

# Automated Discovery of Mimicry Attacks

Jonathon T. Giffin

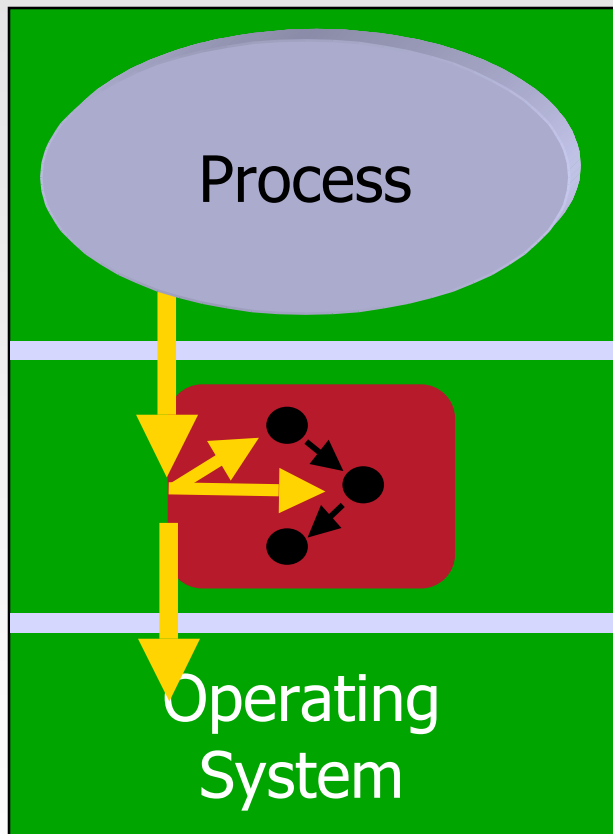
Georgia Institute of Technology  
giffin@cc.gatech.edu

Somesh Jha    Barton P. Miller

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

Presenting research done at the University of Wisconsin

# Model-Based Anomaly Detection



- Detect deviations from normal system call execution behavior
- Models of allowed system call sequences

[Crosbie & Spafford 1995]  
[Forrest *et al.* 1996—1999]  
[Lee *et al.* 1998—2004]  
[Warrander *et al.* 1999]  
[Ye 2000]  
[Wespi, Dacier, & Debar 2000]  
[Jha, Tan, & Maxion 2001]  
[Sekar *et al.* 2001]  
[Wagner & Dean 2001]  
[Dasgupta & Gonzalez 2002]  
[Giffin *et al.* 2002—2006]  
[Feng *et al.* 2003—2004]  
[Gao *et al.* 2004—2005]  
[Gopalakrishna *et al.* 2005]  
and more...



# Model-Based Anomaly Detection

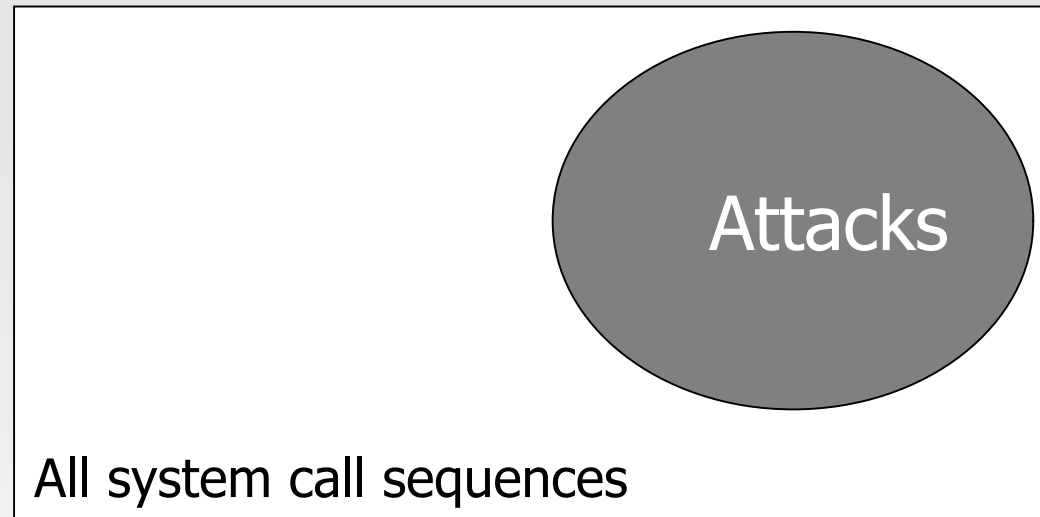
---

So we're safe... right?



# Model-Based Anomaly Detection

---



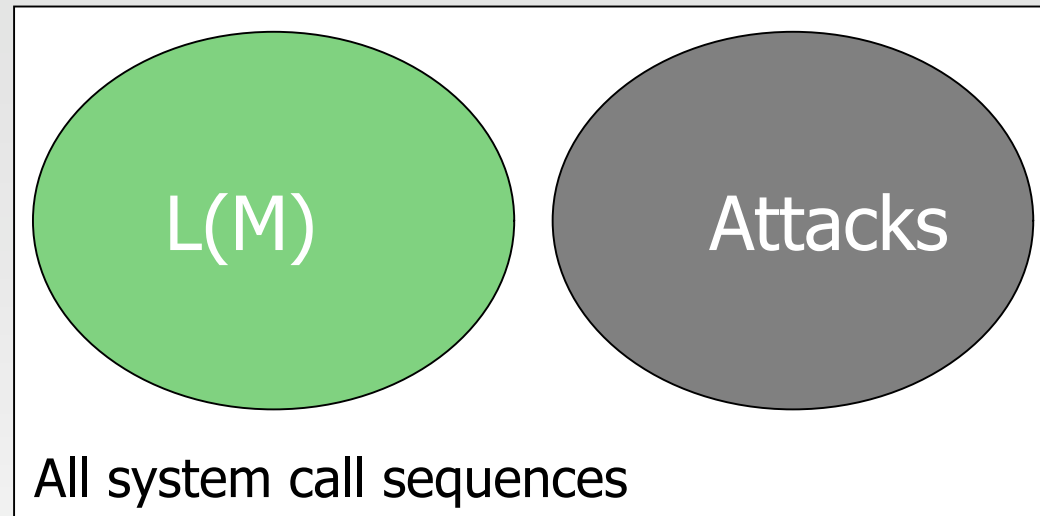
## Attack:

System call sequence  
that maliciously alters  
operating system



# Model-Based Anomaly Detection

---



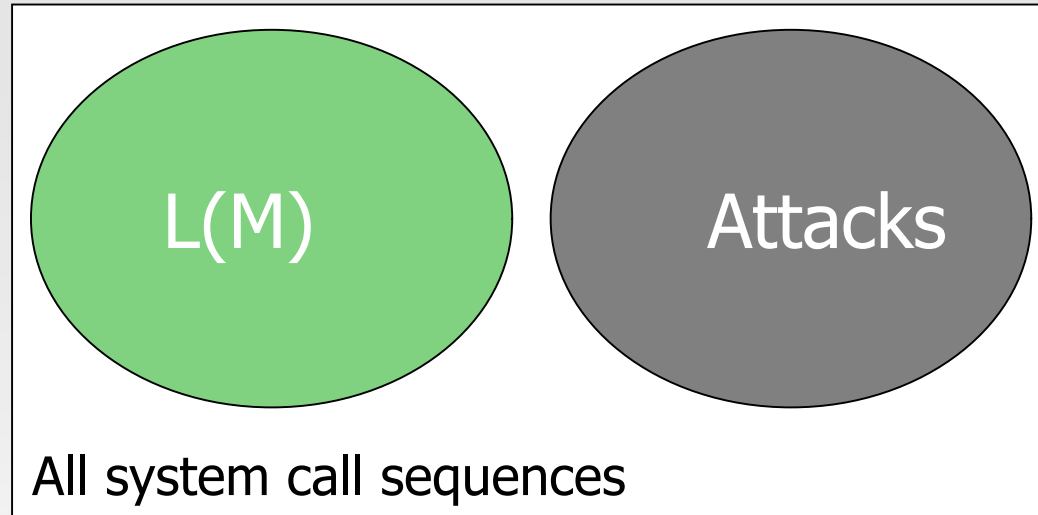
## IDS Ability:

Detect anomalous sequences  
of system calls



# Model-Based Anomaly Detection

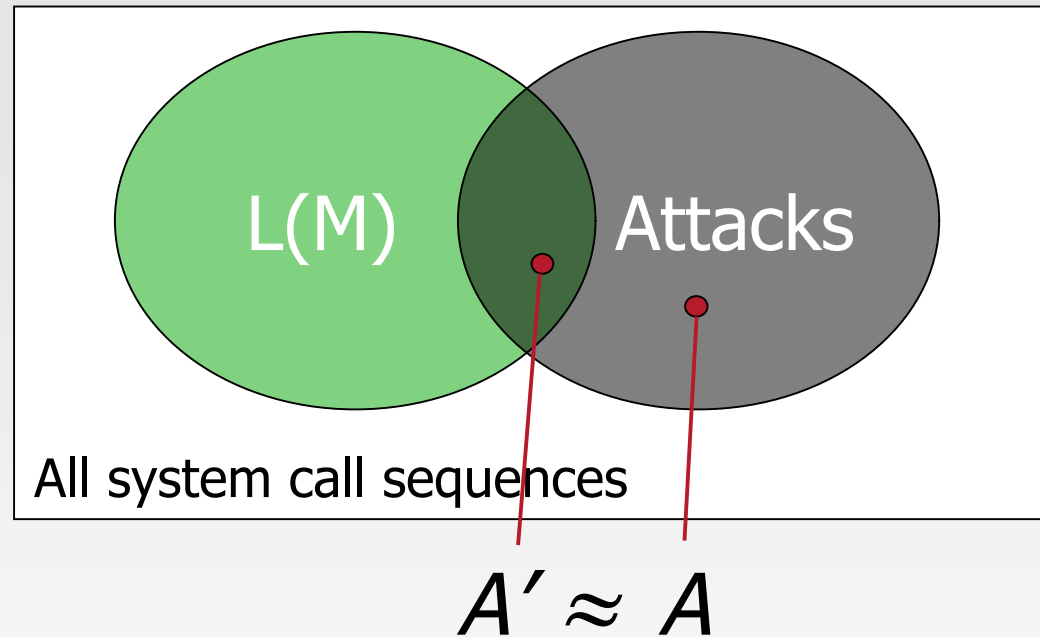
---



**IDS Goal:**  
Detect attacks



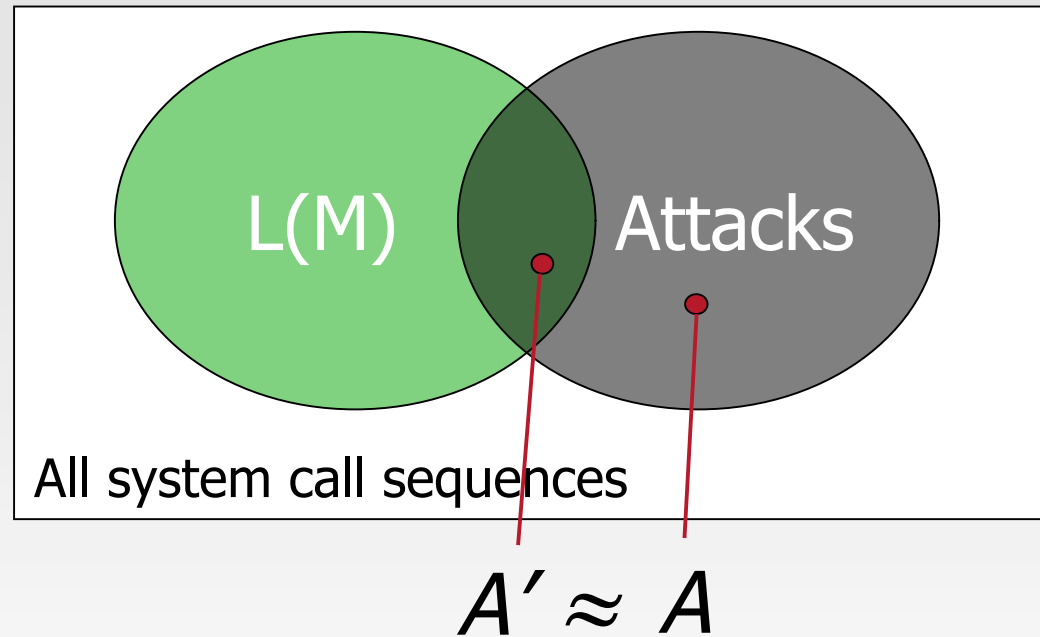
# Mimicry Attacks



[Tan, McHugh, Killourhy, & Maxion 2002]  
[Wagner & Soto 2002]



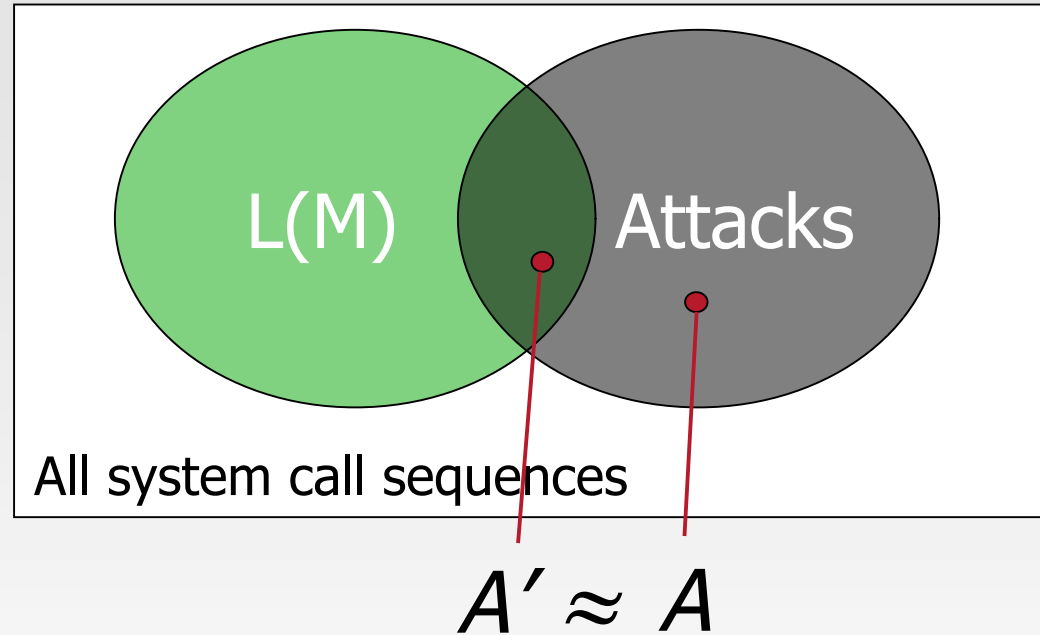
# Contributions



Automated discovery of undetected attacks  $A'$   
with no prior knowledge of *obfuscations*



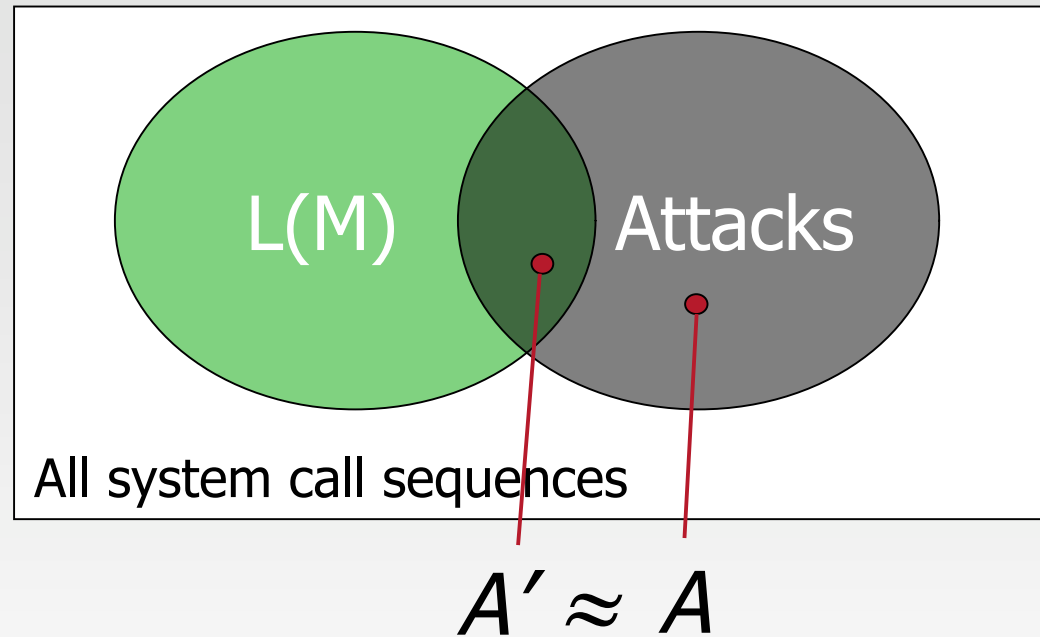
# Contributions



Automated discovery of undetected attacks  $A'$   
with no prior knowledge of  $A$



