Capsicum and Casper

a fairy tale about solving security problems

BSDCan, Ottawa, 2016

Mariusz Zaborski

<m.zaborski@wheelsystems.com>
<oshogbo@FreeBSD.org>

Outline

1. Do we need sandbox?
2. seccomp(2)
3. pledge()
4. Capsicum
5. CloudABI
6. Casper

Do we need a sandbox?

cat(1)

Ambient authority

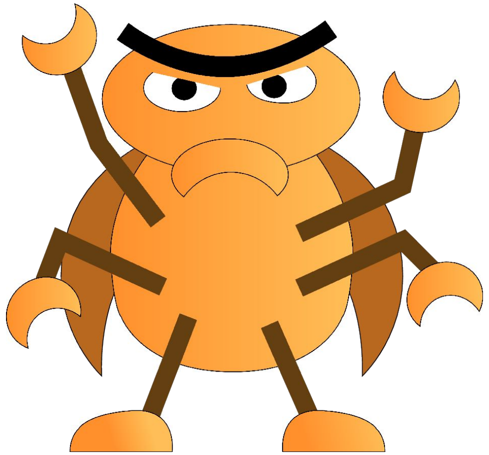
Threat Mitigation Techniques

- ASLR
- canneries
- NX bit

Do we need a sandbox?

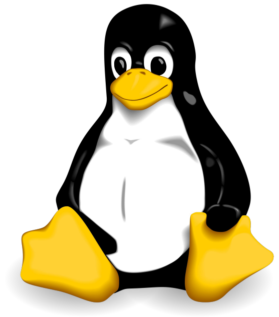
Mateusz "j00ru" Jurczyk and Gynvael Coldwind in 2010 - 2014 using fuzzing techniques contributed to:

- 1120 bug fixes in ffmpeg
- 60 CVE in flash
- 568 unique crashes in Adobe Reader



seccomp(2)

seccomp(2)



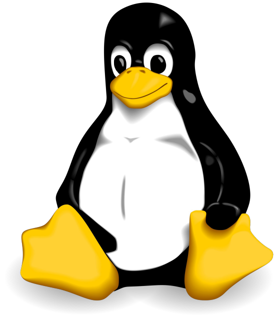
- 2005
- Linux
- seccomp(2)
    - Former prctl(2) - PR_SET_SECCOMP
    - Very very former - /proc/self/seccomp
- SECCOMP_SET_STRICT
    - Allowed read(2), write(2), _exit(2), sigreturn(2)
- SECCOMP_SET_MODE_FILTER
    - Berkeley Packet Filter (BPF)

https://www.kernel.org/doc/Documentation/prctl/seccomp_filter.txt

seccomp(2)

```
/* Simple helpers to avoid manual errors (but larger BPF programs). */
#define SC_DENY(_nr, _errno) \
        BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K, __NR_ ## _nr, 0, 1), \
        BPF_STMT(BPF_RET+BPF_K, SECCOMP_RET_ERRNO|(_errno))
#define SC_ALLOW(_nr) \
        BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K, __NR_ ## _nr, 0, 1), \
        BPF_STMT(BPF_RET+BPF_K, SECCOMP_RET_ALLOW)

/* Syscall filtering set for preauth. */
static const struct sock_filter preauth_insns[] = {
        /* Ensure the syscall arch convention is as expected. */
        BPF_STMT(BPF_LD+BPF_W+BPF_ABS,
                offsetof(struct seccomp_data, arch)),
        BPF_JUMP(BPF_JMP+BPF_JEQ+BPF_K, SECCOMP_AUDIT_ARCH, 1, 0),
        BPF_STMT(BPF_RET+BPF_K, SECCOMP_FILTER_FAIL),
        /* Load the syscall number for checking. */
        BPF_STMT(BPF_LD+BPF_W+BPF_ABS,
                offsetof(struct seccomp_data, nr)),
        SC_DENY(open, EACCES),
        SC_ALLOW(getpid),
        ...
};

static const struct sock_fprog preauth_program = {
        .len = (unsigned short)(sizeof(preauth_insns)/sizeof(preauth_insns[0])),
        .filter = (struct sock_filter *)preauth_insns,
};

if (prctl(PR_SET_SECCOMP, SECCOMP_MODE_FILTER, &preauth_program) == -1)
        debug("prctl(PR_SET_SECCOMP)",);
```

libseccomp(3)

