RZ 3253 (# 93299) 06/26/00
Computer Science 45 pages

Research Report

Integration of Host-based Intrusion Detection Systems into the Tivoli Enterprise Console

Christian Gigandet

IBM Research
Zurich Research Laboratory
8803 Rüschlikon
Switzerland

LIMITED DISTRIBUTION NOTICE

This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g., payment of royalties).

IBM Research
Almaden · Austin · Beijing · Delhi · Haifa · T.J. Watson · Tokyo · Zurich

Integration of Host-based Intrusion Detection Systems into the Tivoli Enterprise Console

Christian Gigandet

IBM Research, Zurich Research Laboratory, 8803 Rüschlikon, Switzerland

Abstract

For many years, the security has become a main issue in Internet community. Intrusion detection systems (IDSes) are security tools which should indicate when someone is breaking into a system. However, as this thesis shows, the local exploits are not well detected and IDSes generate a huge amount of data that should not always be considered as intrusion. There is no universal IDS, all IDSes have different strengths and weaknesses. The detection of intrusion can be improved by correlating different information sources and by considering the environment in which they occur.

Contents

Chapter 1

Introduction

The Internet is the world's largest computer network, connecting several million computers to each other. More and more companies are on the net. This huge network is one of the main business place which is conducted nowadays. Lots of sensible data is shared since the coming of e-commerce, internet-banking and many other applications. All these different businesses speak in terms of security. Security risks increase as companies migrate their business applications to support intranet and extranet activities. Since the recent distributed denial-of-service attacks, some of the biggest companies, like Yahoo, have suffered. Security became the key word of the internet community. A lot of network-device companies are developing their own network security solution. One of the best known is certainly CISCO, but they do not have the monopoly. The number of firms which propose other solutions is always growing. A network security solution is a notion to qualify several tools acting for the protection of your network. The most frequently used are firewalls, anti-virus, network scanners, and Intrusion Detection Systems (IDSes).

Many research reports have been written in this field. The Global Security Analysis Laboratory (GSAL) at IBM Research has analyzed several network-based intrusion detection systems and has integrated them in the Tivoli Enterprise Console [1]. This report completes the previous research in testing two host-based intrusion detection systems of Internet Security Systems (ISS) and Axent. The contribution of host-based compared or associated with the network-based IDSes is discussed.

This project has several goals. The first one is to get first hand experience with two market leaders of host based IDSes. Then the second aim is to test them with a variety of attacks to detect their weaknesses. This project enriches the Internet Engineering Task Force (IETF) work [2] in order to define a format to represent the host-based IDS alerts. Finally we want to design, implement and test the integration into the Tivoli Enterprise Console (TEC).

The experience with products means, of course, to install them, but furthermore to see what are their possibilities in terms of configuration and usability. Their functionality must be well understood in order to be able to exploit all their capabilities.

These products must be tested in order to see what kind of alerts are generating by the host-based IDSes. Two kind of events needs to be tested. First, events related to a normal host-based activity and then, some attacks from outside and inside. Many attacks are available in a Vulnerability Database, maintaind by IBM, called Vulda.

The IETF work proposes a common representation of the alarms triggered by the IDSes. The actual format is mainly based on the network aspect and this project aims to complete it by defining the needed host based events.

The motivation of integrating the IDSes events in a common framework is that, as this report shows, there is no universal IDS and the gain in correlating many sources is important. This helps eliminating false alarms and increase the amount of detectable events.

The structure of this report is as follows; chapter 2 introduces the IDSes. The results obtained by testing two IDSes are described in chapter 3. The integration process is presented in chapter 4. Chapter 5 describes the classes added to the class hierarchy[1] proposed by the internet draft [2] and explains the implementation details. The conclusion can be found in chapter 7.

[1] representation of the alarms triggered by the IDSes

Chapter 2

Background

The Internet contains a lot of information describing, step by step, how to penetrate systems. This is one of the reasons why hacking is becoming a popular new hobby. More and more companies are under hacker's attacks. Mainly based on research reports [3, 4, 5], this chapter introduces the intrusion detection systems and presents their different characteristics.

2.1 Network-security

Setting up a high-level secured network is more and more difficult. Each day lots of new vulnerabilities are discovered and can immediately be exploited by malicious hackers. Even if a lot of security measures are deployed, it is impossible to be sure that no security holes are present. In fact, the holes can be due to misconfiguration, software, hardware, protocol, and other sources.

Basically, a network can be under different types of attacks. It can be attacked by external parties or internal users could try to abuse their privileges in exploiting some weaknesses in the system. A FBI report on computer crime notes that the primary threat comes from full-time employees [6].

A good security strategy should be based on a layer model. The first defensive line must control the access from outsid and from inside. This configuration should be tested with some security scanners like Satan [7] or Nessus [8]. Finally a trace of all the actions performed by the users should be kept. This is frequently done with log files and auditing.

2.1.1 Firewall

A lot of companies use one or several firewalls to protect them from outside attackers. A firewall is a set of related programs, located at a network gateway server, that protects the resources of a private network from users of other networks. Basically a firewall filters all network packets. If a packet fits with the company's inner policy, it is accepted and rejected in the other case.

"We set up firewalls, so our network is secure" is a comment often heard in the security industry. Unfortunately, firewalls can be complicated to set up and are subject to misconfiguration that can leave the network unprotected. Furthermore firewalls must guarantee some degree of access which may allow for probing weaknesses. In addition a firewall does not prevent from internal misuses. An unprotected connection can be opened while utilizing a modem.

2.1.2 User privileges

Since the first multiuser platforms, a lot of development has been made in order to provide a user friendly platform and to control resource access. The most common systems used the concept of "user rights" which allows the manager to define different privileges. However, the user rights do not prevent the user from exploiting weaknesses in order to increase there privileges.

2.2 Intrusion detection

An IDS aims to detect computer misuses, meaning each activity that does not conform with the internal policy. Such activity can be caused by internal users or external parties trying to exploit vulnerabilities. An alert should be generated when someone tries to penetrate the system and when someone has penetrated it.

Generally the term intrusion defines actions from the outside. However for the intrusion detection systems community, an activity of a malicious insider is also an intrusion. The detection process is composed of several main phases:

- First the IDS acquires information about its environment. It can be from a workstation, a server, a firewall, a network component, the network itself, and others.
- Then some form of analyzis is performed to determine the system's state: normal or under attack.
- When an errorenous state is detected the system must react. It can be with a passive response like a simple notification or an active response as to kill a session.

2.2.1 Information sources

There are many information sources available that can be used by the IDSes. There are basically two different types:

1. Information taken from the network: network packets. That means the IDS captures the packets on the network and analyzes them.
2. Information available on the host, from the operating system, different application logs, data files system, etc...

Depending on the information source, the IDS is called *network-based* or *host-based*. The network-based analyze the stream of packets on the network, whereas the host-based evaluate information that can be found on the host.

2.2.2 Detection Method

There are two different approaches to analyze the collect of data. The first one tries to define a normal behavior[1] of the system. When it is defined, the IDS simply looks for a eventual deviation [2]. This first way of analyzing is not often used. It is really difficult to define what the usual behavior is. Such a definition is unique for each company and of course can depend on many external factors. An IDS which analyzes information this way is a called *behavior-based* IDS.

The other way concentrates on the event itself and performs pattern matching to recognize a particular attack by its signature. That means we must already know the attack signature in

[1] which can be based on statistics, expert systems and neural networks.

order to detect it. This is the reason why the signature database must be frequently updated. At the present time, almost all commercial systems use this form of analyzis [4]. They are called *knowledge-based* IDSes.

2.2.3 Network-based vs. Host-based

In theory, a single product can include both approaches: host-based and network-based. However they are often separate in two commercial products which are more or less integrate. They are also complementary. In fact, in deploying both network and host based IDS solutions, both network traffic and host exploits are monitored [9]. Basically the network-based approach provide the earliest possible warning. It detects the attack before the damage is done. The host-based approach will indicate, most of the time, a successful intrusion. After an intrusion, indications about the intruder activity on the target system can be provided.

Both network and host-based systems are affected by a variety of implementation difficulties and limitations [4]. The network-based are vulnerable, among of the things, to packet spoofing and packet fragmentation [10]. It is not always possible to decipher the packets for some other reasons which can be due to protocols or encryption. Furthermore the network IDS can only process packets on low-speed networks (10 Mbps) and must be installed on a shared segment. On the other side, the host-based system does not have all the solutions. They do not monitor direct information like network-packets, but only indirect auditing information created by the system or applications which has a performance impact. Host-based IDSes sometimes can also interfere with regular host operation.

2.3 Products

The IDSes are more and more advanced. The first commercial IDS was released in 1991 [4]. Although many different products are available on the market, IDSes are still in research and development. There are many research reports about this topic. Table 2.1 shows different commercial IDS characteristics taken from [4].

Another class of software exists, which includes some detection functionality. These are host wrappers and personal firewalls. These tools can be configured to look at network packets, connection attempts, or login attempts on a computer[11].

Product	Deployment Strategy		Information Source							Method		Real-time Processing		Initial Release
	net-based	host-based	network packets	OS	Web server	router	firewall	file system	other	knowledge	behavior	yes	no	year
1. Anzen Flight Jacket	✗		✗							✗	✗	✗		1997
2. Centrax	✗	✗	✗	✗						✗	✗	✗		1998
3. Computer Misuse Detection System (CMDS)		✗		✗			✗			✗	✗	✗		1991
4. Cross-Site for Security	✗		✗							✗		✗		1998
5. CyberCop Monitor		✗		✗						✗		✗		1999
6. Intruder Alert		✗		✗		✗	✗		✗	✗		✗		1992
7. Kane Security Monitor(KSM)		✗		✗						✗		✗		1996
8. NetProwler	✗		✗							✗		✗		1997
9. Net-Ranger	✗		✗							✗		✗		1997
10. Reactive Intrusion Detection (RID)		✗					✗			✗		✗		1998
11. RealSecure	✗	✗	✗							✗		✗		1996
12. SecureCom Switches	✗		✗							✗		✗		1997-8
13. SecureNet PRO	✗		✗							✗		✗		1997
14. SessionWall-3	✗		✗							✗		✗		1997
15. SMARTWatch		✗			✗			✗		✗		✗		1997
16. Stakeout	✗		✗							✗		✗		1997
17. Tripwire		✗						✗		✗			✗	1998

Table 2.1: Current IDS Products [4]

Chapter 3

Intrusion Detection Products Testbed

This chapter contains results obtained by testing two different host-based intrusion detection systems: RealSecure system agent version 3.2 from Internet Security Systems (ISS) and Intruder Alert version 3.0 from Axent. The goal is not to compare these two products, but rather to see the capabilities of actual commercial host-based IDSes. ISS and Axent are two leaders in the security market and the evaluation of their host-based solutions should be representative. In order to be as clear as possible the results of the two products are presented in two different sections.

