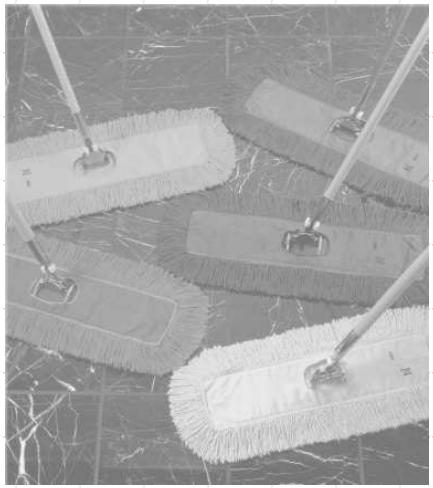
MOPS

MOdelchecking Security Properties



David Wagner
U.C. Berkeley

The Problem

- Security holes are often in the software
  - Software bugs are a leading cause of security vulnerabilities
- Security programming is pitfall-laden
  - It's too easy to unintentionally violate implicit usage rules of OS API's

Improving Software Quality

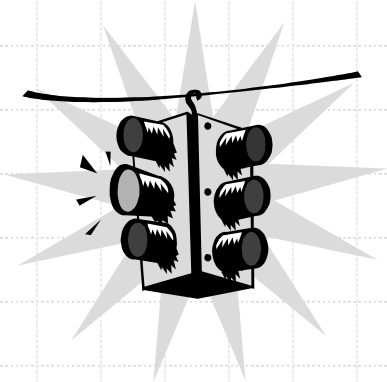
- If secure programming is hard, let's build tools that make it easier to get security right
- An approach: *enforce defensive coding*
  - Enumerate rules of prudent security coding
  - Use tools to automatically verify that software follows these rules
- Project goal: explore a novel approach to this

A High-Level View

- Compile-time analysis of C source code
  - For developers:
    - Integrating MOPS into build process catches bugs as soon as they're introduced
    - Think of MOPS like a type system; would you program without one?
  - For auditers:
    - MOPS can analyze legacy code to help with code reviews of existing packages
  - A perfect match for open source

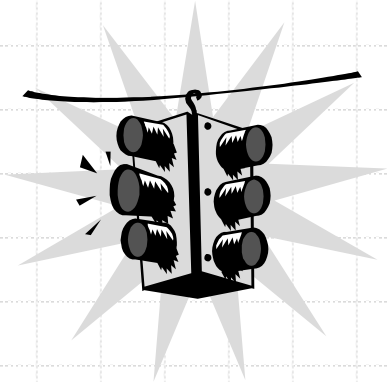
Uh-Oh…

- But: full software verification is totally impractical. Isn't this idea hopeless??



No Problem!

- But: full software verification is totally impractical. Isn't this idea hopeless??
- Answer: *Wrong! We can do a lot.*



Lightweight Verification

- How to make verification practical:
  - Check application-independent properties
    - Reduce specification costs through reuse
  - Support only a subclass of properties
    - Temporal safety properties: i.e., "ordering"
  - Exploit advances in modelchecking
    - Analyze control flow; ignore data flow
  - Be conservative: warn when unsure

Prudent Coding Rules

Key insight:

- Many rules are finite-state machines
  - Good for "ordering properties"
  - Intuitive for programmers

Prudent Coding Rules

Key insight:

- Many rules are finite-state machines
  - Good for "ordering properties"
  - Intuitive for programmers

Example of a rule:

- Avoid calling `exec()` or `system()` with root privilege

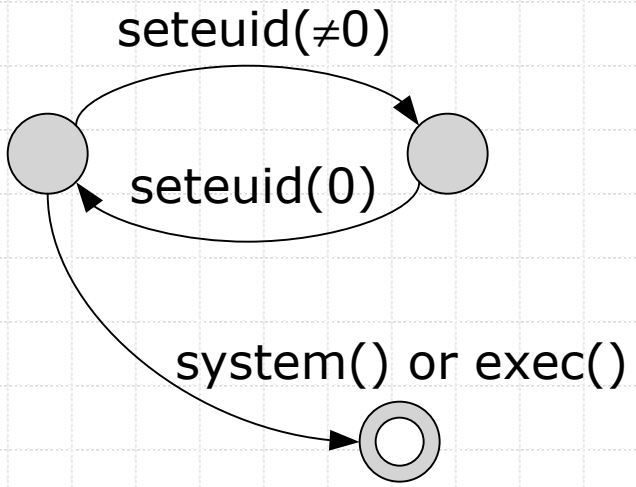
Prudent Coding Rules

Key insight:

- Many rules are finite-state machines
  - Good for "ordering properties"
  - Intuitive for programmers

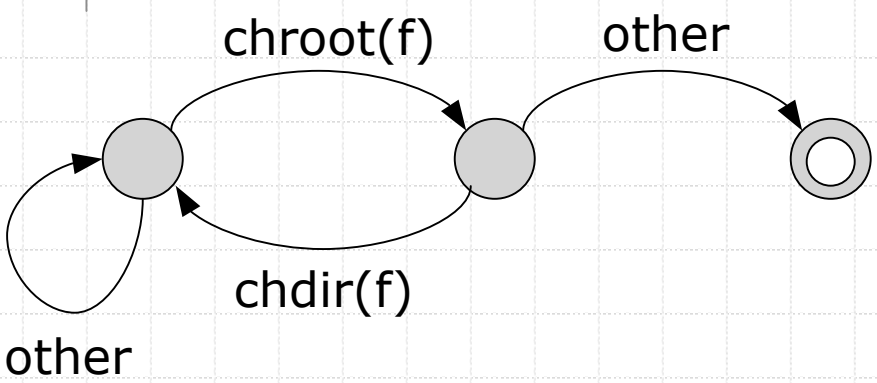
Example of a rule:

- Avoid calling `exec()` or `system()` with root privilege

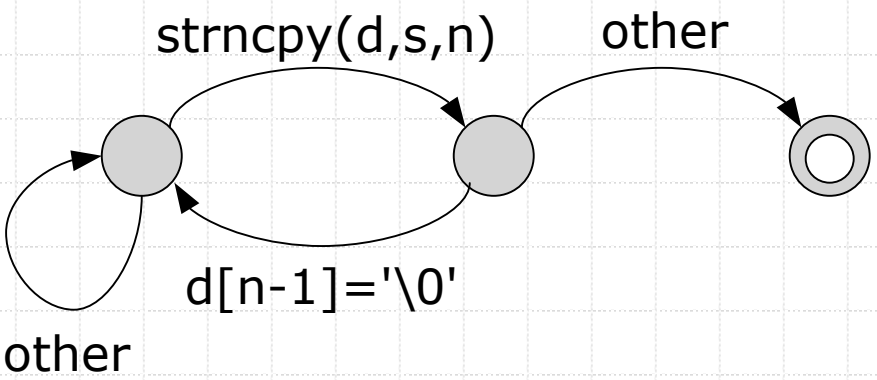


More Example Rules

- After `chroot(f)`, immediately call `chdir(f)`

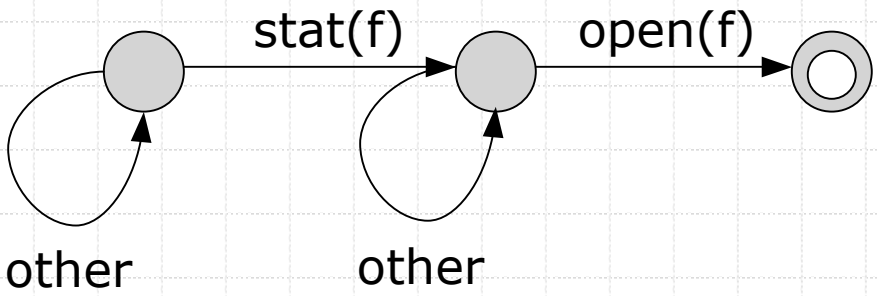


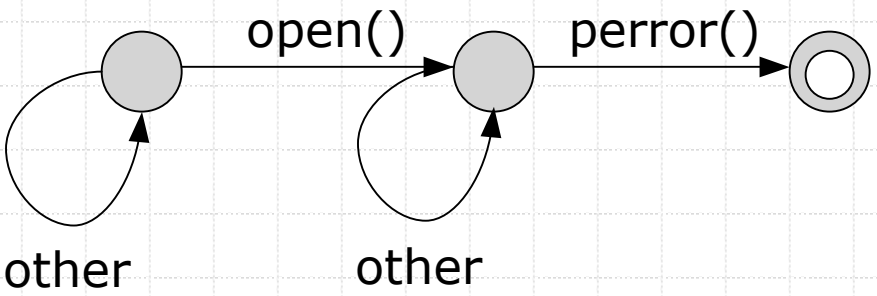
- Always follow `strncpy(d,s,n)` by `d[n-1] = '\0'`



More Example Rules (2)

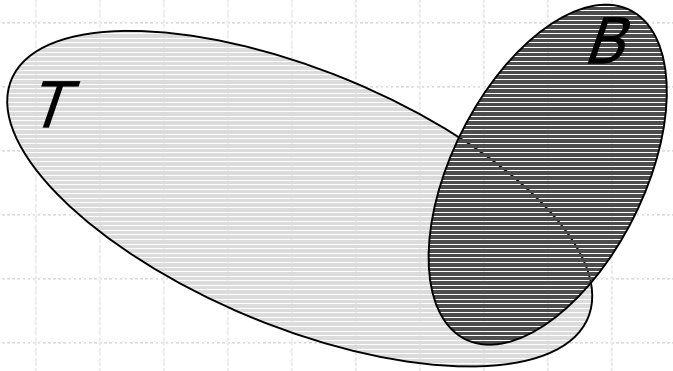
- A `stat(f)` followed by `open(f)` is awfully suspicious (race conditions)
- In a setuid program, `open()` followed by `perror()` is very dangerous





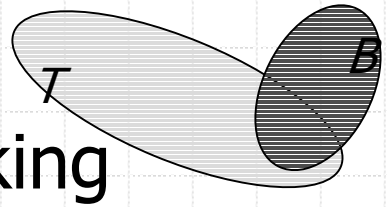
Under the Hood

- How to check whether code satisfies a property
  - Let Σ = set of security-relevant events,
    B = set of "bad" traces that violate the property,
    T = set of feasible traces ($T, B \subseteq \Sigma^*$)
  - If $T \cap B = \emptyset$, then the property is respected



Under the Hood

- How to check whether code satisfies a property
  - Let Σ = set of security-relevant events,
    B = set of "bad" traces that violate the property,