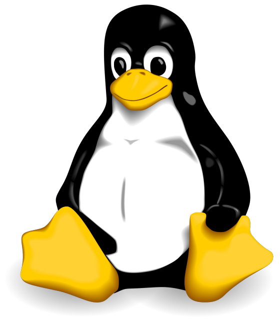


- seccomp_init()
- seccomp_rule_add()
- seccomp_load()

```
seccomp_init(SCMP_ACT_ERRNO(5));
seccomp_rule_add(SCMP_ACT_ALLOW, SCMP_SYS(close), 0);
seccomp_rule_add(SCMP_ACT_ALLOW, SCMP_SYS(dup), 0);
seccomp_rule_add(SCMP_ACT_ALLOW, SCMP_SYS(write), 0);
seccomp_rule_add(SCMP_ACT_ALLOW, SCMP_SYS(exit), 0);
seccomp_load();
```

https://github.com/seccomp/libseccomp

seccomp(2)



- Chrome/Chromium
- OpenSSH
- Vsftpd
- LXD
- Firefox
- FirefoxOS
- Cjdns

pledge()

pledge()

- OpenBSD project
- formerly known as tame
- similar concept to seccomp
- dividing the program into two parts
  - the initialization stage and the main loop
- a simple interface

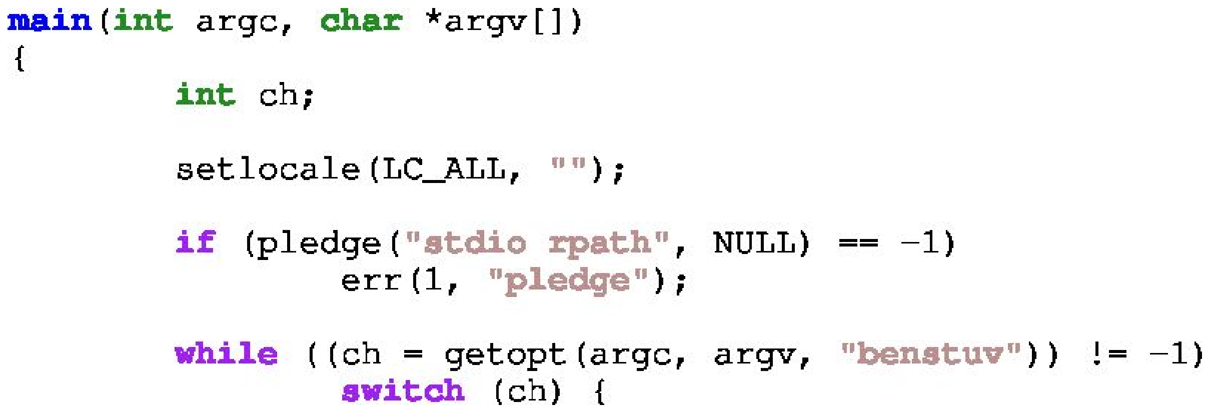
```
pledge(const char *promises, char *whitepath[]);
```

- whitepath - not yet implemented
- used in over 400 programs

pledge() - promises

- 25 promises, a few examples:
  - `stdio` - allows for the allocation of memory and performance of basic io operations
  - `rpath` - allows for functions which can only cause read-only effects on filesystems
  - `wpath` - allows systems call which may cause write-effects on filesystems
  - `cpath` - allows for functions which may create new files
  - innet - allow for functions which operates in the AF_INET and AF_INET6
  - `proc` and `exec` - allows fork and to execute another program

pledge() - usage example in cat



```
main(int argc, char *argv[])
{
        int ch;

        setlocale(LC_ALL, "");

        if (pledge("stdio rpath", NULL) == -1)
                err(1, "pledge");

        while ((ch = getopt(argc, argv, "benstuv")) != -1)
                switch (ch) {
```

pledge()

- bgpd
- dhclient
- dhcpd
- dvmrpd
- eigrpd
- file
- httpd
- Iked
- ldapdldpd

- mountd
- npppd
- ospfd, ospf6d
- pflogd
- radiusd
- relayd
- ripd
- scriptsmtpd
- syslogd

- tcpdump
- tmux
- xconsole
- xdm
- x server
- ypldap
- pkg_add

pledge() - issues

- execv turns off sandbox

  every fourth program uses it
- hardcoded paths in kernel
  - open(2) files like /etc/localtime
  - readlink(2) /etc/malloc.conf
- One template ???
- Reload configuration ???

