UNIVERSITY OF WATERLOO
Cheriton School of Computer Science

CS 458/658 **Computer Security and Privacy** **Fall 2020**

ASSIGNMENT 1 - SOLUTIONS

Total marks: 70
Written Response Questions TA: Sina Faraji
Programming Question TA: Nils Lukas, Rasoul Akhavan Mahdavi

Written Response Questions [26 marks]

For written questions, 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.

Question 1 [Total: 18 marks]

Based on a true story ...

you've been hired by the social media company "ButtonSmash" to analyze a recent security breach on their platform. ButtonSmash employs a username/password authentication system and users are asked to provide an email address and a phone number in case they forget their password (password reset) or want to enable an optional two-factor authentication (2FA).

Please note that the answers below have didactic purposes. There aren't unique solutions to most the questions and if you have supported your answers with reasonable arguments, you will be graded accordingly.

1. (8 marks) In the following, Identify which of availability, confidentiality, integrity, and/or privacy is violated with a brief explanation.

   (a) (2 marks) Eve, the attacker, was obsessed with the username @*CryptoBob*. Determined to acquire it, she tracked down Bob and followed him for days. Finally, Eve got her shot and caught Bob logging into his ButtonSmash account at a cafe and carefully watched him enter his password.

   Confidentiality: Eve now has access to data she is not authorized to.

   (b) (2 marks) Eve quickly tried to log into Bob's account but found out that 2FA was enabled. She then asked Bob if she could use his phone to call her mom and Bob kindly agreed. With Bob's phone number, Eve then did what is known as a *SIM swap* attack. She called the telecommunication company's customer support:

Agent: "Hello! How can I help you?"
Eve: "I lost my phone yesterday and I'm waiting for a very important phone call. Could you please port the phone number to my new SIM card?"
Agent: "Sure, no problem!"

Integrity: The new SIM card data is not the correct data and has no affiliation with Bob.

(c) (2 marks) Bragging about her successful hack on a public forum, Eve got a message from Eva, a vengeful former ButtonSmash employee. Eva knew about an admin page that would give them ultimate power over the platform. She helped Eve to make the following phone call to ButtonSmash's employee office:

Agent: "ButtonSmash's employee office, This is Bryan speaking."
Eve: "Hi, I'm Alice, one of the new interns. Jack told me to contact you for the admin page credentials.
Agent: "Yes, so the username is *adminsmash* and the password is *button1234*"

Confidentiality: Eve has access to a system she is not authorized to.

(d) (2 marks) After logging into the admin page, Eve and Eva were surprised by how much damage they could do to the platform. As their first move, they cut out all the other employees from accessing the admin page, making themselves almost unstoppable.

Availability: Employees that were supposed to have access to the admin page can no longer access the system.

2. (8 marks) For each of the following scenarios, Answer whether the threat is one of interception, interruption, modification and/or fabrication. Briefly explain your choice of compromise.

(a) (2 marks) On the admin page, Eve started to change email addresses of multiple accounts to an email address of her own. Then she normally requested a password reset email and changed those accounts' passwords.

Modification: Eve, an unauthorized user is tampering with other users' data.

(b) (2 marks) Next, she started reading through private messages of her favorite celebrities' accounts and downloaded all their data onto her hard drive.

Interception: Eve is illicitly copying data files she isn't authorized to have access to in the first place.

(c) (2 marks) Eve's master plan was to promote a Bitcoin scam. She started posting on behalf of celebrities and political figures that as a COVID-19 charity all bitcoins sent to a specific address would be awarded with double the amount.

Fabrication: Eve is inserting counterfeit objects into the database of ButtonSmash on behalf of other users.

(d) (2 marks) Some of the users noticed the scam and started posting to let other users know that it's not legit. As a counter measure, Eve deactivated those accounts.

Interruption: Assets on ButtonSmash are no longer available to authorized users and they can't use the system.

3. (2 marks) It seems like ButtonSmash suffers from a flawed access control design. Propose an access control structure that would properly protect the resource allocation on the admin page.

   ButtonSmash should use a Role-Based Access Control system. There are two major flaws.

   - Separate access management into two roles adhering to "separation of duties":

     • "Employee Manager" verifies employee information and sends over data to "Access Control Manager" to provide employee with credentials.

   - Hierarchical roles on the admin page itself e.g.:

     • Admin Role: Employees with administrative privileges in the system i.e. full control.
     • Senior Employee Role: Employees that have an established reputation in the company and can access assets based on their department/responsibilities.
     • Entry Employee Role: New employees that could have access to e.g. testing/deployment assets.

   An intern could fall into the Entry Employee Role.

Question 2 [Total: 8 marks]

1. In the aftermath of the attack, FBI started an investigation to arrest Eve. They knew that a few months back, the public forum that Eve was using, got hacked and private messages and email addresses of its users were leaked. Reading through Eve's messages, FBI agents found

a bitcoin address that belonged to a cryptocurrency exchange. They then requested from the exchange to provide details of the account associated with that address. It turned out that the email address of the account matched Eve's email address on the forum and even better, Eve had completed the exchange's identity verification. The agents tracked down Eve and realized that she is trying to cash out her gains. Impersonating the buyer, the agents set up a meeting with Eve at a local library and when Eve brought out her laptop to transfer the funds, they connected a flash drive to her laptop that copied all of Eve's data files.

