Secure Programming and Code-Reviews

-

..by learning from other People's Mistakes

Thomas Biege <thomas@suse.de>

What will I talk about?

- Application Design
- Programming Errors
- Cryptography and Randomness
- Secure Socket Layer (SSL aka TLS)

Application Design

A SetUID Application

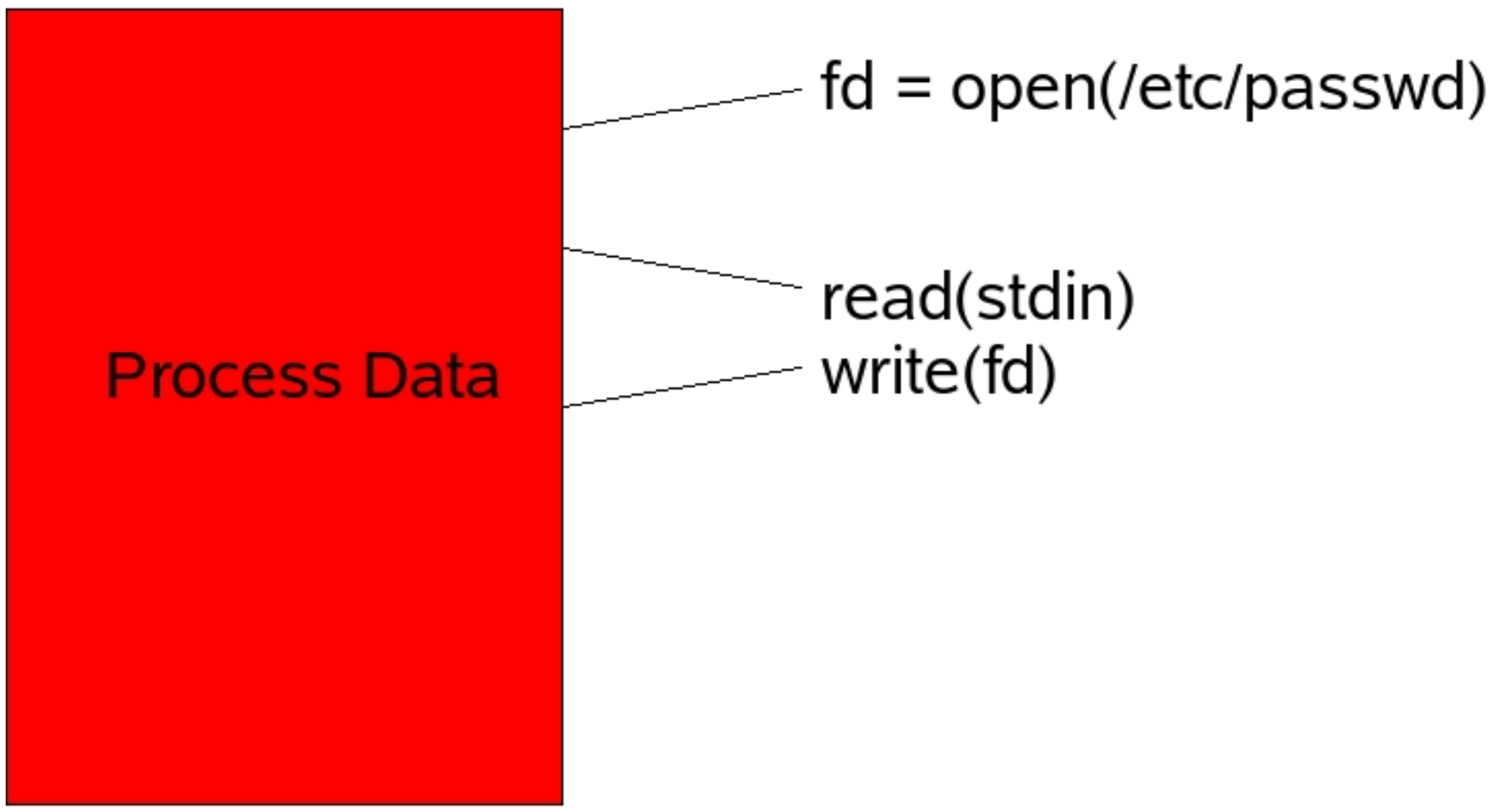
A Content Management System

A SetUID Application

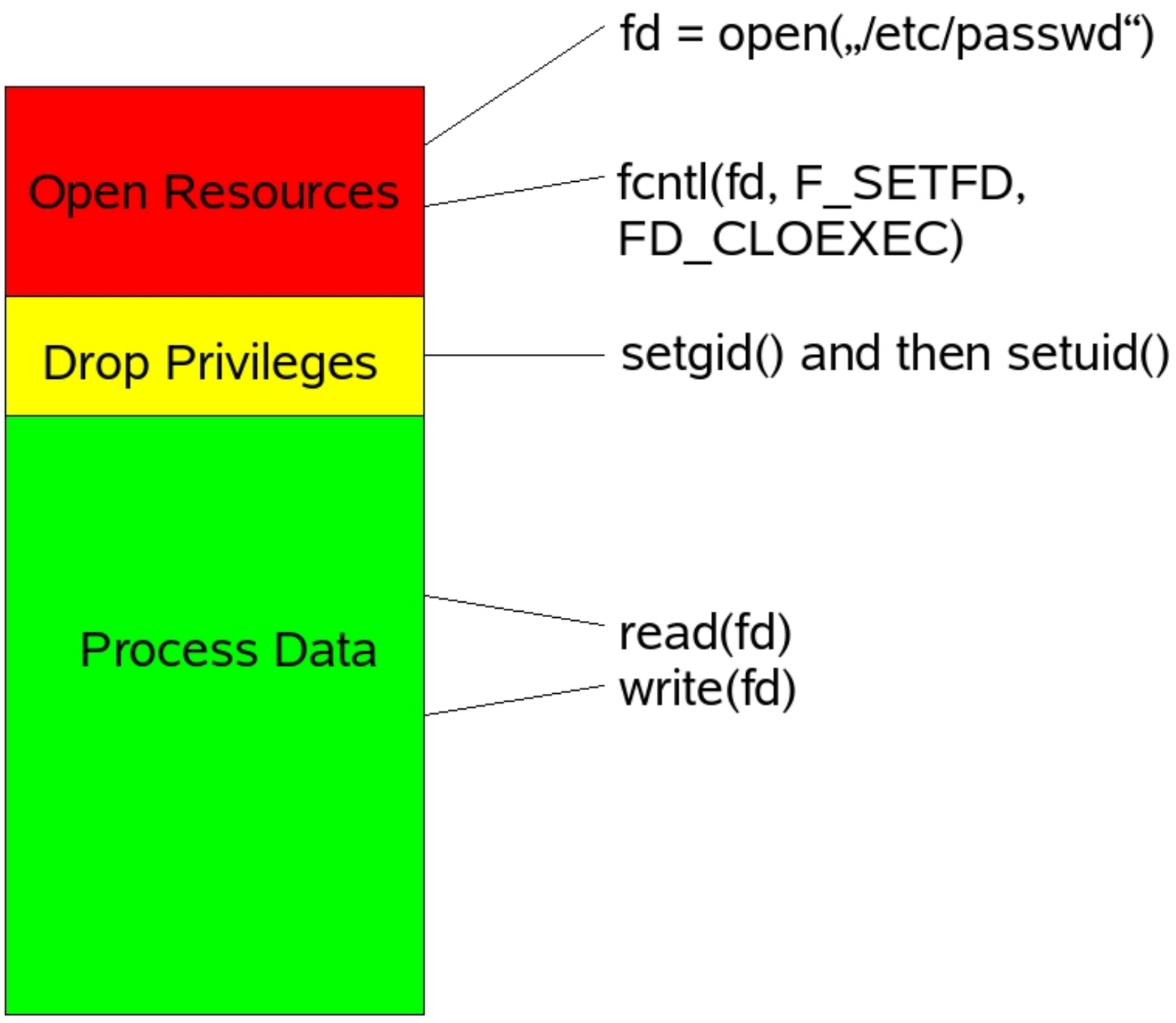
Rules

- Just don’t write it. Your Design is broken...
- … and your Code will be broken too. ;-)
- drop Privileges as early as possible
- KISS
- Code Review... No, not by you.

