Access Control and Privacy Policies (1)

Email: christian.urban at kcl.ac.uk
Office: S1.27 (1st floor Strand Building)
Slides: KEATS

Lavabit

My Fellow Users,

I have been forced to make a difficult decision: to become complicit in crimes against the American people or walk away from nearly ten years of hard work by shutting down Lavabit. After significant soul searching, I have decided to suspend operations. I wish that I could legally share with you the events that led to my decision. I cannot. I feel you deserve to know what's going on—the first amendment is supposed to guarantee me the freedom to speak out in situations like this. Unfortunately, Congress has passed laws that say otherwise. As things currently stand, I cannot share my experiences over the last six weeks, even though I have twice made the appropriate requests.

What's going to happen now? We've already started preparing the paperwork needed to continue to fight for the Constitution in the Fourth Circuit Court of Appeals. A favorable decision would allow me resurrect Lavabit as an American company.

This experience has taught me one very important lesson: without congressional action or a strong judicial precedent, I would _strongly_ recommend against anyone trusting their private data to a company with physical ties to the United States.

Sincerely,
Ladar Levison
Owner and Operator, Lavabit LLC

Defending the constitution is expensive! Help us by donating to the Lavabit Legal Defense Fund here.

Lavabit email service closed down on 8 August 2013.
goo.gl/bgSrVp

Also Bad Guys

Annonymous Hacker operating a 10k bonnet using the ZeuS hacking tool wrote:

"FYI I do not cash out the bank accounts or credit cards, I just sell the information (I know, its just as bad, hur dur...), there isn't even a law against such in most countries, dealing with stolen information is most of the time a legaly greyzone (I was just as surprised when I looked it up), I'm not talking about 3rd world countries, but about European like Spain (The Mariposa botnet owner never got charged, because a botnet isn't illegal, only abusing CC information is, but that did other guys)."

goo.gl/UWluh0

This is a Misconception!

Without encryption:

Alice ⟶ NSA ⟶ Bob

With encryption:

Alice ⟶ NSA ⟶ Bob

The NSA can probably not brute force magically better than the "public".

Security Engineers

Security engineers require a particular **mindset**:

> "Security engineers — at least the good ones — see the world differently. They can't walk into a store without noticing how they might shoplift. They can't use a computer without wondering about the security vulnerabilities. They can't vote without trying to figure out how to vote twice. They just can't help it."
>
> —Bruce Schneier
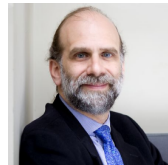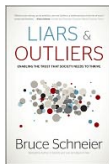
Chip-and-PIN

- Chip-and-PIN was introduced in the UK in 2004
- before that customers had to sign a receipt
- Is Chip-and-PIN a more secure system?

(Some other countries still use the old method.)

Yes ...

> "Chip-and-PIN is so effective in this country [UK] that fraudsters are starting to move their activities overseas," said Emile Abu-Shakra, spokesman for Lloyds TSB (in the Guardian, 2006).

- mag-stripe cards cannot be cloned anymore
- stolen or cloned cards need to be used abroad
- fraud on lost, stolen and counterfeit credit cards was down £60m (24%) on 2004's figure

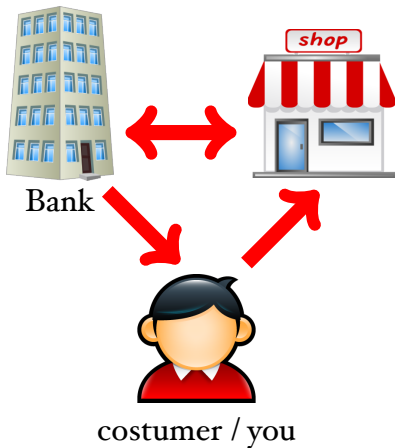# But let's see ...

