Analysis of Packed and Obfuscated Program Binaries

Kevin Roundy
University of Wisconsin

Paradyn/Dyninst Week
April 27-28, 2009

Analysis-Resistant Malware

GhostNet

- Hacked embassies all over SE Asia

Conficker

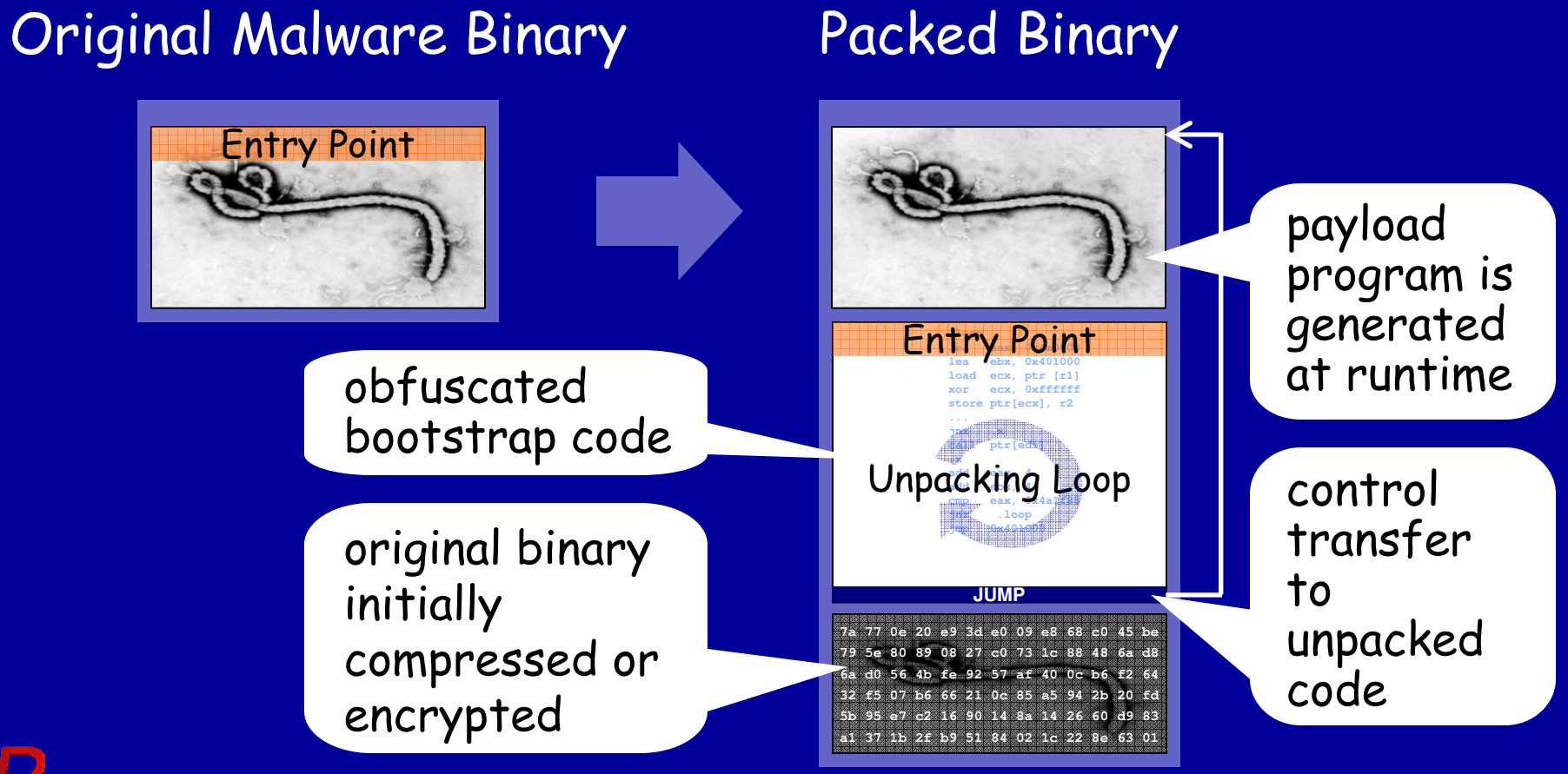
90% of malware resists analysis [1]

