UNIVERSITY OF WATERLOO
Cheriton School of Computer Science

CS 458/658 **Computer Security and Privacy** **Spring 2016**

Erinn Atwater and Doug Stinson

ASSIGNMENT 2

Assignment due date: **Thursday, June 23, 2016 3:00 pm**

Total marks: 68
Written Response Questions TA: Justin Tracey (Office hours: Thursday 12:00–1:00 pm in DC 3332)
Programming Question TA: Nabiha Asghar (Office hours: Monday 2:30–4:00 pm in DC 3332)

Please use Piazza for all communication. Ask a private question if necessary. The TAs' office hours are also posted to Piazza for reference.

Written Response Questions [33 marks]

Note: For written questions, please be sure to use complete, grammatically-correct sentences. You will be marked on the presentation and clarity of your answers as well as the content.

1. [Total: 15 marks] **Identification/Authentication Protocols**

   (a) [11 marks] In class you were taught that challenge-response protocols can be used for authentication purposes. Consider a scenario involving two friends, Alice and Bob, who often communicate with each other using online chatting software. Each of them possesses a 'key' K, a secret mutual password (known only to Alice and Bob), which they shared with each other offline. In addition, they have access to a publicly known keyed function f, whose inverse function f^{-1} is extremely hard to compute. In order to mutually authenticate themselves to each other, Alice and Bob are going to use the following secure mutual identification protocol.

   Either Alice or Bob may initiate the following interactive protocol. The following describes the protocol when Bob initiates:

   - Bob sends a random string r_1 to Alice.
   - Alice chooses a random string r_2, computes $y_1 = f(K, \text{“}Alice\text{”}||r_1||r_2)$ and sends r_2 and y_1 to Bob. ($x||y||z$ is the concatenation of x, y, and z).
   - Bob computes $x_1 = f(K, \text{“}Alice\text{”}||r_1||r_2)$. If $x_1 \neq y_1$, he aborts the chat session, claiming that the person on the other end is not Alice. If $x_1 = y_1$, Bob computes $x_2 = f(K, \text{“}Bob\text{”}||r_2)$ and sends x_2 to Alice.

- Alice computes $y_2 = f(K, \text{“}Bob\text{”}||r_2)$. If $y_2 \neq x_2$, she aborts the chat session, claiming that the person on the other end is not Bob. Otherwise, she continues the chat session with Bob.

The protocol specification recommends that r_1 and r_2 should be random 64-bit strings. Bob has an executable version of the function f with the following prototype:

```
uint64 MyF(uint64 K, unsigned char NameTag[20], uint64 r)
```

(where `char`, as usual, is 8 bits, and `uint64` is an unsigned 64-bit integer).

Note that `MyF()` cannot correctly implement the protocol: its third parameter, `r`, is too small for $r_1||r_2$. Bob does not have the code of `MyF()`, and cannot modify its implementation to fit the protocol. Rather, Bob decides to modify the protocol to fit the implementation of `MyF()`, and Alice accepts the modifications.

For each modification below, show that it is insecure. To do so, you need to describe how Mallory can impersonate Bob without access to K (i.e. after Alice and Mallory run the protocol, Alice believes she is communicating with Bob).

We will make the following assumptions:

- Mallory knows the protocol but not the value of K,
- Mallory can read all the messages exchanged between Alice and Bob (i.e. she can see r_1, r_2, y_1 and x_2),
- Mallory can compute f but not its inverse f^{-1},
- Mallory can initiate as many chat sessions as she wants, with any user, and
- Mallory can intercept and modify messages sent between Alice and Bob (i.e., she can carry out man-in-the-middle attacks).

i. [5 marks] Bob believes the name tags are redundant and removes them. He passes r_1 encoded as a `char` array as the second parameter of `MyF()`, and r_2 as the third parameter of `MyF()`.

ii. [2 marks] Bob decides to ignore the recommendation. He uses: $18302628885633695807 \& r_1 + r_2$ (and just r_2 when there is no r_1) as the third parameter of `MyF()`. “&” is the bitwise AND operator.

iii. [2 marks] Bob chooses to replace $r_1||r_2$ by $r_1 + r_2 (\text{mod } 2^{64})$ in the protocol. The result is a 64-bit integer, which Bob passes as the third parameter of `MyF()`.

iv. [2 marks] Alice and Bob have managed to implement the protocol correctly. However, Alice's random number generator is faulty. Every time she re-boots her computer, the first challenge she issues in this protocol (when she is the initiator) will be the same.

