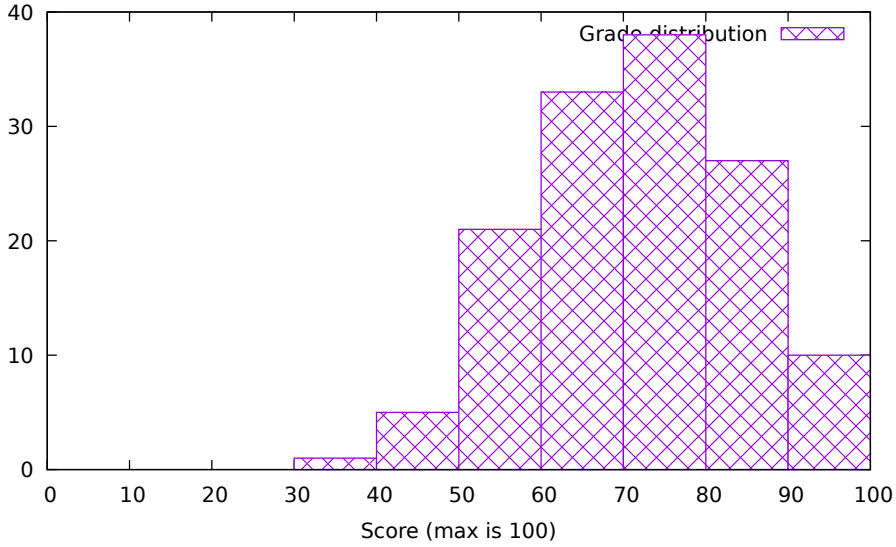
Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.858 Spring 2018

Quiz I Solutions

Mean 72.0 Standard deviation 12.5



I Paper reading questions

1. [5 points]:

Which of the following attack scenarios are addressed by some aspect of Google's security architecture as described in their whitepaper?

A. True / False Google engineers inserting backdoors into software. **Answer:** True; code reviews and approvals.

B. True / False A CPU manufacturer such as Intel inserting a backdoor into their CPU. **Answer:** False; no plan to deal with CPU bugs is described in the whitepaper.

C. True / False A user of GMail inadvertently running malware on their computer that steals their email (and sends it over TCP to an attacker's server). **Answer:** False; no plan to deal with malware on end-user computers is described in the whitepaper.

D. True / False A user of GMail has a sneaky roommate who watches the user type in his password. **Answer:** True; two-factor authentication helps.

E. True / False An adversary gets a job as a data center operator and steals a server's hard drives in order to obtain user data. **Answer:** True; metal detectors in a data center.

2. [4 points]: Which of the following statements are true about U2F (as described in the assigned reading "Universal 2nd Factor (U2F) Overview")?

(Circle True or False for each choice.)

A. True / False U2F is a stronger second factor than sending an SMS code to a user's smartphone.

Answer: True. SMS codes are vulnerable to a MITM attack.

B. True / False An attacker that knows a user's password can easily guess the U2F key to access the user's account.

Answer: False. The key on their dongle is a strong second factor that attackers cannot guess.

C. True / False A U2F USB dongle prevents malware on the user's computer from stealing the user's second factor to authenticate as that user even when the user's computer is turned off.

Answer: True. The key is never exposed directly to software on the user's computer.

D. True / False A server using U2F can reliably determine that the user who is attempting to login is indeed behind the computer that sent the login request.

Answer: False. The adversary may have stolen the USB dongle.

3. [4 points]: Which of the following recommendations should one follow according to Paul Youn? **(Circle True or False for each choice.)**

A. True / False Sprinkle two-factor authentication everywhere. **Answer:** True.

B. True / False When designing a defense for a system like AirBnB, one should focus on particular attacks. **Answer:** False.

C. True / False One should think of computer security as an engineering discipline. **Answer:** True.

D. True / False One should learn how to penetrate computer systems to learn how to think like an adversary. **Answer:** True.

