

Confinement

(Running Untrusted Programs)

Chester Rebeiro

Indian Institute of Technology Madras

Untrusted Programs

Untrusted Application

- Entire Application untrusted
- Part of application untrusted
 - Modules or library untrusted

Possible Solutions

- Air Gapped Systems
 - Virtual Machines
 - Containers
- (all are coarse grained solutions)

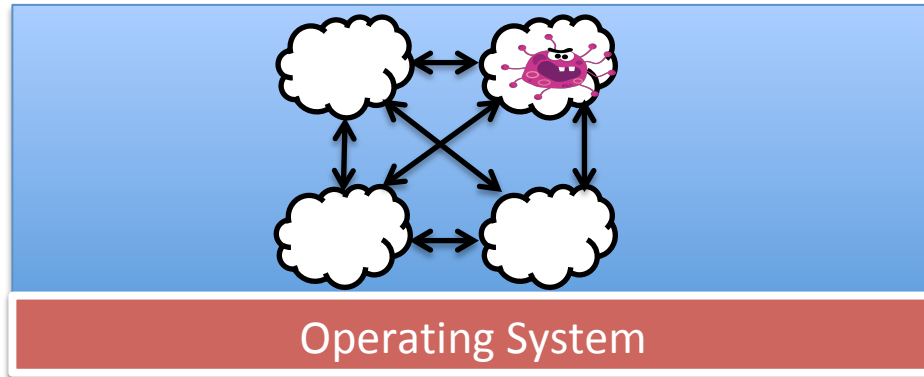
Vulnerable Applications

- A vulnerability in one application compromises the entire application

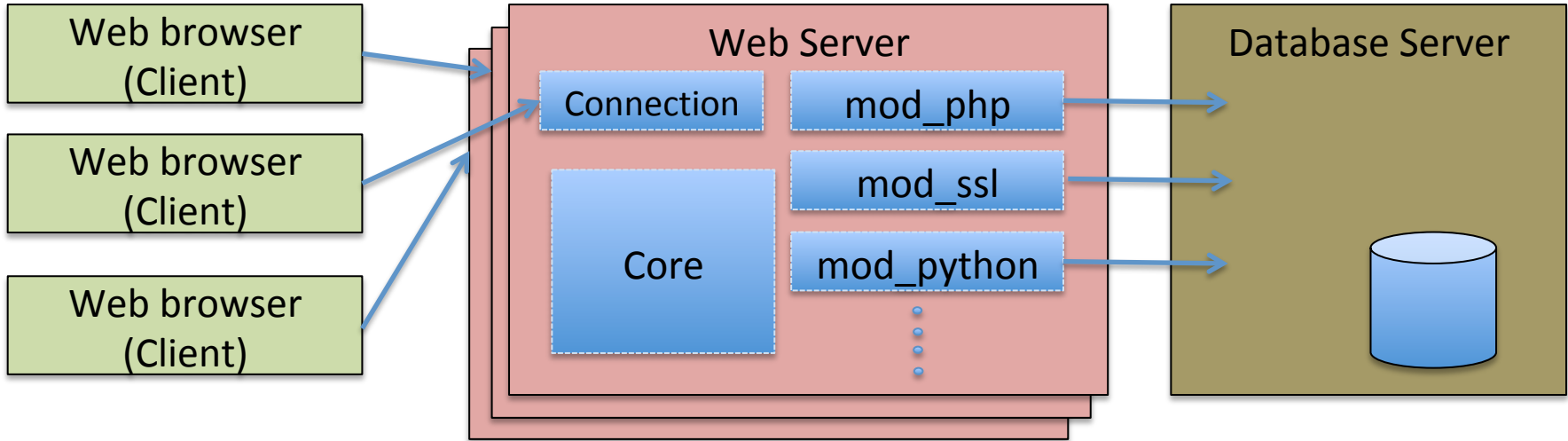


Confinement (using RPCs)

- Run each module as a different process (different address spaces)
 - Use RPCs to communicate between modules
 - Hardware ensures that one process does not affect another



