CTSRD

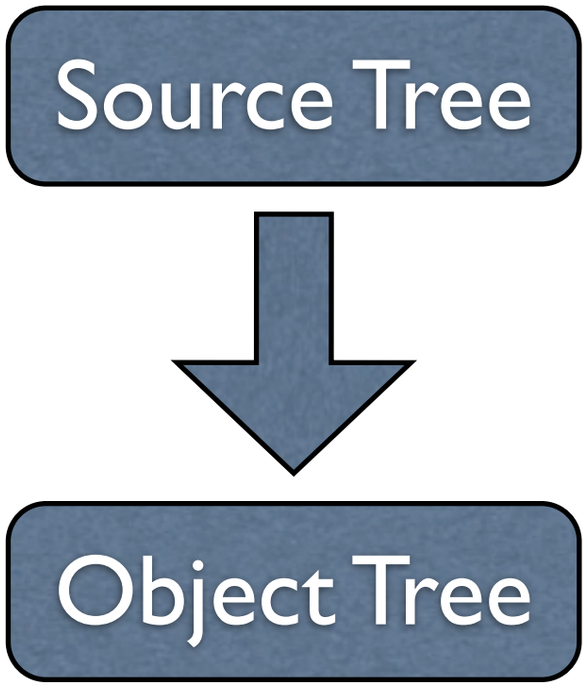
CRASH-worthy Trustworthy Systems Research and Development

Building FreeBSD Without Root Privilege

Brooks Davis
SRI International

FreeBSD Dev Summit, BSDCan 2013
May 17, 2013

Approved for public release. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