3.1 Axent: Intruder Alert v3.0

Intruder Alert (ITA) is a host-based IDS composed of four components:

- The manager is a UNIX daemon, Windows NT service or NetWare loadable module used to organize agents and administer policies. It supervises the communication between all others ITA components (agents, ITA view and ITA admin).
- The agent is also a UNIX daemon, Windows NT service or NetWare loadable module. It is used to monitor all the events and perform defined actions.
- ITA Admin (Win95/98/NT) is a graphical user interface used to organize the monitoring. Thus one can organize agents in domains, activate or disable policies, add log files to monitor, and so on.
- ITA View (Win95/98/NT) is also a graphical user interface used to view events captured by Agents.

Intruder Alert is a multi-platform tool. The manager and the agents can be installed on several UNIX platforms, WinNT and NetWare. The agents are organized in domains where the same policies are applied.

The term policy can have different meanings. For Axent, a policy is a particular event or group of events. It can be a basic event like a "su root" or a composed event like a "Network Probe". A policy contains one or several rules composed of three clauses: *select, ignore*, and *action*. The *select* and *ignore* clauses filter all the information sources, to catch the desired events, while the *action* clause defines the response.

Several functions are used. The complete list is in table 3.1 and 3.2. Basically the events are caught by pattern matching (system message functions). The possible responses are either notifi-

cations (email, ITA view), or actions on a specific flag (raised up, raised down), or a more active response (disconnect a session, script execution). Hereafter you will see some policy examples used an the next chapters.

Function	Description
System Messages	select events that contain specific text (like a keyword search)
ITA Rule	select events based on another rule
User	events generated by a specified user (not known for each event!)
ITA Command	events generated by a specific command which can be sent with ITA View
STATUS	select a specific Intruder Alert Status messages
ITA error	based on error messages (manager.log, agent.log)
Flag	select raised flags. When many flags are used inside a policy, the boolean condition and/or can be used
Date	within a range of time
Timer	select event when a specific timer has ended

Table 3.1: *select* and *ignore* clause functions

Function	Description
Append to a file	could be on another file
Send E-mail	send an email
Notify(action)	Send a message to the term
Pager	message on a pager
Kill process	only if the event contains a "PID:XXXX"
Disconnect Session	kill the session. Event must contain "Session ID:XXXX"
Raised Flag	raised a flag for a defined period of time. It can be raised globally on all agent or just locally
Cancel Flag	cancel flag
Execute command	Execute a program or a script. Two variables can be used, {user} and {event file}
Record to ITA View(action)	record the event into the ITA View GUI
Disable user Account	Exepted for accounts having root privileges. Looks for "username: XXXX".
Run shared Actions	Define several actions witch can be used by all other rules as reference

Table 3.2: *action* clause functions

3.1.1 Testbed Setup

The installation process consists of two phases. First the installation of the manager(s) and the agent(s), and then the GUI tool's installation. During the manager and agent installation, information like the owner of the ITA files, the name of the administrator, a password (no relation with the unix password) and other things are requested. Here is the architecture we decided to set up:

- Manager and Agent on AIX system (aix1)
- Agent on Solaris system (sun1)

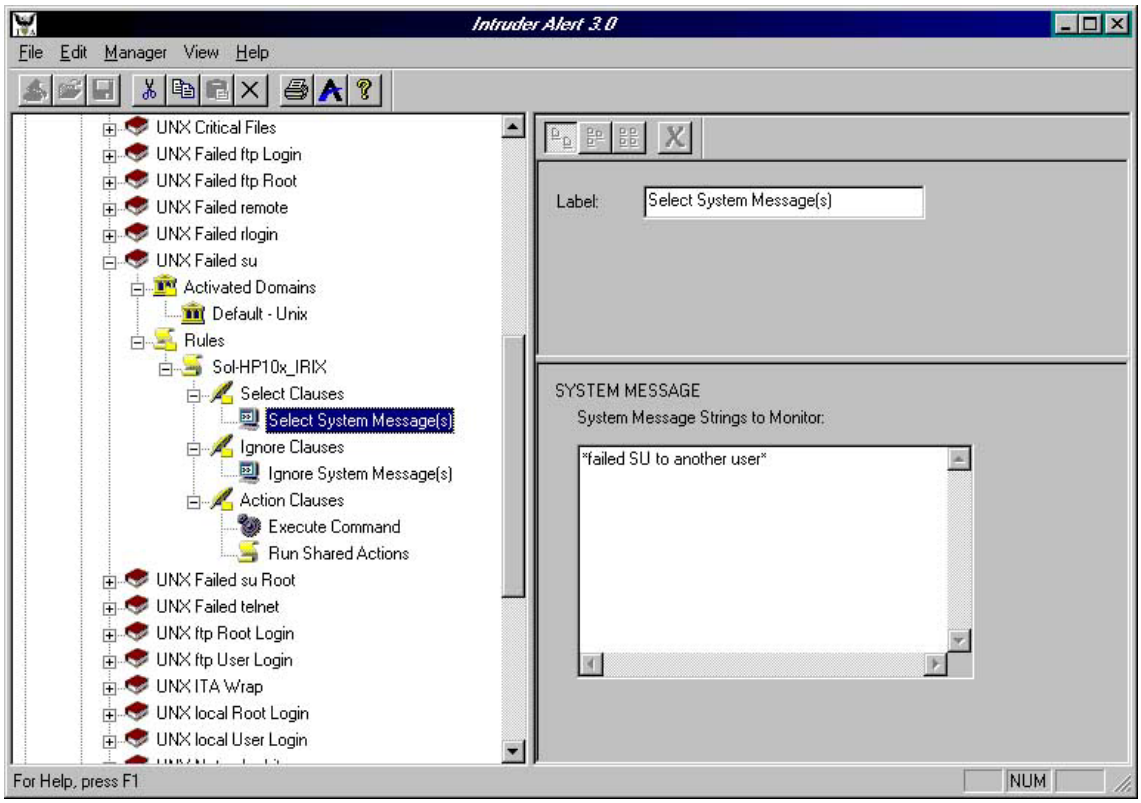

Figure 3.1: A screenshot of ITA admin. You can see the pattern matching used for the UNX failed su.

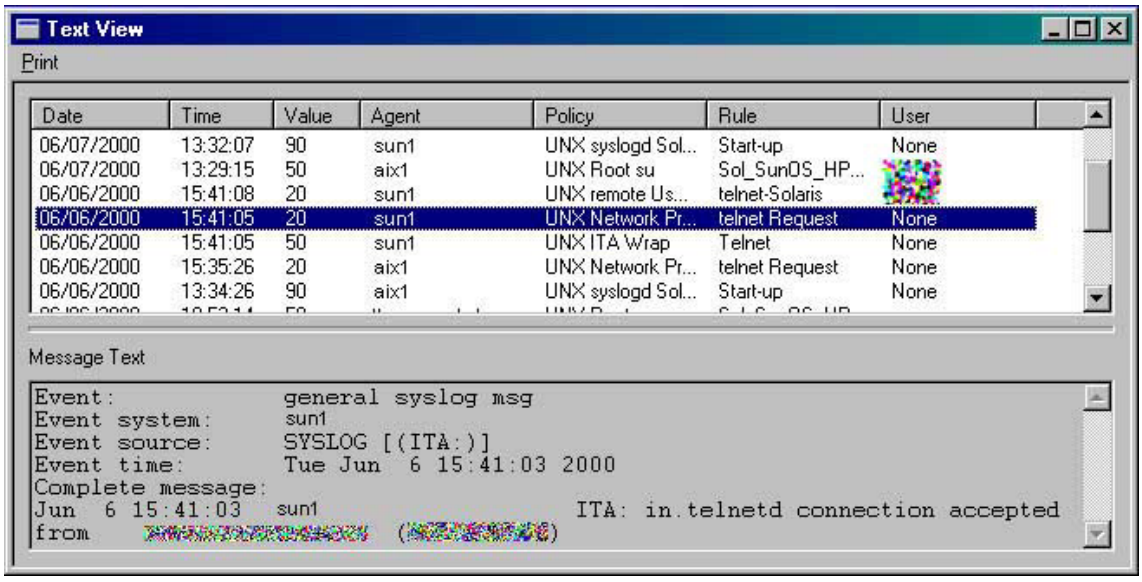

Figure 3.2: A screenshot of ITA view. In the upper window the list of monitored alerts is presented, while some description available in the lower window. Host addresses have been hidden for obvious reasons.

- ITA Admin on Win98 (win1)
- ITA View on Win98 (win1)

When all components are installed, things are not done. The following sources can be monitored:

- Syslog messages contains operating system messages.
- Wtmp collects login and user-process information.
- Btmp collects failed login information. It is not available on all UNIX platforms.
- process accounting collects user process information and numerous other processing activities.
- C2 audit logs collect various types of system calls and events depending on the configuration in the system (HP-UX, solaris and OSF/1).
- It is possible to look at external log files as the HTTP log file or other application logs.

The syslog file (/omniguard/ita/system/system-name directory) receives event data from the syslog daemon. Events related to the activated inet daemon services [12] can be captured only if a service wrapper is associated with the syslog daemon. TCP wrapper was already installed on aix1. We installed the ITA wrapper on sun1. The other UNIX collectors receive event data from the UNIX operating system. A collector daemon, collogd, reads the collector files and pipes event data to the agent. Then the agent processes events according to its activated policies. Figure 3.3 shows the events collection under unix.

The setup does not activate the C2 auditing [13]. It has to be done manually. First the unix system must be configured. Then it must be initialized in the ita daemon.

The C2 audit process logs numbers of events. Each event belongs to a specific class. During the unix C2 configuration phase we must decide the classes we want to audit. Here are the default classes for a Solaris system.

- no: invalid class
- fr: file read
- fw: file write
- fa: file attribute access
- fm: file attribute modify
- fc: file create
- fd: file delete
- cl: file close
- pc: process
- nt: network
- ip: ipc
- na: non-attribute
- ad: administrative
- lo: login or logout
- ap: application
- io: ioctl
- ex: exec
- ot: other
- all: all classes

It is not simple to decide which class should be audited. The C2 audit belongs to a lower level of the system and can consume lots of resources. For example the auditing of all file accesses will quickly increase the log file's size. However auditing modifications (write, create, delete) on root's files may help to detect some attacks. This is why these modifications should be supervised. Here are the classes we have configured:

- fw, fc and fd for user root
- na, ad, lo and ex for all users

ITA provide no information about how to configure the C2 audit and which classes should be activated. That means clearly that depending on the configuration set up, some events can be missed without any advice from ITA.