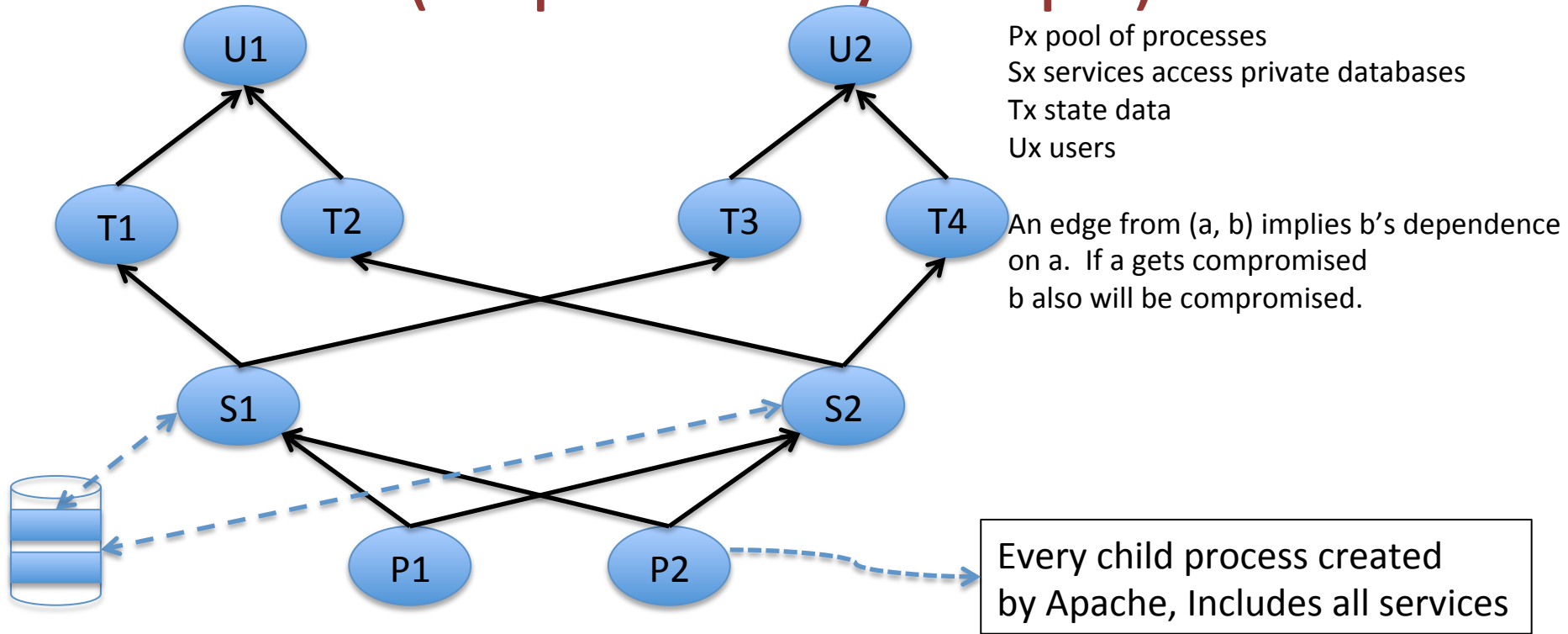
Typical Web Server



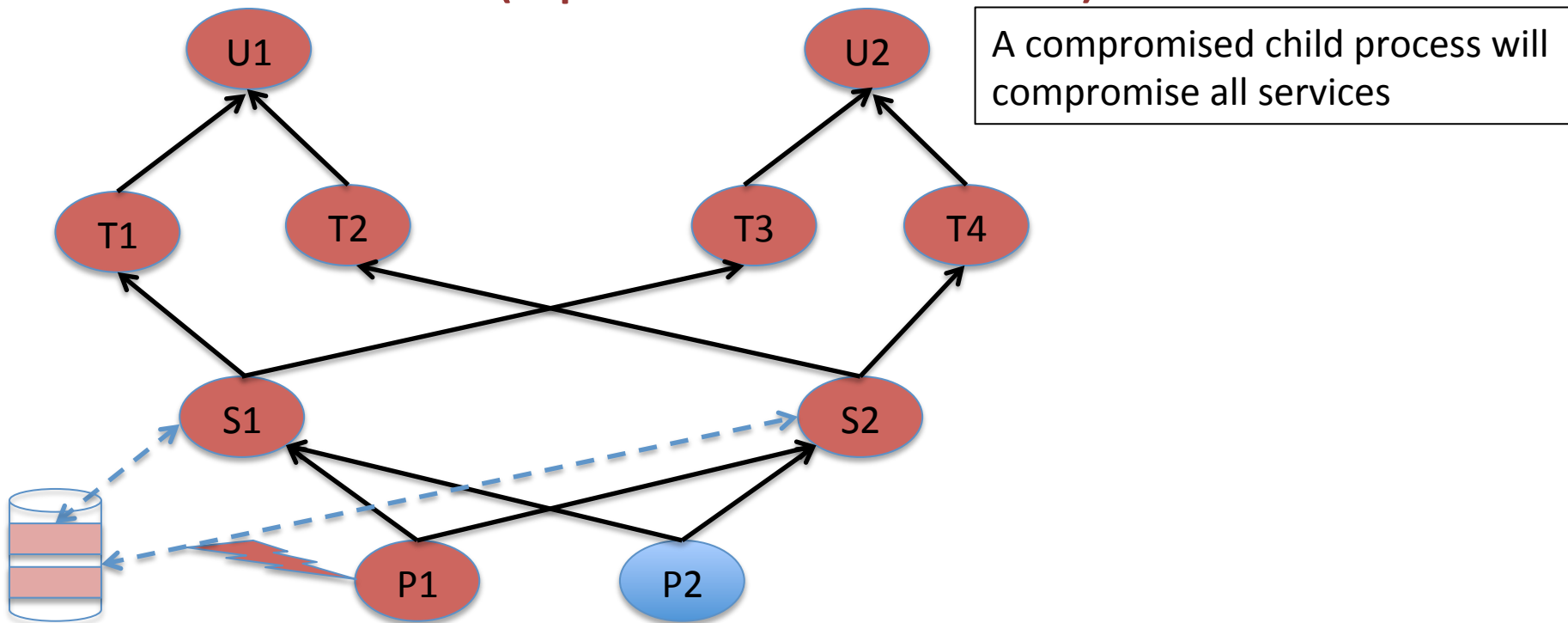
- single address space holds multiple web servers
- Every new client creates a new process
- HTTP interfaces restrict access to the database server
- Security achieved by coarse grained access control mechanisms in the data base server
- A vulnerability in any component can ripple through the entire system

https://www.usenix.org/event/usenix04/tech/general/full_papers/krohn/krohn.pdf

Apache Webserver (Dependency Graph)



A compromised process (Apache Webserver)



Known attacks on Web Servers

- A bug in one website can lead to an attack in another website
example: Amazon holds credit card numbers. If it happens to share the same web server as other users this could lead to trouble.
- Some known attacks on Apache's webserver and its standard modules
 - Unintended data disclosure (2002)
users get access to sensitive log information
 - Buffer overflows and remote code execution (2002)
 - Denial of service attacks (2003)
 - Due to scripting extensions to Apache

Principle of Least Privileges

Aspects of the system most vulnerable to attack are the least useful to attackers.

- Decompose system into subsystems
- Grant privileges in fine grained manner
- Minimal access given to subsystems to access system data and resources
- Narrow interfaces between subsystems that only allow necessary operations
- Assume exploit more likely to occur in subsystems closer to the user (eg. network interfaces)
- Security enforcement done outside the system (eg. by OS)

OKWS Webserver

(designed for least privileges)

each independent service runs in an independent process

Do not expose more code/
services than required!
Tradeoff security vs performance

Each service should run in a separate chroot jail

Allow access to only necessary files.

Each process should run as a different unprivileged user.

