Project 2-3:
Intrusion Detection System

Information Security

1 Introduction

1.1 General Summary

An intrusion detection system looks for known attacks. A simple type of intrusion detection system looks for attack signatures. This assignment will require you to detect some known attacks and log them.

2 Network Integrity (50%)

You will only have a single program for this assignment. It must be named *ids* on Linux/Unix platforms, or *ids.exe* on Windows platforms. You will only need to worry about monitoring TCP traffic. Modern IDS programs check for a variety of things, you will be asked to check for only three significant signatures of attack.

1.) Detect for OS remote detection. You should be able to use your code from a previous lab to accomplish this fairly easily. Consider a remote hosts that sends 5 packets with different Syn, Ack, and Fin combinations within 20 seconds to be malicious.

2.) Check for rudimentary(not stealth) style of port scans. To do this, simply monitor traffic and watch for a single IP trying to connect to sequential ports. For this assignment, consider a host trying connecting to 5 sequential ports to be malicious.

3.) Implement the ability to detect buffer overflow attacks. This can be done by sniffing data from the network traffic and looking for a group of NOP (no operation) commands. Using NOP is a common occurrence when an attacker writes a buffer overflow program. Consider a packet load containing 5 or more sequential NOP commands to be malicious. Also, instead of searching for the exact machine code of the NOP command, consider the ASCII letters ”NOP” to be the NOP command.

In order to read packets, use libpcap if you are programming in a Linux/Unix type of environment, or WinPcap if you are programming in a Microsoft Windows environment. They can natively be used with C/C++ and have bindings to Java as well.

For Linux/Unix users, the libpcap library can be downloaded from http://www.tcpdump.org.

For Windows useres, the winpcap library can be downloaded form http://winpcap.polito.it/

Whether you use libpcap, or winpcap, read the tutorial from each website, which will give you a very good understanding and examples of how pcap works. It is also recommended you read and understand the supplied examples that come with the source code download.

For ease of programming, only worry about IPV4 and disregard IPV6.

3 Intrusion Notification (35%)

When an attack signature is identified, it should log the suspicion to a file named *ids.txt*, including: the type of intrusion that was detected, where it was from, and what time it occurred. It should then completely block the offending IP using your code from your previously created firewall program.

The following is an example of what an *ids.txt* file might look like.

```
110704-11:37.31 192.168.0.102 Port Scan
110804-03:28.22 192.168.0.103 OS Fingerprint
110904-05:27.54 192.168.0.104 Buffer Overflow
```

The file should be in this format:

```
MMDDYY-HH:MM.SS <IP> <Reason>
```

4 User Interface (10%)

Your program will have menu that must look and behave like the following:

1) Start IDS
2) Stop IDS
3) View Current Traffic
4) View Block List
5) View Current Firewall Rules
6) Unblock User
7) Quit

4.1 Start IDS

Start monitoring all traffic, and begin analyzing for criteria specified in below sections. If activation works, respond “Started IDS!”, else respond “Error! Unable to start IDS!”, with appropriate error message.

```
# Start the firewall, should give positive feedback
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
1
Started IDS!

# If there is an error
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
1
Error! Unable to start IDS!
Can’t Open Device
```

4.2 Stop IDS

Stop monitoring traffic, and clear all lists of blocked users and firewall rules. If deactivation works, respond “Stopped IDS!”, else respond “Error! Unable to stop IDS!”, with appropriate error message.

```
# Start the firewall, should give positive feedback
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
2
Stopped IDS!
```

```
# If there is an error
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
1
Error! Unable to stop IDS!
IDS already stopped
```

4.3 View Current Traffic

Simply start dumping all sniffed traffic to the console traffic for 10 seconds. Use the standard pcap way of presenting the data.

