CS161 Homework 1

Vedaank Tiwari

TOTAL POINTS

75 / 120

QUESTION 1

1 Policy 10 / 10

✓ **+ 10 pts** **Correct**

+ 0 pts Incorrect. Please read the course policy again.

QUESTION 2

2 Collaboration 10 / 10

✓ **+ 10 pts** **Correct**

+ 0 pts Incorrect. Please read the course policies again.

QUESTION 3

3 Security Principles 20 / 20

✓ **+ 5 pts** **Login creds on a HTML comment: Don't rely on security by obscurity**

✓ **+ 5 pts** **Giving a design contractor access to database: Least Privilege**

✓ **+ 5 pts** **Alerting on ever user action: Consider human factors**

✓ **+ 5 pts** **Unable to change aspects of the website: Design security in from the start.**

+ 0 pts Blank/Incorrect

QUESTION 4

Vulnerable Code 40 pts

4.1 a 10 / 10

✓ **+ 5 pts** **Line 5**

✓ **+ 5 pts** **Buffer overflow vulnerability, unchecked boundaries, overwrite return address, stack overflow, etc.**

+ 0 pts Other

4.2 b 10 / 10

✓ **+ 5 pts** **Reasonable Explanation (Overflow the buffer with arbitrary input/malicious code. OR Over-write the return address. OR New return address points to attacker's code.)**

✓ **+ 5 pts** **Return address location is 0xBFFFFB4C (saved ebp + 4).**

- 5 pts More than 5 sentences submitted.

+ 0 pts other

4.3 c 10 / 10

✓ **+ 10 pts** **Check the length before filling buffer OR Use some memory safe input function.**

+ 0 pts Other

4.4 d 0 / 10

+ 10 pts Yes. Canary can detect stack change.

+ 10 pts No. Canary value can be guessed via format string vulnerabilities

✓ **+ 0 pts** **Other**

QUESTION 5

5 Reasoning About Memory Safety 5 / 40

✓ **+ 5 pts** **Part a: hex != NULL**

+ 5 pts Part a: size(hex) = 1 + LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Valid Explanation for preconditions: (dectohex accesses the value store at hex w/o checking if hex is a valid pointer OR Need enough space for hex representation + null pointer)

+ 5 pts Part b: 0 <= j < LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Part c: 0 <= j < LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Part c: 0 < k <= LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 10 pts Valid Explanation for loop invariants: (loop 1 executes no more than LEN times and j is incremented after (b) OR k is only ever incremented

and loop exits after LEN times)

- **1 pts** off by on errors

+ **0 pts** other

QUESTION 6

6 Feedback 0 / 0

✓ **+ 0 pts Correct**

CS 161 Efficient Algorithms and Intractable Problems
Fall 2018 Nick Weaver HW 1

Due on 2018-09-19, at 11:59 pm

1 How many project slip days do you get?

Solution 0 Slip Days

1 Policy 10 / 10

✓ **+ 10 pts Correct**

+ 0 pts Incorrect. Please read the course policy again.

2 Collaboration

Solution No collaboration of this manner is allowed

2 Collaboration 10 / 10

✓ **+ 10 pts Correct**

+ 0 pts Incorrect. Please read the course policies again.

3 Security Principles

Solution Getting on the cryptocurrency hype, one day Bob decides to set up his own exchange. He sets up all the infrastructure, but worries about forgetting the password, so Bob hides his login credentials in an HTML comment on the login page.
DON'T RELY ON SECURITY BY OBSCURITY, (password hidden in a comment) COMPLETE MEDIATION (there is a back door left by password in the comment), DEFENSE IN DEPTH (No way to recover is password is lost, only 1 point of security), FAIL-SAFE DEFAULTS (No, no default mode to deal with attackers), DETECT IF YOU CANT PROTECT (No detection of tampering), PROACTIVELY STUDY ATTACKS (No studying of attacks to find out how to deal with them), DESIGN SECURITY IN FROM THE START (No viable security designed in whatsoever), SEPARATION OF RESPONSIBILITIES (Once root access is broken, product is 100 percent vulnerable

Eventually, Bob manages to gather a large user-base and realizes his site looks like a back-end developer trying to learn CSS, so he contracts out front-end work to Mallorys Do-No-Evil design firm (for an incredible price too!). He gives them an account with access to his front-end and back-end codebase, and databases of user information as well.
LEAST PRIVILEGE (Bob gives way too much information in is account for the design upgrade),

Finally, his site looks amazing but now Bob is worried about his users security, so he sets up emails to be sent on every user event (logins, web pages visited, transactions, messages, password changes, etc). Bob now rests, assured that his web app is one of the most secure to ever exist.
CONSIDER HUMAN Factors (bombarding with notifications essentially nullifies the notifications as users won't care anymore, SECURITY IN ECONOMICS (Less is more in this case, fewer notifications will increase the importance of each notification, improving security in that way), PROACTIVELY STUDY ATTACKS (annay attack is not studied to fix vulnerabilities in the future), DETECT IF YOU CANT PROTECT (since Bob technically detects every user event, there is not real detection of security leaks, this just dumps all user activity to the user and because there is so much information, it will likely be ignored)