II Buffer overflows

Consider the following code snippet from lab 1's `http.c`:

```
1  void http_serve(int fd, const char *name)
2  {
3      void (*handler)(int, const char *) = http_serve_none;
4      char pn[1024];
5      struct stat st;
6
7      getcwd(pn, sizeof(pn));
8      setenv("DOCUMENT_ROOT", pn, 1);
9      strcat(pn, name);
10     split_path(pn);
11
12     if (!stat(pn, &st))
13     {
14         /* executable bits -- run as CGI script */
15         if (valid_cgi_script(&st))
16             handler = http_serve_executable;
17         else if (S_ISDIR(st.st_mode))
18             handler = http_serve_directory;
19         else
20             handler = http_serve_file;
21     }
22
23     handler(fd, pn);
24 }
```

4. [10 points]: Point out two ways to exploit this code fragment. One of them should require running code on the stack, and the other should work even if the operating system marked the stack as non-executable (NX). For each exploit, describe it briefly, referring to specific lines of code above as needed.

Answer: Vulnerability: strcat overflows pn buffer. Exploit 1: overwrite handler with unlink. Exploit 2: overwrite return address and put shell code inside the pn buffer.

The stack layout of `http_serve` is as follows:

```
0xbffff59c: return address
0xbffff598: saved ebp
0xbffff58c: variable handler
0xbffff18c: variable pn
0xbffff134: variable st
```

5. [10 points]: For `zookd-nxstack`, what input should the adversary provide to `http_serve` to launch a return-to-libc attack to delete `/home/httpd/grades.txt`? (Assume the address of the `unlink` libc function is 0xBCBCBCBC.) Write down the sequence of bytes that the adversary should provide in the `name` argument.

Answer: Several components. First, 1024 bytes of "A", minus the length of cwd (i.e., /home/httpd). Then, 0xBCBCBCBC. Then, 16 "A"s. Then we're at the arguments to http_serve(). The first argument comes first, so put "0xBF FF F5 A4". Then "/home/httpd/grades.txt" with the null terminating byte.

III Baggy bounds checking

Assume you compile the zookws sources with Baggy Bounds as described in the paper "Baggy Bounds Checking: An Efficient and Backwards-Compatible Defense against Out-of-Bounds Errors" by Akritidis et al. Refer to the code from the previous question about Buffer Overflows. Assume that both zookws and all system libraries are compiled with Baggy Bounds checking.

6. [6 points]: Will baggy bounds prevent `strcat` from writing past the end of `pn`? If so, briefly explain why? If not, briefly explain why not?

Answer: Yes, it will prevent. pn is exactly 1024 bytes.

7. [6 points]: If the bound on `pn` is 1028 instead of 1024 will baggy bounds prevent `strcat` from writing past the end of `pn`? If so, briefly explain why? If not, briefly explain why not?

Answer: No, rounds up to 2048 bytes, can write past the end. However, this does not affect any other variables.

IV Capsicum

Ben Bitdiddle observes that FreeBSD already uses integers to represent processes (i.e., FreeBSD process IDs are 32-bit integer values), and decides to simplify Capsicum's design by keeping the original FreeBSD process IDs instead of Capsicum's process descriptors.

8. [12 points]: What problem might arise in Ben's design that uses PIDs instead of Capsicum's design that uses process descriptors?

Answer: A process running in capability mode might be able to kill other processes running under the same user ID, since the process can name all PIDs in the global namespace, rather than just the process descriptors it has been granted. In Capsicum's design, a sandboxed process would not be able to do this; it would be able to kill only those processes for which it has process descriptors.

V Native Client

Consider the validator for Native Client as shown in Figure 3 of the NaCL paper ("Native Client: A Sandbox for Portable, Untrusted x86 Native Code").

9. [14 points]: Note that if `inst_is_indirect_jump_or_call(IP)` is true, then IP is not added to `JumpTargets`. Ben's wonders why not adding is important. He modifies the code to add IP to `JumpTargets` in both branches of that `if` statement. Show a fragment of code for a module that an adversary could write, that would pass Ben's modified validator, and that would allow to the code to jump to an arbitrary (possibly not-32-byte-aligned) address `a`. (Briefly explain your answer.)

