Environment-Sensitive Intrusion Detection

Jonathon T. Giffin *Somesh Jha* *Barton P. Miller*

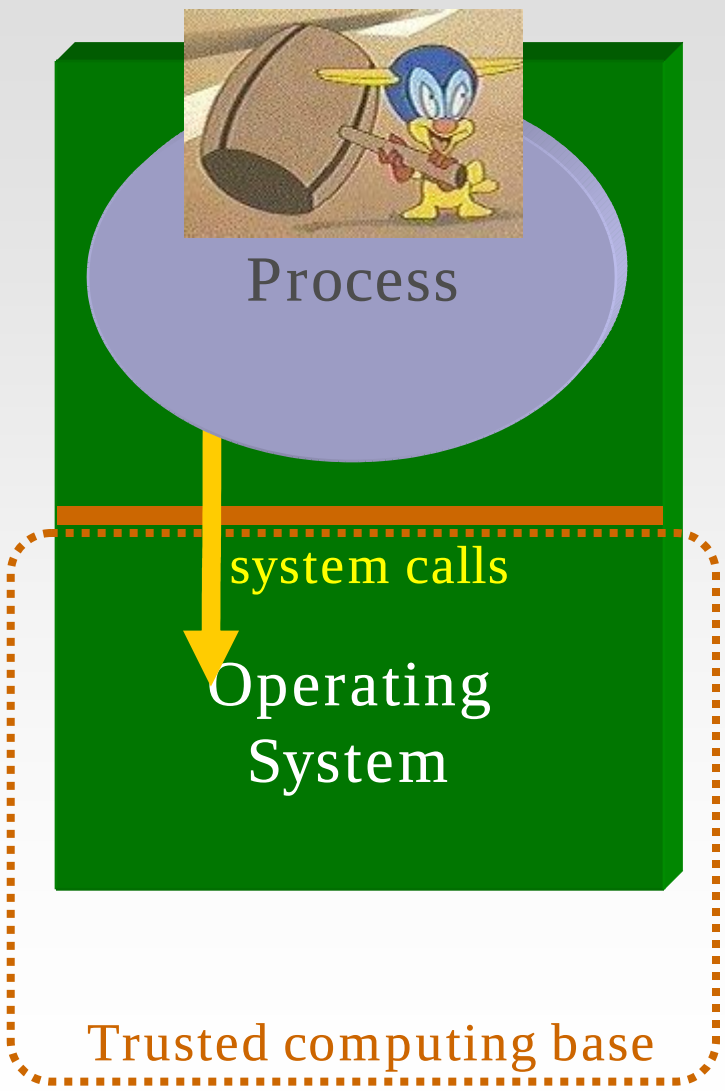
University of Wisconsin
{giffin,jha,bart}@cs.wisc.edu

David Dagon *Wenke Lee*

Georgia Institute of Technology
{dagon,wenke}@cc.gatech.edu

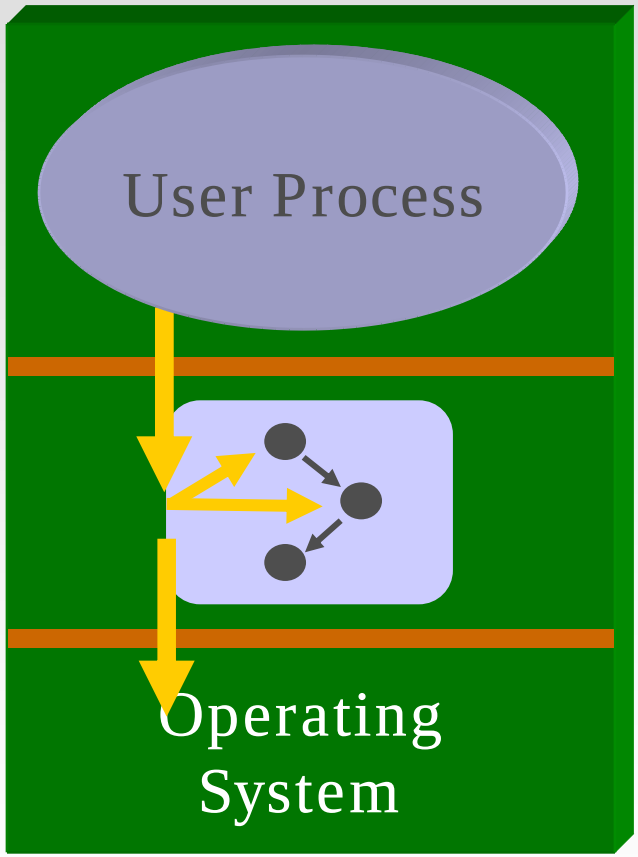
RAID 2005

Worldview



- Running processes make operating system requests
- Changes to trusted computing base done via these requests
- Attacker subverts process to generate malicious requests

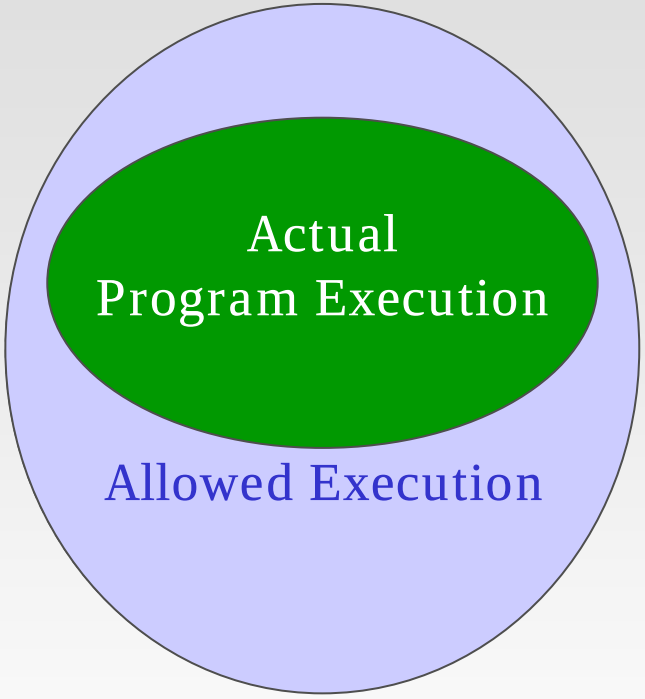
Model-Based Anomaly Detection



- Detect deviations from normal execution behavior

[Ko *et al.* 1994] [Forrest *et al.* 1996] [Lee *et al.* 1997] [Ghosh *et al.* 1999] [Sekar *et al.* 2001] [Wagner & Dean 2001] [Kruegel *et al.* 2003] [Giffin *et al.* 2004] [Feng *et al.* 2004] [Lam & Chiueh 2004] [Gao *et al.* 2004] [Gopalakrishna *et al.* 2005] [Abadi *et al.* 2005]

Models from Static Binary Analysis



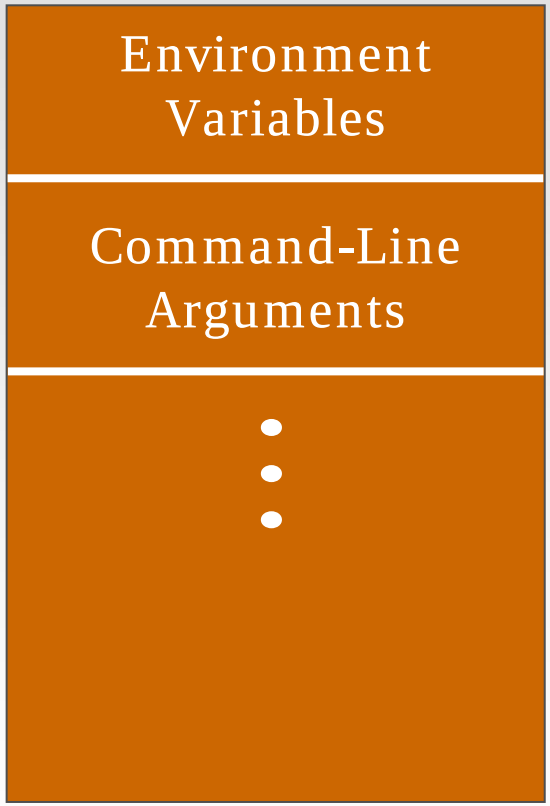
- Model overapproximates correct execution
- Close gap between model & correct execution
  - System call argument recovery
  - Context-sensitive models restrict system call sequences

Drive Through the Holes...

- Overwrite program data to alter execution

  [Chen, Xu, & Sezer 2005]

mailx memory contents:



Drive Through the Holes...