- Dynamically generated code
- Overwritten code
- Obfuscated control flow

[1] Packer (r)evolution, *Panda Research, 2008*

Dynamically Generated Code

75% of malware apply tools to *pack* their binaries, generating code at runtime [1]

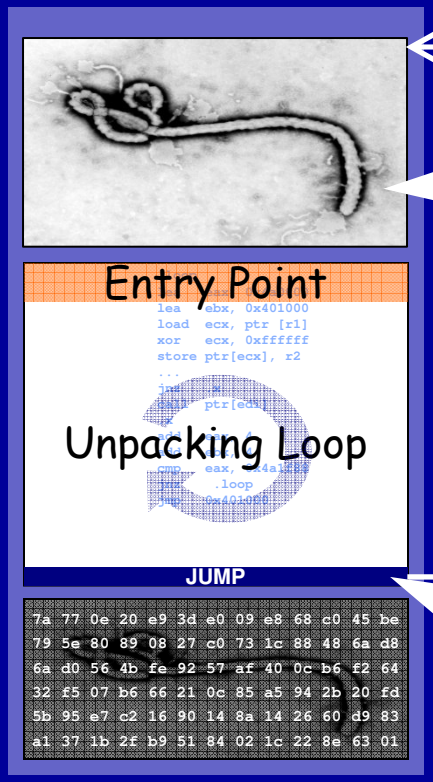


Dynamically Generated Code

75% of malware apply tools to *pack* their binaries, generating code at runtime [1]

Original Malware Binary

Packed Binary



payload program is generated at runtime

control transfer to unpacked code

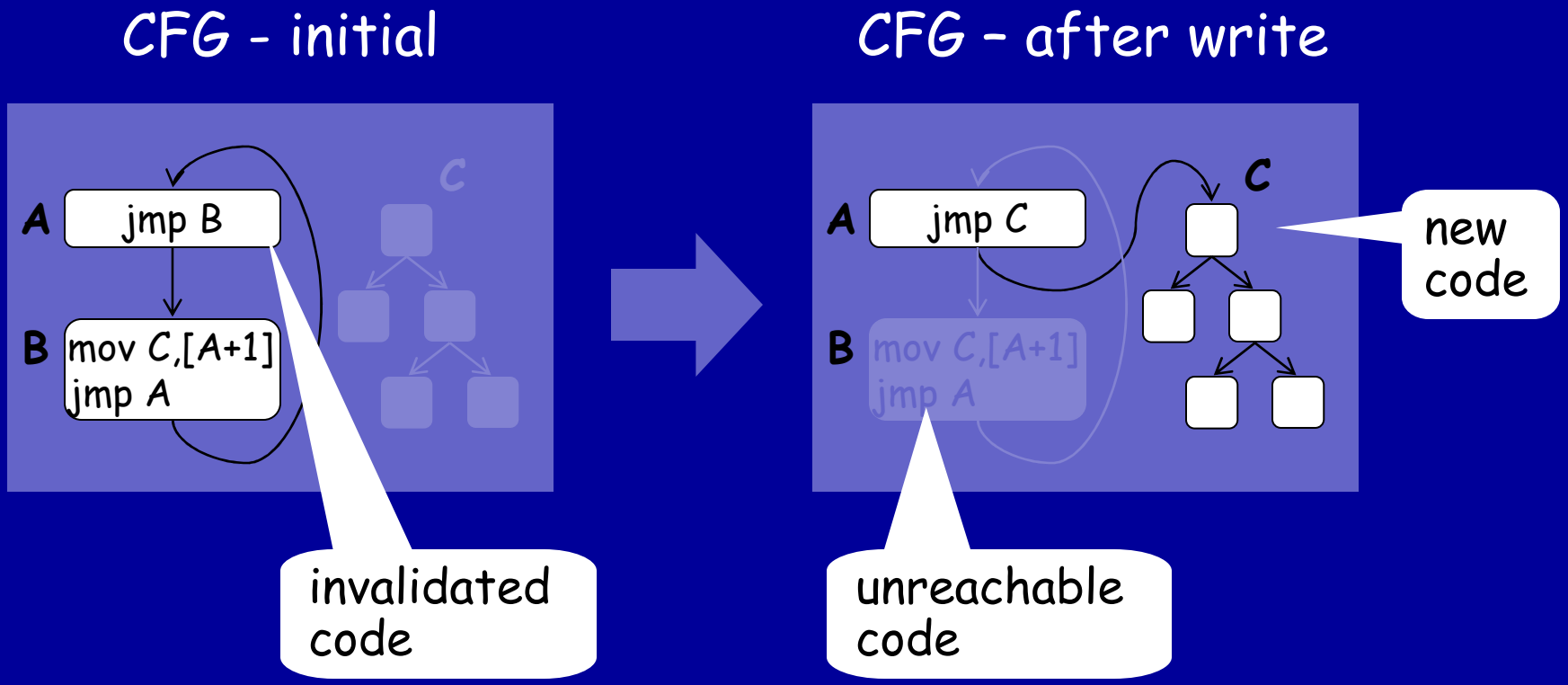
Packer growth rate of 6-8% per month [2]

