Misplaced Trust: Kerberos Version 4 Session Keys

Bryn Dole, Sun Microsystems

Steve Lodin, Delco Electronics

Gene Spafford, Purdue University

Kerberos Version 4 Vulnerability

- An implementation problem
- Random keys had only 20 bits of entropy.
- Keys could be guessed in seconds.
- Pre-computing the keys allowed “guessing” in microseconds.
- Result: The security of Kerberos Version 4 was compromised.

What Went Wrong?

- Underestimated the challenges of RNGs
- The repaired RNG never got called.
- Code review failed to detect that the old RNG was still in use.

Software Engineering Breakdown

- Breakdown in process
  - Owner of code was ineffective in getting code reviewed.
  - Fix occurred during migration to Version 5.
  - Multiple code trees compounded the problem.
  - No regression testing

Trusting Software

- What types of systems do we trust?
  - Open systems, with public source code
  - Older, mature systems
  - Systems based on secure protocols and standards
  - Designed by smart people
- Kerberos had them all.

Why Trust Open System Design?

- Security through obscurity does not work.
  - Anything can be reverse engineered.
- Openness provides the means for public scrutiny.
- If you want to make sure software works as advertised, check it out yourself.

Faults of Open System Design

- Open design is no guarantee of security.
  - There is no assurance that experts will examine the code.
  - No structured code reviews.
  - How much time would you spend looking at someone else’s spaghetti code, if you weren’t getting paid for it?

Mature Software

- Software engineering experience tells us that older software does not guarantee the absence of serious bugs.
  - new features add new bugs
  - bug fixes add new bugs
  - maintaining legacy code is difficult
  - newer releases may halt work on older versions

Trusting Secure Protocols

- Have to be implemented correctly.
- The Needham-Schroeder exchange used by Kerberos is ***provably*** secure.
- Must use protocols for what they were designed.
  - Example: SSL for authentication

Secure Algorithms

- Algorithms such as DES, IDEA, MD5, SHA, etc.
  - All benefit from being open standards
  - Increases trust
- They must be used correctly to ensure security.

Conclusions

- The importance of Random Numbers should not be underestimated.
  - They are an essential building block that all security protocols depend on.
- Need secure RNGs built into operating systems and hardware.

Conclusions

- Open design is an valuable mechanism for discovering bugs and security flaws, but…
- Publicly available code is no substitute for:
  - Structured code reviews
  - Good software engineering practices
  - Quality testing