- Evasion attack
  - Mailx executed in receive mail mode

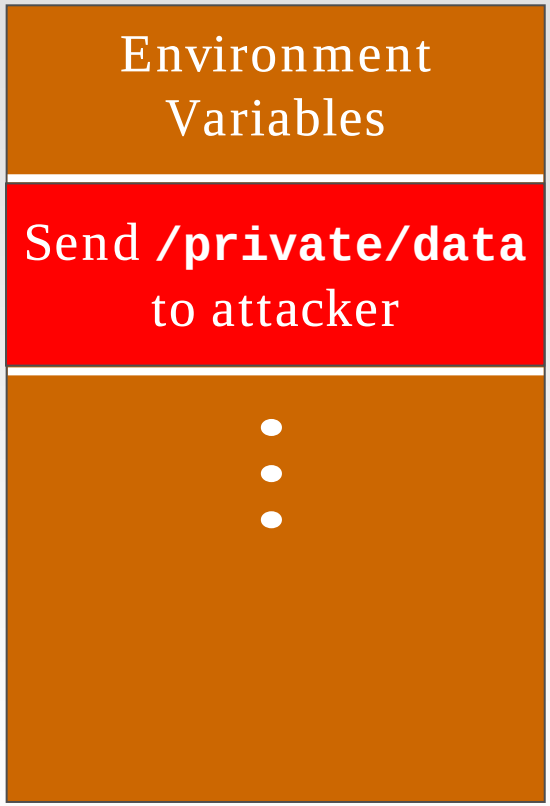
mailx memory contents:

Drive Through the Holes...

- Evasion attack
  - Mailx executed in receive mail mode
  - Attacker exploits vulnerability to alter command-line args
  - Reenters execution at argument processing routine

Undetected by current models

mailx memory contents:



New Contributions

- Static analysis infrastructure:
  - Model construction for dynamically-linked binaries
- Model accuracy:
  - Context-sensitive data-flow analysis for system call argument recovery
  - Environment-sensitive program models
- Model evaluation:
  - Average reachability measure

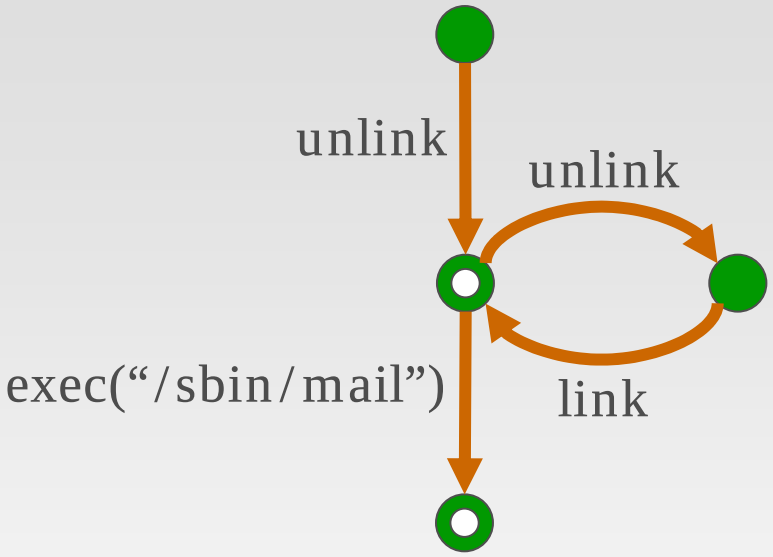
Example

```
void parse_args (int argc, char *argv[])
{
   char *workfile = tempnam(getenv("TMP"), "Mx");
   int execmode = 1;
   char c;

   unlink("/home/user/tmpfile");
   while ((c = getopt(argc, argv, "L:")) != -1)
      switch (c) {
         case 'L':
            execmode = 0;
            unlink(workfile);
            link(optarg, workfile);
            break;
      }

   if (execmode)
      exec("/sbin/mail");
}
```

Prior Model Construction

Unable to construct model for dynamically-linked program

Prior Model Construction



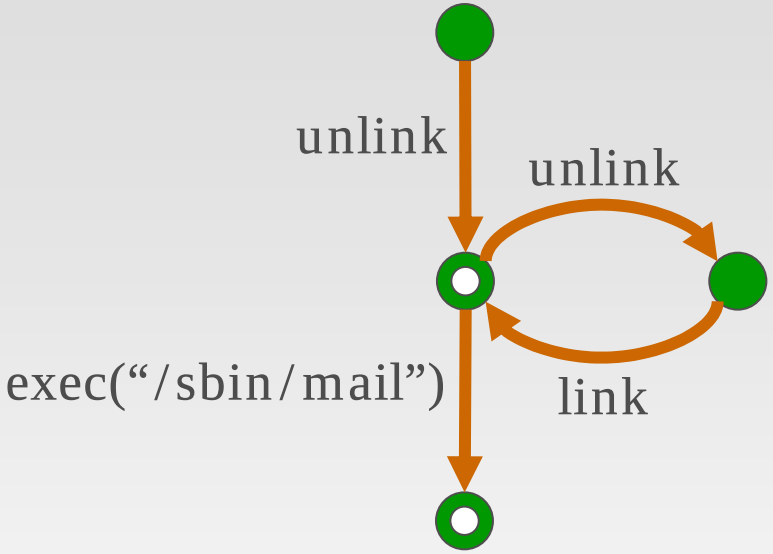
Attack: unlink("/home/user/tmpfile")

unlink("/sbin/mail")

link("/bin/sh", "/sbin/mail")

exec("/sbin/mail")