Answer:

```
    MOV a, %eax
    JMP b
    AND $0xFFFFFFE0, %eax
b:  JMP *%eax
```

VI iOS

10. [12 points]: Alyssa P. Hacker develops iOS applications, and wants to be able to upgrade and downgrade the iOS software on her iPhone, to test her applications with different iOS versions. Assume that she can read and write her iPhone's OS image from her computer via USB. How can she defeat Apple's downgrade protection?

Answer: Save the OS image from each version of iOS that Alyssa wants to return to. It is signed together with her phone's ECID, so she can write it back to the phone and it will boot correctly.

VII Android

Consider the Android system as described in the paper by Enck et al.

Android applications are distributed as `.apk` files, which are just ZIP files. When an application is installed, the `.apk` file is saved in the phone's file system. The installer temporarily extracts the contents of the ZIP file (the Java code, the manifest, and the signature), checks the signature on the Java code and manifest, and asks the user to approve the permissions from the manifest. At boot time, the Android system loads each `.apk` file by extracting the manifest and Java code.

Ben Bitdiddle discovers that the Android installer and the boot-time Android system differ in how they interpret ZIP files. If the ZIP file has two files with the same file name, the Android installer takes the first one, and the boot-time code takes the last one.

11. [13 points]: What aspects of Android's security model can be broken using this bug? Describe a specific attack that Ben can perform.

Answer: The signature on application code and permissions in the manifest is not properly checked. Ben could take an existing app, and modify it to have different code by adding Java code files with the same name but different code contents. Ben could distribute this as an update to the existing app, and it would appear legitimate to the installer.

VIII 6.858

We'd like to hear your opinions about 6.858. Any answer, except no answer, will receive full credit.

12. [2 points]: Are there things you'd like to see improved in the second half of the semester?

Answer:

Lecture content: 6x Dive deeper in lecture (or psets), at the level of quiz questions. 5x More guest lectures. 5x Record lecture video. 4x Lectures are too similar to papers. 3x Chalkboard diagrams in lecture notes. 3x More examples in lecture; maybe exercises? 3x Better lecture notes. 2x Lectures jump around too much. 1-hour lectures. Don't start before 1:05! More connection between lectures. Pick a lecture room closer to main campus.

Paper assignments: 10x Explain background material / acronyms for papers. 7x Post staff answers to reading Qs / answer all student questions. 2x Explain what part of the paper to focus on. 2x Don't require questions for each paper. More guided reading questions.

Papers / topics: 8x More attacks, how to develop attacks. 4x More recent, practical papers / systems. More architectural papers like OKWS. Less privilege separation. More specific, focused state-of-the-art papers. More discussion of threat models. Crypto, signatures.

Quiz: 2x More quiz review materials. Weekly/biweekly review sessions throughout semester (recitations?).

Labs: 9x Faster lab grading. 3x More connection between labs and lecture. More guidelines for lab 1. Extra lab reading material. More independent, deeper labs; less hand-holding. More test cases for labs. Make lab 2 less repetitive. Hints on how to get started with labs. Assign more coding.

TA help: 7x More office hours. 3x Faster Piazza responses.

13. [2 points]: Is there one paper out of the ones we have covered so far in 6.858 that you think we should definitely remove next year? If not, feel free to say that.

Answer:

Disliked papers: 22x Haven. 14x Capsicum. 12x Android. 10x SGX. 7x NaCl. 5x OKWS (old!). 5x U2F. 4x Mandatory pw changes. 4x Google. 3x Baggy. 3x iOS. 1x Spinnaker (paper from 6.824).

(Volunteered list of) liked papers: 4x NaCl. 4x EXE. 1x Capsicum. 1x Baggy. 1x Android. 1x iOS. 1x Google.

Other: 4x Want a more modern version of OKWS. 1x Assign a paper for Paul Youn's lecture, for one of the attacks he mentions.

End of Quiz