

CMSC 426 / 626 – Principles of Computer Security
Prof. Krishna Sivalingam
Fall 2006
Project 2: Secure Programming
Due Dates (On-Line Submission via submit):
Part 1: Your completed project **December 6, 11PM EST.**
Part 2: Your testing of another project **December 13, 11PM EST.**

1 Objectives

The objective of this project is to learn about securing coding techniques and practices. This project is for CMSC 426 students ONLY.

Students taking CMSC 626 will be required to use the coding practices as listed in Task 1 for their Term Project implementations (for the FINAL phase only, NOT for the interim phase due Nov 22). If you are using Java for the Term Project, then your final report should explain how secure coding has been handled.

In particular, CMSC 426 students are required to accomplish the following tasks:

Task 1: Make your Project 1 code robust and secure, as described below. For those who did NOT submit a completed Project 1, you MUST finish the *User Authentication* and *Symmetric Encryption* parts completely, to gain credit for this project.

- ▷ Incorporation of safe string mechanisms from the *safestr* package available from www.zork.org. A sample starting code and documentation will be made available from the class website.
- ▷ Update your Project1 code to make sure that no buffer or integer overflows can occur; and format string vulnerabilities are not present.
- ▷ Input parsing and validation is done completely to accept only acceptable input
- ▷ Follow the standards, to the extent possible, set forth in Coding Standards for C/C++ as outlined in
<https://www.securecoding.cert.org/confluence/display/seccode/CERT+Secure+Coding+Standards>
<https://www.securecoding.cert.org/confluence/display/seccode/Top+10+Secure+Coding+Practices>

Task 1 is worth 60 points.

Task 2: Documentation and Testing:

- ▷ A detailed technical document that describes ALL mechanisms that your program uses to ensure secure execution and to what extent the coding standards and best practices were followed.
- ▷ The use of at least one run-time system that will detect buffer overflows and other problems. Specific examples of run-time systems that will work in the Linux/GCC environment will be provided shortly.
- ▷ A detailed testing plan and results from this testing - this will include both white box and black box testing. It is presumed that you have already taken CMSC 345 (Software Engineering).

Task 2 is worth 20 points.

Task 3: Vulnerability Testing:

By December 7, you will be assigned another student's executable along with his/her README/User Guide. Your task is to conduct black box testing with different invalid inputs and buffer overflow possibilities and DOCUMENT the results of the testing in a report that you will submit by Dec. 13th.

Task 3 is worth 20 points.

2 What to Submit

Name your project directory as PROJECT2 (Note: ALL UPPERCASE)

Now log into one of the campus GL machines. Make sure that ONLY files related to this project are in your submission directory - we do not want any temporary files, music files, etc. Once you are ready to submit, change directory to the directory above PROJECT2, and tar all files in the directory with the command:

```
tar czf project2.tgz PROJECT2
```

Note that this will only include materials for Task 1 and Task 2, that are due by Dec. 6th.

Make sure that all files have been correctly tar-red with the command:

```
tar ztf project2.tgz
```

Then, submit using the command:

```
submit cs426 proj2 project2.tgz
```

The directory should contain the following files:

- ▷ Source Files
- ▷ Makefile
 - Typing command 'make' at the UNIX command prompt, should generate all the required executables.
- ▷ A Script file obtained by running UNIX command *script* which will record a sample of testing with incorrect inputs, etc.
- ▷ The technical report for Task 2 will be acceptable only in PDF format. Make sure that the document is well formatted to clearly identify test cases and whether your program crashed/incorrectly executed for those test cases.
- ▷ a README file containing instructions to compile, run and test your program. The README should document known error cases and weaknesses with the program. You should also document if any code used in your submission has been obtained/modified from any other source, including those found on the web. If you helped any other person and/or took help from/discussed with any other person, please describe it here.
- ▷ a COMMENTS file which describes your experience with the project, suggestions for change, and anything else you may wish to say regarding this project. This is your opportunity for feedback, and will be very helpful.

For Task 3, upload a PDF file containing your report, by Dec 13th, with the command:

```
submit cs426 proj2 Task3-Report.pdf
```

3 Grading

- ▷ Task 1: 60 points
- ▷ Task 2: 20 points
- ▷ Task 3: 20 points

For those who did NOT submit a completed Project 1, you MUST finish the *User Authentication* and *Symmetric Encryption* parts completely, to gain credit for this project.

No README/COMMENTS: -5 points; No Script File: -10 points; Incomplete Compilation: -10 points