Namespaces for security

Jake Edge, LWN.net, jake@lwn.net
Embedded Linux Conference, San Francisco
February 21, 2013

What are we going to be talking about?

- Threats
- Effects
- Defenses
- Namespaces
- Types of namespaces
- Creating namespaces
- Using namespaces
- Examples

What kinds of threats are we talking about?

- Mass attacks
- Network-facing services
- Network clients
- DNS cache poisoning
- Web application flaws
- Cross-site attacks
- ...

What are the effects of typical attacks?

- Service account compromise
  - Can perform any action service could do
    - Network, filesystems, processes
- Network access
  - Spam, DDoS, Botnet
- Filesystem access
  - Confidential information, config settings
- Process access
  - `ptrace(), kill()`
- Privilege escalation

How do we normally avoid those threats?

- Unix permissions
- Users and groups
- Mandatory access control (MAC)
- Capabilities (`CAP_SYS_ADMIN, CAP_NET_ADMIN, ...`)
- Seccomp sandbox
- ...

Namespaces

- Mechanism to partition global resources
- Provides invisibility
- Lightweight virtualization
- Containers
- Testing, debugging
- Security

Types of namespaces

- UTS – Unix timesharing (host and domain name)
- Mount
- Processes (PID)
- Inter-process communication (IPC)
- Networking
- User

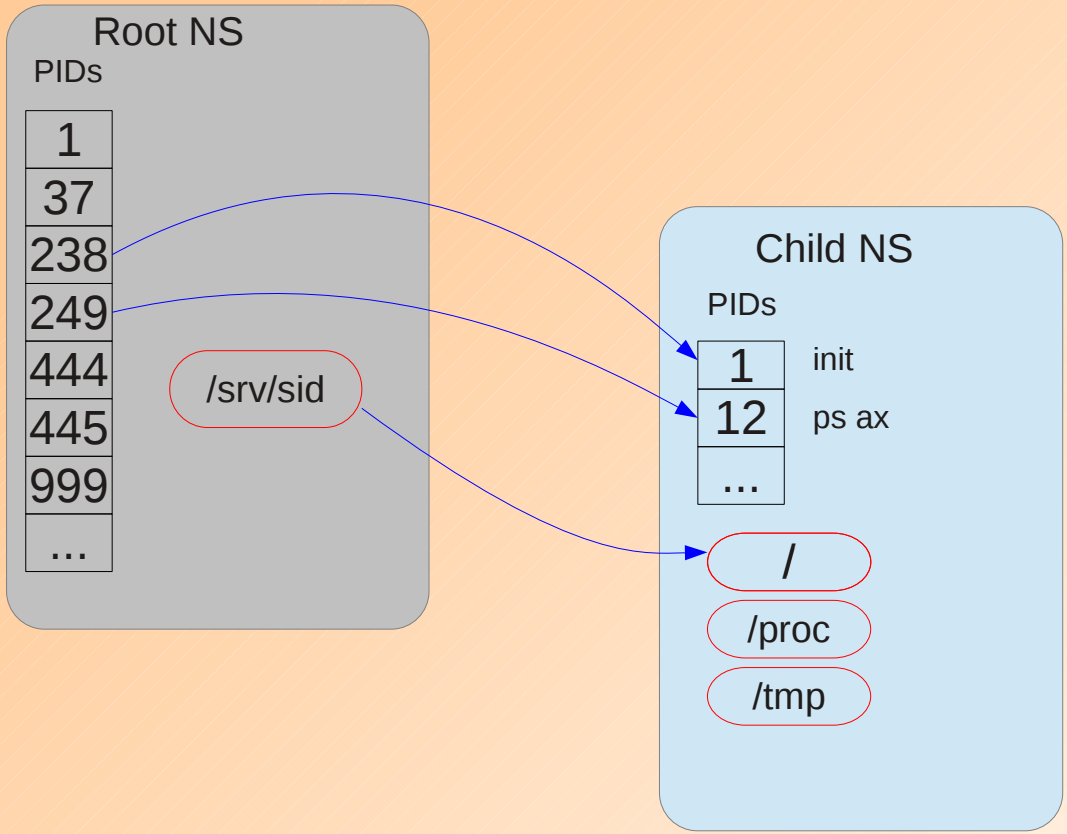
Namespace kernel configuration

- General setup → Namespaces support
- `CONFIG_NAMESPACES, CONFIG_UTS_NS, CONFIG_NET_NS, ...`
- As of 3.8, `CONFIG_USER_NS` depends on network filesystems being turned off

Creating namespaces

- `clone(), unshare(), setns()` system calls
- `CLONE_NEWNS, CLONE_NEWUTS, CLONE_NEWPID, CLONE_NEWNET, CLONE_NEWIPC, CLONE_NEWUSER`
- `clone()` - starts a new process in new namespace(s)
- `unshare()` - creates new namespace(s) without a new process, adds current process to them
- `setns()` - join an existing namespace
- `systemd-nspawn` – useful for noodling with namespaces, source code is useful too

PID and mount namespaces



Using namespaces

- `/proc/PID/ns/{mnt pid uts ipc net user}`
- References the namespaces
- Can be passed to `setns()`

Mount namespace propagation

- Shared, slave, and private mounts

```
# mount --make-shared /
```

```
# mount --make-private /
```

- Recursive variants

```
# mount --make-rslave /
```

- Where do further mounts appear?
- Shared shares both directions, slave just in that direction, private doesn't share at all

Examples

- Set up mount namespace to run update checker, allow RO access to libraries it needs and have private `/tmp`
- Run multiple instances of web application in separate PID namespaces – can't see others
- Combine mount and PID namespace to isolate web application (CMS in PHP, say) further
- Set up a network namespace to run httpd worker process – no access to the network if process is compromised
- Separate network namespaces for local network access vs. internet access – internet-based compromise can't access LAN
- ...

Further reading

- Namespaces in operation series
  - http://lwn.net/Articles/531114/
- Slides available on ELC site and at
  - http://lwn.net/talks/elc2013/