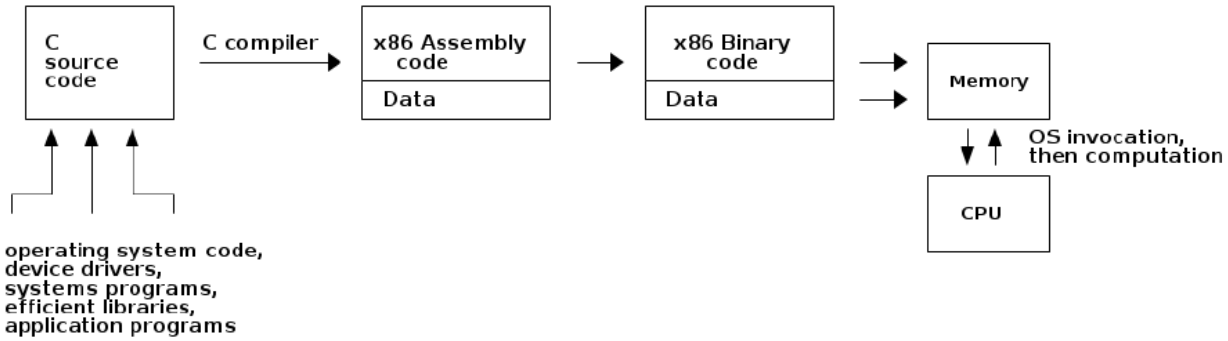
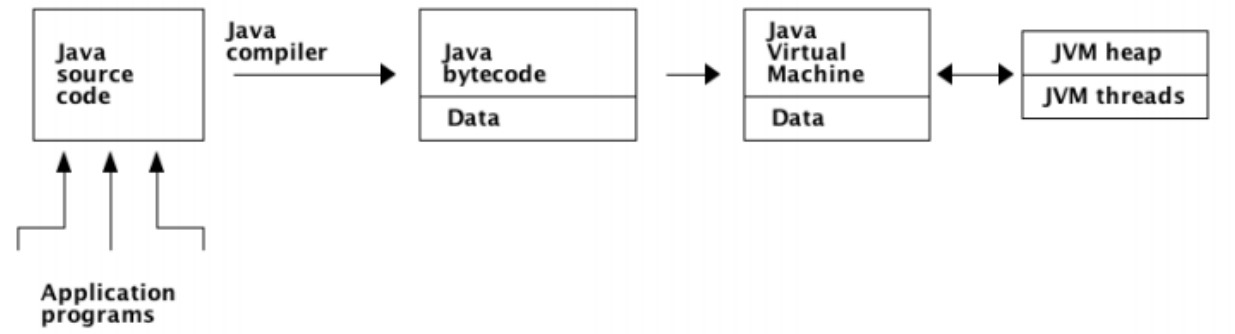
Exam date is the **19th at 09:30 at Appleton** unless anything changes.

I know there is a cheat sheet also made with most of this covered but thought a list all of the specific review questions from lectures with possible answers, would also be good. Please correct, add or alter if things are missing or incorrect.

Program execution





Explain the points of trust that exist when a Linux user runs a program by executing a binary file. (Pics show bits of code where trust differs)

Is this possibly the libraries being used, or that there is no effect on certain directories and files (privacy reasons)?

Buffer overflows

How do they arise?

Buffer overflows occur when a program attempts to put more data in a buffer than it can hold or into a memory area past the buffer. Writing outside the bounds of an allocated memory block can corrupt data, crash the program or cause execution of malicious code.

In what sense are some languages considered immune from buffer overflow attacks?

Languages like Java & Python are considered in most cases immune to buffer overflows. In the case of Java, a **type safe memory model is used** meaning that data falling of the end of an **object** and spilling elsewhere is not possible,

Shellcode must be free of 0x00 (NULL) characters, why?