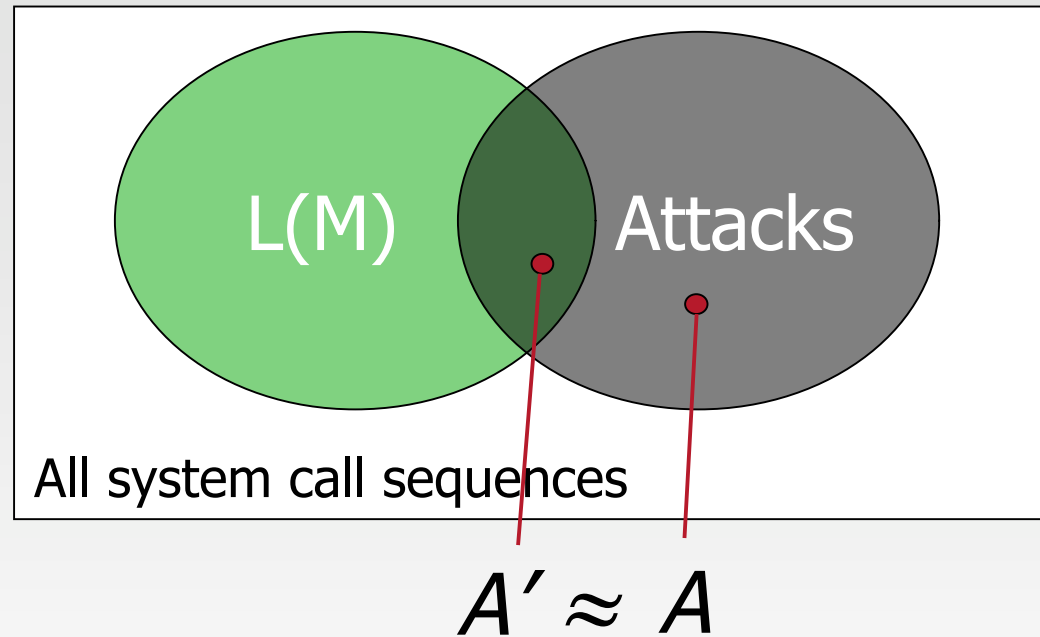
# Contributions



Automated discovery of undetected attacks  $A'$   
with no prior knowledge of *Attacks*



# Implications



Connect **attack discovery** with **attack construction**  
for complete & automatic evasion of detectors

[Kruegel *et al.* 2005]



# Automatic Attack Discovery

---

Attacks defined formally by their **effect**,  
not by **how** the effect is produced.

- Benefits:
  - Not limited to known attacks
  - Equivalence of effect defined formally
  - Formal analysis finds attacks automatically



# Code Example

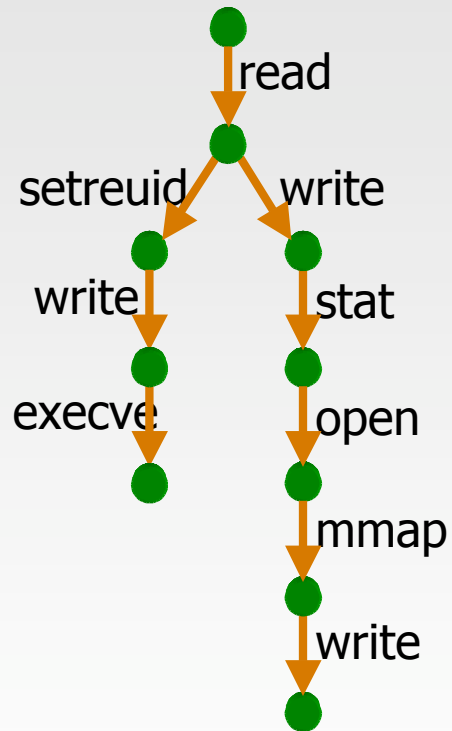
---

```
void main (void) {
    char input[32];
    gets(input);
    if (input[0] == 'x') {
        setreuid(42, -1);
        syslog(1, "Execing file");
        execve(input+2, 0, 0);
    } else if (input[0] == 'e') {
        struct stat buf;
        syslog(1, "Echoing file");
        stat(input+2, &buf);
        int fd = open(input+2, O_RDONLY);
        void *data = mmap(0, buf.st_size, 1, 2, fd, 0);
        write(1, data, buf.st_size);
    }
}
```



# Mimicry Attacks

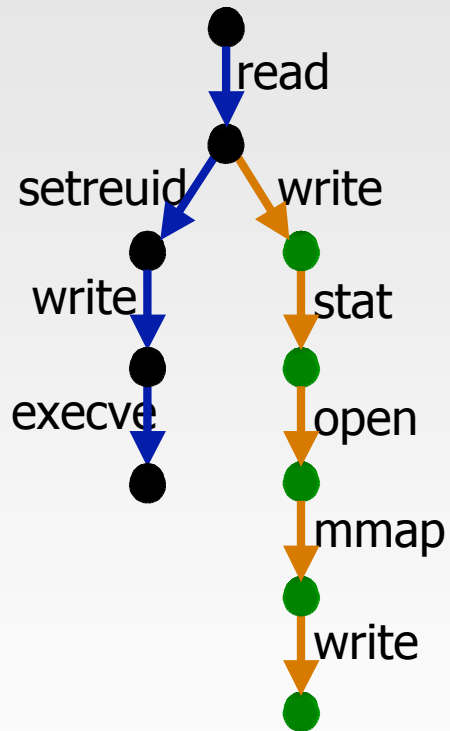
Detected A:  
setuid(0);  
execve("/bin/sh");



Stide



# Mimicry Attacks



Stide

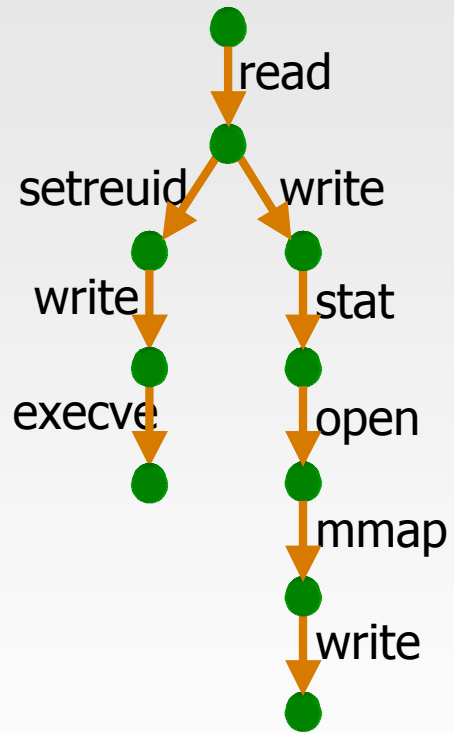
Detected A:

```
setuid(0);  
execve("/bin/sh");
```

Undetected A':

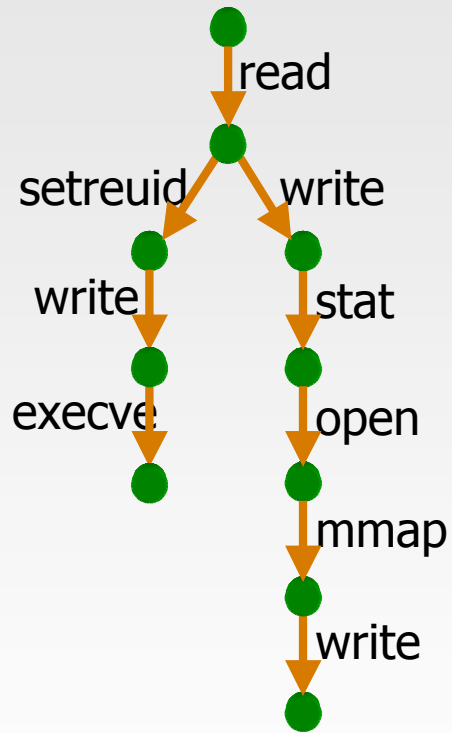
```
read(-1);  
setreuid(0, 0);  
write(-1);  
execve("/bin/sh");
```



