SINE NOMINE
ASSOCIATES

OpenAFS client for macOS

Marcio Barbosa

2021 OpenAFS Workshop

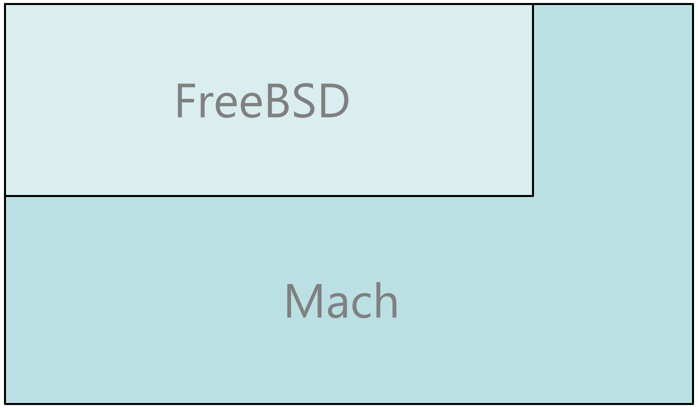
AGENDA

- A high-level view of XNU
- Kernel Extensions
- Securing Modular Architecture
- System Extensions
- Apple Silicon
- Conclusion
- References / Contact

A HIGH-LEVEL VIEW OF XNU

A HIGH-LEVEL VIEW OF XNU

- The Mac OS X kernel is called XNU.
- Stands for X is Not UNIX.
- Microkernel architecture? No, XNU is a hybrid kernel.



MONOLITHIC KERNELS

- "Classic" kernel architecture.
- Predominant in the UNIX and Linux realms.
- All kernel functionality in one address space.
- If any service fails, the whole system crashes.
- Hard to extend.

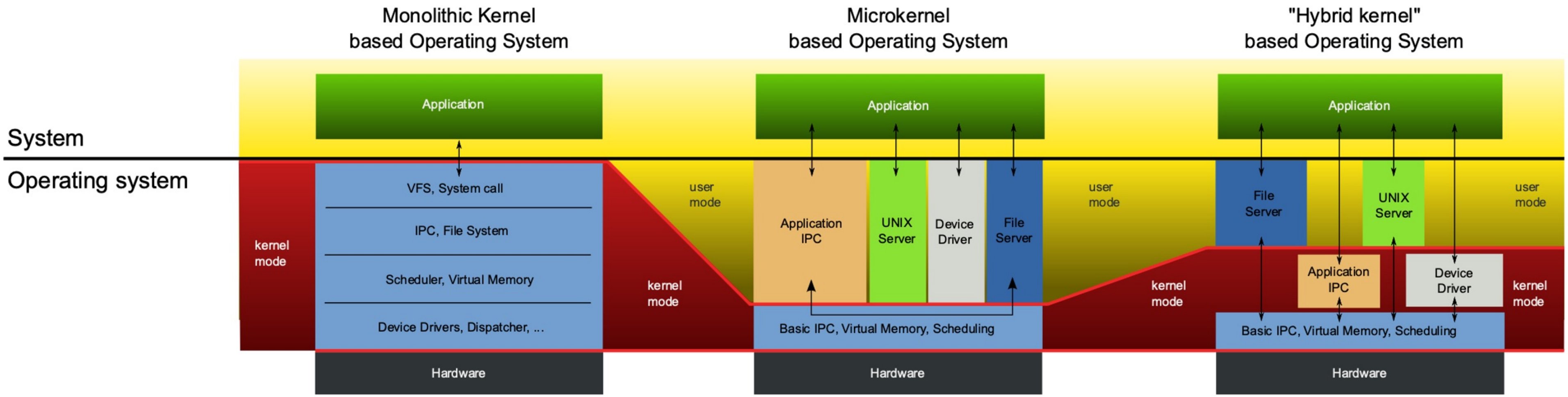
MICROKERNELS

- Consists of only the core kernel functionality.
- The rest of the functionality exported to external servers.
- There exists complete isolation between the individual servers.
- Communication between them is carried out by message passing.
- Failure is contained.
- Monolithic kernel failures usually trigger a complete kernel panic.
- Performance can be an issue.

HYBRID KERNELS

- Hybrid kernels attempt to synthesize the best of both worlds.
- The innermost core of the kernel is self-contained.
- All other services are outside this core, but in the same memory space.
- XNU is a hybrid.
- The kernel is modular and allows for pluggable Kernel Extensions.
- Absence of isolation exposes the system to bugs introduced by KEXTs.