Capsicum

Capsicum

- tight sandboxing (cap_enter(2))
- capability rights (cap_rights_limit(2))

Capsicum

80 capability rights, a few examples

- CAP_FCHMOD
- CAP_READ
- CAP_UNLINKAT
- CAP_APPEND
- CAP_WRITE

Capsicum

Two ways to obtain more capabilities:

- the initialization phase
- delegation

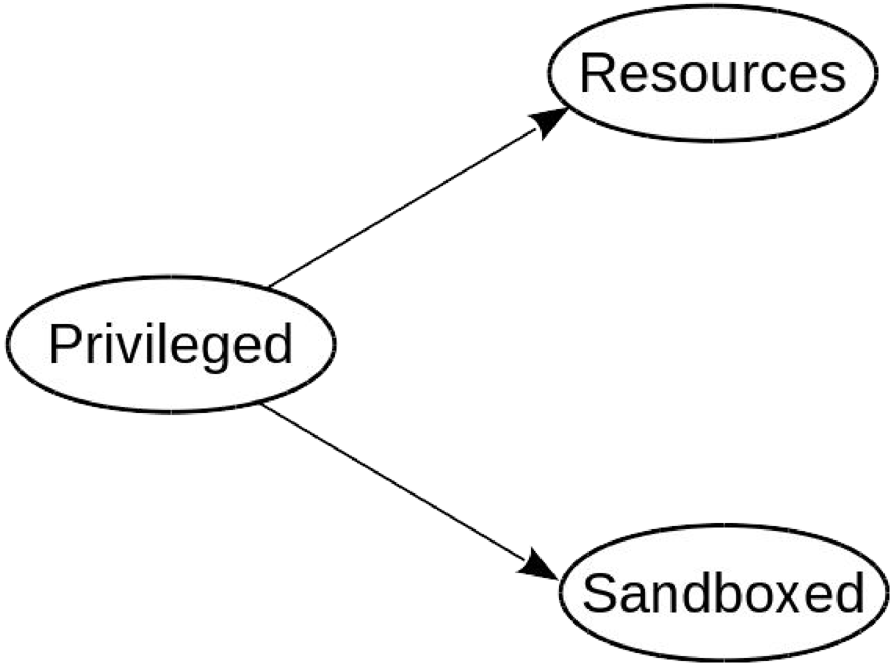
Capsicum - uniq(2)

```
cap_rights_t rights;
...
ifp = stdin;
ifn = "stdin";
ofp = stdout;
if (argc > 0 && strcmp(argv[0], "-") != 0)
        ifp = file(ifn = argv[0], "r");

cap_rights_init(&rights, CAP_FSTAT, CAP_READ);
if (cap_rights_limit(fileno(ifp), &rights) < 0 && errno != ENOSYS)
        err(1, "unable to limit rights for %s", ifn);

cap_rights_init(&rights, CAP_FSTAT, CAP_WRITE);
if (argc > 1)
        ofp = file(argv[1], "w");
else
        cap_rights_set(&rights, CAP_IOCTL);
if (cap_rights_limit(fileno(ofp), &rights) < 0 && errno != ENOSYS) {
        err(1, "unable to limit rights for %s",
            argc > 1 ? argv[1] : "stdout");
}
```

Capsicum - uniq(2)

```
if (cap_enter() < 0 && errno != ENOSYS)
        err(1, "unable to enter capability mode");
```

Capsicum - delegation template



Capsicum

- dhclient(8)
- hastd(8), hastctl(8)
- rwhod(8), rwho(1)
- tcpdump(8)
- kdump(1)
- ping(8)
- uniq(1)
- auditdistd(8)
- sshd(8)
- pkg(8)
- chromium

