Intrusion Detection System

Objectives

- To know what is Intrusion Detection system and why it is needed.
- To be familiar with Snort IDS.

Overview

Intrusion Detection Systems (IDS), firewalls, and honeypots are all security measures used to ensure a hacker is not able to gain access to a network or target system. An IDS and a firewall are both essentially packet filtering devices and are used to monitor traffic based upon a predefined set of rules. A honeypot is a fake target system used to lure hackers away from the more valuable targets. As with other security mechanisms, IDSs, firewalls and honeypots are only as good as their design and implementation. It is important to be familiar with how these devices operate and provide security as they are commonly subjects of attack. So, in this lab we will talk about IDSs.

Definition of IDS

- An intrusion detection system (IDS) gathers and analyzes information from within a computer or a network, to identify possible violations of security policy, including unauthorized access, as well as misuse.
- An IDS is also referred to as a “packet-sniffer,” which intercepts packets that are traveling along various communication mediums and protocols, usually TCP/IP.
- The packets are then analyzed after they are captured.
- An IDS evaluates a suspected intrusion once it has taken place, and signals an alarm.
IDS Types and Components

IDS can be divided into two broad categories: network-based intrusion detection systems (NIDSs) and host-based intrusion detection systems (HIDSs).

Network-based intrusion detection systems (NIDSs)

NIDSs examine packets on the network and look at the data in an attempt to recognize an attack. A NIDS makes use of a computer that has its NIC placed in promiscuous mode. This basically means that the NIC accepts all data packets it sees, not just the ones specifically addressed to it. If the system is operating on a hub, this requires nothing more than plugging the NIDS into the hub. If a switch is being used, a port must be mirrored or spanned. This action configures the switch to direct traffic from either specific ports or a specific virtual LAN (VLAN) to the port you have specified to be used by the IDS. One advantage of a NIDS is that it can support many sensors so that the system can monitor the demilitarized zone (DMZ), the internal network, or specific nodes of the network. The disadvantage of a NIDS is that even if it can see certain types of traffic (e.g., encrypted), it doesn’t mean that it knows what the traffic is actually doing. Another disadvantage of a NIDS is that it will not detect attacks against a host made by an intruder who is logged in at the host’s terminal. If a network IDS along with some additional support mechanism determines that an attack is being mounted against a host, it is usually not capable of determining the type or effectiveness of the attack being launched. Some examples of a NIDS include Snort (www.snort.org), and Cisco Intrusion Detection System.

Once an attack is detected, a NIDS can perform one or more of the following functions:

- Configure the firewall to filter out the IP address of the intruder.
- Launch a separate program to handle the event.
- Play an audio file that says “Attack is taking place.”
- Save the packets in an evidence file for further analysis.
- Send an entry to a system log file.
- Send an e-mail, a page, or a cell phone message to the network administrator.
- Terminate the TCP.

**Host-based intrusion detection systems (HIDSs)**

HIDSs only monitor traffic on one specific system. HIDSs typically do not place the NIC in promiscuous mode, and therefore do not have to deal with the level of traffic that a NIDS would. Promiscuous mode can be CPU-intensive for an older and slower computer. HIDSs looks for unusual events or patterns that may indicate problems. HIDSs excel at detecting unauthorized accesses and activity. As an example, if a word processor starts accessing an e-mail program and is sending hundreds of emails, the HIDSs would be alerted. HIDSs can also look at the state of a system and verify that all contents appear as expected. Both NIDSs and HIDSs can be configured to scan for attacks, track a hacker’s movements, and alert an administrator to ongoing attacks. Some examples of HIDSs are:

- **Tripwire** ([http://sourceforge.net/projects/tripwire](http://sourceforge.net/projects/tripwire)).
- **Samhain** ([http://la-samhna.de/samhain](http://la-samhna.de/samhain)).
- **Swatch** ([http://swatch.sourceforge.net](http://swatch.sourceforge.net)).
- **RealSecure** ([http://www.iss.net](http://www.iss.net)).

