



TEMP files and RANDOM

Temporary Files

Problem Statement

Survey of UNIX functions

Randomness

Need

Types of random numbers

Devices

1



Temporary Files

- Space for temporary files is found in directories such as /tmp, /var/tmp or C:\TEMP, where everyone can write
- Space may be purged regularly (e.g., "every night, files older than 5 days are deleted") and during reboot
- Space used by many UNIX or Windows utilities, installers and programs
- UNIX systems are often configured so that this space is not counted as part of user quota
 - Allow large, temporary jobs

2



Temporary Files Issues

- Need an unpredictable name to avoid a collision between links and your files or directories
- There is a race condition between testing if a file exists and creating it
- Need correct permissions
- There is a race condition between creating and setting permissions
- Need OS support!

3



Name Collisions Attacks

- What if the name of your temporary file (lock file or other) in /tmp is constant or predictable?
 - Your program using a lock file may never run or do what it is supposed to
 - It is easy to make a symlink pointing to a sensitive file
- Symlink attacks are easier if the name of the temporary file is predictable

4



How Not to Choose a Random Name

- Use the process ID
- Use the user ID
- Use the time of day
- Use a counter
- Use a bad random number generator
- etc...

5



OS Support for Temp Files

- The following take a filename "template" as input
 - mktemp - generate temporary file name (unique)
 - mkstemp - also creates the file but also opens it
 - mkstemp - generate temporary file name with suffix
 - mkdtemp - create a directory
- Overwrite part of a template to create a unique name
- Some of these functions used to create names using parts of the date or process ID, etc... and were insecure

6



mktemp(1)(3)

- Section (1): command line (shell scripts)
 - BSD/MacOS X:
 - creates file with mode 0600
unique name
- Section (3): C programs
 - Race condition between getting the name and creating the file!
 - The program must use "open" with the O_CREAT | O_EXCL flags, and loop until the file is successfully created, or use a different function

7



int mkstemp(char *template)

- Creates name
- No race condition!
- Recommended function
- *mkstemp()* replaces contents of string pointed to by *template* by unique filename, and return a file descriptor for the file open for reading and writing.
- string in *template* like filename with six trailing 'X' s; *mkstemp()* replaces each 'X' with a character from the portable filename character set. Characters chosen not to duplicate existing file
- Returns -1 if no suitable file has been created

8



Need for Random Numbers

- Unique file or directory names
- Session IDs that carry proof of authentication (nonces), passwords
- Games (data, behavior, opponent generation, character generation)
- Encryption (session keys)
- Cryptographic protocols (zero-knowledge)
- http cookies


9



What is "Random"?

- *Sequence of cryptographically random numbers*
a sequence of numbers n_1, n_2, \dots such that for any integer $k > 0$, an observer cannot predict n_k even if all of n_1, \dots, n_{k-1} are known
 - Best: physical source of randomness
 - Electromagnetic phenomena
 - Characteristics of computing environment such as disk latency
 - Ambient background noise

10



How Random Numbers Are Generated

- Linear Congruential Generators
 - Simple way to generate pseudo-random numbers
 - Easily cracked
 - Produce finite sequences of numbers
 - Each number is tied to the others
 - Some sequences of numbers will never be generated
- Cryptographic random number generators
- Entropy sensors (i.e., extracted randomness)

11



What is "Pseudorandom"?

- *Sequence of cryptographically pseudorandom numbers*: sequence of numbers intended to simulate a sequence of cryptographically random numbers but generated by an algorithm

Very difficult to do this well

- Linear congruential generators [$n_k = (an_{k-1} + b) \bmod n$] broken
- Polynomial generators [$n_k = (a_j n_{k-1}^j + \dots + a_1 n_{k-1} + a_0) \bmod n$] broken too
- Here, "broken" means next number in sequence can be determined

12



To break a LCPNG (1)

```
#include <stdio.h>
#include <stdlib.h>
static unsigned int X[ ]={1,254,130,141,225,2,28,63,120}

int gcd(int m, int n) {
    int r = m % n ;
    while (r != 0) {
        m = n ;
        n = r ;
        r = m % n ;
    }
    return n;
}
```

13



To break a LCPNG (2)

```
int main() {
    int i;
    int last_gcd = -1;
    for (i = 1; i < 6; i++) {
        int x1 = X[i] - X[0];
        int x2 = X[i+2] - X[1];
        int x3 = X[i+1] - X[0];
        int x4 = X[i+1] - X[1];
        int d = abs(x1*x2 - x3*x4);
        if (last_gcd == -1)
            last_gcd = d;
        else
            last_gcd = gcd(last_gcd, d);
        printf ("%d & %d & %d & %d & %d & %d & %d\n",
            i, x1, x2, x3, x4, d, last_gcd);
    }
    return 0;
}
```

14



To break a LCPNG (3)

With input $X[] = \{1, 254, 130, 141, 225, 2, 28, 63, 120\}$

	$X_i - X_0$	$X_{i+2} - X_1$	$X_{i+1} - X_0$	$X_{i+1} - X_0$	$d(i, j)$	gcd
1	253	-113	129	-124	12593	12593
2	129	-29	140	-113	12079	257
3	140	-252	224	-29	28784	257
4	224	-226	1	-252	50372	257
5	1	-191	27	-226	5911	257

So $n=257$

Then use Extended Euclidean Algorithm to find x and y so that $x \cdot (X_1 - X_0) + y \cdot m = 1$. Then $a = x \cdot (X_2 - X_1)$

Finally, $b = X_1 - a \cdot X_0 \pmod{257}$

15



Seeded Random Number Generators

- Pseudo-random generators depend solely on a seed, which determines the entire sequence of numbers returned
- How random is the seed?
 - Process ID, UserID: Bad Idea
 - Current time: if you are running NTP (Network Time Protocol) all systems are synchronized up to some precision. If you use the time, can guess which seed used (microsecond part might be difficult to guess, but is limited)

16



Roll Your Own Generator?

- What matters is not only the average and the variance of the numbers generated
- All sequences of numbers must be possible
- LCGs travel definite, limited "paths" through the universe of possible sequences
- Need to incorporate entropy as it becomes available
- Need to avoid betraying the internal state of the generator ...
- It is difficult to do correctly

17



Which Generator to use?

- Read description, **avoid** Linear Congruential Generators such as these:
 - "C" rand(3)
 - rand (Windows)
 - Perl rand
 - C# Random
 - PHP rand

18



Good Generators

- Hardware-based
 - Noise
- Cryptographical quality software, entropy-seeded
 - Fast, secure
- Pure Entropy
 - Random timing of events
 - Packets
 - Mouse movement, clicks
 - Keyboard
 - Slow

19



Linux/UNIX Devices

- /dev/random:
 - MacOS X: same as urandom
 - Linux: this is a blocking call that returns only when sufficient entropy has been captured
 - Good for seeding pseudo-random number generators
- /dev/urandom:
 - Implements a fairly complex algorithm that varies between "random" and a well-seeded LCG depending on the availability of entropy
 - Non-blocking call

20