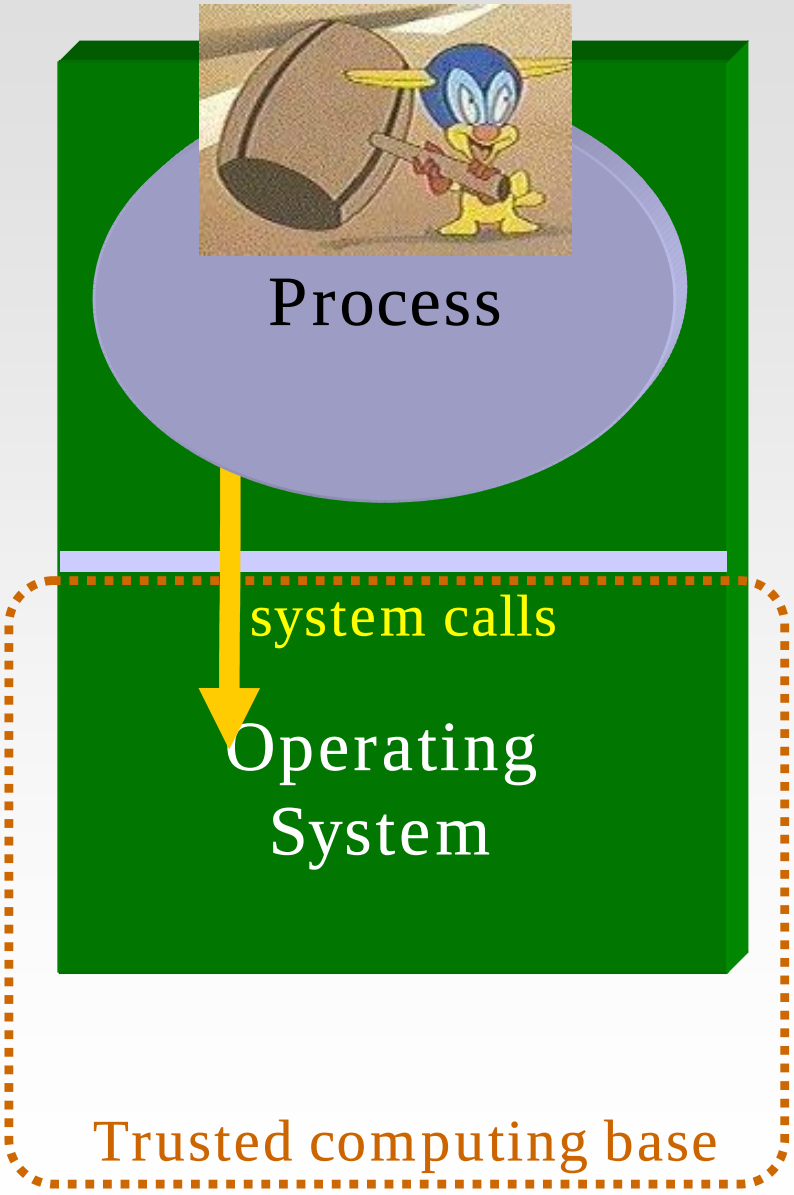
Environment-Sensitive Intrusion Detection

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

Wisconsin Safety Analyzer
University of Wisconsin

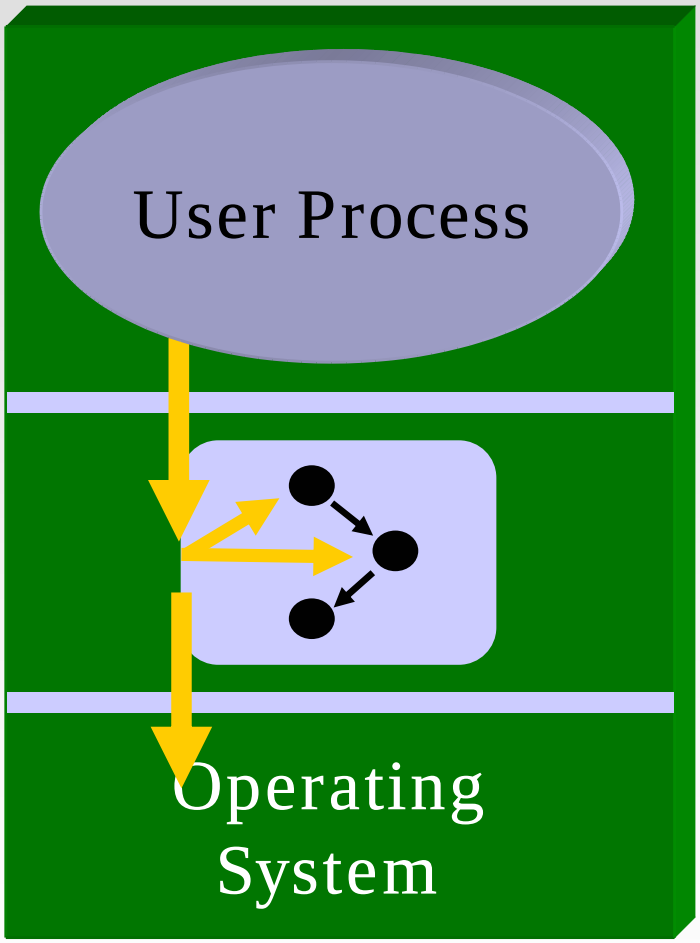
{giffin,jha,bart}@cs.wisc.edu

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 Intrusion Detection



- Detect deviations from normal execution behavior
- Dyck model
  - Defines allowed sequences of system calls
  - Context sensitive for high precision

New Contributions

- Model precision:
  - Context-sensitive data-flow analysis for system call argument recovery
  - Environment-sensitive program models
  - 77% to 100% gain in model precision
- Model evaluation:
  - Average reachability measure
- Static analysis infrastructure:
  - Model construction for dynamically-linked binaries

