#### Session objectives

- Be familiar with the most common implementation errors leading to security vulnerabilities
- Start developing a good methodology for secure design and implementation

#### • 2010 CWE/SANS Top 25 Most Dangerous Software Errors

- Robert Seacord: Secure Coding in C and C++
  - https://www.securecoding.cert.org/confluence/ display/seccode/Top+10+Secure+Coding+Practices



3/1

Autumn 2011 - Week 12

#### Prof Hans Georg Schaathun

Software Security

Top Vulnerabilities

## Common Weakness Enumeration

- 2010 CWE/SANS Top 25 Most Dangerous Software Errors
  - http://cwe.mitre.org/top25/index.html
- A very few key vulnerabilities behind most incidents
- Massive benefit from controlling the top few



Software Security Information Security

Prof Hans Georg Schaathun

University of Surrey/Ålesund University College



			TY OF CEY
Prof Hans Georg Schaathun	Software Security	Autumn 2011 - Week 12	1/1
	The session		
Security or Useabili	ty		

- This chapter is largely about software bugs
  - Is this security?
  - ... or is it useability?
- Answer is yes
  - Bugs are user (programmer) mistakes useability.
  - Many bugs *cause* security vulnerabilities.
- Useability is a prerequisite of security.



#### Top Vulnerabilities

#### Top 9

- Improper neutralisation of input during web page generation (Cross-Site Scripting)
- Improper neutralisation of Special Elements in SQL Commands (SQL Injection)
- Buffer overflow without Checking of Input Size
- Cross-Site Request Forgery
- Improper Access Control (Authorisation)
- 8 Reliance on Untrusted Inputs in a Security Decision
- Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)
- Unrestricted Upload of File with Dangerous Type
- Improper neutralisation of Special Elements used in an OS Command

Prof Hans Georg Schaathun	Software Security	Autumn 2011 – Week 12	7 / 1

Input Checking

## Top 9

- Improper neutralisation of input during web page generation (Cross-Site Scripting)
- Improper neutralisation of Special Elements in SQL Commands (SQL Injection)
- O Buffer overflow without Checking of Input Size
- Cross-Site Request Forgery
- Improper Access Control (Authorisation)
- 8 Reliance on Untrusted Inputs in a Security Decision
- Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)
- Our Content of The State of
- Improper neutralisation of Special Elements used in an OS Command

## **Trusting Input**

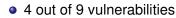
Most of the top vulnerabilities relate to user input ...

- Cross-Site Scripting
- SQL Injection
- Reliance on Untrusted Input
- File upload
- Path traversal
- Special elements in OS commands

Integrity of Code and Data ...

## Input Checking

Prof Hans Georg Schaathun



- very similar instances of input checking
- E.G. SQL injection
  - SELECT \* FROM users WHERE name='John' ;

Software Security

Input Checking

- Now, say the user enters a name, instead of using ' John'
  - $\bullet$  SELECT  $\star$  FROM users WHERE name='\$n' ;
- What if the user enters
  - Mary' ; DROP TABLE users ; ... '
- What happens?

SURREY

Autumn 2011 - Week 12

#### Input Checking

#### What may happen

SELECT \* FROM users WHERE name='Mary' ; DROP TABLE users ; ... ''

#### • We select user Mary, and then drop the table

- Successful availability attack the table is destroyed
- The string delimiter (') in the input
  - allows the user to terminate the string (which was expected)
  - and add another command (which was not expected)

SELECT \* FROM users WHERE name='Mary'' ; DROP TABLE users ; ... ''

- The special character is escaped
  - and treated as part of the string
- The offending Command is now part of the name
  - and not harmful

What should happen

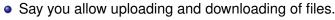
#### Autumn 2011 - Week 12 Prof Hans Georg Schaathun Software Security 12/1Prof Hans Georg Schaathun Software Security Input Checking Input Checking

## **Cross-Site Scripting**

http://www.phpnuke.org/user.php?op=userinfo&uname= <script>alert (document.cookie);</script>

- Malicious code passed as an HTTP GET argument
- Principle as before
- No input checking in the web page
  - causes execution of code from the user
- No limit to what this can achieve
- Other web pages (other sites)
  - can hide code actually loading the URL
  - no user interaction at all

Source: http://www.cgisecurity.com/xss-faq.html



- the user specifies the filename
- a directory is hardcoded and prepended
- so the user enters foobar. jpeq
  - it becomes /opt/archive/foobar.jpeg
  - safe enough

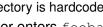
Path traversal

• What if the user enters .../.../etc/passwd?



13/1

Autumn 2011 - Week 12





#### Input Checking

#### **Dangerous Filenames**

- Some systems deduce file type from filename
  - e.g. \*. jpg for JFIF/EXIF images
  - e.g. \*.php for PHP scripts
- Dangerous: the filename is under user control
- Allows user to mark data as code and vice versa
- E.g. uploaded files on a web server
  - do you allow the users to upload PHP scripts?
  - might your server execute them?