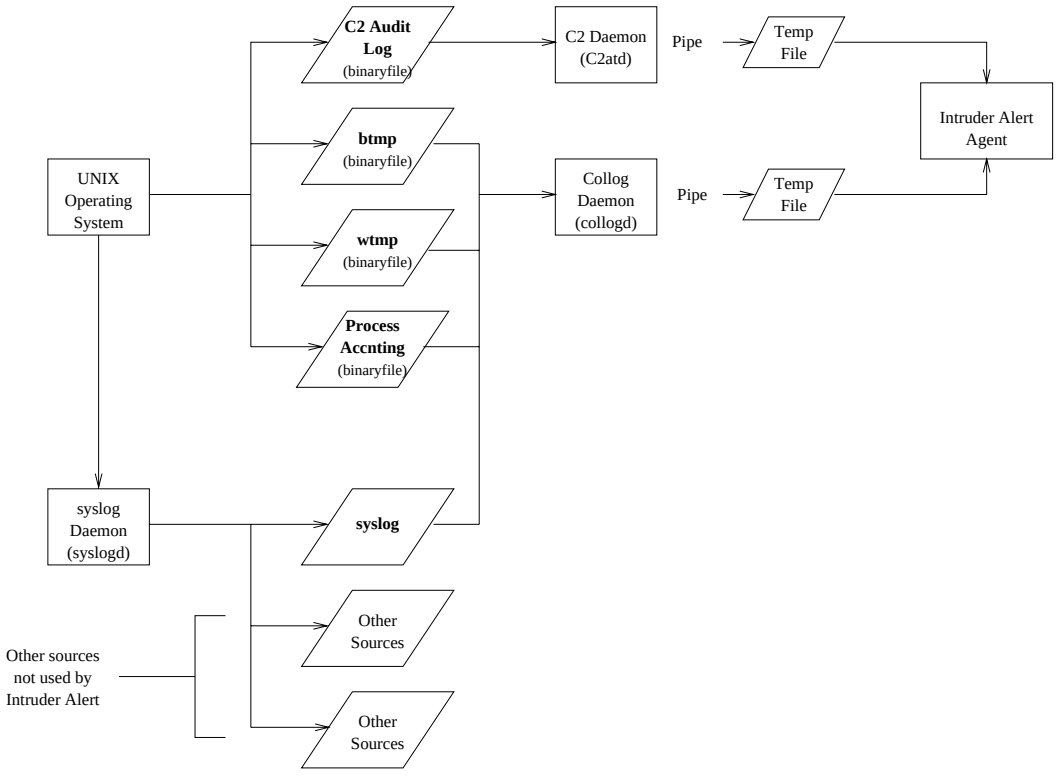

Figure 3.3: ITA Event Collection on a UNIX source [14]

3.1.2 Policies and Documentation

Policies can be imported from several sources. Some policies are present when installing ITA. Then you can download new policies from the Axent web site and it is also possible to create our own policies. Axent gives two kind of policies; generic policies available for all systems and policies composed for a specific system (unix, NT or NetWare).

Unix And Generic Policies

Only generic policies and policies related to unix systems have been tested. They are either default policies delivered with the installation or downloadable ones. Axent offers no policy to monitor the C2 audit. To check what kind of events could be monitored thanks to our C2 audit configuration, we have created a C2 collector policy to catch all the C2 events. Basically it contains a "*" in the *select* clause and results are appended to a file. However we finally decided that we do not want to spend too much time in creating our own C2 policy. Of course, even if we have not used this source, the C2 audit has some advantages. Information comes from the kernel itself and not from a deamon like the syslog. That means that the C2 events are more safe and of course quicker.

In table 3.3 the tested policies are presented, some of them have been modified during the project. Please note that a policy can work for an AIX but not for a Solaris. The last three policies are related to the ITA wrapper and therefore have only been activated on sun1.

Attacks Signatures

Hereafter some of Axent attack signatures[3] are presented. The goal is to give the reader an idea about the defensive strategy proposed by axent. We will show that no direct solution is proposed to detect the local exploits. ITA can detect following exploit only through eventual files changes.

cron Exploit

Attack Profile:
A UNIX process-scheduling tool known as "cron" can be used for more than just performing routine system maintenance. Rumors gathered from the hacker underground imply that cron was used in the September 13th attack on the *New York Times* web site...
AXENT Strategy:
Intruder Alert can be configured to monitor cron files and notify the security administrator when any changes occur. The following files should be monitored:

/usr/spool/cron/crontabs/*
/usr/lib/cron/cron.*
...

NIS/NIS+ Flooding

Attack Profile:
An NIS/NIS+ vulnerability has been discovered that has the potential to flood a local network with NIS traffic and possibly shut it down.
...

[3] http://www2.axent.com/swat/swat/03a_atk.htm

Policies	Description
ITA Reports	Generates ITA Reports (Agent load Report,agent Policy Report, Agent Active Datastream Report) when an ITA command is sent via ITA View
Shared Notification	Record actions in ITA View for FYIs, Alerts, Emergencies
UNX[1] Critical Files	Monitors files every 8 hours(long term) and a small group of highly critical files every 20 seconds(short term). The groups long term and short term are defined within text file (uxcrit_L.lst, uxcrit_S.lst) and can be modified[2]. These two groups are default groups, but we can add other groups with different time frames. This policy detects when, within the time frame, a file is deleted, modified or created. This is done by using MD5 and other checksums.
UNX Failed ftp Login	Detects a failed ftp login (not root) on HP
UNX Failed ftp Root	Detects a failed ftp root login on HP
UNX Failed remote	Detects failed remote login (Xwindows login on Solaris, telnet and rlogin on HP)
UNX Failed rlogin	Detects a failed rlogin (root or user) on Solaris
UNX Failed su	Detects failed su to another user (not root)
UNX Failed su Root	Detects failed su to root
UNX Failed telnet	Detects a failed telnet login (not root) on solaris
UNX ftp Root Login	Detects a failed ftp root logins on Solaris, SunOS, HP
UNX ftp User Login	Detects ftp user logins on Solaris SunOS, HP
UNX local Root Login	Detects local root login on HP, AIX
UNX local User Login	Detects local user logins on HP, AIX
UNX Netprobe Lite	Detects a character generator request, an echo request, an anonymous ftp login from a Satan probe, a remote shell deamon connection from an illegal port, a ftp deamon connection from an illegal port, and a virtual memory error.
UNX remote Root Login	Detects root logins (Xwindows, telnet and rlogin root login on Solaris and AIX, . . .)
UNX remote User Login	Detects user logins (Xwindows, telnet and rlogin user login on Solaris and AIX, ...)
UNX Root su	Detects an su to/from root
UNX su to another	Detects an su to another user (not root)
UNX SYN Flood	Detects an SYN Flood system attack
UNX syslogd Sol_HP	Detects startup/graceful-shutdown of syslog daemon on Solaris and HP
UNX System Problems	Detects some system problems
ITA Errors	Reports some ITA error like a connection problem
ITA Status	Generate some internal messages
UNX Network Probes	Detects someone probing your network for vulnerabilities. (may require system configuration changes)
UNX ITA Wrap	TCP-UDP wrapper that reports a inetd service connection (comsat, exec, finger, ftp, login, shell, talk, telnet, uucp)
UNX 3 Failed Root Logins-on3	Detect three failed ftp, remote login or su from Root user
UNX 3 Failed User Logins	Detect three failed ftp or su

Table 3.3: UNIX and Generic Policies

The command "finger username@system.domain.com" asks the system to look up information about a username. If the username cannot be determined from the local password file, the system will try to find it in the global NIS password table.
...
NIS flooding will result, likely cause network timeouts and dropped packets as well.
AXENT Strategy:
The AXENT Technologies, Inc. information security SWAT team recommends that finger services be disabled on all systems configured to use NIS

Syslogd

Attack Profile:
If the syslog daemon is not running, events will no longer be recorded to syslog and the security administrator will be unable to monitor system activity. Killing the syslog daemon at the start of an intrusion and restarting it afterwards helps intruders "cover their tracks" and hide their progress.
AXENT Strategy:
AXENT's Intruder Alert can detect when the syslog daemon is not running, when it has been started, and in some cases[4] when it has been killed

Buffer overflow Vulnerability
in statd(1M)

Attack Profile:
A buffer overflow has been discovered in the statd(1M)program that will allow a local or remote user to gain root access. For example, it is possible[5] for a statd attack to add an entry to the /.rhosts file...
AXENT Strategy:
AXENT's Intruder Alert can detect critical system file tampering, such as ./rhosts modified...

These Axent strategy examples show the defensive solutions proposed by Axent against several well known exploits. Axent proposed no real solutions; if the finger service is disabled, a NIS-Flooding attack will be avoided. But we expected another solution to detect and defend this attack. Further only a normal syslog daemon shutdown is detected (Policy UNX syslogd), but not a forced one (kill -9). Lots of exploits like a cron exploit [9] or a statd buffer overflow vulnerability [9] is detected through file changes. That would mean a hacker who has gained root access with statd, can be detected only if he is doing some changes on the files.

[4] by "in some cases", understand that when syslog is stopped by a normal way, a shutdown is detected, but not when it is stopped by a kill -9

[5] In fact, it is possible, but this is not the only thing...