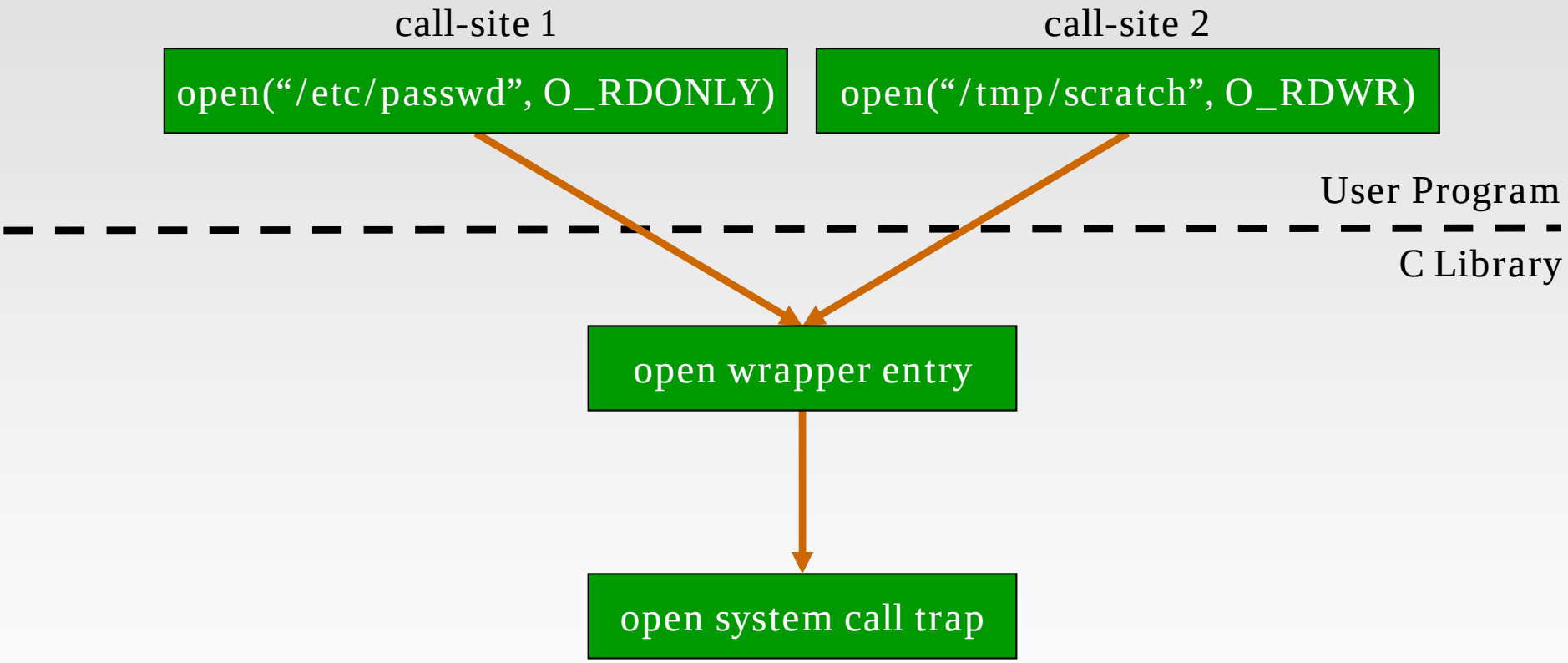
Community Context

- Commercial
  - Startup company by Chieuh & Hsu (SUNY Stony Brook)
    - Rether Networks
  - Control-flow restrictions similar to ours
  - Improperly handle indirect function calls
  - No data-flow analysis
- Academic
  - Dyck model is most advanced & precise statically-constructed control-flow model
  - Most advanced data-flow analysis
  - Our analyses designed to counter attacks proposed by Wagner (UCB), Reiter & Song (CMU), McHugh (CERT)
  - Collaborate with Wenke Lee (Georgia Tech)

Data-Flow Analysis

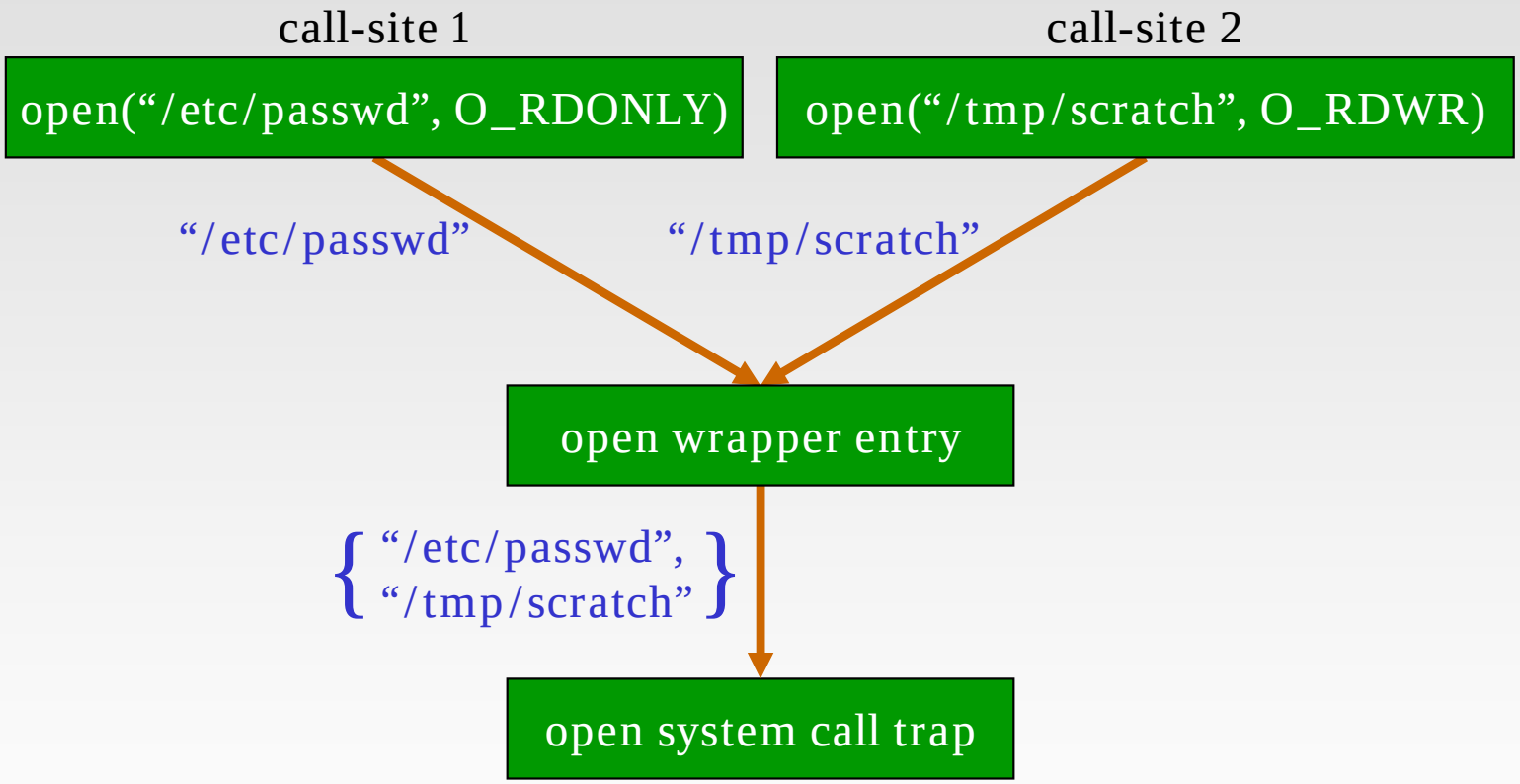
- System-call argument recovery
  - Constrain system-call arguments to only data values used in program code
- Track flows of data possible in program execution
  - Previous context-insensitive analysis loses precision at points of execution convergence
  - New context-sensitive analysis preserves precision
    - Associated arguments
    - Unknown arguments

