Secure programming

Teaching Arrangements

- Course Coordinator:
  - Yuval Yarom
  - Ingkarni Wardli 4.23
  - yval@cs.adelaide.edu.au
  - **Do not expect me to know who you are!!!**
- Tutor
  - Sioli O'Connell
  - a1690418@student.adelaide.edu.au
- Online resources available on Canvas
  - https://myuni-canvas.adelaide.edu.au/courses/36233
    - Not yet ready

Admin

- No lecture on week 3

Secure Programming

- Advanced course in computer security
- Covers four main topics
  - Common vulnerabilities
  - Mitigation techniques
  - Cryptographic primitives
  - Side-channel attacks

Assumed knowledge

- C/C++
  - The programming language is C, but if you know C++, learning C is relatively easy.
- Computer Systems
  - Machine language, caches, memory management unit, number representation, calling conventions.
- Operating Systems
  - Processes, threads, scheduling, virtual memory, file systems.

Submission guidelines

- Markers are instructed to not mark your assignments if you fail to follow instructions.
  - Acceptable: .pdf, .tar, .tgz.
    - Contents must match the file name extension
  - Not acceptable: .doc, .docx, .zip, .rar, etc.
- Do not submit binaries or other automatically generated files. (PDF are an exception)
- Every file you submit must display your name and your student numbers
  - In some cases there are specific requirements on how these are to be displayed.

Books

- Common vulnerabilities and some mitigation techniques:
  - M. Howard and D. LeBlanc “Writing Secure Code”
  - M. Howard, D. LeBlanc and J. Viega “24 Deadly Sins of Software Security”
- Cryptographic primitives
  - Bruce Schneier “Applied Cryptography”
- Side channel attacks and other mitigation techniques
  - No books yet

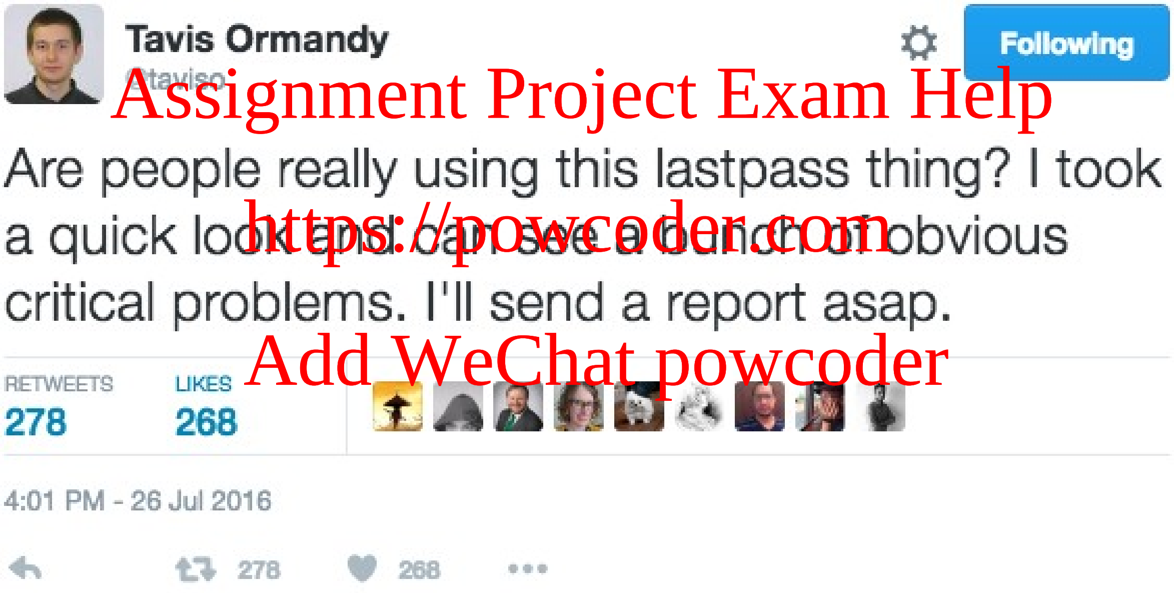
What is this course about?