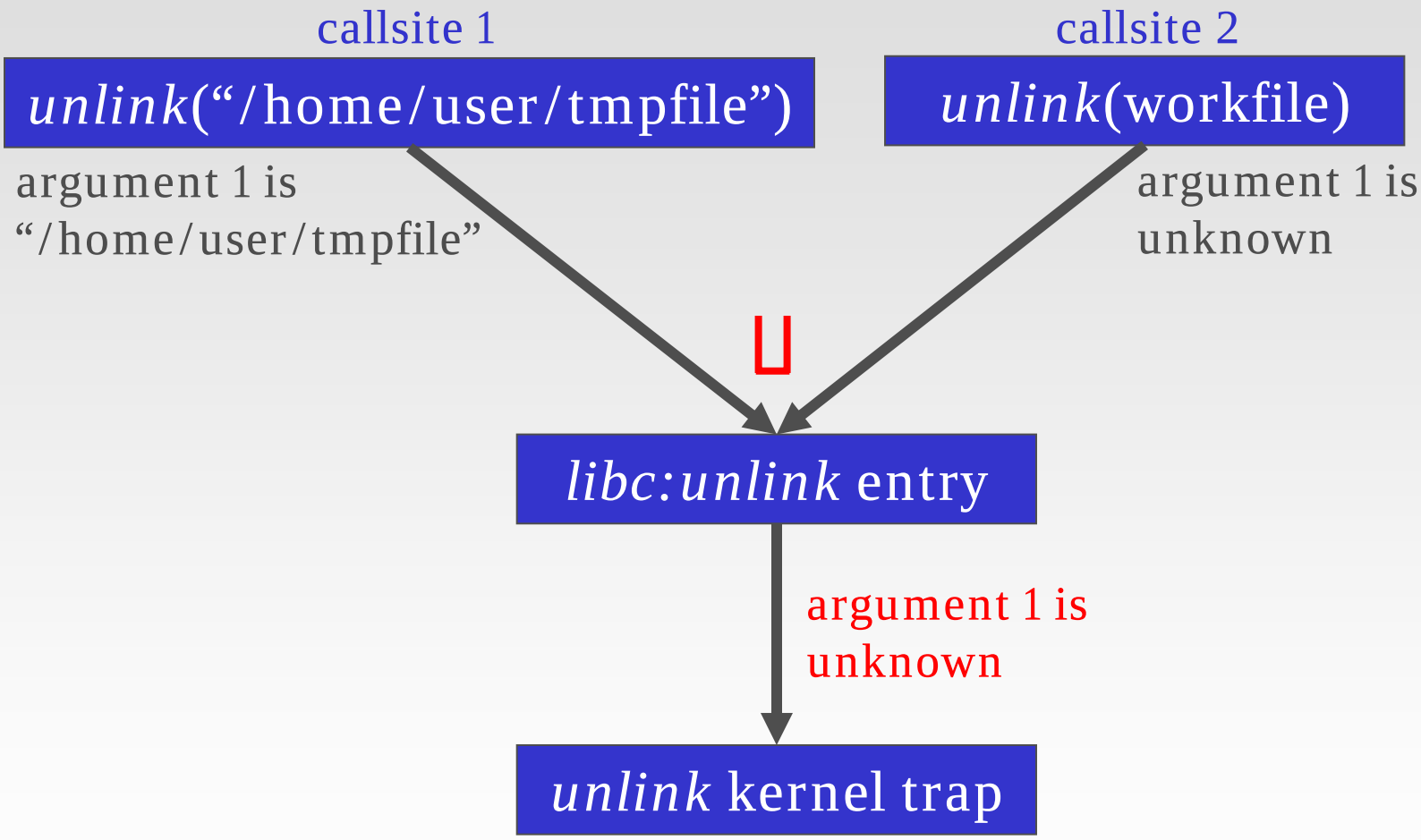
Prior Model Construction



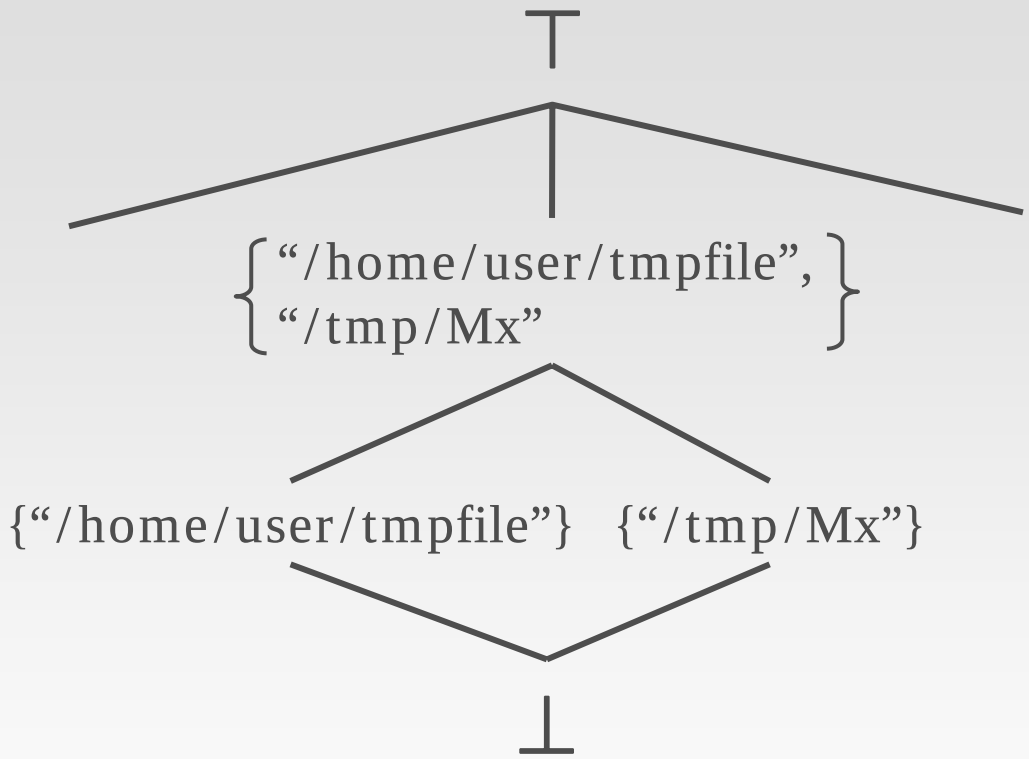
Limited recovered arguments

No correlated branching

Context-Insensitive Argument Recovery



Context-Insensitive Argument Recovery



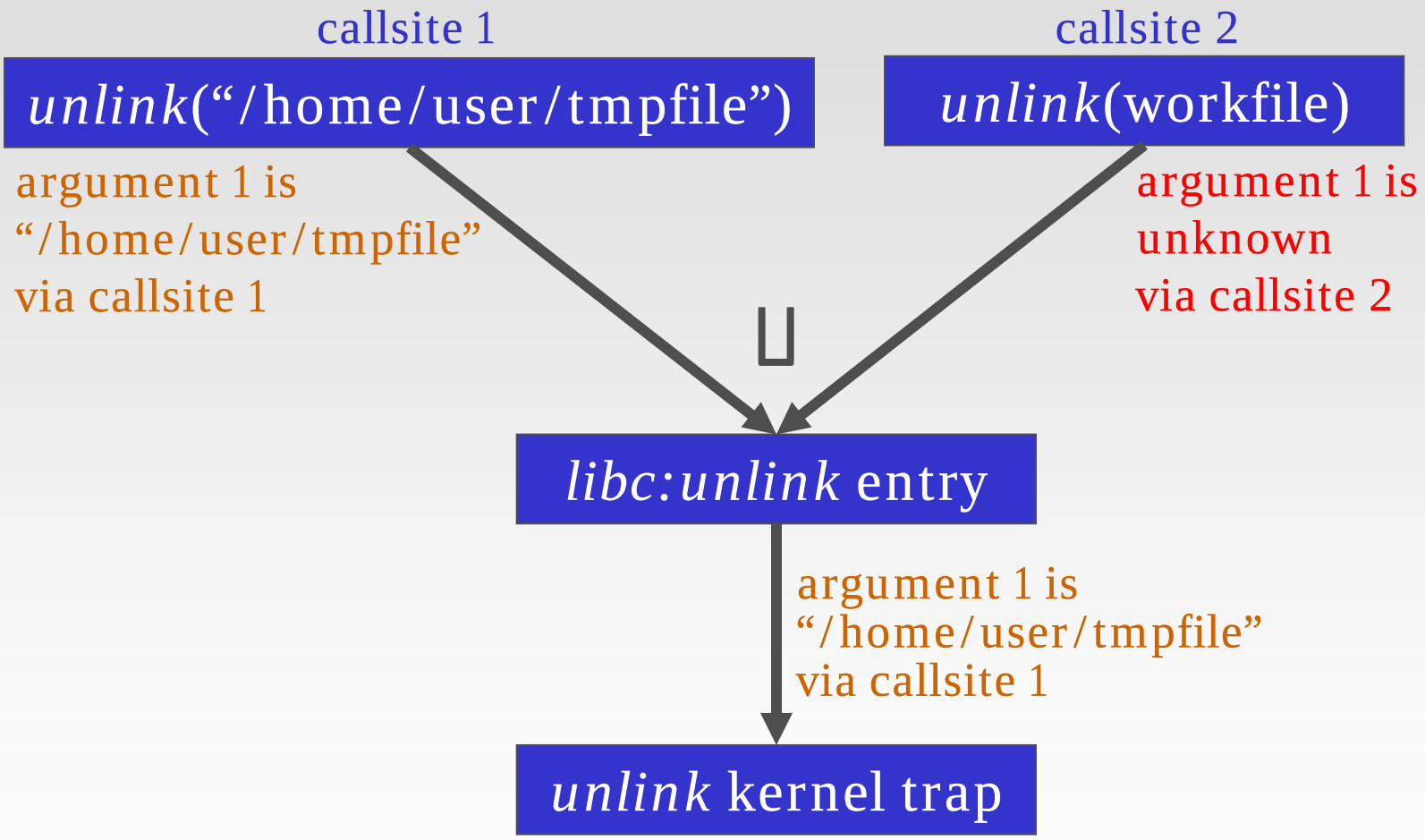
Anything ⊔ ⊤ = ⊤

Context-Sensitive Argument Recovery

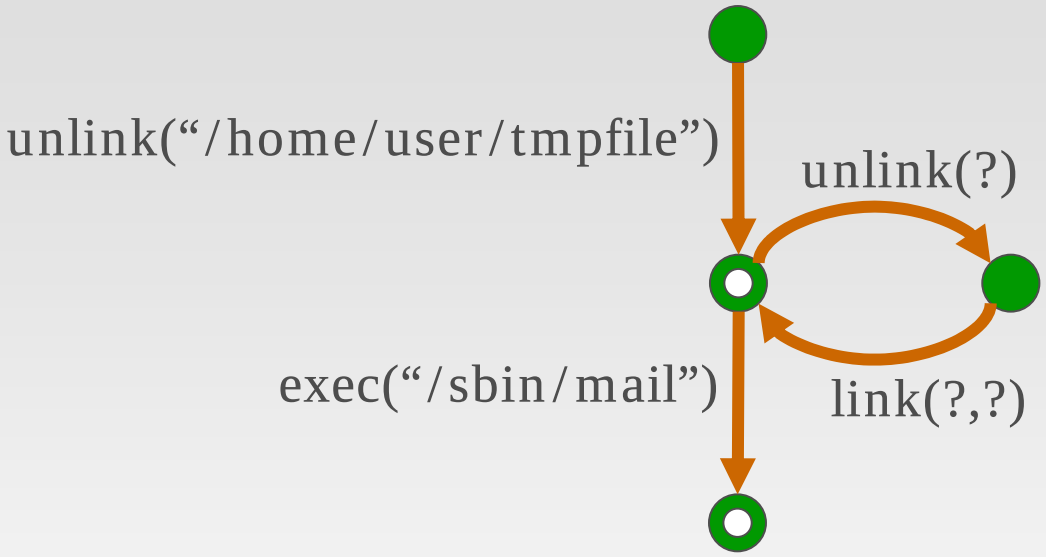
- Augment lattice values with calling context
- $\sqcup$ maintains context

$$\text{Value X in context 1} \sqcup \top \text{ in context 2} = \text{Value X in context 1}$$

Context-Sensitive Argument Recovery



Context-Sensitive Argument Recovery



Attack: unlink("/home/user/tmpfile")

unlink("/sbin/mail")

link("/bin/sh", "/sbin/mail")

exec("/sbin/mail")