3.1.3 Results

Events related to each policy have been generated from the local network through an Ethernet connection or on the local machine itself. The results obtained are presented in tables 3.4 and 3.5. The field *doc* refers to the documentation. It indicates whenever the policy should work. The field *detected* shows a "✓" if the policy works, "—" indicates that the event was not detected and "±" indicates that some rules of the policy work but not all of them. The majority of the policies require no more explanation, but if the *comments* field contains *"see explanations"*, there are precisions below. This is especially the case for policies made of several rules.

policies	doc	detected	source	comments
UNX Critical Files	✓	✓		
UNX Failed ftp login				
UNX Failed ftp Root				
UNX Failed Remote	✓	✓	btmp	
UNX Failed rlogin	✓	✓	wtmp	
UNX Failed su	✓	✓	syslog	
UNX Failed su Root	✓	✓	syslog	
UNX Failed telnet	✓	✓	wtmp	
UNX ftp Root login	✓	✓	wtmp	
UNX ftp user login	✓	✓	wtmp	
UNX local ROOT login				
UNX local User login				
UNX Netprobe Lite	✓	—		see explanations
UNX Network Probes	✓	±	syslog	see explanations
UNX remote Root Login	✓			Cannot be tested root remote login denied
UNX remote User Login	✓	✓	wtmp	
UNX Root su	✓	✓	syslog	
UNX su to another	✓	✓	syslog	
UNX SYN flood	✓	✓	netstat	
UNX syslogd Sol_HP	✓	±		shutdown only
UNX system Problems	✓	—		
UNX 3 failed Root logins	✓	✓	syslog	see explanations
UNX 3 failed user logins	✓	✓	syslog	see explanations
UNX ITA Wrap	✓	✓	syslog	

Table 3.4: Results on Solaris

UNX Netprobe Lite

This policy does not work properly. It should detect one of these following events; a character generator request, an echo request, an anonymous ftp login from a Satan probe, a remote shell deamon connection from an illegal port, a ftp deamon connection from an illegal port, and a virtual memory error. This policy should catch all the events by pattern matching. The matching is done with strings like "*daytime*", "*chargen*", "*ftpd: connection from * on illegal port*", and others. We have not succeeded in reproducing them even with a Satan probe on the host. Either the syslog is not well configured or these strings are not the correct ones for the systems tested. As we used the wrapper given by Axent on sun1, it should work. No further documentation is given, therefore, we cannot say more.

In addition, it is really simple to generate false alarms. This can be done by exploiting a

Policies:	doc	detected	source	Comments
UNX Critical Files	✓	✓		
UNX Failed ftp login				
UNX Failed ftp Root				
UNX Failed Remote				
UNX Failed rlogin				
UNX Failed su				
UNX Failed su Root				
UNX Failed telnet				
UNX ftp Root login				
UNX ftp user login				
UNX local ROOT login	✓			cannot be tested
UNX local User login	✓			cannot be tested
UNX Netprobe Lite	✓	—		see explanations
UNX Network Probes	✓	±	syslog	see explanations
UNX remote Root Login	✓	✓	wtmp	
UNX remote User Login	✓	✓	wtmp	
UNX Root su	✓	✓	syslog	
UNX su to another	✓	✓	syslog	
UNX SYN flood	✓	✓	netstat	
UNX syslogd Sol_HP	✓	✓		
UNX system Problems	✓	✓		to be tested

Table 3.5: Results on AIX

weakness of the patterns. In fact, there are several only one-word pattern like "daytime", "chargen" or "echo". Thus a telnet request with "echo" as username generates a netprobe lite alert.

UNX Network probe

This policy is designed to detect some network probing. It is composed of several rules; collector, ftp request, telnet request, uucp/sendmail request, lpd/remsh/echo request, network probe, possible normal probe, possible heavy probe and heavy probe end. Table 3.6 contains the related clauses for each rule. As explained hereafter, some rules are only temporary states not recorded to ITA View. However to verify them, they have been modified to be recorded in ITA View. For a better understanding, there are some explanations for each rule below.

Collector: this rule is generally desactivated. It is used to collect all events and append them to a file. Basically it is done by setting the *select* field to "*". We can also look at this rule as an aggregation of all activated source events.

Ftp request: this detects an ftp request. The *select* clause referres to the messages generated by syslog. When the pattern matches, a flag *Ftp request* is raised for a 2-minute period. If a ftp request occurs when the flag is already up, nothing happens. That means the time is not reset. This way of processing will be discussed later.

Telnet request: the processing is similar to ftp request rule.

Uucp/sendmail: this rule detects an uucp connection or a sendmail attack (wiz, debug). We have launched the sendmail attack from Nessus [8]. To get an alert, the last pattern has been added. That means the patterns used were not the correct one for our syslog messages. This illustrates the importance of having good patterns to process the matching. Surly it can depend on the wrapper configuration associated to the syslog deamon.

Lpd/remsh/echo request: this rule looks for three events; lp request, remsh request and an echo requests. A remsh is a simple rsh on HPUX and AIX. This is why the pattern should be different. In fact, a remsh on aix1 generate the same syslog message as an rsh. The pattern to catch this event should be *rshd[*]* for aix1. Of course, this string depends on the system we are looking at. The echo service is defined as an intern service in the inet daemon and does not generate a syslog message. Thus it cannot trigger an alert.

Network probe: this rule is a temporary state. It is activated by three flags up: the Ftp_flag, the Telnet_flag and the USend_flag. A timer (network probe mark) starts for 30 seconds. Under an lp/remsh/echo request, the timer is canceled and the possible heavy probe activated. If the timer reaches its deadline, the possible normal probe rule is activated.

Possible normal probe: it occurs at the end of the timer Normal probe mark.

Possible heavy probe: it occurs when four different requests have been detected: ftp, telnet, uucp/sendmail, and echo request.

Heavy Probe end: we can only say this rule is activated if an event matches with *ftp_login* while we are in the possible heavy probe state. All the other flags are held down.

Table 3.6: UNX Network probe policy

Collector

Select	Ignore	Action
*	-	append to a file

FTP request

Select	Ignore	Action
string message: *ftpd connection* *ftpd[*]: connection* *inetd[*]: ftp/tcp: Connection* *inetd[*]: ftp[*] from*	Ftp_Flag	Ftp_flag(2min)

Telnet request

Select	Ignore	Action
string message: *telnetd connection* *remote_login* *inetd[*]: telnet/tcp: Connection* *inetd[*]: telnet[*] from*	Telnet_Flag	Telnet_flag(2min)

Uucp/sendmail request

Select	Ignore	Action
string message: *uucpd connection* *uucp*local_login* *sendmail[*]: "wiz" command* *sendmail[*]: "debug" command* *inetd[*]: uucp/tcp: Connection* *inetd[*]: uucp[*] from* *sendmail[*]:*: "debug" command** [6]	USend_Flag	USend_flag(2min)

continued on next page

[6] this string has been added

Table 3.6: *continued*

Select	*Ignore*	*Action*
Lpd/remsh/echo request		
Select	Ignore	Action
string message: *lpd[*]: * *rlpdaemon connection* *remshd[*]: Connection* *inetd[*]: printer/tcp: Connection* *inetd[*]: echo[*] from*	echo_Flag	echo_flag(2min)
Network probe		
Select	Ignore	Action
Flag Telnet + FTP + USend	Flag Heavy or Normal	Normal_flag(2min) Normal Probe mark (30 seconds)
Possible Normal Probe		
Select	Ignore	Action
Timer Normal Probe mark	Flag Heavy or echo	raised down Flags: Telnet, Ftp, USend, Normal
Possible Heavy Probe		
Select	Ignore	Action
Flag echo + Normal	Flag Heavy or echo	raise down all flag without heavy probe Flag Heavy(2min) cancel timer Normal probe mark
Heavy Probe end		
Select	Ignore	Action
Flag Heavy + string: *ftp_login*		raise down all flag Flag Heavy(2min)

The ignore clause

As mentioned before, a request is ignored when the corresponding flag is already up. With an example we present the weakness of doing it this way.

Let's take the example of the network probe rule with the following scenario:

- 0 : uucp/sendmail request
- 1'30: telnet request
- 1'50: uucp/sendmail request
- 2'10: ftp request

The Network probe rule occurs when the three flags ftp request, telnet request and uucp/-sendmail request are raised. These flags are raised for 2 minutes once the event occurs. With the default configuration, the timers are not restarted within the 2-minute window, even if another request occurs during that time frame. Thus with the scenario exposed before, the timer of the uucp/sendmail flag is not restarted when the second request arrives at 1'50. At 2'00 this flag is reseted and when the ftp request appears, no network probe message is generated. However the last three requests (1'30 telnet, 1'50 uucp/sendmail, 2'10 ftp) are in a sliding window of less than 2 minutes. A network probe should have been triggered depending on what we want to do. Either we want to catch the three requests in a 2-minute period since the first one has occured, or we want to catch the three requests in a 2-minute sliding window. The default configuration of this policy works like the first solution. Following, a solution is proposed to activate an event when 3 different events occur in a defined period of time.

Three Different Events in a Defined Period of Time

The goal is to generate an alert when three different events A,B and C occur in a defined period of time X. The order occurence must not be important. The solution is presented in table 3.7. It is similar to the previous, except for the *ignore* clause.

rule	Clauses	
Event A	Select	select system messages
	Ignore	-
	Action	raise flag A for X seconds
Event B	Select	select system messages
	Ignore	-
	Action	raise flag B for X seconds
Event C	Select	select system messages
	Ignore	-
	Action	raise flag C for X seconds
Alarm	Select	Flag A,B and C
	Ignore	-
	Action	reset all flags

Table 3.7: 3 Different Events in a Defined Period of Time

The reason Axent has not proceed in such way is certainly because this solution consumes more resources. In fact, by setting an ignore clause, the number of fired events is reduced compared to this solution. That clearly means better performance, because an event must be checked with less fired rules.

UNX 3 failed Root Logins-on3

This policy is based on the following failed events: ftp, remote, and su. It has been tested on Solaris with "failed su" events. A limitation due to syslog messages has been observed. A syslog is generated only if it is different to the previous one, otherwise it is kept in a kind of stack while no other messages occur. Then all the same syslog message are grouped and the following message appears:

"...last message repeated X times"

It is possible that the 2nd and 3rd fail are sometimes not immediately reported in the syslog. This causes troubles to the efficiency of this policy, because an alarm can be missed. In fact, the

1st failed flag can be already down when the 2nd and 3rd attempts are reported. If other events between each "failed su" are generated, it works well.