Prevent interfering with other processes

Narrow set of database access privileges

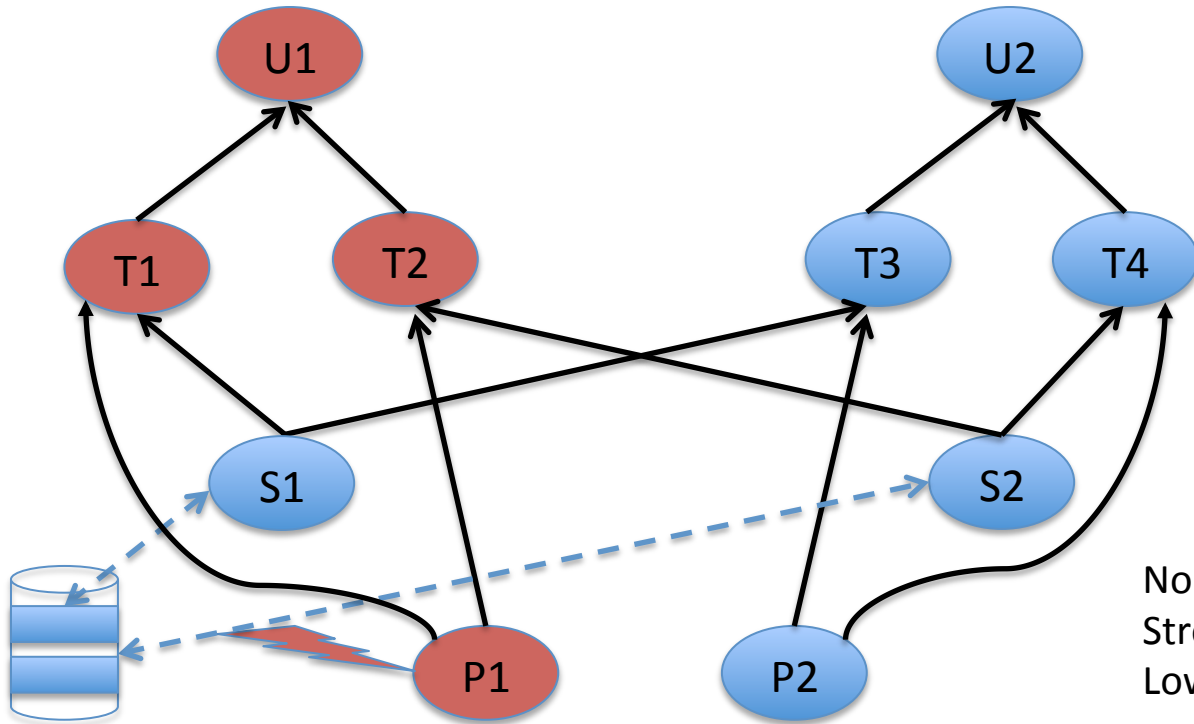
Prevent unrequired access to the DB service

Achieving Confinement

Through Unix Tools

- **chroot:** define the file system a process can see
if system is compromised, the attacker has limited access to the files. Therefore, cannot get further privileges
- **setuid:** set the uid of a process to confine what it can do
if system runs as privileged user and is compromised, the attacker can manipulate other system processes, bind to system ports, trace system calls, etc.
- **Passing file descriptors:** a privileged parent process can open a file and pass the descriptor to an unprivileged child
(don't have to raise the privilege of a child, to permit it to access a specific high privileged file)

