file named facts.dat. The facts.dat file contains the facts about the packets that are captured using Ethereal.
4. Execute the project using "Run" button. This button executes the WNIDES expert system by taking Wnides.clp as the rule database and facts.dat as the facts file.
4.2 Testing WNIDES

The testing of WNIDES Expert System was carried out at three different phases of the development.

**Knowledge Acquisition Test:** In this stage, testing on the accuracy of the knowledge was done to make sure all the knowledge and rules in the system are totally accurate with the knowledge that collected from Snort.

**Structure and Design Test:** In this stage, the structure of the knowledge and the user interface has been tested to ensure the ability of WNIDES on solving the intrusion detection issues. Besides that, the test on the system interface also done in this stage, it is to ensure that the users are comfortable with the design.

**Full Prototype Test:** In this stage, testing of the whole system has been done to ensure it reach the goals, where it will provide solution and advice to the users so that the users can use WNIDES to detect any intrusions.

These three phases of testing were carried out by different test cases. These are described in the following sub-sections.

**4.2.1 Single Rule Testing**

The goal of single rule testing was to test each and every rule and make sure that the WNIDES Expert system was reliable. In order to test each single rule different facts which would fire the appropriate rules were prepared and were loaded to WNIDES. WNIDES responded positive to each of these test cases. An example output of such a test case is shown in figure 4.1 below:
4.2.2 Single Domain Rule Testing

The goal of single domain rule testing was to reveal any error while executing all the rules in a specific domain of WNIDES rule database. In order to test single domain rule testing different test cases were developed for each domain and were tested against WNIDES expert system. The system responded positive in this test. One such example of a domain specific test case is shown in figure 4.2 below.
4.2.3 WNIDES Rule Database Testing

The goal of WNIDES rule database testing was to reveal any error while executing all the rules of WNIDES rule database by loading a single facts file. In order to test single domain rule testing a test case was developed in such a way that each rule had a corresponding fact in the facts file. This facts file along with the rule database was loaded into WNIDES Expert System. The WNIDES Expert System Fulfilled the test. The example of this test case is shown in figure 4.3 below.
4.2.4 Stress Testing

The goal of any stress test is to identify thresholds. In the case of WNIDES, the stress test was carried out under conditions when there was excess traffic over the network. In these conditions, some of the packets might go undetected. As a result the system might fail to detect potential intrusions. In order to perform stress test multiple web-pages of WNIDES were opened and each thread of WNIDES was loaded with facts file containing 855 facts. WNIDES performed well under stress condition as shown in figure 4.4 below when 855 facts were loaded.
Figure 4.4: Stress Test Result
5. FUTURE WORK

This project has some limitations that need to be handled in the future versions to make it a more robust system. Some of the future works required are as follows

5.1 A Better Knowledge-base

In the development of WNIDES, the Snort rules were used as the knowledge base for the signatures of attacks. These rules are based on the network traffic generated by Snort and contain attributes related to it. Effort should be made to find a better knowledge base that represents signatures of intrusions that are more efficient and that are accepted by a larger group of people.

5.2 An Inbuilt Packet-sniffing Tool

Ethereal was used as a packet-sniffing tool for capturing network traffic. In the future, effort should be made to develop a packet-sniffing tool that is part of WNIDES. This tool should capture the network traffic at real-time and convert the packets captured into the form that conforms to the data contained in the knowledge base of WNIDES.
Intrusion Detection is still a fledgling field of research. However, it is beginning to assume enormous importance in today's computing environment. The combination of facts such as the rapid growth of the Internet, the vast financial possibilities opening up in electronic trade, and the lack of truly secure systems make it an important and pertinent area of study.

The Web-based Network Intrusion Detection Expert System using JESS (WNIDES) is the first attempt towards developing a Network-based Intrusion Detection System using the Java Expert System Shell (JESS) on the web. About fifty different types of simulated attacks were tested against WNIDES rules and the results were found to be correct.

WNIDES may not be a full-fledged Intrusion Detection System due to its reliance on Snort and lack of required hardware infrastructure like setting up a large network with many hosts to provide various levels of network traffic; nevertheless, it will be the first step towards developing a powerful Intrusion Detection System that uses the best tools in existence today.
ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to Dr. Mario Al. Garcia, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his expertise in the area of Expert Systems, excellent guidance and encouragement during the period of this project work.