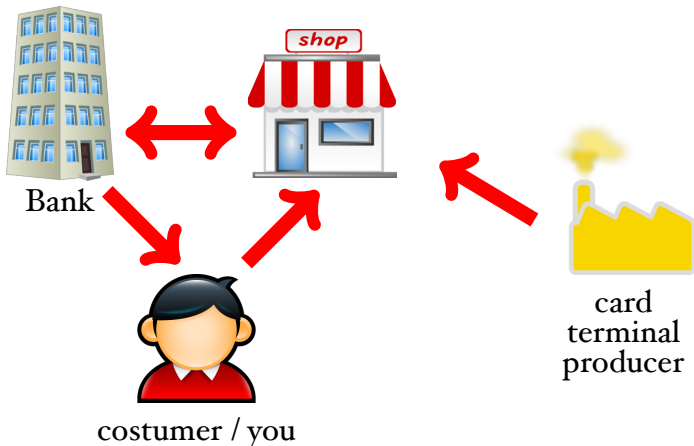Bank

shop

costumer / you

But let's see ...

Bank

costumer / you

But let's see ...

Bank

shop

costumer / you

But let's see ...

Bank

shop

costumer / you

card
terminal
producer

Chip-and-PIN

- A "tamperesitant" terminal playing Tetris on youtube.
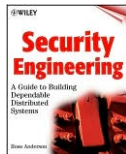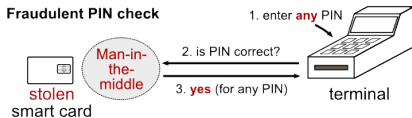
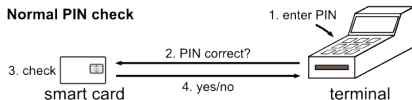 (http://www.youtube.com/watch?v=wWTzkD9M0sU)

Chip-and-PIN

- in 2006, Shell petrol stations stopped accepting Chip-and-PIN after £1m had been stolen from customer accounts

- in 2008, hundreds of card readers for use in Britain, Ireland, the Netherlands, Denmark, and Belgium had been expertly tampered with shortly after manufacture so that details and PINs of credit cards were sent during the 9 months before over mobile phone networks to criminals in Lahore, Pakistan
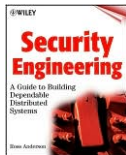
Chip-and-PIN is Broken

- man-in-the-middle attacks by the group around Ross Anderson

Normal PIN check

1. enter PIN

2. PIN correct?

3. check   smart card

4. yes/no   terminal

Fraudulent PIN check

1. enter **any** PIN

Man-in-the-middle

2. is PIN correct?

3. **yes** (for any PIN)

stolen smart card   terminal

on BBC Newsnight
in 2010 or youtube

Chip-and-PIN is Really Broken

- same group successfully attacked this year card readers and ATM machines
- the problem: several types of ATMs generate poor random numbers, which are used as nonces

The Problem ...
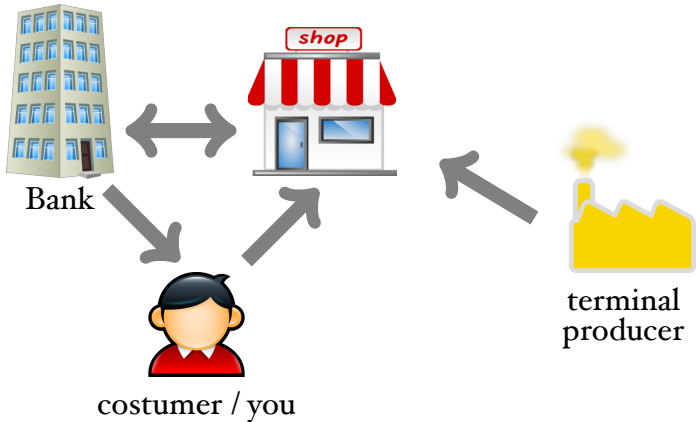
- the burden of proof for fraud and financial liability was shifted to the costumer
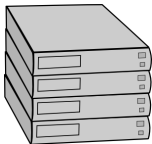
Being Screwed Again

- **Responsibility**
  "You understand that you are financially
  responsible for all uses of RBS Secure."

 https://www.rbssecure.co.uk/rbs/tdsecure/terms_of_use.jsp

Web Applications

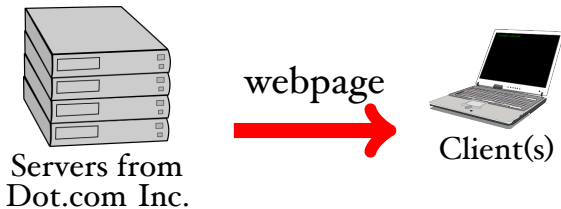Servers from
Dot.com Inc.

