OPNsense®

Securing networks made easy

Tooling Around With FreeBSD

A tale of scripting a custom firewall distribution

About Franco

- From Leipzig, Germany
- Studied computer science
- Worked on several firewall/network appliances based on Ubuntu, FreeBSD and OpenBSD
- DragonFlyBSD alumni
- Worked on mandoc, dhclient, netmap, FreeBSD ports
- OPNsense as a volunteer since 2014, joined OPNsense/Deciso full time in 2021
- Contact: franco@opnsense.org

About OPNsense

- "a pretty GUI" since 2015
- Over ¼ million actively updated installs
- Based on pfSense/m0n0wall, BSD licensed
- MVC/API capabilities and plugins collection
- PHP, Python and POSIX shell
- As close to stock FreeBSD as possible
- 2 fixed major releases per year, bi-weekly stable updates
- Suricata integration and port maintenance
- OpenBSD dhcrelay to FreeBSD since ISC is EoL (PR: 281584)
- Maintaining dhcp6c from dhcp6 a.k.a. wide-dhcpv6

About this talk

- Informative and entertaining
- Not in-depth technical
- Case study, not a how-to guide
- Not chronological, mostly logical order
- OPNsense solutions to FreeBSD challenges
- FreeBSD benefits from OPNsense challenges

Talk available at

https://pkg.opnsense.org/talks/

Ok, let’s go!

Historic perspective

- Code work started in late 2014
- First release in January 2015
- Poudriere was still young back then
- pkg(ng) was gaining traction
- Quarterly did not exist
- pkg-base was nowhere near ready
- FreeBSD used SVN, pfSense used Git

Problem scope

- A distribution with a base and kernel, hundreds of packages (direct and indirect dependencies), custom packages and several images
- Source and ports patches were not applied to their respective repository and instead lived in the build repository as actual patch files which started to fail to apply on related upstream changes
- Interactions of builds with the host build system
- The build tools were using FreeSBIE (last release in 2007)

Motivation

”If something requires more than 90 seconds of time, write a script to automate that."

- Easy to use, traceable, chroot'ed, save-point-based and open
- Add minimal shell scripting around the existing src and ports repositories
- Rewrite build tools from scratch to avoid FreeSBIE
- Use git exclusively and fill the gap with git-svn if needed
- Use pkg as the main tool both in building images as well as updating systems
- Only handle a subset of ports to maintain and build but provide the whole ports tree nonetheless
- Fast deployment and testing with user base which requires more user facing tools that handle individual components

Terminology

- *tools*: build tools gluing everything together
- *src*: slightly modified FreeBSD src repo
- *ports*: slightly out of sync FreeBSD ports repo
- *core*: provides a functional backend and GUI/API to use as a firewall / networking device
- *plugins*: optional GUI/API packages
- *update*: script collection to update and modify the other components

https://github.com/opnsense

Here is what happened…

Tools reborn (1)

- Ended up using Makefile dependencies exactly like FreeSBIE after figuring out how it actually worked ¯_(ツ)_/¯
- Shell scripts underneath to generate the actual file sets used as safe-points for progress
- Hold custom configurations across different versions for all components
- Use individual build steps to construct composite build steps
- Reuse progress and provide prebuilt components for fast (and perhaps custom) image build
- Document all of it: https://github.com/opnsense/tools/blob/master/README.md

Tools reborn (2)

- Getting it set up:
  - # pkg install -y git && cd /usr && git clone https://github.com/opnsense/tools && cd tools && make update
- Building an image:
  - # make dvd

Side quest: opnsense-update

- wrap pkg-upgrade and handle multiple core packages as alternatives (community, business, development)
- emulate freebsd-update, but created with a standard FreeBSD build targets?
- cross-ABI update capability, but offline!
- handle multiple mirrors and multiple repositories gracefully
- The package mirror has the “latest” link but also offers raw directory access to each stable release package repository
- Examples:

    Install the debug kernel: # opnsense-update -zkr dbg-24.7.4

    Do a major upgrade: # opnsense-update -u

    Do a minor upgrade: # opnsense-update -bkp

Source tree

- Decision to reuse of base.txz and kernel.txz as a primary source for these components
- Today we still strip a few components to use their ports counterparts: KRB5, Unbound, OpenSSH, OpenSSL(*)
- Kernel is built on top of GENERIC with minor changes for RSS and a few drivers for legacy reasons
- What about signatures for both base and kernel archives plus additional mtree files?
- Embed version and build metadata into the archives
- Base.txz also needed a way to handle obsolete files removal
- Debug kernel build and automatic handling (textdump vs. vmcore)

Side quest: opnsense-sign / opnsense-verify

- pkg-bootstrap uses an ominous pkg.txz.sig?
- Take the pkg-bootstrap code and make it verify arbitrary files
- Create signatures and fingerprints from pkg signatures
- Hooks into pkg fingerprints of active package repository
- Lists active repositories and their priority
- A canonical way to get ${ABI} expanded for scripting
- Use it in opnsense-update
- Example:
  - # opnsense-verify kernel-dbg-24.7.4.txz && echo verified ok

Ports tree

- Needed a way to use git -- a repo copy was available, but we ended up using a separate tree for maintenance purposes
- "Skim" tool to bring in a port and all its dependencies from a directory structure so that it was CVS-agnostic reusing a given ports.conf stored in the tools anyway
- Build ports, retain package progress, only keep relevant packages in a resulting packages archive
- But Go and Rust... auxiliary package dependencies are built and kept but removed for the final build pass
- Distfile cache as an optional archive file and build step
- Build a clean chroot for the build
- Optionally check and rebuild packages which have older versions than currently built and other development/QA/debug tools
- Audit build step to check for vulnerabilities in the packages set

Side quest: opnsense-revert

- Revert given packages to a previous package build state
- Requires to extend Latest/ directory to offer signatures for all packages intead of only pkg.txz
- Requires mirror to hold all stable release package repositories, but manageable given that we do not build all the ports
- Examples:

revert to a previous package:

opnsense-revert -r 24.7.1 dhcp6c

development snapshot of mpd5 for testing:

opnsense-revert -z mpd5

Custom packages

- Not wanting to taint the ports tree with actual code
- Not to overcomplicate builds by adding more external repo references
- "plugins” look like ports which are basically packages with GUI pages and a dependency to the actual third-party software implemented
- “core” from yet another repository for existing layout reasons and to split work areas for core team and community contributors
- Adding packaging support to both these repositories so Makefiles imitate the bare minimum pkg-create behaviour
- Supporting multiple flavours ("variants” to avoid confusion) of the same package: development, community and business packages; cpu-microcode-amd and cpu-microcode-intel have different dependencies but share the same plugin code directory

Side quest: opnsense-patch

- Ship script fixes for core and plugin components using GitHub magic
- Fetch patch and apply it, error handling, cache patches
- Extend the patch scope to installer and update components
- Limit here is patching scripts, patching binaries is out of scope and better done with opnsense-revert
- Examples:

Patch default core component: # opnsense-patch 842075ca

Patch a preconfigured component: # opnsense-patch -c update ecad25653

Patch using full URL ("zero" config):

opnsense-patch https://github.com/opnsense/plugins/commit/4f9e0308

Image handling

- Support a variety of different images assembled from prebuilt base, kernel and packages sets
- Users can prefetch the sets from the mirror to skip all the time-consuming build steps
- Only takes 1-2 minutes to assemble images including all the components discussed
- Support bhyve tester, signatures, compression and distribution set helpers
- Per-device support if needed (Deciso offers pre-flashed ZFS installations on the native hardware)

And now what?

Real world problem (1)

https://forum.opnsense.org/index.php?topic=42373.0

Users report kernel panics from 24.7.1 to 24.7.2:

--- trap 0xc, rip = 0xffffffff804d7de7, rsp = 0xfffffe00715ddb20, rbp = 0xfffffe00715ddb40 ---
agp_close() at agp_close+0x57/frame 0xfffffe00715ddb40
giant_close() at giant_close+0x68/frame 0xfffffe00715ddb90
devfs_close() at devfs_close+0x4b3/frame 0xfffffe00715ddc00
VOP_CLOSE_APV() at VOP_CLOSE_APV+0x1d/frame 0xfffffe00715ddc20
vn_close1() at vn_close1+0x14c/frame 0xfffffe00715ddc90
vn_closefile() at vn_closefile+0x3d/frame 0xfffffe00715ddce0
devfs_close_f() at devfs_close_f+0x2a/frame 0xfffffe00715ddd10
_fdrop() at _fdrop+0x11/frame 0xfffffe00715ddd30
closef() at closef+0x24a/frame 0xfffffe00715dddc0
closefp_impl() at closefp_impl+0x58/frame 0xfffffe00715dde00
amd64_syscall() at amd64_syscall+0x100/frame 0xfffffe00715ddf30
fast_syscall_common() at fast_syscall_common+0xf8/frame 0xfffffe00715ddf30

Real world problem (2)

- The clue: current process = 31 (**zpool**)
- The actual change in /usr/local/etc/rc:

```
- export HOME PATH REQUESTS_CA_BUNDLE
+ ZPOOL_IMPORT_PATH=/dev
+ export HOME PATH REQUESTS_CA_BUNDLE ZPOOL_IMPORT_PATH
```

- Since the kernel was crashing on boot we could not undo the change with opnsense-revert or opnsense-patch
- Ended up building an image with the change reverted to verify that was the cause

Real world problem (3)

- Confirmed with the image agp(4) was crashing on ZFS systems, but the impact seemed very limited.
- Truss revealed the command “env ZPOOL_IMPORT_PATH=/dev zpool import –Na” would open every node in /dev, but since ZFS implementation was switched for FreeBSD 13 zpool-import did not do the right thing anymore since the path was not set by default when before it was.
- Disabling agp(4) in the kernel has no operational impact on the relevant devices exhibiting the panic.
- Opened a bug report in FreeBSD and let the user produce a core dump with a debug kernel using opnsense-update.
- The core dump at first glance revealed a memory corruption during agp_close()
- There must be a separate regression in the agp(4) driver?

Real world problem (4)

```
From dc2e20244c96db04c5699e4f17718f47411e6837 Mon Sep 17 00:00:00 2001
From: Mark Johnston <markj@FreeBSD.org>
Date: Thu, 29 Aug 2024 13:12:19 +0000
Subject: [PATCH] agp: Set the driver-specific field correctly

PR:             281035
Reviewed by:    mhorne
MFC after:      1 week
Fixes:          437ea82ce7fc ("agp: Handle multiple devices more gracefully")

(cherry picked from commit 12500c14281dc62ddeac4c5e1e6eabd1e380f11c)
---
 sys/dev/agp/agp.c | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

diff --git a/sys/dev/agp/agp.c b/sys/dev/agp/agp.c
index f7eb906fc57c..8db1e13f08de 100644
--- a/sys/dev/agp/agp.c
+++ b/sys/dev/agp/agp.c
@@ -254,7 +254,7 @@ agp_generic_attach(device_t dev)
        mdargs.mda_uid = UID_ROOT;
        mdargs.mda_gid = GID_WHEEL;
        mdargs.mda_mode = 0600;
-       mdargs.mda_si_drv1 = sc;
+       mdargs.mda_si_drv1 = dev;
        mdargs.mda_si_drv2 = NULL;

        unit = device_get_unit(dev);
```

Real world problem (5)

- 437ea82ce7fc was committed in 2021

Real world problem (5)

- 437ea82ce7fc was committed in 2021

```
From 4f8959b9f4bba6e2018594c12132cca10e9a4367 Mon Sep 17 00:00:00 2001
From: Warner Losh <imp@FreeBSD.org>
Date: Wed, 26 Feb 2020 19:39:52 +0000
Subject: [PATCH] Remove support for FreeBSD 4.x and earlier from agp driver

Compile tested only, but do we still need this driver?
```

Real world problem (5)

- 437ea82ce7fc was committed in 2021

```
From 4f8959b9f4bba6e2018594c12132cca10e9a4367 Mon Sep 17 00:00:00 2001
From: Warner Losh <imp@FreeBSD.org>
Date: Wed, 26 Feb 2020 19:39:52 +0000
Subject: [PATCH] Remove support for FreeBSD 4.x and earlier from agp driver

Compile tested only, but do we still need this driver?
```

- Now there is a fix and an answer to the question :)

Thank you for listening! <3