Automating Configuration Troubleshooting with Dynamic Information Flow Analysis

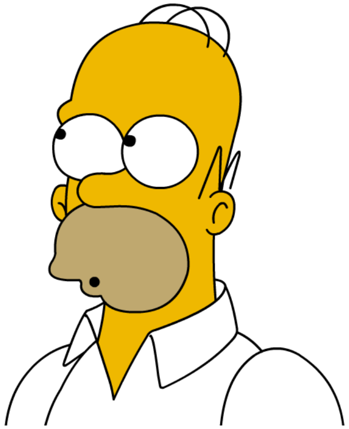
Mona Attariyan

Jason Flinn

University of Michigan

Configuration Troubleshooting Is Difficult

Software systems difficult to configure + Users make mistakes



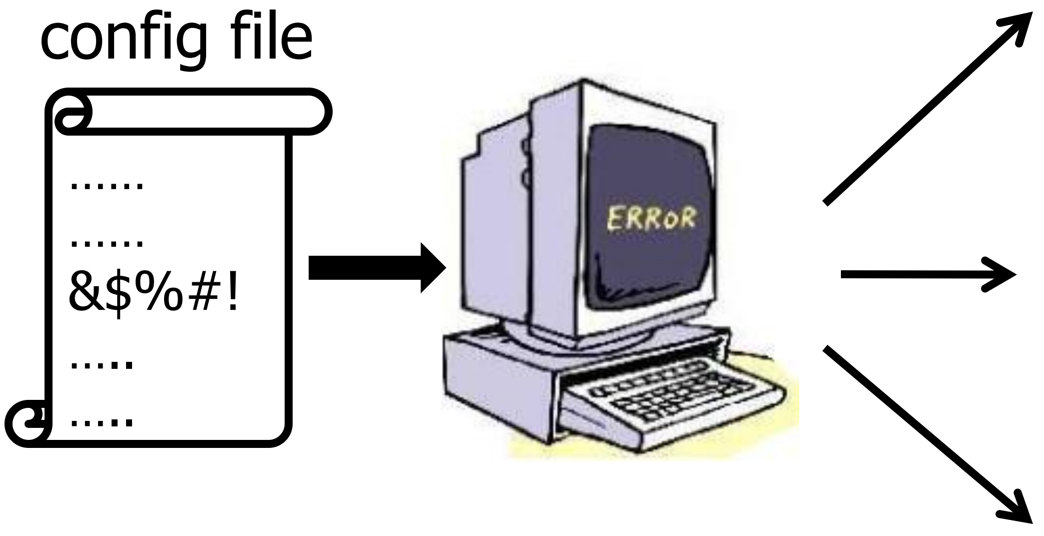
Configuration Troubleshooting Is Difficult

Software systems difficult to configure + Users make mistakes

Misconfigurations happen

Configuration Troubleshooting Is Difficult

What To Do With Misconfiguration?

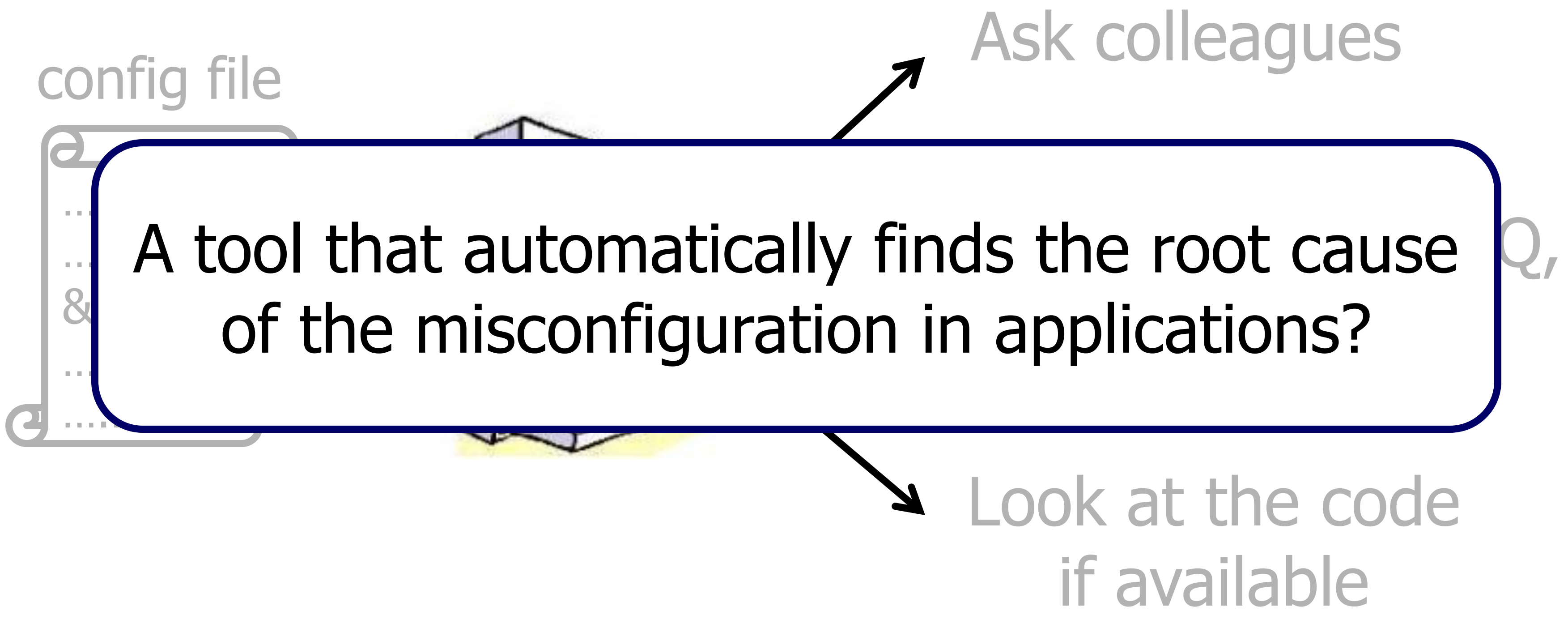


Ask colleagues

Search manual, FAQ, online forums

Look at the code if available

What To Do With Misconfiguration?



A tool that automatically finds the root cause of the misconfiguration in applications?

ConfAid

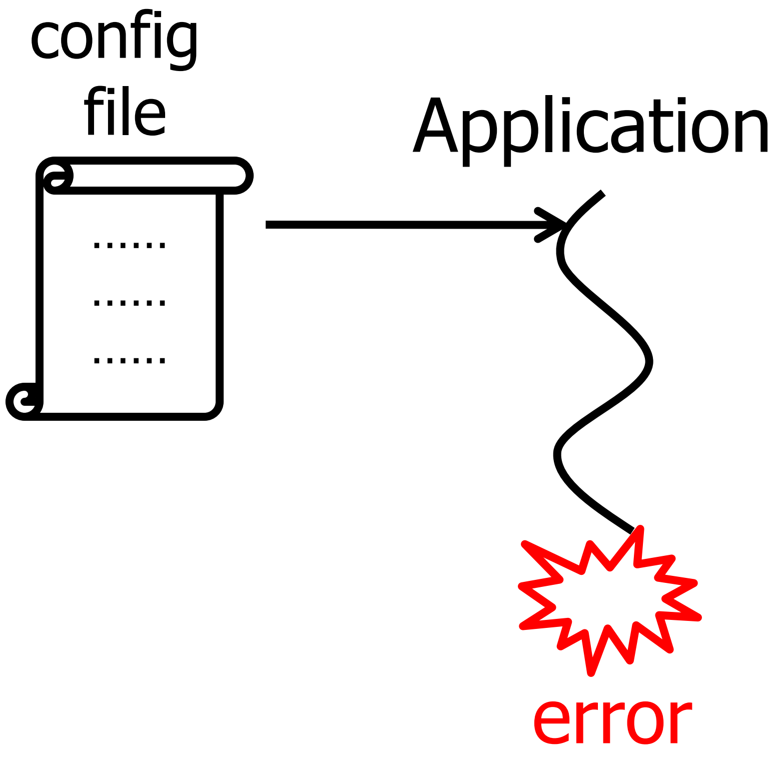
Insight

Application code has enough information to lead us to the root cause

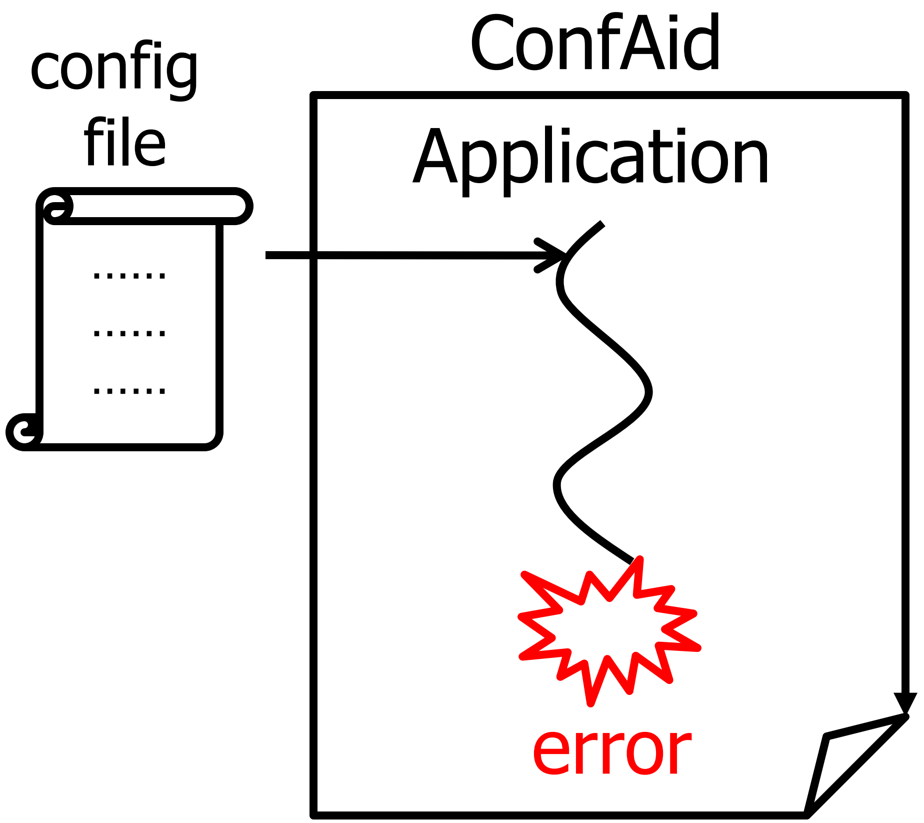
How?

