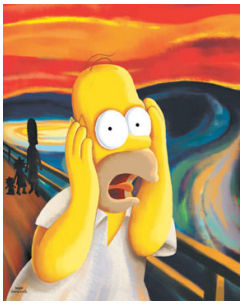
Holub Associates
www.holub.com

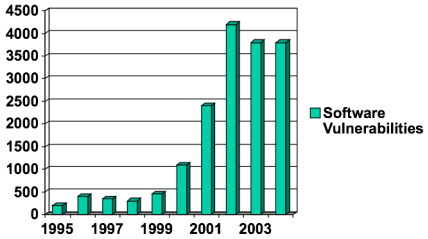
Security 101

An Introduction to Security
Allen I. Holub
www.holub.com

We aren't scared enough



The problem is growing



Why?

- Lauching automated attacks over the network is easy.
- Web Services and SOA applications are particularly vulnerable
  - Legacy applications that aren't secure are exposed to the web.
- Extensible systems (applets, acctiveX) invite attacks.

Security is about risk and liability

- It's essential to associate monetary risk with software risk.
- If the cost of fixing a security breach is higher than the cost of writing off the loss, businesses will take the loss.
- Security is all about lowering risk to a reasonable level, not eliminating risk.

Other meta Issues

- Programmers don't understand security.
- Network guys can't make a program secure.
- If you break the code, how do you use the information without tipping off the other guy that you've broken the code?
- The most secure communication is face to face.
- It's a pain… Even the people who invented private-key encryption rarely use it.

Can you prove it in court?

- Even the best encryption doesn't stand up to a court order.
- A timestamp in a database means nothing unless it's "digitally signed" by a impartial third party.
- Can you trace an entire eBusiness transaction from start to finish and prove that every step was carried out by the right entity?

People are human, not stupid

- Any system that depends on abnormal behavior is insecure. The following behaviors are reasonable:
  - "Hi. This is Fred from IT. Can I have your password so I can check the system?"
  - "I can't remember 50 passwords, so I use the same password everywhere."
    - At one point 80% of the passwords at Berkeley were characters from the Lord of the Rings.
  - "I can't remember long passwords."
  - "I don't have a clue what all that junk in the Security-Options dialog means!"
    - "If I enable security, I can't browse!"
  - "The email came from a friend and got through the virus check, so why can't I click on it?"

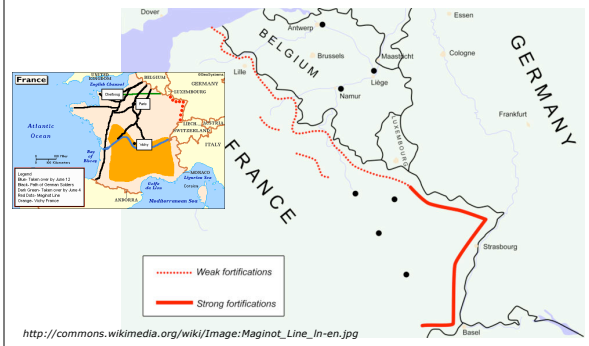
Worry about the right thing!

- Nobody intercepts credit-card transmissions on the internet.
- Lots of people hack into merchant databases and "harvest" credit-card numbers by the thousand.
  - Until recently, VISA did not require credit card numbers to be encrypted.
  - Even now, most merchant databases are still not encrypted, since there's no mandatory audit requirement.
  - There are solutions (e.g. CitiCard single-use numbers)

Perimeter Defenses in Theory

http://www.roxanneardary.com/blog/castles-palaces-chateau/

Perimeter Defenses in Reality



http://commons.wikimedia.org/wiki/Image:Maginot_Line_In-en.jpg

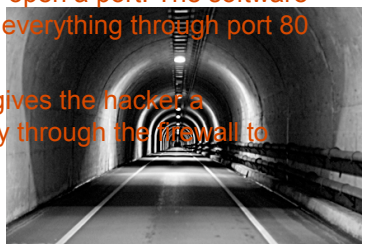
Software (not Application) Security

- The notion of "application security" is misguided.
  - You can't protect vulnerable software.
  - It's easy to pretend you've solved the security problem if you see it as a fortification problem.
    - An application-security testing tool (or consultant) can only rate your software in a scale of:

DEEP TROUBLE to **DON'T KNOW**

Firewalls are ineffective

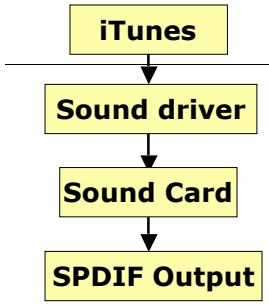
- When the network guys made it impossible to open a port. The software guys just did everything through port 80
- SOA/SOAP gives the hacker a superhighway through the firewall to your server.