UNX 3 failed User Login

This policy is based on either failed su, or failed ftp. It is composed of three rules; 1st failed, 2nd failed, and 3rd failed. Table 3.8 contains the different clauses. The default configuration does not work well. Three failed su (or ftp) give four alerts being: 1st failed, 2nd failed, 3rd failed, and 1st failed. In fact, two events (1st and 3rd) can be active at the same time. This is illustrated in figure 3.4. Two solutions are proposed to improve the default setting of this policy:

1. Generate an alert if three failed attempts appear in a 1-minute period once the 1st one occured.
   Solution: change the *select* clause of the 3rd rule to: select messages + **1st flag** + 2nd flag
2. Since the 1st failed attempt, the second has T1 minutes to occur and then the third has T2 minutes. That means a maximum period of (T1+T2) minutes.
   Solution: change the *ignore* clause of the 1st rule to: 1st flag **or 2nd flag**

rule	Clauses	
1st	Select	select system messages
	Ignore	flag 1
	Action	raise flag 1 for 1 minute
2nd	Select	select system messages + flag 1
	Ignore	flag 2
	Action	raise flag 2 for 1 minute
3rd	Select	select system messages + flag 2
	Ignore	-
	Action	Alarm + reset flag 1 and 2

Table 3.8: UNX 3 failed User Logins

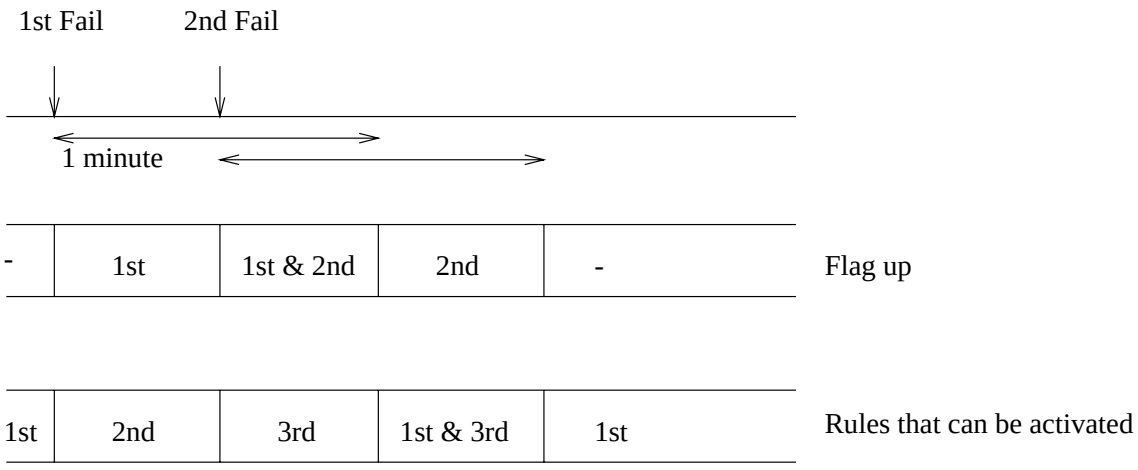

Figure 3.4: 3 Failed User Logins, schema

UNX SYN Flood

This policy can be activated only through an ita status message. That means ITA status policy must be activated in order to activate it. Basically a shell script is running periodically. It does a netstat command and handles the results to know if some SYN requests are on process. Depending on the number of SYN packets received an alarm is generated.

Network Attacks

Several attacks have been launched to sun1 to see the reported events. We have tested network scanners like ISS scanner, Satan and Nessus. The event reported are shown in table 3.9. Other attacks like bonk [15], echo [15], and nestea [15] generated no alert. To detect network attacks Axent recommends to integrate NetProwler with Intruder Alert. NetProwler is a network-based IDS. The only outside events that Intruder Alert can detect are ftp request, rsh request, telnet request, and SYN flooding.

Conclusion

As this section shows, ITA gives us the possibility to interact directly with the rules. The rules defines are more or less complicated depending on the event we wanted to catch. Lot of tools like timer, flags, etc... can be used giving us a lot of flexibility to do more than a basic pattern matching. That means to express rules to deal with the alerts. The results obtained will be discussed more deeply hereafter, but we can already mentioned that the reported events are mainly related to the normal host activity meaning that the real local exploits are only detected through eventual file changes.

3.2 ISS: RealSecure v3.2 - System Agent

Realsecure is an IDS composed of two modules:

- Detectors: software that looks for attacks and can generate responses.
- Management console: graphical user interface to manage detectors and collect alerts in a database.

The detectors are either network engines or system agents. Both are looking for different events and can send them to the console. The network engine is the network-based part of Realsecure and will no longer be discussed. The system agents monitor the activity on an individual host.

The system agent runs on WindowsNT or on SPARC Solaris. It is an application which typically uses less than 1% of the CPU. The system agent and the network engine cannot be installed on the same computer. The console should run on a different system than the network engine, since both can be processor and memory intensive. The console runs only on WindowsNT. Its needs in resources depend on the number of detectors which are managed.

Several responses can be launched by the System Agent when an alert is triggered. First, a notification with an alarm on the console, via email or through an SNMP trap. Alerts can also be stored in a database. Some active responses are also possible; e.g. a user account can be disabled.

3.2.1 Testbed Setup

Here is the architecture we have set up:

ISS Scanner

Policy	rule
UNX ITA Wrap	telnet request ftp request
UNX Network probes	telnet request uucp/sendmail request = possible normal probe Ftp request
UNX Failed telnet	solaris
UNX 3 failed user login	remote_only_fail

NESSUS

Policy	rule
UNX ITA Wrap	telnet (a lot) FTP (a lot) finger (a lot) Exec
UNX Network probes	telnet request uucp/sendmail request = possible normal probe (2x) Ftp request
UNX Failed telnet	solaris (a lot)
UNX 3 failed user login	remote_only_fail (a lot)

SATAN

Policy	rule
UNX ITA Wrap	telnet FTP finger (a lot) Exec Login Uucp Shell
UNX Network probes	telnet request uucp/sendmail request = possible normal probe Ftp request
UNX Failed telnet	solaris

Table 3.9: Events reported under ISS, Nessus, and Satan on sun1

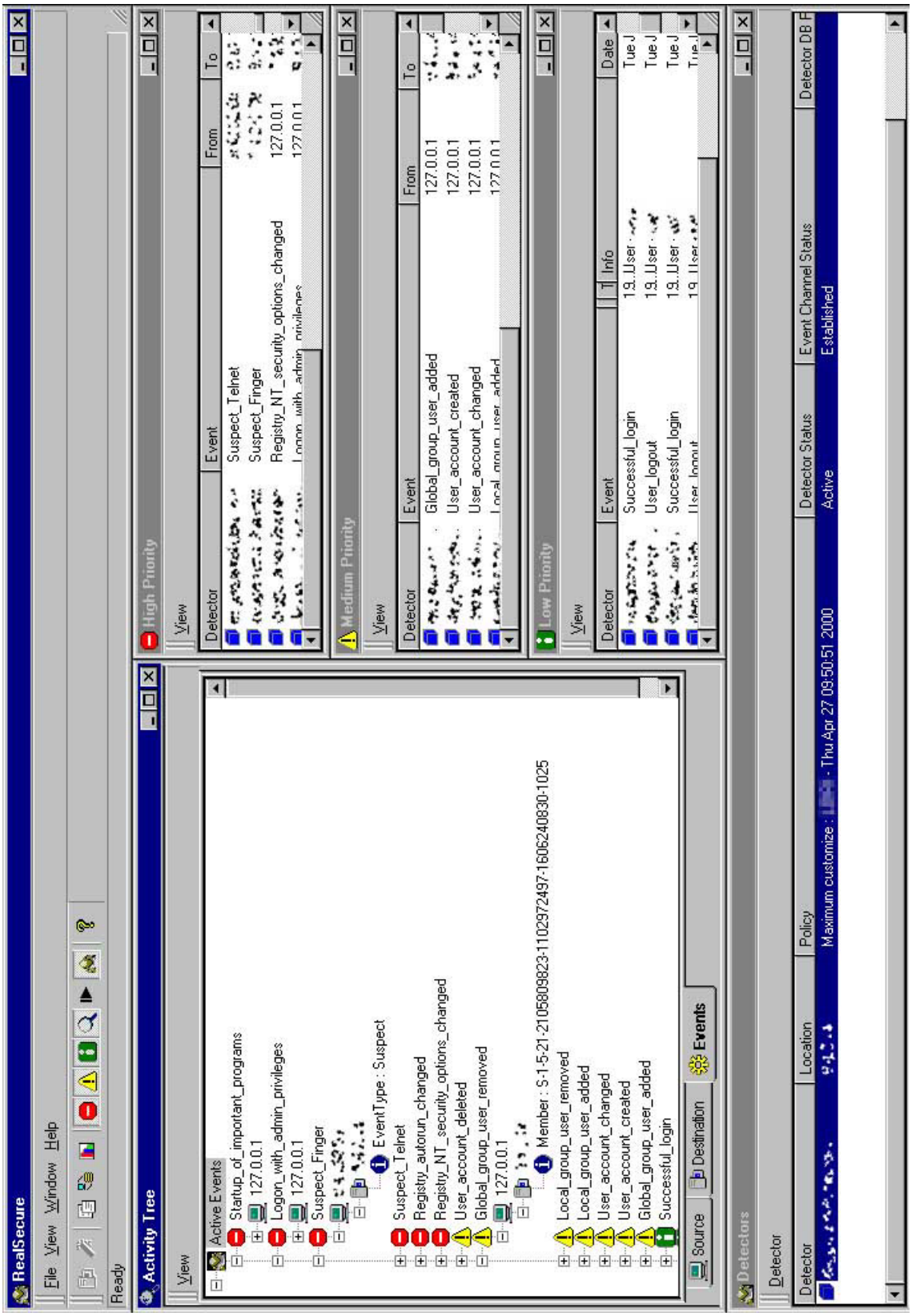

Figure 3.5: A screenshot of the Realsecure Console. Host addresses have been hidden for obvious reasons.

- Console management on WindowsNT (win2)
- System agent on WindowsNT (win3)

The installation is done through a windows setup. Nothing special is required. To transmit data from the detectors to the console and vice-versa, RealSecure can use encryption. Win3 belongs to the IBM global network. This is the reason why only the local administrative privileges have been used. Events related to global administrative actions have not been tested.

3.2.2 Policies

As mentioned before, policy does not mean the same for all companies. A RealSecure policy defines the global configuration of the detectors. The policy specifies what kind of system activity must be detected, the priority of each event and the response. RealSecure delivers three pre-configured policies for the system agent: default, light, and maximum. The maximum policy enables all events. The pre-configured policies cannot be edited; they will never be lost. To edit a policy, we must create a custom one. The custom policy is a copy of an existing policy which can be edited. We have tested all events in the maximum policy. However not all of them were sent to the console. This is the reason why we have to create a custom policy (maximum customize) derived from the maximum policy. This custom policy is similar to the maximum one except that all events are sent to the console. That allows us to see the results in the console.

There are several kinds of events: security events, syslog events, suspect connection events, and user-defined events. Only the security events and suspect connection events have been tested. In fact, there is no syslog on NT and we do not want to create user-defined events. Events defined are presented with the results.

3.2.3 Results

There are two kinds of events tested: security events and suspect events. The security events are "atomic" NT events related to the action happening on the local machine, while suspect events monitor different requests from the outside. Results are presented in table 3.10.