4.4 View Block List

List all IP addresses that are currently blocked and the reason for being blocked. Below are the three possible reasons for being blocked.

```
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
4
1-192.168.0.102 - Port Scan
2-192.168.0.103 - OS Fingerprint
3-192.168.0.104 - Buffer Overflow
```

The output should be in this format:

```
<User#>-<SIP> - <Reason>
```

4.5 View Current Firewall Rules

Print rules in order they were entered. Make sure to number them, so the user can tell what rule they want to delete if needed.

```
1. Start IDS
```

```
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
5
1-192.168.0.102/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
2-192.168.0.103/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
3-192.168.0.104/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
```

The output should be in this format:

```
<Rule#>-<SIP>/<SMask>:<SPort> <DIP>/<DMask>:<DPort> <Protocol> <Action>
```

4.6 Unblock User

Have user enter number that corresponds to the IP address from the View Block List command, then unblock this user. This means you must keep track of which firewall rule belongs to which blocked user. If unblocking works, respond “Unblocked User!”, else respond “Error! Unable to unblock user!”, with appropriate error message.

```
# Delete the second user
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
6
2
Unblocked user!

# The user is not in the blocked list anymore
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
4
1-192.168.0.102 - Port Scan
2-192.168.0.104 - Buffer Overflow
```

```
# Now print the firewall rules, and see corresponding rule is gone
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
5
1-192.168.0.102/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
2-192.168.0.104/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1

# If you try to delete the third user, we would now get an error
1. Start IDS
2. Stop IDS
3. View Current Traffic
4. View Block List
5. View Current Firewall Rules
6. Unblock User
7. Quit
6
3
Error! Unable to unblock user!
There is no user 3
```

4.7 Quit

Stop all firewall and packet monitoring processes and exit gracefully.

5 Testing

Along with your assignment, include a document *test.txt* that includes the resulting firewall rules, and output from *ids.txt* from running the following commands on your computer:

```
# This tests for an OS fingerprinting attack
hping2 localhost -SAF
hping2 localhost -SA
hping2 localhost -S
hping2 localhost -SF
hping2 localhost -AF
# Tests for buffer overflow
# nop.txt should be a text file containing "NOPNOPNOPNOPNOP"
hping2 localhost -E nop.txt
# Tests for port scan
hping2 localhost -p +5200
```

If done correctly, your test.txt file would look like the following:

```
1-192.168.0.101/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
2-192.168.0.101/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
3-192.168.0.101/255.255.255.0:0 0.0.0.0/0.0.0.0:0 0 1
110704-11:37.31 192.168.0.101 OS Fingerprint
110804-03:28.22 192.168.0.101 Buffer Overflow
110904-05:27.54 192.168.0.101 Port Scan
```

Please note that this example should work, even though the packets are getting blocked by the firewall, because pcap reads the packages straight from the device before the OS gets them. This isn't the most amazing example, but allows you to test your program without the need of another computer on the network.

6 Required Document (5%)

In addition to your source code and your output you are required to submit a 1-3 page document that is flawlessly written. No spelling errors. No grammatical errors, etc. If the document is deemed unprofessional (e.g.; a significant number of grammatical or spelling errors it will be assigned a grade of zero.)

Your document should state clearly the status of the assignment. Is it done or not? Does it meet all of the requirements. If anything is missing state clearly what is missing. Failure to give the status will result in a grade of zero for the entire assignment. Providing a that is false or significantly misleading will also result in a zero for the entire assignment.

Make sure your document is well-written, succinct, and easy to ready. If you had problems with the assignment, give a detailed discussion of the problems and how you handled the problem.

7 Assignment Submission (5%)

You are required to make both an electronic submission, and a paper submission. You may turn in your paper submission, before class on the due date, or in my mailbox before class time on the due date. For the paper submission, make sure to do the following:

* Make sure your completeted handout is on top, with your name on it.
* Next, attach all source code files printed out.
* Also, submit a printed copy of the required document(see above).
* Make sure they are all stapled together.

Submit your electronic copy using Blackboard (attach the assignment as a compressed archive file(.zip,.tgz,.tbz2,.rar), make sure you do the following:

* The name of the compressed archive should have your name and project number(IE: Jordan Wilberding Project 2-2.zip)
* Include the source code and a README file on how to run each program.
* Include your status should come right at the beginning of the document.
* Include your e-mail address in the Comment field when submitting the assignment through the Digital Drop Box.
* If for any reason you are submitting the assignment more than once, indicate this in the Comment filed by including the word COMPLEMENT.
* Include a shell script or batch file that will compile your programs when executed. Including a Makefile (see GNU Make for details) would be most appropriate but any form of script or building tool will do.
* Include some form of executable that will run each individual program. If you are doing C/C++ for example, include the compiled executable, if you are using Java, or a fully interpreted language such as perl, include a shell script or batch file that will give the proper arguments and perform the execution.