Dynamic information flow analysis on application binaries

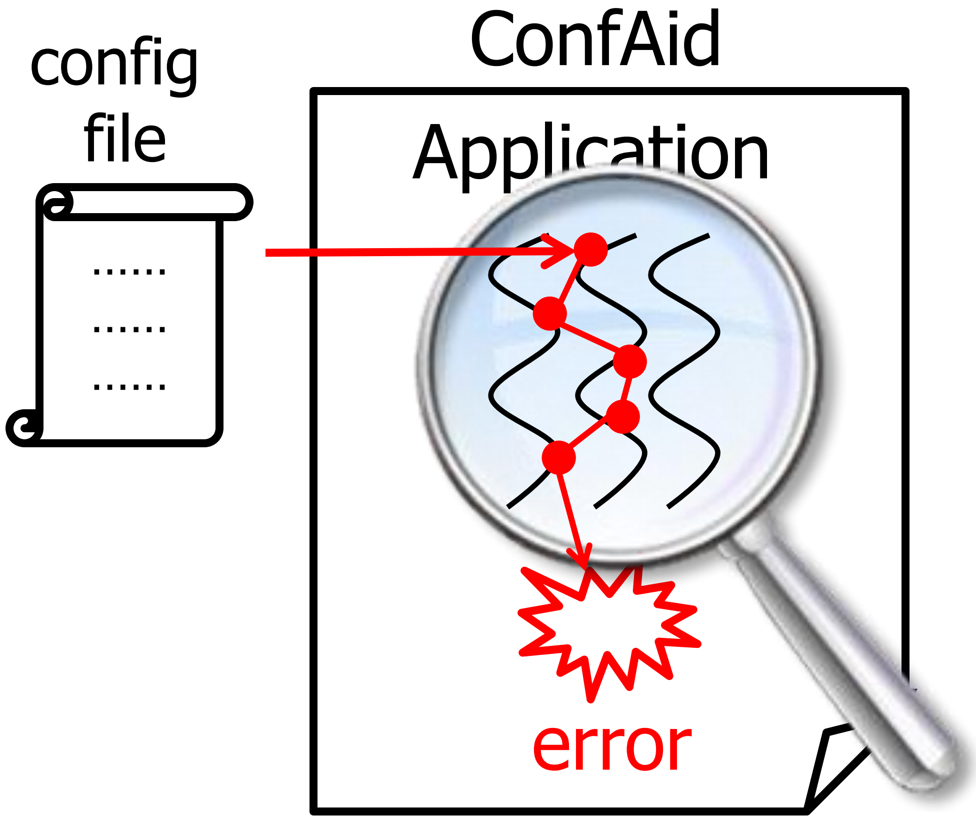
How to Use ConfAid?



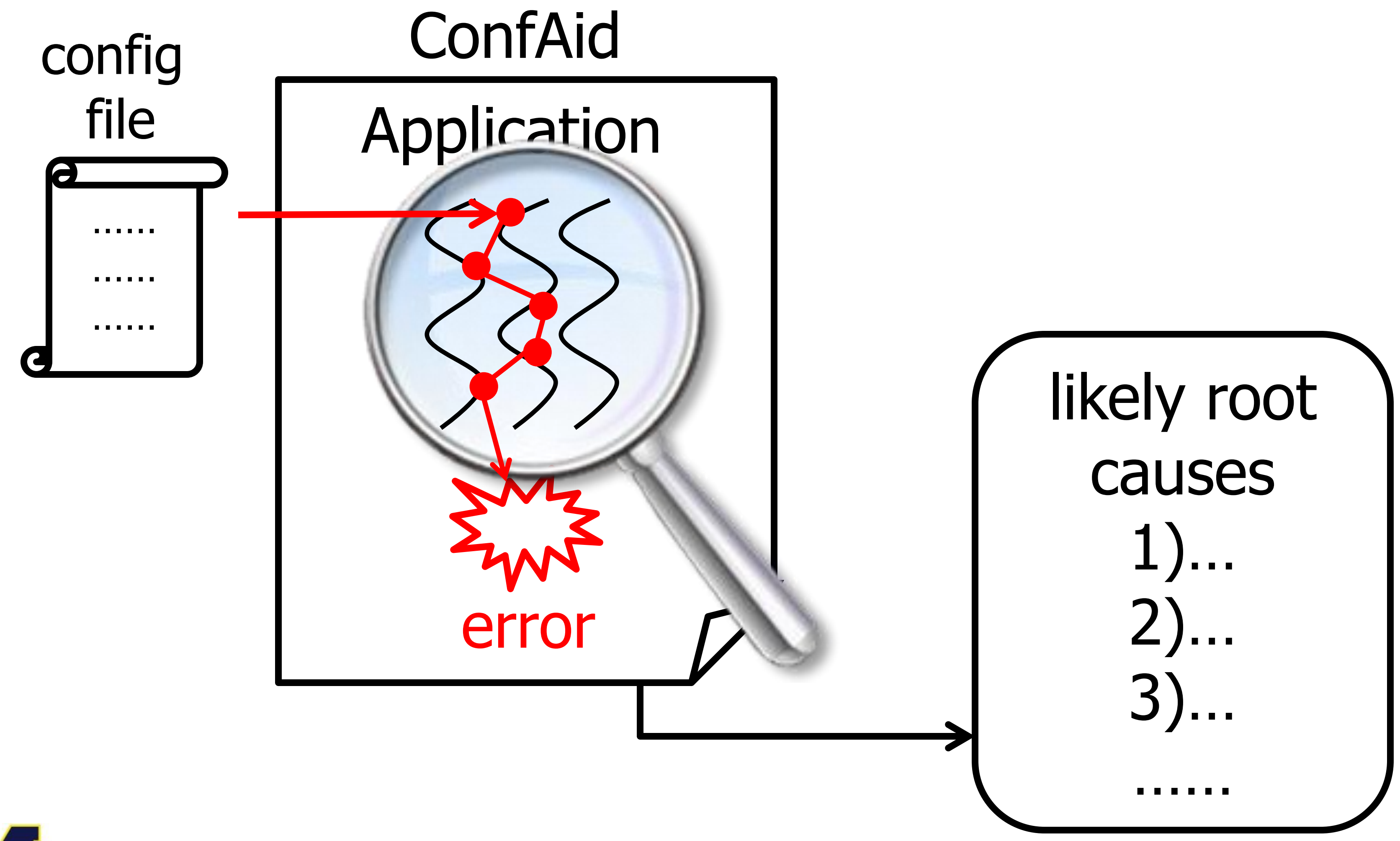
How to Use ConfAid?



How to Use ConfAid?



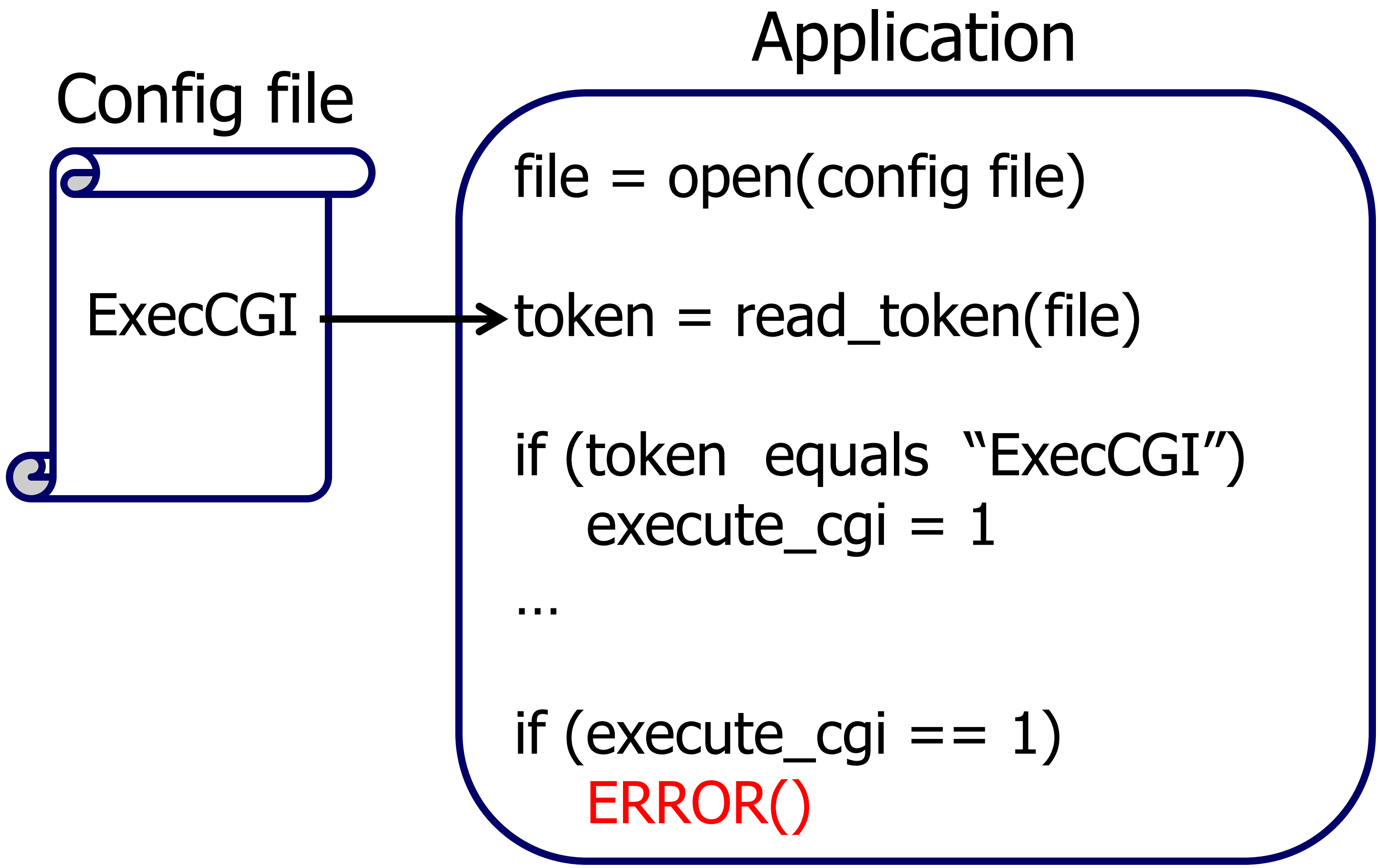
How to Use ConfAid?



Outline

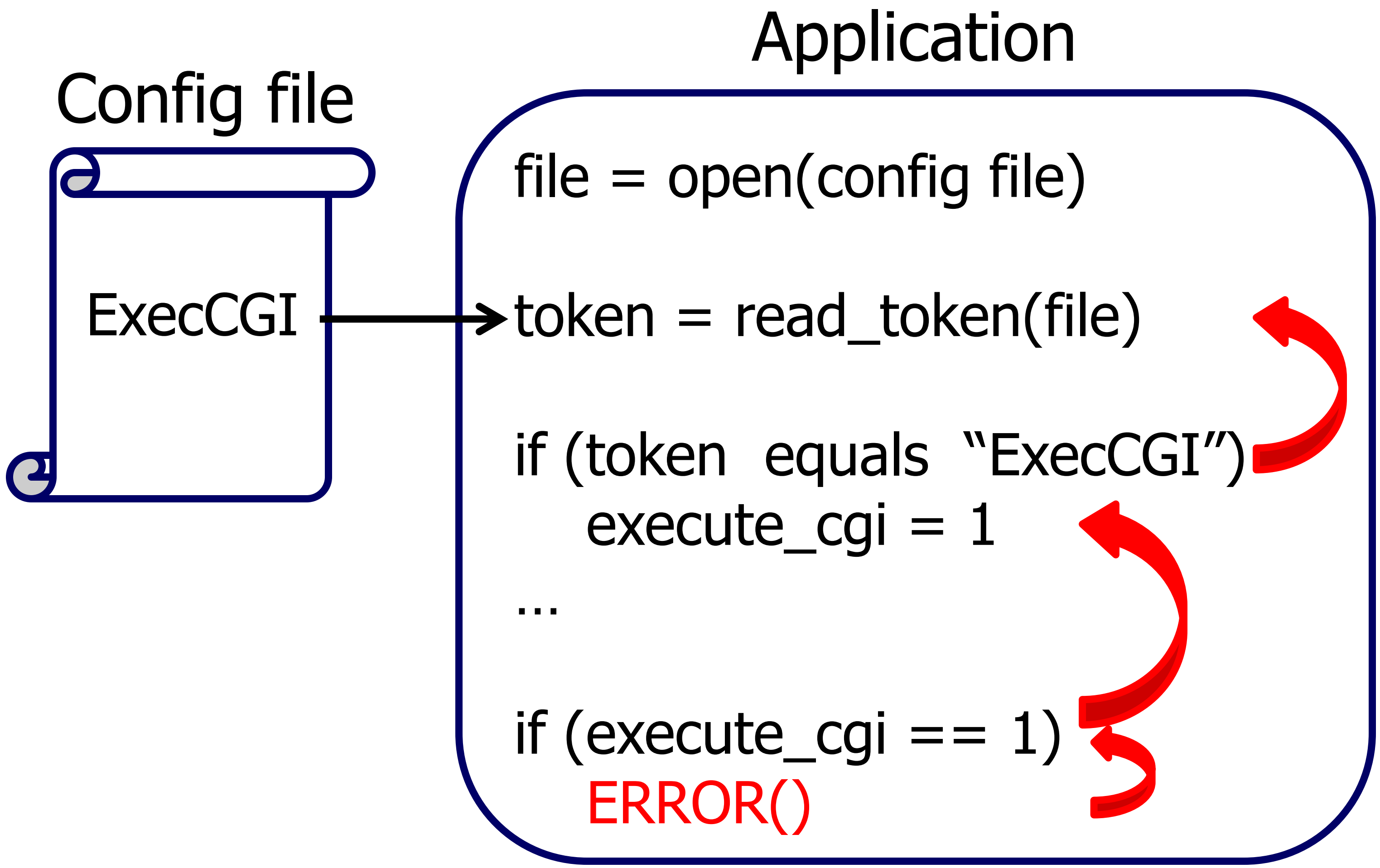
- *Motivation*
- How ConfAid runs
- Information flow analysis algorithms
- Embracing imprecise analysis
- Evaluation
- Conclusion