(b) [4 marks] Alice's card is locked because she incorrectly entered her debit card PIN several times instead of her credit card PIN. She calls her bank, and the bank tells her that she can change her PIN on an ATM but not over the phone. She needs to insert her credit card into an ATM of the bank and enter the correct PIN twice. What are the authentication factors used by the bank to authenticate Alice? How do Alice and the bank identify each other?

2. [Total: 10 marks] **Behavioural Authentication**
Alice usually forgets to lock her cellphone, so she has developed an application for her cellphone that locks it if a finger swipe of the user does not match Alice's finger swipe pattern.[1] The false acceptance rate (FAR) of the application is 7% (i.e., the phone incorrectly identifies a stranger's finger swipes as Alice's swipes 7% of the time). The false rejection rate (FRR) of the application is 12% (i.e., the phone incorrectly identifies Alice's swipes as a stranger's swipes 12% of the time). Being rejected frequently, Alice decides to add a *window* of eight swipes. A rejected swipe increases the rejection counter by one. Her application locks the cellphone as soon as the rejection counter reaches four, and the counter is set to zero every eight swipes (the window is non-overlapping). Suppose Alice knows that her cellphone is being used by others 8% of the time.

   (a) [4 marks] What are the new values for the FAR and FRR of the application for each non-overlapping window (eight swipes)?

   (b) [6 marks] If the cellphone locks within the first 8 swipes, what is the probability that it is locked correctly (i.e., a stranger was using the cellphone)?

   Hint: You should use Bayes' Rule for conditional probabilities to solve this question: `http://en.wikipedia.org/wiki/Bayes'_theorem`

3. [Total: 8 marks] **Ransomware**
The Globe and Mail published an article on May 20 entitled Ransomware in real time: How hackers infiltrate secured systems. This article can be found at the following link:

   `http://www.theglobeandmail.com/technology/ransomware-in-real-time-how-hackers-infiltrate-secured-systems/article30111818/`

   The purpose of this question is to explore several issues raised by the article. Please give your answers in clear, complete sentences. You may wish to do some research on the internet to help answer questions. This is acceptable provided you cite your sources appropriately.

   (a) [2 marks] Teslacrypt encrypted files using AES, which is a secret-key cryptosystem (the same key is used for encryption and decryption). Why would this be preferable to using a public-key cryptosystem, where the encryption key is public but the decryption key is secret?

   (b) [2 marks] In at least one version of Teslacrypt, an encrypted version of the AES key was stored on the victims computer in a file key.dat. Methods were found to decrypt the encrypted AES key, thus enabling the encrypted files to be decrypted without paying a ransom. However, these techniques were not widely publicized. What would be the motivation for trying to keep these methods of defeating the attack secret?

   (c) [2 marks] What does the author mean when he says that it cycles that same 128-bit block 14 times?

   (d) [2 marks] On May 18 a universal decryption key for Teslacrypt was released by the developers. Why might they have done this?

[1] This technology is called *implicit authentication*, and is currently an active topic of research.

Programming Question [35 marks]

Background

You are tasked with performing a security audit of a custom-developed *web application* for your organization. The web application is a content sharing portal, where registered and confirmed users can post, comment, and vote on articles. It is known that the content sharing portal was *very poorly written*, and that in the past, the web server has been compromised to allow users to log in without a password, or allowed clients to perform restricted actions such as voting as another user. As you are the only person in your organization to have a background in web security, you are tasked with examining the web application and *identifying any violations of certain security principles*, and *demonstrating that these are exploitable*, so that your organization may properly fix the vulnerabilities. Instead of having the full source code however, you are given nothing except outside access to the server. You are given the userid `alice` with the password `passw0rd` to try out the system.

Application Description

In *The Protection of Information in Computer Systems*, Section I.A., Saltzer and Shroeder present eight design principles for trusted systems. Unfortunately, many real systems still fail to follow the principles faithfully. For this question, you are to examine a simple web application in order to identify violations of Saltzer and Shroeder's principles.

You are asked to find five sploits violating the security principles. **Each sploit must perform a unique persistent[2] action by a different (non-`alice`) user of the portal.** Moreover, each sploit must exploit a different vulnerability in the portal. For each sploit, you should do the following:

1. Write a program to exploit each vulnerability that you uncover (4 marks each).
2. Explain how each sploit works and how you were able to find it (2 marks each).
3. State exactly which one of Saltzer and Shroeder's eight design principles was violated, and explain (1 mark each).