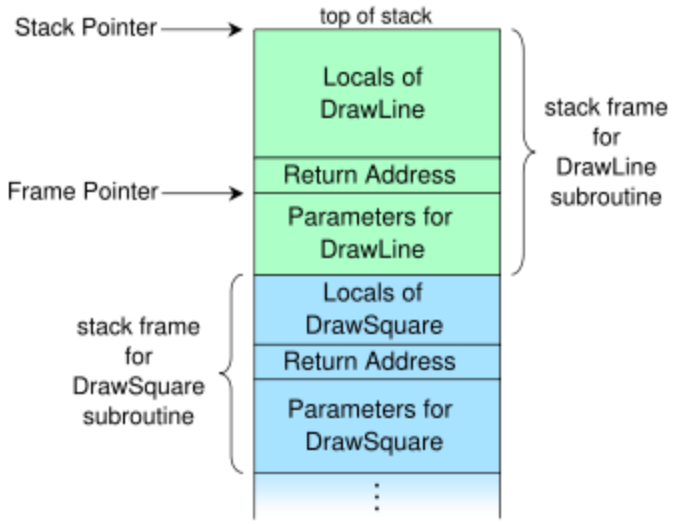
Null (‘\0’) is a string delimiter which instructs all C string functions (and other similar implementations) to, once found, stop processing the string (a null-terminated string).

Runtime stack basics

Describe how function parameters and local variables are allocated on the runtime stack.

The runtime stack(program or function stack) holds stack frames for each function that is invoked. the exact method of implementation for this mechanism differs by OS, language, compiler(both type and options) and the CPU.
Parameters may be passed to the function body on the stack or in registers. The mechanism used for this is called the calling convention.
Local variables are allocated space on the stack.

A frame pointer can be used to locate arguments and local variables.



Example diagram from http://en.wikipedia.org/wiki/Call_stack

Stack overflows

Explain how uncontrolled memory writing can let an attacker corrupt the value of local variables.

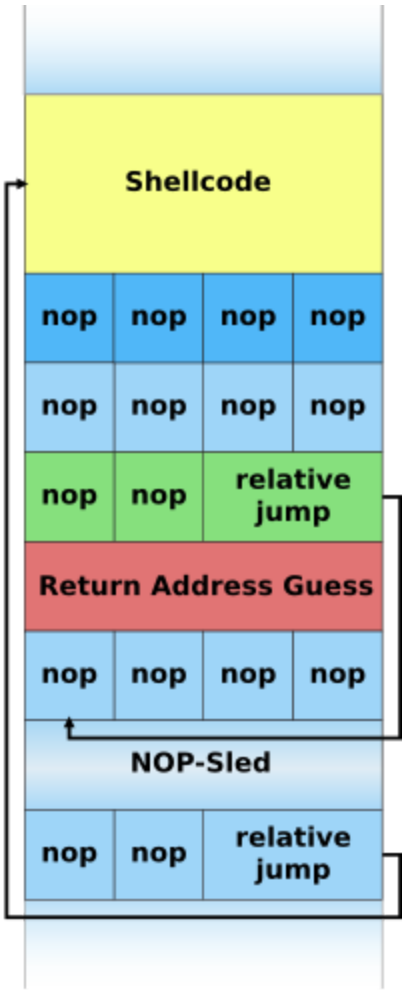
Local variables are put close together on the stack. If a stray write goes beyond the size of one variable it could corrupt another.
tricky in practice as variable location may not be known.
Diagram in lecture 4 slides 12/13.

Explain how an attacker can exploit a stack overflow to execute arbitrary code.

If the attacker can overwrite the return address then they may set it to point at some known piece of code in the application or a shared library or at their own code located somewhere in memory.
Three steps given in lecture:

1. store executable code somewhere in memory. It must reside within the address space of the exploited process (i.e. the buffer, as an argument to the process, or in the attacker's shell)
2. use stack overflow to direct execution there
3. code does something useful to attacker

Draw an example stack during a stack overflow attack with a NOP sled. Showing some possible addresses for the shellcode location and return address.



Example Diagram from:
http://en.wikipedia.org/wiki/Buffer_overflow#NOP_sled_technique

Heap overflows

Explain the API functions used to interface to heap allocation in C.

malloc(size): Tries to allocate “size” bytes, a chunk of memory reserved for the heap. The contents of which is not changed. returns pointer to the block if it succeeds.

calloc(size): Similar to malloc but initialises content of memory in block to zero.

free(ptr): frees up the allocated space at ptr. Has no return value.