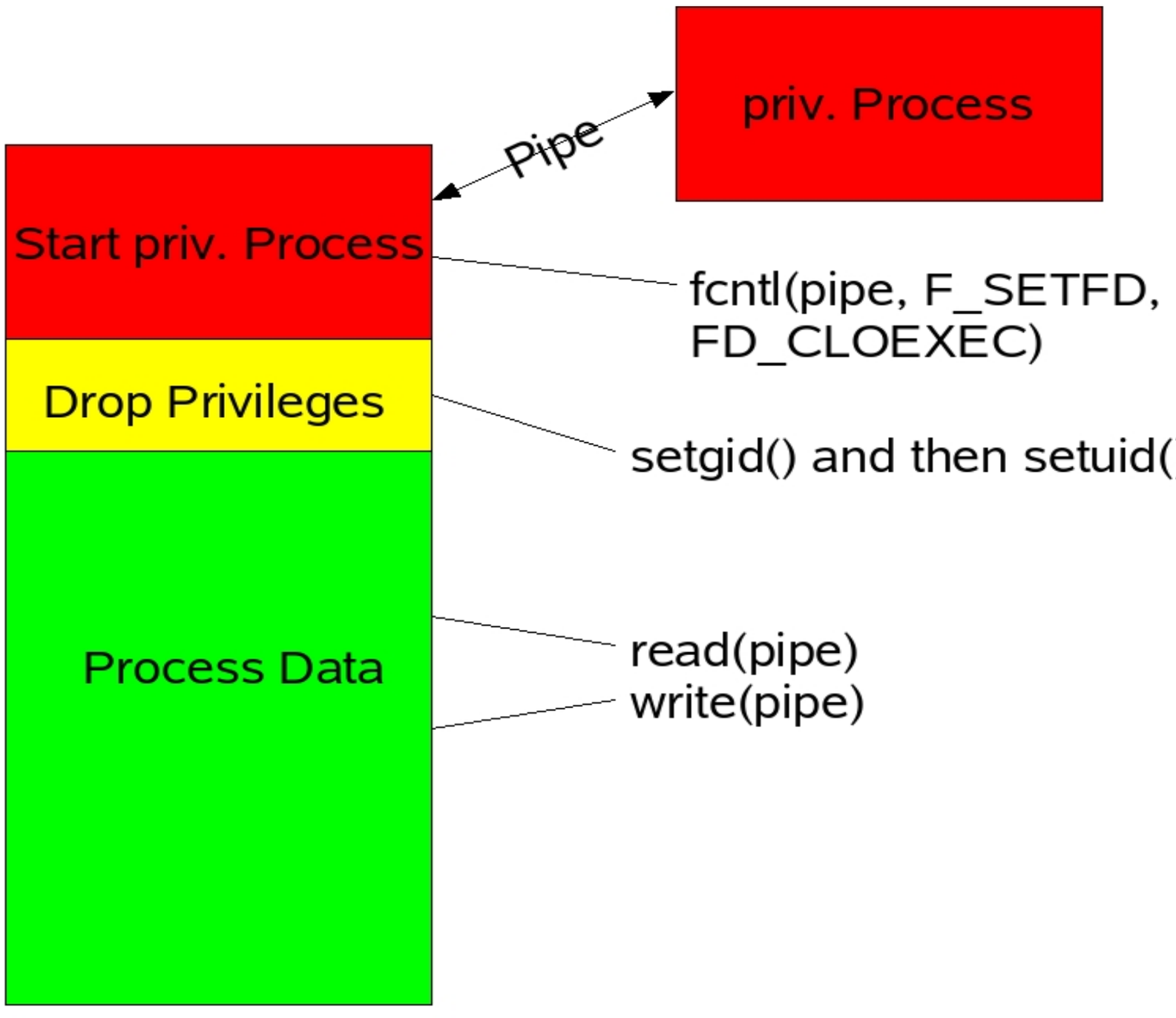
The Bad Idea



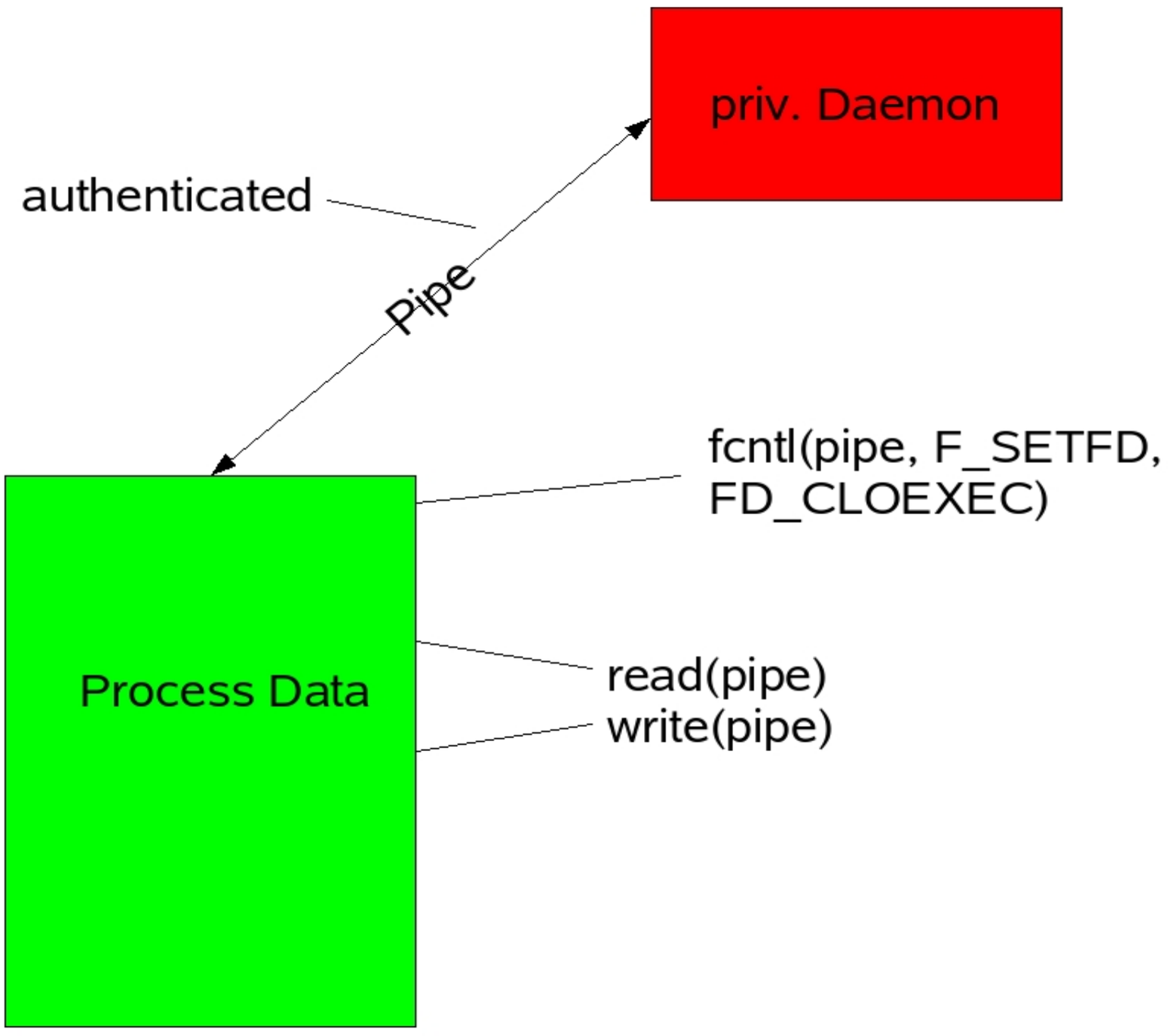
Drop Privileges



Fork a Process



Daemon