MONOLITHIC, MICROKERNELS, AND HYBRID



Golftheman, Public domain, via Wikimedia Commons
https://commons.wikimedia.org/wiki/File:OS-structure2.svg

KERNEL EXTENSIONS

KERNEL EXTENSIONS

- No kernel can completely accommodate all the hardware, peripheral devices, and services available.
- KEXTs are kernel modules, which may be dynamically inserted or removed on demand.
- Augments kernel functionality with entirely self-contained subsystems.
- Runs in kernel mode, and therefore has full access to kernel space.
- Kernel modules offer power, but they pose an even greater risk.
- Stability and the security of the entire Operating System can be compromised.

SINE NOMINE ASSOCIATES

SECURING MODULAR ARCHITECTURE

CODE SIGNING

- On macOS, GateKeeper requires application bundles to be signed.
- This has been a requirement since macOS 10.7 (Lion).
- On macOS 10.10, KEXT will not be loaded if it is not signed.
- Code signing cannot vouch for code purity of purpose, but it can validate the origin of the code.
- OpenAFS provides a set of scripts to build and sign the package.

NOTARIZATION

- Checked by Apple for malicious components.
- Notarization is not App Review.
- Introduced in macOS 10.14 (Mojave).
- Since April 7, 2019, all KEXTs must be notarized.
- OpenAFS provides a script to help users to notarize the package.

KEXT SECURITY REQUIREMENTS

- Extra security considerations must be enforced:
- Kexts must be owned by the uid of root, and the gid of wheel.
- Permissions on the directories must be at most 755 - that is, rwxrwxr-x.
- Any files in the Kext must be at most 644 (rw-r--r--).

SYSTEM EXTENSIONS

SYSTEM EXTENSIONS

- Kernel extensions can compromise the stability of the system.
- Still, extending the system is one of the key features of any OS.
- How can we extend the system without compromising its stability?
- Apple introduced System extensions on macOS 10.15 (Catalina).
- Allows the extension of the OS without requiring kernel-level access.
- If a system extension crashes, the rest of the system is unaffected.

SYSTEM EXTENSION VS KERNEL EXTENSION

- Kernel extensions and system extensions serve the same purpose.
- Kernel Extensions let developers load code directly into the kernel.
- Challenges in terms of development, security, and stability.
- System extensions don't compromise the security or stability of macOS.
- An effort by Apple to phase out the use of KEXTs.