Client(s)

- What are pitfalls and best practices?

Web Applications

Servers from
Dot.com Inc.

GET request

Client(s)

- What are pitfalls and best practices?

Web Applications

webpage →

Servers from Dot.com Inc.

Client(s)

- What are pitfalls and best practices?

Web Applications

POST data

Servers from
Dot.com Inc.

Client(s)
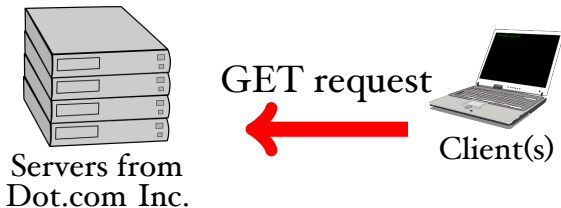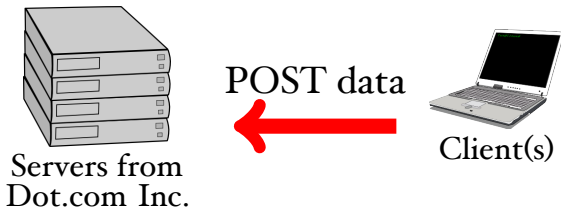
- What are pitfalls and best practices?

Scala + Play

a simple response from the server:

```scala
1  package controllers
2  import play.api.mvc._
3
4  object Application extends Controller {
5
6    // answering a GET request
7    val index = Action { request =>
8      Ok("Hello world!")
9    }
10 }
```

alternative response:
Ok("<H1>Hello world!</H1>").as(HTML)
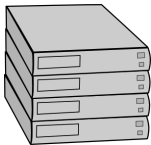
```scala
object Application extends Controller {

  // GET request -> present login form
  val index = Action { request =>

    val form =
      """<form method="post">
          Login: <input type="text" name="login"><br>
          Password: <input type="password" name="password"><br>
          <input type="submit"></form>"""

    Ok(form).as(HTML)
  }

  // POST data: processing the login data
  val receive = Action { request =>

    val form_data = Form(tuple("login" -> text, "password" -> text))

    val (login, password) = form_data.bindFromRequest()(request).get

    Ok("Received login: " + login + " and password: " + password)
  }
}
```
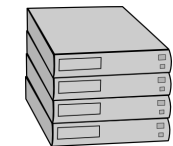
Cookies

GET request

Servers from
Dot.com Inc.

Client

Cookies

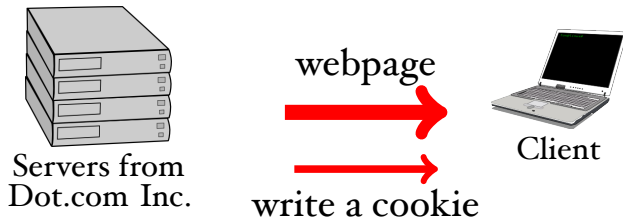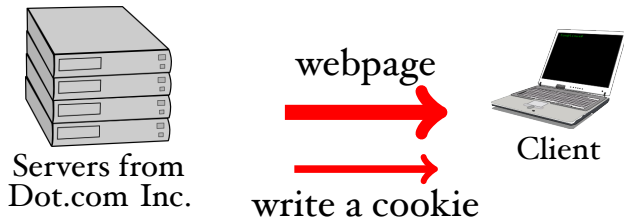Servers from Dot.com Inc.