75% of new malware uses custom packing [1]

[1] Malware Prevalence August 2008, *Panda Research*, 2008
[2] Exepacker Blacklisting Part 2, *Virus Bulletin*, March 2007

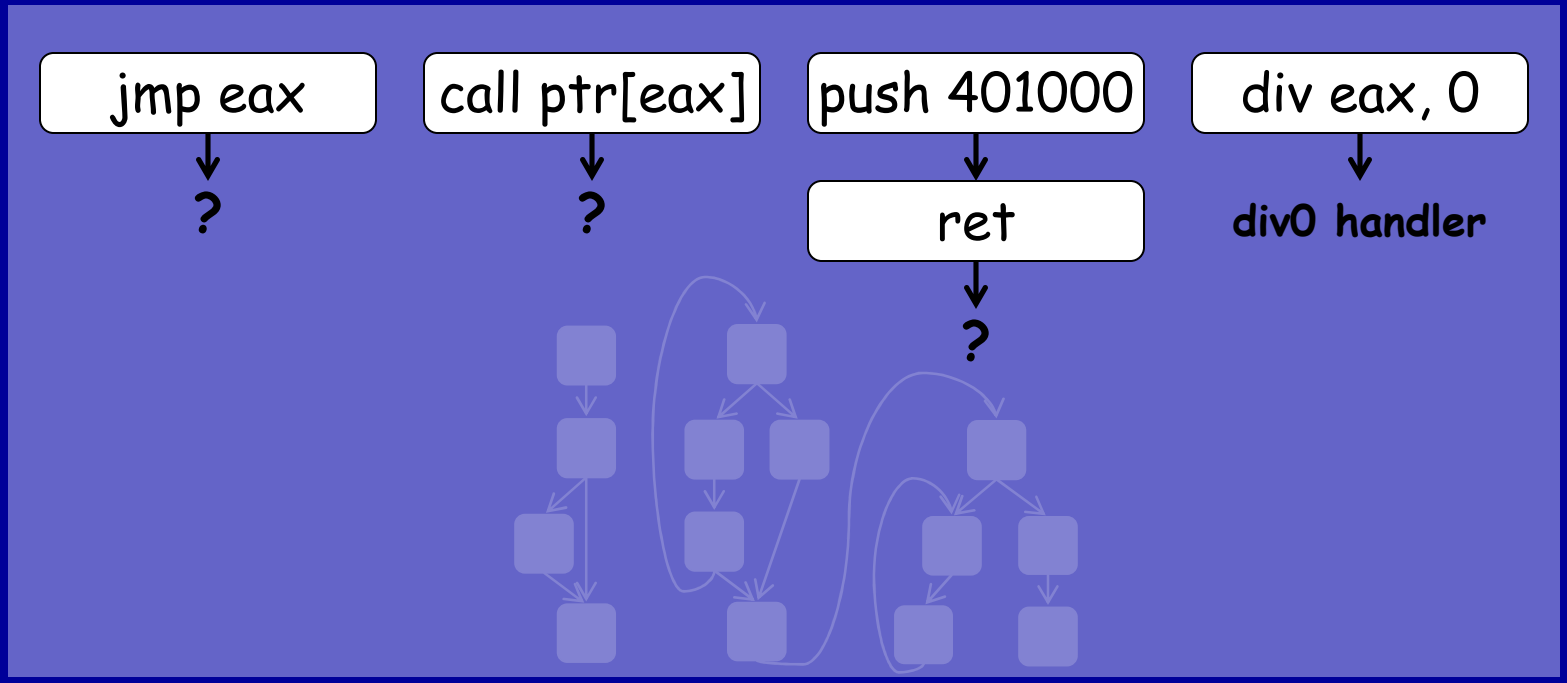
Overwritten Code

Self-modifying programs overwrite code at runtime



Obfuscated Control Flow

Statically un-analyzable control flow hides code



Our Goal

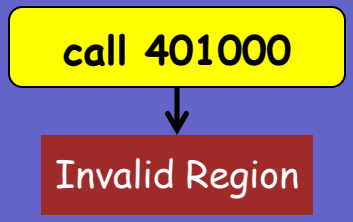
Analyze the code,
monitor and control the program

Solution: Hybrid analysis

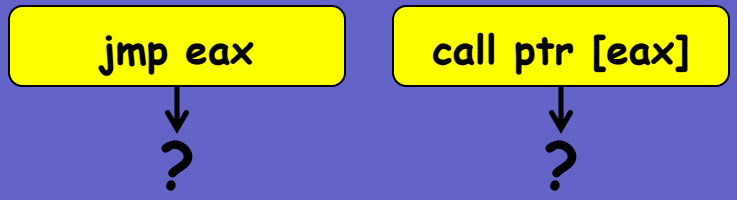
- Parsing — *static techniques*
- Instrumentation-based code discovery
- Code overwrite detection — *dynamic techniques*
- Signal- and exception-monitoring