Environment Sensitivity

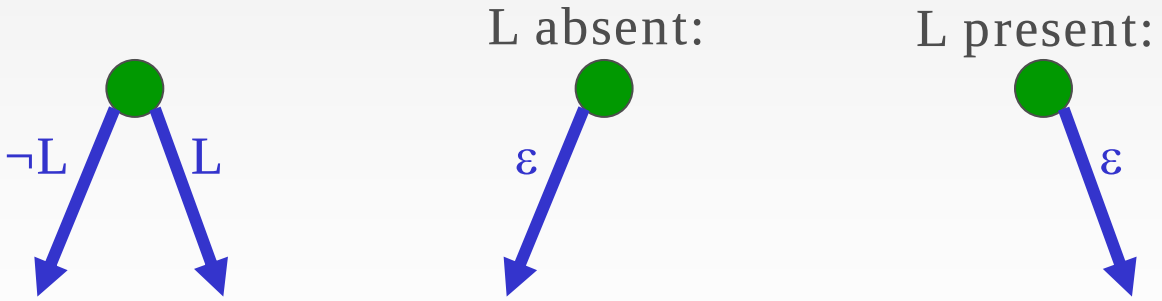
- Three sources of data used in a program
  - Statically encoded in binary
  - Runtime input from user, network, disk, ...
  - Load-time input from command-line, environment variables, & configuration files
- Static analyzer constructs model template
- Execution monitor instantiates model at program load time

Model Template

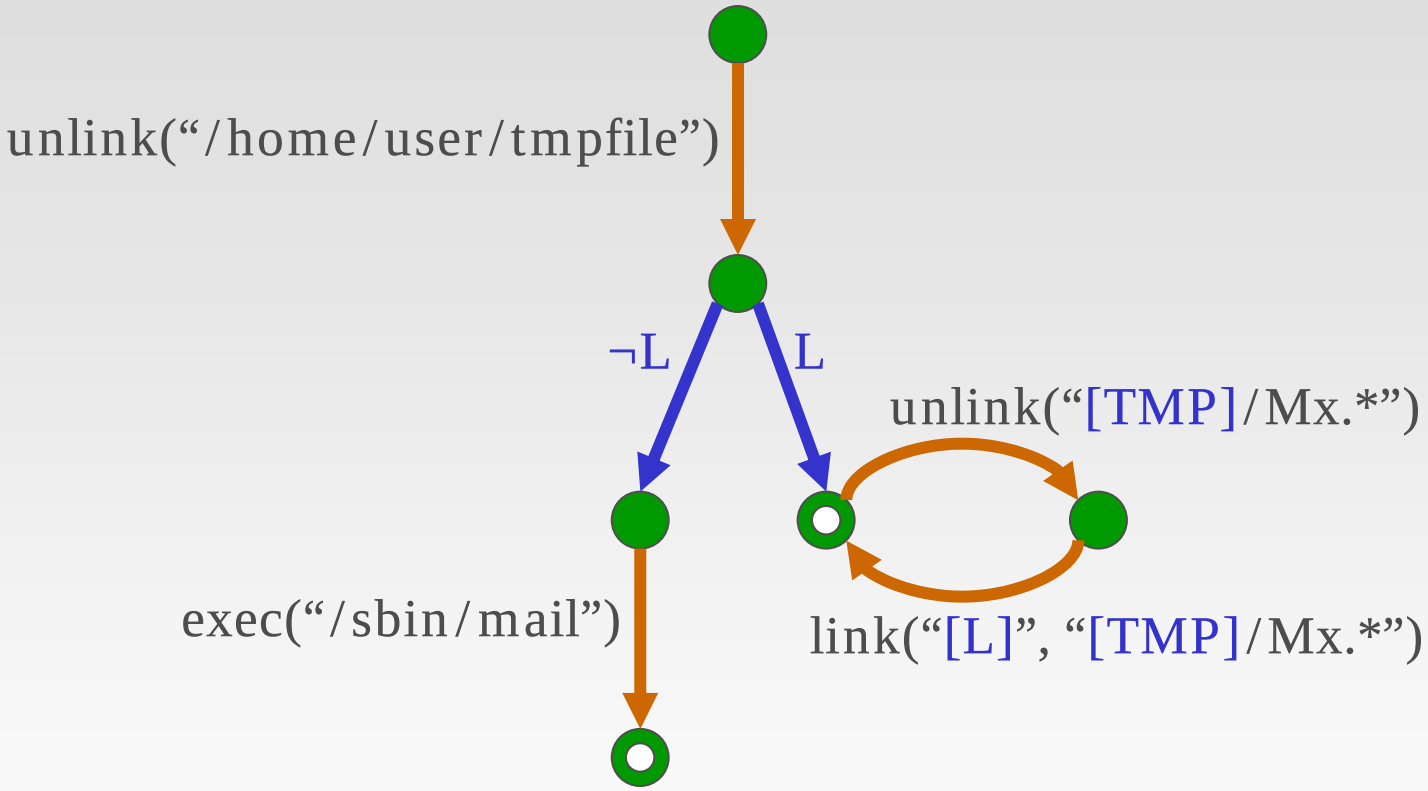
- Encode environment dependencies into model
- Data-flow dependencies

unlink("[TMP]/Mx.*")
link("[L]", "[TMP]/Mx.*")