Execution Convergence



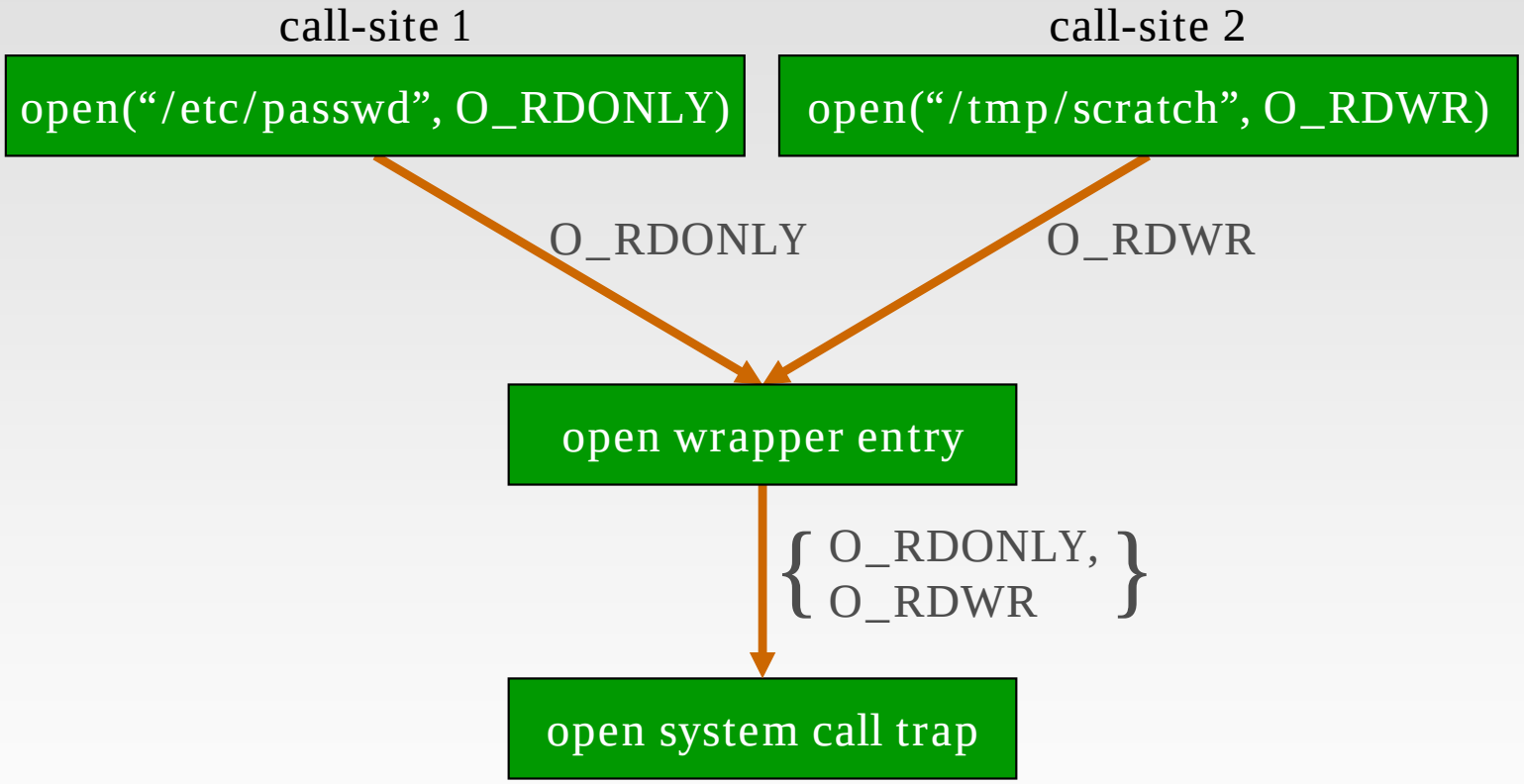
Context-*In*sensitive Data-Flow Analysis