- Security is all about protecting assets
- Secure software protects the assets that the software uses
  - **Confidentiality**
  - **Integrity**
  - **Availability**
- The aim of this course:
  - Give you (some of) the tools for developing secure software

Where is software security required?

- Managing users passwords?

New LastPass vulnerabilities



Tavis Ormandy
@taviso

Are people really using this lastpass thing? I took a quick look and can see a bunch of obvious critical problems. I'll send a report asap.

RETWEETS 278 LIKES 268

4:01 PM - 26 Jul 2016

278 268

Where is software security required?

- Managing users passwords?
- Validating Web site certificates?

Goto fail

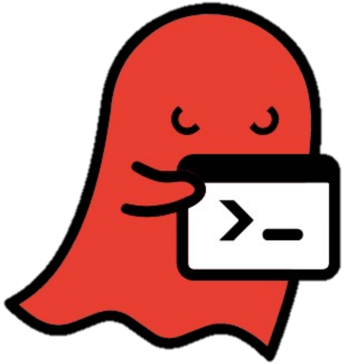
```
if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
    goto fail;
if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
    goto fail;
    goto fail;
if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
    goto fail;

err = sslRawVerify(ctx, …
```

Where is software security required?

- Managing users passwords?
- Validating Web site certificates?
- Resolving host names?

GHOST

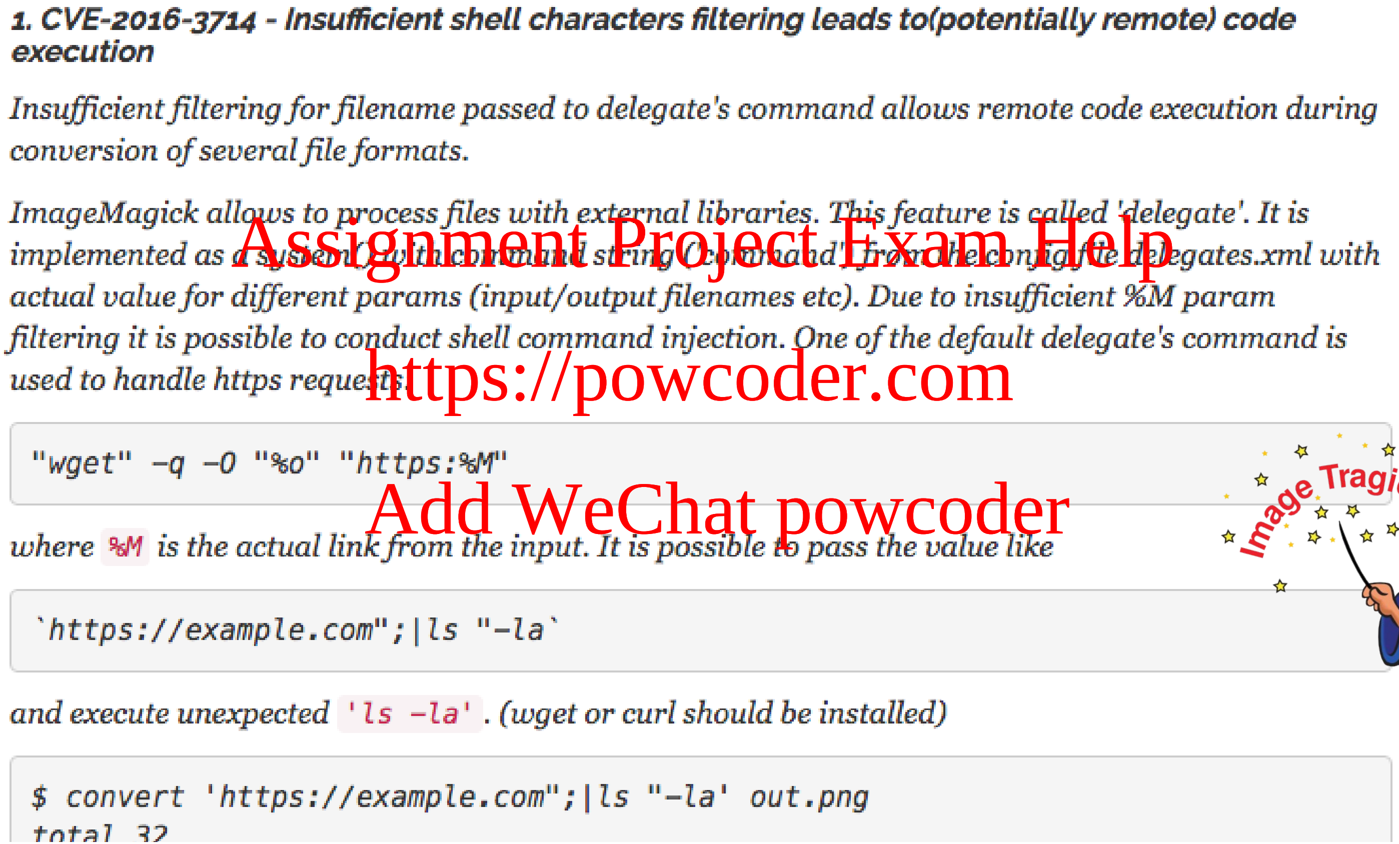
```
size_needed = (sizeof (*host_addr)
         + sizeof (*h_addr_ptrs) + strlen (name) + 1);

.
.
.
host_addr = (host_addr_t *) *buffer;
h_addr_ptrs = (host_addr_list_t *)
        ((char *) host_addr + sizeof (*host_addr));
h_alias_ptr = (char **) ((char *) h_addr_ptrs +
        sizeof (*h_addr_ptrs));
hostname = (char *) h_alias_ptr + sizeof (*h_alias_ptr);
.
.
.
   resbuf->h_name = strcpy (hostname, name);
```