Most IDSs consist of more than one application or hardware device. *IDSs are composed of the following parts:*

- **Network sensors**: Detect and send data to the system.
- **Central monitoring system**: Processes and analyzes data sent from sensors.
- **Report analysis**: Offers information about how to counteract a specific event.
**Database and storage components**: Perform trend analysis and store the IP address and information about the attacker.

**Response box**: Inputs information from the previously listed components and forms an appropriate response.

**Ways to Detect an Intrusion**

Intrusion detection engines or techniques can be divided into two distinct types or methods: **signature and anomaly**.

A **signature-based or pattern-matching** IDS relies on a database of known attacks. These known attacks are loaded into the system as signatures. As soon as the signatures are loaded into the IDS, it can begin to guard the network. The signatures are usually given a number or name so that the administrator can easily identify an attack when it sets off an alert. Alerts can be triggered for fragmented IP packets, streams of SYN packets (DoS), or malformed ICMP packets. The alert might be configured to change to the firewall configuration, set off an alarm, or even page the administrator. Figure 1 shows an example of how a signature-based IDS works.

![Figure 1: Signature-based IDS](image)

The biggest disadvantage of signature-based systems is that they can trigger only on signatures that have been loaded. A new or obfuscated attack may go undetected. Snort is a good example of a signature-based IDS.
Anomaly-detection systems require the administrator to make use of profiles of authorized activities or place the IDS into a learning mode so that it can learn what constitutes normal activity. Figure 2 shows this overall process. A considerable amount of time needs to be dedicated to make sure that the IDS produces few false negatives. If an attacker can slowly change his activity, over time the IDS may be fooled into thinking that the new behavior is actually acceptable. Anomaly detection is good at spotting behavior that is greatly different from normal activity. As an example, if a group of users who log in only during the day suddenly start trying to log in at 3 a.m., the IDS can trigger an alert that something is wrong.

![Figure 2: Anomaly – based IDS](image-url)
Snort

Overview
Snort is a freeware IDS developed by Martin Roesch and Brian Caswell. It’s considered a NIDS that can be set up on a Linux or Windows host. Although the core program has a command-line interface, two popular GUIs can be used: SnortSnarf and IDScenter. Snort operates as a network sniffer and logs activity that matches predefined signatures. Signatures can be designed for a wide range of traffic, including IP, TCP, UDP, and ICMP. If you have never used an IDS, you might be surprised at the number of alerts it will produce in just a few hours upon being connected to the Internet.

Platform Compatibility
Snort can be run on both Linux and Windows. It can also be run on other platforms, such as FreeBSD, Solaris, and Mac OS X. If you are going to run Snort on a Linux system, you can take advantage of some precompiled binaries that are already available for use. You also have the option of running it from a CD-based Linux OS, such as BackTrack. While the choice of Linux or Windows may be a no brainer for some purists, there are advantages and disadvantages for each platform.
Lab Experiment

1. Snort over windows System

Requirements:

In this experiment we need at least two machines. We will setup snort on one machine and use other machines as attackers.

Procedures :

- **Sniffer Mode**

Sniffer mode works just as the name implies. It configures Snort to sniff traffic.

Let’s take a moment as this point to verify Sniffer mode:

1. To check whether Snort was properly configured, open two command prompts.
2. At one of the command prompts, navigate to the C:\snort\bin folder, and enter *snort –W*. You should see a list of possible adapters on which you can install the sensor. The adapters are numbered 1, 2, 3, and so on.
3. At the c:\snort\bin prompt, enter *snort –v –ix* or *snort –dev –ix* or *snort -d*, where *x* is the number of the NIC to place your Snort sensor on.
4. Switch to the second command prompt and ping another computer. When ping is complete, switch back to the command prompt window running Snort, and press Ctrl+C to stop Snort.
**IDS Mode**

**Building Snort Rules**

1. The first thing to do after installation is to configure the local network.

2. Distinguish the internal from external traffic.

3. Open up `C:\Snort\etc\snort.conf` with Notepad and find the line
   
   ```
   var HOME_NET any
   ```
   
   and replace "any" with the IP range and subnet mask.
   
   i.e. `var HOME_NET 192.168.1.5/24`

4. If you have more than one internal subnet you can specify them all by
   putting them in brackets and separating them with a comma. i.e.
   
