

Building an IDS using OPNET

Guiomar Corral, Agustin Zaballos, Jaume Abella, Carlos Morales

Enginyeria i Arquitectura La Salle, Universitat Ramon Llull, Spain, EUROPE

Passeig Bonanova, 8, 08022 Barcelona Tlf: +34 93 2902400, Fax: +34 93 2902416

E-mail: {guiomar, zaballos, jaumea, is06200 }@salleURL.edu

Abstract

Influenced by recent international terrorist attacks, security has become a very important matter. Government is investing a great amount of money to improve security, as big companies do. One of the most important security fields, where part of the security investment is focused on, is information security. All networks and computer systems where information is stored are becoming an important research field. This field is where this project takes place. The most common network security automated components are Firewalls, Intrusion Detection Systems (IDS) and Honeypot Systems among others. This project's objective is to implement an IDS for OPNET, which can be used to simulate how it works in a corporate network.

Introduction

When a user of an information system takes an action that was not legally allowed to take, it is called an intrusion. The more network access an organization provides, the more computer intrusions occur and it is becoming a major issue. Legions of hackers and cybernetic terrorists represent serious vulnerabilities to our networked society. How to detect and stop them is a priority, but it represents a big challenge to the computer science. Intrusion Detection Systems tools are what we have to fight those intrusions.

This paper is focused on the studies of network intrusions in a simulated environment. In this paper we present a tool which objective is to simulate the way a normal IDS works.

Intrusion Detection Systems

Intrusion detection involves determining that some entity, an intruder, has attempted to gain, or worse, has gained unauthorized access to the system [SIE-99]. Intrusion Detection Systems are programs used to manage a big amount of information, analyze it and deal with attacks.

Most IDSs attempt to detect a suspected intrusion, and then they alert a system administrator. Technology for automated reaction to intrusion is just beginning to be consolidated. The original idea behind automated ID is often credited to Anderson's paper on "How to use accounting audit files to detect unauthorized access" [AND-80] when the emphasis was on single computer systems.

IDS Classification

Based on how we decide to classify an IDS we have two different classifications [BRU-01]. If our classification is focused on how data is collected, we have Network based

or Host based systems. But if it is focused on how data is analyzed there are two groups, the detection of anomalies and the detection of misuses. The first approach, called anomaly detection, is to define and characterize correct static form and/or acceptable dynamic behavior of the system, and then to detect wrongful changes or wrongful behavior [SIE-00]. The second approach, called misuse detection, involves characterizing known ways to penetrate a system.

Our approach

OPNET is a tool used to simulate the way networks run. This project concentrates on a Network Intrusion Detection System, commonly known as NIDS. A NIDS keeps track of all the data that goes through it, trying to detect known attacks based on attack signatures described with rules.

The IDS that we have implemented here, is based on misuse detection. Misuse detection is concerned with catching intruders who are attempting to break into a system using some known technique. Ideally, a system security administrator would be aware of all known vulnerabilities and would eliminate them. By this way our NIDS can be placed in any TCP/IP network just knowing a good amount of TCP/IP vulnerabilities, and without having to learn the acceptable network behavior every time, as should be done with any anomaly detection system.

Rule Attack Database

The NIDS implemented in this project will be able to parse the SNORT rule attack database. A good NIDS needs a good and extended attack signature database. But creating and maintaining an up-to-date database which is good enough is out of this project, because it implies a big effort. This is the reason why this NIDS must be able to process SNORT based rules. This is an independent database, an open source and general public license rule database. Just as SNORT is an open source NIDS, it has a great amount of rule developers who maintain the up-to-date attack signature database, it is well documented and it is easy to create new rules.

Simulation of Network Intrusion

The actual Intrusion detection Systems are more effective in detecting misuse intrusions, because signatures are known, but are poor in detecting new attacks. Such task is an interesting and difficult problem. To find out new intrusion detection algorithms against new intrusive activities in a real environment is security's future. But these studies can be performed in a real environment or in

a simulated environment. Let's summarize these two approaches, followed by a description of our approach.