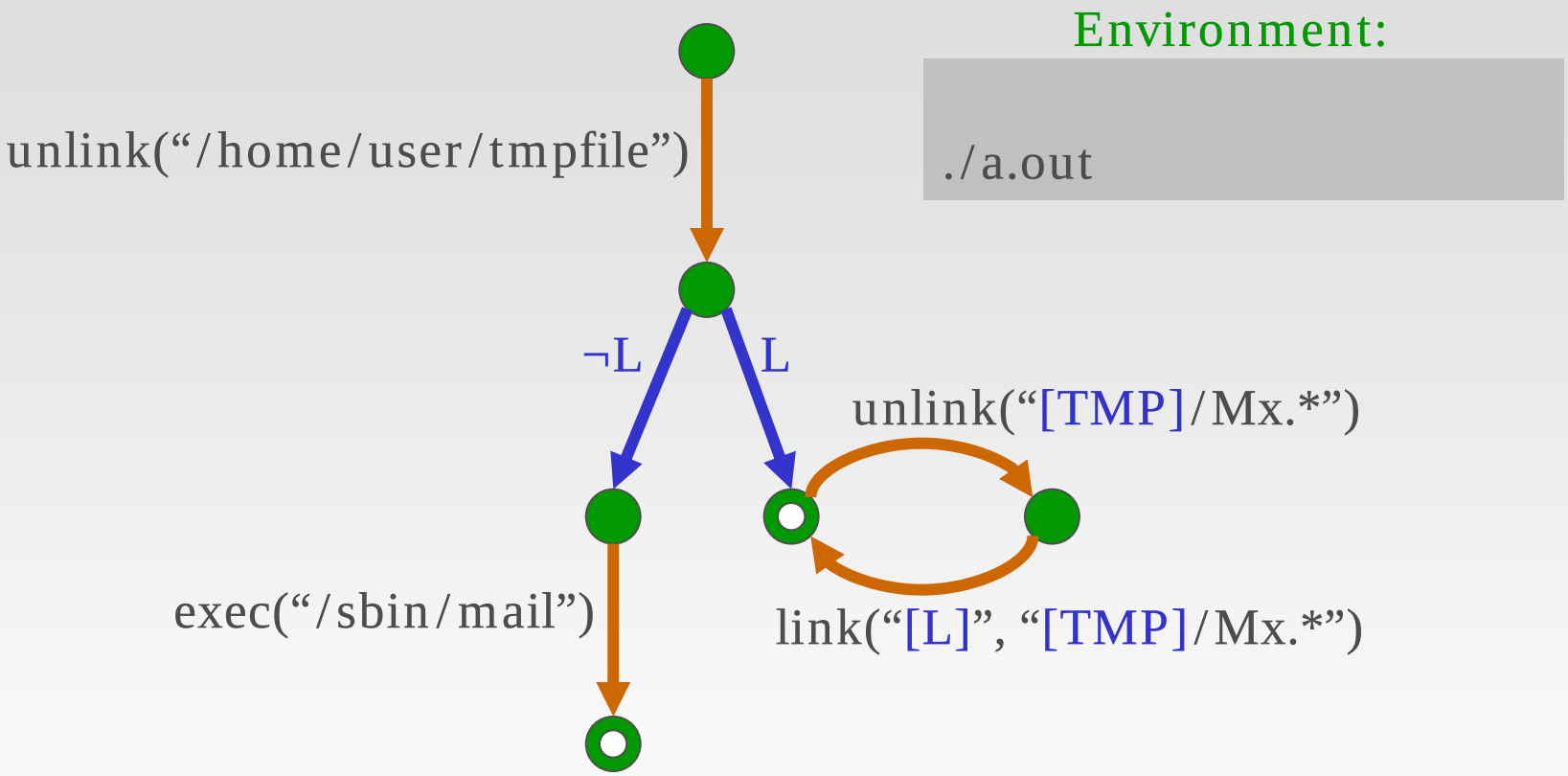
- Control-flow dependencies



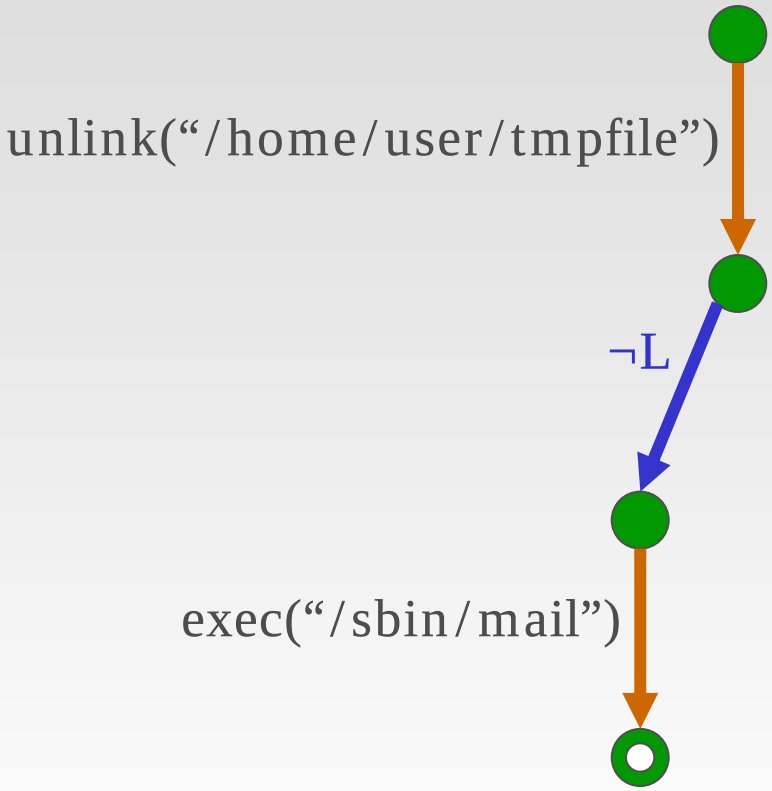
Model Template



Model Instantiation



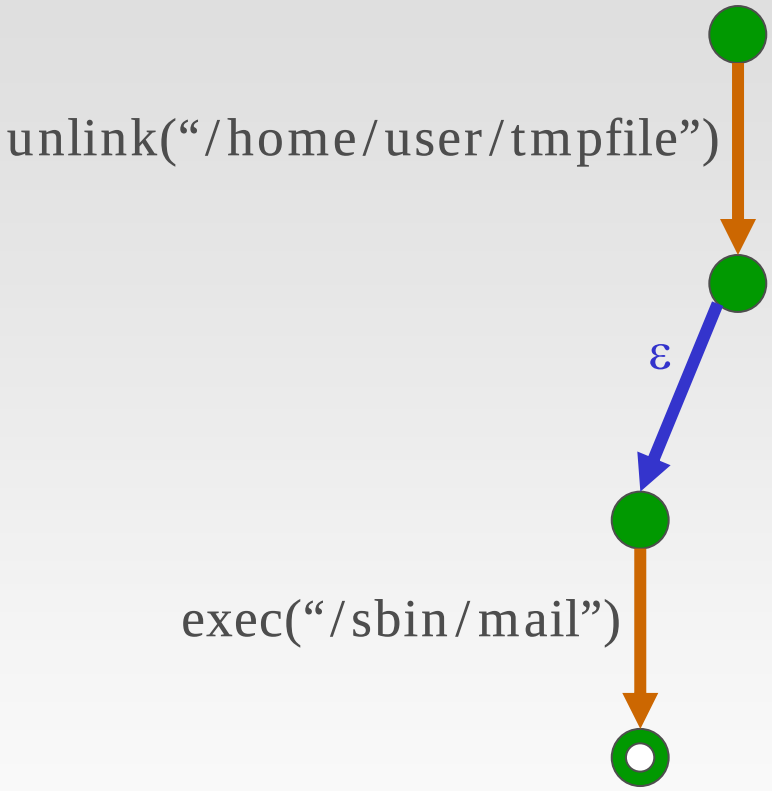
Model Instantiation



Environment:

./a.out

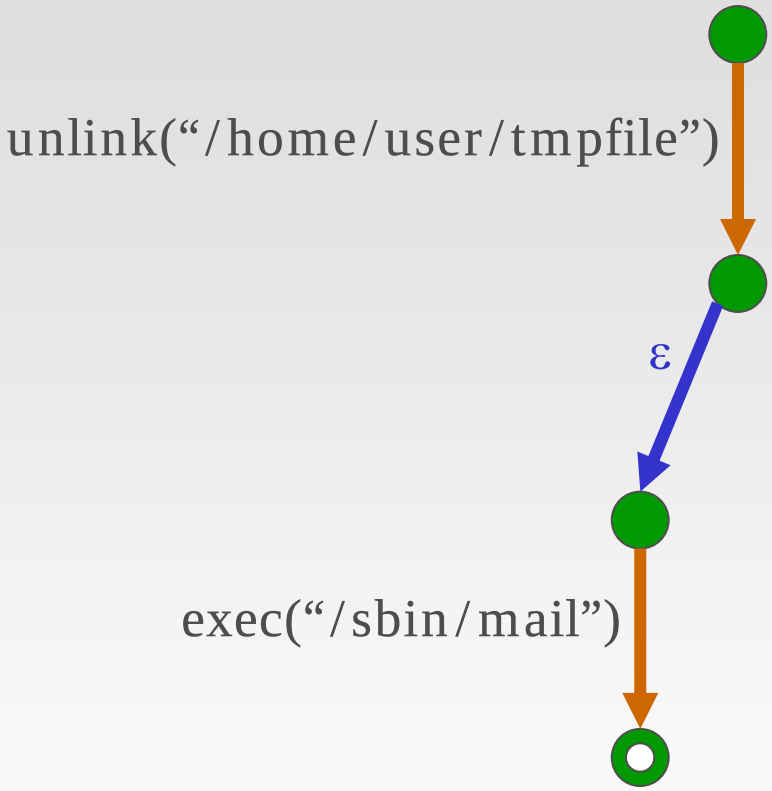
Model Instantiation



Environment:

./a.out

Model Instantiation

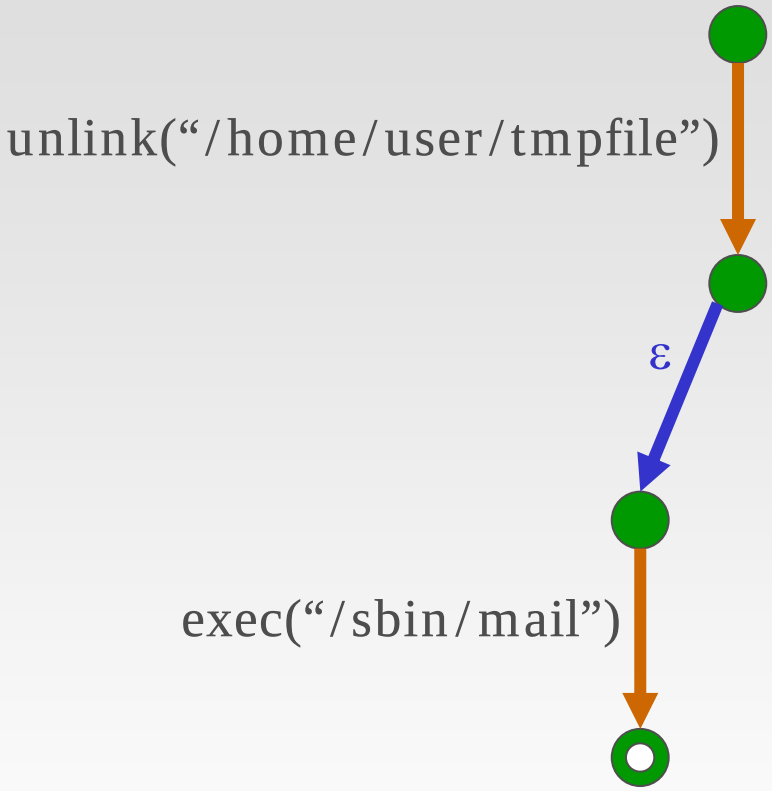


Environment:

./a.out

unlink(“/home/user/tmpfile”)
unlink(“/sbin/mail”)
link(“/bin/sh”, “/sbin/mail”)
exec(“/sbin/mail”)

