Finding Security Vulnerabilities Using Coccinelle

Linux Security Summit 2012
Kees Cook
(pronounced "Case")
keescook@chromium.org
Overview

- Where to look
- How to look
- Focus on copy_from_user
- Generate "design pattern" whitelist
- Track output
- More ideas
Where to look

- **scary userspace/kernel boundaries**
  - ioctl
  - copy_from_user
  - copy_to_user
  - netlink

- **interfaces with few consumers**
  - rare network protocols (SCTP, RDS)
  - video DRM (mostly just Xorg)
  - network diagnostics (handful of debug tools)
  - new syscalls
  - compat layer
How to look

- favorite editor
- grep
- **coccinelle**
  - start-up tutorial on LWN from 2009
  - many good upstream bug-finding examples
  - my simple parallel run harness
- **sparse**
- **smatch**
Focus on copy_from_user

- Started with ___copy_from_user
  - very few callers, used grep
  - Intel DRM ([CVE-2010-2962](CVE-2010-2962))
  - RDS ([CVE-2010-3904](CVE-2010-3904), Dan Rosenberg)

- Poked at copy_to_user
  - kernel leaks, used [simple coccinelle rule](simple_coccinelle_rule)
  - net ioctl ([CVE-2010-3861](CVE-2010-3861))
  - net ioctl ([CVE-2010-4655](CVE-2010-4655))

- Moved on to copy_from_user
  - ~4000 callers, used [complex coccinelle rule](complex_coccinelle_rule)
  - usb io-warrior ([CVE-2010-4656](CVE-2010-4656))
  - v4l compat ([CVE-2010-2963](CVE-2010-2963))
Generate "design pattern" whitelist

1. match the scary function
2. run and study output
3. add rule for safe usage pattern
4. goto 2
Track output

- still have **800 lines of output**
- switch to diff-style output
- diff the outputs over time
- want to be automated
More ideas

- other good targets?
  - mispatterns in ioctls
  - detecting partially initialized stack variables
  - more?