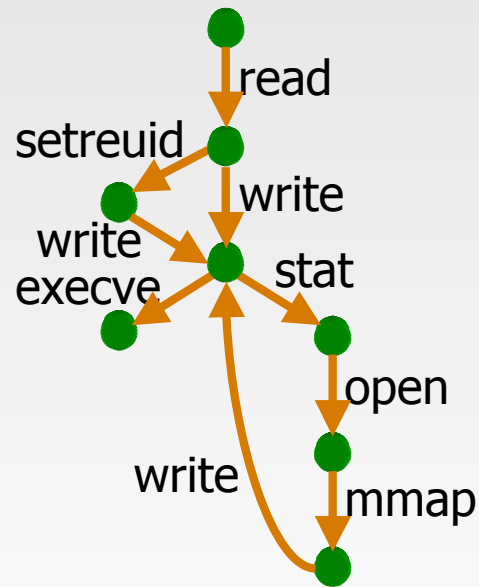


Stide



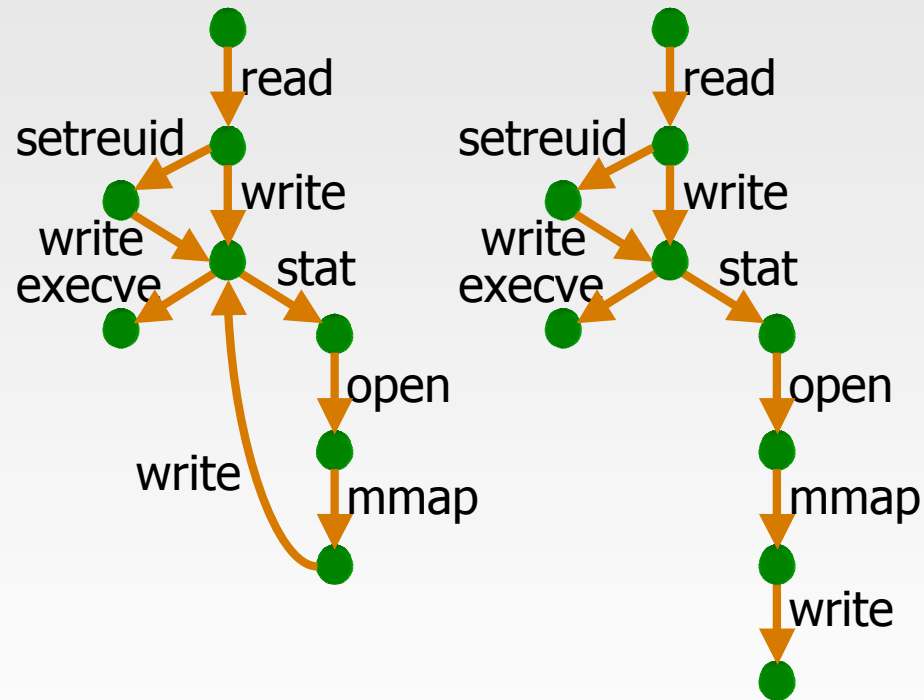


Stide



Digraph

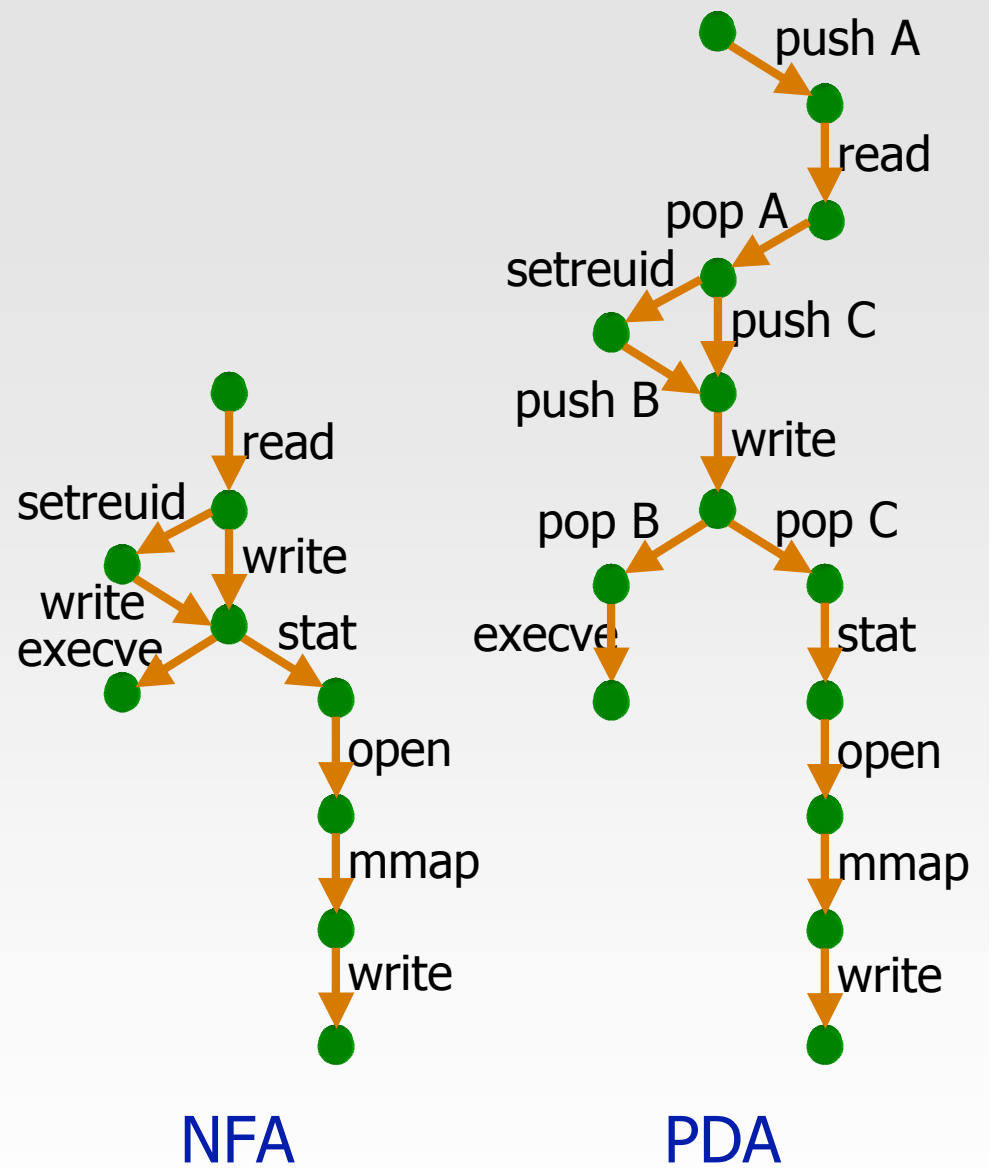




Digraph

NFA

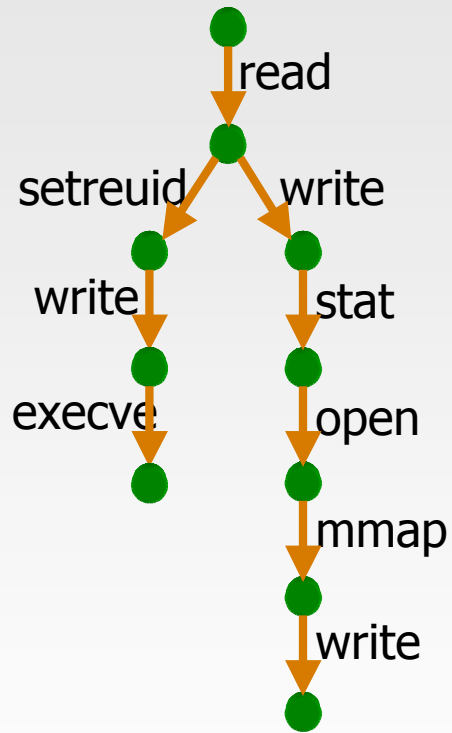




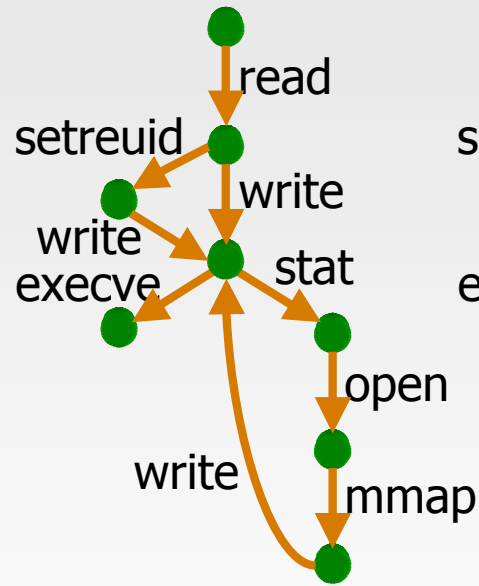
NFA

PDA

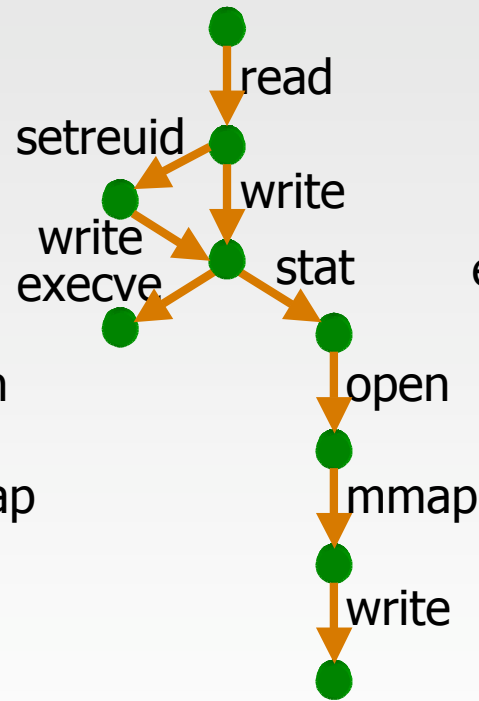




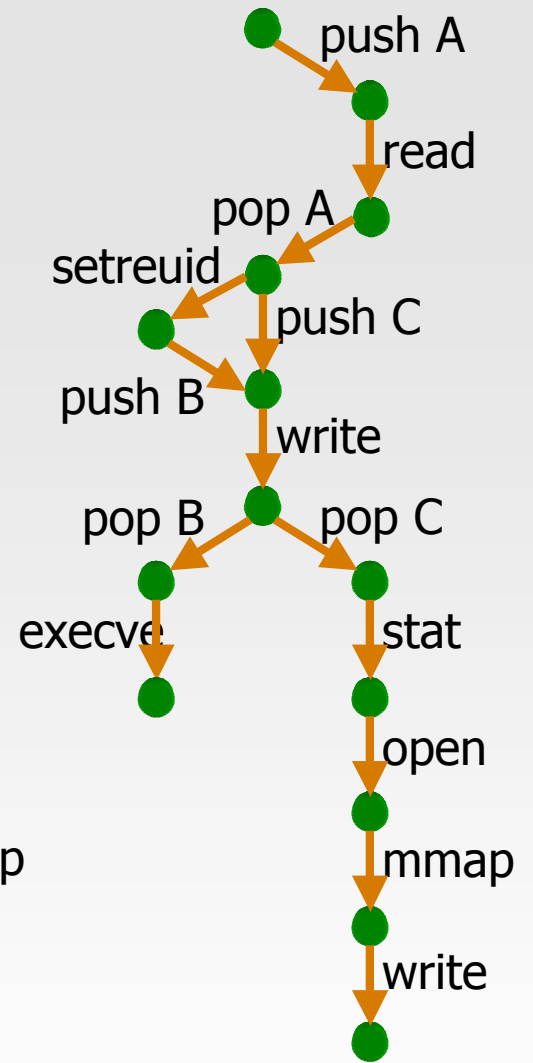
Stide