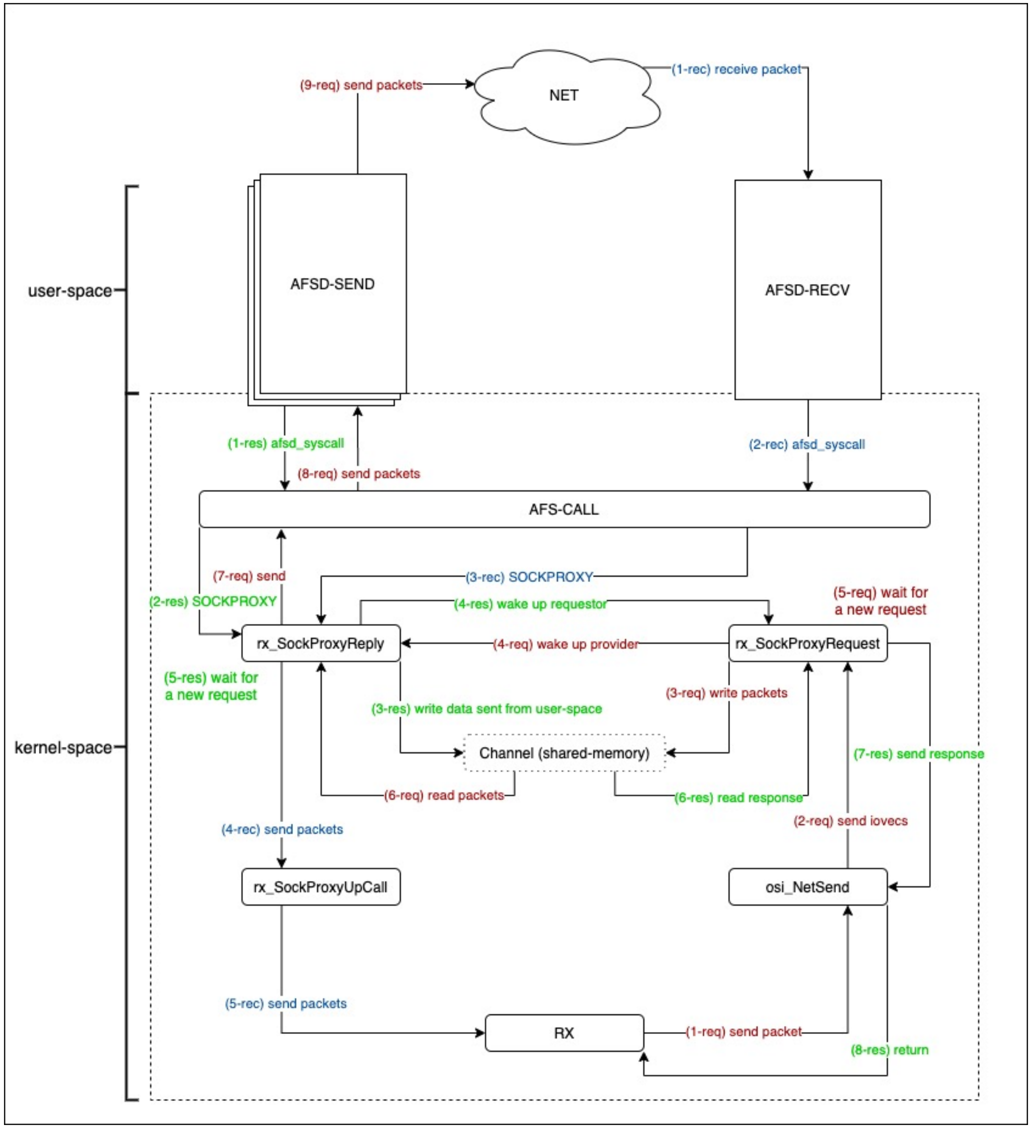
DEPRECATED KERNEL EXTENSIONS

- Apple deprecating macOS kernel extensions.
- In macOS 11, KEXTs using certain deprecated KPIs no longer load.
- Network kernel extensions are deprecated.
- No longer provides an in-kernel socket KPI, extensively used by RX.

SOCKPROXY

- Move pieces of RX to user-space.
- Background daemons responsible for sending and receiving packets on behalf of RX.

- Challenges:
  - Increased complexity.
  - Performance.
  - Room for improvement.



NET

(9-req) send packets

(1-rec) receive packet

user-space

AFSD-SEND

AFSD-RECV

kernel-space

(1-res) afsd_syscall

(8-req) send packets

(2-rec) afsd_syscall

AFS-CALL

(7-req) send

(2-res) SOCKPROXY

(3-rec) SOCKPROXY

(4-res) wake up requestor

(5-req) wait for a new request

rx_SockProxyReply

(4-req) wake up provider

rx_SockProxyRequest

(5-res) wait for a new request

(3-res) write data sent from user-space

(3-req) write packets

Channel (shared-memory)

(7-res) send response

(6-req) read packets

(6-res) read response

(4-rec) send packets

(2-req) send iovecs

rx_SockProxyUpCall

osi_NetSend

(5-rec) send packets

RX

(1-req) send packet

(8-res) return

APPLE SILICON

APPLE SILICON

- Transitioning away from Intel chips to Macs built with its own Apple silicon chips.
- macOS 11 (Big Sur) is equipped with tools to help developers with this transition.
- Back in 2004, Apple moved from PowerPC to Intel x86.
- Users can run Intel apps even if those apps haven't been updated, thanks to Rosetta 2.
- Rosetta 2 cannot translate kernel extensions.

ARM64E

- All KEXTs must support the arm64e architecture.
- Pointer Authentication:
  - Detect and guard against unexpected changes to pointers in memory.
- Variadic Functions
  - x86_64 and arm64 architectures have different calling conventions.