Argument 1 analysis



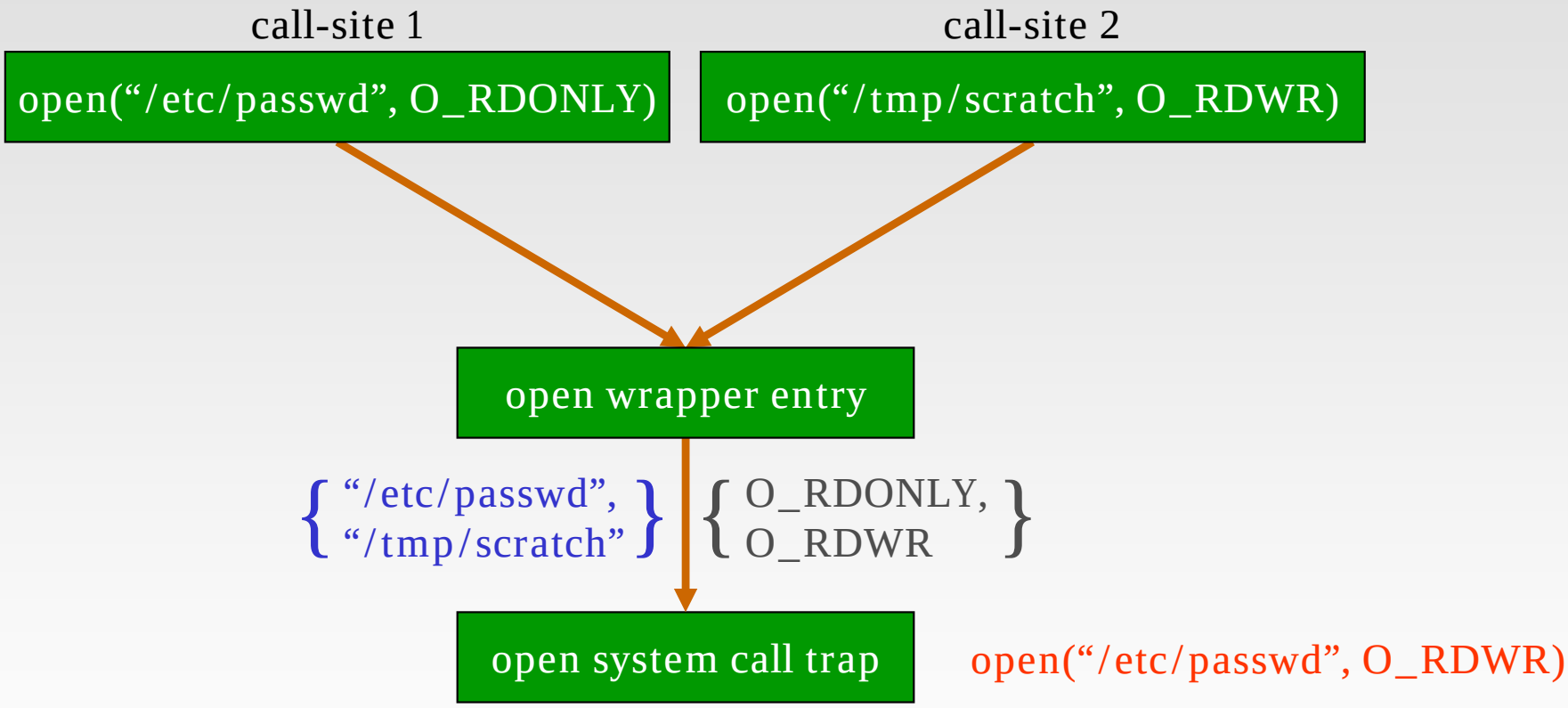
Context-Insensitive Data-Flow Analysis

Argument 2 analysis



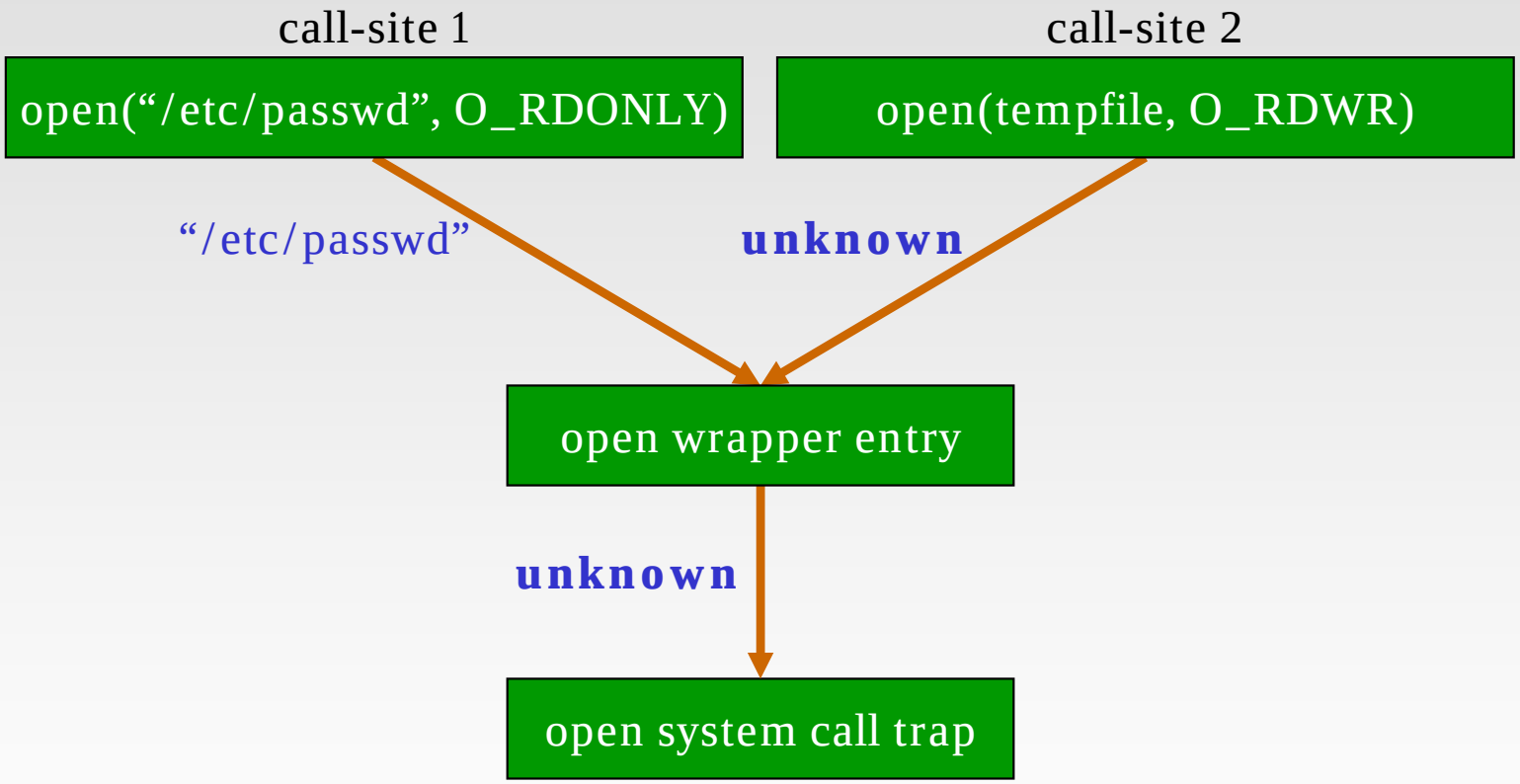
Context-Insensitive Data-Flow Analysis

Argument associations are lost



Context-Insensitive Data-Flow Analysis

Unknown values destroy information

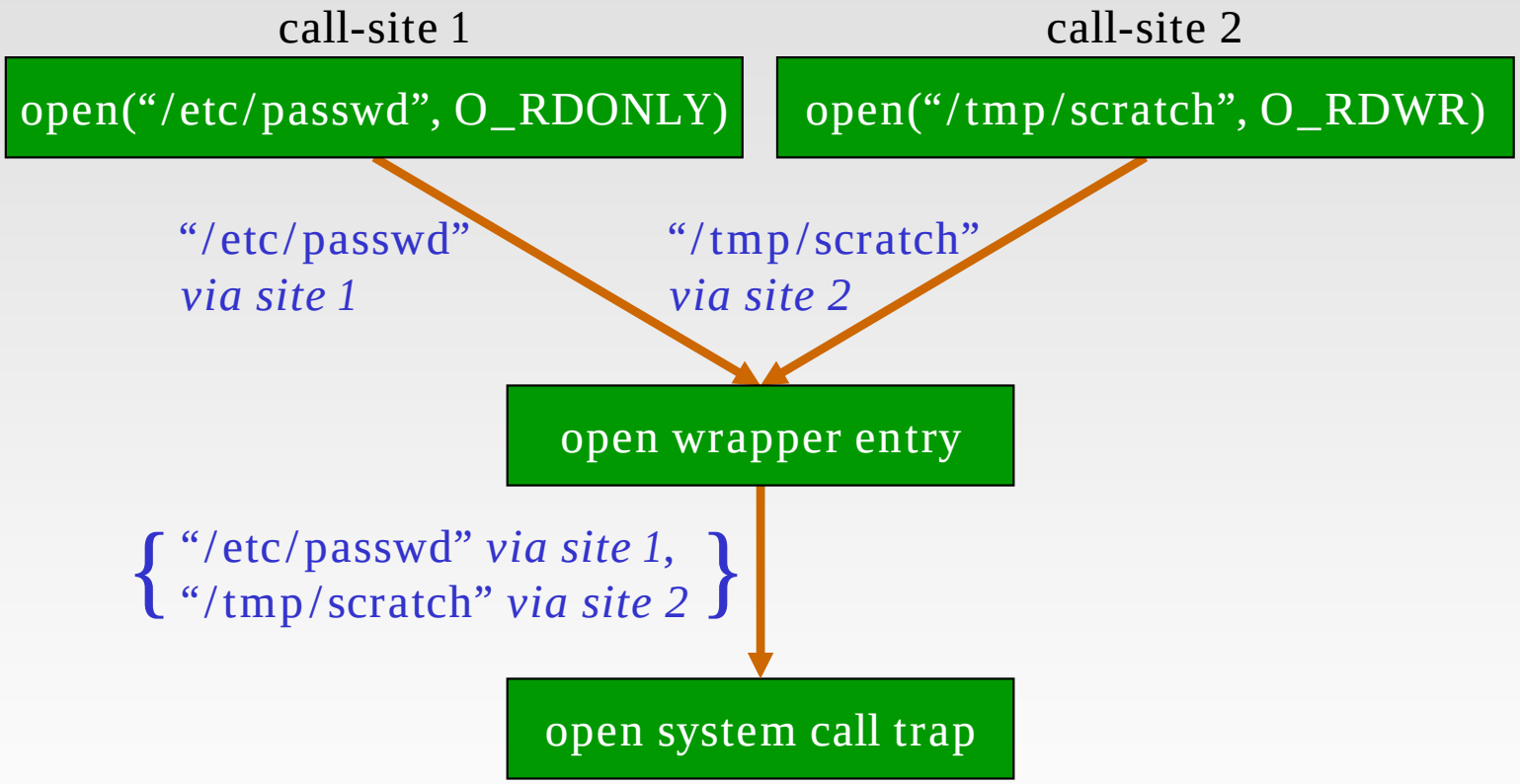


Context-Sensitive Data-Flow Analysis

- Model construction
  - Annotate argument values with calling context passing those arguments
  - Annotated values flow through program to system call trap sites
- Model enforcement
  - Identify calling context by reading function call site addresses from call stack
  - Enforce argument constraints specific to context