How Developers Find Root Cause



```
file = open(config file)

token = read_token(file)

if (token  equals  "ExecCGI")
    execute_cgi = 1
...

if (execute_cgi == 1)
    ERROR()
```

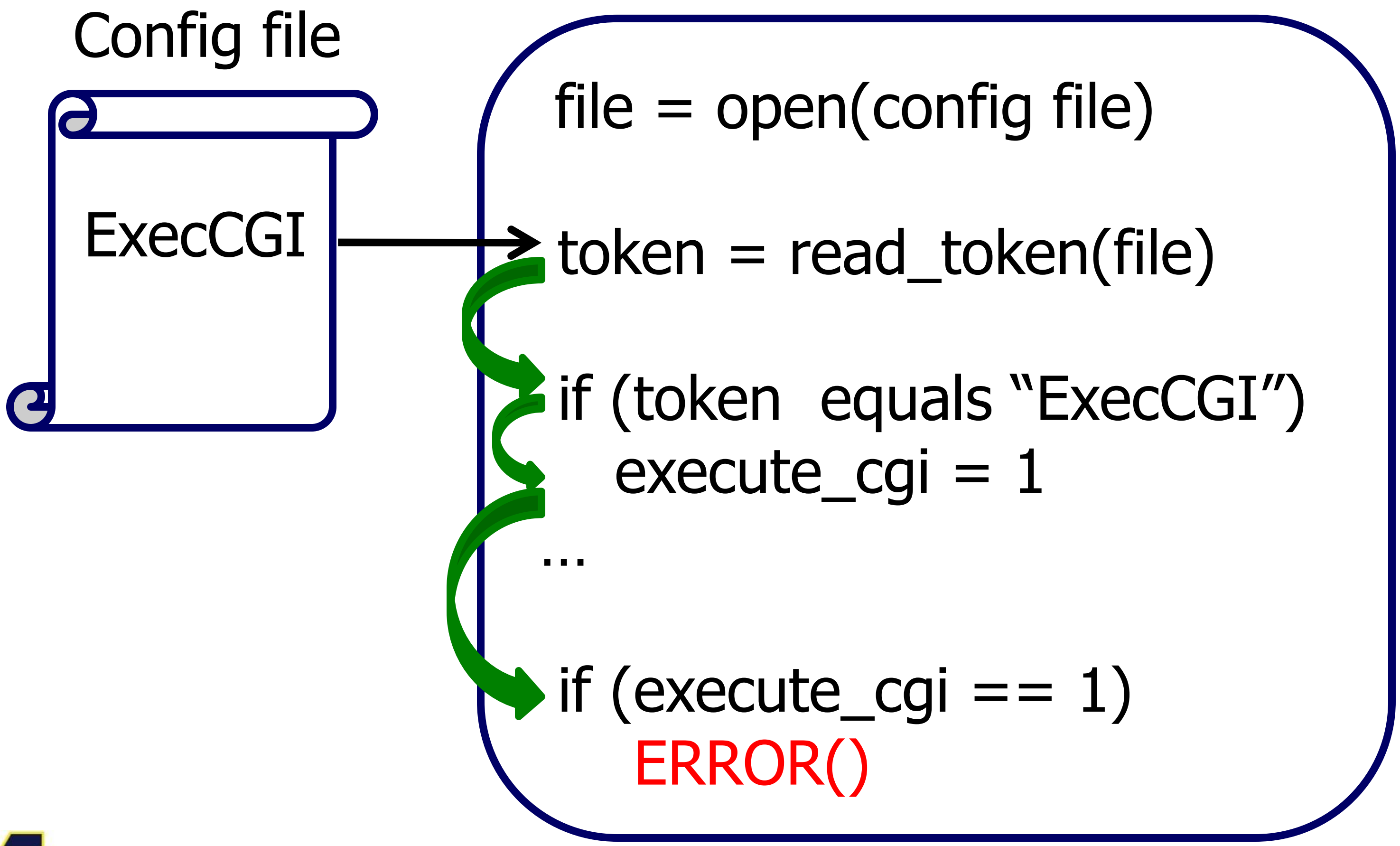
How Developers Find Root Cause



```
file = open(config file)

token = read_token(file)

if (token  equals  "ExecCGI")
    execute_cgi = 1
...

if (execute_cgi == 1)
    ERROR()
```

How ConfAid Finds Root Cause

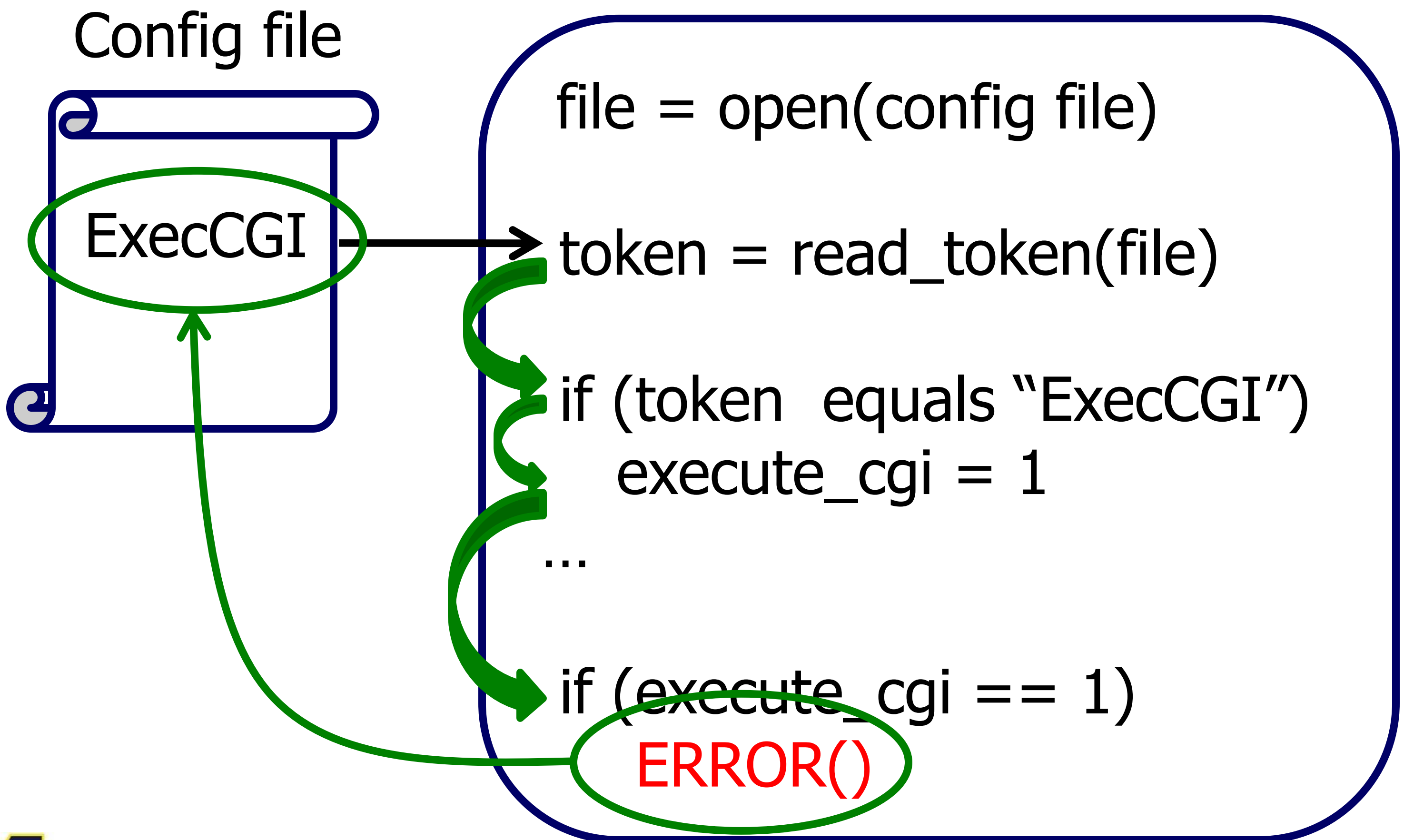
- ConfAid uses taint tracking



```
file = open(config file)

token = read_token(file)

if (token  equals "ExecCGI")
    execute_cgi = 1
...

if (execute_cgi == 1)
    ERROR()
```