Control Transfers to Instrument

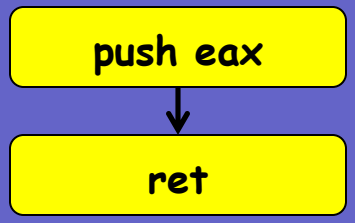
Invalid control transfers



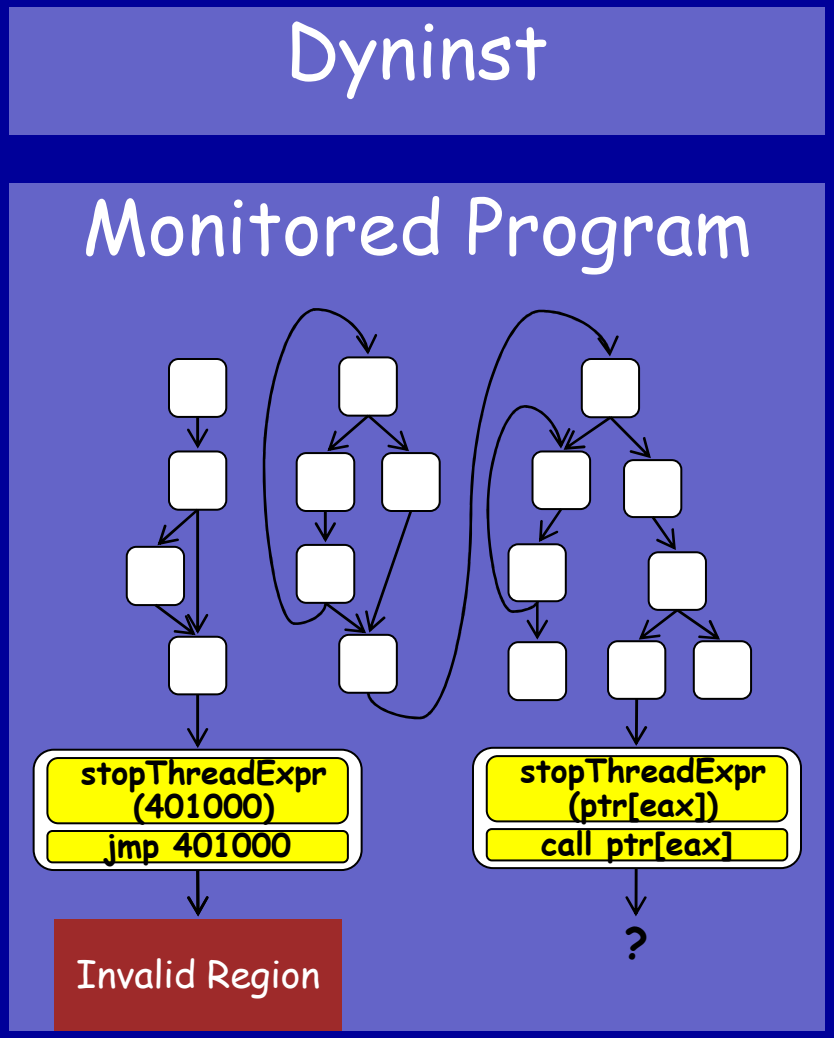
Indirect jumps/calls



Return instructions



Instrumentation-Based Discovery



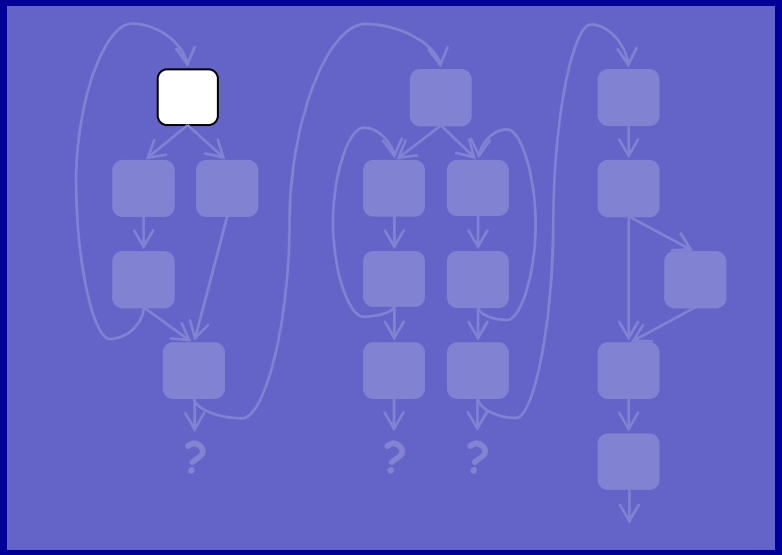
`stopThreadExpr`