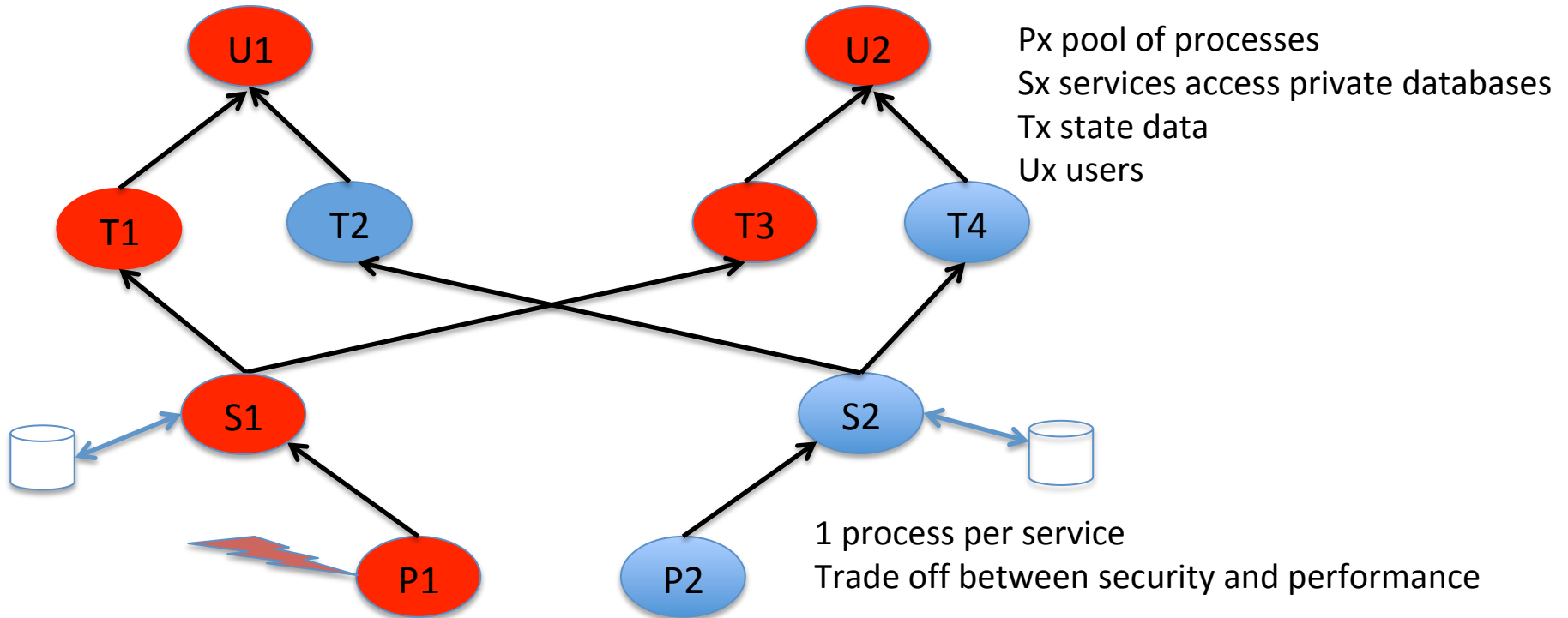
Strict Confinement



If a user process is compromised,
then data corresponding to that
process is compromised

No sharing of services or processes;
Strong confinement;
Low performance due to too many
processes
(1 process per user)

OKWS



OKWS Design



okld

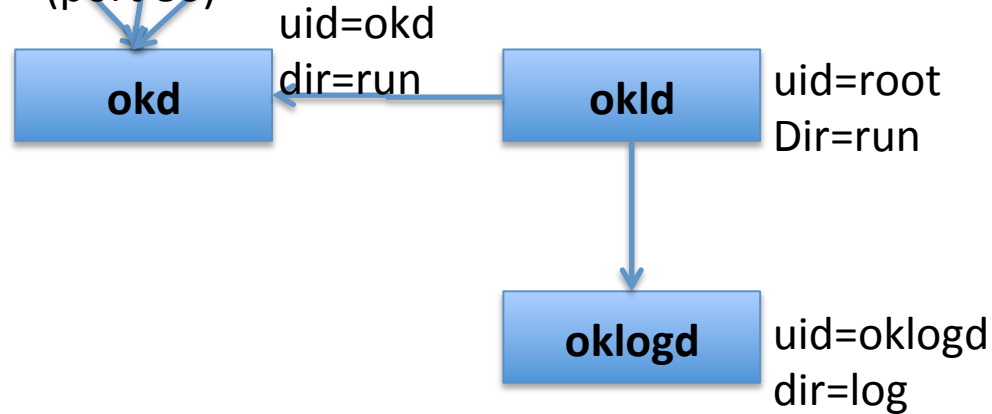
uid=root
dir=run

runs as superuser; bootstrapping; chroot directory is run monitors processes; relaunched them if they crash