Where is software security required?

- Managing users passwords?
- Validating Web site certificates?
- Resolving host names?
- Processing images?

ImageTragick



1. CVE-2016-3714 - Insufficient shell characters filtering leads to(potentially remote) code execution

Insufficient filtering for filename passed to delegate's command allows remote code execution during conversion of several file formats.

ImageMagick allows to process files with external libraries. This feature is called 'delegate'. It is implemented as a system() with command string ('command') from the config file delegates.xml with actual value for different params (input/output filenames etc). Due to insufficient %M param filtering it is possible to conduct shell command injection. One of the default delegate's command is used to handle https requests:

```
"wget" -q -O "%o" "https:%M"
```

where `%M` *is the actual link from the input. It is possible to pass the value like*

```
`https://example.com";|ls "-la`
```

and execute unexpected `'ls -la'`*. (wget or curl should be installed)*

```
$ convert 'https://example.com";|ls "-la' out.png
total 32
```

Source: imagetragick.com

Where is software security required?

- Managing users passwords?
- Validating Web site certificates?
- Resolving host names?
- Processing images?
- **Everywhere!**

Thinking like an attacker – Bike lock

- Lock has key and some kind of cable/chain to link bike to the bike rack
- Engineer: focuses on making lock unbreakable. Resistant to being picked or cut. Best lock in the world

- Attacker / security engineer: focuses on the whole system. How can the **system** fail?
- **What does fail mean?**

What does **fail** mean?