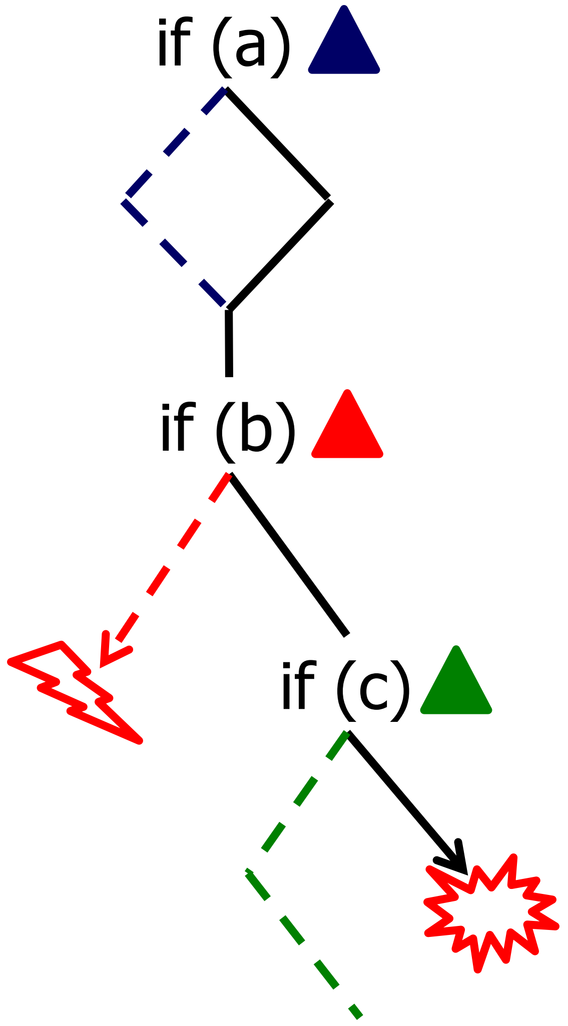
How ConfAid Finds Root Cause

- ConfAid uses taint tracking

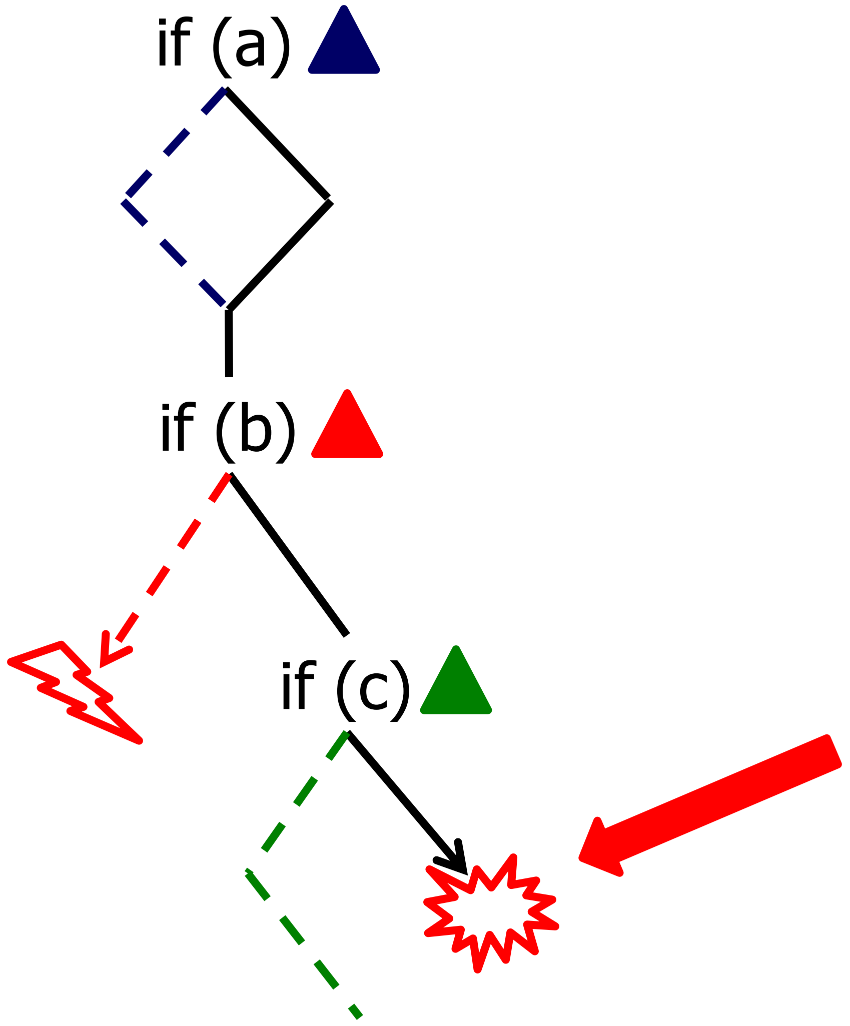


```
file = open(config file)

token = read_token(file)

if (token  equals "ExecCGI")
    execute_cgi = 1
…

if (execute_cgi == 1)
    ERROR()
```

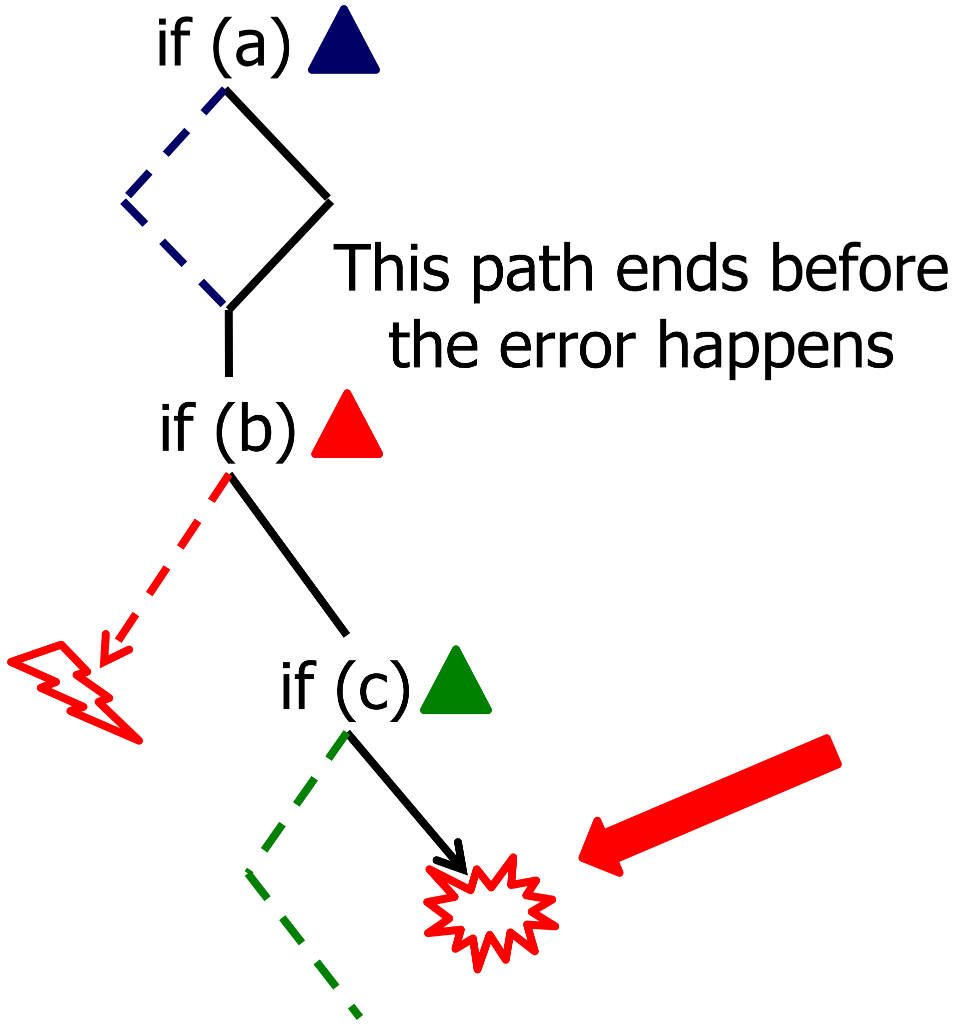
How to Avoid Error?



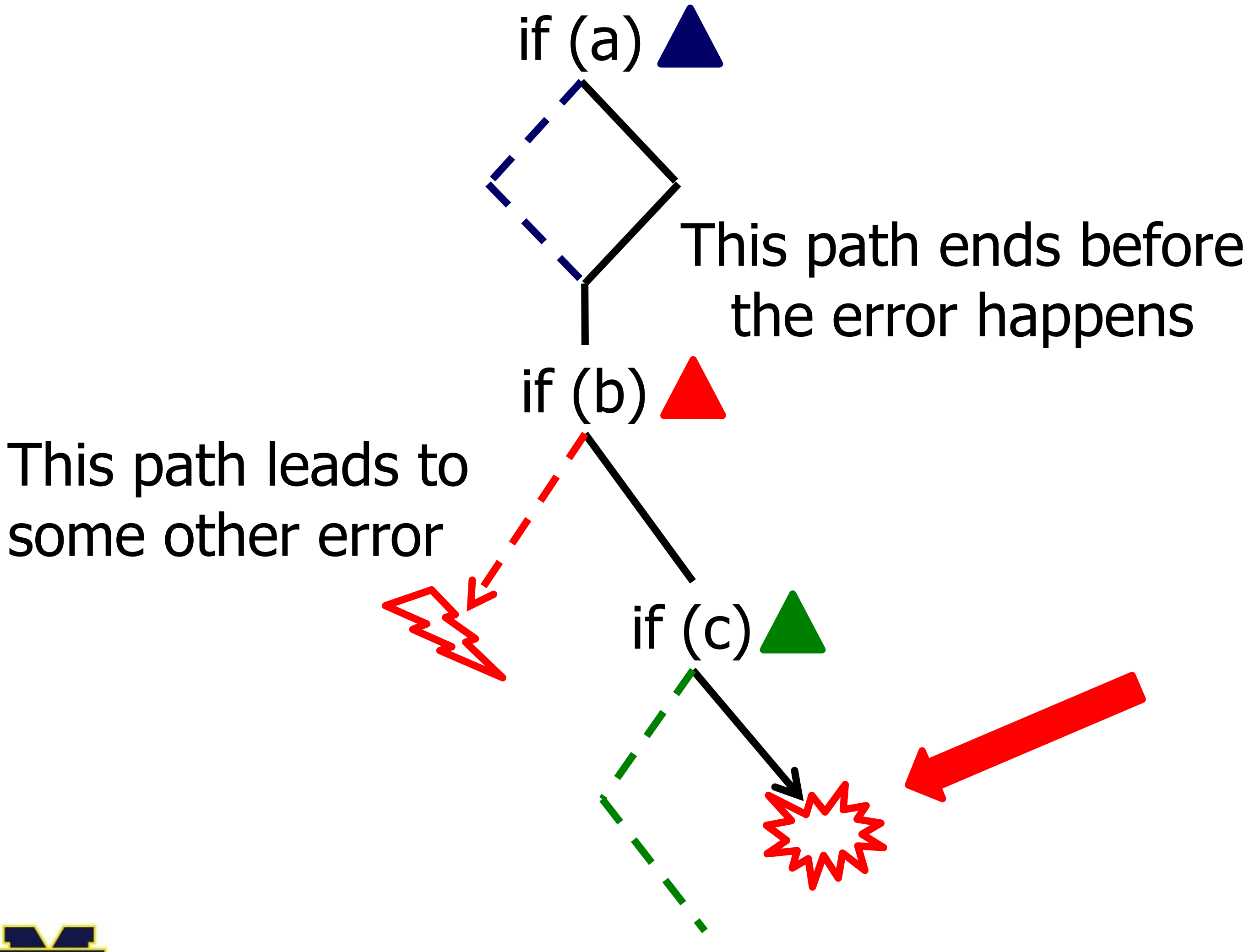
How to Avoid Error?



How to Avoid Error?