Model Instantiation

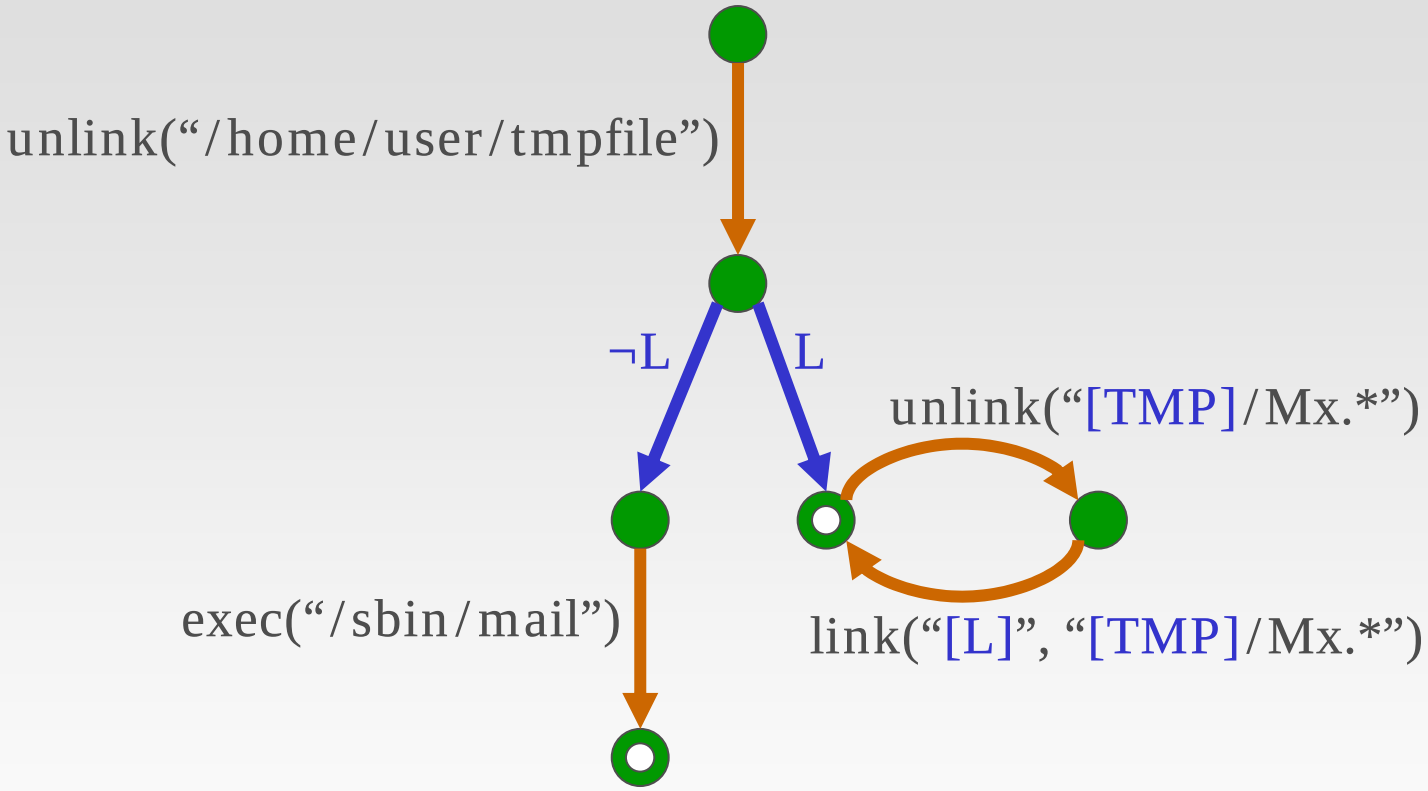


Environment:

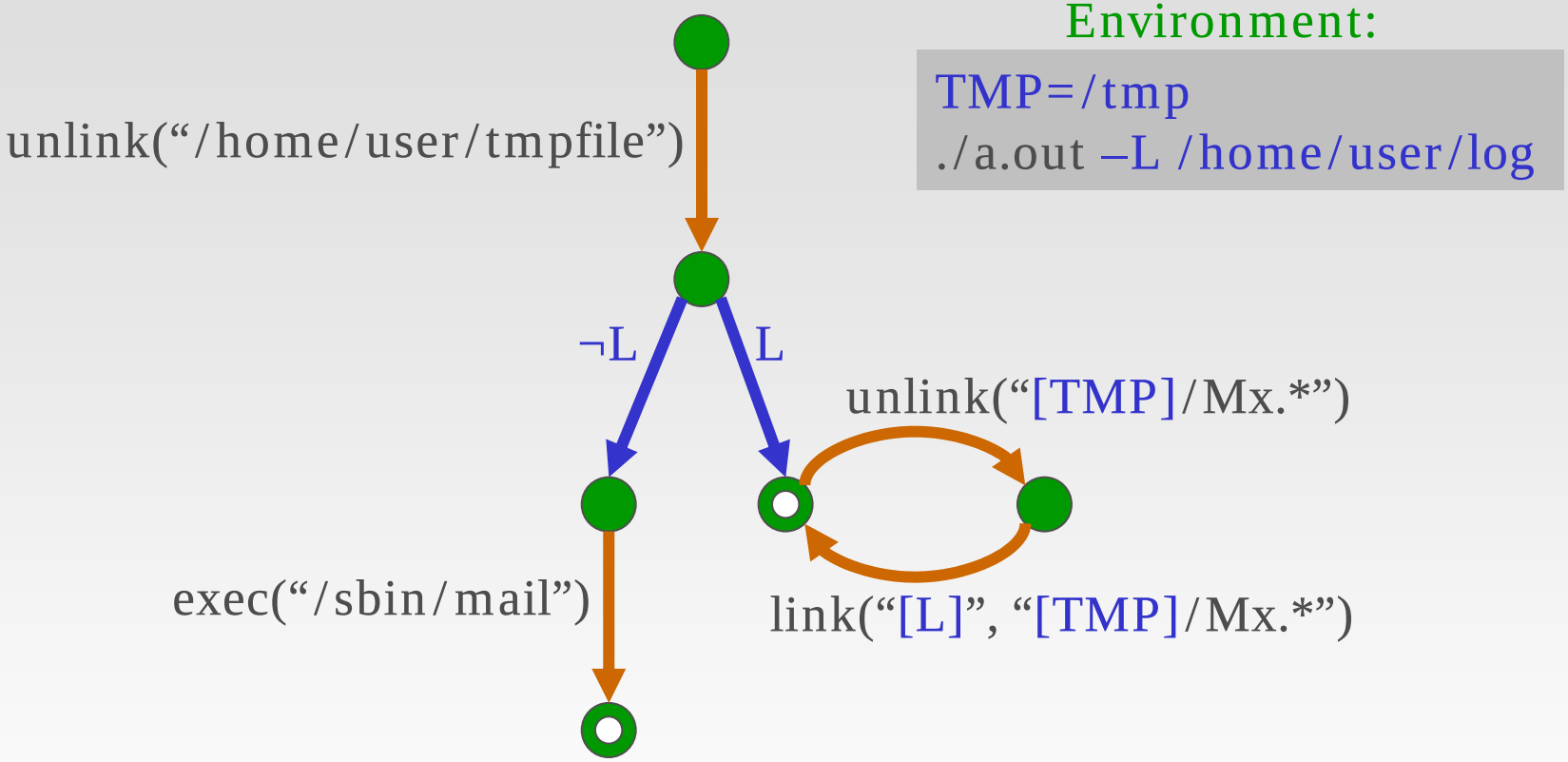
./a.out

unlink(“/home/user/tmpfile”)
unlink(“/sbin/mail”)
link(“/bin/mail”, “/sbin/mail”)
exec(“/sbin/mail”)