My sincere thanks to Dr. David Thomas, Associate Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest, unending support and warm wishes that provided me the much-needed motivation in completing my project.

My sincere thanks to Dr. Long-zhuang Li, Assistant Professor of Computer Science, Texas A&M University – Corpus Christi, for his keen interest in the field of Network Security, his patience and timely help has been very useful.

And finally heartfelt gratitude to all the faculty and staff of the Department of Computing and Mathematical Sciences, my colleagues and friends for their moral support.
BIBLIOGRAPHY AND REFERENCES


This section describes Jess, an expert system shell and scripting language written entirely in Sun Microsystem’s Java language. Jess supports the development of rule-based expert systems which can be tightly coupled to code written in the powerful, portable Java language. Jess is a programmer's library. The library itself is written in Java. Jess has an interactive command-line interface. Jess can be used in two overlapping ways [Jess 2004].

- First, it can be a rule engine - a special kind of program that very efficiently applies rules to data. A rule-based program can have hundreds or even thousands of rules, and Jess will continually apply them to data in the form of a knowledge base. Often the rules will represent the heuristic knowledge of a human expert in some domain, and the knowledge base will represent the state of an evolving situation (an interview, an emergency). In this case, they are said to constitute an expert system. Expert systems are widely used in many domains. Among the newest applications of expert systems are as the reasoning part of intelligent agents, in enterprise resource planning (ERP) systems, and in order validation for electronic commerce.

- Jess language is also a general-purpose programming language, and furthermore, it can directly access all Java classes and libraries. For this reason, Jess is also frequently used as a dynamic scripting or rapid application development environment. While Java code generally must be compiled before it can be run, a line of Jess code is executed immediately upon being typed. This allows you to experiment with Java APIs interactively, and build up large programs incrementally. It is also very easy to
extend the Jess language with new commands written in Java or in Jess itself, and so the Jess language can be customized for specific applications.

Jess is therefore useful in a wide range of situations. Jess's rule engine uses an improved form of a well-known algorithm called Rete to match rules against the knowledge base. Jess is actually faster than some popular expert system shells written in C, especially on large problems, where performance is dominated by algorithm quality [Jess 2004].

Jess can be used in many ways. Besides the different categories of problems Jess can be applied to, being a library, it is amenable to being used in many different kinds of Java programs. Jess can be used in command-line applications, GUI applications, servlets, and applets. You can develop Jess applications (with or without GUIs) without compiling a single line of Java code. You can also write Jess applications which are controlled entirely by Java code you write, with a minimum of Jess language code. The most important step in developing a Jess application is to choose architecture from among the almost limitless range of possibilities. One way to organize the possibilities is to list them in increasing order of the amount of Java programming involved [Jess 2004].

1. Pure Jess language scripts. No Java code at all.
2. Pure Jess language scripts, but the scripts access Java APIs.
3. Mostly Jess language scripts, but some custom Java code in the form of new Jess commands written in Java.
4. Half Jess language scripts, with a substantial amount of Java code providing custom commands and APIs; main() class provided by Jess.
5. Half Jess language scripts, with a substantial amount of Java code providing custom commands and APIs; `main()` class written by programmer.

6. Mostly Java code, which loads Jess language scripts at runtime.

7. All Java code, which manipulates Jess entirely through its Java API. This option is not fully supported at this time, but will in a future release.
APPENDIX B: SNORT RULES

Snort comes with a rich set of rules. These rules are divided into different files. Each file represents one class of rules. The source code distribution of Snort has the following set of rules [Snort 2006]:

- attack-responses rules
- backdoor rules
- bad-traffic rules
- chat rules
- ddos rules
- deleted rules
- dns rules
- dos rules
- experimental rules
- exploit rules
- finger rules
- ftp rules
- icmp-info rules
- icmp rules
- imap rules
- info rules
- local rules
- misc rules
• multimedia rules
• mysql rules
• netbios rules
• nntp rules
• oracle rules
• other-ids rules
• p2p rules
• policy rules
• pop3 rules
• porn rules
• rpc rules
• rservices rules
• scan rules
• shellcode rules
• smtp rules
• snmp rules
• sql rules
• telnet rules
• tftp rules
• virus rules
• web-attacks rules
• web-cgi rules
• web-client rules
• web-coldfusion rules
• web-frontpage rules
• web-iis rules
• web-misc rules
• web-php rules
• x11 rules
APPENDIX C: WNIDES PROGRAM LISTING