OKWS Design

External connections

(port 80)



Launch okd (demux daemon) to route traffic to appropriate service ;

If request is valid, forwards the request to the appropriate service

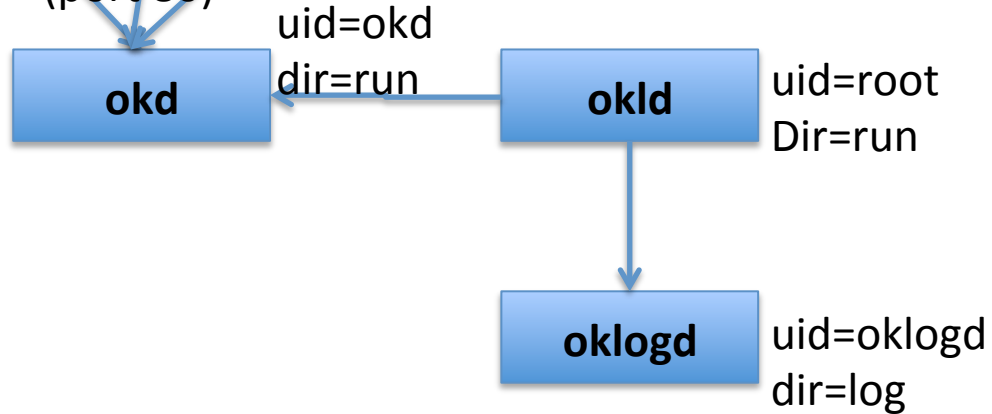
If request is invalid, send HTTP 404 error to the remote client

If request is broken, send HTTP 500 error to the remote client

OKWS Design

External connections

(port 80)

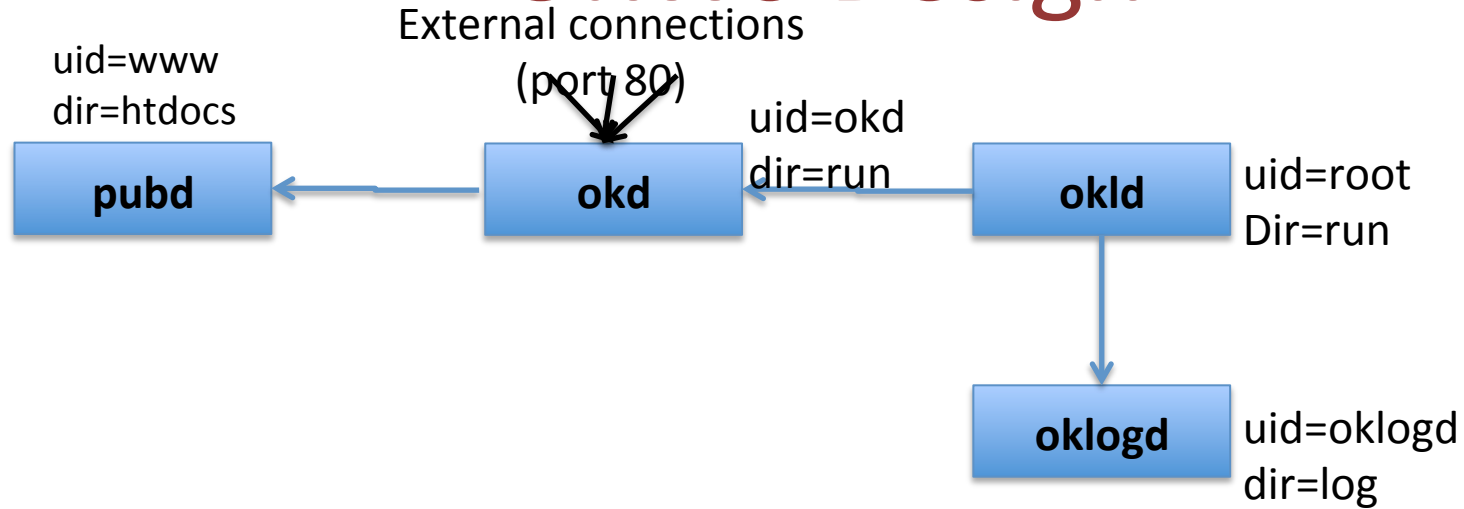


oklogd daemon to write log entries to disk

chroot into their own runtime jail (within a jail, each process has just enough access privileges to read shared libraries on startup, dump core files if crash)