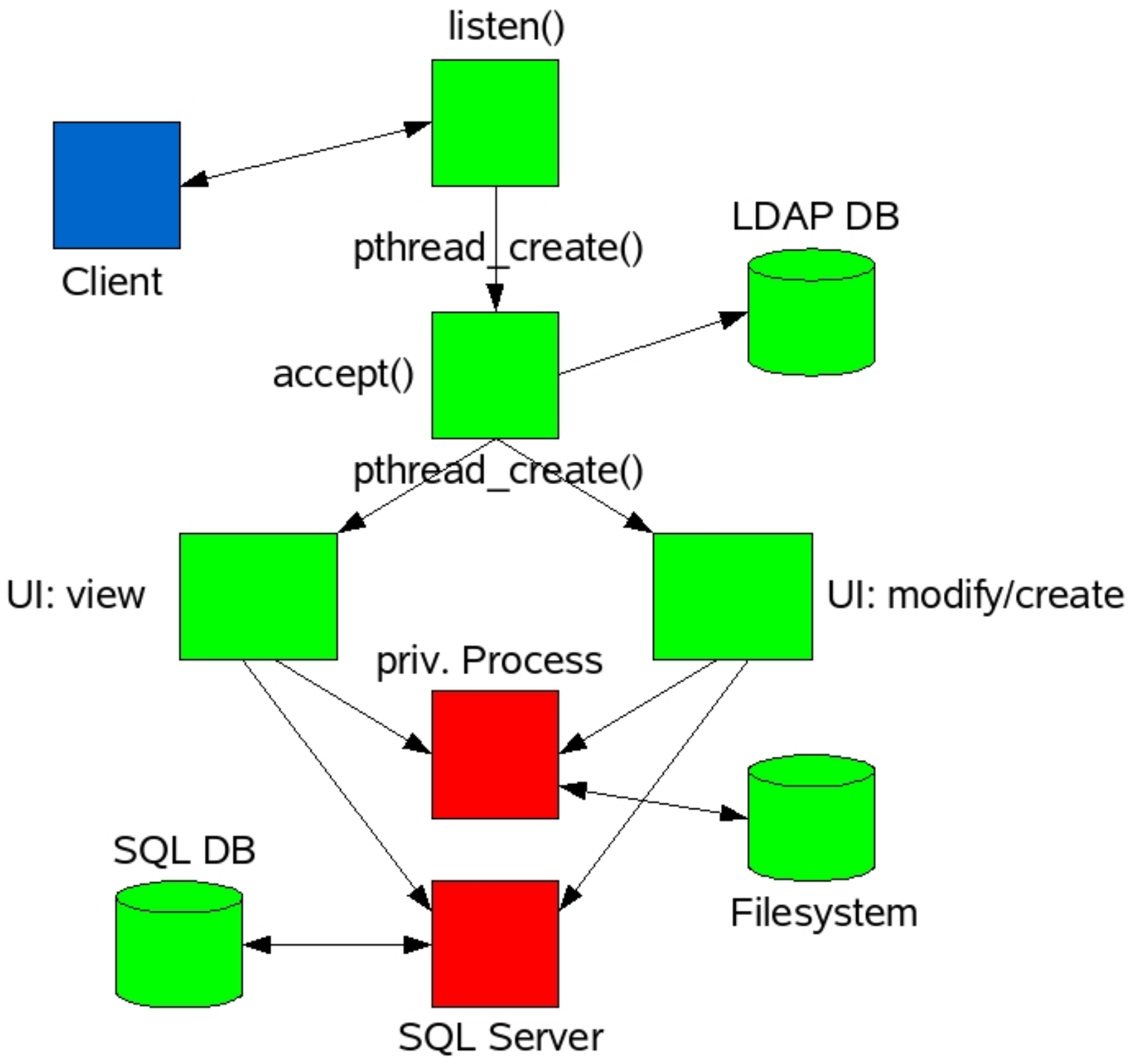

priv. Daemon
authenticated
Pipe
fcntl(pipe, F_SETFD,
FD_CLOEXEC)
Process Data
read(pipe)
write(pipe)

