Programming Assignment #7

Intrusion Detection System

---

1. Introduction

An Intrusion Detection System (IDS) looks for known attacks. A simple type of IDS looks for attack signatures. This assignment will require you to detect some known attacks and log them.

2. Details

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 www.tcpdump.org.

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

Whether you use libpcap, or winpcap, read the tutorial from each website. Also, read and understand the examples that come with the source code.

For ease of programming, only worry about IPV4. You get 50% extra credit if you also handle IPV6.

3. Network Integrity (45%)

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 **(15%)**. 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 **(15%)**. 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 **(15%)**. 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.

4. Intrusion Notification (10%)

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 file should be in this format:

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

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

```
20071107-11:37.31 192.168.0.102 Port Scan
20071108-03:28.22 192.168.0.103 OS Fingerprint
20071109-05:27.54 192.168.0.104 Buffer Overflow
```

5 User Interface (20%)

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
```

5.1 Start IDS

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

Example: Start the IDS, should give positive feedback

```
$ ./ids
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
IDS Started!
```

Example: If there is an error

```
$ ./ids
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!
Cannot Open Device
```

5.2 Stop IDS

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

Example: Stop the IDS, 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
IDS Stopped!
```

Example: If there is an error

```
$ ./ids
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
Error! Unable to stop IDS!
IDS already stopped.
```

5.3 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.

5.4 View Block List

List all IP addresses that are currently blocked and the reason for being blocked.

Example: 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>
```

5.5 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.

Example:

```
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 the following format:

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

5.6 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 "User Unblocked!", else respond "Error! Unable to unblock user!", with descriptive error message.

Example: Unblock the 2nd 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
User 2 unblocked!
```

Verify user is no longer 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
2-192.168.0.104 - Buffer Overflow
```

Now print the firewall rules to check the 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 unblock the 3rd user you should 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
```

5.7 Quit

Stop all firewall and packet monitoring processes and exit gracefully.

6. Testing Your Application (15%)

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:

Example: Tests for an OS fingerprinting attack

```
hping2 localhost -SAF
hping2 localhost -SA
hping2 localhost -S
hping2 localhost -SF
hping2 localhost -AF
```

Example: Tests for buffer overflow; nop.txt should be a text file containing "NOPNOPNOPNOPNOP"

```
hping2 localhost -E nop.txt
```

Example: 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
20071107-11:37.31 192.168.0.101 OS Fingerprint
20071108-03:28.22 192.168.0.101 Buffer Overflow
20071109-05:27.54 192.168.0.101 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 allows you to test your program without the need of another computer on the network.

7. Required Document (5%)

In addition to your source code and your output you are required to submit a memo to your instructor 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, then it will be assigned a grade of zero.

If you're not sure about the format for a memo, then just search for "memo format" on Google.

Your memo should state clearly the status of the assignment. Is it done or not? Does it meet all of the requirements? If anything is missing then state clearly what is missing from your work.

Failure to give a status for the assignment will result in a grade of zero for the entire assignment. Providing a status that's 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 read. If you encountered problems with the assignment, then provide a detailed description of those problems and the solution(s) you found.

8. Assignment Submission (5%)

You are required to submit your work online, using the Blackboard. Late work will be accepted, however it is subject to late penalties as described in the syllabus.

Here are the requirements for your submission:

- Submit your electronic copy using the Blackboard (attach the assignment as a compressed archive file (.zip, .tgz, .tbz2, .rar)
- The name of the compressed archive should be: *fistName-lastName*-PA-*assignmentNumber.zip* (e.g. Jane-Doe-PA-7.zip)
- Include (i) your memo, (ii) a completed handout with your results and your name on it, (iii) the source code, (iv) a README file that explains how to build and on how to run each program, (v) compiled executable(s) and (vi) a shell script or batch file that will compile your programs when executed; a Makefile (see GNU Make for details) would be great, however any form of script or building tool will do
- 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 field by including the word COMPLEMENT

Id