# Intrusion Detection
## Programming Assignment #3
CptS 425/580—Network Security
Due: 29 April 2009, 11:59 PM

## 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 Alice, Eve, and Bob, respectively. In order to simplify this assignment, each message that is sent will be encoded in the following manner.

```
<MSG>
  <SRC>source</SRC>          message source—variable length string
  <DEST>destination</DEST>   message destination—variable length string
  <USER>user</USER>          user/process sending the message—var.len. string
  <VALUE>value</VALUE>       numeric value (floating point format)
  <TEXT>text</TEXT>          variable length (multi-line?) string
</MSG>
```

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.

# 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.

## 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.

# Deliverables & Testing (0 pts)

**1) Source code tarball & documentation (0 pts, 3 penalty pts)**

Email Dr. McKinnon and the TA one PDF file and a gzip'd tar file of your source code. The email message shall conform to these requirements:

| | |
|---|---|
| subject line: | `CptS425: Program#3 from LAST_NAME` |
| tarball file name: | `cpts425p3LAST_NAME.tar.gz` |
| write-up name: | `cpts425p3LAST_NAME_writeup.pdf` |

**2) Makefile / Building your programs (0 pts, 3 penalty pts)**

Executing the command: `tar -xzf cpts425p3LAST_NAME.tar.gz` shall extract your source code and a Makefile in to the current working directory. Executing `make` shall build/compile your programs. If your programs do not need to be compiled (e.g., you are using an interpreted scripting language), then your Makefile's 'all:' target may be empty.

**Additional Notes:**

- You may implement this assignment in the programming language of your choice.
- Program demonstrations will occur on departmental machines (Sloan 353 for Pullman students, West 151 for Tri-Cities students) unless prior approval is granted by the instructor.