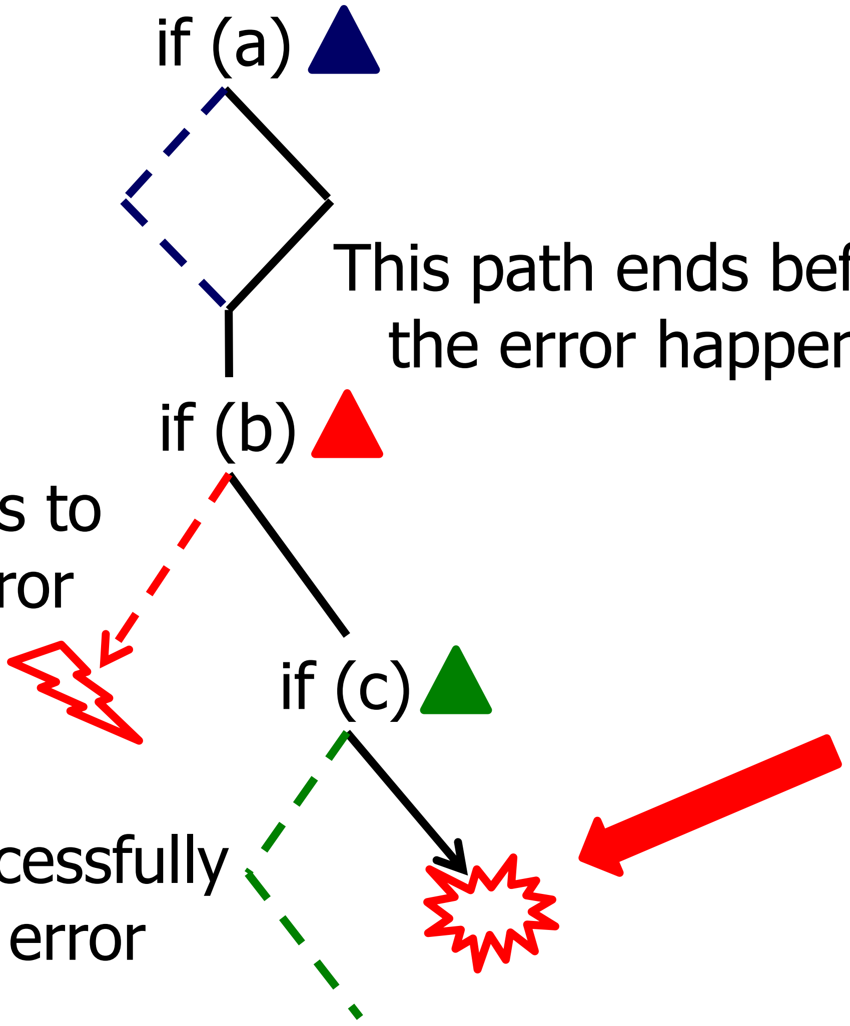


How to Avoid Error?



if (a)
This path ends before the error happens
if (b)
This path leads to some other error
if (c)

How to Avoid Error?

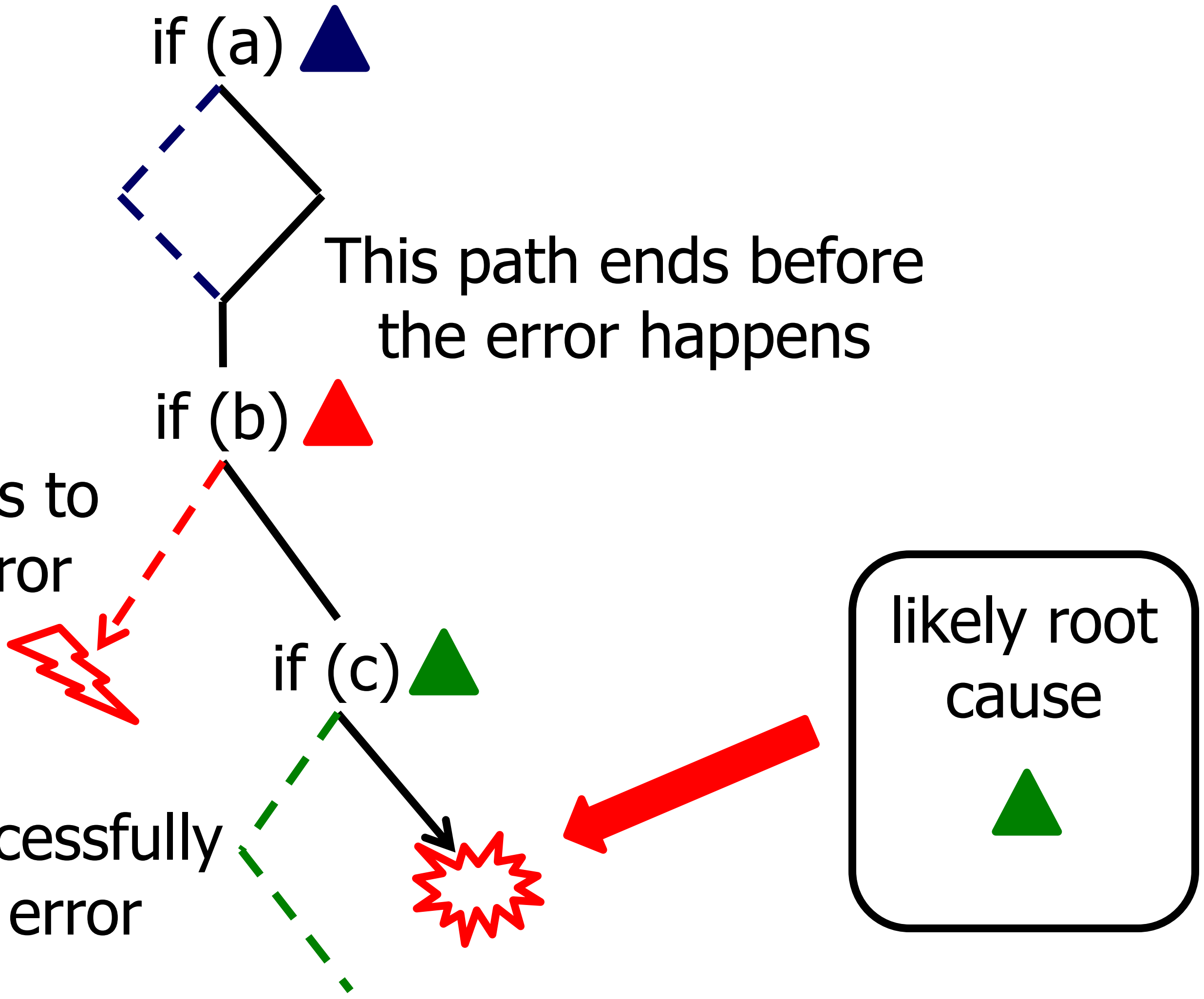


This path ends before the error happens

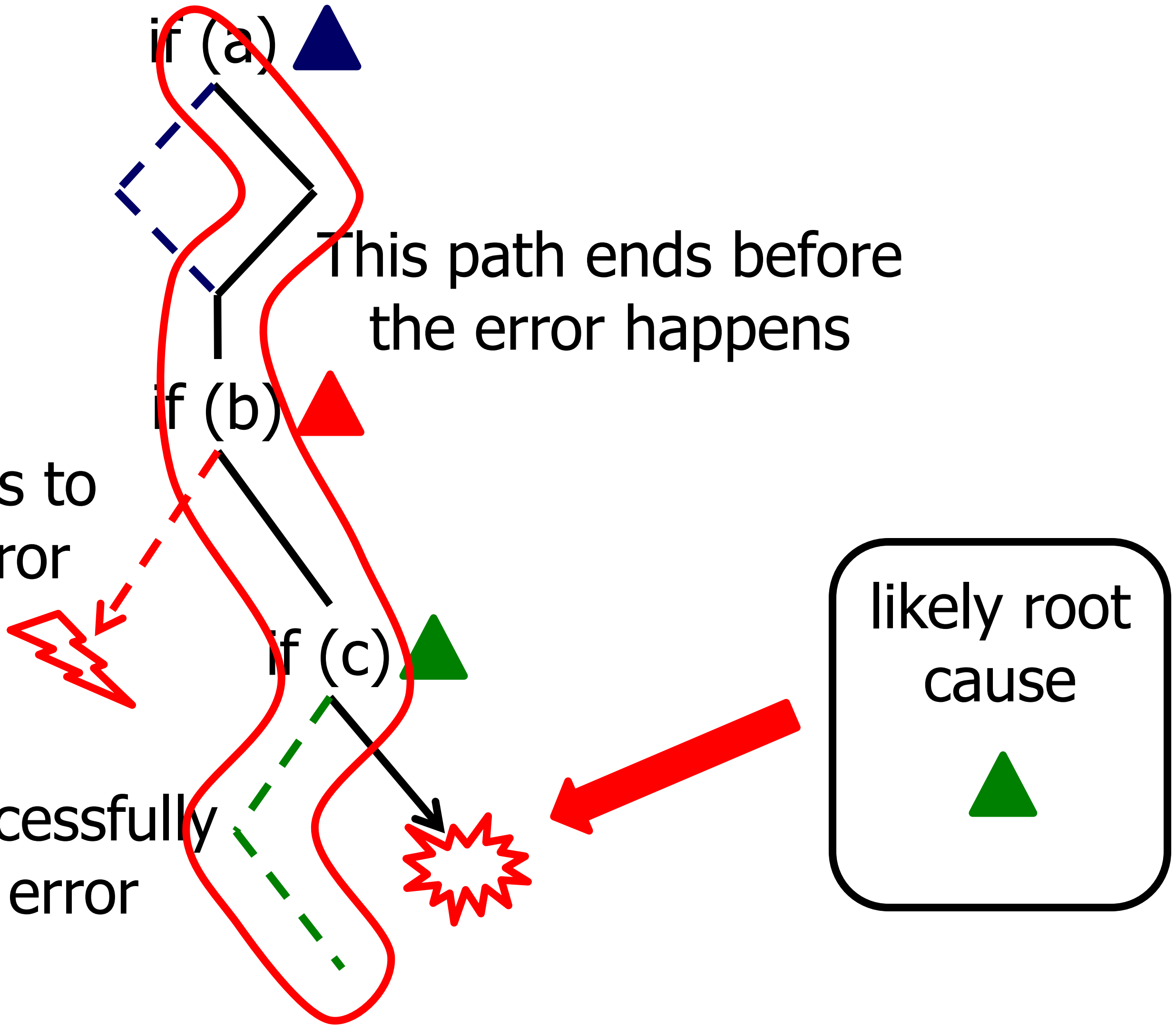
This path leads to some other error

This path successfully avoids the error

How to Avoid Error?


if (a)
This path ends before the error happens
if (b)
This path leads to some other error
if (c)
likely root cause
This path successfully avoids the error

How to Avoid Error?



if (a)
This path ends before the error happens
if (b)
This path leads to some other error
if (c)
likely root cause
This path successfully avoids the error

Outline

- *Motivation*
- *How ConfAid runs*
- Information flow analysis algorithms
- *Embracing imprecise analysis*
- *Evaluation*
- *Conclusion*

Data Flow Analysis

$T_x = \{{\color{red}\blacktriangle}, {\color{navy}\blacktriangle}\} \iff$ value of x might change, if tokens ${\color{red}\blacktriangle}$ or ${\color{navy}\blacktriangle}$ change

Taint propagates via **data flow** and **control flow**

$$x = y + z, \quad \begin{array}{l} T_y = \{{\color{red}\blacktriangle}, {\color{navy}\blacktriangle}\} \\ T_z = \{{\color{green}\blacktriangle}, {\color{navy}\blacktriangle}\} \end{array} \Longrightarrow T_x = \underbrace{\{{\color{red}\blacktriangle}, {\color{green}\blacktriangle}, {\color{navy}\blacktriangle}\}}_{T_y \cup T_z}$$

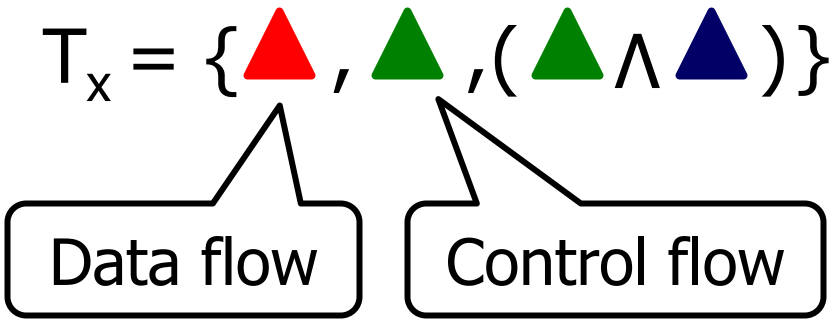
Control Flow Analysis

T_c = {▲} T_a = {▲}

T_x = {▲}

```
/* c = 0 */
/* x is read from file*/

if (c == 0) {
    x = a
}
```

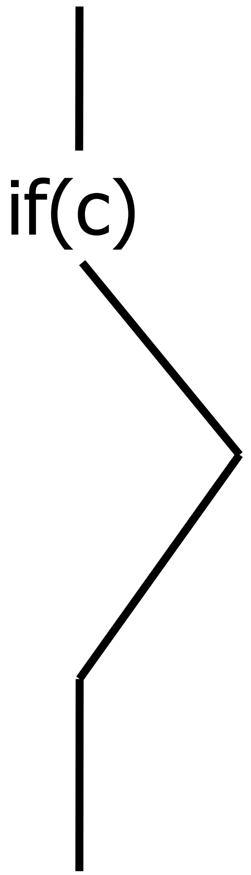
➡ What could cause x to be different?



