Own The Stack
FreeBSD from a vendor’s perspective

BSDCan 2022

By: Antranig Vartanian & Faraz Vahedi
**id antranigv**
- Antranig Vartanian
- Co-Founder & CEO @ illuria Security, Inc.
- Daemon @ Armenian BSD User Group
- Native Tongue: Elixir/Erlang, POSIX Shell
- Past: CTO, Systems Engineer
- Love: Unix and Film Photography
- Runs: Jabber.am
  Armenian Tech Forums
  Systems We Love -- Armenia

**id kfv**
- Faraz Vahedi
- Systems Engineer @ illuria Security, Inc.
- Into Compilers, Maths, Physics, and UNIX
- Operating Systems: BSD and illumos fams
- Main Language: C

https://antranigv.am

https://kfv.io/
Agenda

- What we do
- Choosing an operating system for your appliance
  - Why FreeBSD
- Development Flow
  - Git(ea), Build(bot), Package Poudriere, Ship
- Path to vendorship
- Unforeseen issues and corresponding solutions
  
  "These are not the tools you are looking for" -- Obi-Wan Freebie
- Conclusion
- Q&A
What we do

We create virtual minefields inside the infrastructure to Detect, Deceive, and Deter malicious actors.

illuria's deception technology uses decoys and lures to break the attackers' decision cycle, forcing them to reveal themselves.

*a.k.a Honeypots on Steroids!!!*
Choosing an operating system

- **Spoiler Alert: Most of the time it’s FreeBSD**
  - Unless: for embedded systems with specific drivers
  - Unless: Team wants $TOOL, not available on FreeBSD

- **Our needs:**
  - not PITA, always POLA
  - commercial friendly (BSDL?)
  - responsible community
  - single source of truth
  - easy to "replicate" and "own"
Choosing an operating system (cont.)

- **Linux**
  - GPL → hard to modify, legally
  - Changes blazingly fast → hard to maintain
  - Divided community → hard to get proper and quick answers

- **illumos**
  - Unknown to the majority of *our* team
  - Very different than Linux and *BSDs → harder to teach

- **FreeBSD**
  - We already knew the nuances…
  - Centralized community → We know where to ask what
  - Single solution for a single problem which has been around for years
Benefits of using FreeBSD (as a vendor)

- We came for the license, we stayed for the technology (and the community)
- All brilliant things are in the base
  - ZFS
  - DTrace
  - Jails
- Many nice things are developed with base-in-mind, including but not limited to
  - `poudriere(8)`
  - `vm-bhyve(8)`
  - `dwatch(1)`
- All good, BUT, easy to fork, and tricky to maintain, especially for a startup.
Disclaimers

It's a startup, so

- we're a team of 5 people working on everything
- we can't spend 500 USD/mo for a "build server"
- we can't wait hours till things compile
- we can’t afford giant resources
- we want to focus on our core problem

But it's also team of *BSD lovers, so…

These are our stories, we hope they help you :-)

Tech Stack | Software

- **OS**: FreeBSD (obviously)
- **Programming Languages:**
  - Elixir/Erlang/OTP
  - FreeBSD Shell
  - JavaScript
  - Oberon
  - Rust
  - C
- **CI**: BuildBot
- **VCS**: Git on Gitea
- **Packaging**: Poudriere
- **Shipping**: Poudriere + Scripts
- FreeBSD (obviously)
- Everything is a Jail
  - We use `jailer` for Jail automation (we built it, open-sourcing soon!)
- Non-FreeBSD things (Linux, Windows) are in VMs
  - We use `bhyve` with `vm-bhyve`
- “Server SSO" using NIS + NFS + AutoFS
  - DON'T TRY THIS AT HOME. We love old things that Just Works™
- Everything is tunneled among multiple locations between multiple continents
  - One of those locations (Armenia) is not FreeBSD-friendly (CDN-wise)
Development Flow

1. Commit
2. Gitea
3. App
4. Ports
5. BuildBot
6. Clone
   Make
   ...
   ...
   ...
   make iso
   make vm
   ssh pkg0
   update.sh
   ...
   ...
   make pkg!
7. update ports
   copy
   build new ports
8. pkg.company.com
Development Flow; Our Typical Port

```
... PORTVERSION= ${GIT_REVISION}
MASTER_SITES= http://downloads.build.example.com/webapp/
DISTNAME= ${PORTNAME}-${GIT_HASH}-${GIT_REVISION}
DIST_SUBDIR= ${PORTNAME}

GIT_HASH=8e0bf8502b6cc45e2e4b0b29723077b26c4b46cf
GIT_REVISION=88
...

.include <bsd.port.mk>
```
Development Flow; Update Port on Success