Digraph



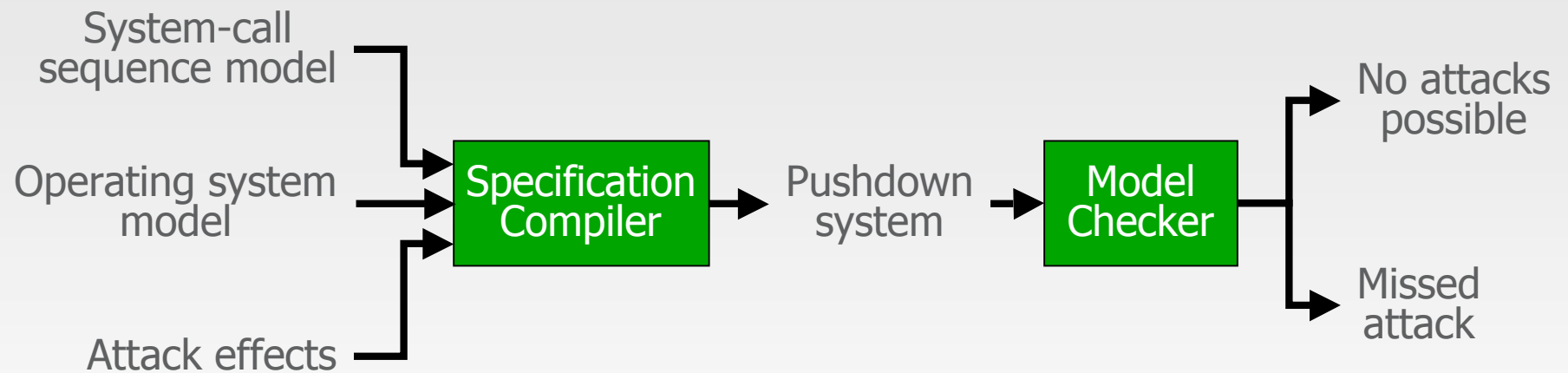
NFA



PDA



# Automatic Attack Discovery



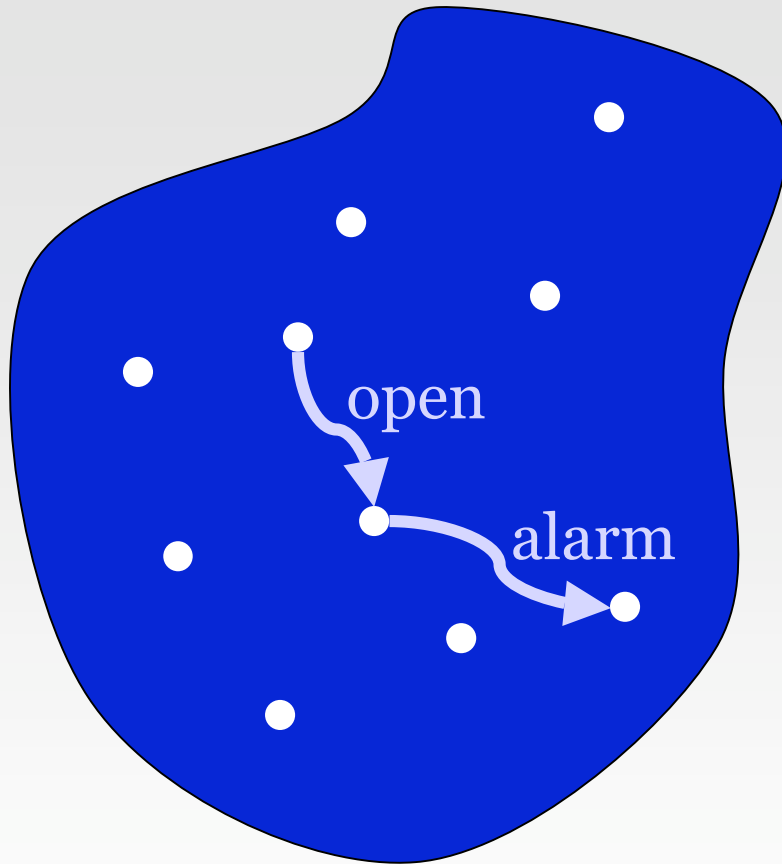
# Operating System Model

---

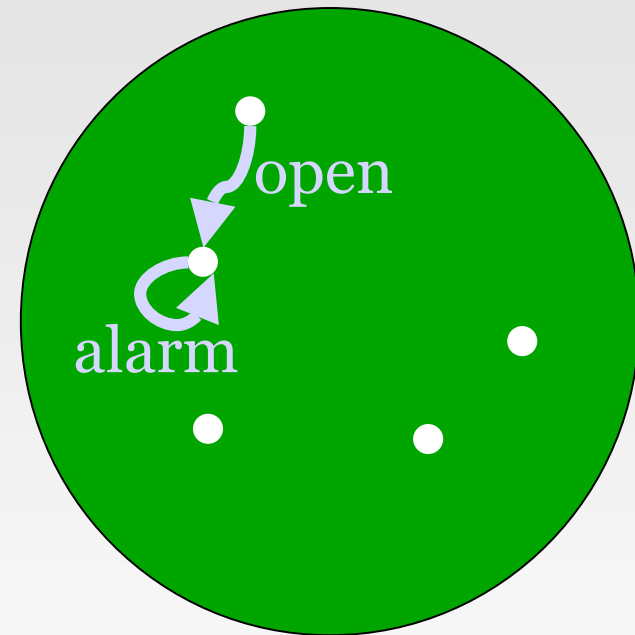
- Model finite operating system state
  - State variables  $V$
  - Initial assignments to state variables
  - System-call transition relations
  
