Song
Spring 2015

CS 161
Computer Security

Discussion 6

February 24 & 25, 2015

Question 1 ***Security Principles*** **(10 min)**

We discussed the following security principles in lecture:

A. Security is economics
B. Least privilege
C. Failsafe defaults
D. Separation of responsibility
E. Defense in depth
F. Psychological acceptability
G. Human factors
H. Complete mediation
I. Know your threat model
J. Detect if you can't prevent
K. Don't rely on security through obscurity
L. Design security in from the start

Identify the principle(s) relevant to each of the following scenarios:

1. New cars often come with a valet key. This key is intended to be used by valet drivers who park your car for you. The key opens the door and turns on the ignition, but it does not open the trunk or the glove compartment.
2. Many home owners leave a house key under the floor mat in front of their door.
3. Convertible owners often leave the roof down when parking their car, allowing for easy access to whatever is inside.
4. Warranties on cell phones do not cover accidental damage, which includes liquid damage. Unfortunately for cell phone companies, many consumers who accidentally damage their phones with liquid will wait for it to dry, then take it in to the store, claiming that it doesn't work, but they don't know why. To combat this threat, many companies have begun to include on the product a small sticker that turns red (and stays red) when it gets wet.
5. Social security numbers, which we all know we are supposed to keep secret, are often easily obtainable or easily guessable.
6. The TSA hires a lot of employees and purchases a lot of equipment in order to stop people from bringing explosives onto airplanes.

Solution: (Note that there may be principles that apply other than those listed below.)

1. Principle of least privilege. They do not need to access your trunk or your glove box, so you don't give them the access to do so.
2. Unfortunately we often do rely on security through obscurity. The security of your home depends on the belief that most criminals don't know where your key is. With a modicum of effort, criminals could find your key and open the lock.
3. Security is economics. Even if they left the top up, it would be easy for a criminal to cut through it. If the criminals did that, it would cost the owner the cost of the items in the car and the cost of a new roof!
4. Detect if you can't prevent. People will try to scam cell phone manufacturers, and there is nothing the companies can do to stop this. But they can (and do) detect when people have voided their warranty via liquid damage.
5. Design security in from the start. Social security numbers were not designed to be authenticators, so security was not designed in from the start. The number is based on geographic region, a sequential group number, and a sequential serial number. They have since been repurposed as authenticators.
6. Security is economics. They spend a lot of money to protect airplanes, lives, and the warm/safe/fuzzy feeling that people want to have when they fly.

Question 2 *TOCTTOU* (10 min)

A *time-of-check-to-time-of-use* (TOCTTOU) vulnerability is a software bug[1] that occurs when an attacker can exploit the time window between the *check* of a condition and the *use* of the result of that check.

A classic scenario that frequently suffers from TOCTTOU bugs involves programs that have the *setuid* (**set u**ser **id** on execution)[2] access right set. setuid allows users to execute a program with the permissions of the program's owner. This behavior is useful when you want to let unprivileged users perform some privilege operation. Famous example of setuid programs include `ping` and `traceroute`. Both of these programs require root privilege for some of their network operations, but through the use of setuid can be executed by any user on the system.

Since setuid programs can run with elevated privilege it is the responsibility of the program to ensure that it isn't unwittingly giving additional privileges to the executing user. For example, a root owned setuid program has the ability to open any file on the system. This means that a setuid program must check the access rights of the user that invoked it before opening files. Such a check is shown in the example below.

This access right check is where TOCTTOU bugs can occur. If an attacker manages to alter the file after the permission check, yet before it is used, it is possible to replace the file with a symbolic link to a different (sensitive) file. The code snippet below illustrates this problem.

```
/* (1) Check file ownership */
if ( (0!=stat("file", &st)) ||
     (st.st_uid!=ALLOWED_USER) )
   exit(1);


/* (3) Write to /etc/passwd */
fd = open("file", O_WRONLY);
write(fd, buffer, sizeof(buffer));
```

TOCTTOU vulnerability.

```



/* (2) Change file after check */
symlink("file", "/etc/passwd");


```

Attacker changes `file` to `/etc/passwd`.

What mechanism is needed to fix the TOCTTOU vulnerability above? Rewrite the example to be secure.

HINT: `fstat()` works just like `stat()`, but takes a file descriptor rather than a string. A file descriptor is an abstract indicator for accessing a file, and is returned by the `open()` call above.

[1] A TOCTTOU bug is a special type of *race condition*.
[2] https://en.wikipedia.org/wiki/Setuid

Solution: The challenge with this TOCTTOU vulnerability is to ensure the file system cannot be changed between two system calls. Dean et al. showed that this is a hard problem, despite its conceptual simplicity. UNIX systems have adopted variants of common file system calls that operate on file handles rather than file names. These calls are prefixed with an `f`, such as `fstat`, `fchown`, etc. Because file handles are a private mapping to a file, they cannot be changed by another program and are not subject to race conditions with other applications. Using this mechanism, the example above can be rewritten as follows.

```
fd = open("file", O_WRONLY);
if ( (0!=fstat(fd, &st)) || (st.st_uid!=ALLOWED_USER) )
   exit(1);

write(fd, buffer, sizeof(buffer));
```

Question 3 ***TCB (Trusted Computing Base)*** **(10 min)**

In lecture and the reading, we discussed the importance of a TCB and the thought that goes into designing it. Answer these following questions about the TCB:

1. What is a TCB?

2. What can we do to reduce the size of the TCB?

3. What components are included in the (physical analog of) TCB for the following security goals:

   (a) Preventing break-ins to your apartment

   (b) Locking up your bike

   (c) Preventing people from riding BART for free

   (d) Making sure no explosives are present on an airplane

Solution:

1. It is the set of hardware and software on which we depend for correct enforcement of policy. If part of the TCB is incorrect, the system's security properties can no longer be guaranteed to be true. Anything outside the TCB isn't relied upon in any way.
2. Privilege separation can help reduce the size of the TCB. You will end up with more components, but not all of them can violate your security goals if they break.
3. (This list is not necessarily complete)

   (a) the lock, the door, the walls, the windows, the roof, the floor, you, anyone who has a key

   (b) the bike frame, the bike lock, the post you lock it to, the ground

   (c) the ticket machines, the tickets, the turnstiles, the entrances, the employees

   (d) the TSA employees, the security gates, the "one-way" exit gates, the fences surrounding the runway area

A final note: do not hesitate to ask for help! Our office hours exist to help you. Please visit us if you have any questions or doubts about the material.