On success, buildbot changes the following user-defined macros in the port’s Makefile and commits to the main branch:

- \texttt{GIT\_HASH}=90c6a35b280734b72cef1509f44a5da75aadd765
- \texttt{GIT\_REVISION}=27

\texttt{+GIT\_HASH}=56a947ff8144d18fbbf719d002e7182aee830feb
\texttt{+GIT\_REVISION}=28
Development Flow; Separation

- **NO_BUILD=NO**
  - Skip the build step *(man 7 ports)*

- Clear separation of building and packaging
  - Building is done by BuildBot
  - Packaging is done by *poudriere*
Development Flow; Tarball to Package

- programX-abcdef-42.tgz
- programX-latest.tgz

Gitea

Ports

BuildBot

SAM (Snake and Make)
- ssh pkg0 update.sh

update ports copy
- build new ports
- make packages!

pkg.company.com
Ports Tree: Why Are We Shaking?

Maintaining Ports Tree

- **Option one: Fork FreeBSD's Ports tree and add your own stuff**
  - **Pros:** Single Ports tree to use internally
  - **Cons:** You have to regularly pull your copy and maintain your Makefiles
  - Best practice of usage: When your software relies on FreeBSD ports

- **Option two: Have your own Ports tree and merge it with other trees**
  - **Pros:** Update other copies only when needed
  - **Cons:** Need a reliable way to merge
  - Best practice of usage: When your software is "standalone"
Merging Port Trees

Portshaker:

- Simple (and single) configuration
- Simple to run

echo 'cloning company ports tree'
/usr/local/bin/portshaker -u company

echo 'merging port trees'
/usr/local/bin/portshaker -m default

echo 'merging done!'
/usr/local/bin/poudriere bulk -f /root/company-devel \
   -j company-devel-130 -p default
Building & Packaging; Final thoughts…

We have, at this point

- CI that builds
- Poudriere that packages
- Everything automated

We need

- Package server (via HTTP)
- With authentication
pkg.conf with HTTP_AUTH

/usr/local/etc/pkg/repos/example.conf

example: {
    url: "http://pkg.example.com/{ABI}/devel",
    enabled: yes,
    env: { HTTP_AUTH: "basic:*:user0:thepassword" }
}
Release

If neither userland nor kernel is altered you do not need to build them to make an image:

- `poudriere-image(8)` could be used for your ISO images with a little hack\(^1\)
- And a simple script could do the job for your VM images

\(^1\) poudriere ISO/USB images are not meant to be installable, they are just live images. But it doesn’t mean you cannot make them installable - we’ll go over this in a bit…
ISO/USB images

- `poudriere jail -c ...` (it shall contain a kernel)
- Fetch distribution tarballs (base.txz, kernel.txz, etc.)
- Add tarball of your files\(^1\) next to other distfiles
- Run `make-manifest.sh` from `/usr/src/release/scripts` to update the MANIFEST file (beware it prints to `stdout` - read the code, it’s small)
- Run `poudriere-image(8)` with the following options:

  - `-t iso, -j <jail_name>, -c <directory_to_copy>\(^2\)`
  - `-n <iso_name> -h <hostname>`

(distribution files are required to have an installer)

---

\(^1\) Custom Dist Structure
\(^2\) Overlay Structure
Example for your custom tarball

custom/
  └── usr
      └── local
          ├── etc
          │   └── pkg
          │       └── repos
          │           └── custom.conf
          └── some_dir
              └── some_subdir
                  └── file_b
The overlay structure

```
overlay/
    └── usr
        └── freebsd-dist
            ├── MANIFEST
            ├── base.txz
            ├── kernel.txz
            └── custom.txz
                └── ...
        └── local
            └── ...
```

Distfiles for 13.1-RELEASE:
https://download.freebsd.org/releases/amd64/13.1-RELEASE/
VM Images (ZFS)

- Create a file of a specific size
- Create a memory disk (mdconfig)
- Partition it (gpart) [1]
- Create a zpool
- Create datasets and set their properties [2]
- Take care of distributions and configuration files (bsdinstall distfetch, distextract, and config could be used)
- Make your changes
- Export the pool
- Detach the memory disk

[1] Partitioning
[2] Datasets

Special thanks to bofh@ for his help
Partitioning the memory disk

```bash
> gpart create -s gpt ${MD}
> gpart add -a 4k -s 512k -t freebsd-boot ${MD}
> gpart add -a 4k -t freebsd-zfs -l gpt_root ${MD}
> gpart bootcode -b /boot/pmbr -p /boot/gptzfsboot -i 1 ${MD}
```
CAUTION: Make sure you’re either on UFS or your desired pool name for the image differs from your system’s.