Unfortunately for him, one day he wakes up to his website being featured on a well known news site after a data leak. Pressured by an internet mob, he hires a contractor to find all the issues with his site. However, fixing the website ended up being a different story, as much of the code was written (uncommented) in a late-night coffee-fuled frenzy, and Bob finds that he cant change any aspect of the website without breaking it in its entirety. In a panic, Bob announced the closure of his site and goes into hiding.

DESIGN SECURITY IN FROM THE START (Security needs to be implemented from the beginning to ensure defense in depth, a wrapper container security may be secure for certain tasks, but once broken, is useless) , DETECT IF YOU CANT PROTECT (Bob can at least detect vulnerabilities, he doesn't even do this), PROACTIVELY STUDY ATTACKS (again no studying of attacks here so improvement in the future is more difficult)

3 Security Principles 20 / 20

- ✓ + 5 pts Login creds on a HTML comment: Don't rely on security by obscurity
- ✓ + 5 pts Giving a design contractor access to database: Least Privilege
- ✓ + 5 pts Alerting on ever user action: Consider human factors
- ✓ + 5 pts Unable to change aspects of the website: Design security in from the start.
- + 0 pts Blank/Incorrect

4 Vulnerable Code

(a) What is the line number that has a memory vulnerability and what is this vulnerability called?

Solution There is a buffer overflow possible in line 5 in the scanf function, as it takes in a normal string and loads it into buffer. This can be overflowed to point to some shellcode, and gain user access.

(b) How an attacker would take advantage of the vulnerability.

Solution

An attacker can overflow the buffer with approx 20 bytes of random code to fill the 16 byte buffer and + 4 because EIP is 4 bytes after EBP. Then make a new EIP to point to some shellcode 4 bytes after this EIP to gain user access. EIP is at 0xBFFFFB4C (EBP + 4)

(c) What would you change to fix the problem?

Solution

Using fget() rather than scanf() will be more secure, esp the more complicated the formatting is, the harder it will be to overflow the buffer in this regard. Or we could check the length of the input before filling the buffer.

(d) Would stack canaries help?

Solution

No, because of the scanf() vulnerability, we can essentially feed the code whatever we want, extract the canary, and brute force it in this manner. Canaries don't really help against format string vulnerabilities like this.

4.1 a **10 / 10**

✓ **+ 5 pts** **Line 5**

✓ **+ 5 pts** **Buffer overflow vulnerability, unchecked boundaries, overwrite return address, stack overflow, etc.**

+ 0 pts Other

4.2 b **10 / 10**

✓ **+ 5 pts** **Reasonable Explanation (Overflow the buffer with arbitrary input/malicious code. OR Over-write the return address. OR New return address points to attacker's code.)**

✓ **+ 5 pts** **Return address location is 0xBFFFFB4C (saved ebp + 4).**

- 5 pts More than 5 sentences submitted.

+ 0 pts other

4.3 C **10 / 10**

✓ **+ 10 pts** **Check the length before filling buffer OR Use some memory safe input function.**

+ 0 pts Other

4.4 d **0 / 10**

+ **10 pts** Yes. Canary can detect stack change.

+ **10 pts** No. Canary value can be guessed via format string vulnerabilities

✓ **+ 0 pts** **Other**

5 Reasoning about memory safety

(a) Preconditions

Solution decimal is not null, hex is not null, also buffer needs to be large enough for the loop to function. Also decimal must be less that $16^9 and temp can only hold 9 digits in hex$.

(b) Invariant B

Solution

digit is not null and temp is size j

(c) Invariant C

Solution

Hex is size j + 1 (for null byte at end) and tmp is size j.

5 Reasoning About Memory Safety **5 / 40**

✓ **+ 5 pts Part a: hex != NULL**

+ 5 pts Part a: size(hex) = 1 + LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Valid Explanation for preconditions: (dectohex accesses the value store at hex w/o checking if hex is a valid pointer OR Need enough space for hex representation + null pointer)

+ 5 pts Part b: 0 <= j < LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Part c: 0 <= j < LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 5 pts Part c: 0 < k <= LEN, where LEN is the true length of the integer in hex, i.e. ceil(log_16(decimal))

+ 10 pts Valid Explanation for loop invariants: (loop 1 executes no more than LEN times and j is incremented after (b) OR k is only ever incremented and loop exits after LEN times)

- 1 pts off by on errors

+ 0 pts other

6 Feedback 0 / 0

✓ + 0 pts Correct