Give two examples of risky behaviour.

Using malloc can have memory contain unpredictable or garbage values.
Dereferencing a null pointer that has been freed could lead to “undefined” behaviour.
Not explicitly deallocating (freeing) a block of memory can cause memory leak issues, since the memory locations are allocated but not being used.

Show how overflowing one heap-allocated variable can corrupt a second.
If two ranges of memory are allocated directly after each other in code and the memory is not randomised with ASLR then they should also be near to each other on the heap. Overflowing one variable will therefore eventually overflow the other one.
Sketch how a heap overflow attack can exploit memory allocation routines to write to locations controlled by an attacker.

Other vulnerabilities

Explain type confusion errors, giving an example.

When a program allocates or initialises a pointer,object or variable using one type. Then later accesses this using a type that is incompatible.
Eg?

Overflow protection mechanisms

Describe how StackGuard's "canaries" work to stop buffer overflows.

Canaries are used to detect a stack buffer overflow before execution of malicious code can occur. They work by placing an integer with a random value chosen at the program start, just in front of the stack return pointer. This means that in order to overwrite the return pointer the canary must also be overwritten. This can be checked before using the return pointer and detect any change to it.

What kind of attacks are they not effective against?

The use of canaries does not protect against attack overwriting the values of local variables and they also cannot prevent DOS attacks, as by design, the program execution will stop when detecting a modified canary.

Describe how hardware-assisted memory protection can prevent the worst kinds of overflow attacks. In what cases may it be difficult to use?

Segmentation: separate memory into distinct, variable-sized pieces.
Hardware prevents overflowing from one segment into another one. Difficult to use?
Also, writable-/executable flags are enforced by hardware and prevent execution of injected code.

Explain the strategy of program diversification and how it is achieved in ASLR.

Diversification makes many versions of the same program to thwart general attacks based on assumptions of a fixed structure.
ASLR breaks the hard coded static locations and randomises the locations of data or code. Making attacks based on assumptions of these locations harder.

Avoiding overflow vulnerabilities

Explain where bounds checking should be performed, especially to ensure "defence-in-depth".

Bounds checking is a shared responsibility and should be carried out at each point. Programmers, languages, compilers, operating systems and hardware should all check bounds.

List some checks which a programmer or static analysis tool should do to prevent overflow vulnerabilities in released code.

Programmers or tools should:

- Check data lengths before writing
- Check array subscripts are within limits
- Check boundary conditions to avoid OBO
- Constrain size of inputs
- Be aware of dangerous API calls to risky code

CWEs

What does “Improper Neutralization of Special Elements in Output used by a Downstream Component” mean?

This is used to stand for input validation. It must be assumed inputs can be influenced by an adversary.
Their input may eventually lead to a downstream component that executes an OS command or queries a database and should be treated accordingly with distrust, checks, and sanitization if applicable.

OS command injections

Why are OS commands executed by application programs?

These may be used because the programming language has no suitable library or simply for convenience and to save time.

Give two mechanisms by which OS commands may be injected by an attacker.

Command injection could be achieved by meta characters in shell commands or altering environment variables.

SQL injections

Give an example of code that is vulnerable to an SQL injection, explain an exploit and its consequences.

Example in lecture

```
$username = $HTTP_POST_VARS['username'];
$password = $HTTP_POST_VARS['passwd'];
$query = "SELECT * FROM logintable WHERE user = '"
        . $username . "' AND pass = '" . $password . "'";

$result = mysql_query($query);
if (!$results)
        die_bad_login();
```

Guaranteed login! Try with:

```
Username: bob' OR user<>'bob'
password: bob' OR pass<>'bob
```

gives

```
SELECT * FROM logintable WHERE user= 'bob' or user<>'bob' AND pass='bob' OR pass<>'bob
```

SQLi classification

Describe three routes by which SQL injection may occur.

SQL injections can be used by user input, cookies, server variables(e.g. HTTP headers) and second order injection.

Describe three auxiliary motives that an attacker may have when using SQL injection techniques to learn about a target.

Finding injectable parameters.
Database server fingerprinting.
Finding database schema.
Escalating privilege at the database level.