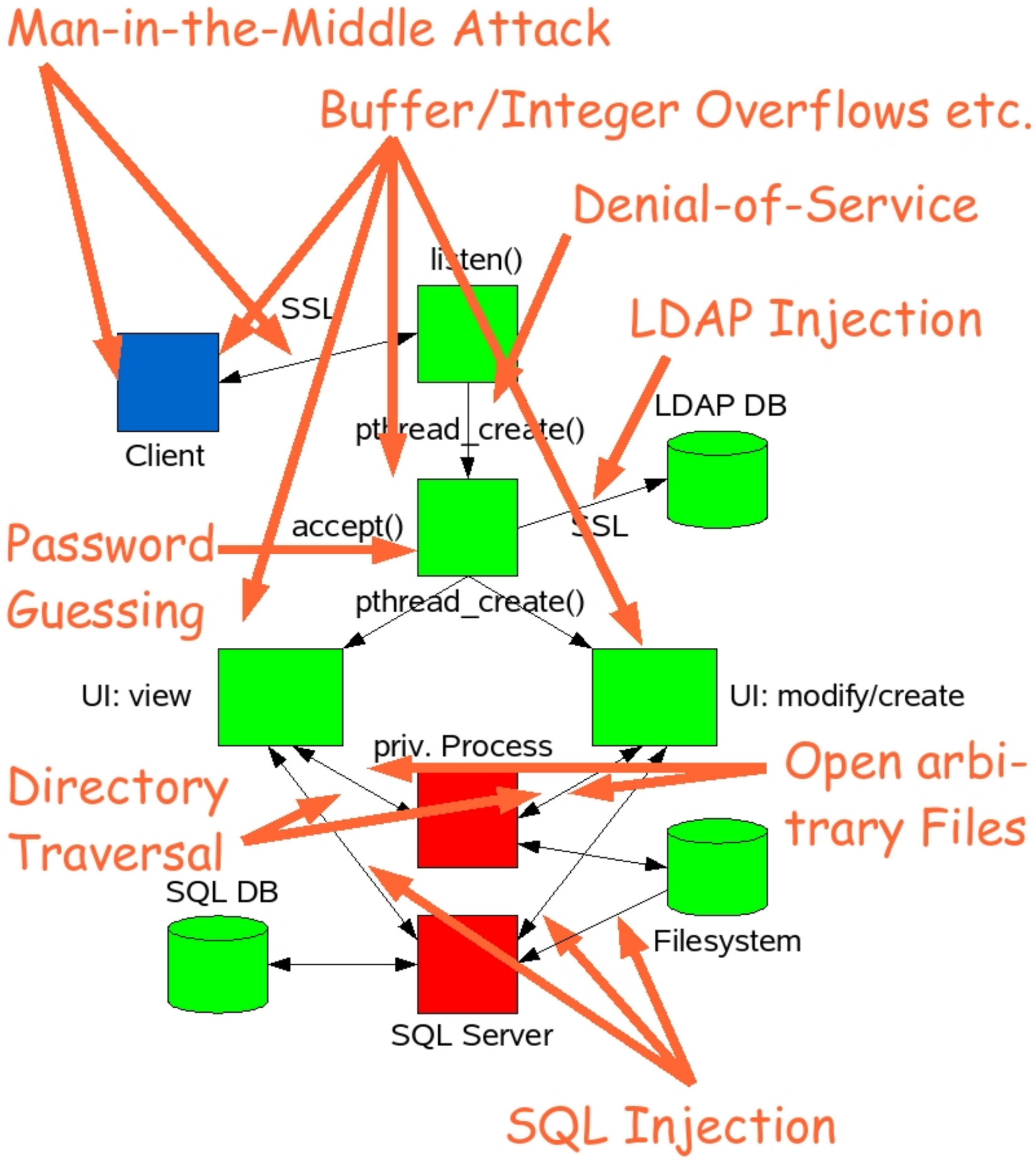
A Content Management System

Requirements

- remote Network Connections (TCP)
- a secure Channel (SSL)
- parallel Sessions
- Authentication (LDAP)
- create, open, edit HTML Files
- upload binary Files into a SQL Database
- view Documents (merge Files)

Overview


listen()
Client
pthread_create()
LDAP DB
accept()
pthread_create()
UI: view
UI: modify/create
priv. Process
SQL DB
Filesystem
SQL Server


Man-in-the-Middle Attack
Buffer/Integer Overflows etc.
Denial-of-Service
LDAP Injection
Password Guessing
Directory Traversal
Open arbi-trary Files
SQL Injection
listen()
SSL
Client
pthread_create()
LDAP DB
accept()
SSL
pthread_create()
UI: view
UI: modify/create
priv. Process
SQL DB
Filesystem
SQL Server

92

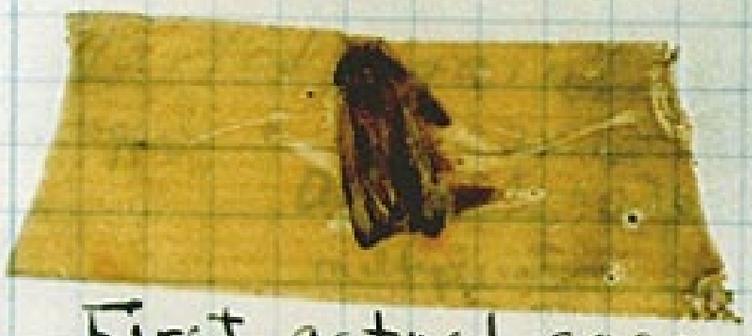
9/9

0800 Arctan started
1000 " stopped - arctan ✓ { 1.2700 9.037 847 025
9.037 846 795 correct
13 wc (032) MP - MC ~~1.9821[illegible]000~~ ~~2.13047[illegible]15 (-3)~~ 4.61592505 9(-2)
(033) PRO 2 2.130476415
correct 2.130676415

Programming Errors

Relays changed

1100 Started Cosine Tape (Sine check)
1525 Started Multy Adder Test.

1545 

Relay #70 Panel F
(moth) in relay.

First actual case of bug being found.

~~1700~~ 1630 Arctangent started.
1700 closed down.