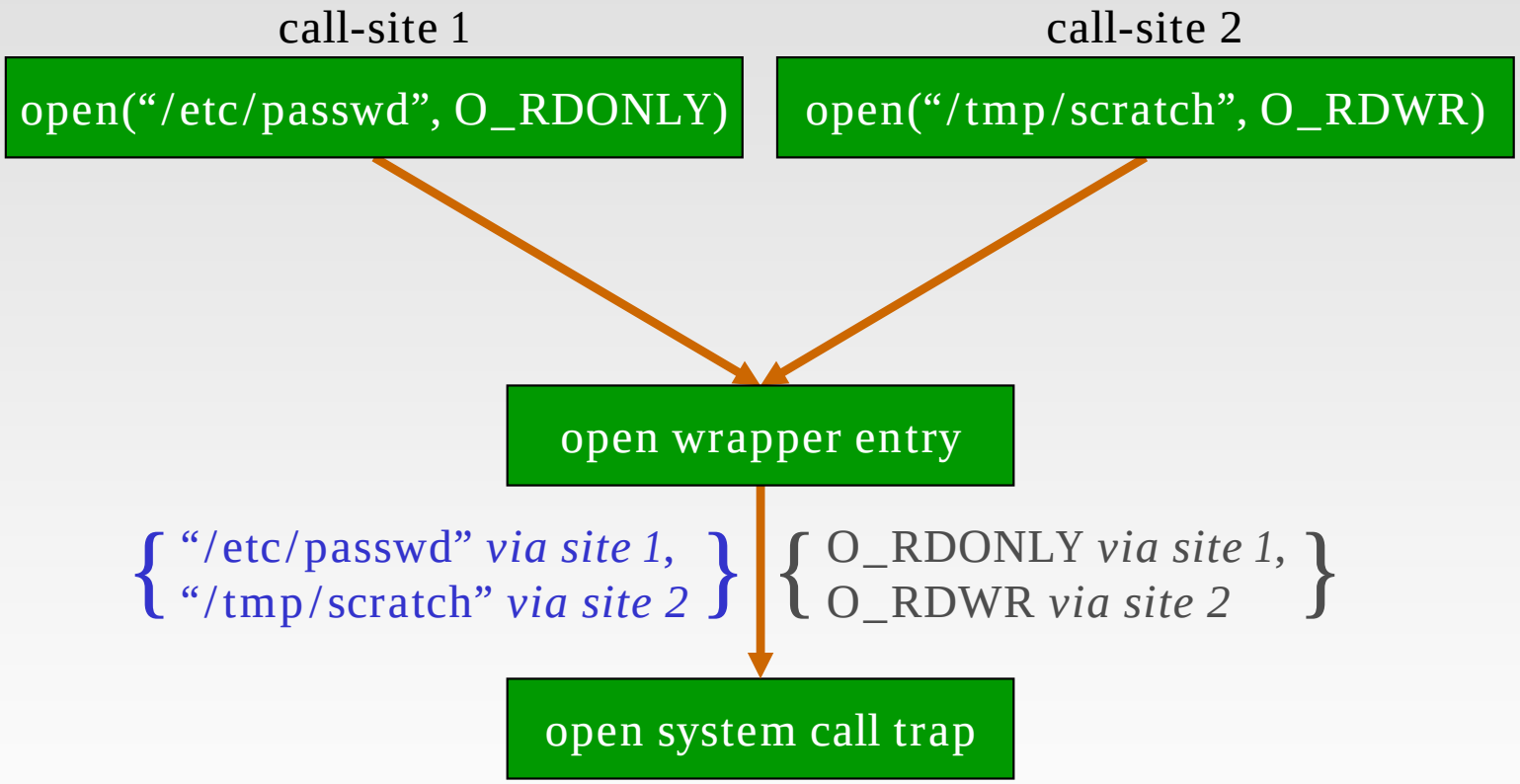
Context-Sensitive Data-Flow Analysis

Argument 1 analysis



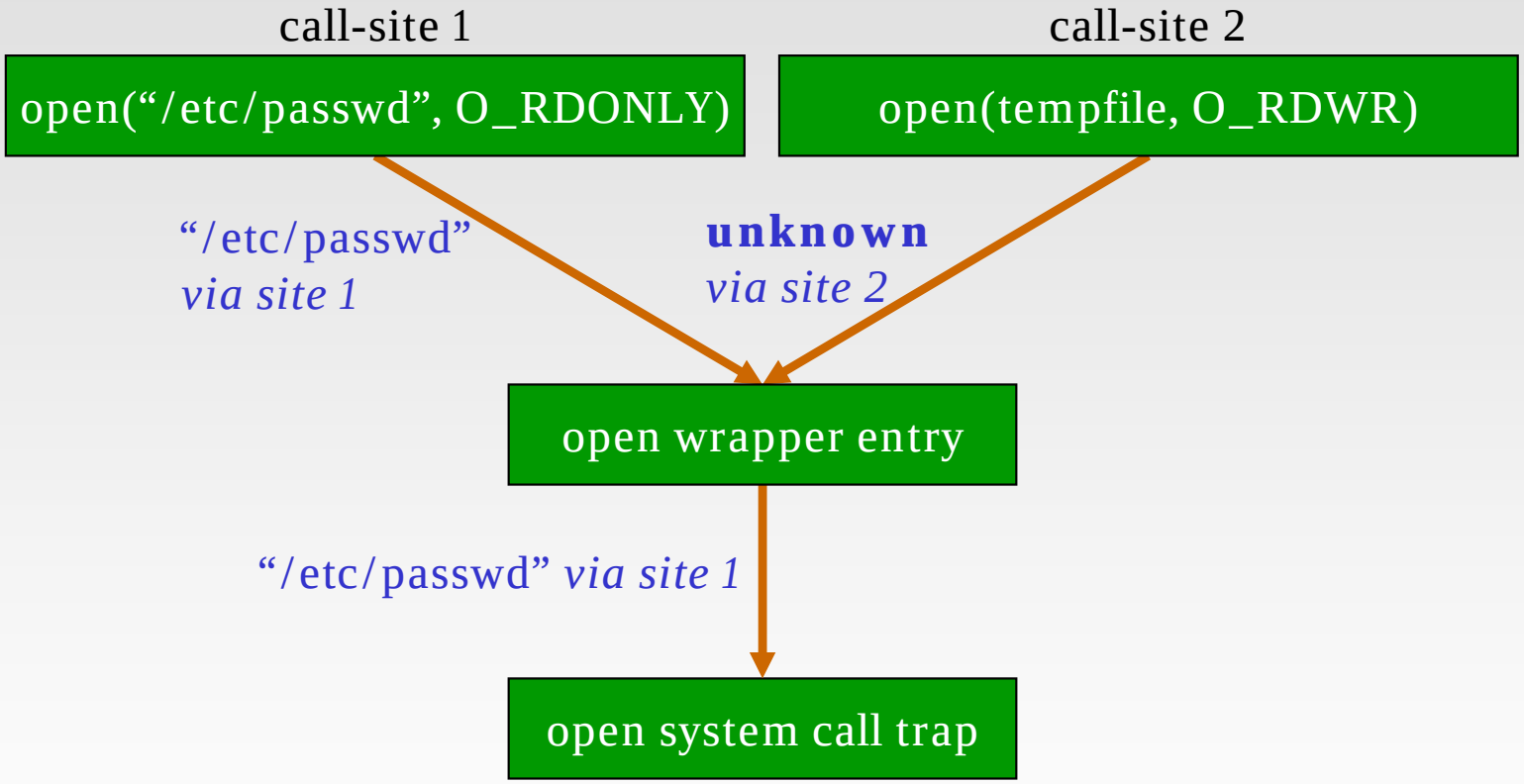
Context-Sensitive Data-Flow Analysis

Context information preserves argument associations



Context-Sensitive Data-Flow Analysis

Unknown values do not destroy information



Environment-Sensitive Models

- Three classes of data values used in programs
  - Static value: not based on input

```
open("/tmp/file", O_RDWR);
```

Environment

  - Input known when program loaded for execution

```
char *tempfile = tempnam(getenv("TMP"), "file");
open(tempfile, O_RDWR);
```

  - Input not known until after execution begins

```
fgets(tmpfile, 100, stdin);
open(tmpfile, O_RDWR);
```

Environment-Sensitive Models

- Environment
  - Command-line parameters
  - Environment variables
  - Configuration files
- Environment-sensitive models restrict allowed execution given environment values

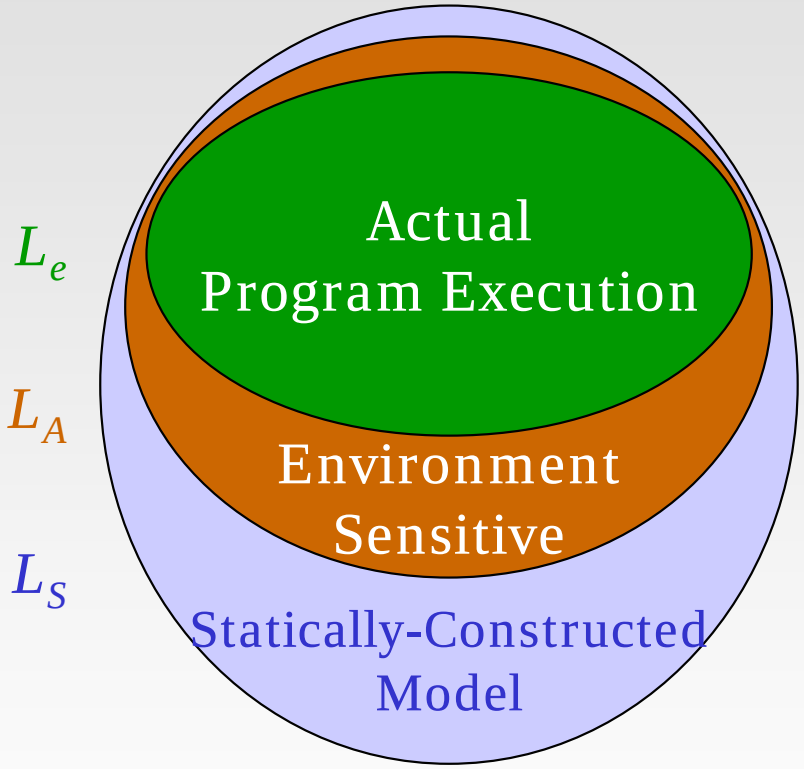
procmail -t

- Requeues failed messages rather than bouncing to sender

httpd -d <pathname>

- Specifies value of server root directory