Capsicum - issues

- high barriers to entry
- libc is not your friend
- libraries are not your friend as well
- magic calls

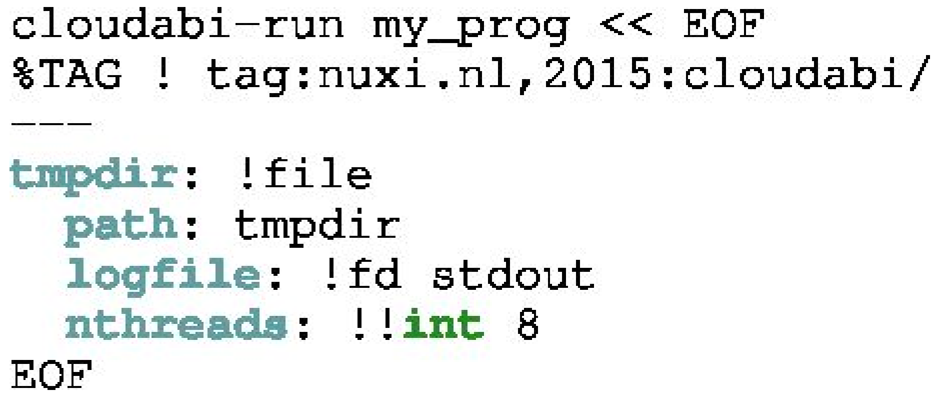
```
/*
 * Cache files required for time(3) and localtime(3)
 * before entering capability mode.
 */
(void) time(&ct);
(void) localtime(&ct);
if (cap_enter() < 0 && errno != ENOSYS)
        err(1, "cap_enter");
```

It is all about reducing TCB

CloudABI

CloudABI

- Designed to use in cloud
- Use Capsicum
- Portable ELF files
- Special runtime environment



```
cloudabi-run my_prog << EOF
%TAG ! tag:nuxi.nl,2015:cloudabi/
---
tmpdir: !file
  path: tmpdir
  logfile: !fd stdout
  nthreads: !!int 8
EOF
```

CloudABI

- YAML file allows to:
- socket
  - bind: 0.0.0.0:12345
  - bind: /unix/domain/socket
- fd
  - stdout
  - stderr
- file
  - path [filename]

CloudABI

- Cloudlibc
  - removes function consider insecure like gets(3) or strcpy(3)
  - only capsicum friendly functions
    - removes open(2), stat(2), wait(2), etc.
    - allows pdfork(2), openat(2), etc.
- compilation checks, not runtime checks

Casper

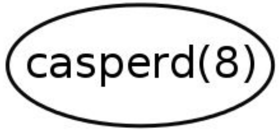
Casper

Provides functionalities which are not available in capability mode through convenient APIs making Capsicum more practical.

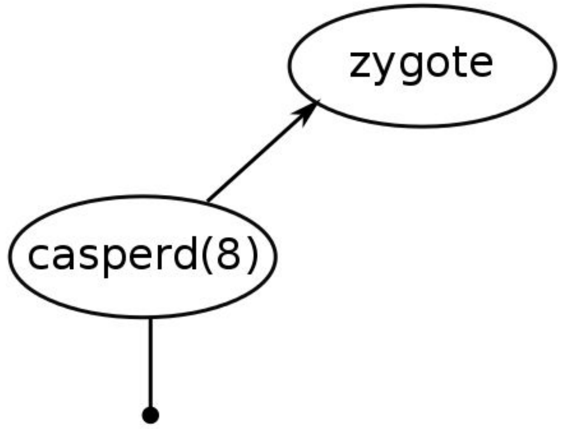
Casper - daemon approach

- *casperd(8)*
- libnv as IPC
- services
- */etc/casper* - list of services
- *libcapsicum* - IPC library
- *libcasper* - services library

