Intrusion Detection
Programming Assignment #3
CptS 425/580—Network Security
Due: 9 Dec. 2005

Objective:
Implement a very simple intrusion detection system that uses both misuse and anomaly
detection algorithms.

Overview:
Design an implement programs to send and receive messages via an intermediate Intrusion
Detection System. If desired you may use the same framework that you developed in Program
#2. Namely, your sender(s), IDS, and receiver(s) would correspond to process A, process E, and
process B, respectively. In order to simply this assignment, each message that is sent will be
encoded in the following manner.

\[
<MSG>
\begin{align*}
<\text{SRC}> & \text{source} </\text{SRC}> & \text{message source—variable length string} \\
<\text{DEST}> & \text{destination} </\text{DEST}> & \text{message destination—variable length string} \\
<\text{USER}> & \text{user</USER>} & \text{user/process sending the message—var.len. string} \\
<\text{VALUE}> & \text{value</VALUE>} & \text{numeric value (floating point format)} \\
<\text{TEXT}> & \text{text</TEXT> & \text{variable length (multi-line?) string}
\end{align*}
\]

Your IDS will be required to implement two misuse detection rules and two anomaly
detection algorithms. This will, of course, likely necessitate the creation of either several
sending programs and/or senders that can be configured to exhibit both ‘good’ and ‘bad’
behavior.

Software Design & Implementation Tasks (75 pts)

1) Audit Log (15 pts)
You IDS should log data about the messages that it receives and whether these messages were
routed on to their intended receiver or simply dropped.

Minimum Requirements: Demonstrate that your IDS can record when messages are received
and whether these messages were forwarded on or dropped/deleted by the IDS.

2) Misuse Detection (30 pts)
Design and implement two misuse detection rules. The parameter(s) for at least one of these
rules must be user-configurable. (I.e., you may not have two rules with hard-coded parameters).

Minimum Implementation Requirements: Demonstrate that both ‘good’ and ‘bad’ messages
may be sent. These two rules must be based upon different message fields.

3) Anomaly Detection (30 pts)
Design and implement two anomaly detection algorithms. The parameter(s) for at least one of
these algorithms must be user-configurable. (I.e., you may not have two rules with hard-coded
parameters). In order to limit down-time during your program demonstration, please update
your results/calculations every few seconds rather than every few minutes.
Minimum Implementation Requirements: Demonstrate that both ‘good’ and ‘bad’ messages may be sent. Demonstrate that your IDS adapts over time to changing definitions of ‘normal’. These two rules must be based upon different message fields.

Written Deliverables (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.

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)

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) Include an appendix that lists the various private / public keys used for one run of your man-in-the-middle attack. (Page limit will vary based on font size, etc.)

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.

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

Additional Notes:
• You may implement this assignment in the programming language of your choice.
• If a set of test files is given, they will be given at least one week before the assignment is due. You will not receive full credit if your tools cannot correctly process these files. But, these files by themselves will not be sufficient to fully test your tools.
• Program demonstrations will occur in the department’s computer lab. (This does not mean that your code has to execute on a departmental machine, you may ssh to another host or bring in a laptop if you choose).