- Evaluates instrumentation predicate
  - `constExpr`
  - `dynamicTargetExpr`
- Causes Dyninst to invoke synchronous callback
  - `cb(instrumentedAddr, targetAddr)`

Code Discovery Algorithm

Hybrid code discovery:

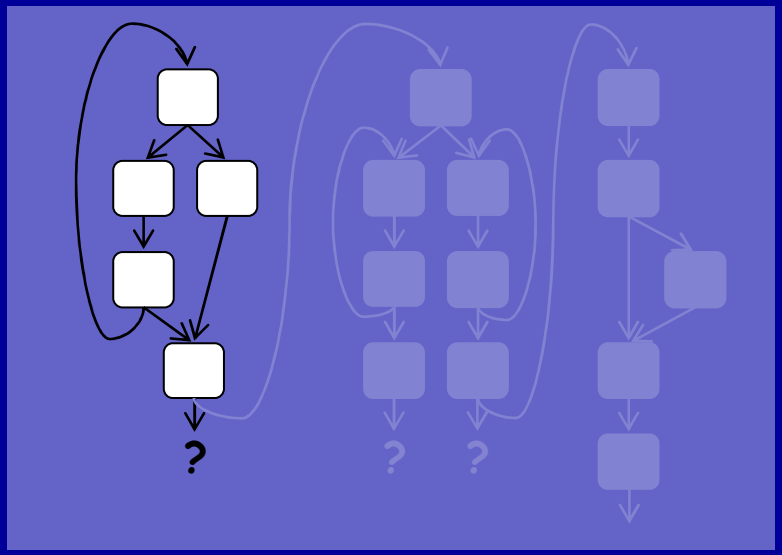
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