Table 3.10: ISS System agent events

Event Name	generated	Comment
Successful_login	✓	
User_logout	✓	
Guest_user_login	✓	
Use_of_user_right	✓	e.g. event viewer
Password_change_failed	✓	
Password_change_successful	✓	
Failed_login-account_locked_out	✓	
Failed_login-bad_username_or_password	✓	
Failed_login-password_expired	✓	
Failed_login-time_restriction_violation		
Failed_login-account_disable	✓	
Failed_login-account_expired		
Failed_login-net_logon_not_active		
Failed_login-not_authorized_for_-console_login		

continued on next page

Table 3.10: *continued*

Event Name	generated	Comment
Failed_login-not_authorized_for_-this_type_of_login		
Failed_login-unknown_error		
Logon_with_admin_privileges	✓	
Logon_with_special_privileges		
Global_group_changed		
Global_group_created		
Global_group_deleted		
Global_group_user_added	✓	
Global_group_user_removed	✓	
Local_group_changed	✓	
Local_group_created	✓	
Local_group_deleted	✓	
Local_group_user_added	✓	
Local_group_user_removed	✓	
Account_policy_changed	✓	
User_account_changed	✓	
User_account_created	✓	
User_account_deleted	✓	
User_right_granted	✓	
User_right_revoked	✓	
Audit_log_cleared	✓	
Audit_policy_change	✓	
User_added_to_global_admin_group		
User_added_to_local_admin_group	✓	
User_admin_right_granted	✓	
User_admin_right_revoked	✓	
Trusted_domain_added		
Trusted_domain_removed		
Startup_of_important_programs	✓	e.g. user manager
Privileged_service_called	✓	e.g. changing a password
Registry_autorun_changed	✓	
Program_execution_started	✓	
Program_exited	✓	
Out_of_virtual_memory		
Disk_space_shortage		
Brute_force_login_attack	✓	6 failed logins or more
Brute_force_login_likely_successful	✓	5 failed logins followed by a correct password
Change_password_attack	✓	
Change_password_attack_likely_successful	✓	
Registry_eventlog_settings_changed	✓	
Registry_NT_security_options_changed	✓	
Registry_remote_edit_changed		
Probing_of_important_files		
Changes_to_important_files	✓	
Failed_change_of_important_files		
Config-log_files_deleted		

continued on next page

Table 3.10: *continued*

Event Name	generated	Comment
Config-log_files_delete_failed		
Authentication_package_loaded		
Logon_process_registerfed	✓	
Some Exchange Events		
Some LDAP Events		
Some SQLServer Events		
Some Oracle Events		
Some Sybase Events		
Some MSSQL Events		
Suspect_portscan	✓	
Suspect_FTP	✓	
Suspect_IMAP	✓	
Suspect_Netstat	✓	
Suspect_POP3	✓	
Suspect_POP2	✓	
Suspect_SMTP	✓	
Suspect_Systat	✓	
Suspect_Telnet	✓	
Suspect_Whois	✓	
Suspect_WWW	✓	
Suspect_Finger	✓	
Suspect_Time	✓	
Suspect_SSH	✓	
Suspect_Sunrpc	✓	
Suspect_Netbus	✓	

Suspect connection

RealSecure is supposed to generate an alarm related to suspicious connections. A suspicious connection is a request to a service (ftp, telnet,...) which is not installed. The services defined are as follows: ftp, imap, netstat, pop3, pop2, smtp, systat, telnet, whois, www, finger, time, ssh, sunrcp, and netbus. A suspicious connection means a request to a not installed service.

We have to generate these alerts by launching several network scanners: Nessus [8], 7th Sphere Portscan 1.1 [16], and Strobe [17]. However, a telnet on the default service ports will also generate an alarm. We can conclude that Realsecure dosen't check the request and only looks at the ports. In table 3.11 we present the results we got with the scanners and the default ports used.

For a better understanding of these suspicious connections, we have done some more experiments:

- If no services are running when the agent is started, all the requests to the default ports mentioned before are reported as suspicious connections. The agent replies even with a warning message saying that this host is protected by ISS. Then we have tried to install a telnet server (Fictional Deamon v3.3 [18]), however, it was impossible. The port seems to have already been binded by RS Agent. To install a telnet server, the agent must be halted. This is certainly the same case for all the other services.
- A telnet server is installed on port 23 (default) and the agent is started. An alert is triggered:

Event Name	Nessus	7th Sphere	Strobe	Telnet
Suspect_portscan	✓	✓	✓	
Suspect_FTP	✓	✓	✓	21
Suspect_IMAP		✓	✓	220
Suspect_Netstat	✓	✓	✓	15
Suspect_POP3	✓	✓	✓	110
Suspect_POP2		✓	✓	109
Suspect_SMTP	✓	✓	✓	25
Suspect_Systat	✓	✓	✓	11
Suspect_Telnet	✓	✓	✓	23
Suspect_Whois		✓	✓	43
Suspect_WWW	✓	✓	✓	80
Suspect_Finger	✓	✓	✓	79
Suspect_Time		✓	✓	37
Suspect_SSH	✓	✓	✓	22
Suspect_Sunrpc	✓	✓	✓	111
Suspect_Netbus	✓	✓	✓	12345

Table 3.11: Events monitored under Nessus, 7thSphere Portscan1.1, Strobe, Telnet on port X

"Detector_Error: Port 23 already in use on system". Then the requests on port 23 generates no alarm. This is normal because a telnet service is installed. It is also possible to install the telnet server on another port (12345 for example).

- NukeNabber [19] is a TCP/UDP port listener used to securify ports. We launched it when the agent was running and we got these messages:

```
[05/08/2000 09:00:46.708 GMT+0200] Port 21 (tcp) is already in use.
[05/08/2000 09:00:46.718 GMT+0200] Port 15 (tcp) is already in use.
                                   ...
[05/08/2000 09:00:46.888 GMT+0200] Port 111 (udp) is already in use.
[05/08/2000 09:00:46.898 GMT+0200] Port 12345 (tcp) is already in use.
```

When the Realsecure agent is started, it looks at all the default ports. A detector error message is generated for the ports already in use whereas it listens on all unused ports to protect them. The agent does not periodically verify if the ports that were used at startup time are still in use. Please pay close attention to the following senario:

1. A service is installed on port X when the RS Agent starts.
2. The service is stopped because of a change in the company's policy.
3. Port X is no more protected and no service is installed behind it.

3.3 General Results

In this section, some general results based on the previous ones are discussed. Two different IDSes have been tested on different systems: UNIX and NT. As mentioned in the background chapter, IDSes take their information from sources available on the host; that means mainly the syslog messages on unix and from events logs on NT. The results depend on the system configuration as well as the IDS configuration.

Two kinds of events are reported; local events related to the system and network events. The majority of local events are well detected. It is not the same for network events. That means the only network events detected are some request on predefined ports. We expected that the IDSes look at the TCP/IP stack (tcpdump) to check and analyze the received network packets. It would allow us to do some more comparisons between the network-based IDS and the host-based. In fact, it would be useful to know which packets travel on the network and which ones are accepted by the host.

One more weakness is that no real local exploit is detected. We think about exploits like statd, cmsd, loadmodule and so on [15]. The only way to detect them is through file changes. That would mean that a hacker who has gained root access can be detected if he is doing some changes on the files. Of course, it is not so simple to keep trace from these exploits, but it might be the job of a host-based IDS.

The events are caught by pattern matching. The efficiency of the IDSes depend on them. The use of patterns cannot really suffer from false alarms, because they are relatively precise; we have already mentioned that one-word patterns matching should be avoided. However some false alarms can originate from the sources themselves. In fact on UNIX system the syslog daemon is opened by default. It accepts remote messages from the network. To protect against such fake syslog, there is an option witch rejects the remote syslog; you can also configure a firewall to prevent from connections to the syslog deamon port. By doing forwarding syslog on a host protected by an IDS, this one can also generate alerts related to events occurring on another host. In this case you must take care of the results obtained.

Chapter 4

Integration

Nowadays, a lot of network products offer the possibility to manage network devices from a single management console. For example, it is possible to configure a router or to get information from a SNMP agent.

4.1 Goals of Integration

For the intrusion detection community, integration is more than representing results of distributed devices in a single console. The integration of the IDSes leads quickly to the concept of correlation. A definition of correlation is given by Edward Amoroso [20]: *"Intrusion correlation refers to the interpretation, combination, and analysis of information from all available sources about target system activity from the purposes of intrusion detection and response"*. As information sources we are going to consider the results of different IDSes. That means we do not consider all real sources like network packets or log files. This is the job of the IDSes. Based on the following observations, this section aims at demonstrating that the correlation is fundamental for information processing:

- Commercial intrusion-detection systems often generate a huge amount of data. Analyzing them directly without *filtering functions* is difficult and is very time consuming
- Sensors may generate a lot of false alerts per day. All this data should not always be considered as alert. There is lot of false alarms [21] and lot of normal local events.
- There is no universal IDS. That means the IDSes have not the same strength and weakness. They information sources and process can be different. We can improve the detection by combining different IDS products.

The goals of the integration of the IDSes in a common framework is to enhance the results in operating some more processes. In the following sections, we present the correlation needs for the network and host-based IDSes. Of course, both should be taken into account for better performances.

4.1.1 Correlation

The IDSes do not have the same strengths and weaknesses. They do not all have the same information sources and do not treat information in the same manner. That means the different IDSes do not miss and do not report the same attacks. Thus it is possible to reduce the false alarms' rate by building rules which contains information to correlate the results. The information

can be of different types. For example, it can be a confidence level which is associated to each alarm. The lower the confidence level is, the higher the probability to get a false alert is. In addition, it is also possible to increase the global attacks' signatures recognized by summing up all single IDS's signatures.

4.1.2 Host-based event interpretation

Almost all events generated by host-based IDSes do not concern direct attacks. They describe user actions that happen on the host. The majority of these actions should not be considered as alerts. Thus we should take into consideration the context in which an event occurred. In fact, the common approaches analyze alarms in isolation, whereas the alarm context could enhance decision making [22]. Following are some examples where alarms should be triggered:

- A "failed password" happens often in every environment and the majority of them should be ignored. However, if such an event happens in the middle of the night when the computer room is closed, an alert should be generated.
- If a strange behavior (e.g. port scan) is detected on a single host, it might be a simple false anomaly and the security manager can ignore it. However, if such a behavior is detected on several hosts of the network, someone is certainly probing it.
- A "failed su" in some environment can occur many times a day, which in others is not normal and should trigger an alert.
- If the system is under attack, i.e. an intruder has gained access, all the actions performed on the systems should be reported to keep traces of intruders.