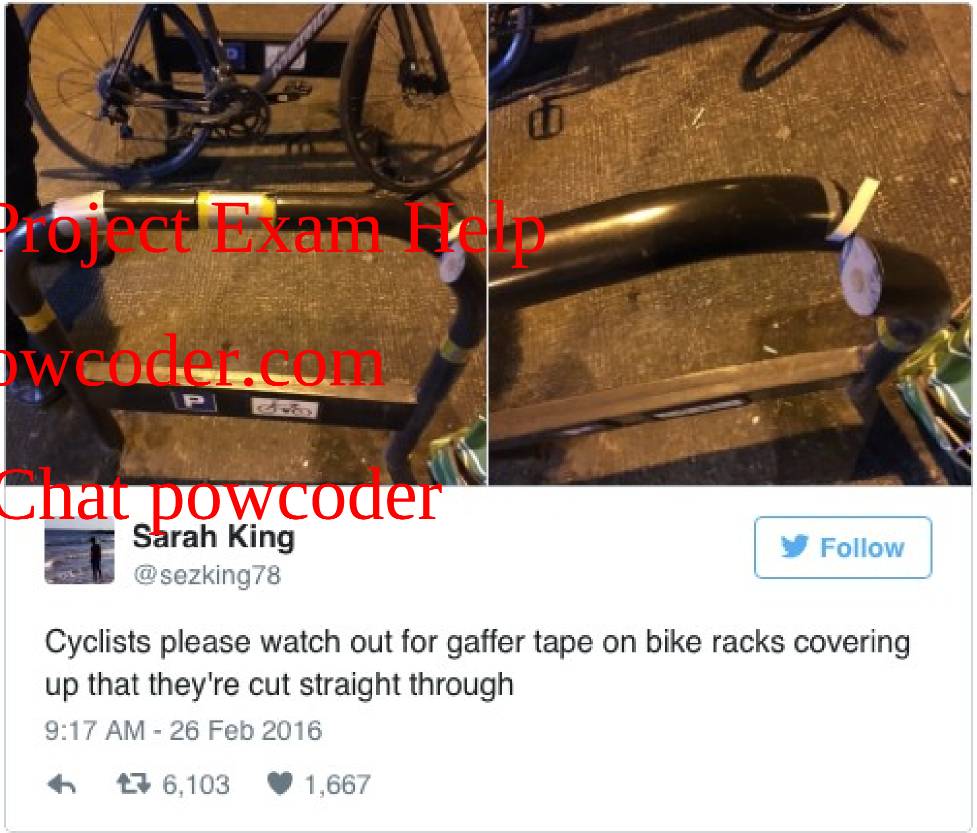
- Obvious: you steal the bike
- Less obvious: you steal part of the bike
- Less obvious: you steal the lock
- Less obvious: you render the bike inoperable
- Less obvious: you render the lock inoperable

Stealing the bike

- The obvious attack – break the lock.
  - However – the lock is likely to be over-engineered.

- Lock and chain may be unbreakable. What about the bike rack?
  - Bolt cutters, angle grinders, oxy torch, shaped explosive charge, axe

Real-life examples



Sarah King
@sezking78

Cyclists please watch out for gaffer tape on bike racks covering up that they're cut straight through

9:17 AM - 26 Feb 2016

Another real-life example

Source: YouTube

Stealing a part of the bike

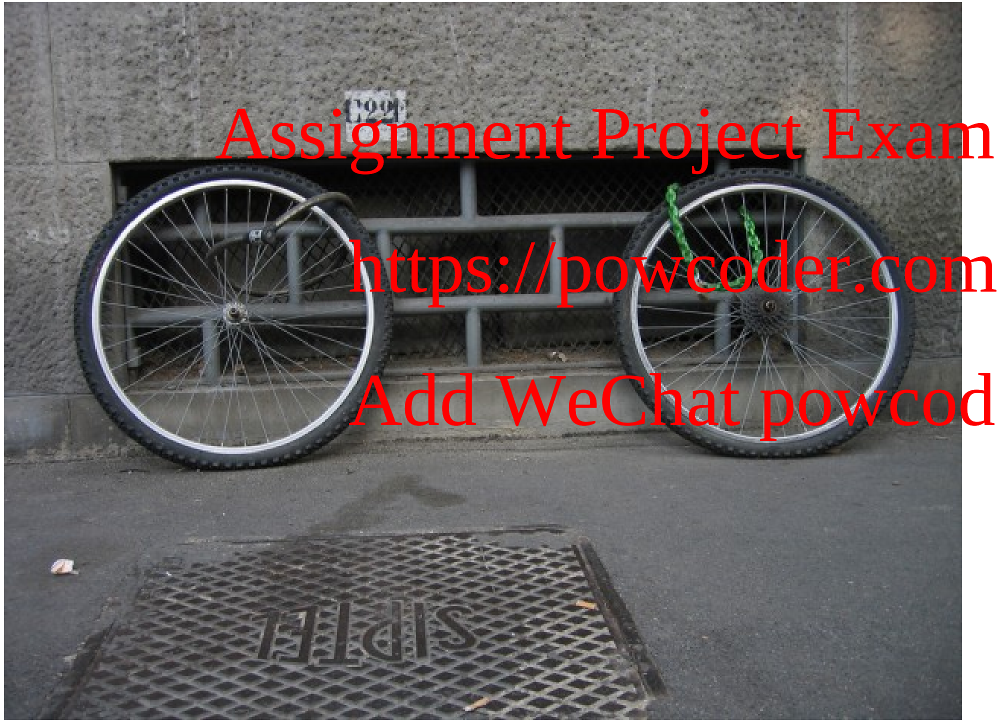
- Use a spanner
  - Or the quick release mechanism

- Leave front wheel attached to post
  - Walk into the local bike shop and complain someone stole your front wheel – slap down $50 and you have a new wheel and a whole new bike!

