Lesson 8.4: Sowing Seeds

SECURITY VULNERABILITIES IN C/C++ PROGRAMMING

Sowing Seeds

Matt Bishop, Ph.D.
Professor of Computer Science,
UC Davis

UCDAVIS
Continuing and Professional Education

Slide 1: Seeding a Generator

Seeding a Generator

Conceptually: pick something different each time

Problem: these are often predictable

- Either exactly or to some narrow range

Netscape's SSL: used a combination of time of day, process PID, parent process PID

- Produced a range that could be easily searched — which 2 UC Berkeley graduate students promptly did

Slide 2: Common Seeding Errors

Common Seeding Errors

Seed from a very limited space

- Example: if your seed is 8 bits, there are only 256 possible initial states for the generator
  - Doesn't matter how big the internal state is

Using current time, or its hash

- If I know about when the generator was seeded, I can usually figure out the seed
  - Note: most clocks have resolution of 1/60 sec or less, even if their system calls return milliseconds!

Slide 3: Common Seeding Errors

Common Seeding Errors

Disclose the seed

- Example: use time of day as seed, and put that in the cleartext header

Slide 4: Some Fallacies

Some Fallacies

Complex manipulation

- If the adversary can use the algorithm and there is not enough unpredictability in the seed, this doesn't help
- The manipulation itself may look good but prove to be very poor

Random selection from a large database

- If the adversary can find the selection, she can reproduce the stream

Slide 5: Picking a Good Seed

Picking a Good Seed

Use a source of physical randomness

- Be sure it really is random!
- Example: audio device on workstation, or radioactivity

Use events on the computer that are (seemingly) random, such as the length of time between keystrokes

Gather system data that is constantly changing (such as the output from *ps gaux*, process ID, and so forth) and hash it using a cryptographic hash

Slide 6: Devices Generating Random Data

Devices Generating Random Data

Combine a cryptographically secure random number generator with information from many random sources

/dev/random

- Returns random bytes until it exhausts the bytes it produced, then blocks until it produces more

Slide 7: Devices Generating Random Data

Devices Generating Random Data

/dev/urandom

- Returns random bytes, but when it exhausts the bytes it produced, it returns pseudorandom ones
- Never blocks, but may not be suitable for cryptographic use