Hortin
PA3

Documentation Tasks (35 pts)
1) Cryptographic Checksum of your Source Code (0 pts)
Compute an md5 or SHA1 checksum of a tar file (or similar archive) of your source code and executable image. Include the name of this file, it's size, and it's checksum in your write-up. Assignments turned in without a checksum will be subject to a grading penalty.

| **Name** | **Size** | **MD5sum** |
|---|---|---|
| **cpts425p3Hortin.tar.gz** | **4387** | **e71e93844152885d47b0caa064bf3770** |

2) Abbreviated Software Design Document (10 pts)
Briefly describe the data structures and algorithms used to implement the three required tools. Give the sources of all third-party code and cryptography tools/mechanisms that you used in this assignment. (1-2 page limit)

**The first thing I did in this assignment was outline the cases that I wanted to check for in both catagories. For misuse I knew the first one that I wanted to check for was correct uses of the tags. It seems only naturaly that a message would be dropped if it was in the improper format. My next misuse case was the proper port user pairs that would be input by the sender. If the sender does not know the name and address of the user the message is to be forwarded to I consider this a misuse. This would alleviate any type of spam message generated by simply trying all ports. Anomaly cases were the next things I came up with and I decided that I would check the rate of incoming messages and drop all messages until that max rate was reachieved. The final check is that you must check to see if the total message length exceeds the value.**

**Like the assignments we have had in the past for this class I decided to do it in python. As I have mentioned before I really like the list processing and string handling operations that make this project a little easier to do. The only new thing I used for this project as opposed to the others was the time module, which I used to calculate project run time as well as append time since epoch to the messages in alice. I used the socket classes that I used in the 2nd assignment, but other than that I simply used processing on lists and string, as well as lists generating from strings.**

**The only other real challenge was to create a version of alice that could be used to properly create messages that could be used to test all of the misuse and anomaly cases. This required a little bit of modification as I went along, but was priceless as a implemented and tested my cases.**

3) Software Test Document (15 pts)
Describe your software test plan and methodology that you used to verify that you implemented the three required software tools correctly. List any know deficiencies. (1-2 page limit)

**For this I implemented the basic alice and bob first. This was so I could experiment with message passing and make sure that the messages were sending and constructing properly. This was fairly easy since I had tested this for the last project. The new part that needed testing was the way I constructed messages so that they met the proper tags. This was fairly easy to test since all it required was reading and sending messages. Once i was satisfied that these simple programs worked I went to move on to program eve.**

**Eve was a more interesting since I needed to first come up with my cases. After I implementedthem to test them I had different versions of alice running. The different test scenarios can be found in README or below.**

```
//USED TO DEMONSTRAIGHT ANOMALY
//Hold enter on alice to flood the max rate, than let go for a while to return service
//Also can enter a long message "Im going to be a dropped message because I exceed the rate limit
and i really hate to do this blah" to send and break wait limit
./alice -p 2000
./bob -p 2001
./eve -f bob 2001 -r alice 2000 -l 130 -s 20


//USED TO DEMONSTRAIGHT MISUSE
//First command used to send a good message
//Second command used to send a message with bad tags
//third command used to send a bad port message

./alice -p 4000 -f msg
./alice -p 4000 -f badtagmsg
./alice -p 4000 -f badportmsg
./bob -p 4001
./eve -f bob 4001 -r alice 4000 -l 1000 -s 20000
```

**I ran all these commands to demo, but I used tested as I went along too. These commands were the most effective way to demo, and it let me speed up the process.**

4) Lessons Learned / Project Evaluation (10 pts)
Discuss the lessons learned from this project. Are there improvements that should be made in a future release? Other possible questions to address: Was this assignment as easy as you anticipated? Did implementing your IDS inspire you to become a network security analyst? Etc.

**This assignment was fairly straightforward. The hardest part was the planning stage where I needed to decide what kind of things I wanted to check for and how to go about that. Once I knew what my plan was the implementation was easy. The most time consisted of thinking about**

**the implementation and than testing it. It is very frustrating to test with this program, because most of the time an error would require the relaunching of 3 applications...and occasionally the reconfiguring. That was the most frustrating part of the assignment for me.**

**This IDS project did not really excite me like the other assignments did. Most of the time it seems like a security analyst would spend a lot of time going through logs. I think that as an officer in communications for the air force writing this assignment helped me think about how I might need to review logs in case of a breach**

5) Time Logs (Optional, no points)
List the time spent during software design, implementation, testing. Also list the time spent writing up your project.

**Implementation- 3 hours**
**Testing – 2 hours**
**Writeup – Developing demo script - 2 hours**