    T = set of feasible traces ($T, B \subseteq \Sigma^*$)
  - If $T \cap B = \emptyset$, then the property is respected



- Framework: software model checking
  - B: finite-state automaton (regular language)
  - T: pushdown automaton (context-free lang.)

Other Technical Advances

- Better modelchecking for security
  - Compaction: for scalability
  - Backtracking: for explaining bugs
  - Automatic model extraction: how to cheaply build a faithful formal model of (parts of) the OS
  - Paper in submission
- Guidance for programmers on privilege management
  - Tutorial paper on pitfalls in setu*id(), and on how to use it safely
  - A safer API for privilege management
  - Paper accepted at Usenix Security 2002

Some Results

- OpenSSH
  - Ssh 2.5.2: properly drops root before exec() *(new)*
  - Ssh, sshd 2.5.2: no set*uid() call will fail *(new)*
- Sendmail
  - 8.10.1: has capabilities bug on Linux *(old)*
  - 8.12.0: fails to drop group privileges properly *(old)*
- Wu-ftpd
  - 2.4β11: has tractorbeaming attack *(old)*
  - 2.4β12: no tractorbeaming attacks -- follows defensive programming rules for setuid, longjmp, signals *(new)*
- Login, crontab, …
  - Have fd-inheritance security holes when run setuid *(new?)*

