Privilege Separation and Pledge

- Theo de Raadt
  OpenBSD
Many small changes to improve security

<table>
<thead>
<tr>
<th>Category</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application software (ports)</td>
<td>(Educating upstream about better practices)</td>
</tr>
<tr>
<td>Own Applications: design &amp; architecture</td>
<td></td>
</tr>
<tr>
<td>(Privilege Separation, Privilege Drop, auditing, ...)</td>
<td></td>
</tr>
<tr>
<td>Address Space and other resources</td>
<td>(ASLR, W^X, cookies, ...)</td>
</tr>
<tr>
<td>Libraries (especially libc)</td>
<td>(strlcpy, arc4random, strict malloc, auditing, ...)</td>
</tr>
<tr>
<td>System call interface</td>
<td>(pledge)</td>
</tr>
<tr>
<td>Kernel</td>
<td>(Some ASLR, W^X, ...)</td>
</tr>
<tr>
<td>Hardware and BIOS</td>
<td>(cry into our beer...)</td>
</tr>
</tbody>
</table>
Privilege Separation

A design pattern — splits a program into processes performing different sub-functions

Each process is designed to operate in a separate security domain

Processes cooperate over pipes using some protocol

Subset of “sandboxing” concept
Separated at birth

(Our own ntpd as an example)

Master process
Runs as root, only does settimeofday()

DNS Servicer
Does DNS lookups

Internet Speaker
Speaks NTP to Internet
Privilege Separation examples

The original 3:

Qmail
Postfix
OpenSSH

And…. Chrome
Defence in Depth

We designed & modified many more programs to use this design pattern

Experience gained with **60 more programs!!!**

Routing daemons, Mail daemons, dhcp tools, tcpdump…

Let’s build a mechanism which enforces security domains!
Major ones..

bgpd, dhclient, dhcpd, dvmrp, eigrpd, file, httpd, iked, ldap, ldpd, mountd, npppd, ntpd, ospf, ospf6, pflogd, radius, relayd, ripd, script, smtpd, syslogd, tcpdump, tmux, xconsole, xdm, X server, ypldap, pkg_add
Pledges are POSIX subsets

Pledge syscall requests that only (a carefully selected) subset of POSIX functionality be permitted

Subsets such as: stdio rpath wpath cpath fattr inet dns getpw proc exec sendfd recvfd ...

Deep functional support in the kernel — more sophisticated than "seccomp"
Privsep – enforce with Pledge

(Our own ntpd as an example)
Processes select own pledge – inline

"I pledge this is the only subset of POSIX I will use"

Make the promise in the code when ready.

```c
imsg_init(ibuf_dns, pipe_ntp[1]);

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

while (quit_dns == 0) {
```

Cannot undo the promise…
Good debugging experience

Most violations result in process being killed

```
234    prog    CALL   socket(AF_LOCAL, 0x1<SOCK_STREAM,0)
234    prog    PLDG  socket, "inet", errno 1 Operation not permitted
234    prog    PSIG   SIGABRT SIG_DFL
234    prog    NAMI   "prog.core"
```

core is dumped — go ahead use gdb
Privsep mistakes identified

Implementation errors found in 10% of privsep programs

Sub-processes did actions beyond design rule! tsk tsk.

ntpd, bgpd, tcpdump, …

Validate program operation matches design rule
Future work

OpenSSH privilege separation is dated, and could be improved...

Continue refining semantics

Cooperate if another OS wants pledge

Observe impact on upstream software, and assist
General Observation

Perfection is impossible to achieve unless an enforcement mechanism keeps us honest.