Computer Insecurity

Steven M. Bellovin
smb@research.att.com
http://www.research.att.com/~smb

The Internet Worm-- 10 Years After

- Worm launched on 10/2/1988.
- First time most people ever heard of the Internet.
- First time most people ever heard of computer hackers…
- Have we learned anything?

How the Worm Spread

- Back door in `sendmail`.
- Password-guessing.
- Transitive trust.
- Buffer overflow in fingerd.

Those problems are still with us -- and now we have mobile code, too.

Back Doors

- Programmers still install extra features (but today we call them "Easter Eggs").
- Administering an organization of computers requires tools to distribute updates. Are these secure?
- Many programs auto-update. Is this mechanism secure?

Passwords

- Users still pick bad passwords.
- System designers still use passwords, despite guessing and “sniffing”.
- One-time password schemes are rare.
- Hardware tokens are even rarer.

Transitive Trust

- If A trusts B and B trusts C, A trusts C.
  - Often, A doesn’t know this.
- Such relationships are often bidirectional.
- The security of a trust graph is thus equal to the security of the weakest link.
- Today, we have a new name for transitive trust: “single sign-on”.

Buffer Overflow

- The single biggest cause of new security holes.
- 9 out of 12 CERT advisories this year describe buffer overflows.
- Many more such holes are reported on various mailing lists.

Why Does This Happen?

- C makes it hard to handle strings well.
  - C++ makes it much easier, but too many people write in C, albeit with a C++ compiler.
- Ordinary prudent code isn’t safe against attack; MAXPATHLEN is an OS limitation, and is not binding on hackers.
- Time pressure.

Time to Market Wins

- Every study shows that life-cycle costs are reduced by good development and testing practices.
- But products no longer have traditional life cycles; instead, each new release has so many more features that it’s a new program.
- Users do the testing; reliability and security don’t build market share.

Mobile Code

- Many forms: Java, Javascript, Word, email.
- The OS no longer helps; protection is up to the application.
  - We have WebOS, WordOS, MailOS, etc.
- But the applications in question are too big, too complex, and too poorly written to do the job.

Why Use Mobile Code?

- Most Web uses are frivolous.
  - Some, such as input validation or logins, are down-right wrong.
- But what about shared documents?
- Many things not considered to be mobile code are complex enough to be treated as such: html, mail-handling scripts, etc.

What Can we Do?

- We need better underlying operating systems.
  - On a PC, the user owns the machine, and traditional OS vs. user boundaries don’t apply.
- Until we get better OS’s (and the tools to manage them), structure applications as operating systems.
  - Why doesn’t the Java VM have system calls?

```
Date: Thu, 15 Oct 1998 07:36:05 -0400 (EDT)
From: security@research.att.com
To: smb@research.att.com
Subject: tcpsuck port 80

TCP message from host universe.campus.luth.se (130.240.193.207): port
3294

Read timeout
32 bytes received
   0:   47455420 2f636769 2d62696e 2f706866   GET /cgi-bin/phf
  16:   0a000000 000000e0 b89b0740 50930408   ...........@P...
```

Vulnerable Clients

- Traditionally, attacks have been against servers.
- Servers are better administered, and less vulnerable.
- But “always-on” PCs are very soft targets indeed.
- *Using @Home, I see a hostile probe every couple of days...*

Where’s the Firewall?

- Employee machines are increasingly outside the firewall.
  - Even if they dial in directly, people surf the Web on their own time.
- There are many more extranet connections to customers, vendors, joint venture partners, outsourced service suppliers, etc.
- We’re losing the guard at the front door.

Cryptography

- Cryptography solves many security problems -- eavesdropping, spoofing, etc.
- But it is used too rarely, and even more rarely used well.
- Nor does cryptography solve the buggy code problem.
  - 2 of the 12 CERT advisories this year were about cryptographic problems.

Why Isn’t Crypto Used?

- It adds complexity, and users -- paying customers -- haven’t demanded it.
- The export rules make it hard.
- If an operating system is insecure, can it even protect a cryptographic key?

Cryptography is Hard

- Proper use of cryptography requires fairly deep knowledge.
- Even the experts often can’t get cryptographic protocols right.
- There’s a lot of snake oil out there.

Going Around Security

- Computer systems don’t exist in a vacuum:
  - Attack the surrounding systems.
  - “Dumpster diving”.
  - Social engineering.
- System must be usable by real people -- how do you recover from lost keys, forgotten passwords, etc.
- Must *bound* or *relocate* insecurity -- it can’t be eliminated.

Bounding Insecurity

- What is the likelihood of a security flaw?
- What might the flaw cost you?
- What will it cost you to close the hole? What will it cost you to close the hole later, after the system is deployed?

Being honest about flaws is easy. Being humble about architectures and code is hard. Remember that complexity is the enemy.

Moving Insecurity

- Use layered security -- protect a weak point with some other mechanism.
- Example: cryptography protects a link, but relies on the security of keys. These are (usually) easier to safeguard.
- Example: firewalls can be attacked, too, but they’re (often) running simpler, cleaner code.

The Systems Perspective

- No one mechanism will buy us security.
- Security has to be built in from the beginning.
- There is no “security pixie dust” that we can sprinkle over existing designs.

Building Secure Systems

- Keep it simple!
- Never rely on obscurity.
- Validate all input.
- Use appropriate defenses, including cryptography, at all points. Any component can be attacked.
- Cater to the real world.
- Keep it simple!