Rules for sploit execution

For your sploits, you should submit five tarballs (sploit[1-5].tar) each containing the following:

- **sploit.sh** - shell script to run your sploit
- any other files needed to run your sploits

[2]Here, "persistent" means that the effect of the sploit should be visible by a guest user visiting the portal. Once you shut down the uml virtual machine, however, the portal will be reset to its original state.

Each script must accept **exactly one command-line argument**, which will be the **HTTP path** of the web application, e.g.

```
./sploit.sh ugsterXX.student.cs.uwaterloo.ca/userid
```

The scripts should *not* be within a directory in the tar file.

When run, each script should do the following:

1. compile the exploit code (if necessary)
2. execute the exploit code
3. output a message saying what it is doing

The purpose of your sploits should be to perform some action without the privilege to do so. In general you will need to impersonate a user in order to do this. For each vulnerability, there is a particular user account to be exploited, and a particular action that the user is capable of doing.

If we can't figure out how your sploit works, you will not earn any marks for it. If in doubt about how to present a sploit (or whether an exploit is acceptable), please send a message on Piazza to the programming TA or check during office hours.

What to hand in

Using the "submit" facility on the student.cs machines, hand in the following files:

a2.pdf A PDF file that contains your answers to all written response questions, plus parts 2 and 3 for each of your exploits.

sploit[1-5].tar (uncompressed) tar files containing your completed exploits for the programming question.

Note 1: You must include your name, your uWaterloo userid, and your student number at the top of the first page of a2.pdf. Failure to do so will result in a deduction of 5 marks from your final score on this assignment! Also, be sure to "embed all fonts" into your PDF files. Some students' files were unreadable in the past; if we can't read it, we can't mark it.

Note 2: We have installed a script on the ugsters that will let you check your submission format. Simply run the command `check` on the ugster from a directory with your prepared submission files. It will do some basic checks on the file names and formats, and output a detailed success or failure message.

Useful Information For Programming Sploits

A web server is running on each of the ugster machines, and a directory has been created using your ugster userid. Your copy of the web application can be found at:

`http://ugsterXX.student.cs.uwaterloo.ca/userid`

...where `ugsterXX` and `userid` are the machine and credential assigned to you previously (see Piazza).

If you need to reset the webserver, simply access the following URL:

`http://ugsterXX.student.cs.uwaterloo.ca/reset/userid`

This will delete all changes made to your copy of the web application, and restore it to its original state.

As with the previous assignment, the ugster machines are only accessible to other machines on campus. If you are working from home, you will need to first connect through another machine (such as `linux.student` or the campus VPN).

Note 1: If one of your exploits forces the application into an unusable state, you can restore the default state by accessing the URL above. If the web server itself becomes unusable, please report this on Piazza, along with a description of what you were doing when the problem occurred. *Do not perform denial of service attacks* on either the ugster machine or the web server.

Note 2: **Do not** connect to the web server with a userid different from the one that is assigned to you. Any student that is caught messing around with another student's web application by connecting to the wrong URL will receive an **automatic zero** on the assignment, and further penalties may be imposed.

Writing your exploits

You may use any language of your choosing (within reason) to exploit the server; the only restriction is that your exploits run correctly from the ugster machines. You may use external tools to aid you if they are not directly required for exploit execution (i.e. they help you gather information, etc.). If your programming language of choice is not currently supported on the ugsters, you can request that the appropriate packages be installed (as long as you do it well in advance of the assignment due date).

The tools **netcat** and **telnet** may be useful in writing and testing your exploits. These tools enable you to make network connections from a command line. There are plenty of resources on the Internet that explain how these tools can be used to interact with a web server via HTML GET commands; Google is your friend. A good starting point might be:

`http://www.stearns.org/doc/nc-intro.v0.9.html`

A basic understanding of HTML forms will also be helpful for completing this assignment; a good starting point for learning about forms might be:

`http://www.cs.tut.fi/~jkorpela/HTML3.2/5.25.html`

Here are some other links that you may find useful:

- cURL: `http://curl.haxx.se/docs/manpage.html` (information about headers, GET, POST)
- cURL Scripting: `http://curl.haxx.se/docs/httpscripting.html` (if you don't know how HTML forms work)
- Shebang: `http://en.wikipedia.org/wiki/Shebang_(Unix)` (if you are new to scripting)
- HTTP: `http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol` (if you are new to how HTTP/web works)
- Web sessions: `http://en.wikipedia.org/wiki/Session_(computer_science)` (go to the section on Web Server Session Management)