Model Template



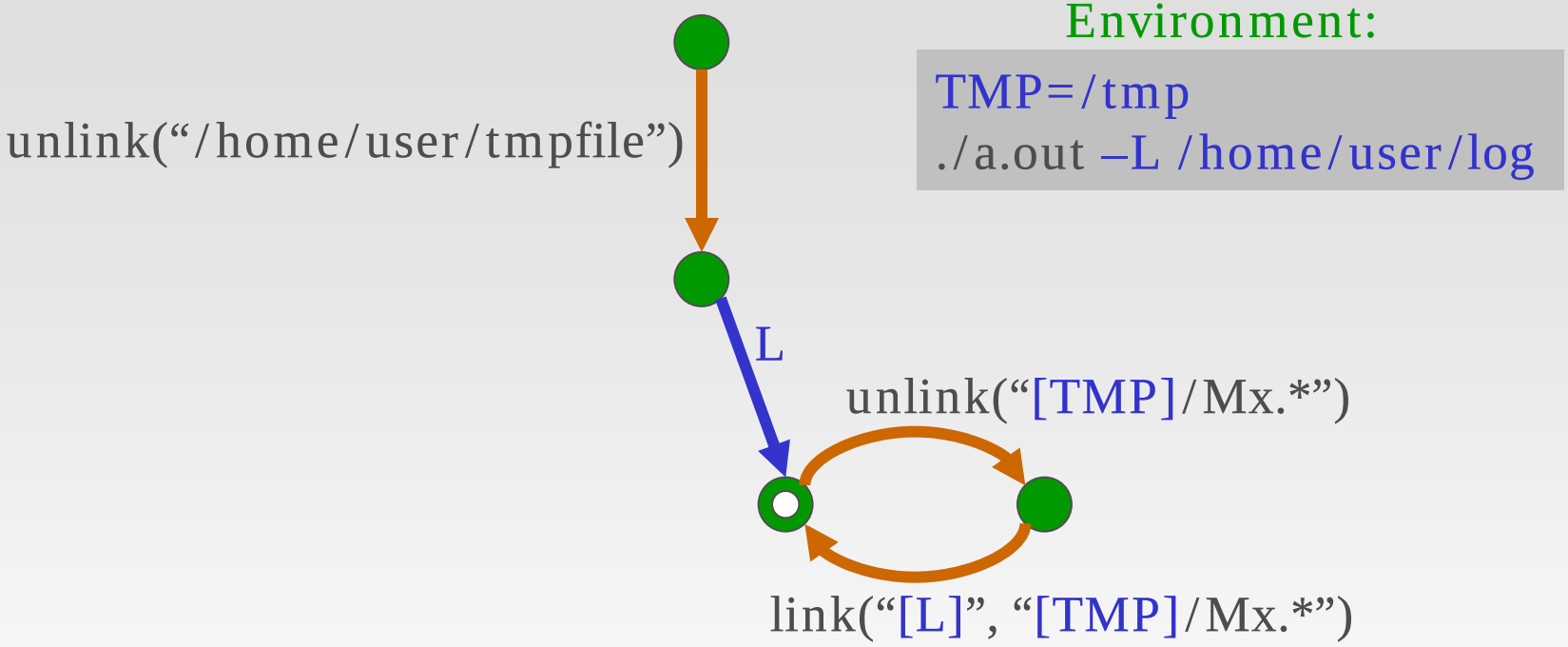
Model Instantiation



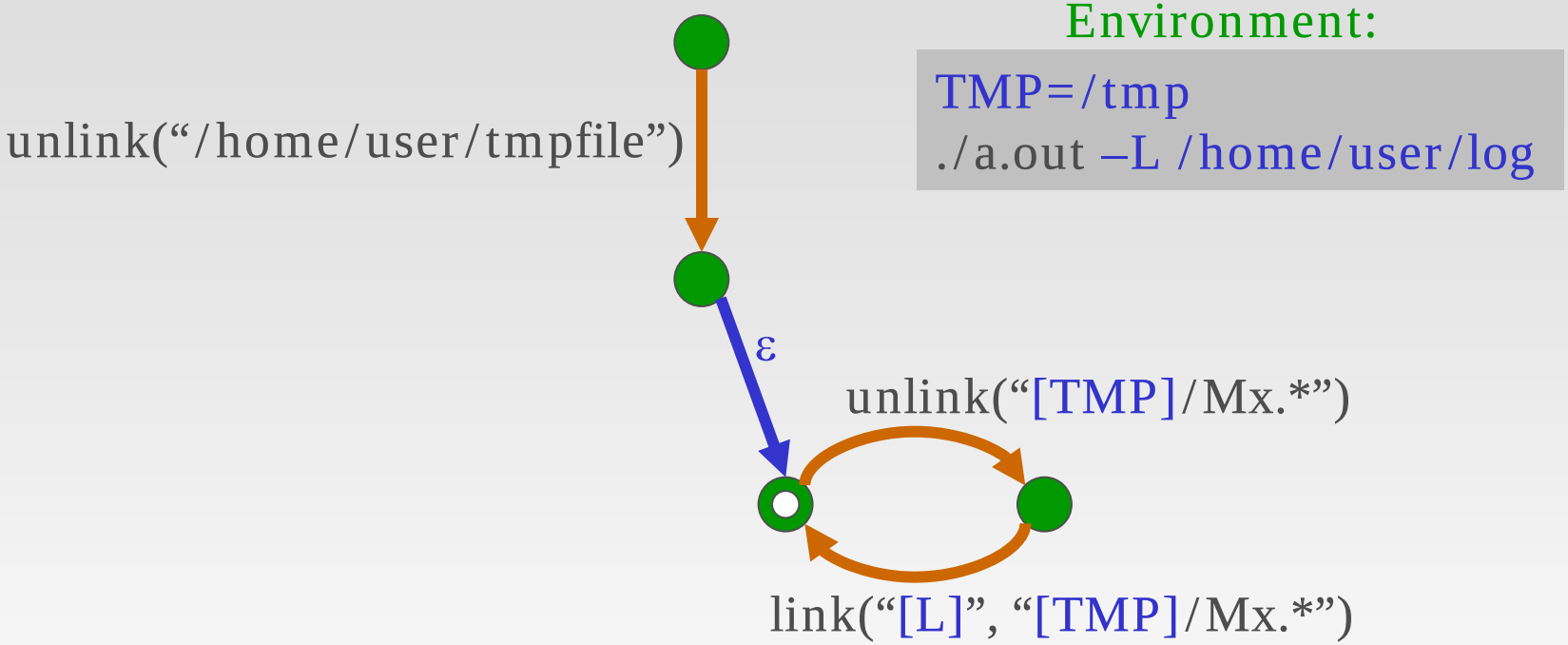
Environment:

TMP=/tmp
./a.out –L /home/user/log

Model Instantiation



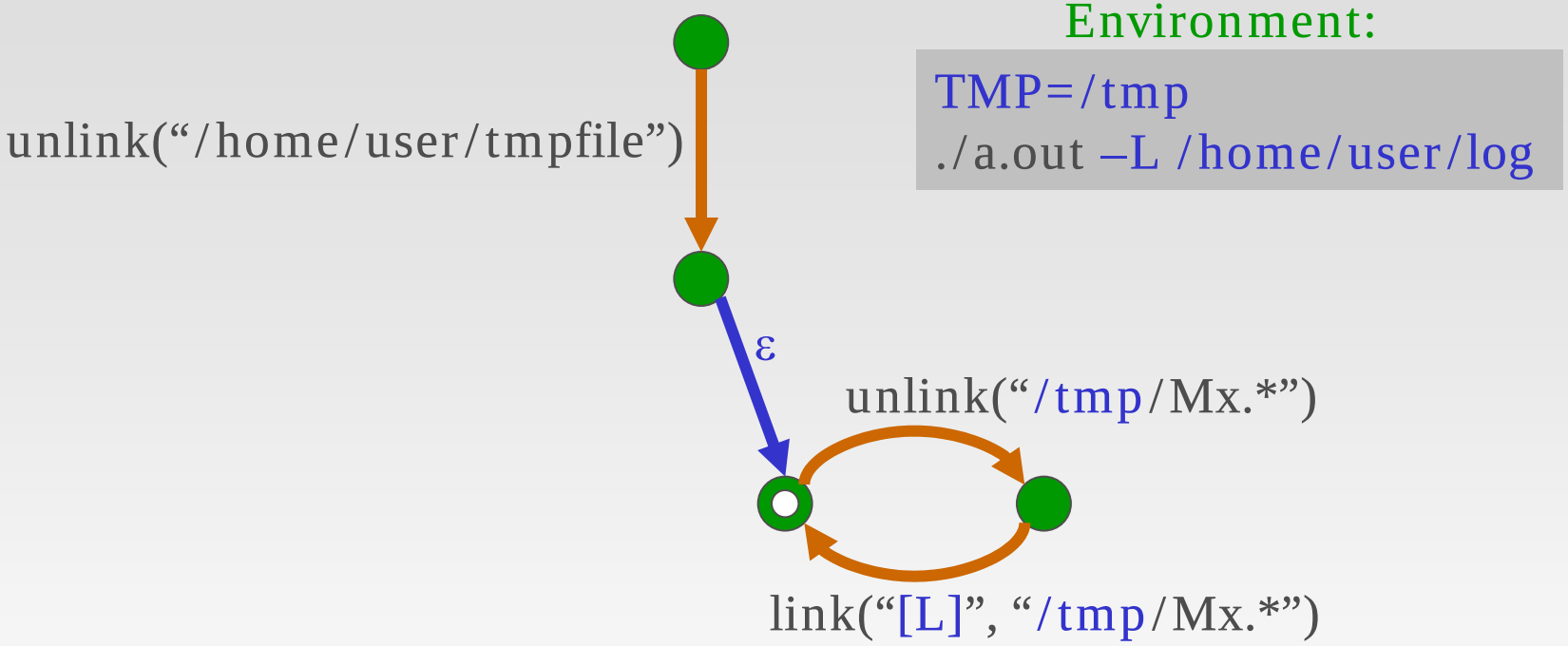
Model Instantiation



Environment:

TMP=/tmp
./a.out –L /home/user/log

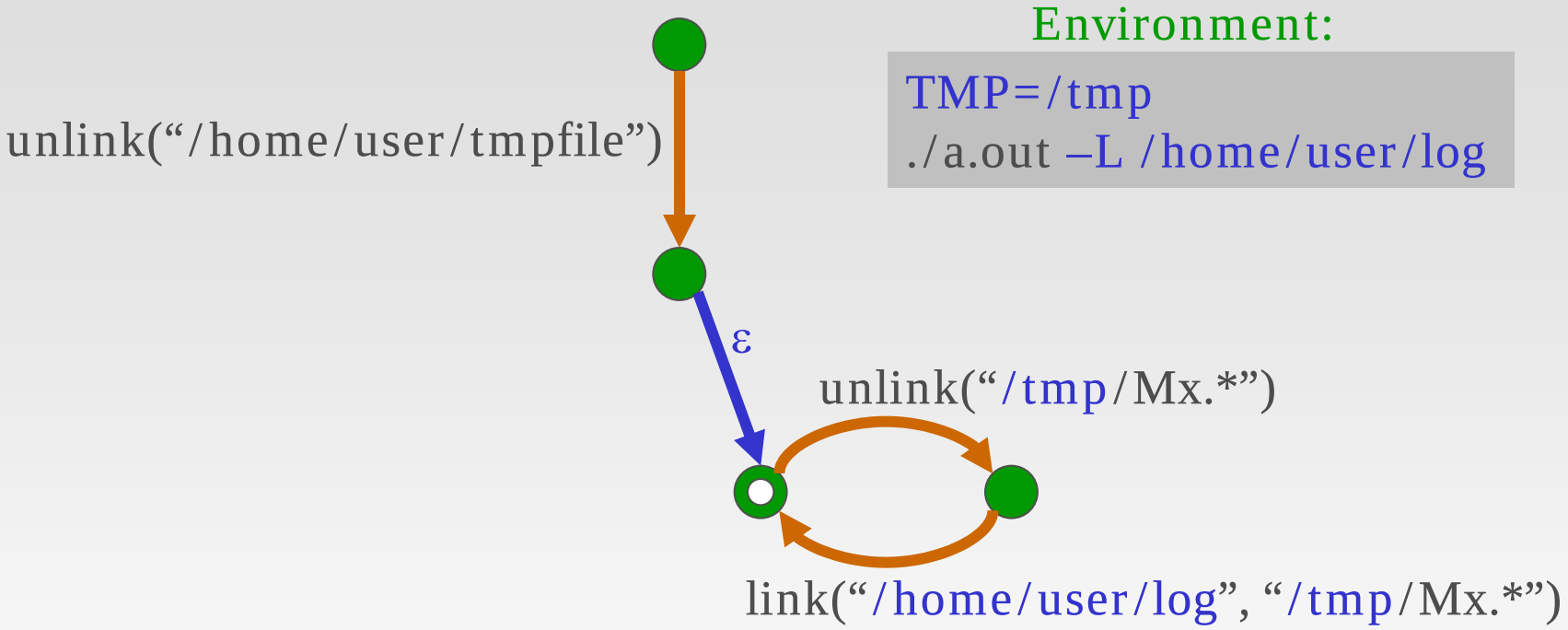
Model Instantiation



Environment:

TMP=/tmp
./a.out –L /home/user/log

Model Instantiation



Relinking attack again not possible

Result

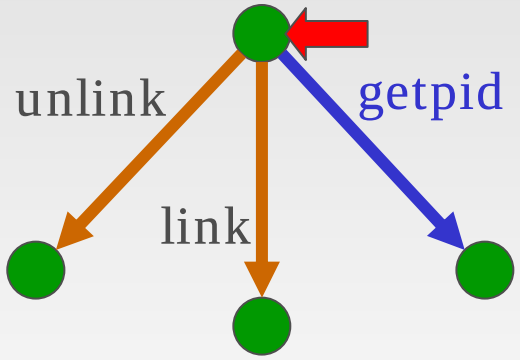
- Program models that
  - Encode data flows among shared objects
  - Enforce arguments in specific calling contexts
  - Adapt to execution environment

Test Programs

Program	*Instruction Count*	*Environment Dependencies*
procmail	374,103	Program branching System call arguments open
mailx	207,977	Program branching System call arguments open, creat, unlink
gzip	196,242	System call arguments unlink, chown, chmod
cat	185,844	System call arguments open

Average Reachability Measure

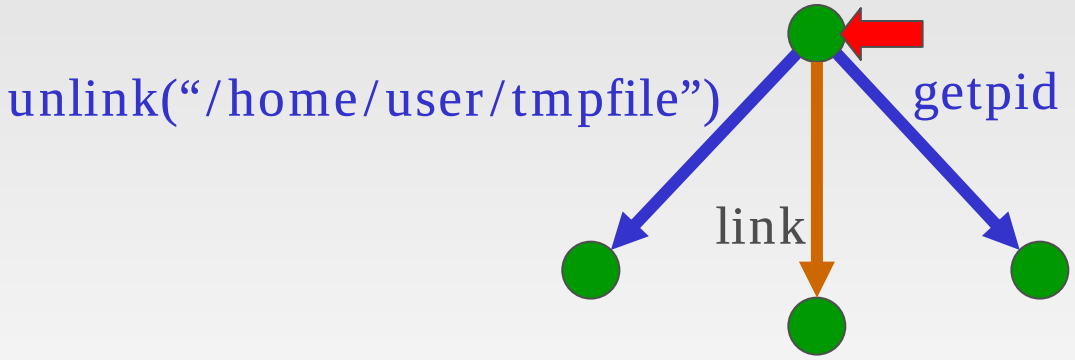
- Average opportunity for attacker to insert malicious system call, with CFL-reachability



- Known argument constraints convert malicious calls to safe calls

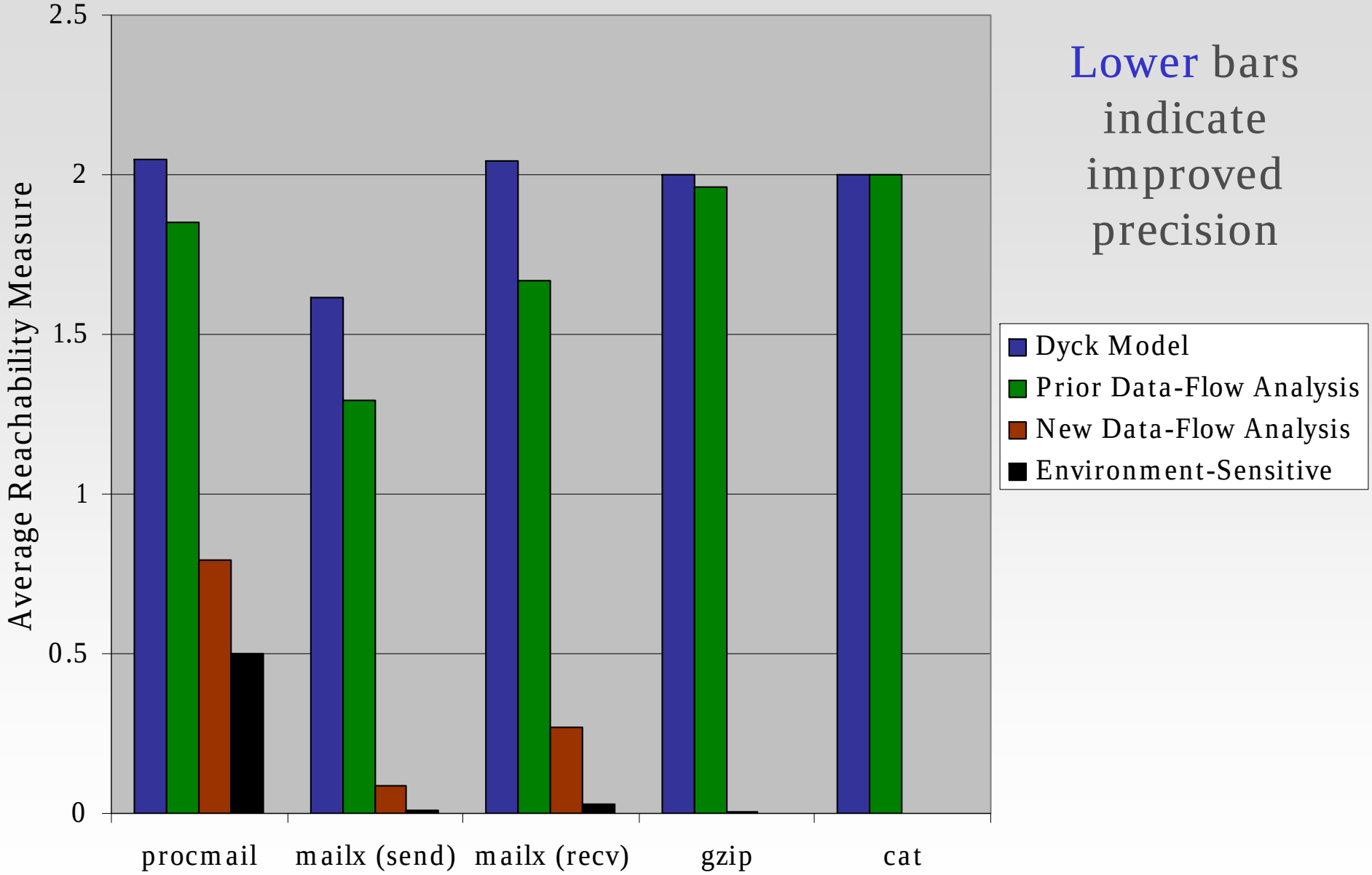
Average Reachability Measure

- Average opportunity for attacker to insert malicious system call, with CFL-reachability



- Known argument constraints convert malicious calls to safe calls

Model Precision



Lower bars indicate improved precision

Summary

- Need to close the gap between behavior allowed by models & actual possible execution.
- Context-sensitive argument recovery improves constraints on attacks via better program analysis.
- Environment-sensitive program models restrict evasion attacks by customizing model to execution environment.

Questions?

... or send us email:

Jonathon T. Giffin	giffin@cs.wisc.edu
David Dagon	dagon@cc.gatech.edu
Somesh Jha	jha@cs.wisc.edu
Wenke Lee	wenke@cc.gatech.edu
Barton P. Miller	bart@cs.wisc.edu