/commons

; Web-based Network Intrusion Detection Expert System using Jess (WNIDES)
; WNIDES is a network intrusion detection system built with expert system
; technology using Java Expert System Shell (JESS).
; Name: Vamshi K Kankanala
; Degree: Master of Science
; Major: Computer Science
; Semester: Spring 2006

(deftemplate packet
  "A network packet"
  (slot protocol) ; Protocol of the packet
  (slot source_ipaddr) ; IP Address of the source
  (slot source_port) ; Port Number of the source
  (slot direction) ; Direction of flow of the network traffic
  (slot destination_ipaddr) ; IP Address of the destination
  (slot destination_port) ; Port Number of the destination
  (slot TTL) ; Time To Live value of the packet
  (slot TOS) ; Type of Service value of the packet
  (slot ID) ; Fragment Identification Number of the packet
  (slot IpLen) ; Length of the IP packet
  (slot DgmLen) ; Length of the datagram
  (slot Seq) ; Value of the Sequence Number
  (slot Ack) ; Acknowledgement Number value
  (slot TcpLen) ; Length of the TCP packet
  (slot content) ; Payload content of the packet
)

; The rules form the knowledge base of WNIDES.
; Each rule identifies the signature for a particular intrusion.
;; Rules are fired when the facts from the working memory match them.
;;**********************************************************************
;;****************
;;**********************************************************************
;;****************
;; VIRUS RULES
;;**********************************************************************
;;****************

(defun rule1 "Snowwhite Trojan"
  (packet (protocol tcp)
    (source_ipaddr any)
    (source_port 110)
    (direction ->)
    (destination_ipaddr any)
    (destination_port any)
    (content "Suddlently"))
=>
  (printout t "Alert, Virus-Snowwhite Trojan Incoming" crlf)
)

(defun rule2 "pif Worm"
  (packet (protocol tcp)
    (source_ipaddr any)
    (source_port 110)
    (direction ->)
    (destination_ipaddr any)
    (destination_port any)
    (content ".pif"))
=>
  (printout t "Alert, Virus - Possible pif Worm" crlf)
)

(defun rule3 "NAVIDAD Worm"
  (packet (protocol tcp)
    (source_ipaddr any)
    (source_port 110)
    (direction ->)
    (destination_ipaddr any)
    (destination_port any)
    (content "NAVIDAD.EXE"))
=>
  (printout t "Alert, Virus - Possible NAVIDAD Worm" crlf)
)
(defrule rule4 "bad traffic rule"
  (packet (protocol tcp)
    (source_ipaddr $HOME_NET)
    (source_port 0)
    (direction <>)
    (destination_ipaddr $EXTERNAL_NET)
    (destination_port any))
=>
  (printout t "Alert, BAD TRAFFIC tcp port 0 traffic" crlf)
)

(defrule rule5 "bad traffic rule"
  (packet (protocol udp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction <>)
    (destination_ipaddr $HOME_NET)
    (destination_port 0))
=>
  (printout t "Alert, BAD TRAFFIC udp port 0 traffic" crlf)
)

(defrule rule6 "bad traffic rule"
  (packet (protocol tcp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port any))
=>
  (printout t "Alert, BAD TRAFFIC data in TCP SYN packet" crlf)
)

;;;**********************************************************************
;;; CHAT RULES
;;; These signatures look for people using various types of chat programs
;;; (Eg: AIM, ICQ, IRC) which may be against corporate policy.
(defrule rule7 "chat rule"
(packet (protocol tcp)
(source_ipaddr $HOME_NET)
(source_port any)
(direction ->)
(destination_ipaddr $EXTERNAL_NET)
(destination_port any)
(content "User-Agent:\ICQ")
=>
(printout t "Alert, CHAT ICQ Access" crlf)
)

(defrule rule8 "chat rule"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port 80)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "Content-Type:\ application/x-icq")
=>
(printout t "Alert, CHAT ICQ forced user addition" crlf)
)