Worry About the Right Thing (2)

- Firewalls don't protect against denial of service or bug-based attacks.
- Firewalls have bugs too!
- The attacker might be inside the firewall!
- **A bug in a subroutine in an app server is behind all of the above, and can be accessed through all of them.**

Security must be systemic



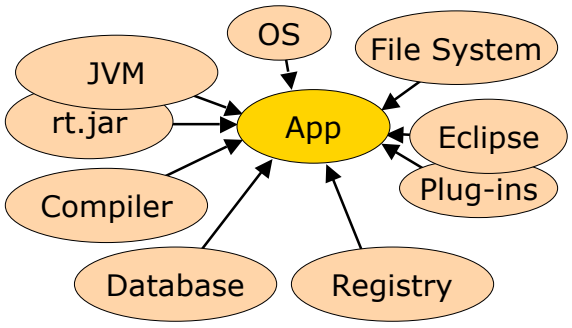
Secure

Not!

Some security comes from limiting sound quality to 128-bits

But here's the real problem

No Program is an Island



Don't rely on information from the Web

- Nothing on the internet is trustworthy.
  - Even if you wrote the web page.
  - Don't believe a "total" price figured in JavaScript.
  - Don't believe data validated in JavaScript.
  - Don't expect your client-side app to never send invalid data.

Consider your compiler

- Did you download it off the web using an insecure connection?
- Did you check the "signature"?
- How do you know that some bad guy hasn't given you a compiler that inserts evil code into your program?
  - Evil code could inject a virus or "bot" into your host operating system, for example.
- Same argument applies to every development tool, plugin, and library that you download.

The language matters

- Language choice is often mythology based.
  - E.g.: C++ is faster. Development is faster in PHP.
- **C and C++ are inherently insecure**.
  - They permit bugs like buffer overflows and dangerous type conversions.
  - The complexity of the language makes it hard to find bugs.
- "Scripting languages" (Ruby, PHP, Python, etc.) are less secure, still.
- Use Java or C#

Hackers exploit bugs and flaws

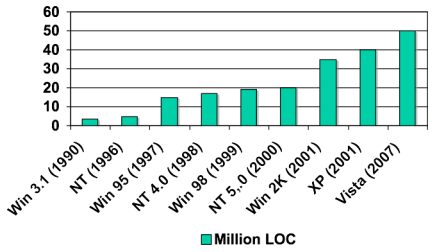
- Attacks that don't exploit human factors exploit
  - bugs (errors in the code) and
  - flaws (errors in the architecture). E.g.: ActiveX, SOAP
- All software has bugs in it.
- Firewalls don't protect against bugs.
- The more popular or pervasive the system, the more people will try to attack it. (e.g. Windows)
  - Monoculture is bad.

A couple simple security bugs

- C/C++
  - ```
    f(){ char x[128];
         x[128] = 0;
         gets( x );
    }
    ```
- Java
  - ```
    String s =
         getPasswordFromUser();
    ```


Bugs vs. code size

- The number of bugs tend to increase with the square of the code size.



http://en.wikipedia.org/wiki/Source_lines_of_code

Guess which is most secure

Red Hat 6.2	17
Red hat 7.1	30
Debian 4.0	283
Open Solaris	9.7
Free BSD	8.8
Mac OS X	86 (kernel size: .79)
Linux Kernel	5.2
Windows Vista	50

SLOC (Million)

Think about static analysis

- Human's just can't see the problem
- Tools analyze your code, looking for potential security-related bugs.
  - Coverity (www.coverty.com)
  - Fortify (www.fortifysoftware.com)
  - Ounce Labs (www.ouncelabs.com)
  - Secure Software (www.securesoftware.com)

"Magic crypto fairy dust"

- **Cryptography does not make a system secure.**

Basic principles

- **Secrecy ≠ Security.**
  - ***Secrecy***: You can't find the safe.
  - ***Security***: You can't open the safe, even if you know how it works.
  - **Secret systems are never secure!**
    - The best way to assure that an encryption algorithm is secure is to have thousands of knowledgeable people try to break it.
- **Security ≠ Technology**
  - Security comes from well-thought-out protocols (in the diplomatic sense).
  - Technology only gives you a means to implement a portion of the protocol.