Simulation in Real Environments

In real environments, real traffic and real systems are being used, and the intrusive activities can be generated by running known exploit scripts. This approach has the advantage that background traffic is sufficiently realistic, which eliminates the need of simulating normal user activities, because they are implicitly placed in the traffic, and therefore the workload of tests can be significantly reduced. But this approach has some drawbacks [WAN-01]:

- The testing environment is exposed to the risk of attacks during simulation.
- Background traffic may contain intrusive data, and therefore increase the inaccuracy of test results.
- Systems could be interrupted by simulated attacks. Using destructive attacks, normal system operation could be compromised.

In this approach, realistic environments are compromised, due to the high risk of the actual performing tests.

Simulation in OPNET

OPNET is the chosen simulated network environment to be used in this project. In this environment, we are going to import network traffic, sniffed previously with TCPDUMP. That traffic resides in binary files, which contain intrusion traffic simulating various network attacks. NMAP [NMAP] and NESSUS could be used to generate such attacks, meanwhile a network sniffer such as TCPDUMP captures network traffic, with the attack and network sniffing activities all occurring in a controlled lab environment, with no risk for real environments. The captured file, which includes the attack data and normal user data if desired, will be used in our intrusion simulation experiment. OPNET software provides a module called ACE (Application Characterization Environment), which can be used to define a profile based on imported packet traces, supporting packet formats from various sources including TCPDUMP binary files. We are using our approach because of the flexibility of selecting parts of the data packets and slicing large data files into more manageable pieces prior to simulation. Although OPNET flexibility permits us to study traffic, we need to compare at the first steps of the development process, results between the built IDS and the real IDS, so we can check the correctness of our solution.

We are using OPNET for our research because of the several benefits it offers. OPNET provides the most complete tools, and a complete user interface for topology design and development. Another advantage of using OPNET is that it has been extensively used and there is wide confidence in the validity of the results it produces. OPNET enables realistic analysis of performance measures and the effectiveness of intrusion detection techniques. [SRA-02]

Simulation Applications

This project can be used for many applications. It is interesting to simulate an attack against a network, and be able to check if security systems will work properly. With this tool we can predict if there should be more NIDS, or if the NIDS being used can control all the data that crosses the network. Another possible scenario for this tool is after an attack has been committed. All network data can be imported with the ACE module into OPNET and the network administrator can check out how the attack was executed, what effects it had and so on. and all this in a safe and simulated environment.

Design and Implementation

All the project work is inside the node domain. Specified among the node model attributes, the user will be able to define the file where all the SNORT based rules reside. Likewise, other configuration variables are going to be among the promoted arguments. Following this design, the NIDS can be used inside the IT Guru as an autonomous system.

Internal Security

IDS systems can be composed into modules of distinct functionality. And such components should be re-usable in a different context than they were originally developed for. The internal structure is divided into two main modules. The main purpose of the first one is recollecting the traffic transferred across the network. The second one verifies if data is intruder suspicious. These two groups of modules work together in the same node, and are connected between them with packet streams.

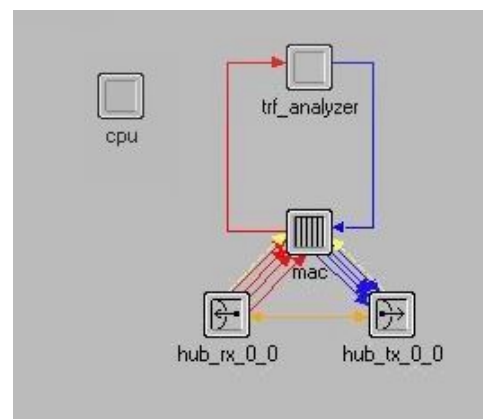


FIGURE 1: General IDS node view

Traffic Collector

The first one of the three modules is in promiscuous mode and recollects all data going across the wire. It also defragments Ethernet frames as they come, if they were fragmented. The Traffic Collector has been designed to work inside an Ethernet environment, which normally is connected to a hub with an external 10BaseT connection

wire. The functionality of the traffic collector is accomplished with three modules: a queue module, a transmitter module and a receiver module. The transmitter and receiver modules are point-to-point type modules. Both are connected with a logical association to indicate that a relationship exists between them. These two modules communicate with external links outside the node and with a standard queue module. The queue module contains additional internal resources, which facilitate buffering and managing all the collected data. This queue module executes the Ethernet_mac_v2 process, and therefore, it accomplishes the needed specifications and therefore this node can be placed in any TCP/IP network.