Input Checking

Passing information to external programs

Improper neutralisation of Special Elements used in an OS Command

- Calling external programs is high risk
  - library calls is lower risk
  - Why is this?
- library calls provide type checking
  - external programs take arguments as strings
  - ... control codes and data are mixed

## 



- rlogin(1) used to allow remote login access to Unix systems
  - rlogin [-luser] hostname
- The rlogin client contacts a remote host which runs login(1)
  - Running rlogin -1 css1hs kyle, would
  - ... on kyle, cause the running of login css1hs.
- Now, login(1) has many uses,
  - login -froot is a forced login (as root)
  - ... no password prompt
- rlogin -1 -froot kyle what happens?
  - login -froot superuser login without password

Software Security

- Unused functionality is exploited.
- ... unless rlogin sanitises the input

## 

# Prof Hans Georg Schaathun

Input Checking

## What to do?

- Two methods (principles):
  - Input checking: reject unexpected input
  - Sanitising: accept the input, make sure it is handled correctly (escaping)

Software Security

- Which is easiest? Which is best?
- Restrictive Input Checking
  - err on the side of caution
  - relatively simple accept a small set of safe inputs
  - availability risk (reject good input)
  - good incidence response allow quick bug fixes
- Sanitising requires comprehensive understanding
  - how to sanitise
  - what is the effect of each possible input?
  - the SQL example cannot be solved by input checking
    - O'Brian is a valid name ...

LINIVERSITY O

T SURREY



17/1

Autumn 2011 - Week 12

#### Input Checking

### Good practice

- Take a critical view of all input
  - Don't trust anyone
- Have a firm understanding of what the input should look like
  - don't accept odd input
- Be aware of any special characters where the data is used
  - be wary of quotation marks ('/"), backslashes, control characters etc.
  - special scenarios like slashes in filenames
- On't use user input if you do not have to
  - e.g. filenames can be generated by the system
- Spend some time on every instance of user input

				REY REY
Prof Hans Georg Schaathun	Software	Security	Autumn 2011 - Week 12	20 / 1
	Other Issues	Management Informa	ation	

Untrusted Inputs in a Security Decision

- What controls can you use against this?
  - Technical
    - Unlikely Intelligent Input needed to choose trusted sources
  - Operational
    - Yes good operational information gathering
  - Managerial
    - Yes choose trust policy

#### **Decision Making**

Reliance on Untrusted Inputs in a Security Decision.

- Decision Making depends on Information
- Where does this information come from?
- What CObIT Criteria are essential for this information?
  - Integrity
  - Reliability of Management Information
- Can your adversaries have forged information?
- Are your decissions steered by your enemy?

# Prof Hans Georg Schaathun Software Security Autumn 2011 – Week 12 22 / 1

Other Issues Management Information

## **Quick Summary**

• Decissions are based on Information

• ensure reliability and integrity of this information



#### Other Issues The last three of the nine

## Top 9

- Improper neutralisation of input during web page generation (Cross-Site Scripting)
- Improper neutralisation of Special Elements in SQL Commands (SQL Injection)
- Buffer overflow without Checking of Input Size
- Cross-Site Request Forgery
- Improper Access Control (Authorisation)
- 8 Reliance on Untrusted Inputs in a Security Decision
- Improper Limitation of a Pathname to a Restricted Directory (Path Traversal)
- Unrestricted Upload of File with Dangerous Type
- Improper neutralisation of Special Elements used in an OS Command

Software Security

#### Prof Hans Georg Schaathun

Other Issues The last three of the nine

#### Why is this code insecure?

char Password[12] ;	
<pre>gets(Password) ; if ( !strcmp( Password, "goodpass" ) ) return true else return (false) ; }</pre>	;
<pre>void main(void) {    bool PwStatus ;</pre>	
<pre>puts ("Enter_password:") ; PwStatus = IsPasswordOkay() ; if (PwStatus == false) {     puts("Access_denied") ;     exit (-1) ; }</pre>	
else puts ("Access_granted") ; }	

#### **Buffer Overflows**

- Classic problem
- Limited memory buffer
  - writing unlimited data objects often user input
  - system does not check the buffer limits
- Clever attackers can
  - overwrite executable code
  - inserting custom code to be executed



## Cross-Site Request Forgery (CSRF)

like the stranger in the airport, asking you to take just this parcel along on the flight ...