UNIVERSAL BINARIES

- A universal binary looks no different than a binary built for a single architecture.
- Binaries for both architectures merged into a single binary.
- We can create an OpenAFS Universal package:

CREATING AN UNIVERSAL OPENAFS CLIENT

- Cherry-pick the relevant patches from the following branch:
  - github.com/marciobarbosa/openafs-dev/commits/mbarbosa/macos-universal-1
- Run the following commands:

  -
    ```
    $ ./regen.sh
    ```

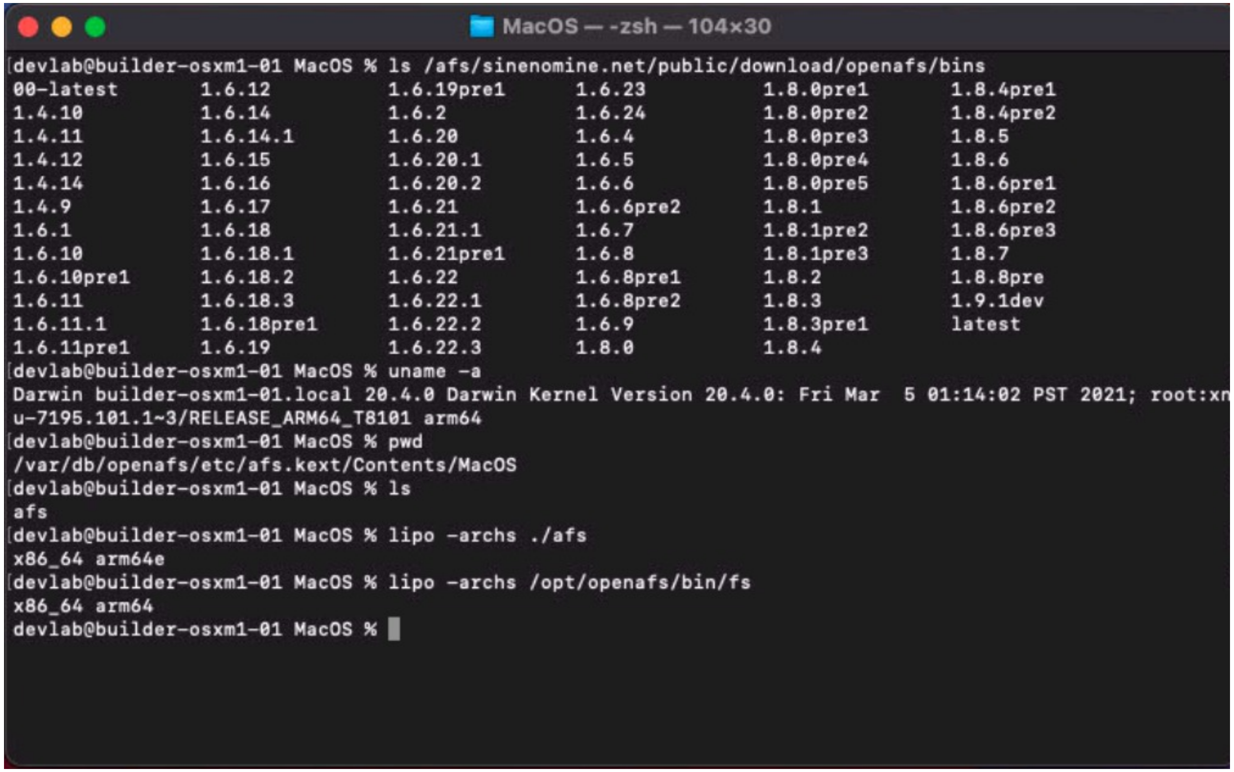
  -
    ```
    $ ARCHFLAGS="-arch x86_64 -arch arm64" ./configure --enable-transarc-paths \
    --with-krb5-conf=/usr/bin/krb5-config --libdir=/Library/OpenAFS/Tools/lib
    ```

  -
    ```
    $ ARCHFLAGS="-arch x86_64 -arch arm64" make dest
    ```

  -
    ```
    $ sudo sh src/packaging/MacOS/pkgbuild.sh -x --csdb <path/to/CellServDB>
    --app-key 'Developer ID Application: <your_dev_id_application>’
    --inst-key 'Developer ID Installer: <your_dev_id_installer>' <arch>/dest
    ```

  -
    ```
    $ sudo ./src/packaging/MacOS/notarize.pl <apple_id> <password> </path/to/dmg>
    ```

CREATING AN UNIVERSAL OPENAFS CLIENT