All these previous examples show that the host-based events are not simple to compute. The same events can be acceptable for some company, but not for another. We cannot accept having an alert each time an action is performed on the host, because it will not be useful. The amount of data needs to be reduced in order to be efficient. This can be done by building rules describing the normal behavior of the system: the profile. Of course these rules are not trivial and are company specific. We will not discuss this in more details.

As explained above, the host-based events need a special treatment to be considered as a real alarm. This can be done by what we call "the profile". The profile must include all information that can lead to a better knowledge of the event's context. The network-based IDSes inform about the current outside attacks, will give you information about the attack that is going on. With this additional information you can see if the attacker was successful or not, just by watching if your host-based system reports anything unusual. Normally when someone breaks into a system, the first thing he will do is to clear the log files or manipulate files like the .rhosts. By monitoring these files as "critical files" and correlating the information with the network based system you can tell more precisely what the attacker did and if he was successful. That means you should double check your system for malicious files.

4.2 TEC console management

The Tivoli Enterprise Console is a rule-based event management application that integrates network, systems, databases, and application management. The TEC acts like a central point collecting information from many sources.

Events can be imported to the Tivoli Enterprise Console by using *event adapters*. When the event adapter receives information from its sources, the adapter formats the information and forwards it to the event server to process it. Then the information is presented in some distributed event consoles (GUI). The console lets the administrator view and respond to the events [23].

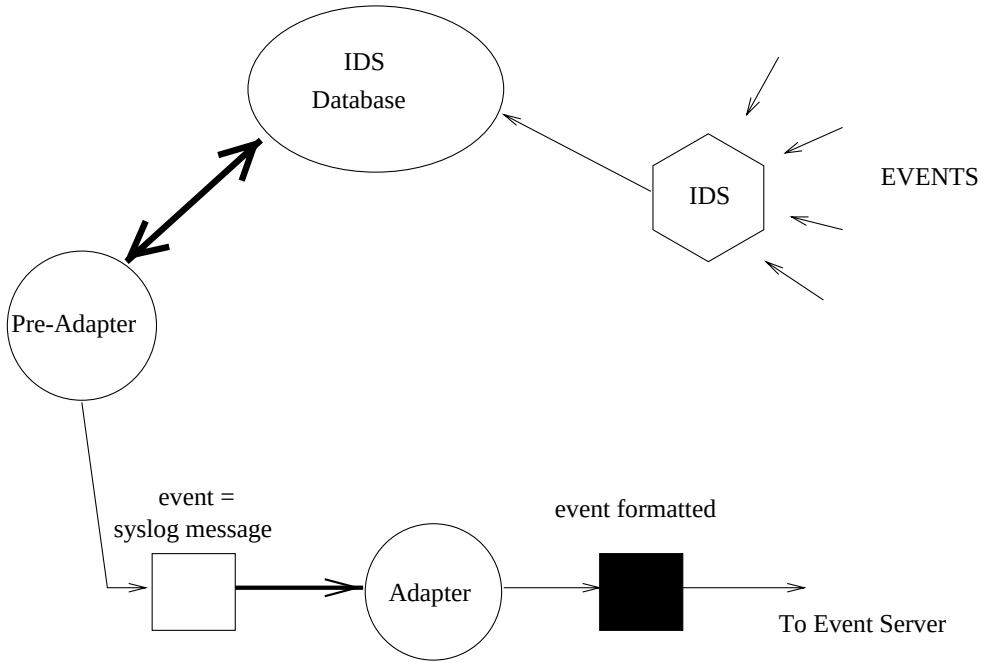

Figure 4.1: Import an IDS event into the TEC

The event adapter transforms source information into event classes before sending information to the TEC server. Many adapters are currently available for TEC 3.6:

- Logfile adapter for the UNIX syslog messages.
- SNMP adapter.
- Windows NT adapter for the NT event log.
- Network management adapters for events coming from OpenView, AIX, SunNet and others.

To import IDS events into the TEC, we need to transform the IDS messages into a readable TEC format which needs to be compatible with the adapters available. We have chosen to use the Logfile adapter. Thus, a module needs to be added to the IDS in order to export its data. It is called the pre-adapter. Its function is to read IDS messages and to generate syslog messages which contain the relevant information. Figure 4.1 shows the whole process.

In the *event server*, the events are represented in a class hierarchy and then they are processed. Basically, they are first logged and checked againse many rules. The rules allow to correlate events, to generate new events, to respond, and so on. The class hierarchy is discussed more precisely in the next chapter.

Chapter 5

Implementation Details

5.1 Pre-Adapters

The pre-adapter aims at exporting information collected by the IDS. It is a module to interface with the IDS and the TEC event adapter. We have chosen to export the event through syslog messages with the TEC Logfile event adapter.

5.1.1 Axent

The manager stores data in a binary file. For the time being Intruder Alert has no option to generate a text file instead of a binary one. The only way to export an Axent event is to use the *execute* command action. Through this command we can execute an SNMP command or a script file. There is the possibility to use two internal variables: {user} and {file event}. {file event} is a temporary file which contains the information related to an alert. There are attributes present for all events and attributes specific to each rule. {user} is the name of the user. This variable is not always defined it sometimes crashes.

When an alert is generated, a perl script *parse.pl* is executed with {file event} as argument. The command executed is the following: *parse.pl* {*file event*}. The script reads the file and, depending on the available attributes, it generates a syslog message which contains information to export.

5.1.2 RealSecure

RealSecure can export event data through SNMP traps, external programs or an ODBC database. A pre-adapter for the RealSecure Network Engine has already been developed in the Global Security Analysis Group (GSAL) at IBM Research, Zurich [1]. This pre-adapter is coded in Java using the JDBC classes to interact with the ODBC database. As the System Agent and the Network Engine share the same database, we have reused the existing pre-adapter code with some adaptations, in order to comply with the needs of the System Agent. The System Agent events are of two different types. The local events and the suspicious connections. The suspicious connection events fall in the network part of the class hierarchy, because a real source IP address is available.

In table 5.1 and 5.2 you can see the two database tables where information is stored [24]. RSLogEvent table contains information common to all events; there are the date when the event was recorded, the signature which should uniquely define an event, the IP source address, the events priority, and so on. Local event fields related to the source are set to a default value and should not be computed. The second table RSLogEventInfo gives some additional information about the events. For the Network Engine, the events' information is almost entirely contained

in the first table; RSLogEventInfo sometimes gives an url or a script name. This is not the case for the local events. In fact, a local event has nothing to do with source information. All the relevant information is stored in the second table; depending on the event, information available is different. Thanks to a timer, the pre-adapter checks regularly if there are new entries in the database to export them.

Name	Description
ID	The unique identifier for the database record
EventDate	The date and the time when the event was recorded
EventName	The name of the recorded event as it appears in the Attack Signatures
ProtocolID	The protocol associated with the event (0-tcp, 1-udp, 2-icmp, 3-unknown)
SourcePort	The port number of the source
DestinationPort	The port number of the destination
SourcePortName	The name of the source port
DestinationPortName	The name of the destination port
SourceAddress	The IP address of the source
DestinationAddress	The IP address of the destination
SourceAddressName	The source machine name
DestinationAddressName	The destination machine name
TCPFlags	This column is currently not in use
ICMPType	The type of ICMP packet
ICMPCode	The code field from the ICMP packet
EventPriority	The priority given to the event (1-high, 2-medium, 3-low)
KillActionSpecified	Whether or not the detector is configured to kill a connection for an event of this type
SourceEthernetAddr	This column is currently not in use
SourceEthernetVendor	This column is currently not in use
DestinationEthernetAddr	This column is currently not in use
DestinationEthernetVendor	This column is currently not in use
RawDataLen	The length of the rawData Field value
RawData	Raw data saved for later viewing through Session Playback
DecodePairCount	The number of decode pairs written as log info records
EngineIP	The IP of the engine from which the event came
EngineType	0 is the Network Engine and 1 is the System Agent

Table 5.1: ISS RSLogEvent table

5.2 Event Class Hierarchy

A Class diagram has already been proposed in an Internet draft model [2] to define data formats to represent the information exported by an intrusion detection system. Our contribution has been to redefine this diagram by adding the eventual host-based part related with the results we

Name	Description
ID	The unique identifier of the entry in the RSLogEventInfo table
LogID	The ID of the entry in the RSLogEvent table
TagName	The name of the decode value
TagValue	The actual value of the decode

Table 5.2: ISS RSLogEventInfo table

got by analyzing the host-based solutions of RealSecure and Axent.

This hierarchy is used to store data received by the TEC of the intrusion detection system. The goal was to group events with similar data. The logical or semantic relations between the attacks have not been taken into account.

5.2.1 Class Definitions

We are not going to explain deeply the existing class diagram, however the global structure is presented. For more information about the classes which do not belong to the host-based part, please refer to the draft itself [2]. While you read, you can refer to figure 5.1.

The top level of the hierarchy is the class event. It does not have a superclass, but might have an OIDs in the SNMP world. The event class is the minimum amount of information that every intrusion detection system must provide for its alerts. There are the signature, the date and other basic information. The second level of the class tree is some kind of a target level. The alert can have multiple targets (class e_multiplehosts) or a single host (e_singlehost). If no information about the source and the destination is reported, the event belongs to a third class (class e_weird). Typically for a host-based IDS, almost all the alerts should belong the the e_singlehost class and its subclasses. We are not going into more details with the others. The e_singlehost class has three subclasses depending on the source of the attacks. The source can be known (es_realorigin), spoofed (es_spoofedorigin), and unknown (es_application). The majority of the events reported by the host-based IDS belongs to the es_application class which contains basically all the events that are happening locally on a machine. In addition to the host, this class contains a description of the used service.

Now we are going to explain more precisely all the subclasses of the es_application class. It contains all the classes which define almost all the host-based events reported by Intruder Alert and RealSecure. The next level contains information about the user which has initiated the event. We have two different classes to take into account that an action can be performed by one or several users. These classes are es_user and es_multipleusers. For the time being only the single user class is used. The next level contains information about the target of the attack; it can be a file, a process, an account, and the audit configuration. Depending on the event, the term account is used either for a user or for a group. If it triggers a group, the event can be related directly to the group or to a group's members. In this case subclasses are derived to define the members. A more formal definition of the classes can be found in the annex.