(defrule rule9 "chat rule"
(packet (protocol tcp)
(source_ipaddr $HOME_NET)
(source_port any)
(direction <>)
(destination_ipaddr $EXTERNAL_NET)
(destination_port 1863)
(content "MSG")
=>
(printout t "Alert, CHAT MSN message" crlf)
)

;;;; SNMP RULES
;;;;********************************************************************

;; SNRP RULES

(defrule rule10 "snmp rule"
(packet (protocol udp)
  (source_ipaddr $EXTERNAL_NET)
  (source_port any)
  (direction ->)
  (destination_ipaddr $HOME_NET)
  (destination_port 161)
  (content "|04 00|"))
=>
(printout t "Alert, SNMP missing community string attempt" crlf)
)

(defrule rule11 "snmp rule"
  (packet (protocol udp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port 161)
    (content "|04 00|"))
=>
(printout t "Alert, SNMP null community string attempt" crlf)
)

(defrule rule12 "snmp rule"
  (packet (protocol udp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port 161:162)
    (content "|02 01 00 04 82 01 00|"))
=>
(printout t "Alert, SNMP community string buffer overflow attempt" crlf)
)

.";********************************************************************
 ;; DDOS RULES
;";********************************************************************

(defrule rule13 "ddos rule"
  (packet (protocol icmp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction <-)
    (destination_ipaddr $EXTERNAL_NET)
    (content "|02 01 00 04 82 01 00|"))
=>
(printout t "Alert, ICMP packet too large" crlf)
)
(destination_port any)
(content "1234")
=>
(printout t "Alert, DDOS TFN Probe" crlf)
)

(defrule rule14 "ddos rule"
(packet (protocol icmp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "AAAAAAAAAA"))
=>
(printout t "Alert, DDOS tfn2k icmp possible communication" crlf)
)

(defrule rule15 "ddos rule"
(packet (protocol icmp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any))
=>
(printout t "Alert, DDOS TFN client BE" crlf)
)

;;*******************************************************************
;;DNS RULES
;;*******************************************************************

(defrule rule16 "dns rule"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 53)
(content "|00 00 FC|")
=>
(printout t "Alert, DNC zone transfer TCP" crlf)
)

(defrule rule17 "dns rule"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 53)
(content ":00 00 FC")
=>
(printout t "Alert, DNC zone transfer TCP" crlf)
)
(packet (protocol udp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 53)
(content "|00 00 FC|"))
=>
(printout t "Alert, DNC zone transfer UDP" crlf) )

(defrule rule18 "dns rule"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 53)
(content "|07|authors"))
=>
(printout t "Alert, DNC named authors attempt" crlf) )

**********************************************************************
* ATTACK RESPONSES
* These signatures are those that happen when the machine has been
* compromised. These should not false that often and almost always
* mean that the machine has been compromised.
**********************************************************************

(defrule rule19 "attack response"
(packet (protocol ip)
(source_ipaddr any)
(source_port any)
(direction ->)
(destination_ipaddr any)
(destination_port any)
(content "uid=0(root)"))
=>
(printout t "Alert, ATTACK RESPONSES id check returned root" crlf) )

(defrule rule20 "attack response"
(packet (protocol ip)
(source_ipaddr any)
(source_port any)
(direction ->)
(destination_ipaddr any)
(destination_port any)
(content "uid=0(root)")
=>
(printout t "Alert, ATTACK RESPONSES id check returned root" crlf)
)

(defrule rule21 "attack response"
(packet (protocol ip)
(source_ipaddr any)
(source_port any)
(direction ->)
(destination_ipaddr any)
(destination_port any)
(content "uid=0(root)")
=>
(printout t "Alert, ATTACK RESPONSES id check returned root" crlf)
)

;;*******************************************************************
;;FINGER RULES
;;*******************************************************************

(defrule rule22 "finger rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 79)
(content "cmd_rootsh")
=>
(printout t "Alert, FINGER cmd_rootsh backdoor attempt" crlf)
)

(defrule rule23 "finger rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 79)
(content "a b c d e f")
)

53
(printout t "Alert, FINGER account enumeration attempt" crlf)
)

(deffact rule24 "finger rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 79)
(content "search"))
=>
(printout t "Alert, FINGER search query" crlf)
)

;;*******************************************************************
;;FTP RULES
;;*******************************************************************

