nuxi

CloudABI: safe, testable and maintainable software for UNIX

Speaker:
Ed Schouten, ed@nuxi.nl

Programme

- **What is wrong with UNIX?**
- What is CloudABI?
- Use cases for CloudABI
- Links

What is wrong with UNIX?

UNIX-based operating systems are awesome, but in my opinion:

- they don’t help you to run software securely.
- they make it hard to reuse and test software.
- systems administration doesn’t scale.

UNIX security model

UNIX security in a nutshell:

- Processes have **credentials**.
- Most objects have **access control lists**.
- Some parts have no adjustable access controls.

- Decent model for login sessions of physical users.
- Pretty bad model for running services.

Security of a web service

A web service would only need to access:

- incoming network connections for HTTP requests,
- optional: a directory containing its data files,
- optional: database backends.

In practice, an attacker can:

- extract a tarball of all world-readable data under `/`,
- register `cron` jobs,
- spam TTYs using the `write` tool,
- turn the system into a botnet node.

Reusability and testability

UNIX programs are hard to reuse and test as a whole.

To explain why, let’s take a look at how these aspects are solved elsewhere, for example in Java, and compare.

Reuse and testing in Java #1

```
class WebServer {
  private Socket socket;
  private String root;
  WebServer() {
    this.socket = new TCPSocket(80);
    this.root = “/var/www”;
  }
}
```

Reuse and testing in Java #2

```java
class WebServer {
  private Socket socket;
  private String root;
  WebServer(int port, String root) {
    this.socket = new TCPSocket(port);
    this.root = root;
  }
}
```

Reuse and testing in Java #3

```
class WebServer {
  private Socket socket;
  private Directory root;
  WebServer(Socket socket, Directory root) {
    this.socket = socket;
    this.root = root;
  }
}
```

Reusability and testability

UNIX programs are typically similar to the first two examples. In many cases:

- parameters are hardcoded.
- parameters are specified in configuration files stored at hardcoded locations.
- resources are acquired on behalf of you, instead of allowing them to be passed in.

Reusable and testable web server

```
#include <sys/socket.h>
#include <unistd.h>

int main() {
    int fd;
    while ((fd = accept(0, NULL, NULL)) >= 0) {
        const char buf[] = “HTTP/1.1 200 OK\r\n”
                           “Content-Type: text/plain\r\n\r\n”
                           “Hello, world\n”;
        write(fd, buf, sizeof(buf) - 1);
        close(fd);
    }
}
```

Reusable and testable web server

Web server is reusable:

- Web server can listen on any address family (IPv4, IPv6), protocol (TCP, SCTP), address and port.
- Spawn more on the same socket for concurrency.

Web server is testable:

- It can be spawned with a UNIX socket. Fake requests can be sent programmatically.

Programme

- What is wrong with UNIX?
- **What is CloudABI?**
- Use cases for CloudABI
- Links

What is CloudABI?

A new UNIX-like runtime environment that allows you to more easily develop:

- software that is better protected against exploits,
- software that is reusable and testable,
- software that can be deployed at large scale.

Based on the ideas of University of Cambridge Computer Laboratory’s Capsicum.

Default rights

By default, CloudABI processes can only perform actions that have no global impact:

- They **can** allocate memory, create pipes, socket pairs, shared memory, etc.
- They **can** spawn threads and subprocesses.
- They **can** obtain the time of day.
- They **cannot** open paths on disk.
- They **cannot** create network connections.
- They **cannot** observe the global process table.

Additional rights: file descriptors

File descriptors are used to grant additional rights:

- File descriptors to directories: expose parts of the file system to the process.
- Sockets: make a process network accessible.
    - File descriptor passing: receive access to even more resources at run-time.

File descriptors have permission bitmasks, allowing fine-grained limiting of actions performed on them.

Secure web service

Consider a web service running on CloudABI that gets started with the following file descriptors:

- a socket for incoming HTTP requests,
- a read-only file descriptor of a directory, storing the files to be served over the web,
- an append-only file descriptor of a log file.

When exploited, a hacker can do little to no damage.

Defence in depth #1

Assume the web service allows users to upload videos that have to be transcoded to a common format:

- Spawn a separate subprocess for the transcoding.
- Only grant the subprocess two pipes for video input and output.

Security bug in video transcoder: attacker can only write garbage output.

Defence in depth #2

Assume the web service has `/~username/` support:

- Don’t start the web service process with a file descriptor to `/home`.
- Run a separate process that can access `/home` and hands out descriptors to `/home/*/www`.

Exploit in web service never yields access to data outside of user WWW directories.

Testability of CloudABI processes

Reusing and testing CloudABI executables is simple:

- Just start the process with a different set of file descriptors.
- It is impossible that the process depends on something not covered by its file descriptors.
- Also makes it a lot easier to migrate processes from one system to another.

nuxi

Cross-platform support

Observation: UNIX ABIs become tiny if you remove all interfaces that conflict with capability-based security.

- CloudABI is a separate ABI. It’s an operating system without a kernel implementation.
- CloudABI only has ~60 system calls. Most of them are not that hard to implement.
- Goal: Add support for CloudABI to existing UNIX-like operating systems.
- Allows for reuse of binaries without recompilation.

Supported platforms

Hardware architectures:

- x86-64

Operating systems:

- FreeBSD: full support
- NetBSD: full support
- Linux: currently in development (“Hello, world”)
- Others: no support (yet), but you can already use these to develop and compile software

nuxi

cloudlibc

cloudlibc: a C library, specifically made for CloudABI.

- Only contains functions that make sense in a capability-based environment.
  - ‘90% POSIX compliant’.
  - Compiler errors when using unsupported constructs.
- Very high testing coverage.
  - ~650 unit tests.
  - Tests are used to ensure consistent behaviour between operating systems.

How to use CloudABI

1. Install a cross compiler: Clang and Binutils.
2. Install cloudlibc.
3. Install additional libraries, such as libc++ for C++14 support.
4. Patch up your operating system kernel to support CloudABI executables.
5. Compile and execute your own code.

Programme

- What is wrong with UNIX?
- What is CloudABI?
- **Use cases for CloudABI**
- Links

‘CloudABI as a Service’

A service where customers can upload executables and have them executed in the cloud.

- Unlike Amazon EC2, there is no virtualization overhead.
- Unlike Amazon EC2, there is no need to do traditional UNIX systems administration.
- Unlike Google App Engine, applications can be written in any language; not just Python/Java/Go.

High-level cluster management

CloudABI as the basis of a cluster management suite:

- Dependencies of software are known up front.
- Allows for smarter scheduling.
  - Automatic capacity planning.
  - Improving locality.
- Automatic migration of processes between systems.
- Automatic routing of traffic on external addresses to internal processes, load balancing, etc.

nuxi

‘Native Client’

Google Native Client:

- Sandbox which is part of Google Chrome that allows you to run native code directly on the client.

CloudABI:

- Native Client, but not coupled to a browser.
- Could be used to safely add scripting/plugin support to any application.

Programme

- What is wrong with UNIX?
- What is CloudABI?
- Use cases for CloudABI
- **Links**

Links

- cloudlibc: GitHub
- FreeBSD patches: GitHub
- NetBSD patches: GitHub

- Capability-based security: Wikipedia
- Capsicum: University of Cambridge Computer Lab

- Email: info@nuxi.nl