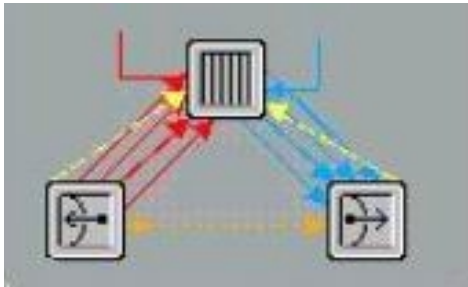


FIGURE 2: Traffic collector

Traffic Analyzer

This component is the most important component inside the implemented IDS. This component analyzes traffic and decides if it corresponds to an attack. This second module also manages all the rules, which means it must parse the file with the SNORT based rules into the simulation program and load them into the process memory. Every time a packet is collected by Traffic Collector, it is passed to this upper module and checked to see if the actual packet activates any rule stored in the attack signature database. These actions are performed by only one processor module, but process definitions in OPNET are enough complex and versatile to do all these operations in only one processor module.

The process that defines the processor module is composed by three forced states and four unforced states. The "init_state" is a forced-state which initializes some global variables. The second forced state purpose is to parse the rules into memory; this is why it is called parseRules. After finishing it, all the variables are initialized and therefore this module can start analyzing suspicious packets, so it goes to the last forced state to wait them, and because of this it, is called WAIT. Every time an incoming packet interrupt is activated, the process goes from WAIT state to ip_arrival state. This unforced state checks if the received IP packet matches any IP rule. If the packet doesn't match any, then the process checks if the IP packet transports any other protocol. If it does, the state diagram jumps into one of the three available protocols states: TCP, UDP or UDP. Each one of them is treated into one unforced state.

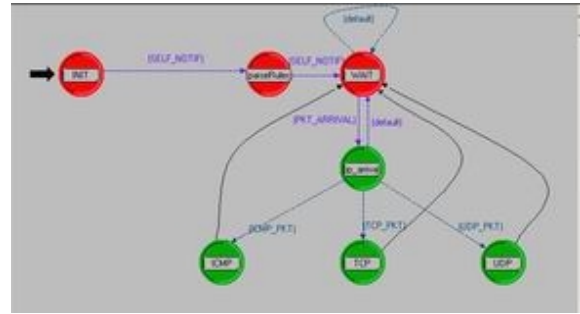


FIGURE 3: Analyzer State Transition Diagram

Generating an Intrusion Data

We use the NMAP scan attack as an example to illustrate the process of our intrusion simulation. NMAP is one of the most used TCP/IP scanners. This causes a flood of packets whose IP header flags are normally illegal. Such behavior is used to guess the attacked operating system by analyzing its response to that traffic. This traffic is captured with TCPDUMP and then imported with the ACE module as explained before. This imported traffic generates a scenario, and the IDS can be introduced inside this scenario.

Generating the Traffic

The scenario used to get traffic is composed by three different computers: one attacker, one victim and a sniffer. The attacker executes the NMAP program against the victim, which has services running on it. In our example, the NMAP program has activated the TCP SYN stealth port scan flag and at the same time it uses TCP/IP fingerprinting to guess the remote operating system. The victim runs three daemons, which are FTP, Web and SSH services. Before port scan is started, the sniffer host starts sniffing all data that goes across the network. For this kind of scan the NMAP program generates approximately 3300 packets, and the TCPDUMP stores them in a binary data file. This file is imported into the ACE module, and so this traffic generates a well defined scan pattern used by the NMAP program. Inside the ACE module we can delete undesired traffic.

Testing the scenario

When previous steps are done, we can create a new project importing the ACE generated file with the Modeler Startup Wizard help, whose steps are easy to follow. Creating the Application, we choose to use the NMAP based on ACE trace File and then, repeat it only once. This will create a simple LAN where we can test the correctness of our IDS. But before we start executing the simulation we need to change the default switch with a hub, so the IDS can read all the traffic generated by the network. The results of these steps are shown in the follow figure.