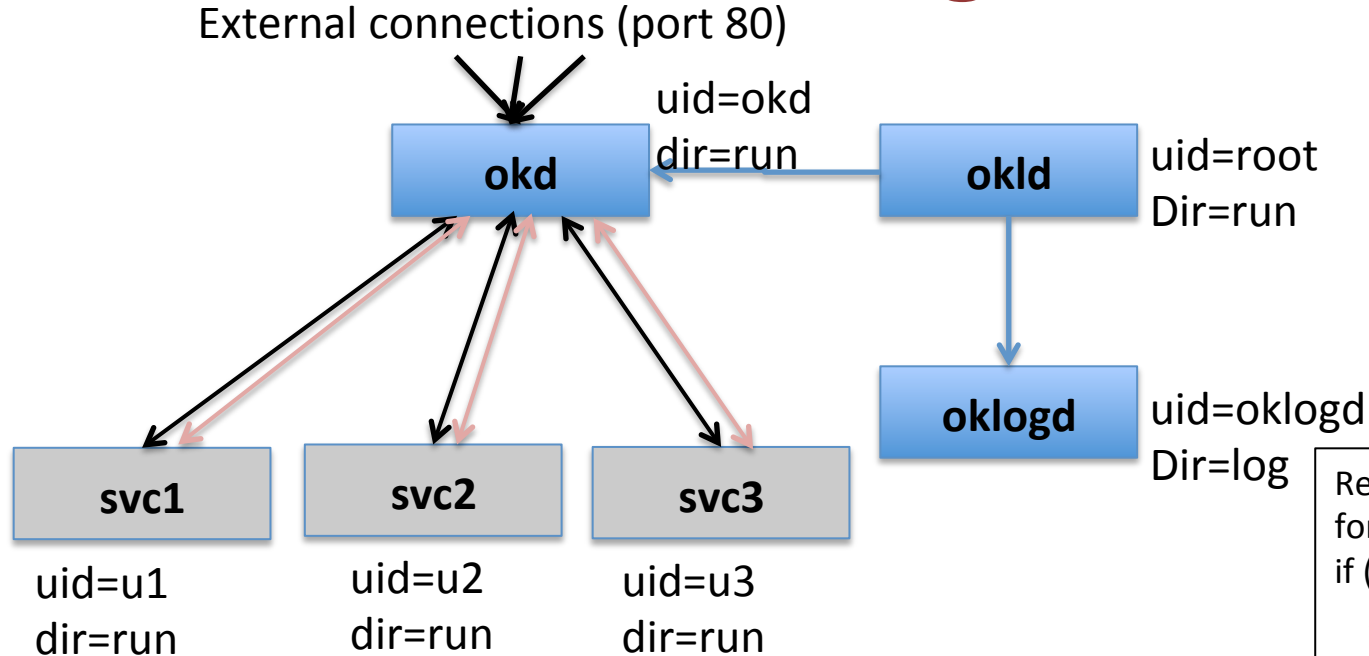
Each service runs as an **unprivileged user**

OKWS Design



pubd: provides minimal access to local configuration files, html files
Read only access to the files

OKWS Design



okld launch services; each service in its chroot with its own uid
Services owned by root with permissions 0410 (can only be executed by user)
okld catches SIGCHLD and restarts services if they crash

```
Request 2 sockets
fork()
if (child process){
    setuid()
    chroot()
    exec()
}
```

Logging

- Each service uses the same logging file
 - They use the oklogd to write into the file via RPCs
 - oklogd runs in its own chroot jail
 - Any compromised service will not be able to modify / read the log
 - A compromised service may be able to write arbitrary messages to the log (noise)