- Web vulnerability
  - trick a user's client to make your request
  - request made with his credentials
- Integrity problem
  - Attackers can forge requests
- The attacker can gain all the priviliges of the user



Autumn 2011 - Week 12 27 / 1

Autumn 2011 - Week 12

25/1

#### Other Issues The last three of the nine

#### Improper Access Control

- Fairly obvious restrict access to authorised users
- But, get the roles right
  - should match business roles
- The exercise for next week explores management of
  - access
  - privileges
  - identity

Prof Hans Georg Schaathun	Software	Security	Autumn 2011 - Week 12	29 / 1	
	Coding Practices	Default Deny			

**Default Deny** 

or principle of least privilege

- Default Deny is a General Principle with many Applications
  - Access Control
  - Input Validation
  - Feature Selection
- Advantage: prevents unnecessary integrity/confidentiality risks
  - accepting risks only when necessary
- Disadvantage: availability risk
  - Mitigation by incident respons  $\rightarrow$  bug fix

## Seacord's 10 Principles

- Validate input
- Heed compiler warnings
- Architect and design for security policies.
- Keep it simple.
- Default deny.
- Adhere to the principle of least privilege.
- Sanitize data sent to other systems.
- Practice defense in depth.
- Use effective quality assurance techniques.
- Adopt a secure coding standard.



# Prof Hans Georg Schaathun Software Security Autumn 2011 – Week 12 31 / 1

Coding Practices Default Deny

#### Input Checking Default deny

- Defining harmful inputs is hard
- Defining correct input is easier
- Default deny will reject the input when in doubt
- Note that the SQL example,
  - the input is both valid and harmful
  - that's why you need sanitisation as well
- You can overdo it
  - many webpages validate email addresses
  - and reject the plus sign (+)
- The plus sign is valid according to the RFC
  - and has a very important function in non-MS mail servers



SURREY

#### Coding Practices Default Deny

# Example: path names

Default deny in input validation

- Suppose you write an application, where users upload files
  - The user can specify a filename, e.g. holiday.jpg,
  - ... and you prepend a directory name, e.g. /public/images/
- How can this be exploited?
- Suppose the users use filename /../../etc/passwd.
- How do we avoid this?
- Input checking is possible;
  - ../ is an illegal substring.

			SITY OF REY	
Prof Hans Georg Schaathun	Software	Security	Autumn 2011 – Week 12	34 / 1
	Coding Practices	Default Deny		

```
Unicode encoding
```

- Each byte has a prefix
  - 0 one-byte character
  - 110 first byte of two-byte character
  - 1110 first byte of three-byte character
  - 11110 first byte of four-byte character
  - 10 second or later byte of multi-byte character
- Remaining bits contain a unicode character number
  - 1 byte : 7 bits
  - 2 bytes : 11 bits (5+6)
  - 3 bytes : 16 bits (4+6+6)
  - 4 bytes : 21 bits (3+6+6+6)
- Only shortest possible representation is legal
  - but illegal representations are often accepted

#### Character Encoding Vulnerabilities in Unicode

- Unicode collects characters for (almost) every language
- UTF-8 is the most common encoding of Unicode
- Variable length characters
  - ASCII (American 7-bit character set) uses one byte
     Ensuring compatibility.
  - Western European (non-ASCII) characters use two bytes
  - More exotic characters require 3 or 4 bytes

# 

35/1

# Prof Hans Georg Schaathun Software Security Autumn 2011 – Week 12

Coding Practices Default Deny

# Exploiting it

- Your application bans filenames containing ../
- But there are many ways to write /
  - / is Unicode 00101111
    - 1 byte : 0010 1111
    - 2 byte : 1100 0000 1010 1111
    - 3 byte : 1110 0000 1000 0000 1010 1111
    - 4 byte : 1111 0000 1000 0000 1000 0000 1010 1111
- So if your system accepts multi-byte forms,
- ... your input checking has to ban all representations of /.
- Default deny makes it easier
  - Accept only the canonical form





#### Coding Practices Default Deny

#### **Canonical Representation**

- UTF-8 is an example of the use of canonical representations
- Several equivalent forms are defined
- Only the shortest form is canonical
- Before a safe comparison can be made
- ... data should be converted into canonical form

#### **Example:** Napster filenames

- Napster was ordered by court to block certain songs
- Solutions

Prof Hans Georg Schaathun

- filter downloads based on filename
- Napster users by-passed this control
  - using equivalent (variations of) the song titles

Software Security

LINIVERSITY ( SURREY 5

39/1

Autumn 2011 - Week 12

- Almost impossible to control
  - title equivalence is defined by the users...
- Blatant breakdown of 'Default Permit'

			l
Prof Hans Georg Schaathun	Software Security	Autumn 2011 – Week 12 38 /	1
	Closure		
Conclusions			

- Secure coding is an essential part of software development
  - relatively new field
- The Top 25 Vulnerabilities database is a good source
  - avoid the Top 5 and you will be better than average ...
  - the list is updated regularly check the latest version
- Practices may vary between languages
  - try to look up a book for whatever language you use