- Example state variables
  - Process user ID
  - File rwx permissions bits
  - File pointed to by file descriptor
  - Executing image
  - File owner
  - File descriptor access



# Operating System Model



Concrete OS



Abstract Model



# Operating System Model

exec (int <i>fileIndex</i> )		
<p><b>Preconditions:</b> file[<i>fileIndex</i>].valid <math>\wedge ( ( \text{file}[\textit{fileIndex}].\textit{owner} = \textit{euid}</math>     <math>\wedge \text{file}[\textit{fileIndex}].\textit{owner\_x}</math>     <math>\vee ( \text{file}[\textit{fileIndex}].\textit{owner} \neq \textit{euid}</math>         <math>\wedge \text{file}[\textit{fileIndex}].\textit{group} = \textit{egid}</math>         <math>\wedge \text{file}[\textit{fileIndex}].\textit{group\_x}</math>     <math>\vee ( \text{file}[\textit{fileIndex}].\textit{owner} \neq \textit{euid}</math>         <math>\wedge \text{file}[\textit{fileIndex}].\textit{group} \neq \textit{egid}</math>         <math>\wedge \text{file}[\textit{fileIndex}].\textit{world\_x} ) )</math></p>	<p><b>Preconditions:</b> <math>\neg \text{file}[\textit{fileIndex}].\textit{valid}</math></p> <p><b>Postconditions:</b> no state change</p>	...
<p><b>Postconditions:</b> image' = <i>fileIndex</i></p>		



# Attack Effects

---

**Attack effects** are boolean formula  $b$  over OS state variables  $V$

– “Executed root shell”

$b$ : `image = /bin/sh  $\wedge$  euid = 0`

– “Added new user to system”

$b$ : `file[/etc/passwd].written = true`



# Automatic Attack Discovery

---

- **Goal:** Find  $A \in L(M)$  such that executing  $A$  produces unsafe OS state
- **Pushdown model checking**
  - Show  $M \models \Box \neg b$
  - Counterexample trace is undetected attack
  - Model-checking pushdown systems for LTL safety properties decidable in polynomial time

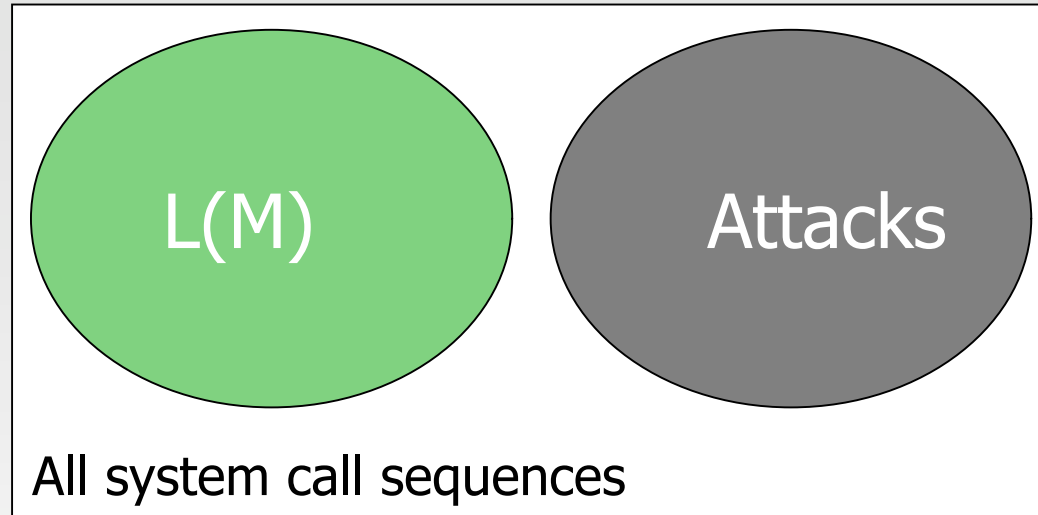
[Bouajjani, Esparza, & Maler 1997]

[Finkel, Willems, & Wolper 1997]

[Esparza & Podelski 2000]



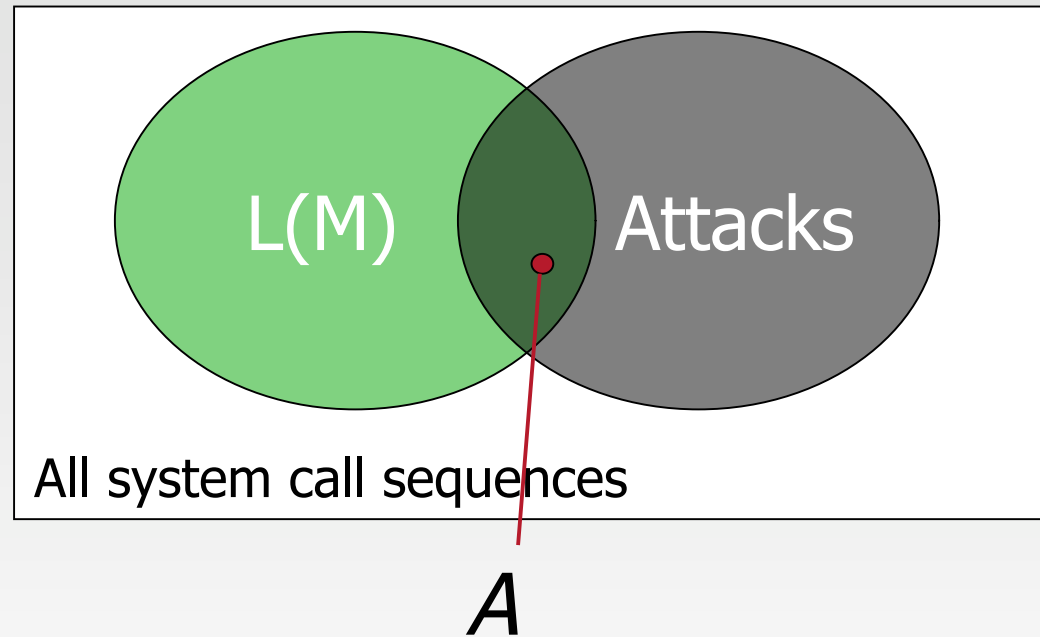
# Model-Based Anomaly Detection



$M \models \square \neg b$  holds  
No attack



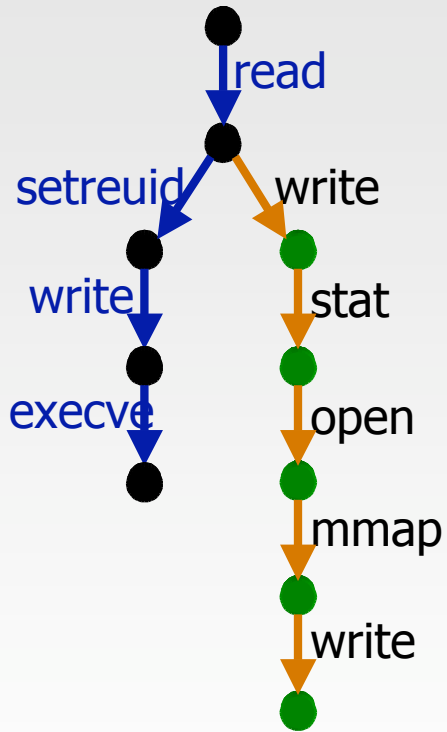
# Automatic Attack Discovery



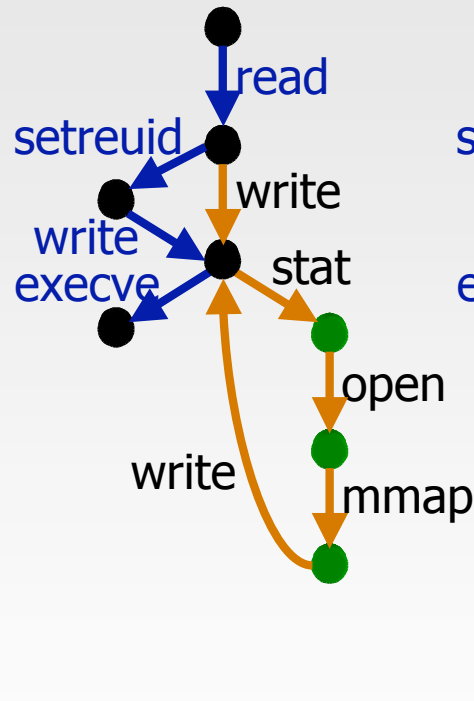
$M \models \square \neg b$  does not hold  
Attack sequence found