GET request

read a cookie

Client

Cookies

Cookies

- cookies: max 4KB data
- cookie theft, cross-site scripting attacks
- session cookies, persistent cookies, HttpOnly cookies, third-party cookies, zombie cookies
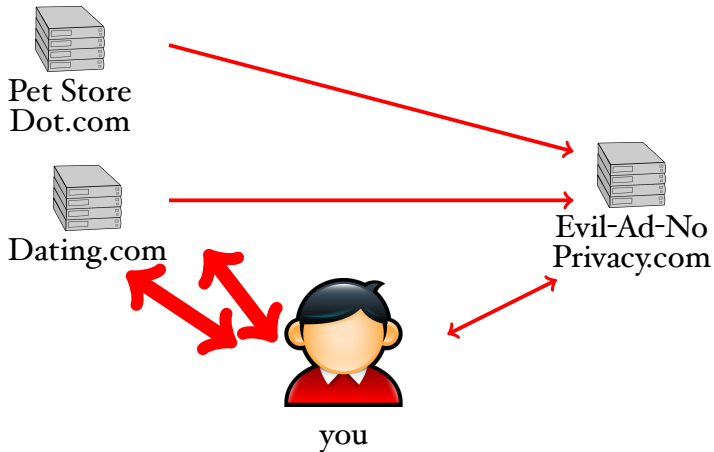
Cookies

EU Privacy Directive about Cookies:

"In May 2011, a European Union law was passed stating that websites that leave non-essential cookies on visitors' devices have to alert the visitor and get acceptance from them. This law applies to both individuals and businesses based in the EU regardless of the nationality of their website's visitors or the location of their web host. It is not enough to simply update a website's terms and conditions or privacy policy. The deadline to comply with the new EU cookie law was 26th May 2012 and failure to do so could mean a fine of up to £500,000." →BBC News

- session cookies, persistent cookies, HttpOnly cookies, third-party cookies, zombie cookies

- While cookies are per web-page, this can be easily circumvented.

My First Webapp

GET request:

1. read the cookie from client
2. if none is present, set visits to 0
3. if cookie is present, extract visits counter
4. if visits is greater or equal 10,
   print a valued customer message
   otherwise just a normal message
5. increase visits by 1 and store new cookie with client

```scala
object Application extends Controller {

  def gt_cookie(c: Option[Cookie]) : Int = c.map(_.value) match {
    case Some(s) if (s.forall(_.isDigit)) => s.toInt
    case _ => 0
  }

  def mk_cookie(i: Int) : Cookie = {
    Cookie("visits", i.toString)
  }

  // GET request: read cookie data first
  def index = Action { request =>

    val visits_cookie = request.cookies.get("visits")
    val visits = gt_cookie(visits_cookie)

    val msg1 = "You are a valued customer who has visited this site %d
    val msg2 = "You have visited this site %d times."
    val msg =
      if (visits >= 10) msg1.format(visits) else msg2.format(visits)

    //send with new cookie
    Ok(msg).as(HTML).withCookies(mk_cookie(visits + 1))
  }
}
```

- data integrity needs to be ensured

```scala
object Application extends Controller {

  //SHA-1, SHA-256
  def mk_hash(s: String) : String = {
    val hash_fun = MessageDigest.getInstance("SHA-1")
    hash_fun.digest(s.getBytes).map{ "%02x".format(_) }.mkString
  }

  def gt_cookie(c: Option[Cookie]) : Int =
    c.map(_.value.split("/")) match {
      case Some(Array(s, h))
        if (s.forall(_.isDigit) && mk_hash(s) == h) => s.toInt
      case _ => 0
    }

  def mk_cookie(i: Int) : Cookie = {
    val s = i.toString
    Cookie("visits", s + "/" + mk_hash(s))
  }

  def index = Action { request => ... }
}
```

- the counter/hash pair is intended to prevent tampering

SHA-1

- SHA-1 is a cryptographic hash function (MD5, SHA-256, SHA-512, ...)
- message $\rightarrow$ digest
- no known attack exists, except brute force

SHA-1

- SHA-1 is a cryptographic hash function (MD5, SHA-256, SHA-512, ...)
- message $\rightarrow$ digest
- no known attack exists, except brute force