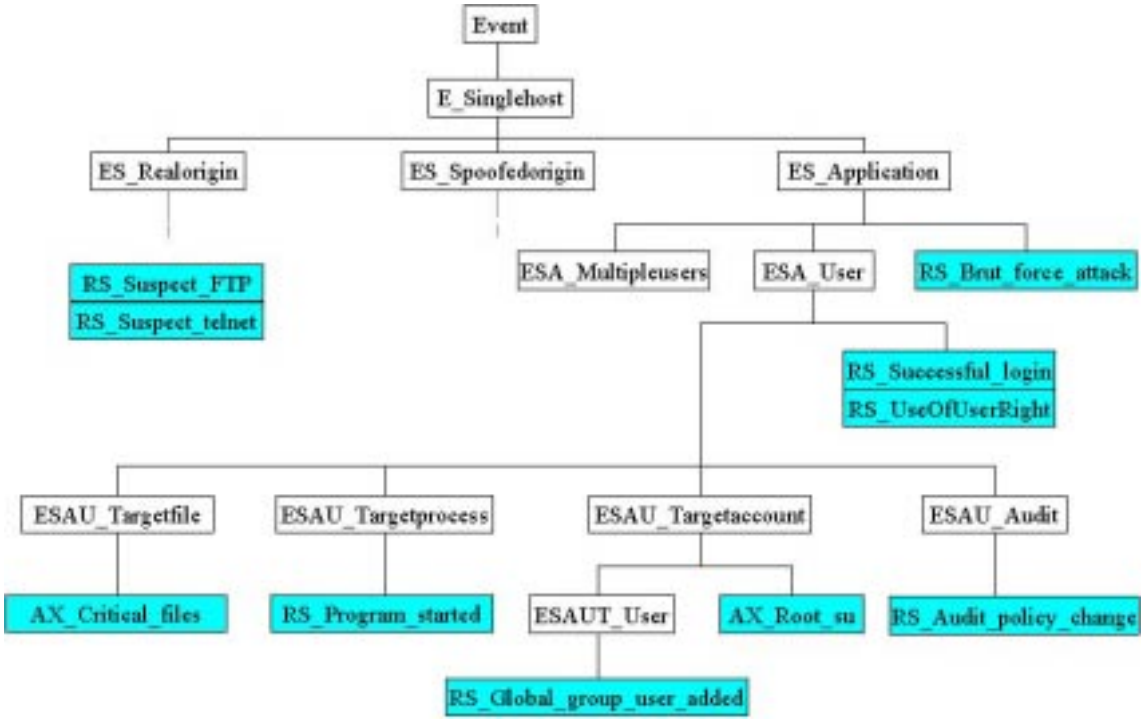

Figure 5.1: Class diagram, host-based part. In the complete tree, the classes ES_realorigin and ES_spooforigin contain not represented sublasses [2]. Some RealSecure events (RS) and Axent events (AX) are represented in the diagram.

Chapter 6

Conclusion

The goals of this thesis were to get familiar with two market leader host-based IDSes, to test them with some attacks in order to generate all the possible host-based events. This task has enabled us to enrich the current IETF proposition for a common format to represent the alerts generated by the IDSes. Finally the two products have been successfully integrated into the Tivoli Enterprise framework thanks to the implementation achieved in the course of the project.

The first part has taken a lot of time to understand all the possible functionalities of these products. We have seen that depending on the product we have more or less capabilities to express rules to deal with the alerts. Intruder Alert offers us the possibility to use tools like flags and timer, whereas Realsecure does some simple pattern matching against the events it takes directly from the NT security event log.

The IDS events have been generated in most case by doing simple host tasks like *failed su*. We have seen that the majority of the host events do not help us to detect intrusions; they are normally just information messages about the host activities. Another aspect is that local exploits are rarely logged, because the only chance to catch them is to set up monitoring of some critical files. As already mentioned, depending on the environment, the same event can be considered as an intrusion or not. This is why we need some more processing on the IDSes event. It is possible to do this in exporting them. IETF works in defining a common format to represent the IDSes alert.

The products tested have generated UNIX and NT events. We have defined new classes to the class hierarchy proposed the IETF work [2] in order to represent them. That allows other product to get the IDS alerts and improve the results by doing more processing. In fact, IDSes generate a huge amount of alarms, so that it is quite impossible for an administrator to keep up with all of them.

To demonstrate the validity of the format defined, we have implemented the integration in the Tivoli Enterprise Console. That has been done by exporting the IDS events in a readable TEC format. This could be achieved by adding a *module* to the IDS, called the preadapter. Then the events can be imported in the TEC and are represented in the defined format. This work might end up being part in the new Tivoli product called *Tivoli Secureway Risk Manager*.

We can conclude in saying that, as this thesis shows, it is not because you have set up an IDS that you network is secure. IDSes are not magical tools. They are an added layer to your security model to improve it. However there is still a lot of things to do in this field.

Chapter 7

Acknowledgment

First I would like to specially thank Candid Wüest for his daily collaboration. He deeply helped me during the preparation of my thesis with his comments, suggestions and answers to all of my questions. Of course, we also had some great free time, between the intensive hard work time.

Many thanks to all the members of IBM Zurich Laboratory for always being friendly and helpful when needed. I will not forget Guy Wald and Céline Steenkeste; we shared the same office and had enjoyed swapping jokes. Guy was also a precious LaTeX's reference.

My gratitude goes especially to each person who helped me draw up this report. Many thanks.

Finally, I am grateful to Hervé Debar and Marc Dacier, my company supervisors, and Refik Molva from Eurecom Institute, for giving me the chance to develop this thesis. It would not have been possible without them.

Appendix A

Event Class Hierarchy, Host-Based part

Baroc file describing the class hierarchy.

```
#
# Classes related  to host-based IDSes,  application information, user
# information.
#

TEC_CLASS :
    ES_Application ISA E_SingleHost
    DEFINES {
      esa_ptyinfo: STRING, default = "none";
      esa_srcport: STRING, default = "none";    # Service name or pointer
      esa_dstport: STRING, default = "none";
      esa_svcname: STRING, default = "none";
      esa_pid    : INTEGER;                     # New attribute of ServiceID
    };
END


TEC_CLASS :
    ESA_User ISA ES_Application
    DEFINES {
      esau_username  :  STRING, default = "none";
      esau_userid    :  STRING;
      esau_userdomain:  STRING;
      esau_purpose   :  STRING;
      esau_additional:  STRING;    # actually used only as a
                  # privilege field
    };
END


TEC_CLASS :
    ESAU_Targetaccount ISA ESA_User   # account can be a user or a group
    DEFINES {
      esaut_accountname  :  STRING, default = "none";
      esaut_accountid    :  STRING;
      esaut_accountdomain:  STRING;
      esaut_purpose      :  STRING;
      esaut_additional   :  STRING;   # actually used only as a
                          # privilege field
    };
END

TEC_CLASS :
    ESAU_Targetfile ISA ESA_User
    DEFINES {
      esaut_filename   :  STRING, default = "none"; # name of the file
                                  # (with the path)
      esaut_accessflags:  STRING;
    };
END

TEC_CLASS :
    ESAU_Targetprocess ISA ESA_User
    DEFINES {
      esaut_processid   :  INTEGER;
      esaut_processname :  STRING;
    };
END

TEC_CLASS :
    ESAU_Auditpolicy ISA ESA_User
```

```
    DEFINES {               # Actually only for NT audit
      esaua_SystemSuccess     : STRING;
      esaua_SystemFailure     : STRING;
      esaua_LogonSuccess      : STRING;
      esaua_LogonFailure      : STRING;
      esaua_ObjectAccessS     : STRING;
      esaua_ObjectAccessF     : STRING;
      esaua_PrivilegeUseS     : STRING;
      esaua_PrivilegeUseF     : STRING;
      esaua_DetailedTrackingS : STRING;
      esaua_DetailedTrackingF : STRING;
      esaua_PolicyChangeS     : STRING;
      esaua_PolicyChangeF     : STRING;
      esaua_AccountMgmtS      : STRING;
      esaua_AccountMgmtF      : STRING;
    };
END


TEC_CLASS :
    ESAUT_User ISA ESAU_Targetaccount
    DEFINES {
      esautu_accountname  : STRING, default = "none";
      esautu_accountid    : STRING;
      esautu_accountdomain: STRING;
      esautu_purpose      : STRING;
      esautu_additional   : STRING;   # actually used only as a
                      # privilege field or domain field for
                      # account policy changed
    };
END
#-------------------------------------------------------------------------
```

Bibliography

[1] Stéphane Schitter. Integration of Intrusion Detection Products in the Tivoli Enterprise Console. Master's thesis, Eurecom Institute, June 1999.

[2] Hervé Debar, Ming-Yuh Huang, and David J. Donahoo. Intrusion Detection Exchange Format Data Model. Internet Engineering Task Force - INTERNET-DRAFT, March 2000.

[3] Hervé Debar, Marc Dacier, and Andreas Wespi. A Revised Taxonomy for Intrusion-Detection Systems. IBM Research Division, October 1999.

[4] Kathleen A. Jackson. Intrusion detection system product survey. Research report LA-UR-99-3883, Los Alamos National Laboratory, June 1999.

[5] Hervé Debar, Marc Dacier, and Andreas Wespi. Towards a taxonomy of intrusion-detection systems. *COMPUTER NETWORKS, The International Journal of Computer and Telecommunications Networking*, 1999.

[6] David L. Carter and Andrea J.Katz. Computer Crime: An Emerging Challenge for Law Enforcement, December 1996.

[7] Satan's home page. http://www.fish.com/satan.

[8] Renaud Deraison. The nessus project. http://www.nessus.org.

[9] Axent web page. http://www.axent.com.

[10] Thomas H. Ptacek. Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection. Secure Networks, Inc, Janary 1998.

[11] Sans Institute resources. http://www.sans.org/newlook/resources/IDFAQ/ID_FAQ.htm.

[12] Aix man pages.

[13] Solaris documentation. http://sunline.epfl.ch:8888/ab2/coll.47.5/SHIELD/@Ab2PageView/564.

[14] AXENT. *Installation and User manual for Intrudrer Alert.*

[15] Common Vulnerabilities and Exposures. http://www.cve.mitre.org.

[16] 7th sphere portscan. http://www.7thsphere.com.

[17] Packet Storm. http://packetstorm.securify.com.

[18] Fictional deamon v3.3. http://http://www.fictional.net/index.html.

[19] Nukenabber port listener. http://www.nukenabber.com.

[20] Edward Amoroso. *Intrusion Detection.* IntrusionṄet Books, 1999.

[21] Stefan Axelsson. The Base-Rate Fallacy and its Implications for the Difficulty of Intrusion Detection. Chalmers University of Technology, Sweden, May 1999.

[22] Stefanos Mananaris, Marvin Christensen, Dan Zerkle, and Keith Hermiz. A Data Minig Analysis of RTID Alarms. IBM report.

[23] Tivoli. *TME 10 Enterprise Console version 3.6 documentation.*

[24] ISS. *RealSecure documentation.*