More Results

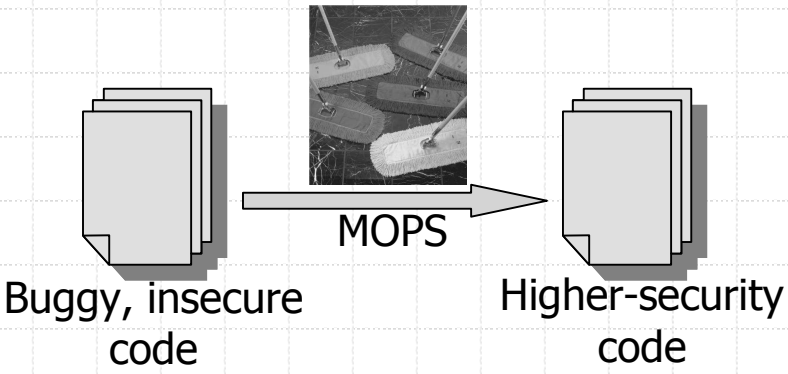
- Buggy manual pages
  - `setuid(2)` in RH Linux 7.2: omits capabilities
  - `setgid(2)` in RH Linux 7.2: incorrectly claims gid 0 is special
  - `setreuid(2)` in FreeBSD 4.4: incorrectly claims ruid/euid can always be swapped
- Buggy operating systems
  - Linux kernel 2.4.18: fsuid invariant violated; security risk (our proposed fix accepted by Linus)
- Moral: Formal models are powerful

Status of MOPS

- Fully functional first cut
  - Parses anything gcc will; allows specification of user-defined properties
  - Some limitations (work-in-progress):
    - Doesn't come with a database of rules of defensive programming … yet
    - UI, build integration isn't "pretty" … yet
- Publicly released -- come and get it!

  http://www.cs.berkeley.edu/~daw/mops/

MOPS Project Summary



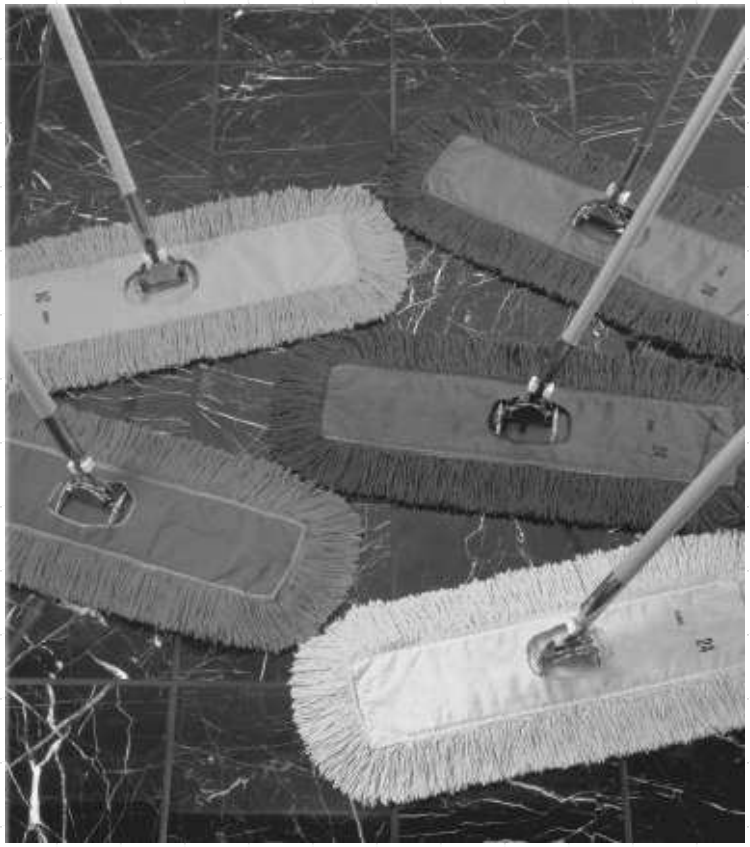
MOPS: building more secure software

Our main contributions:

- Novel techniques for improving software assurance of open source software through model-checking & lightweight formal methods
- Verification of certain security properties of important open source software; several security bugs found & fixed
- Release of our tool, MOPS, to open source community

Conclusion

- MOPS: making security programming safer



http://www.cs.berkeley.edu/~daw/mops/