```
[devlab@builder-osxm1-01 MacOS % ls /afs/sinenomine.net/public/download/openafs/bins
 00-latest       1.6.12          1.6.19pre1      1.6.23          1.8.0pre1       1.8.4pre1
 1.4.10          1.6.14          1.6.2           1.6.24          1.8.0pre2       1.8.4pre2
 1.4.11          1.6.14.1        1.6.20          1.6.4           1.8.0pre3       1.8.5
 1.4.12          1.6.15          1.6.20.1        1.6.5           1.8.0pre4       1.8.6
 1.4.14          1.6.16          1.6.20.2        1.6.6           1.8.0pre5       1.8.6pre1
 1.4.9           1.6.17          1.6.21          1.6.6pre2       1.8.1           1.8.6pre2
 1.6.1           1.6.18          1.6.21.1        1.6.7           1.8.1pre2       1.8.6pre3
 1.6.10          1.6.18.1        1.6.21pre1      1.6.8           1.8.1pre3       1.8.7
 1.6.10pre1      1.6.18.2        1.6.22          1.6.8pre1       1.8.2           1.8.8pre
 1.6.11          1.6.18.3        1.6.22.1        1.6.8pre2       1.8.3           1.9.1dev
 1.6.11.1        1.6.18pre1      1.6.22.2        1.6.9           1.8.3pre1       latest
 1.6.11pre1      1.6.19          1.6.22.3        1.8.0           1.8.4
[devlab@builder-osxm1-01 MacOS % uname -a
 Darwin builder-osxm1-01.local 20.4.0 Darwin Kernel Version 20.4.0: Fri Mar  5 01:14:02 PST 2021; root:xn
 u-7195.101.1~3/RELEASE_ARM64_T8101 arm64
[devlab@builder-osxm1-01 MacOS % pwd
 /var/db/openafs/etc/afs.kext/Contents/MacOS
[devlab@builder-osxm1-01 MacOS % ls
 afs
[devlab@builder-osxm1-01 MacOS % lipo -archs ./afs
 x86_64 arm64e
[devlab@builder-osxm1-01 MacOS % lipo -archs /opt/openafs/bin/fs
 x86_64 arm64
 devlab@builder-osxm1-01 MacOS % ▌
```

SECURITY POLICIES

- Apple introduced more security policies for Kernel Extensions.
- Strictest rules come with M1 Macs.
  - New platform security setting which blocks the loading of all third-party KEXTs by default.
- In macOS 11, the loading of KEXTs requires extra actions.
- Apple Silicon:
  - Reboot your Mac with Apple silicon into Recovery mode.
  - Set the security level to Reduced security.
  - Allow the loading of third-party KEXTs.
  - Reboot back to macOS.

SECURITY POLICIES

- Intel and Apple Silicon:
  - Open the Security & Privacy System Preferences.
  - Authenticate to make changes.
  - Allow the system to load your KEXT.
  - Wait for the system to load the KEXT and rebuild the auxiliary KEXT collection.
  - Reboot to load the new auxiliary KEXT collection.

SECURITY POLICIES

- There are three security policies for a Mac with Apple silicon:
  - Full Security: The system behaves like iOS and iPadOS and allows only booting software which was known to be the latest that was available at install time.
  - Reduced Security: This policy level allows the system to run older versions of macOS. Because older versions of macOS inevitably have unpatched vulnerabilities, this security mode is described as Reduced. This is also the policy level required to support booting kernel extensions (KEXTs) without using a mobile device management (MDM) solution and Automated Device Enrollment with Apple School Manager or Apple Business Manager.
  - Permissive Security: This policy level supports users that are building, signing, and booting their own custom XNU kernels. System Integrity Protection (SIP) must be disabled before enabling Permissive Security mode. For more information, see System Integrity Protection in Apple Platform Security.

CONCLUSION

- Apple has confirmed that future Mac computers powered by its own chips will not support kernel extensions at all.
- "This is why developers are being strongly encouraged to adopt system extensions before kernel extensions support is removed from macOS for future Mac computers with Apple silicon." Apple Platform Security May 2021.
- Kernel programming interfaces (KPIs) will be deprecated as alternatives become available.
- No concrete deadline has been announced.

CONCLUSION

- For now, a complete transition to System Extensions isn't possible.
  - Some KEXTs that use KPIs do not yet have System Extension alternatives.
- No alternative system extension API to develop file systems.
  - NSFileProvider?
  - Fuse-like alternatives?
- OpenAFS is still alive!

Thank you!

REFERENCES / CONTACT

- References:
  - Mac OS X and IOS Internals: To the Apple's Core - Jonathan Levin.
  - OpenAFS code.
- Contact:
  - mbarbosa@sinenomine.net
- Questions?