Real-life examples

Source: blog.priceonomics.com

Real-life example 5



Source: simplisafe.com

Real-life example

Source: www.npr.org

Real-life example 7

Source: scmp.com

Other Options

- Render the lock inoperable
  - Superglue
  - Oxy torch
  - Broken key

- Steal the lock
  - May be harder than stealing the bike
- Render the bike inoperable
  - Easy. Little benefit for the attacker.

Real-life example 8

Why security is hard

- Functional requirements vs. security requirements

Functional	Security
Editors can delete documents	*Only editors are able to delete documents*
What can be done	Who cannot do what

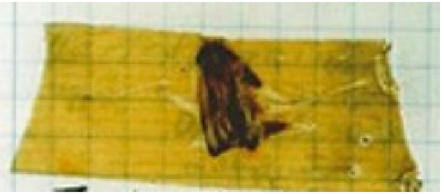
Assumptions

- We deal with security requirements by making assumptions
  - The only way to delete documents is by clicking 'Delete Document' on the Web interface.
  - Editors do not share their passwords
  - Attackers do not have access to the database
  - Attackers do not have access to the database server
  - Attackers are not going to use tactical nuclear weapons to destroy documents
- All too often, the assumptions are implicit

Vulnerabilities and bugs

- Vulnerability: A flaw in a product that makes it infeasible—even when using the product properly—to maintain the required level of security

- Bug: An implementation error that results in an unintended behaviour

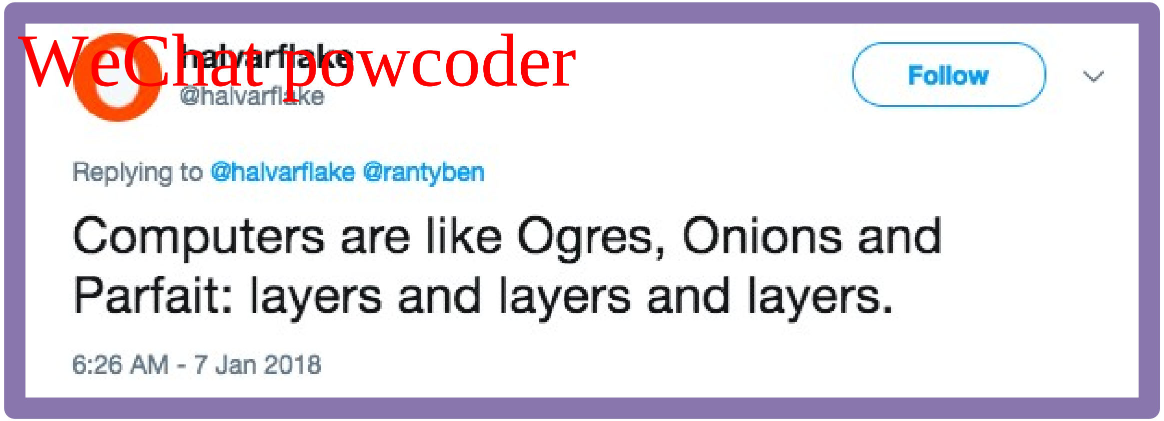


- Not every vulnerability is a bug
  - But many are

- Not every bug is a vulnerability
  - It may be hard to identify "safe" bugs
  - Eliminating bugs also eliminates vulnerabilities

Abstractions and bugs

- Abstractions are the main tool we use to manage complexity
- An implementation of an abstraction provides an interface
- The consumer of the abstraction uses it to provide higher-level abstractions
- Bugs are, usually, the result of failed abstractions



halvarflake
@halvarflake

Replying to @halvarflake @rantyben

Computers are like Ogres, Onions and Parfait: layers and layers and layers.

6:26 AM - 7 Jan 2018