Popa
Spring 2018

CS 161
Computer Security

Discussion 2

# Week of January 29, 2018

**Question 1** ***Software Vulnerabilities*** **(20 min)**

For the following code, assume an attacker can control the value of `basket` passed into `eval_basket`. The value of `n` is constrained to correctly reflect the number of elements in `basket`.

The code includes several security vulnerabilities. **Circle *three* such vulnerabilities** in the code and **briefly explain** each of the three.

```
1  struct food {
2          char name[1024];
3          int calories;
4  };
5
6  /* Evaluate a shopping basket with at most 32 food items.
7     Returns the number of low-calorie items, or -1 on a problem. */
8  int eval_basket(struct food basket[], size_t n) {
9          struct food good[32];
10         char bad[1024], cmd[1024];
11         int i, total = 0, ngood = 0, size_bad = 0;
12
13         if (n > 32) return -1;
14
15         for (i = 0; i <= n; ++i) {
16                 if (basket[i].calories < 100)
17                         good[ngood++] = basket[i];
18                 else if (basket[i].calories > 500) {
19                         size_t len = strlen(basket[i].name);
20                         snprintf(bad + size_bad, len, "%s ", basket[i].name);
21                         size_bad += len;
22                 }
23
24                 total += basket[i].calories;
25         }
26
27         if (total > 2500) {
28                 const char *fmt = "health-factor --calories %d --bad-items %s";
29                 fprintf(stderr, "lots of calories!");
30                 snprintf(cmd, sizeof cmd, fmt, total, bad);
31                 system(cmd);
32         }
33
34         return ngood;
35 }
```

*Reminders:*

- **snprintf(buf, len, fmt, ...)** works like **printf**, but instead writes to **buf**, and won't write more than **len - 1** characters. It terminates the characters written with a '\0'.
- **system** runs the shell command given by its first argument.

**Solution:** There are significant vulnerabilities at lines **15/17**, **20**, and **31**.

Line **15** has a fencepost error: the conditional test should be $i < n$ rather than $i <= n$. The test at line **13** assures that **n** doesn't exceed 32, but if it's equal to 32, and if all of the items in **basket** are "good", then the assignment at line **17** will write past the end of **good**, representing a buffer overflow vulnerability.

At line **20**, there's an error in that the length passed to **snprintf** is *supposed* to be available space in the buffer (which would be **sizeof bad - size_bad**), but instead it's the length of the string being copied (along with a blank) into the buffer. Therefore by supplying large names for items in **basket**, the attacker can write past the end of **bad** at this point, again representing a buffer overflow vulnerability.

At line **31**, a shell command is run based on the contents of **cmd**, which in turn includes values from **bad**, which in turn is derived from input provided by the attacker. That input could include shell command characters such as pipes ('|') or command separators (';'), facilitating *command injection*.

Some more minor issues concern the **name** strings in **basket** possibly not being correctly terminated with $'\backslash 0's$, which could lead to reading of memory outside of **basket** at line **19** or line **20**.

Note that there are no issues with format string vulnerabilities at any of lines **20**, **29**, or **30**. For each of those, the format itself does not include any elements under the control of the attacker.

**Question 2** ***Security Principles*** **(15 min)**

We discussed the following security principles in lecture (*or in the lecture notes*, which you are responsible for reading):

A. Security is economics
B. Least privilege
C. Know your threat model
D. Defense in depth
E. Consider human factors
F. Design in security from the start
G. Ensure complete mediation
H. Division of trust
I. Consider Shannon's Maxim

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. It is not worth it to use a $400 bike lock to protect a $100 bike.
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 were not originally designed as a secret identifier. Nowadays, they are often easily obtainable or guessable.
6. Even if you use a password on your laptop lockscreen, there is software which lets a skilled attacker with specialized equipment to bypass it.
7. Shamir's secret sharing scheme allows us to split a "secret" between multiple people, so that all of them have to collaborate in order to recover the secret.
8. DRM encryption is often effective, until someone can reverse-engineer the decryption algorithm.
9. Banks often make you answer your security questions over the phone. If you use a random password as the answer to a security question, an attacker can often convince the phone representative by claiming "I just put in some nonsense for that question".

**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. Shannon's Maxim. 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. It is more expensive to buy $400 bike lock than to simply buy a new bike to replace it.
4. There are probably two most relevant factors. "Consider human factors": people will always try to lie and you must account for that when creating a system. More importantly, "Design in security from the start": it's prudent to try to add ways to detect something when creating the phone rather than trying to determine water damage after-the-fact.
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. Know your threat model: most petty thieves do not have access to this software. (The software referenced is pcileech. The corresponding hardware is on my wishlist. -Keyhan Vakil)
7. Division of trust: require everyone to come together to produce the secret, preventing one person from using the secret alone.
8. Shannon's Maxim. You must assume the attacker knows the system, so DRM encryption is not effective.
9. Consider human factors. The phone rep is inclined to believe the attacker is not malicious (social engineering).

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

In lecture, 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