   ```
   var HOME_NET [10.1.1.0/24,192.168.1.0/24]
   ```

5. Next, define the external network, by finding the line `var EXTERNAL_NET any`.

6. Replace “any” with the IP address(es) of the external networks, or you can
   leave "any" to set all the networks not defined as HOME_NET as external.

Next, define the services on our network.

7. Find the following lines and replace `$HOME_NET` with the IP address(es) of
   the server(s) running the service
   
   - `var DNS_SERVERS $HOME_NET`
   - `var SMTP_SERVERS $HOME_NET`
   - `var HTTP_SERVERS $HOME_NET`
   - `var SQL_SERVERS $HOME_NET`

8. Configure dynamic loaded libraries, Change the following line
   `dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/`
   with
   
   `dynamicpreprocessor directory C:\Snort\lib\snort_dynamicpreprocessor`
   
   and this line
   
   `dynamicengine /usr/local/lib/snort_dynamicengine/lsf_engine.so`
   with
   
   `dynamicengine C:\Snort\lib\snort_dynamicengine/sf_engine.dll`
9. Configure output plugins, Change the following two lines

```
include classification.config
include reference.config
```

with

```
include c:\snort\etc\classification.config
include c:\snort\etc\reference.config
```

- add this line to write the log details

```
outputalert_fast : alerts.ids
```

after the following line

```
outputlog_tcpdump: tcpdump.log
```

- **Note**: you must create file alerts.ids in log folder in `c:\snort\log`

10. Add any runtime config directives

11. Customize your rule set

- Modify the path of rules folder as follows

```
var RULE_PATH ..\rules

var RULE_PATH c:\snort\rules
```

**NOTE**: don’t forget to download rules from snort.org and add it in rules folder

- each rule you want to activate you must add this line for it; as an example if we want to activate icmp-info rule we add:

```
include $RULE_PATH/icmp-info.rules
```

**Note**: snort.conf file contains a sample snort configuration, you can built your own in the same structure as your network required as mentioned in the above steps with different name.
Testing Snort

Example One:
1- Do the previous steps from 1 to 11.
2- Add the following lines in snort.conf file (remove comment if this line found and commented)
   include $RULE_PATH/icmp.rules
   include $RULE_PATH/icmp-info.rules
3- form console ; run the following command
   snort –c c:\snort\etc\snort.conf –l c:\snort\log –i X "where x is # of interface”
4- try to ping your machine where snort installed from any other machine.
5- [Get result] go to log folder and open alerts.ids file ; you will find who are pinging your machine and how many times he do this.

Example Two: Built your custom rules
1- Create new file with iugaza.rules name in rules folder where snort installed.
2- Open the file and add the following line:
   alert tcp any any -> any any(content:"www.iugaza.edu.ps" ;
   msg:"someone browsing IUG site now ";id:10000020;rev:1;)
3- add the following line to snort.conf file to activate your rule:
   include $RULE_PATH\iugaza.rules
4- form console ; run the following command
   snort –c c:\snort\etc\snort.conf –l c:\snort\log –i X “where x is # of interface”
5- open www.iugaza.edu.ps
6- [Get result] go to log folder and open alerts.ids file and see the message appeared.
Exercises:

1- Do the three parts of this lab:
   
i. Use snort as sniffer

ii. Customize config file as required for your network and do previous changes, enable any rule and show the results on your report.

iii. Create simple filter and show the result.

BONUS:

I. Repeat part one on any Linux platform.

II. How to configure snort to work as Intrusion Prevention System IPS?