Buffer **O**verflows

modular **A**rithmetic

Race **C**onditions

Format **S**tring **B**ugs

Keyword **I**njection

Buffer Overflows

ziptool - *strncat(3)*

```
switch(oper)
{
    case MOUNT_DISK:
        strncpy(cmd, MOUNT_CMD, strlen(MOUNT_CMD)+1);
        strncat(cmd, " -t ext2 ", 9);
        strncat(cmd, dev, strlen(dev)+1);
        strncat(cmd, " ", 1);
        ...
```

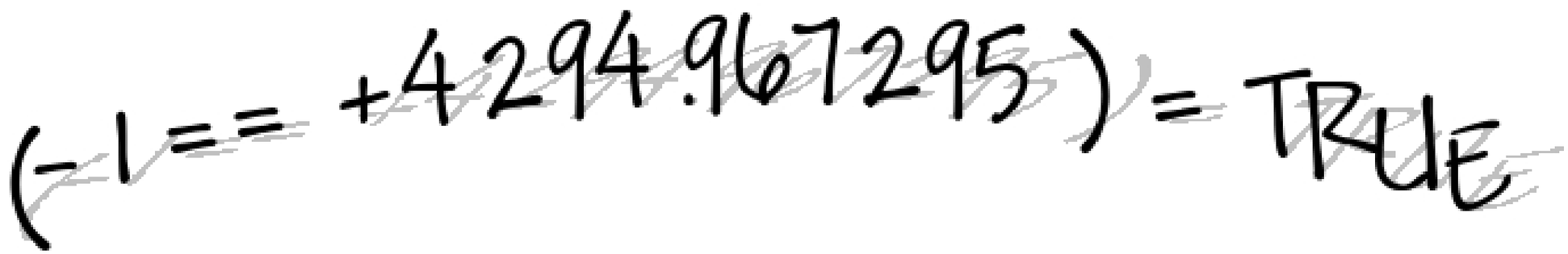
What to avoid?

- blindly using `strcpy(), strcat(), sprintf()`
- using `strncpy(), strncat(), snprintf()` with *wrong* Parameters
- Loops to copy Arrays w/o Bounds-Checking
- writing Functions with Array Parameter but w/o Array-Size Parameter
- trusting Input... be paranoid!

Integer Overflows

Type Mismatch

Signed Issues



unsigned

1111 1111

+ 1 →

unsigned

0000 0000

---

signed

1000 0001

type cast →

unsigned

1000 0001

-32.767

32.767

---

signed

1000 0001

-32.767

type cast →

unsigned

1111 1111 1000 0001

4.294.934.529

smalltalk: Type Mismatch

```
memset (crgb, 0, 256 * sizeof (unsigned int *));
for (a = 0; a < ncolors; a++)
{
    char1 = colorTable[a].string[0];
    char1 = (unsigned char)colorTable[a].string[0];
    if (crgb[char1] == NULL)
    {
        crgb[char1] = (unsigned int *) ...
```

X11: Signed Issue?

```
unsigned int i;
int i;

for (i = nbytes; --i >=0;)
      *dst++ = *src++;
```

xpdf: 32 Bit vs. 64 Bit

```
size_t size;
[...]
if(size*sizeof(XRefEntry)/sizeof(XRefEntry) != size)
{
  error(-1, "Invalid 'size' inside xref table.");
  ok = gFalse;
  errCode = errDamaged;  #define SIZEOF_MYDATA 100
}

entries = (XRefEntry *)gmalloc(size * sizeof(XRefEntry));

for (i = 0; i < size; ++i)
{
     entries[i].offset = 0xffffffff;
     entries[i].used = gFalse;
}
[...]
```

Samba: Valgrind is evil ;-)

```
void *malloc_array(size_t el_size, unsigned int count)
{
if (count >= MAX_ALLOC_SIZE/el_size) {
      return NULL;
}

#if defined(PARANOID_MALLOC_CHECKER)
      return malloc_(el_size*count);
#else
      return malloc(el_size*count);
#endif
```

Samba: Valgrind is evil ;-)

```
void *malloc_(size_t size)
{
	#undef malloc

	/* If we don't add an amount here the glibc memset
	seems to write one byte over. */

	return malloc(size+16);

	#define malloc(s) __ERROR_DONT_USE_MALLOC_DIRECTLY
}
```

What to take care off?

- Test for upper/lower Limits and Overflows
- Take care: Overflows are **undefined for signed Integers**
- Be type-safe!
- does not only affect C!

Test

```
size_t amount; // size_t is unsigned and
               // portable between different
               // architectures

if( amount * sizeof(struct hdr) / sizeof(struct hdr) != amount )
{
        log("integer overflow occurred with 'amount'!");
        exit(-1);
}

hdr_ptr = (struct hdr *) malloc(amount * sizeof(struct hdr));
if(hdr_ptr == NULL)
{
        log("unable to allocate memory for hdr!");
        exit(-2);
}
```

Race Conditions

Where?

- Filesystem
- multithreaded Code
- distributed Databases
- Signals and Interval Timer (Process)
- etc.

RC Server (multi-threaded)

```
const char *
rcd_prefs_get_mid(void)
{
    static char *mid = NULL;
    RCBuffer *buf;

    g_free (mid);
    mid = NULL;

    buf = rc_buffer_map_file (SYSCONFDIR "/mcookie");
    if (!buf)
        return NULL;

    mid = g_strndup (buf->data, 36);
    mid[36] = '\0';

    rc_buffer_unmap_file (buf);

    return mid;
}
```

RC Server (multi-threaded)

- *Double Free* Bug due to missing Locking in multi-threaded Situation
  - Result: Crash!
- Lessons learned:
  - use Locking (mutex)
  - do not use static Buffers

CASA’s Unix Domain Socket

CASA's Unix Domain Socket
The Attack

- **Attacker:** `ln -s /etc/passwd /tmp/.novellCASA`
- **CASA:** calls `UnixEndPoint(/tmp/.novellCASA)`
- **CASA:** `UnixFileInfo()` is a Wrapper for `stat(2)` and follows symbolic Links: `stat(`*`/etc/passwd`*`)`
- **CASA:** `/etc/passwd` is root-owned and therefore the Check for the Ownership succeeds

CASA’s Unix Domain Socket The Attack

- **Attacker:** removes the Link, creates a real Socket, and attaches his Daemon
- **CASA:** connects to this Socket (by Name)
- **Attacker:** steals every Secret from the Client

How to avoid them

- use Filedescriptors not Filenames
- `O_NOFOLLOW, O_CREAT|O_EXCL`
- `setgid(2), initgroups(3), setuid(2)` (Order!)
- `fork(2)` + drop Privileges
- `mkstemp(3)`
- `mktemp(1)`
- lock shared Resources while using them