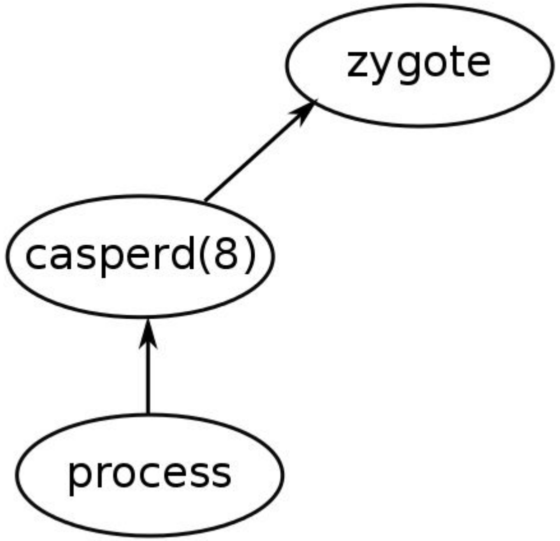
Casper - daemon approach



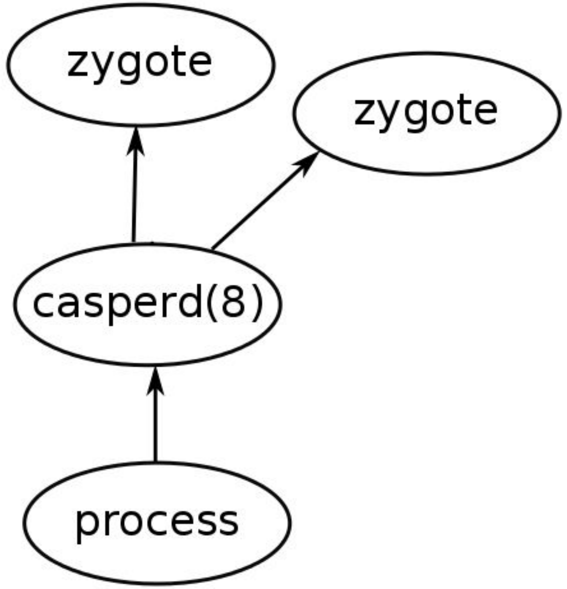
Casper - daemon approach



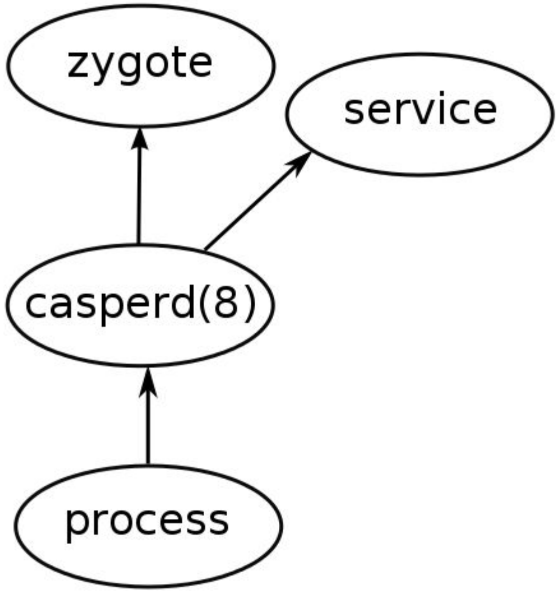
Casper - daemon approach



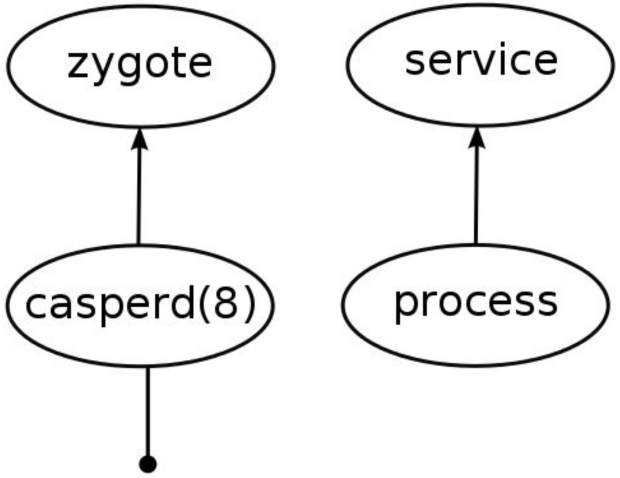
Casper - daemon approach



Casper - daemon approach



Casper - daemon approach



Casper - issues

Service workers are children of the Casper daemon

- different credentials
- different resource limits
- different working directory
- different umask
- different MAC labels