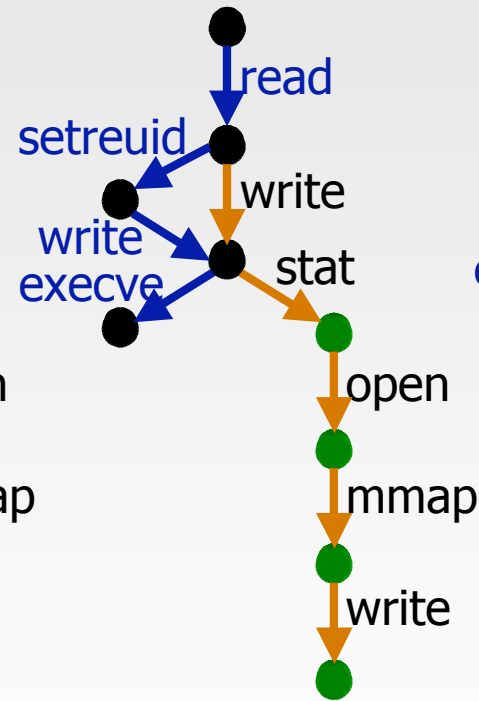
image = /bin/sh  $\wedge$  euid = 0



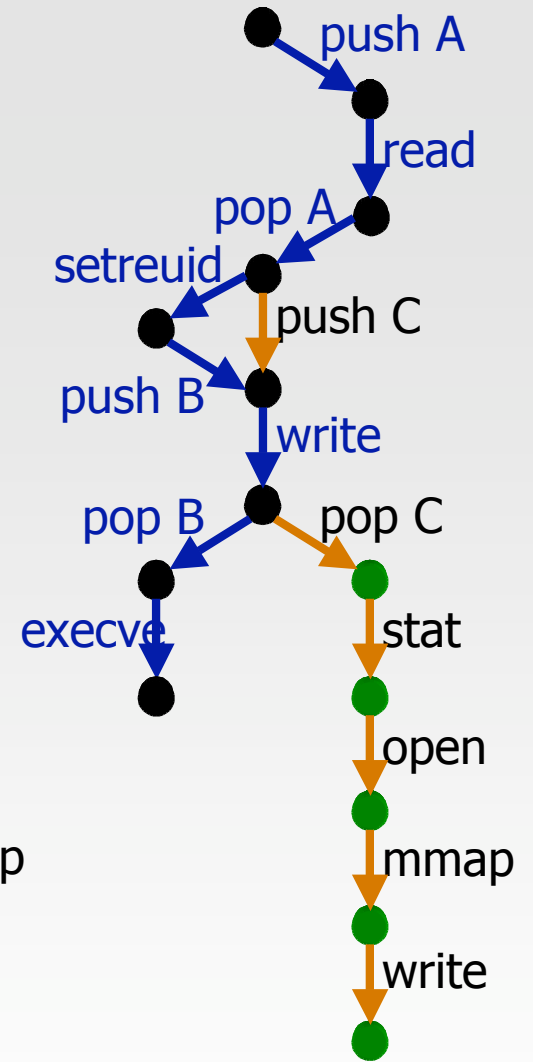
Stide



Digraph



NFA



PDA

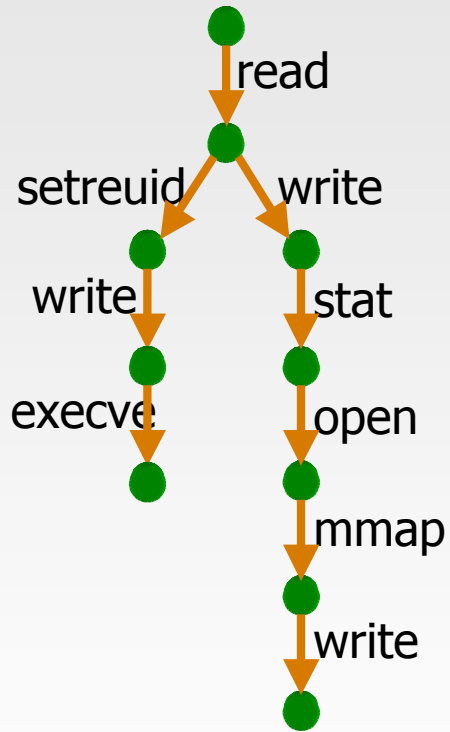


**image = /bin/sh ^ euid = 0**

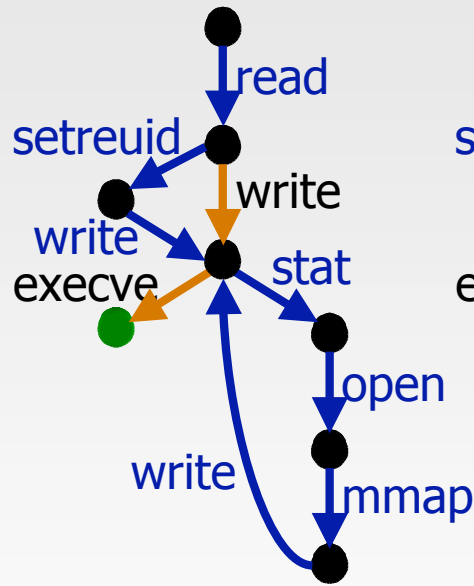
```
read(-1)  
setreuid(0, 0)  
write(-1)  
execve("/bin/sh")
```