Environment-Sensitive Models



- Actual program execution in environment e

$$L_e$$

- Statically-constructed model

$$L_S = \bigcup_{e \in E} L_e$$

- Environment-sensitive model

$$L_A = \bigcup_{e \in E' \subseteq E} L_e \subseteq L_S$$

Environment-Sensitive Models

- Construction: build model template
  - System-call arguments
  - Program branch behavior

```
char *tempfile = tempnam(getenv("TMP"), "file");
if (getopt(argc, argv, "L") == 'L') {
  open(tempfile, O_RDWR);
}
```

Environment-Sensitive Models

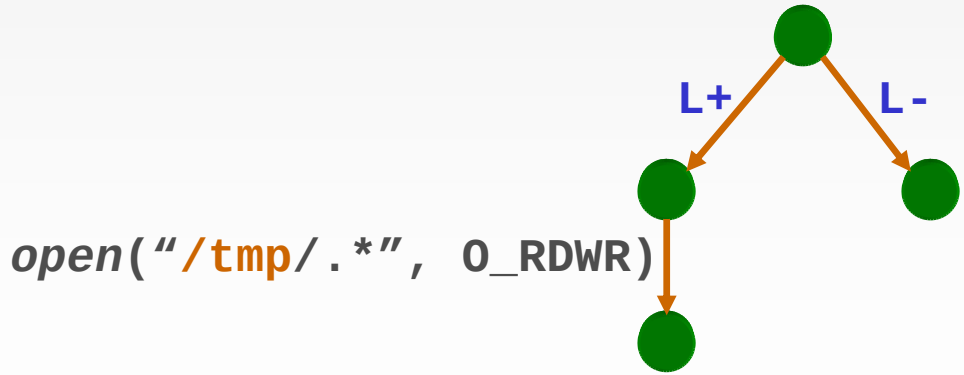
- Enforcement: instantiate model in current environment
  - Update system-call arguments
  - Prune unreachable paths

```
set TMP=/tmp
```

Environment-Sensitive Models

- Enforcement: instantiate model in current environment
  - Update system-call arguments
  - Prune unreachable paths

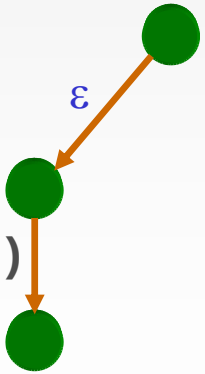
```
set TMP=/tmp
./a.out -L
```



Environment-Sensitive Models

- Enforcement: instantiate model in current environment
  - Update system-call arguments
  - Prune unreachable paths

```
set TMP=/tmp
./a.out -L
```