(deffact rule25 "ftp rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 21)
(content "CEL")
=>
(printout t "Alert, FTP CEL overflow attempt" crlf)
)

(deffact rule26 "ftp rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 21)
(content "CWD")
=>
(printout t "Alert, FTP CWD overflow attempt" crlf)
)

(deffact rule27 "ftp rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 21)
(content "CWD")
=>
(printout t "Alert, FTP CWD overflow attempt" crlf)
)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 21)
(content "CMD")
=>
(printout t "Alert, FTP CMD overflow attempt" crlf)
)

;;*******************************************************************
;;ICMP RULES
;;These rules are potentially bad ICMP traffic. They include most of
;;the ICMP scanning tools and other "BAD" ICMP traffic
;;*******************************************************************

(defrule rule28 "icmp rules"
(packet (protocol icmp)
(source_ipaddr SEXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "|495353504e475251|")
)=>
(printout t "Alert, ICMP ISS Pinger" crlf)
)

(defrule rule29 "icmp rules"
(packet (protocol icmp)
(source_ipaddr SEXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "ABCDEFGHIJKLMNOPQRSTUVWABCDEFGHI")
)=>
(printout t "Alert, ICMP L3retriever Ping" crlf)
)

(defrule rule30 "icmp rules"
(packet (protocol icmp)
(source_ipaddr SEXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "\0000000000000000000000000000000000000000\")
=>
(printout t "Alert, ICMP Nemesis v1.1 Echo" crlf)
)

;;ICMP-INFO RULES
;;These rules are standard ICMP traffic. They include OS pings, as
;;well as normal routing done by ICMP. There are a number of "catch
;;all" rules that will alert on unknown ICMP types.
;;*****************************************************************************

(defun rule31 "icmp-info rules"
  (packet (protocol icmp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port any))
=>
(printout t "Alert, ICMP IRDP router advertisement" crlf)
)

(defun rule32 "icmp-info rules"
  (packet (protocol icmp)
    (source_ipaddr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port any))
=>
(printout t "Alert, ICMP IRDP router selection" crlf)
)

(defun rule33 "icmp-info rules"
  (packet (protocol icmp)
    (source_ipadr $EXTERNAL_NET)
    (source_port any)
    (direction ->)
    (destination_ipaddr $HOME_NET)
    (destination_port any)
    (content "\10111213141516171819\1a1b1c1d1e1f")
  )
=>
(printout t "Alert, ICMP PING *NIX" crlf)
)
(defrule rule34 "imap rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 143)
(content "Login {")
=>
(printout t "Alert, IMAP login buffer overflow attempt" crlf)
)

(defrule rule35 "imap rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 143)
(content "AUTHENTICATE {")
=>
(printout t "Alert, IMAP authenticate overflow attempt" crlf)
)

(defrule rule36 "imap rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 143)
(content "AUTH")
=>
(printout t "Alert, IMAP auth overflow attempt" crlf)
)

(defrule rule37 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(destination_ipaddr $HOME_NET)
(destination_port 143)
(content "INFO")
=>
(printout t "Alert, IMAP info overflow attempt" crlf)
)
(source_ipaddr $EXTERNAL_NET)
(source_port 80)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "Connection closed by foreign host") =>
(printout t "Alert, INFO Connection Closed MSG from Port 80" crlf)
)

(defrule rule38 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 21)
(content "PASS")
) =>
(printout t "Alert, INFO FTP No Password" crlf)
)

(defrule rule39 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SMTP_SERVERS)
(destination_port 25)
(content "BattleMail")
) =>
(printout t "Alert, INFO battle-mail traffic" crlf)
)

;;**********************************************************************
*****
;; MULTIMEDIA RULES
;; These signatures look for people using streaming multimedia
;; technologies. Using streaming media may be a violation of corporate
;; policies.
;;**********************************************************************
*****

(defrule rule40 "multimedia rules"
(packet (protocol tcp)
(source_ipaddr $HOME_NET)
(source_port 58
...
(source_port any)
(directoin ->)
(destination_ipaddr $EXTERNAL_NET)
(destination_port 80)
(content "User-Agent: Quicktime")
=>
(printout t "Alert, MULTIMEDIA Quicktime User Agent access" crlf)
)