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    ...
} else {
    y = a
}
```

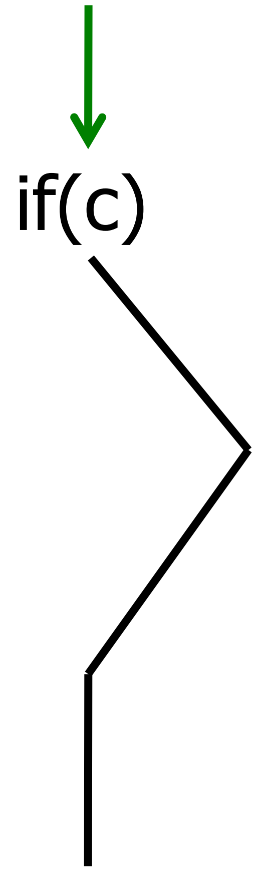
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    …
} else {
    y = a
}
```

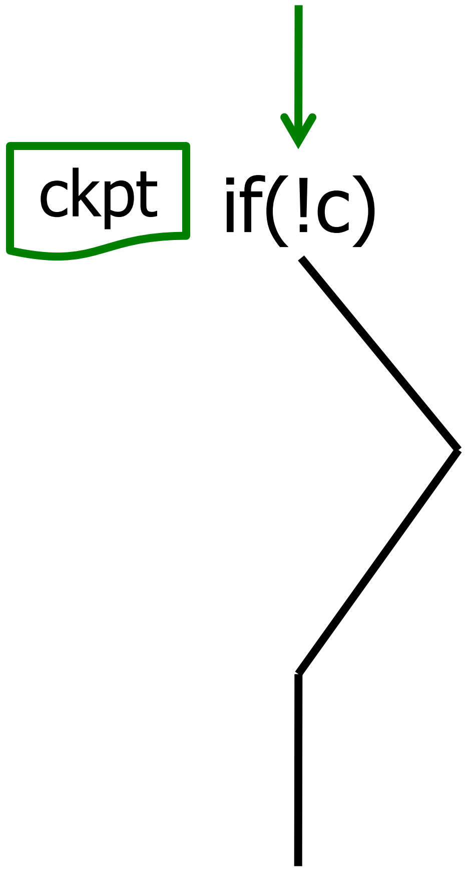
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    …
} else {
    y = a
}
```

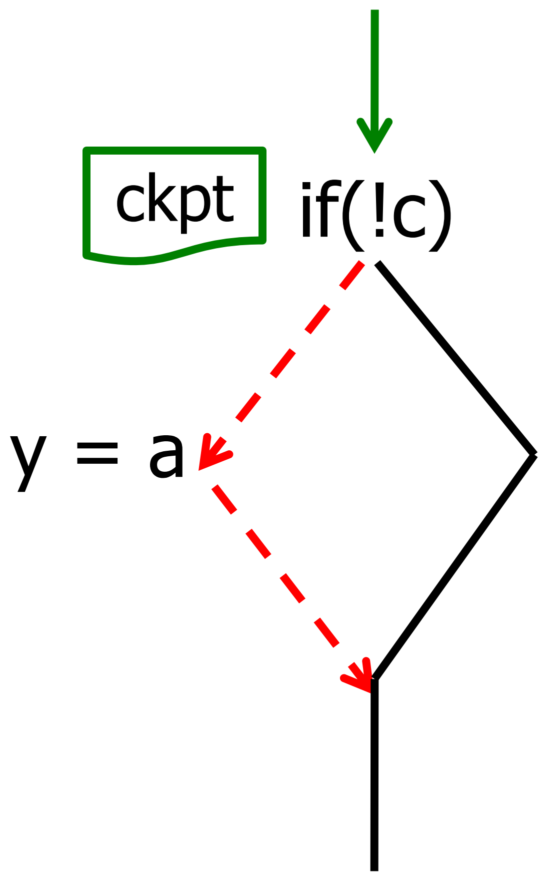
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    …
} else {
    y = a
}
```

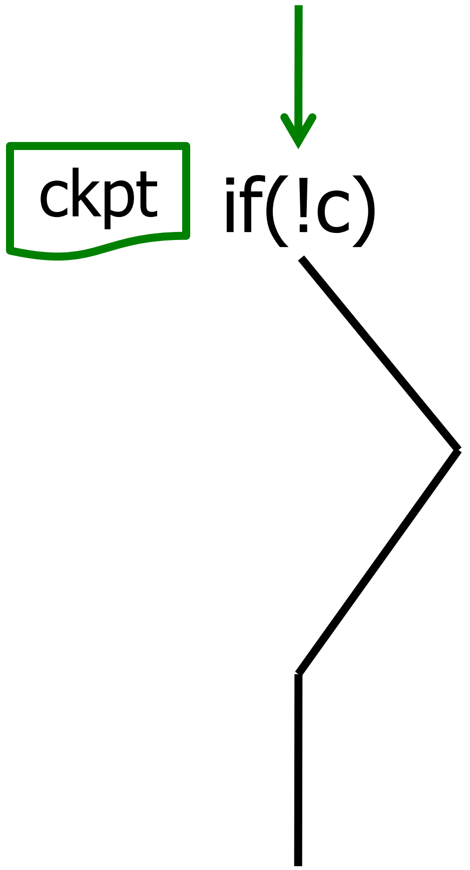
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    ...
} else {
    y = a
}
```

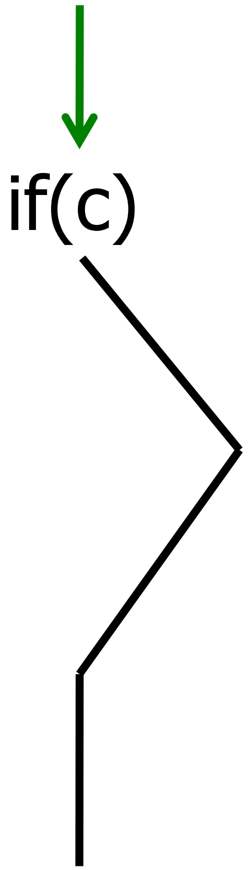
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    …
} else {
    y = a
}
```

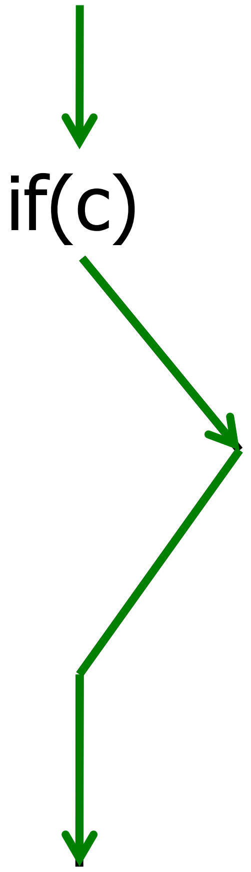
➡ y depends on c



Alternate Path Exploration

```
/* c = 1*/
/* y is read from file*/

if (c) {
    /*taken path*/
    …
} else {
    y = a
}
```

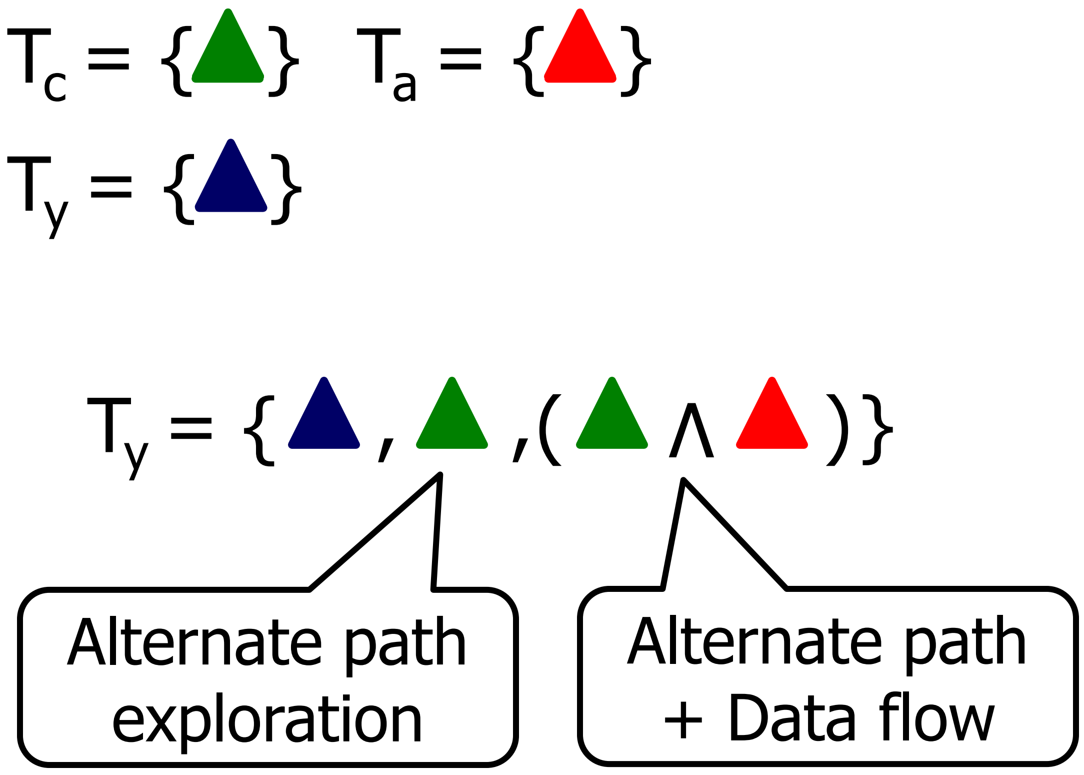
➡ y depends on c



Effect of Alternate Path Exploration

```
/* c = 1*/
/* y is from file*/

if (c) {
    …
} else {
    y = a
}
```

➡ What could cause y to be different?