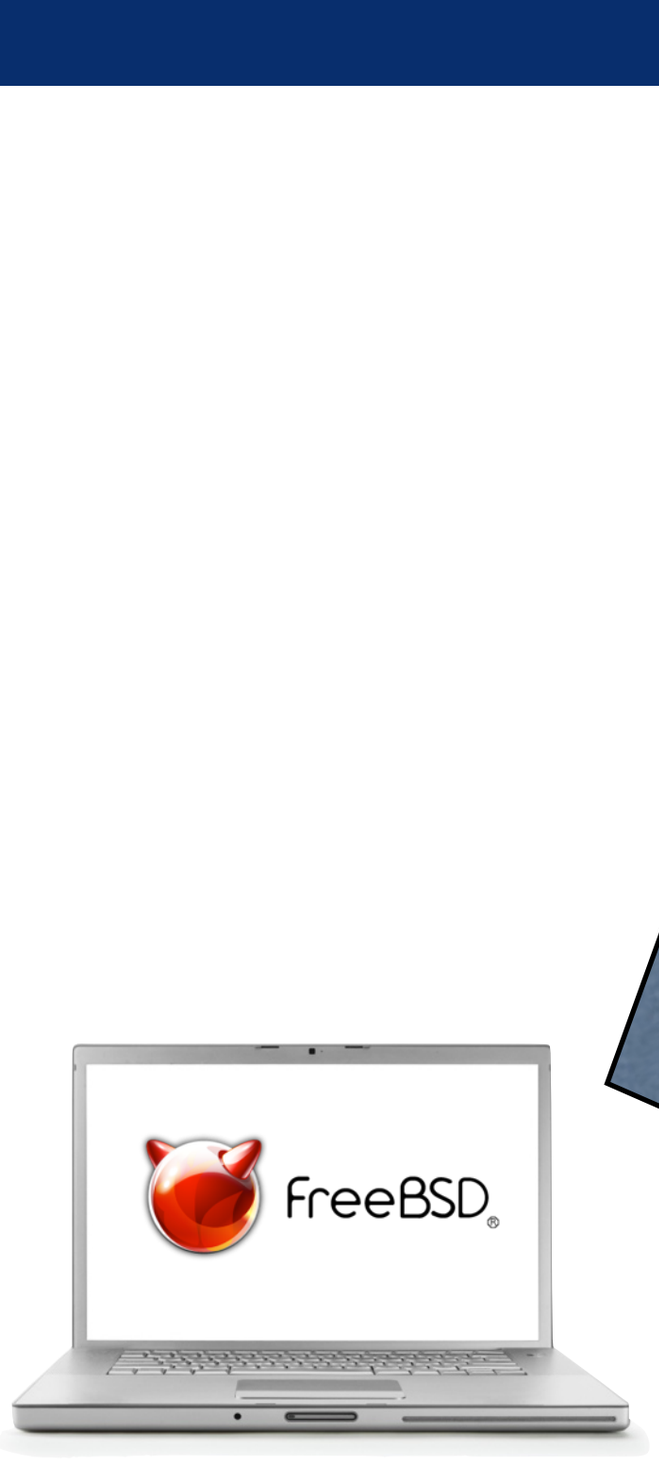

Source Tree
Object Tree

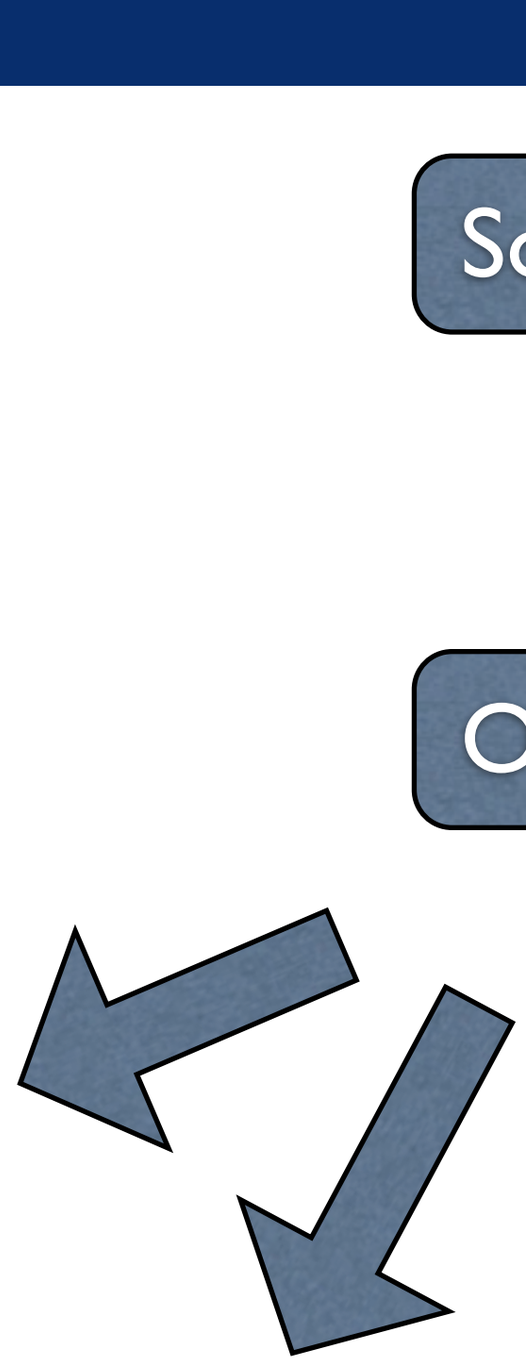

FreeBSD

Source Tree

Object Tree

Source Tree



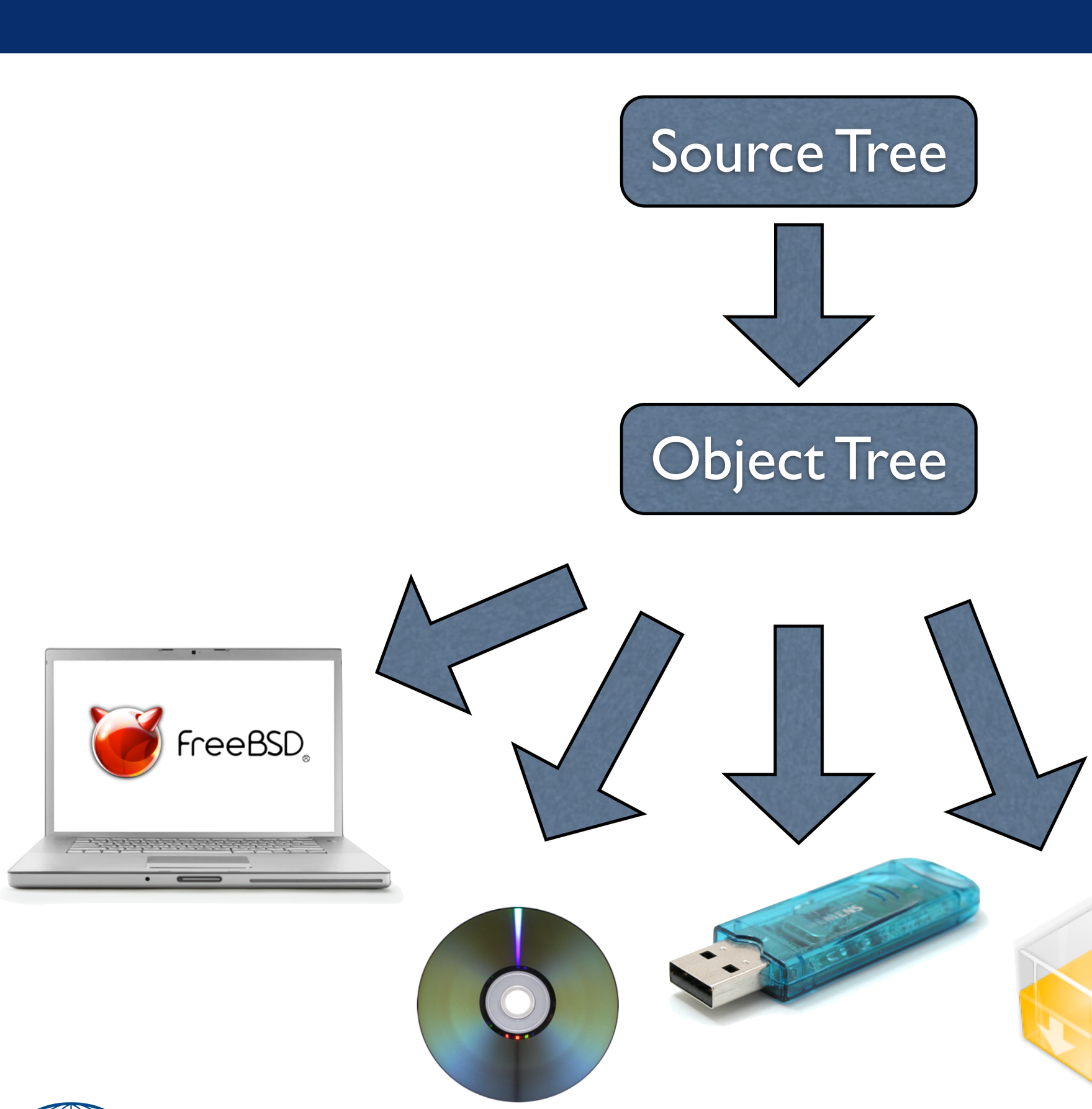


Object Tree

Source Tree

Object Tree

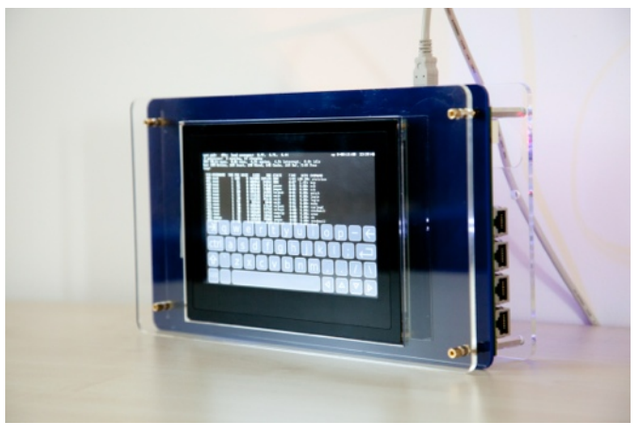
Source Tree



Object Tree

Source Tree

Object Tree



Source Tree

Object Tree

Problems of Privilege

- User errors compounded
- Trust too much code
- May not have root access
- Often no need

The User/Group Database Problem

```
$ make installworld
ERROR: Required auditdistd user is missing, see /usr/src/UPDATING.
*** [installcheck_UGID] Error code 1

Stop in /usr/src.
*** [installworld] Error code 1

Stop in /usr/src.
$
```

- Related to the privilege problem
- Why require unused users?