For each of the following methods of defence, explain how Eve could have used it to her advantage to avoid getting caught. Provide context that fits the narrative above.

(a) (2 marks) Preventing.

- Fly to country where FBI has no jurisdiction.

(b) (2 marks) Deflecting.

- Verify exchange's account with family member identity that has nothing to do with the attack.

(c) (2 marks) Deterring.

- Use software that encrypts hardware on connection of unauthorized devices to the laptop.

(d) (2 marks) Detecting.

- Use strong cryptography to encrypt and sign your messages, ask others to do the same to identify possible impersonators.

Programming Question [40 marks]

Problem Description

Background

You are tasked with testing the security of a custom-developed password-generation application for your organization. It is known that the application was *not written with best practices in mind*, and that in the past, this application had been exploited by some users with the malicious intent of *gaining root privileges*. There is some talk of the application having *four or more vulnerabilities*! As you are the only person in your organization to have a background in computer security, only you can *demonstrate how these vulnerabilities can be exploited* and *document/describe your exploits* so a fix can be made in the future.

Application Description

The application is a very simple program with the purpose of generating a random password and optionally writing it to `/etc/shadow`. The usage of `pwgen` is as follows:

```
Usage:  pwgen [options]
Randomly generates a password, optionally writes it to /etc/shadow
Options:
 -s, --salt <salt>  Specify custom salt, default is random
 -e, --seed [file]  Specify custom seed from file, default is from stdin
 -t, --type <type>  Specify different hashing method
 -w, --write        Write the password to /etc/shadow.
 -h, --help         Show this usage message
Hashing algorithm types:
0 - DES (default)
1 - MD5
2 - Blowfish
3 - SHA-256
4 - SHA-512
```

Note that the parameters for the options have to be specified like the following: "`--seed=temp.txt`", not "`--seed temp.txt`". If you use the short form of the option, it must be like "`-etemp.txt`" (no "=" between). There may be other ways to invoke the program that you are unaware of. Luckily, you have been provided with the source code of the application, `pwgen.c`, for further analysis. You will also be provided with some shellcode. The goal is to exploit four different vulnerabilities in the `pwgen.c` file to end up in a shell with root privileges.

The executable `pwgen` is *setuid root*, meaning that whenever `pwgen` is executed (by any user), it will have the full privileges of *root* instead of the privileges of the user that invokes it. Therefore, if an outside user can exploit a vulnerability in a setuid root program, he or she can cause the program to execute arbitrary code (such as shellcode) with the full permissions of the root user. If you are successful, running your exploit program will execute the setuid `pwgen`, which will perform some privileged operations, which will result in a shell with root privileges. (Note that the root password in the virtual environment is a long random string, so there is no use in attempting a brute-force attack on the password. You will need to exploit vulnerabilities in the application.)

There are at least five vulnerabilities in the program,

1. There is a buffer overflow vulnerability on line 264, where `strcpy()` is used to copy the `filename` argument for the `-e` option which is the name of the file containing the user provided seed for the randomness. By providing an appropriate string, the return address can be overwritten.
   This can be fixed by using the `strncpy()` function instead which takes the buffer size to copy over as a third parameter.

2. There is a format string vulnerability on line 269, where `fprintf()` is used to print a buffer related to the usage string. The argv[0] parameter can be set to an arbitrary value by the attacker by using `execve` to call pwgen. Reaching line 2689in the pwgen source file requires to make the `unlink` function fail, e.g., by creating a directory at `/tmp/pwgen_random`, which makes the function `check_perms()` return 1. Then, the attacker can exploit a format string vulnerability through format specifiers such as `%n`.
   This can be fixed by using the form `fprintf(stderr, "%s", buffer).`

3. There is a TOCTTOU vulnerability between the `check_perms()` and `fill_entropy()` functions as called by `parse_args()`. `check_perms()` ensures that `/tmp/pwgen` is a regular file owned by the user, and `fill_entropy()` writes the user-provided seed to this file. The user can start `pwgen` with the `--seed` option, at which point `pwgen` will perform the check and wait for user input. Before providing input, the user can create a symlink at `/tmp/pwgen` pointing to `/etc/passwd`. The user can then supply the input, whose contents will be written to `/etc/passwd`. This can be used to overwrite or clear the root password. The user can then log in as root using the `su` utility (this can be automated using `expect`).
   This can be fixed by using `mkstemp()` to create the temporary file. Another option would be to use an in-memory buffer rather than a temporary file, and use OpenSSL's `RAND_seed()` rather than `RAND_load_file()` in this case.

4. There is an incomplete mediation vulnerability in the `get_uid()` and `get_gid()` functions. These functions trust that the `HOME` environment variable contains the home directory

of the current user (and, as a result, assumes that this is unique in the first place). In fact, the user can set this value to whatever they like. If they set this value to `/root` or any directory which is not the home directory of any valid user, these functions will return the id 0. This can be used to trick `pwgen` into setting root's password rather than the current user's. The user can then log in as root using the `su` utility and the password printed out by `pwgen` (this can be automated using `expect`).
This can be fixed by getting the uid and gid with the `getuid()` and `getgid()` functions.

5. By changing the umask to be overly permissive in the UML, running `pwgen -w` overwrites `/etc/shadow` and updates its permissions, allowing it to become writable by any user, thus allowing a user to overwrite the password for root. This can be fixed by forcing the program to create `/etc/shadow` with sufficiently restrictive permissions (which may be facilitated via `open()`).