- but dictionary attacks are very effective for extracting passwords (later)

```
1   object Application extends Controller {
2
3     val salt = "my secret key"                    should be random
4
5     //SHA-1, SHA-256 + salt
6     def mk_hash(s: String) : String = {
7       val hash_fun = MessageDigest.getInstance("SHA-1")
8       hash_fun.digest((s + salt).getBytes).map{ "%02x".format(_) }.mkStr
9     }
10
11    def gt_cookie(c: Option[Cookie]) : Int =
12      c.map(_.value.split("/")) match {
13        case Some(Array(s, h))
14          if (s.forall(_.isDigit) && mk_hash(s) == h) => s.toInt
15        case _ => 0
16      }
17
18    def mk_cookie(i: Int) : Cookie = {
19      val s = i.toString
20      Cookie("visits", s + "/" + mk_hash(s))
21    }
22
23    def index = Action { request => ... }
24  }
```

Unix Passwords

- passwords are **not** stored in clear text
- instead /etc/shadow **contains**

  name:1QIGCa$/ruJs8AvmrknzKTzM2TYE.:other_info

- $ is separator
- 1 is MD5 (actually SHA-512 is used nowadays, 6)
- QIGCa is salt
- ruJs8AvmrknzKTzM2TYE → password + salt

  (openssl passwd -1 -salt QIGCa pippo)

Password Blunders

- in late 2009, when an SQL injection attack against online games service RockYou.com exposed 32 million plaintext passwords

- 1.3 million Gawker credentials exposed in December 2010 containing unsalted(?) MD5 hashes

- June 6th, 2012, 6 million unsalted SHA-1 passwords were leaked from linkedIn

Web user maintains 25 separate accounts but uses just 6.5 passwords

Brute Forcing Passwords

- How fast can hackers crack SHA-1 passwords?

Brute Forcing Passwords

- How fast can hackers crack SHA-1 passwords?
- The answer is 2 billion attempts per second using a Radeon HD 7970

graphics card
ca. £300

password length	time
5 letters	5 secs
6 letters	500 secs
7 letters	13 hours
8 letters	57 days
9 letters	15 years

5 letters $\approx 100^5 = 10$ billion combinations
(1 letter - upper case, lower case, digits, symbols ≈ 100)

Passwords

How to recover from a breakin?

Passwords

How to recover from a breakin?

- Do not send passwords in plain text.
- Security questions are tricky to get right.
- QQ (Chinese Skype) authenticates you via contacts.
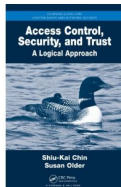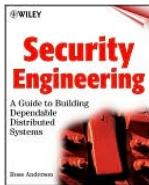
This Course

- break-ins (buffer overflows)
- access control
  (role based, data security / data integrity)
- protocols
  (specification)
- access control logic
- privacy

  Scott McNealy:
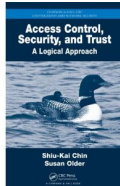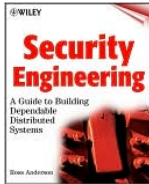 *"You have zero privacy anyway. Get over it."*

Books + Homework

- there is no single book I am following

Books + Homework

- there is no single book I am following

- The question "Is this relevant for the exams" is not appreciated!

  Whatever is in the homework sheets (and is not marked optional) is relevant for the exam. No code needs to be written.

Take-Home Points

- Never store passwords in plain text.

- Always salt your hashes!

- Use an existing algorithm; do not write your own!

Thinking as a Defender

- What are you trying to protect?
- What properties are you trying to enforce?

- Who are the attackers? Capabilities? Motivations?
- What kind of attack are we trying to protect?
- Who can fix any vulnerabilities?

- What are the weaknesses of the system?
- What will successful attacks cost us?
- How likely are the attacks?

Security almost always is **not** free!

The Security Mindset

- How things can go wrong.
- Think outside the box.

The difference between being criminal is to only **think** about how things can go wrong.

Maps in Scala

- map **takes a function, say f, and applies it to every element of the list:**

  List(1, 2, 3, 4, 5, 6, 7, 8, 9)

  List(1, 4, 9, 16, 25, 36, 49, 64, 81)