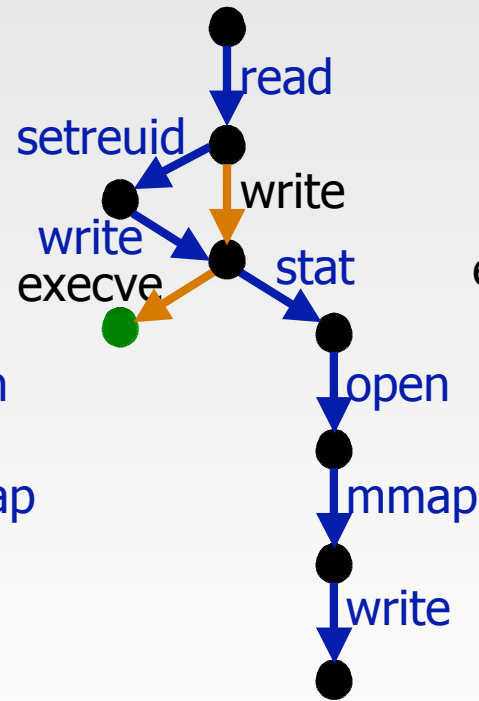
file[/etc/passwd].written = true



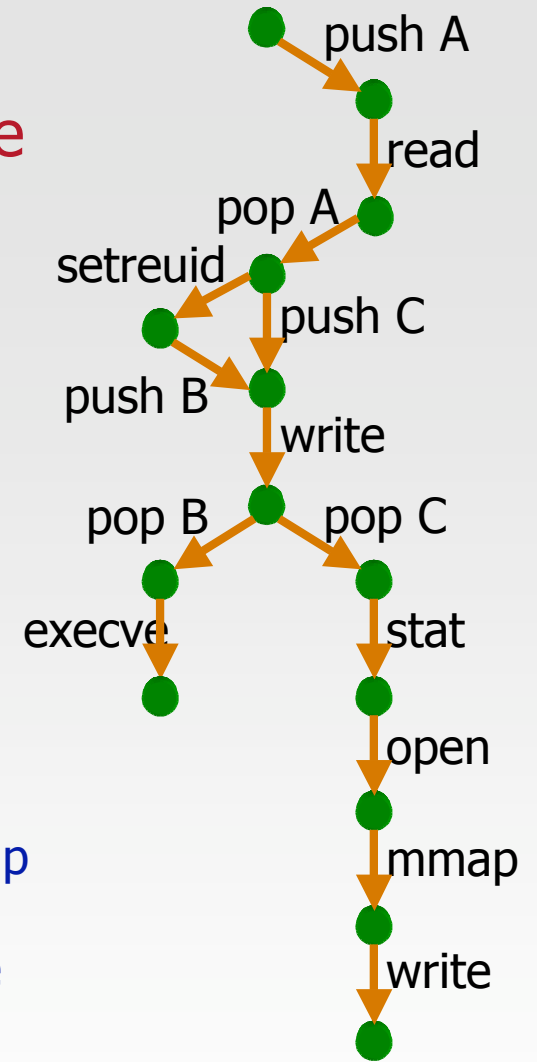
Stide



Digraph



NFA



PDA



`file[/etc/passwd].written = true`

`read(-1)`

`setreuid(0, 0)`

`write(-1)`

`stat(0, 0)`

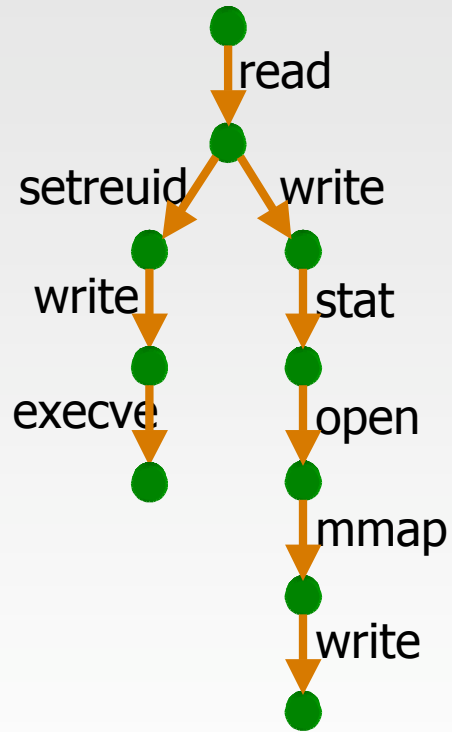
`open("/etc/passwd", O_WRONLY | O_APPEND) = 3`

`mmap(0, 0, 0, 0, 0, 0)`

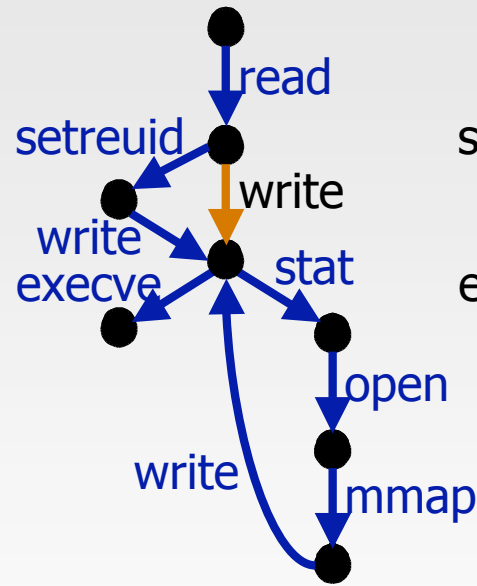
`write(3)`



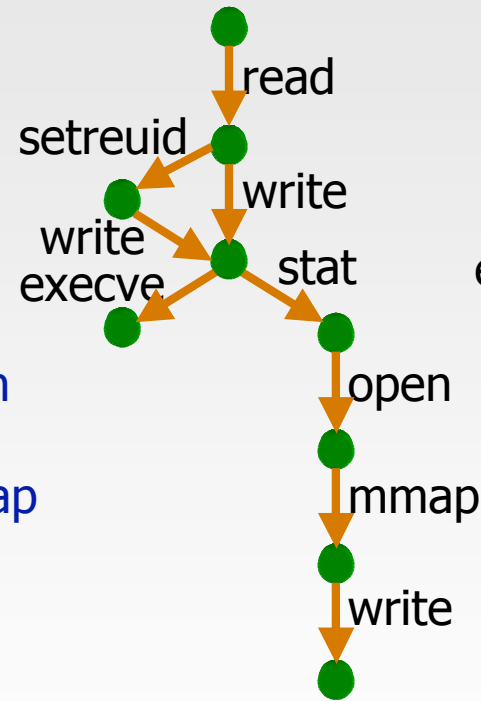
image = /bin/sh  $\wedge$  file[/etc/passwd].written = true



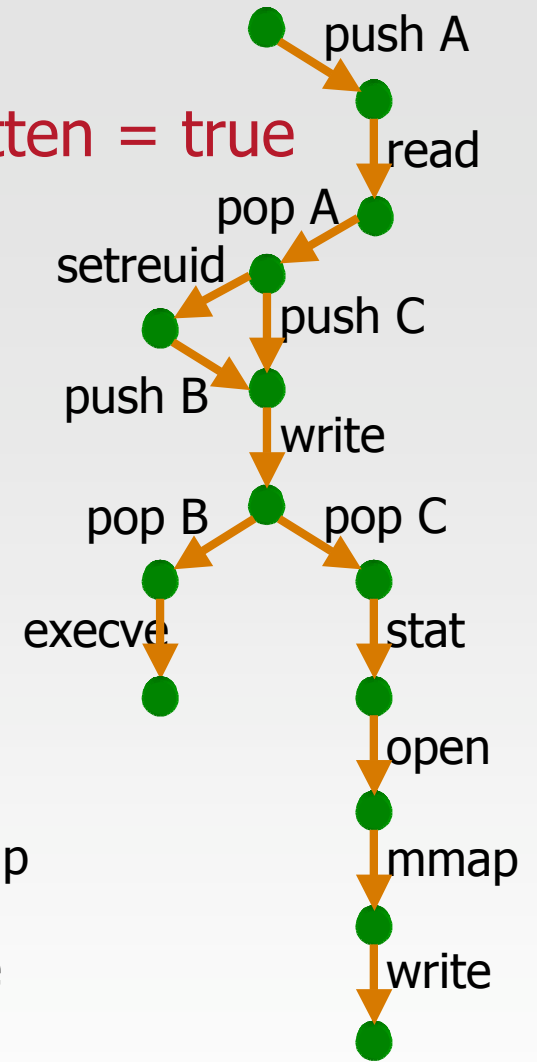
Stide



Digraph



NFA



PDA



`image = /bin/sh ^ file[/etc/passwd].written = true`

```
read(-1)  
setreuid(0, 0)  
write(-1)  
stat(0, 0)  
open("/etc/passwd", O_WRONLY | O_APPEND) = 3  
mmap(0, 0, 0, 0, 0, 0)  
write(3)  
execve("/bin/sh")
```



# Automatic Attack Discovery

---

- Automating previously manual mimicry attack discovery
  - Known attacks detected by NFA-based Stide model  
[Forrest *et al.* 1996]
  - Undetected mimicry attacks manually constructed  
[Tan *et al.* 2002]  
[Wagner & Soto 2002]
  - We automatically find undetected attacks



# Automatic Attack Discovery

## Stide Model Prevented Attack

	<i>wu-ftpd</i>	<i>restore</i>	<i>traceroute</i>	<i>passwd</i>
Execute rootshell	No	No	Yes	Yes
Write to /etc/passwd	No	No	No	No
Set /etc/passwd world-writable	Yes	Yes	Yes	No
Set /etc/passwd attacker-owned	Yes	Yes	Yes	No



# Automatic Attack Discovery

## Model-Checking Runtime (s)

	<i>wu-ftpd</i>	<i>restore</i>	<i>traceroute</i>	<i>passwd</i>
Execute rootshell	0.34	0.75	2.38	2.70
Write to /etc/passwd	0.39	0.73	1.33	0.71
Set /etc/passwd world-writable	39.10	74.41	0.90	2.02
Set /etc/passwd attacker-owned	41.11	65.21	1.15	1.81



# Conclusions

---

- Apply model-checking to:
  - Automatically find mimicry attacks
  - Prove that no mimicry attacks exist
- Benefits
  - No prior knowledge of attack sequences
  - No prior knowledge of attack obfuscations
  - Able to prove absence of mimicry attacks



# Conclusions

---

So... am I safer?

- **Yes:** Sequence-based models prevent some malicious behavior...
- **...but:** Some attacks remain undetected
  - Automatically discover weaknesses in the model
  - Argument restrictions expected to prevent additional attacks



# Conclusions

---

## Attack against traceroute writing to /etc/passwd

```
close
munmap
open("/etc/passwd", O_RDWR | O_APPEND) = 3
fcntl64
fcntl64
fstat64
mmap2
read
close
munmap
write(3)
```



Thank You!

Questions?

Jonathon T. Giffin

Georgia Institute of Technology  
giffin@cc.gatech.edu

Somesh Jha    Barton P. Miller

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

RAID 2006