Outline

- *Motivation*
- *How ConfAid runs*
- *Information flow analysis algorithms*
- Embracing imprecise analysis
- *Evaluation*
- *Conclusion*

Embracing Imprecise Analysis

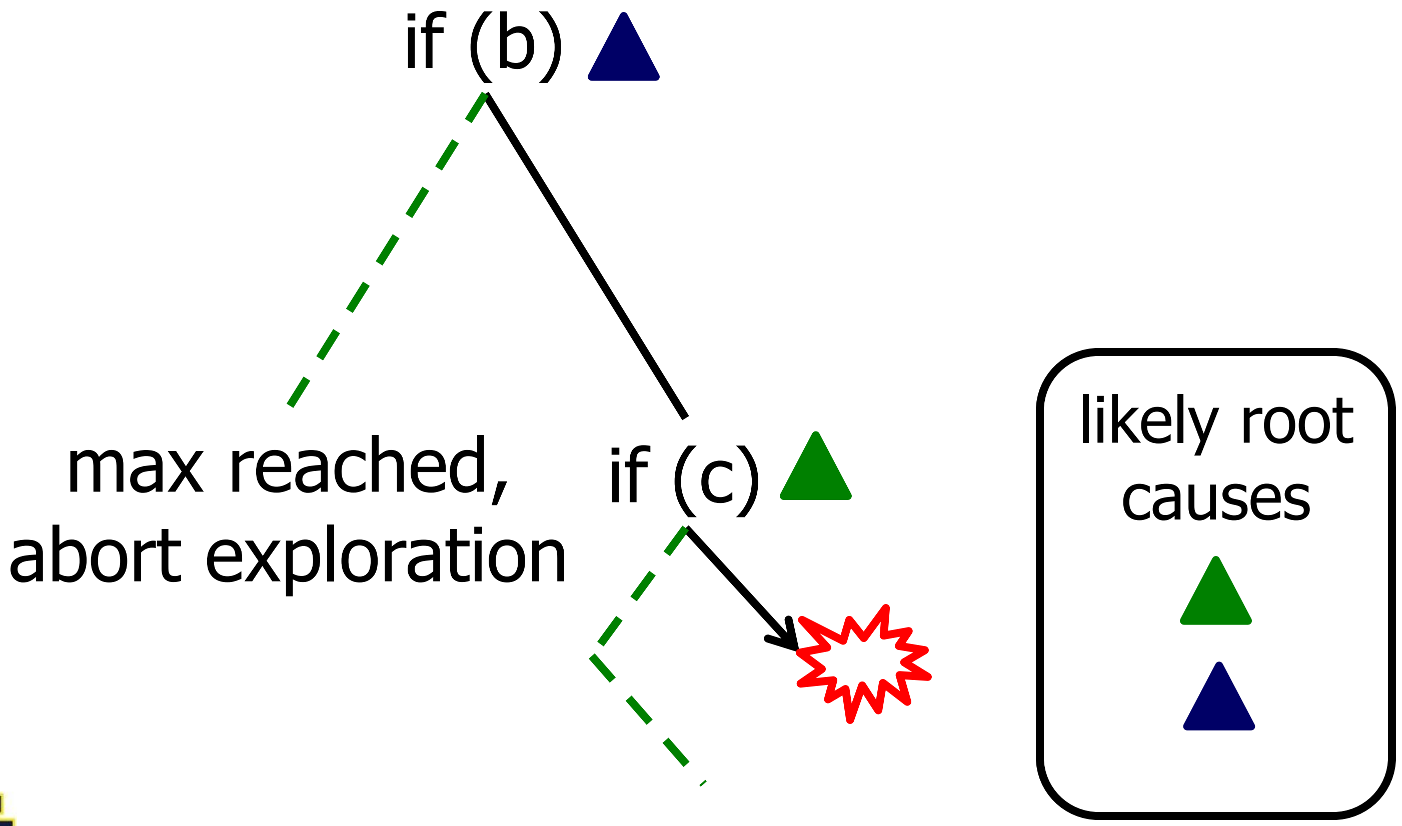
- Complete and sound analysis leads to:
  - poor performance
  - high false positive rate

- To improve performance $\nearrow$ Bounded horizon heuristic, $\searrow$ Single mistake heuristic

- To reduce false positives $\longrightarrow$ Weighting heuristic

Bounded Horizon Heuristic

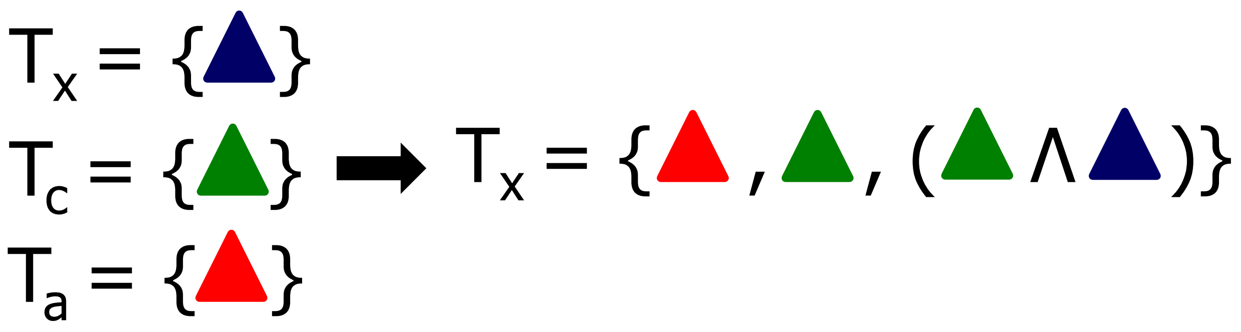
- Bounded horizon prevents path explosion
- Alternate path runs a fixed # of instructions



Single Mistake Heuristic

- Configuration file contains a single mistake
- Reduces amount of taint and # of explored paths

```
/* x=1, c=0*/

if (c == 0) {
    x = a
}
```

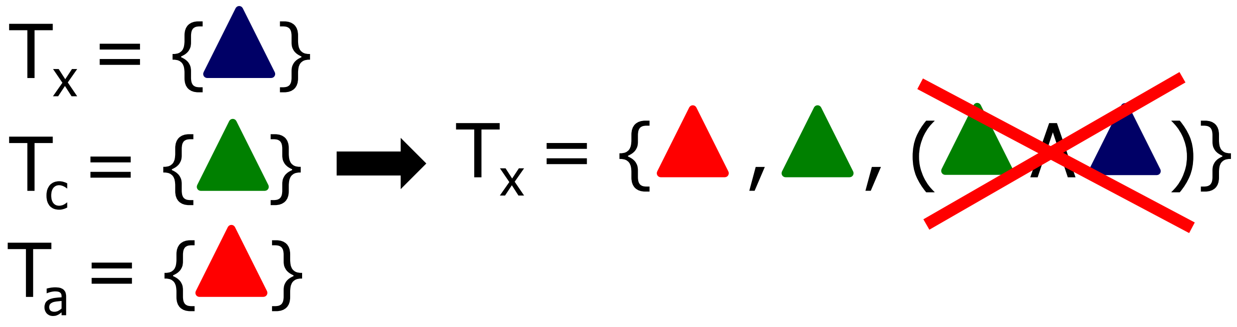


$T_x = \{\blacktriangle\}$
$T_c = \{\blacktriangle\}$
$T_a = \{\blacktriangle\}$
$T_x = \{\blacktriangle, \blacktriangle, (\blacktriangle \wedge \blacktriangle)\}$

Single Mistake Heuristic

- Configuration file contains a single mistake
- Reduces amount of taint and # of explored paths

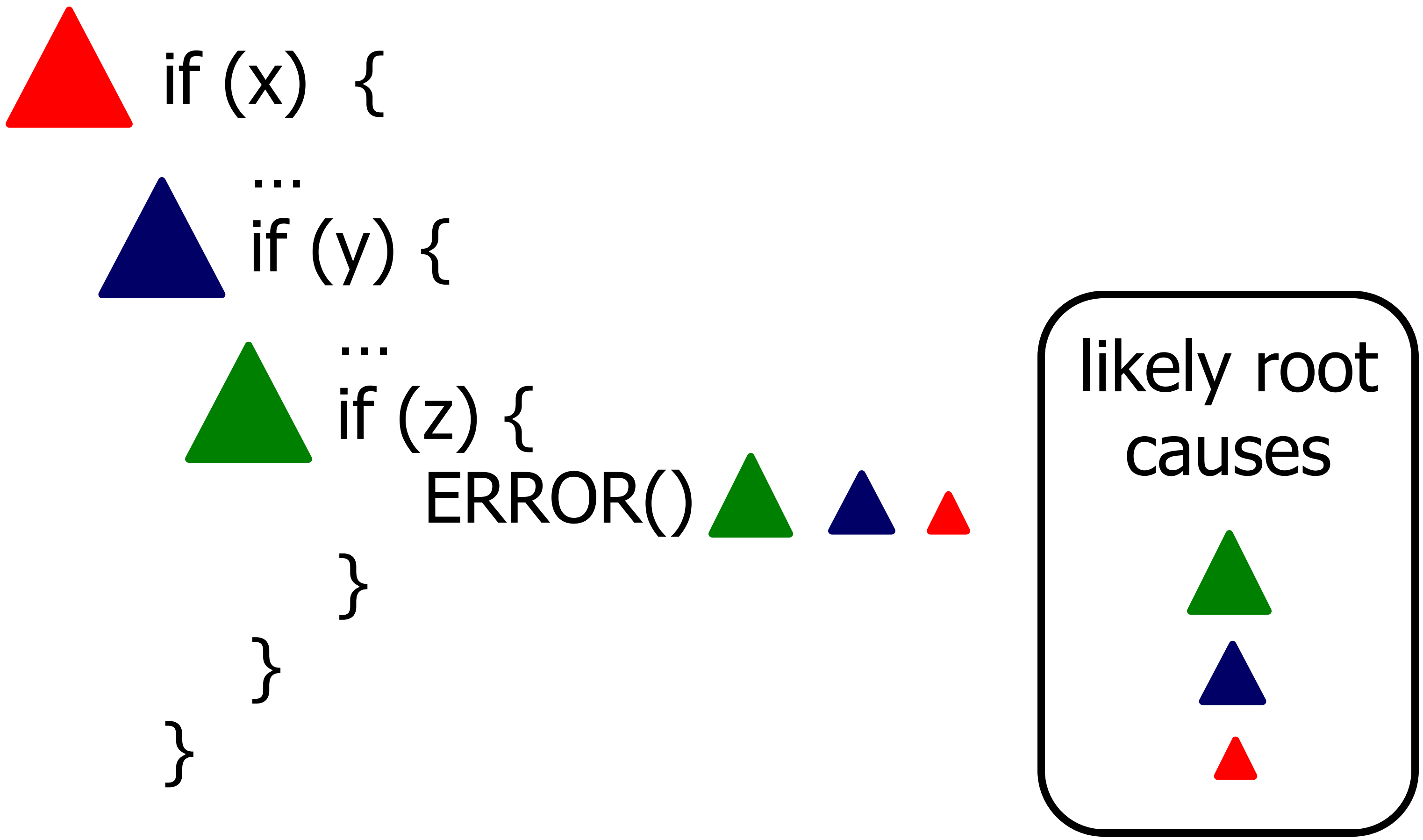
```
/* x=1, c=0*/

if (c == 0) {
    x = a
}
```



Weighting Heuristic

- Insufficient to treat all taint propagations equally
  - Data flow introduces stronger dependency than ctrl flow
  - Branches closer to error stronger than farther branches
- Assign weights to taints to represent strength level
  - Data flow taint gets a higher weight than ctrl flow taint
  - Branches closer to error get higher weight than farther

Example of Weighting Heuristic



```
if (x)  {
    …
    if (y) {
        …
        if (z) {
            ERROR()
        }
    }
}
```

likely root causes

Heuristics: Pros and Cons

	Bounded horizon	Single mistake	Weighting
Simplify control flow analysis		✓	
Improve performance	✓	✓	
Reduce FP		✓	✓
Increase FP	✗		
Increase FN	✗	✗	✗

FP = False Positive, FN = False Negative

ConfAid and Multi-process Apps

- ConfAid propagates taints between processes
  - Intercepts IPC system calls
  - Sends taint along with the data

- ConfAid currently supports communication via:
  - Unix sockets, pipes, TCP and UDP sockets
  - Regular files

Outline

Evaluation

- ConfAid debugs misconfiguration in:
  - OpenSSH 5.1 (2 processes)
  - Apache HTTP server 2.2.14 (1 process)
  - Postfix mail transfer agent 2.7 (up to 6 processes)

- Manually inject errors to configuration files
- Evaluation metrics:
  - The ranking of the correct root cause
  - The time to execute the application with ConfAid

Data Sets

- Real-world misconfigurations:
  - total of 18 bugs from manuals, forums and FAQs
- Randomly generated bugs:
  - 60 bugs using ConfErr [Keller et al. DSN 08]

How Effective is ConfAid ?

