CS458 - A1

Name: Darren Poon
Userid: dyhpoon
UWID: 20311433

sploit3.c

The third vulnerability in the backup application is that it does not check whether the user has privilege to overwrite files the destination directory when restoring files. First, let looks at the following codes in main function.

```
original = argv[2];
// set up paths for copy operation
usrbkp = get_user_backup_path();
if (usrbkp == NULL) {
    fprintf(stderr, "Failed to get user backup directory\n");
    return 1;
}
remote = malloc(strlen(usrbkp)+1+strlen(original)+1);
sprintf(remote, "%s/%s", usrbkp,original);
if ((res = check_forbidden(remote))) {
    fprintf(stderr, "Not allowed to access path %s\n", remote);
    return 1;
}
switch (cmd) {
    case CMD_BACKUP:
        if (copyFile(original, remote) < 0) return 1;
        printf("Copied %s to backup\n", original);
        break;
    case CMD_RESTORE:
        if (copyFile(remote, original) < 0) return 1;
```

We can see that copyFile(remote,original) is called under the restore option. However, there is no checking in whether the user has privilege in the destination path (original) to restore files.

Since the application is setuid, attackers can choose to rewrite any files they want by using the backup application. To be more specific, an attacker may want to create a file named “passwd” with password/user he created, backup the file, change directory to /etc, and restore the “passwd” file. Now, the /etc/passwd is replaced by an attacker,

and he can gain root access by using the information he created in “passwd”.
To fix this vulnerability, the application must check whether the current user has privilege to access to that directory. If yes, then the user can overwrite/place the file to that directory. Otherwise, the application should not allow the user to perform any restore actions.

sploit4.c

The fourth vulnerability in the backup application is that after restoring the file, the ownership of that restored file is changed to the current user.

```
int chown_recursive(char * path){
    userid = get_userid();
    pid = fork();
    if (pid < 0) {
        fprintf(stderr, "Fork failed\n");
        return -1;
    }
    if (pid > 0) {
        waitpid(pid, &status, 0);
        if (WIFEXITED(status) == 0 || WEXITSTATUS(status) < 0)
            return -1;
        chown(path, getuid(), getgid());
    }else {
        execlp("/bin/chown", "/bin/chown", "-R", userid, dir1, NULL);
        // reached only in case of error
        return -1;
    }
```

chown_recursive(original) is called when restoring a file, this will change the ownership of the restored file, and this is dangerous. Suppose a file does not belong to the attacker, and the attacker can use this vulnerability to backup and restore the target file. Now, the ownership of the file is changed to the attacker and he can modify this file the way he wants. This could be bad when the attacker tries to change the ownership of sensitive files like /etc/passwd.
To fix this vulnerability, the backup application should not change the ownership of the file they restore, and should have keep the ownership of the file same as before. On the other hand, the backup application should not allow user to overwrite files that have higher privilege than the user.

<u>Freestyle Response Questions</u>

1.

a) From the perspective of Kate, "Kate's desktop wallpaper prints being freely shared" is an example of threat.

b) Prevent: State the copyright on each wallpaper prints.
Customers are warned if they share the prints, they will be sued by Kate.

Deter: Remove the sharing links/contents of Kate's wallpaper prints in the internet.
Before desktop wallpaper prints being shared over the internet, it must be uploaded to the internet, so Kate can ask the owner of the sites to remove those wallpaper prints that belong to Kate.

Deflect: Upload other artists' wallpaper prints legally to the internet.
So people will less likely to download Kate's wallpaper prints.

Detect: Hire someone to surf the internet and see if there is any print is being shared freely.
Kate will be noticed immediately when her prints is being shared freely.

Recover: Buy insurance.
Claim insurance when Kate lost money due to wallpaper prints being freely shared.

2.

a) Fault tolerance. Utilizing "sandboxing" to separate parts of execution that run untrusted input is fault tolerance. Even in the presence of fault, the system continues to operate with no significant loss of functionality or performance under sandboxing.

b) Fault removal. Patch Tuesday is fault removal because it is a security patch updates in order to remove or fix the known faults that cause errors to the system.

c) Fault forecasting. A peer code review before any code is accepted is fault forecasting. The code review let reviewers to predict likely faults that will occur if code is accepted, so that they can remove or fix the faults before any code is accepted.

d) Fault prevention. Enforcing rules on password to make them harder to crack is fault prevention. Since enforcing the rules on password prevent attackers from dictionary-based attack by reducing the chances of guessing the correct password.

e) Fault removal. Mail servers use intelligent spam filters to reduce the number of spam messages are fault removal. Since most of the spam messages are filtered, and it greatly reduces the number of spam messages we receive.

f) Fault prevention. The system requires students to authenticate before accessing information is fault prevention. Since authentication allow only you to access your information only, and prevent others to access your information. Thus, others cannot introduce any faults to your system.