Justin Ong 25 April 2022

CX4067 2020-2021 Semester 2

Question 1

a) Object reuse – Process gets access to resources used by previous process
Storage residues – Aforementioned resources leaking data from a previous process to the new process

Threats: leak sensitive information eg encryption keys/data

Countermeasures:
1) OS level: prevent a process from reading memory it did not write to
2) Use calloc instead of malloc (which initializes the memory to zeros)

b) Heartbleed: User-provided length allowing out-of-bounds read. Sending a message and a malformed length (length field larger than length of actual message sent) results in server returning more information than was sent, potentially leaking sensitive information like TLS keys/application information like session cookies

Lec5, Slide 58 (Rev 2/15/2018)

c) Uninitialized memory has been used as a source of randomness.

Lec5, Slide 66 (Rev 2/15/2018)

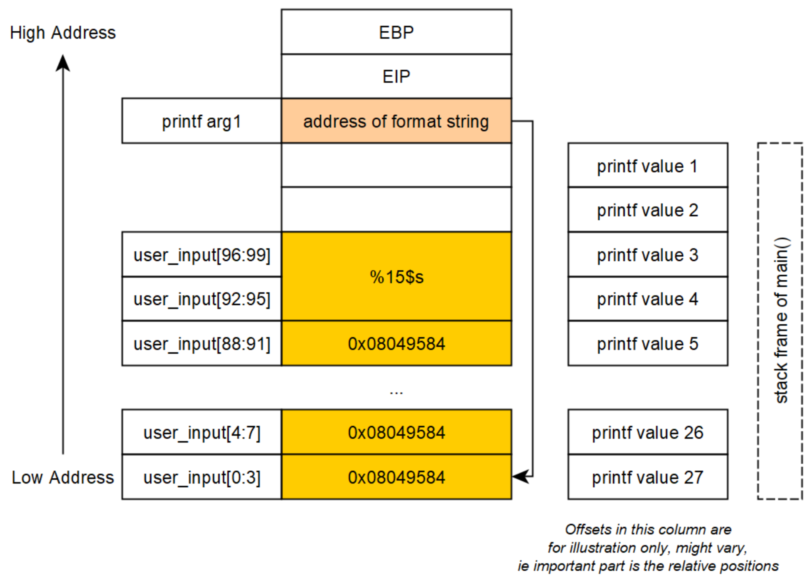
Debatable whether this is ever a good idea today – why not use an actual cryptographically secure PRNG like /dev/random?

Question 2

a)
1) Buffer overflow on line 7: unbounded read into buffer of fixed size 100
2) Format string bug on line 8: printf with token string of user input

b) Leak string at 0x08049584. ASLR must be disabled.

c) user_input will be indexed from the end, so printf offset needs to be increased accordingly.
Payload: "\x84\x95\x04\x08" * 23 + "%15$s"



Question 3

a) Same Origin Policy: Page from a given origin (website) cannot access pages from another origin

   Given a page https://a.com:

   1) http://a.com – Different protocol
   2) https://a.com:8080 – Different port (*if unstated, URL defaults to port 80*)

b) Reflected XSS: allows malicious code to run under origin of website.
   Example: See Lec9, Slide 42 (Rev 25/3/21)

c) CSRF attacks are launched from a different origin (example: malicious page on attacker.com makes requests to target.com [*hence the name cross-site*]), while XSS attacks are launched from the same origin (example: page on target.com runs malicious JavaScript injected by attacker through reflected XSS in part b)

   Send a nonce with every request. See Lec9, Slide 63 (Rev 25/3/21)

Question 4

a)
   1) Use parameterized interfaces
   2) Sanitization

b) Taint analysis can be used to detect the passing of unsanitized data from user input (eg query parameter/form data) into a sensitive sink (eg SQL query).

   Three factors affecting precision of taint analysis:

   1) Flow sensitivity
   2) Context sensitivity
   3) Alias analysis

   See Lec10, Slide 37 (Rev 25/3/21)

c) Meta-characters are characters with special meanings eg string terminators (Lec7, Slide 17 (Rev 17/2/22))

   ‘ (single quote) is a string terminator for both XPath and SQL

   Attack: SQL injection
   SELECT * FROM users WHERE username=’<u>’ OR 1=1;#</u>’ AND password=’<u>anything</u>’;
   (underlined parts are attacker input)

Question 5

Discussed in tutorial 10