

Pledge, and Unveil, in OpenBSD

Bob Beck
beck@openbsd.org

Pledge: Realistic subsets of POSIX functionality

- Named subsets "dns" "stdio" "wpath" "rpath"
- Easy to learn to add to programs
- No Subtle Behaviour changes
- No error returns
- Illegal operations crash the program. (SIGABRT)

Pledge Helps Privsep

- Privsep programs start as root, split up into different users and processes for different functions – often contained with chroot, communicate via pipe.
- Consider OpenBSD ntpd – 3 processes
- Ntp – pledges “stdio inet”
- DNS – pledges “stdio dns”
- Master pledges “settime”
- Each pledge matches function – makes it safer
- A TLS https connection for NTP constraints is made in the internet speaker without any filesystem access at all (certificates loaded in memory before pledge)

Pledge makes privdrop safer

- Privdrop programs start as root and then drop privs to reduce further risk
- Ping is the classic example – starts with setuid root, opens the raw socket, drops back to the invoking user.
- On OpenBSD – it also pledges “stdio inet dns”
- Now much safer for the invoking user too!

Pledge brings privdrop to non setuid programs

- Programs that do NOT start as root/setuid can not privdrop
- Consider a program run by normal users - nc
- OpenBSD nc is a swiss army knife of ways to connect to things
- Starts with a generous pledge
- Reduces according to what operation is being done
 - Also ends up at “stdio inet” for the typical case of tcp communication
- “A different pledge for each blade”

Chrome

Chrome is already designed for privsep with 5 types of processes on OpenBSD it is pledged

- GpuProcess → “stdio drm prot_exec recvfd sendfd”
- RenderProcess → “stdio rpath flock prot_exec recvfd sendfd ps”
- ApilnProcess → “stdio prot_exec recvfd sendfd”
- ApiPluginProcess → “stdio prot_exec recvfd sendfd”
- UtilityProcess → "stdio rpath cpath wpath fattr sendfd recvfd"

But chrome is not setuid – can't contain them with chroot. Pledge takes syscalls away – but not filesystem access – it could still go after my ssh keys.

Unveiling Unveil

- Limit filesystem access
- `unveil(const char *path, const char *flags);`
- Access to path that is not unveiled returns `ENOENT`
- Flags → “`rwxc`”
- Access to an unveiled path disallowed by flags returns `EACCESS`

Unveil

- Unveiling a directory unveils everything underneath it in the filesystem
- Unveiling a non directory unveils by name in the containing directory
- Deep support in kernel in name lookup, saving directory vnodes, associated names
 - Keep lookup costs largely in the unveil'ing process.
- New pledge for unveil

Very usable in OpenBSD base

- 40 odd programs, /sbin utilities, ftp, syslogd, spamd, bgpctl ...
- Many very simple: unveil("/dev", "rw")
- Often right before pledge.
- Not quite at the holy grail yet...

When will it be available?

- Probably one more refactor of internal implementation.
- Theo's magic 50 programs rule
- We know it's right when we can do chrome.
- Probably for OpenBSD 6.4

Pledge Exec Promises

- Supported today, but evolving.
- Specify pledge promises for future exec'ed children
- Not easy to use today
- Really needs to be combined with unveil
- More experience with real programs and unveil should lead us to a better semantic here.


Unveil, Execpromises, Then?

Stargazing down the road we know we really have it right when we can usefully provide
pledge(1)

pledge [[-p path]...] [-P promises] command [arguments]

- Pledge make build?
- Such things are probably a little ways away. But it's nice to have goals.

It's all about making big things safer

- Pledge and Unveil are designed to make big programs safer
- Applies to multiple different sorts of programs, privsep, privdrop unprivileged
- Easy to use and learn compared to many other techniques
- Designed to make even complex things safer! OpenSSH, httpd, acme-client, chrome – the stuff we use frequently
- Even openssl(1)  is pledged on OpenBSD

Questions?
#AMA