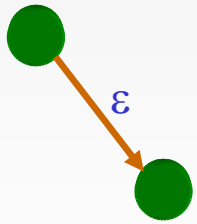


open(“/tmp/.*”, O_RDWR)

Environment-Sensitive Models

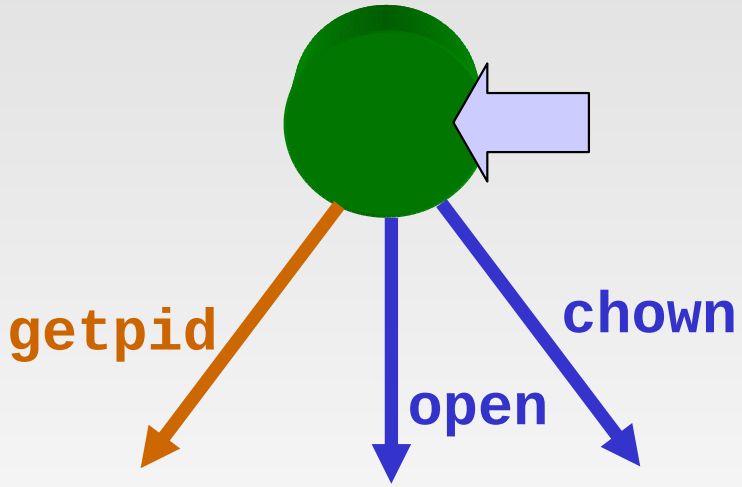
- Enforcement: instantiate model in current environment
  - Update system-call arguments
  - Prune unreachable paths

```
set TMP=/tmp
./a.out
```



Average Reachability Measure

- Average opportunity to undetectably insert malicious system call into system call stream



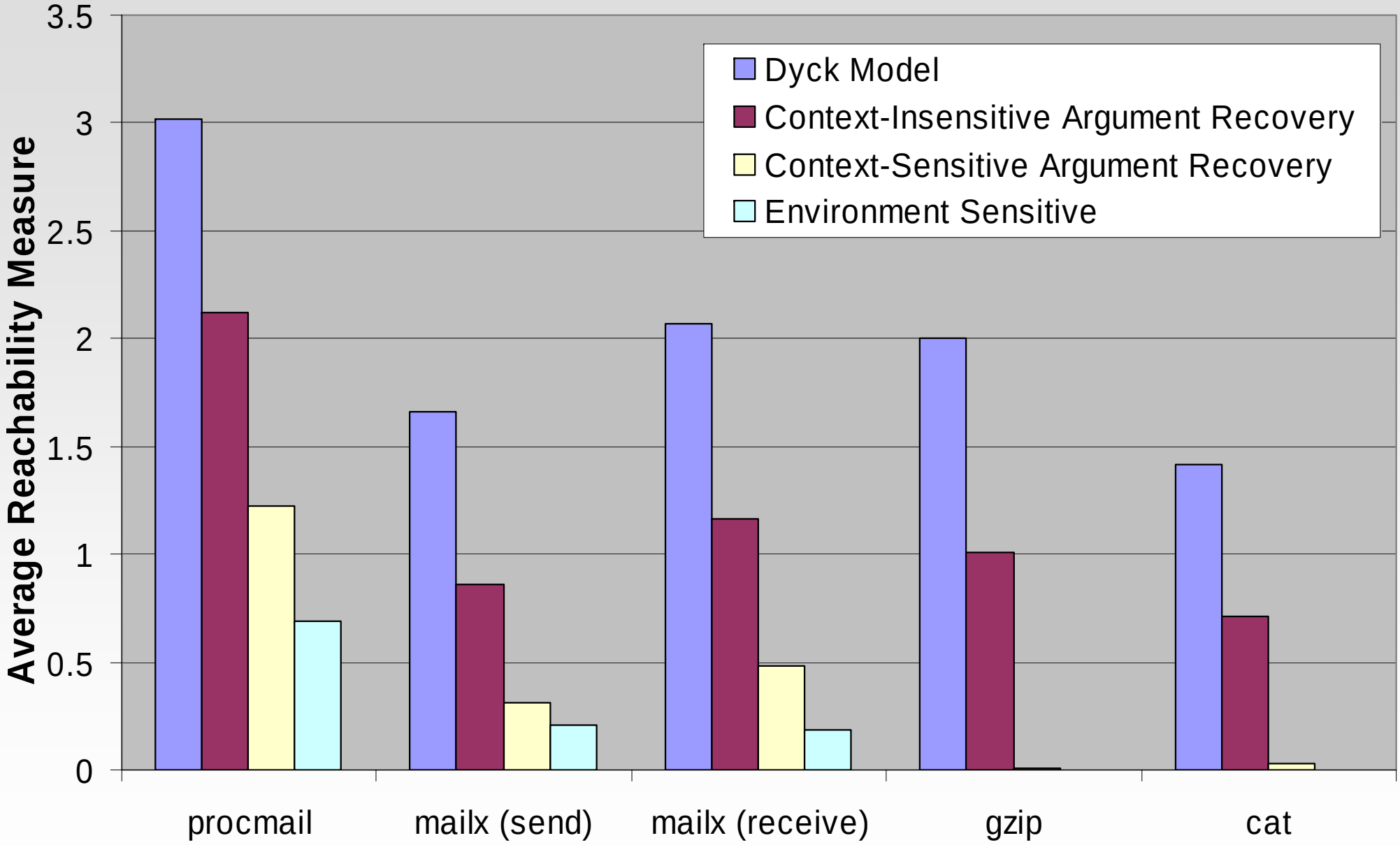
- Context-free reachability suitable for models with function call & return events
- Implemented using WPDS++ library

[CMU/UW MURI & ONR contract to GrammaTech]

Test Programs

Program	Number of Instructions
procmail	374,103
mailx	207,977
gzip	196,242
cat	185,844

Analyses Improve Model Precision



Questions