NOTE: swap volumes are not discussed - consult manuals if they are required.

```bash
> zpool create -R /mnt zroot /dev/${MD}p2
> zfs create -o mountpoint=none zroot/ROOT
> zfs create -o mountpoint=/ zroot/ROOT/default
> zfs create -o mountpoint=/tmp -o exec=on -o setuid=off zroot/tmp
> zfs create -o mountpoint=/usr -o canmount=off zroot/usr
```
ZFS Datasets and their properties (p.2)

> zfs create zroot/usr/home
> zfs create -o setuid=off zroot/usr/ports
> zfs create zroot/usr/src
> zfs create -o mountpoint=/var -o canmount=off zroot/var
> zfs create -o exec=off -o setuid=off zroot/var/audit
> zfs create -o exec=off -o setuid=off zroot/var/crash
> zfs create -o exec=off -o setuid=off zroot/var/log
> zfs create -o atime=on zroot/var/mail
> zfs create -o setuid=off zroot/var/tmp
> zpool set bootfs=zroot/ROOT/default zroot
VM Images (UFS)

Building a UFS image is easier than both ISO and ZFS images.

$ poudriere image -c <overlay> -n <name> -h <hostname>\ -f <pkg-list> -j <jail> -w <swap> -b -s <size> -t usb

“-b” is used to place the swap first to allow the primary partition to be grown on demand, and “-f <pkg-list>” specifies a list of packages to be pre-installed. For the latter, you should have used poudriere-bulk(8) first.
FreeBSD Release; HOW WE DO IT

- Unmodified Kernel and User-land
  - `poudriere-image(8)` for ISO/USB images
    - Custom dist files, `custom.txz`
  - Simple overlay
Example for our custom tarball

custom/
  └── usr
      └── local
          ├── etc
          │   └── pkg
          │       └── repos
          │           └── custom.conf
          └── company
              └── latest
                  ├── packageX-latest.tgz
                  └── packageY-latest.tgz
The overlay structure

- etc
  - rc.local
- usr
- freebsd-dist
  - MANIFEST
  - base.txz
  - kernel.txz
  - custom.txz
- local
  - bin
    - custom_installer
Please choose the appropriate terminal type for your system.

Common console types are:

- ansi Standard ANSI terminal
- vt100 VT100 or compatible terminal
- xterm xterm terminal emulator (or compatible)
- cons25w cons25w terminal

Console type [vt100]:

```
read TERM
TERM=${TERM:-vt100}
export TERM

/usr/local/bin/custom_installer
```
echo "starting installer"

SET HOSTNAME
exec 3>&1; _hostname=$(dialog --backtitle "LureOS Installer" --inputbox "Set \nHostname" 0 0 2>&1 1>&3);
exec 3>&-;

SET DISK

... 

cat <<EOF > /tmp/install.script
DISTRIBUTIONS="kernel.txz base.txz custom.txz"
export ZFSBOOT_VDEV_TYPE=stripe
export ZFSBOOT_DISKS=${_disk}
...
#!/bin/sh
sysrc hostname="${_hostname}"
...
EOF
bsdinstall script /tmp/install.script
echo "starting installer"
...
cat <<EOF > /tmp/install.script
...
#!/bin/sh
sysrc hostname="${_hostname}"
sysrc sshd_enable="YES"
REPOS_DIR="/etc/pkg/" pkg bootstrap -y
pkg add /usr/local/company/latest/packageX-latest.tgz
sysrc mydaemon_enable="YES"
passwd root
...
EOF
bsdinstall script /tmp/install.script
FreeBSD Release; VM Images

- `truncate -s 10G disk0.img ; mdconfig -f disk0.img`
- `copy "generated" install.script`
  - `export ZFSBOOT_DISK=md0`
  - Hardcode some values
- `bsdinstall script install.script`
- Host shall be using UFS
  - `bsdinstall exports all zpools, for some reason...`
Unforeseen Issues & Corresponding Solutions

- disk0.img size: used vs real
- `poudriere-image(8)` requires a kernel
- qemu-img from qemu-tools
- The Jail running `poudriere` requires it too!
These are not the tools you are looking for

https://github.com/freebsd/poudriere/wiki/poudriere-image.8

https://github.com/michaeldexter/occambsd

https://github.com/michaeldexter/imagine

https://github.com/5u623l20/vultr-freebsd-zfs

/usr/libexec/bsdinstall

/usr/share/bsdconfig
Caring := Sharing

- src
  - Good testing
  - Requires time
  - Large change? make it gradual
  - Follow up

- Ports
  - Ports are for everyone, not just for $WORK

- Docs
  - Apologies, we've been lazy :-(
That's all folks!
Thanks
Q&A
{av,kfv}@illuriasecurity.com
illuria.com