Format

Bugs

- published in Year 2000
- *printf*-like Functions
- read: **`%x, %p`**
- write: **`%n`** (does not work with current *glibc*)

wrong

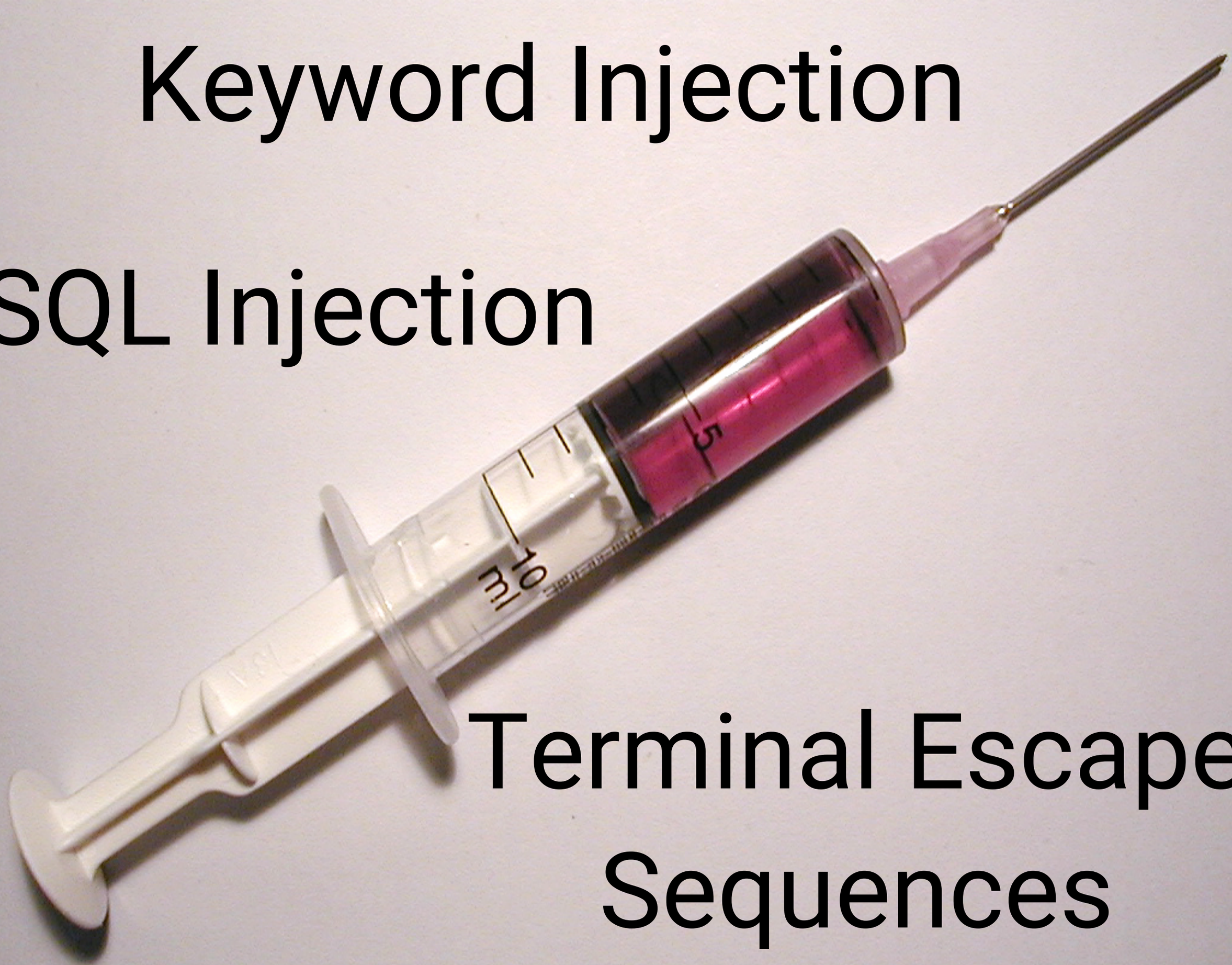
```
snprintf(buf, sizeof(buf), user_data)
```

right

```
snprintf(buf, sizeof(buf), “%s”,
         user_data)
```

Keyword Injection

SQL Injection



Terminal Escape Sequences

ntop - *popen(3)*

```
if(num == 0) putenv("QUERY_STRING=");

sprintf(line, "%s/cgi/%s",
getenv("PWD"), cgiName);

if((fd = popen(line, "r") == NULL)
{ ...

        cgiName = ";/bin/rm -rf /"
```

AMaViS: Execute Shell

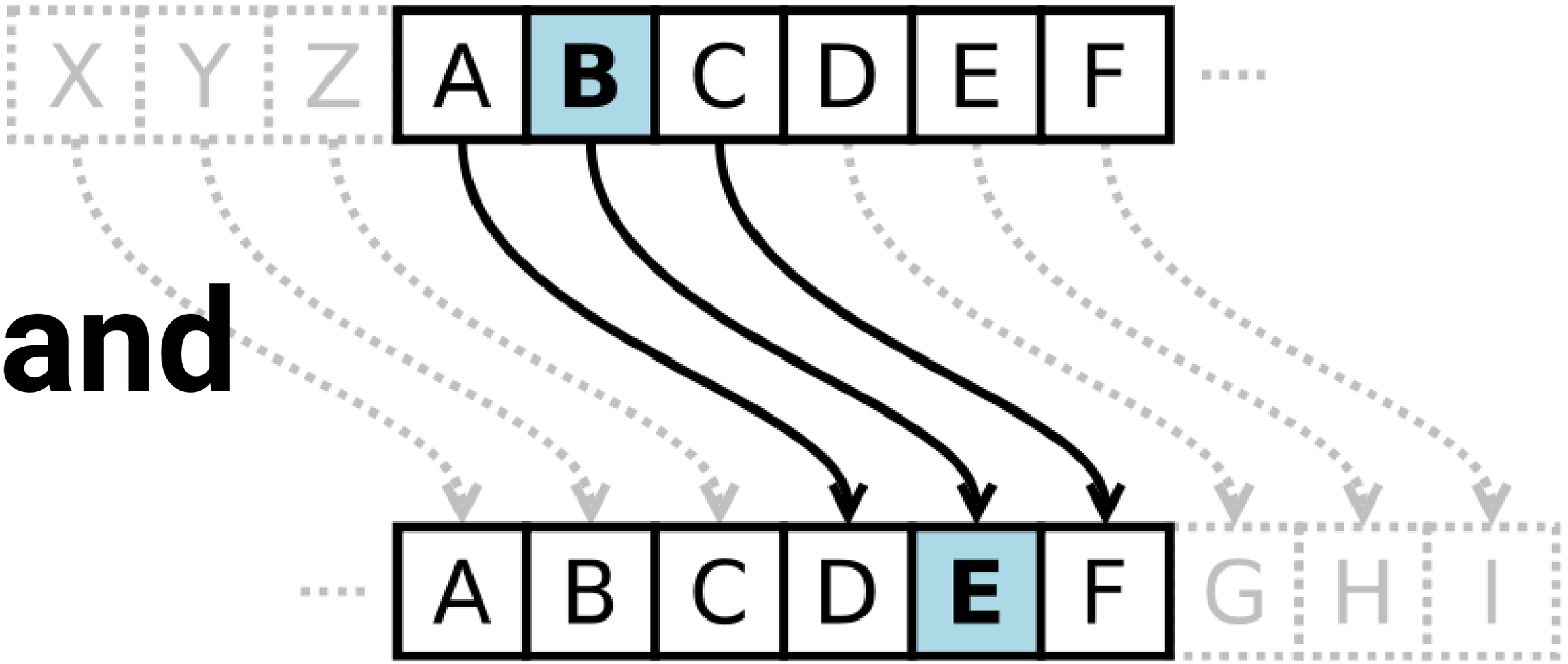
```
cat <<EOF| ${mail} -s "VIRUS IN YOUR MAIL TO $7" $2

                V I R U S  A L E R T

  Our viruschecker found a VIRUS in your eMail to "$7".
              We stopped delivery of this eMail!

    Now it is on you to check your system for viruses

  For further information about this viruschecker see:
                  http://aachalon.de/AMaViS/
          AMaViS - A Mail Virus Scanner, licenced GPL
EOF
```