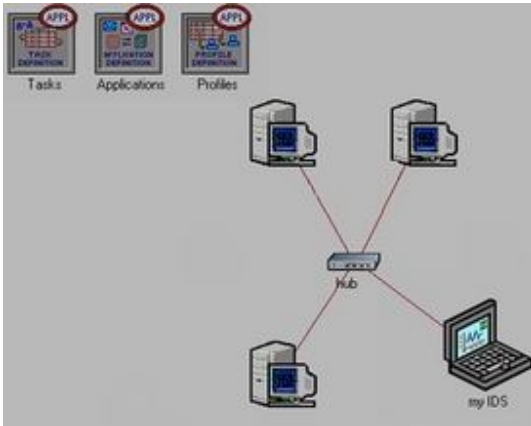


FIGURE 3: Scenario where the IDS is tested

The last work to be done is to execute the simulation and compare the results with those we expected. Before starting the discrete event simulation we should check that all variables are correctly inserted; those variables are related with the task configurations. There should be one row inside the profiles configuration specifying the imported traffic. This traffic can not be repeated more than once and, therefore, the attack traffic is present only once. The used configuration is shown in the next figure.

Applications	[...]
rows	1
row 0	
Name	traficoimportado
Start Time Offset (seconds)	exponential (3600,000000)
Duration (seconds)	End of Last Task
Repeatability	[...]
Operation Mode	Serial (Ordered)
Start Time (seconds)	uniform (100,110)
Duration (seconds)	End of Simulation
Repeatability	[...]
Inter-repetition Time (seconds)	constant (300)
Number of Repetitions	constant (0)
Repetition Pattern	Serial

FIGURE 4: Profile configuration

The IDS node only checks packets received in its interfaces. Therefore this node is connected to a hub, but the configuration must expose the promiscuous mode as enabled. Although the parameters should be configured too, they are correctly inserted by default as shown in the figure.

Ethernet	
Ethernet Parameters	[...]
Address	Auto Assigned
Frame Bursting	Enabled
Operational Mode	Full Duplex
Promiscuous Mode	Enabled
Segment Processing Delay	promoted
ARP	
IDS	
if_analyzer.output.rules	alerts.bd
if_analyzer.snort.rules	all.rules

FIGURE 5: IDS configuration

Conclusions and future work

In this paper we reported how to create a Network based Intrusion Detection System, with the imported SNORT IDS Rules database as misuse knowledge. The results reported by the IDS were satisfactory. All the attacks alerted by the IDS were previously detected by the real SNORT with the same traffic. Therefore we can extrapolate the problem using more than just one IDS system and only three hosts as shown here, to bigger network systems.

As future work, we plan to enhance the traffic analyzer kernel and be able to connect any kind of external program and develop new algorithms, whose developers could work with the same imported traffic as we did.

Bibliography

- [AND-80]
James P Anderson
Computer Security Threat Monitoring and Surveillance
<http://csrc.nist.gov/publications/history/ande80.pdf>
- [BRU-01]
Guy Bruneau
The History and Evolution of Intrusion Detection
<http://www.sans.org/rr/whitepapers/detection/344.php>
- [NMAP]
<http://www.insecure.org/nmap/>
- [SIE-99]
Robert Sielken
Application Intrusion Detection Systems: The Next Step
http://www.cs.virginia.edu/~jones/IDS-research/Documents/Application-IDS_Jones-Sielken.doc
- [SIE-00]
Robert Sielken
Computer System Intrusion Detection: A Survey
<http://www.cs.virginia.edu/~jones/IDS-research/Documents/jones-sielken-survey-v11.pdf>
- [SRA-02]
Shabana Razak
Network Intrusion Simulation Using OPNET
<http://www.cs.ucf.edu/csdept/faculty/lang/pubs/opnet2002.pdf>
- [WAN-01]
Tao Wan
ItruDetector: A Software Platform for Testing Network Intrusion Detection Algorithms
<http://www.acsac.org/2001/papers/54.pdf>