Casper - issues

- different cpu set
- different process group and tty
- different /dev/std{in,out,err} and /dev/fd/*

```
$ diff -du <(cat a) <(cat b)
--- /dev/fd/11
+++ /dev/fd/13
```

Casper - issues

- different routing table *(setfib(1))*
- harder to audit/ktrace
- one point of failure

Casper - solution?

- Create new syscall to copy all settings of a process
- Allow to copy them over Unix Domain
- Available only by root
- What with descriptors?

Process descriptors

- pdfork(2)
- Capsicum friendly
- Can be monitored by kqueue(2), select(2) or poll(2)
- Still waiting for pdwait(2)
- wait(2) called with -1 ignores process descriptors
- close(2) will terminate child

Casper - the new architecture

service workers are children of **the actual process**

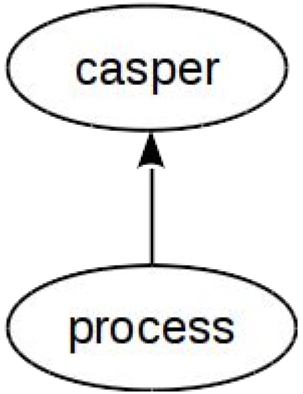
- pdfork(2)
- Reduce the number of modules
  - libcasper
  - services
- Dynamic linking
- API did not changed

Casper - problems and limitations

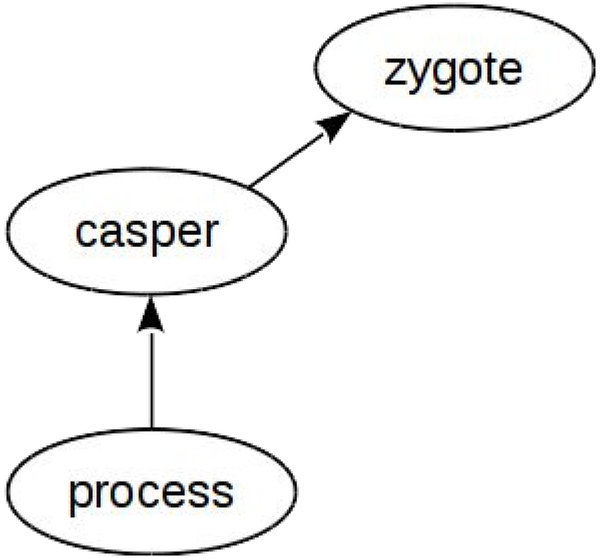
- changing capabilities, credentials etc.
- unable to globally shutdown Casper