Ok, what should I do?

- Filter/Sanitize untrusted Input
- White-List vs. Black-List
- **a-zA-Z0-9\s** is often sufficient
- use Escape-Functions, like `mysql_real_string_escape()`
- avoid Functions that use the Shell, like `popen(3), system(3), glob()` (Perl), `<>` (Perl), etc.

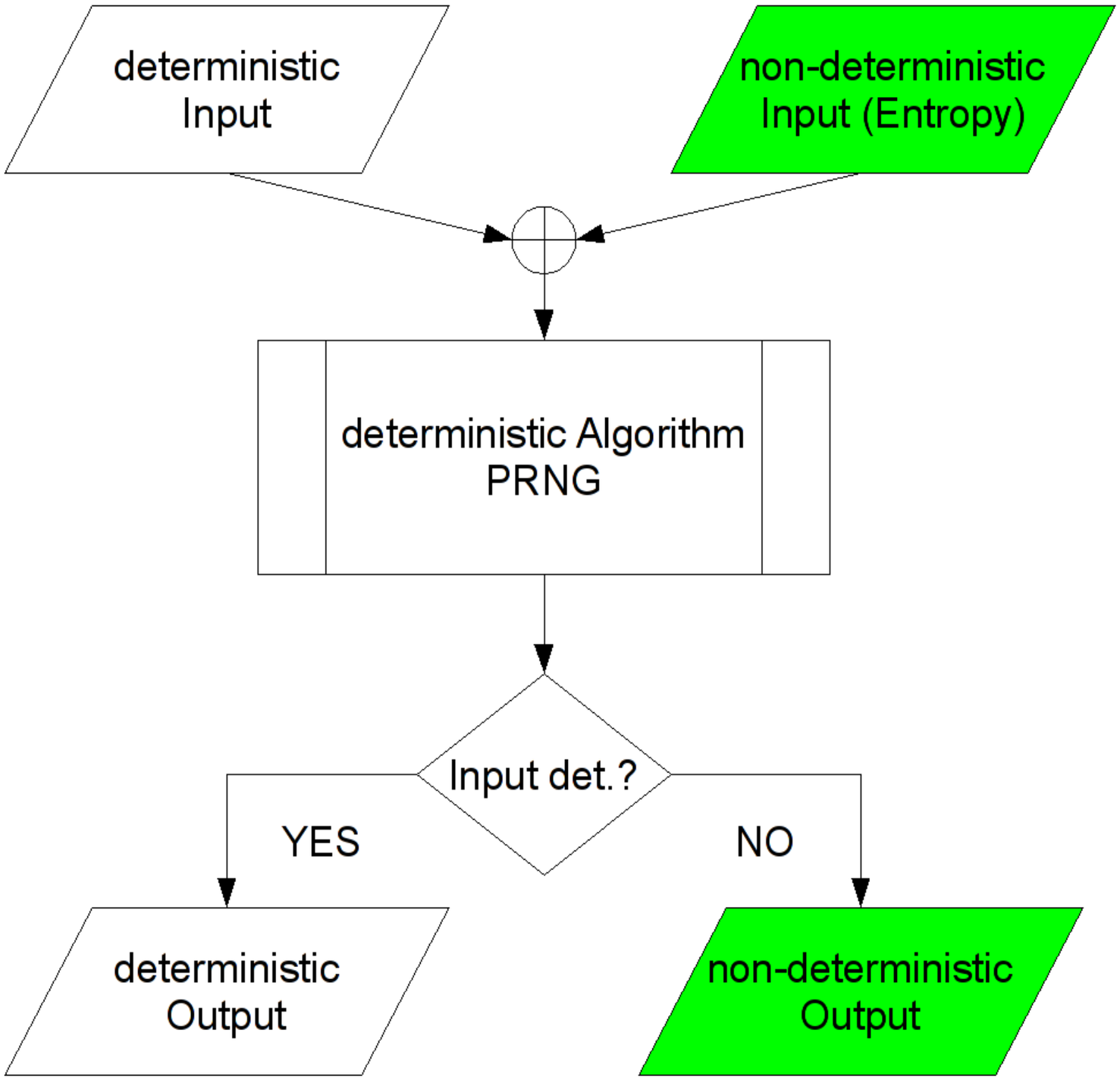
Cryptography

and



Randomness

"Anyone who considers arithmetical methods of
producing random digits is, of course, in a state of sin."
– John von Neumann