Correct root caused ranked first or second for all 18 real-world bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47-49	2	2	2	1	0
Apache	88-93	3	1	0	2	0
Postfix	27-29	5	5	0	0	0

How Effective is ConfAid ?

Correct root caused ranked first or second for all 18 real-world bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47-49	2	2	2	1	0
Apache	88-93	3	1	0	2	0
Postfix	27-29	5	5	0	0	0

72%

How Effective is ConfAid ?

Correct root caused ranked first or second for all 18 real-world bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47-49	2	2	2	1	0
Apache	88-93	3	1	0	2	0
Postfix	27-29	5	5	0	0	0

72% 28%

How Effective is ConfAid ?

Correct root caused ranked first or second for all 18 real-world bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47-49	2	2	2	1	0
Apache	88-93	3	1	0	2	0
Postfix	27-29	5	5	0	0	0

72% 28% 0%

How Effective is ConfAid ?

Correct root caused ranked first or second for 55 out of 60 randomly-generated bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47	17	1	1	0	1
Apache	88	17	1	0	1	1
Postfix	27	15	0	2	0	3

How Effective is ConfAid ?

Correct root caused ranked first or second for 55 out of 60 randomly-generated bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47	17	1	1	0	1
Apache	88	17	1	0	1	1
Postfix	27	15	0	2	0	3

85%

How Effective is ConfAid ?

Correct root caused ranked first or second for 55 out of 60 randomly-generated bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47	17	1	1	0	1
Apache	88	17	1	0	1	1
Postfix	27	15	0	2	0	3

85%

7%

How Effective is ConfAid ?

Correct root caused ranked first or second for 55 out of 60 randomly-generated bugs

	Total tokens	First	First tied w/1	Second	Second tied w/1	Worse than second
OpenSSH	47	17	1	1	0	1
Apache	88	17	1	0	1	1
Postfix	27	15	0	2	0	3

85% 7% 8%

How Fast is ConfAid?

Average execution time for real-world bugs: 1m 32s

	Average Execution Time
OpenSSH	52 seconds
Apache	2 minutes 48 seconds
Postfix	57 seconds

Average time for randomly-generated bugs: 23s

OpenSSH	7 seconds
Apache	24 seconds
Postfix	38 seconds

Conclusion

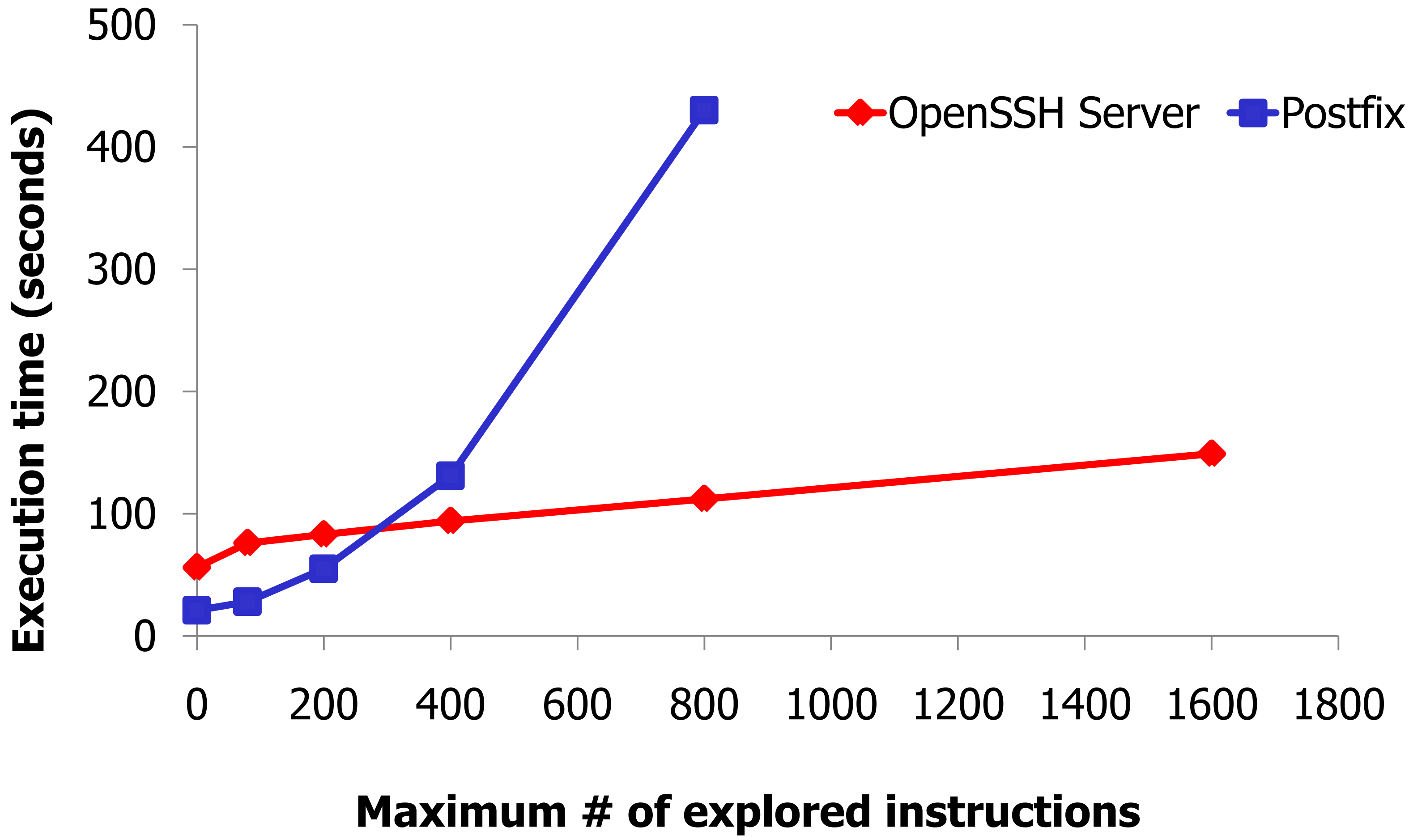
- ConfAid automatically finds root cause of problems
- ConfAid uses dynamic information flow analysis
- ConfAid ranks the correct root cause as first or second in:
  - 18 out of 18 real-world bugs
  - 55 out of 60 random bugs
- ConfAid takes only a few minutes to run

Questions?

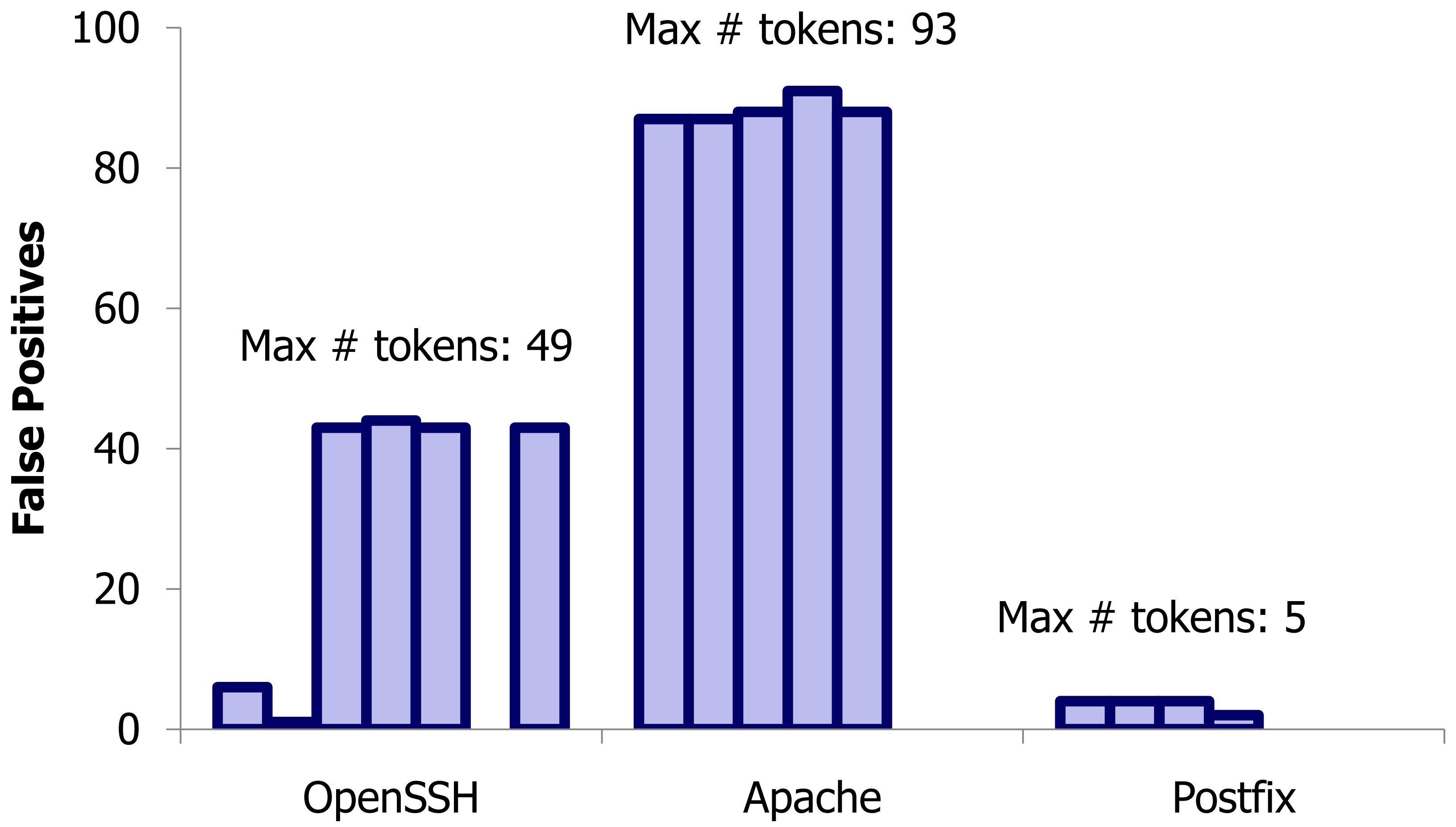
What if there are multiple mistakes?

- ConAid may or may not report all
- For independent mistakes, ConfAid first finds the one that led to the first failure
- For dependent mistakes, ConfAid may report all based on their effect on program

Effect of Bounded Horizon Heuristic



Effect of Weighting Heuristic


False Positives
Max # tokens: 49
Max # tokens: 93
Max # tokens: 5
OpenSSH
Apache
Postfix