(defvar rule41 "multimedia rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port 80)
(directoin ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "Content-type: audio/x-ms-wma\r\n")
=>
(printout t "Alert, MULTIMEDIA Windows Media audio download" crlf)
)

(defvar rule42 "multimedia rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port 80)
(directoin ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "Content-type: video/x-ms-asf\r\n")
=>
(printout t "Alert, MULTIMEDIA Windows Media Video download" crlf)
)

;; MYSQL RULES
;; These signatures detect unusual and potentially malicious mysql traffic.
;; These signatures are not enabled by default as they may generate false
;; positive alarms on networks that do mysql development.
(defvar rule43 "mysql rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port 3306)
(content "|0A 00 00 01 85 04 00 00 80 72 6F 6F 74 00|")
=>
(printout t "Alert, MYSQL root login attempt" crlf)
)

(defrule rule44 "mysql rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port 3306)
(content "|0f 00 00 00 03|show databases")
)=>
(printout t "Alert, MYSQL show databases attempt" crlf)
)

;;*******************************************************************
;; NETBIOS RULES
;;*******************************************************************

(defrule rule45 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 139)
(content "|00|.|00|E|00|M|00|L")
)=>
(printout t "Alert, NETBIOS nimda .eml" crlf)
)

(defrule rule46 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 139)
(content "|00|.|00|E|00|M|00|L")
)=>
(printout t "Alert, NETBIOS nimda .nws" crlf)
)
(defrule rule47 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 139)
(content "R|00|I|00|C|00|H|00|E|00|D|00|2|00|0")
=>
(printout t "Alert, NETBIOS nimda RICHEDO.DLL" crlf))

(defrule rule48 "info rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 139)
(content "BEAVIS")
=>
(printout t "Alert, NETBIOS RFParalyze Attempt" crlf))

;; NNTP RULES
;;*******************************************************************

(defrule rule49 "nntp rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port 119)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port any)
(content "200")
=>
(printout t "Alert, NNTP return code buffer overflow attempt" crlf))

(defrule rule50 "nntp rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 119)
(content "AUTHINFO USER")
=> 
(printout t "Alert, NNTP AUTHINFO USER overflow attempt" crlf)
)

;;*******************************************************************
;;; ORACLE RULES
;;; These signatures detect unusual and potentially malicious oracle
;;; traffic.
;;;*******************************************************************

(defrule rule51 "oracle rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port $ORACLE_PORTS)
(content "EXECUTE_SYSTEM")
)=> 
(printout t "Alert, ORACLE EXECUTE_SYSTEM attempt" crlf)
)

(defrule rule52 "oracle rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port $ORACLE_PORTS)
(content "connect_data\{command=version\}")
)=> 
(printout t "Alert, ORACLE connect_data\{command=version\} attempt" crlf)
)

(defrule rule53 "oracle rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port $ORACLE_PORTS)
(content "description=\{\")
(printout t "Alert, ORACLE misparsed login response" crlf)
)

(defrule rule54 "oracle rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $SQL_SERVERS)
(destination_port $ORACLE_PORTS)
(content "select")
=>
(printout t "Alert, ORACLE select union attempt" crlf)
)

;;*******************************************************************
;; POP2 RULES
;;*******************************************************************

(defrule rule55 "pop2 rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 109)
(content "FOLD")
=>
(printout t "Alert, POP2 FOLD overflow attempt" crlf)
)

(defrule rule56 "pop2 rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 109)
(content "FOLD /")
=>
(printout t "Alert, POP2 FOLD arbitrary file attempt" crlf)
)

(defrule rule57 "pop2 rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 109)
(content "|eb2c 5b89 d980 c106 39d9 7c07 8001|")
=>
(printout t "Alert, POP2 x86 linux overflow" crlf)
)

;; POP3 RULES
;;*******************************************************************
;; POP3 RULES
;;*******************************************************************

(defvar rule58 "pop3 rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 110)
(content "USER")
)=>
(printout t "Alert, POP3 USER overflow attempt" crlf)
)

(defvar rule59 "pop3 rules"
(packet (protocol tcp)
(source_ipaddr $EXTERNAL_NET)
(source_port any)
(direction ->)
(destination_ipaddr $HOME_NET)
(destination_port 110)
(content "AUTH")
)=>
(printout t "Alert, POP3 AUTH overflow attempt" crlf)
)