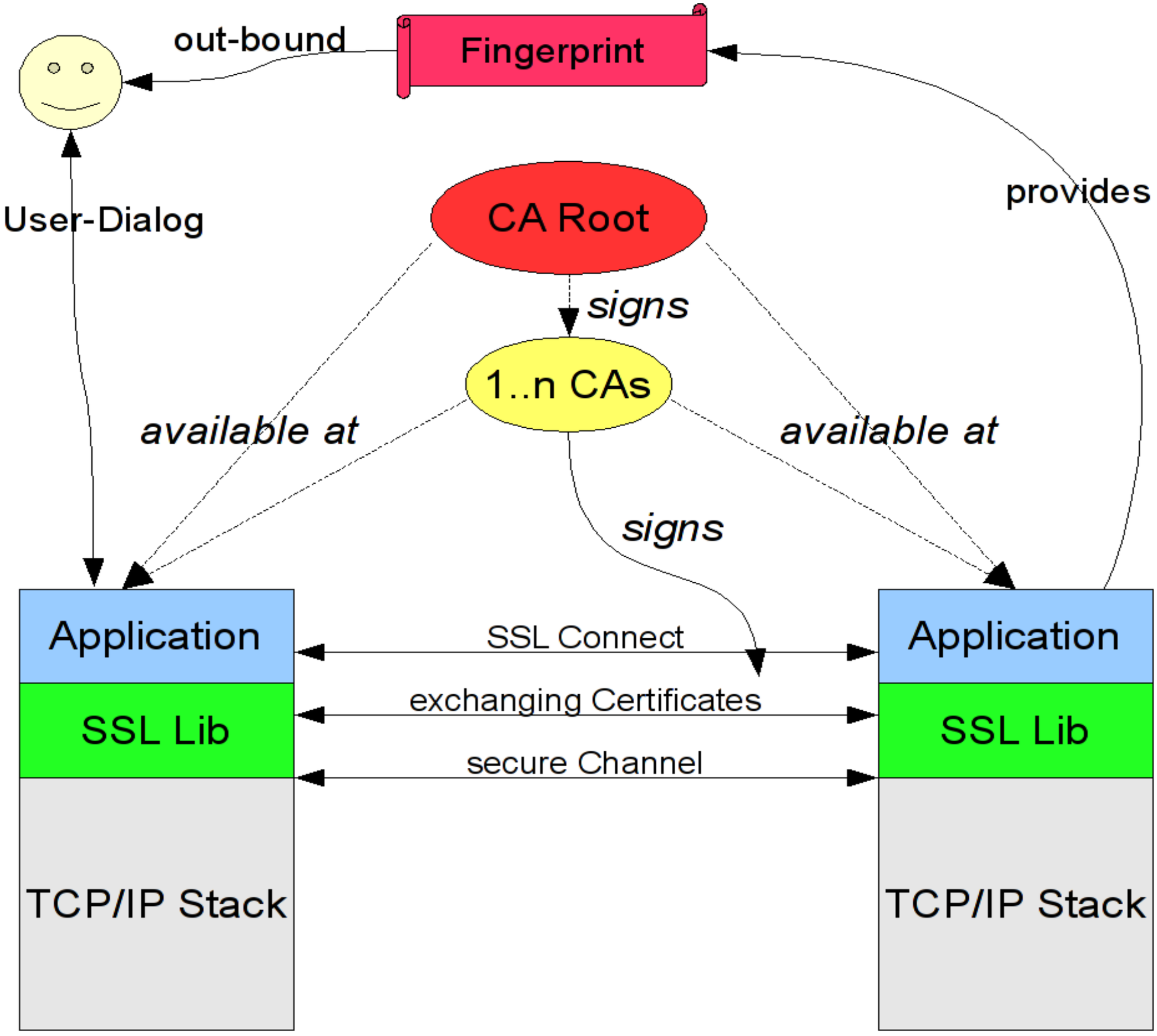
Random Numbers

- max. Entropy: $H_{max} = 1$, if $P = 0{,}5 +/- e$; for $e = 0$
- **statistical** Randomness vs. **cryptographical** Randomness
- important Building Block for Algorithms and Protocols
- ... but hard to find on a deterministic Machine

Dos	Don'ts
use /dev/random (keys) or /dev/urandom	use of fread(3) causes buffering and unneeded waste of entropy
use EGADS or EGD otherwise, or user-generated entropy and use this seed as IV for a Counter-Mode Block-Cipher	using a PRNG from a software library (like rand(3) etc.), only statistical randomness, no *Forward Security*
use a random/changing *Salt* (avoid *Code-Book* attack) for password generation	using the system time, b/c only provides a few bits of entropy
use random/changing IV otherwise the first block in a CBC encryption leaks information	XOR as encryption algo. and timestamp/non-random value as key/*One-Time-Pad*

Secure Socket Layer

SSL and TLS


out-bound
Fingerprint
User-Dialog
provides
CA Root
signs
1..n CAs
available at
available at
signs
Application
SSL Connect
Application
SSL Lib
exchanging Certificates
SSL Lib
secure Channel
TCP/IP Stack
TCP/IP Stack

Bad Idea	Reason/Consequences
do not verify certs	attacker can provide his own cert
too short user-dialog	too less information about the cert, accepted to continue work
missing cert-fingerprint in user-dialog	FP to exactly identify cert (self-signed), accepted to cnt. work
MD5 fingerprint only	MD5 not secure anymore, better use MD5+SHA1
do not be too user-friendly and just import possible missing (CA-)cert	too tricky and can possibly attacked to bypass cert verification

Bad Idea	Reason/Consequences
blindly trusting every scripting language to fully support SSL	Python, for example, provides an ssl class but does not verify the certs, better read the API manual
use SSLv2, out-dated or „exportable“ algorithms	known to be vulnerable
Too small keys	sym. ciphers >= 128 bit asym. ciphers >= 1024 bit
no *ephemermal keying*	no *perfect forward secrecy* (PFS)
PRNG seed with low entropy	keys may be guessed etc., at least as much entropy as key length needed (better more)

What can I do?

- Am I safe with *AppArmor, SELinux, etc.?* No!
- Am I safe with all this nifty *gcc* Options? No!
- So, am I lost? No!
  - fully understand the Prg. Language you use
  - check your Program’s Input (Type, Size, ...)
  - use existing Crypto Libraries when needed
  - if in doubt, ask the Security-Team other Developers

Tools

- *splint* too quickly check C code
- *FindBugs* for Java
- *gdb*/*strace*/*ltrace* to trace/change code while running
- try the various fuzzers availbale online
- *cscope/kscope* to browse the code, generate call-graphs etc.
- *wireshark* to analyze network traffic
- an arbitrary SSL MiM tool (*ethercap*, write your own!)
- much more + the various commercial code analyzers

Readings

- http://www.suse.de/~thomas/papers/
- http://www.dwheeler.com/secure-programs/
- http://www.owasp.org/
- Seacord, Robert; The CERT C Secure Coding Standard; Addison-Wesley Longman
- Viega, John; McGraw, Gary; Building Secure Software: How to Avoid Security Problems the Right Way; Addison-Wesley Longman
- Huseby, Sverre; Shafir, Angelika; Sicherheitsrisiko Webanwendung: Wie Web-Programmierer Sicherheitslücken erkennen und vermeiden; Dpunkt Verlag
- Ferguson, Niels; Schneier, Bruce;Practical Cryptography; Wiley & Sons
- Zeller, Andreas, Why Programs Fail; dpunkt Verlag