CSc 466/566 Computer Security

Assignment 2

Due Oct 10, 2014
Worth 5% (ugrads), 5% (grads)

Christian Collberg
Department of Computer Science, University of Arizona

1. Introduction

In this exercise you will learn how to hack code and protect it from being hacked!

2. Task A: Reverse Engineer an Obfuscated Program [40 points]

1. Download your individualized assignment from `http://www.cs.arizona.edu/~collberg/Teaching/466-566/2014/Assignments/index.html`.
2. Unpack:
   (a) `unzip exam.zip`
   (b) `cd exam/YOURFIRSTNAME_YOURLASTNAME`
   (c) `ls`
3. You will find several files: *3-challenge.c*, *3-answer.c*, *4-challenge.c*, and *4-answer.c*.
4. Your task is to reverse engineer **one of** `3-challenge.c` or `4-challenge.c`, by removing the obfuscation that I (or Tigress, actually) has added.
5. You should edit `3-answer.c` or `4-answer.c` writing your reverse-engineered code in the empty function `SECRET` that has been provided for you.
6. In a successful solution to N-challenge
   (a) `N-answer` should have the same input-output behavior as `N-challenge`;
   (b) `N-answer.c` should be idiomatic C, i.e. have a structure that "looks like normal C written by a human";
   (c) all unnecessary code (i.e. code added or transformed by the obfuscator) will have been removed.

   For example, I expect loops in the source program to have corresponding loops in the recovered program, flattened or virtualized code will have been returned to its pre-obfuscated state, and compound data types (arrays, structs, and unions) should be identified as such.
7. Create a `README-A.txt` file with the following information, exactly in this format:

```
FIRSTNAME:     Bob
LASTNAME:      Jones
EMAILADDRESS:  bob@cia.gov
UNDERGRADUATE,MASTERS,PHD (U/M/P): P
PROBLEM (3/4): which problem did you solve
TOOLS:         what techniques did you use
TECHNIQUES:    what tools, if any did you use
TIME:          how long did it take you
DIFFICULTY:    was it easy/hard
CHALLENGES:    what, in particular, did you find particularly easy or hard
COMMENTS:      was this a reasonable exam (too easy, too hard), did you enjoy it,
               comments about the course in general if you wish, etc.
```

8. If you solve 3-challenge, you can get a maximum of 35 points. If you solve 4-challenge, you can get a maximum of 40 points.

3. Task B: Crack a Program [30 points]

1. Download your individualized assignment as in the previous part.
2. You will find two Linux executable files `5-challenge` and `6-challenge`.
3. Your task is to disable an expired-time check (for 5-challenge) and an expired-time check and a password check (for 6-challenge) that prevent the program from running.
4. Create a `README-B.txt` file with the following information, exactly in this format:

```
FIRSTNAME:     Bob
LASTNAME:      Jones
EMAILADDRESS:  bob@cia.gov
UNDERGRADUATE,MASTERS,PHD (U/M/P):
PROBLEM (5/6): which problem did you solve
TOOLS:         what techniques did you use
TECHNIQUES:    what tools, if any did you use
TIME:          how long did it take you
DIFFICULTY:    was it easy/hard
CHALLENGES:    what, in particular, did you find particularly easy or hard
COMMENTS:      how hard was this, did you enjoy it, what could have been
               done differently, etc.
```

5. If you solve 5-challenge, you can get a maximum of 25 points. If you solve 6-challenge, you can get a maximum of 30 points.

4. Task C: Protect with Tigress [30 points]

To try out Tigress, do the following:

1. Download and unzip the latest version of tigress from `fromhttp://tigress.cs.arizona.edu/#download`.
2. Depending on your shell, set the following environment variables:

```
> setenv TIGRESS_HOME /PATH_TO/tigress-1.3
> setenv PATH /PATH_TO/tigress-1.3:$PATH
   or
> export TIGRESS_HOME=/PATH_TO/tigress-1.3
> export PATH=$PATH:/PATH_TO/tigress-1.3
```

3. Try out Tigress:

```
        tigress --Transform=Virtualize --Functions=main --out=result.c test2.c
```

   This should construct a trivial interpreter from `test2.c` in `result.c`.

4. Some useful commands:

```
*) tigress --help     : Show how to use tigress
*) tigress --options  : Show complete list of options to tigress
*) tigress --license  : Display the tigress license
*) tigress --bibtex   : See how to cite us
*) tigress --apple    : See how to get past some Darwin issues
```

5. Read *all* the documentation of Tigress, at `tigress.cs.arizona.edu`.

Now do the following:

1. Write a short (50–100 lines of code) C program (called `program.c`) that has some sort of "asset" that you would like to protect/hide, such as a license check, an algorithm, or a piece of data.

2. Use Tigress to protect your program. I want you to experiment with writing scripts (commands that call tigress with different sequences of transformations) to get different levels of protection, at different slowdowns.

3. Construct at least *three* different scripts and a makefile (called `makefile-C` that generates the differently protected versions of your program. The makefile should look something like this:

```
all: out1.c out2.c out3.c
out1.c : program.c
   tigress transformations --out=out1.c program.c
out2.c : program.c
   tigress transformations --out=out2.c program.c
out3.c : program.c
   tigress transformations --out=out3.c program.c
```

   In other words, the TA will only type `make -f makefile-C` to generate your protectedd programs.

4. Construct a file `README-C.txt` that describes what you did:

```
   FIRSTNAME:    Bob
   LASTNAME:     Jones
   EMAILADDRESS: bob@cia.gov
   UNDERGRADUATE,MASTERS,PHD (U/M/P): P
   PROGRAM:      what does your program do?
   ASSET:        what asset are you protecting?
   SCRIPT1:      why did you choose the particular sequence of transformations for
                 script one, how well do you think your asset is protected, and
```

```
              what slowdown did you see?
SCRIPT2:      same as SCRIPT1, but for the second script
SCRIPT3:      same as SCRIPT1, but for the third script
COMMENTS:     how hard was this, did you enjoy it, what could have been
              done differently, what transformation did you need from
              Tigress that it does not have, what bugs did you find in
              Tigress, etc.
```

5. Academic Integrity

This is an individual take-home assignment and it is obviously possible for you to get help solving it. This, however, **IS NOT ALLOWED**. You are bound by the University's rules of academic conduct as well as these rules:

1. You may use any technique that you want to work the assignment.
2. You may use any tools (including those that you download from the web, buy, or build yourself) that you want to work the assignment. Previous students report that they have used: IDA 6.1 64bit, gdb, strace, hexedit, ltrace, hxd hex editor, asm, nm, faketime, hexeditor, python, grep, graphviz, VS2010, pen and paper (!), notepad++, Dev-C, GHex, geany, brain (!), bless, codeblocks, WinMerge, freemind.
3. You are not allowed to discuss the assignment with any human. This includes
   - Posting questions and getting help from online forums;
   - Getting help from classmates;
   - Getting help from anyone outside class.
4. You may not *provide* help to your classmates.

If you have any questions about the assignment you should see me or the TA.

6. Submission

Zip up our answers (`3-answer.c` or `4-answer.c`, `5-challenge` or `6-challenge`) and the `README-*.txt` files and upload to `d2l`. Name your zip file `FIRSTNAME_LASTNAME_2.zip`.