What Crypto Buys You

- ***Access control***:
  - Only authorized individuals can access the it.
- ***Confidentiality***:
  - Only authorized individuals can read the text.
- ***Authentication***:
  - The writers are who they say they are.
- ***Non-repudiation***:
  - The writers can't claim they didn't write it.
- **Integrity**
  - The document you received is the one I sent.

How long will it take?

- **Not: "is it breakable?" But: "how long will it take to break it?"**
  - **Will the information have value at that time?**
- **Consider a 4-wheel combination lock. How long to try every combination?**
  - **10,000 possibilities (~13 bits), 1 every 2 seconds == 20,000 seconds (~5.5 hours)**
  - **2 people, each trying ½ the codes: 2.750 hours**
  - **4 people, each trying ¼ the codes: 1.375 hours**
  - **10,000 people, each trying 1 code: 2 seconds**

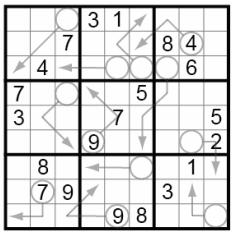
Cost of a Brute-Force Attack

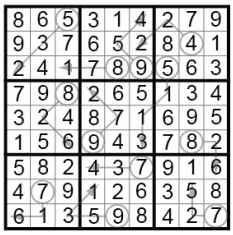
- **Breaking a cipher is a function of:**
  - **number of possible keys (10,000 possibilities = ~13 bits)**
  - **cost of the hardware (number of processors)**
  - **time**
- **Given enough time or enough money, you can crack anything.**
  - **Will the value of the text outlive the time required to break the encryption?**

But… Use HTTPS Everywhere!

- All links on an https:// page must be https:
- A stolen session key let's someone who hasn't logged on access your site.
  - Sessions keys are encrypted when you use https:
  - But you must use it everywhere:
    <img src="images.someFile">

Solving the Problem





Getting business folks to care

- Identify the risks
- Correlate those risks against business goals
- Determine the cost to the business should the risk be realized.
- Use the above to come up with a ranking.
  - H/M/L is sufficient

Making a Business Case

Business Risk	Indicators	Likely -hood	Impact	Cost	Sev erity
Software fails acceptance criteria	Major milestones are missed	H	Company is unable to release product to market.	Revenue loss: $10MM. Marketshare loss: 15%. Damage to brand/reputation: limited.	H
Database corrupted by security breach	Customers report innacurate data. Need to create patches	H	Company will be noncomplient with federal regs. Lawsuits will ensue.	Revenue loss: $8MM. Market share loss: 3%. Damage to brand/reputation: extreme	H

Creating a secure system

- In order of effectiveness:
  1. Code Review
  1. Architectural risk analysis/Design review
  2. Penetration testing
  3. Risk-based security tests
  4. Abuse cases
  5. Security requirements
  6. Security operations

-Gary McGraw

Code Review

- Artifact: the code
- Example risks found: “Buffer overflow on line 32.”

- http://www.gilb.com for code-inspection tips.
- “A Taxonomy of Coding Errors that Affect Security:” http://www.fortify.com/vulncat/
- CERT Secure-programming standards for C/C++: https://www.securecoding.cert.org/

Design Review

- Artifacts: Design documents
  - Problem Statement
  - Use Cases
  - UML
- Examples of risks found: "Failure of web server to authenticate calling code." "Insufficient compartmentalization of modules"

Why do design review?

- IBM determined that the cost weightings for fixing a flaw are:
  - 1 if caught at design time
  - 6.5 if caught in implementation
  - 15 if caught in testing
  - 100 if caught in maintenance.
- HP determined that design review is one of only two "best practices" that both improves quality and reduces development time.
  - (The other was short cycles and regular releases to the users.)

Penetration testing

- If the software can't pass canned tests, then you're in deep trouble.
- If the software passes canned tests, then you still don't know whether it's secure.
- Hiring a hacker usually uncovers susceptibility to only conventional attacks.
- If you don't find the problem until after the system is built, it's too late to fix it in a cost effective way.

Risk-based security testing

- Test cases are based on risk analysis, abuse cases (more below), and known attack patterns.
- Standard test cases also uncover security flaws.