SQLi prevention and detection

How would you repair the prototypical example SQLi vulnerability?

Filtering to sanitize inputs.
Prepared queries using fixed parameters

Describe an automatic technique for preventing and an automatic technique for detecting SQLi vulnerabilities.

Detect suspicious code patterns.
Use static taint analysis to detect data flow from input parameters to queries.

Development

Describe 5 secure development lifecycle activities and the points in which they would be used in a compressed 4-stage agile development method (use case, design, code, test).
Use case: abuse cases, security requirements
Design: Architectural risk analysis
Coding: Code review
Testing: Risk analysis, Penetration testing, security operations

What kinds of security problem is code review better at finding compared with architectural risk analysis?

Low-level, implementation problems such as off-by-one errors, infinite loops, unchecked usage of user input etc.

Why is risk analysis difficult to do at the coding level?

Because risk analysis focuses on high level attack scenarios: who are the attackers, what are the assets they want to steal/manipulate. This requires a view on the whole system, as well as an understanding of the business impact of penetrations and value of the assets. This is usually not possible to see at the coding level.

What is the main drawback of penetration testing, especially when it is applied as an absolute measure of security of a software system?

Penetration testing has no accurate sense of coverage. If using a ready made set will usually only cover easy bugs. System and architecture specific bugs may be ignored.
→ Gives a false sense of security.

HTTP Headers

Describe three possible vulnerabilities for a web application posed by an attacker who fabricates HTTP headers rather than using the web app running via a reliable browser.

1. Change the referer header to point to another page on the same website. This might lead to improper authentication if the website relies on the header instead of cookies for authentication.
2. Change the content of POST requests to e.g. include injection commands.
3. Change the User-Agent

Explain the reasons for using POST rather than GET.

POST helps with confidentiality but is not a lone solution
POST should be used for large data, since long URL's can cause software issues.
GET is not intended to provide side-effects so if meaning to do this use POST (inserting a record in a database, sending an email, etc…)

What security guarantees does it provide?

Both GET and POST are inherently insecure, as they lack confidentiality. For network security they are practically the same, because POST exposes just as much information as GET in HTTP (as it is unencrypted). In order to achieve security, one should use HTTPS, which uses SSL or TLS to establish an encrypted connection. An extremely minor thing to note regarding user security is that POST does avoid explicitly displaying user information in the URL (through obscurity, placing it into the HTTP packet, but still susceptible to sniffing), and caching it (often the default with GET). The major provision of POST is to avoid when GET is improperly used to modify data and cached. In a browser, it can then be refreshed (sending another GET), or navigated by third-party programs such as crawlers and search engines, which will delete entire databases, websites, and other sensitive items (even in programs such as web accelerators, which preload cached data operating *as the logged in user*). This is possible because GET is assumed to be an idempotent and safe action in HTTP. To conclude, POST will protect web applications from accidental data modification of the above fashion, but provide essentially nothing in terms of security to the user since transmission is unencrypted.

Cookies

Consider an online grocery merchant that uses a cookie to store the user's shopping basket, including the list of product IDs and their prices,encrypted using a secret key derived from the SID. What threats might be posed and by whom?

URLs

Recap the 8 components of a URL. From a server side point of view, which of these is trustworthy?

Scheme/protocol name.
// indicator of hierarchical URL
"Login:password@" credentials to access (optional)
address to server for data retrieval
:port port number to connect to (optional)
/path/to/resource hierarchical path
?query_string paramters(optional)
#fragment identifier (optional)

From the web app viewpoint, which of these is it most important to validate in output, to protect your users?

XSS

Explain how session stealing works with XSS. How could a reflected XSS attack steal a session?

1: Attacker injects script onto the server and waits for the user.
2: Server passes the session cookie and the attackers script to a visitor
3: Script runs in the victim's browser and passes the session cookie to the attacker.
4: Attacker passes the stolen cookie making the server think he is the victim.

Object references

Why is it important to add defence-in-depth when configuring web servers? Give three examples of ways in which a web application may be restricted by a (separate) server.

Authorization

How might you design a web based admin front-end to a database server, supposing that the server provides a collection of databases and its own notion of user and authorization?