Outline of the Solution

- Install files as user
  - No suid
- Log correct permissions
- Teach image creation tools about logs
- Use user/group databases from source tree not installed system

Import of NetBSD `mtree`

- Added our features to NetBSD
- Added flavor support for compatibility
- Installed as `nmtree`
  - Will replace `mtree` in 10.0

New `install` options

- From NetBSD
  - `-M` (`-D`, `-h`, `-T`): metadata log
  - `-N`: user/group databases
  - `-U`: unprivileged operation
  - `-l`: hard and symbolic links

Build System Changes

- Remove use of `ln` in install targets
- User `nmtree` to create directories
- `NO_ROOT` and `DB_FROM_SRC`
- Remove duplicate installation of files

How to Use

```
$ make -DNO_ROOT -DDB_FROM_SRC DESTDIR=/path/to/dest \
    installworld
$ make -DNO_ROOT -DDB_FROM_SRC DESTDIR=/path/to/dest \
    distribution
$ make -DNO_ROOT -DDB_FROM_SRC DESTDIR= /path/to/dest \
    installkernel
$ cd /path/to/dest; makefs /path/to/output/image METALOG
```

- Create the log through the usual install process
- Use the log to build an image

Advanced Usage

```
$ makeroot.sh -k keys -K ctsrd \
    -p extras/etc/master.passwd -g extras/etc/group \
    -e extras/mdroot.mtree -e demo/demo.mtree \
    -e extras/ctsrd.mtree -s 26112k -f demo.files \
cheribsd-demo.img /path/to/dist
```

- Add files or subset the image
- `/usr/src/tools/tools/makeroot/`

Status

- Image creation with `tar` and `makefs`
- Shipping releases of CheriBSD
- Limited FreeBSD release support
- Merged to 9-STABLE

To Do

- Tools to audit `METALOG` vs installed files
- Support for partitioned disk images
  - Geom direct dispatch could simplify
- Make all releases without privilege
- Mechanism to install packages in an image

CTSRD

CRASH-worthy Trustworthy Systems Research and Development

External Toolchain Support

Brooks Davis
SRI International

FreeBSD Dev Summit, BSDCan 2013
May 17, 2013

Approved for public release. This research is sponsored by the Defense Advanced Research Projects Agency (DARPA) and the Air Force Research Laboratory (AFRL), under contract FA8750-10-C-0237. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.

Why Support External Toolchains?

- Test new compiler releases
- Allow odd compilers:
  - New (GPLv3) GCC
  - Proprietary/vendor provided
  - Experimental version of `clang`
- Build FreeBSD 10 on systems where `cc` isn't `clang`

Approach

- Top level (`buildworld`, etc) only
- Override `CC`, `CXX`, `CPP`, `AS`, ...
- Compiler not bootstrapped if `XCC` set
- Separate compiler and toolchain overrides
  - *Not the traditional worldview*

How to Use

- Set `XCC`, `XCXX`, `XCPP`, `XAS`, `XLD`, ...
- Alternatively, set:
  - `CROSS_COMPILER_PREFIX`
  - `CROSS_BINUTILS_PREFIX`
  - `CROSS_TOOLCHAIN_PREFIX` sets both

Status

- Committed to HEAD
  - Works with `clang`
- Documentation at:
  - https://wiki.freebsd.org/ExternalToolchain

To Do

- Move into `share/mk`
- Way set `COMPILER_*` variables
- More flexible way to handle compiler warning flags

Q & A