Abuse cases

- Like use cases, but show how the system could be abused by a hacker (or a pathological user).
  - Random input to fields.
  - Extremely long strings.
  - SQL or <script> tags in user input.
  - Direct calls to RPC functions with illegal arguments.

The Mind of a Hacker

- Why would they attack the site?
  - Offended by how bad it is and want to teach you a lesson.
  - For thrills.
  - For the challenge.
  - To impress his friends.
  - To delay large transactions for 15 minutes to do illegal arbitrage or to embezzle the interest.
  - Financial gain: stealing identities, not paying for products.
  - To attack a third party (bogus credit-card charges).

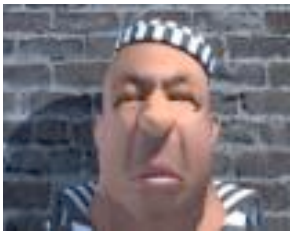
Security Requirements

- Security requirements should be explicitly spelled out in your spec.

Security Operations

- Examine the behavior of the deployed system when under attack.
- Make sure that there's sufficient logging that you can study attacks.

An Example Exploit



SQL Injection

- Consider a simple login screen, with a forgotten-password link.
  - Prompt for an email login
  - Email a password
  - `SELECT someField`
    `FROM someTable`
    `WHERE someField='$EMAIL'`

Test for Vulnerability

- Enter `foo@bar.com'` as an email, yeilding:

```
SELECT someField
FROM someTable
WHERE someField='foo@bar.com''
```

- Will create a SQL error. If the error message isn't "email address unknown," then the site is probably vulnerable.
- Don't ever print the SQL error message!

Prove that it's vulnerable

- Enter `junk' OR 'x'='x` as an email, yielding:

```
SELECT someField
FROM someTable
WHERE someField='junk'
                  OR 'x'='x'
```

- Selects everything from the table!
- Result is:
  - "Login information sent to foo@bar.com"
- Probably the first email in the table.

Guess a few field names

```
SELECT someField
FROM someTable
WHERE someField='x'
          AND email is NULL;--'
```

- Fails if "email" is not a field.
- Keep trying other obvious column names until it works!

Find the Table Name

```
SELECT someField
FROM someTable
WHERE someField='x'
      AND 1=(SELECT COUNT(*)
             FROM tabname);--'
```

- Fails if "tabname" is not the table name.
- Keep trying other obvious table names until it works!

Etc. You can put any SQL in there!

- `DROP tablename`
- `UPDATE tablename`
  - Add yourself!
  - Change a password!
- `xp_cmdshell`
  - In SQLServer, executes an arbitrary OS-level command!

Prepared Statements

- Use "Prepared statements"
  - SQL is precompiled, user input is added later and is not treated as SQL
- Bad:

```
Statement s = connection.createStatement();
ResultSet = s.executeQuery(
    "Select email from table where name="
                                + formField );
```


The Solution

- Good

```
PreparedStatement s =
connection.prepareStatement(
    "select email from table where name=?" );
ps.setString(1, formField);
ResultSet = s.executeQuery();
```


Use Prepared Statements (2)

- Might not work if prepared statements are simulated in the driver.
- Don't do this:

```
Statement s = connection.createStatement();
ResultSet = s.executeQuery(
    "Select email from table where name="
                                + formField );
```

Other Precautions

- Verify user input as safe.
  - Use white-list testing (approve only valid characters as compared to rejecting invalid ones).
- Limit database permissions
  - Login has read-only permission on table
- Assume that the bad guy can get full adminstrator access to machine!
- Limit information in error reports
  - Do not show output from database server!

Cross-Site Scripting

- What if a hacker enters a user name as

```
John<script>document.location.replace(
   http://hacker.heaven.com)</script>
```

- The web site prints "Hello John", while redirecting to another site.
- Can print session ids, cookies, etc.
- Can be inserted by "social" attacks, SQL injection, man-in-the-middle attacks, etc.

Read Further

- Ross Anderson, *Security Engineering*. (ISBN: 0-471-38922-6. [*2nd Ed. ISBN* 0-470-06852-3, projected release 4/08].)
- Gary McGraw, *Software Security*. (ISBN: 0-321-35670-5)
- Hoglund & McGraw, *Exploiting Software*. (ISBN: 0-201-78695-8)
- Howard & LeBlank, *Writing Secure Code, 2nd Ed.* (ISBN:0-735-61588-8)

Q&A

Allen Holub
www.holub.com

Slides at http://www.holub.com/
publications/notes_and_slides