Redirection

What is an open redirect in a web application and why is it undesirable?

An open redirect is an application that takes a parameter and redirects a user to the parameter value without any validation. This vulnerability is used in phishing attacks to get users to visit malicious sites without realizing it.

Handling sensitive data

Describe three ways that a web application programming flaw might result in stored private user data being leaked.

Vulnerability to SQL injection : get information about database, such as username
No input/output validation for user input stored on website: makes XSS possible → allows attacker to steal cookies.
Vulnerable to OS command injection

CSRF

Draw a picture showing how a CSRF attack might work against an online banking user. What might an attacker be able to do?

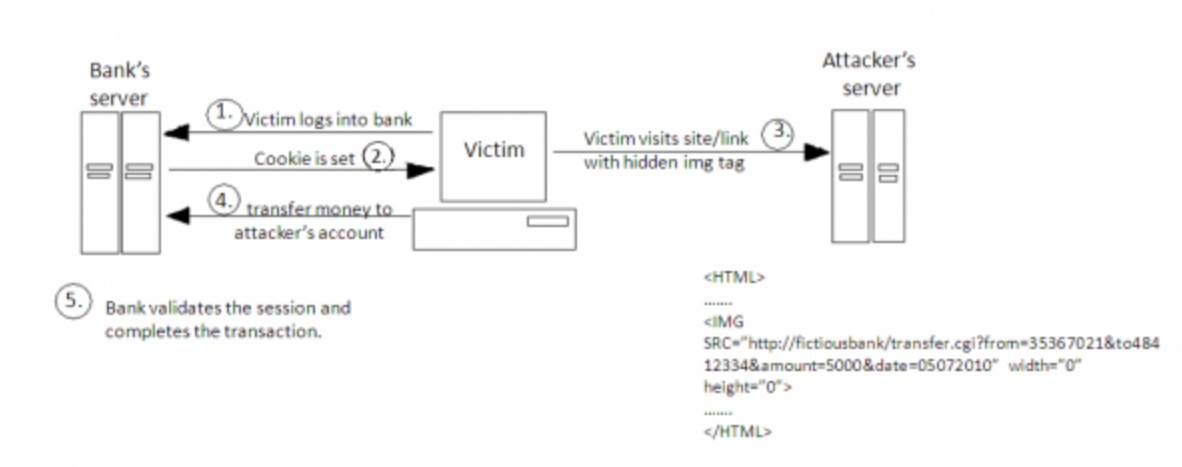


Diagram from http://www.opensourceforu.com/2010/11/securing-apache-part-3-xsrf-csrf/

What does a defence mechanism against CSRF need to be able to do?
Use a good framework that provides built in protection such as “double cookie” trick or special CSRF token.
(Prevent users from running script provided by malicious user without awareness.)

Deployment

Give three recommendations for securing the deployment environment (server, database, application framework) for a web app.

Static versus dynamic analysis

Static analysis requires access to source (sometimes binary) code. What advantages does that enable?

It examines every code path and every possible input. It can also run code before complete. Analysis of already compiled binary code might be interesting for assessing the security risk of existing software where the source does not exist anymore.

Why do practical static analysis tools both miss problems and report false problems?

Security and correctness must be approximated so tool cannot be perfectly precise.

Types of static analysis tool

Apart from type and style checking, describe three other jobs a static analysis tool may perform.

Infer meaning from program understanding.
Ensure no bad behaviour from property checking.
Ensure correct behaviour from program verification.
Detect likely errors using bug finding.

Race Conditions

Using an example based on Unix file handling, describe what a race condition is, and explain how an attacker can exploit it.

Using the file path repeatedly, it could be changed in between function calls.

Data races

Describe the two necessary conditions for a program to contain a data race.

Two or more threads access a shared variable potentially at the same time and at least one of their accesses is a write.

Discuss whether it is possible for a racy program to compute a completely arbitrary value.

No, the output of a racy program depends on the timing of uncontrollable events.

Program securely

Describe two programming techniques that can be used to avoid security issues with race conditions.

Ensure operations occur atomically by enforcing mutual exclusion or using a transaction mechanism.

Question. How can we write API functions that ensure atomicity?