Casper - fork approach

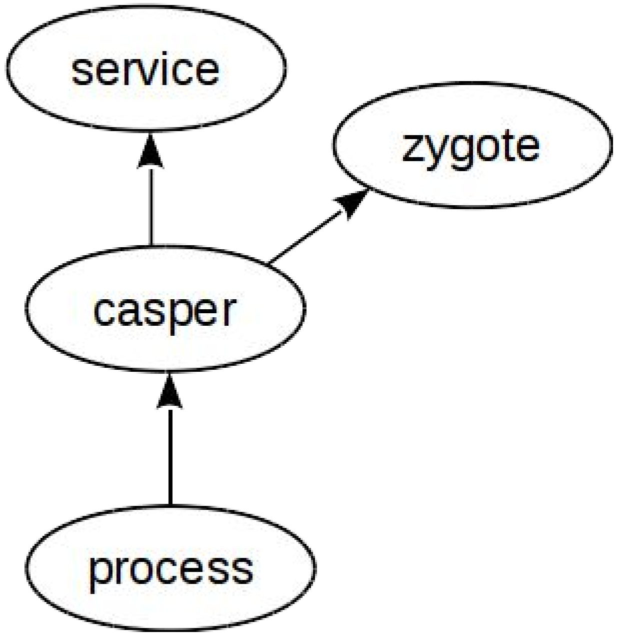
Casper - fork approach



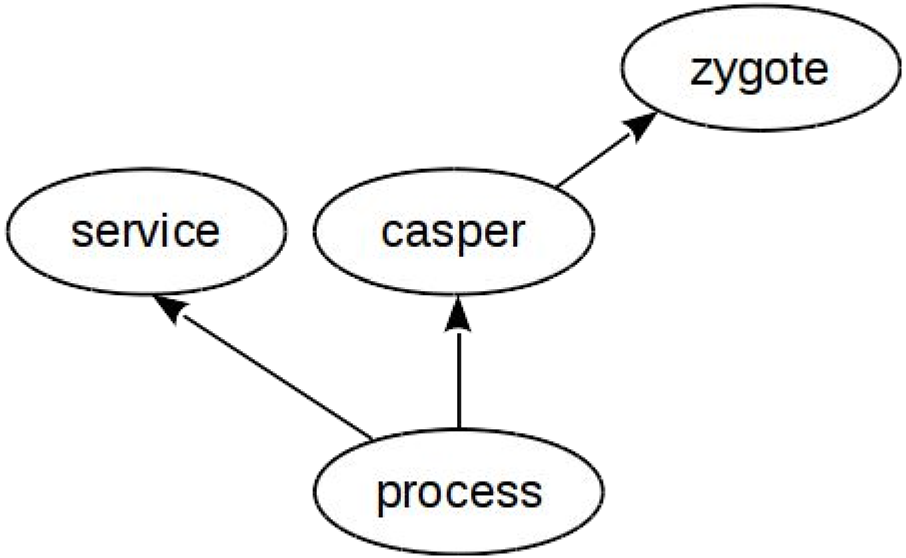
Casper - fork approach



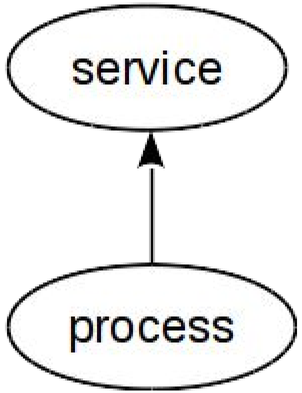
Casper - fork approach



Casper - fork approach



Casper - fork approach



Casper services

- system.dns
- system.grp
- system.pwd
- system.random
- system.sysctl

Casper usage - 1/2

```
#ifdef HAVE_LIBCASPER
        cap_channel_t *capcas, *capdnsloc;
        const char *types[1]
        int families[2];

        capcas = cap_init();
        if (capcas == NULL)
                goto out;
        capdnsloc = cap_service_open(capcas, "system.dns");
        /* Casper capability no longer needed. */
        cap_close(capcas);
        if (capdnsloc == NULL);
                error("unable top open system.dns");
        /* Limit system.dns to reverse DNSlookups. */
        types[0] = "ADDR";
        if (cap_dns_type_limit(capdnsloc, types, 1) < 0)
                error("unable to  limit acces to system.dns service");
        families[0] = AF_INET;
        families[1] = AF_INET6;
        if (cap_dns_family_limit(capdnsloc, families, 2) < 0)
                error("unable to limit access to system.dns service");
#endif /* HAVE_LIBCASPER */
```

Casper usage - 2/2

```
#ifdef HAVE_LIBCASPER
        hp = cap_gethostbyaddr(capdns, (char *)&addr, 4, AF_INET);
#else
        hp = gethostbyaddr((char *)&addr, 4, AF_INET);
#endif
```

libcaspermock

- same API like Casper
- reduce need of doing checks in code

```
#ifdef HAVE_LIBCASPER
        hp = cap_gethostbyaddr(capdns, (char *)&addr, 4, AF_INET);
#else
        hp = gethostbyaddr((char *)&addr, 4, AF_INET);
#endif
```

Future goals

- lower the bar for the new Casper and Capsicum consumers
- publish the system.filesystems or similar services which allow to interact with path namespace
- Improve auditing

Thank you!

<m.zaborski@wheelsystems.com>

<oshogbo@FreeBSD.org>