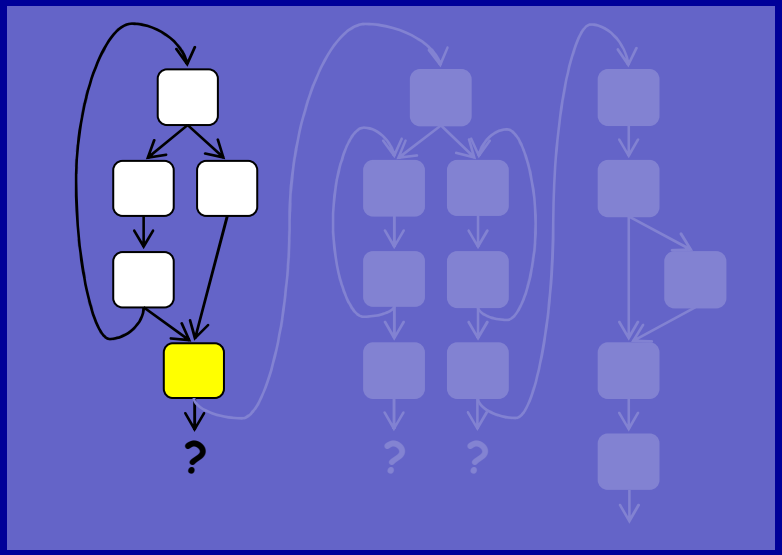
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

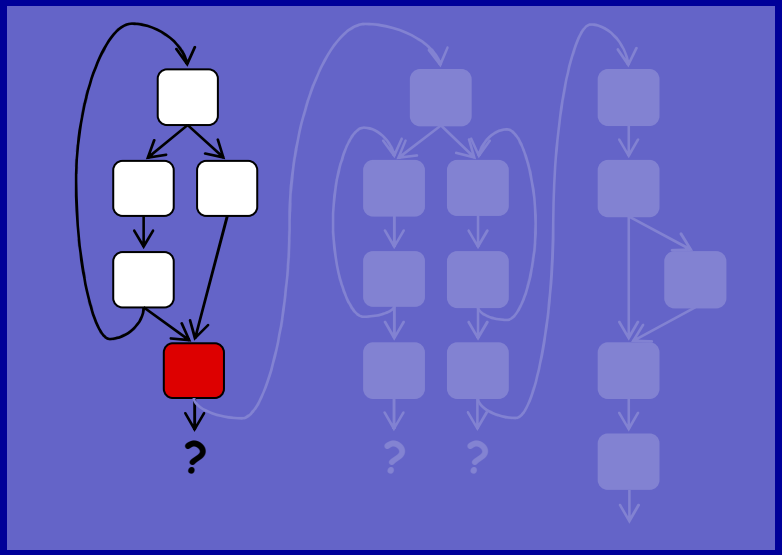
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

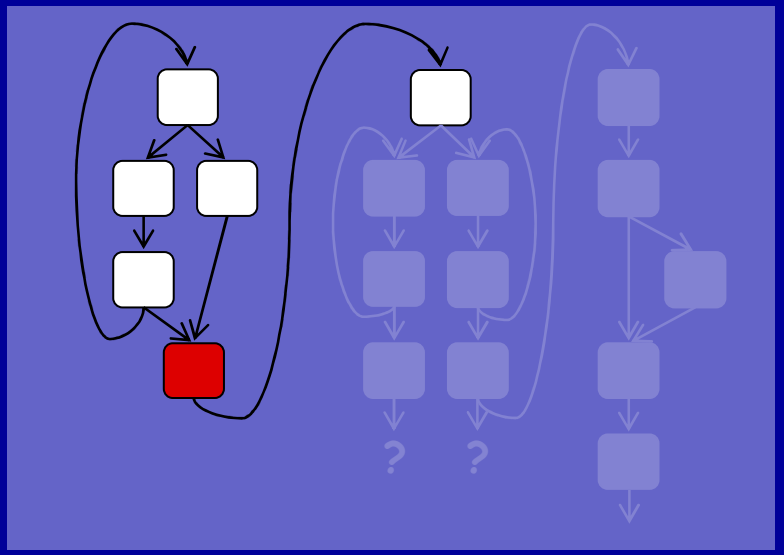
Hybrid code discovery:

Parse from known entry points

Instrument control flow that may lead to new code

Resume execution

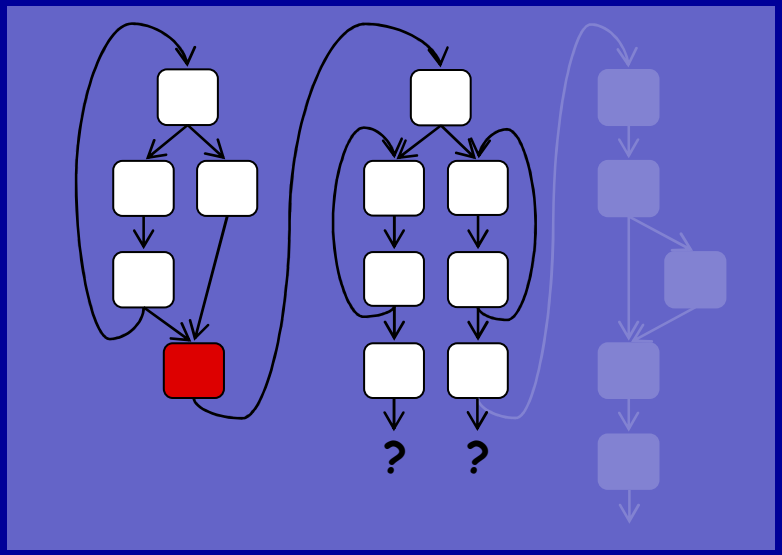
Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

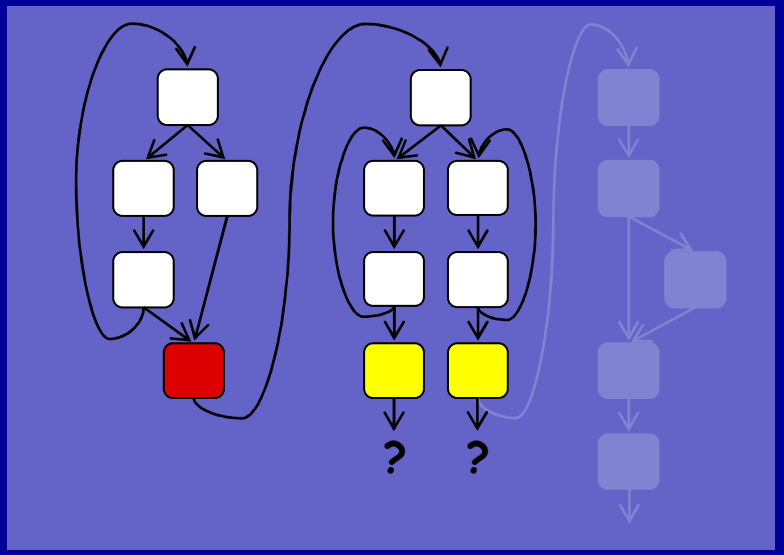
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

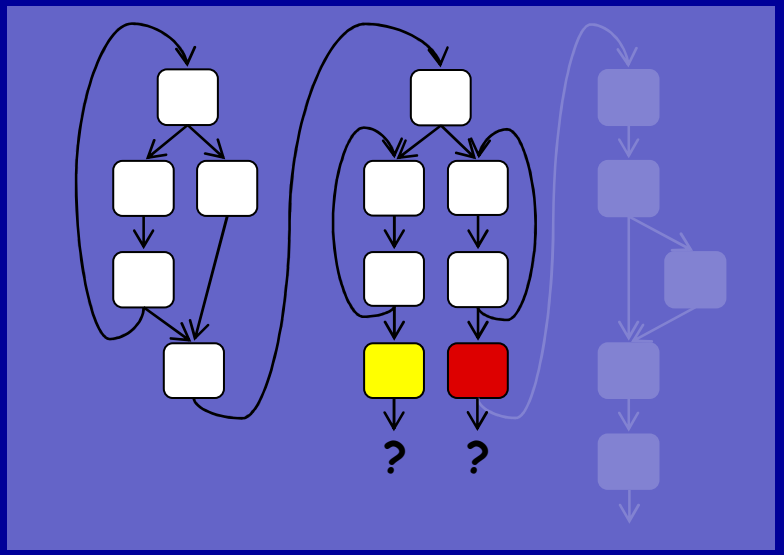
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

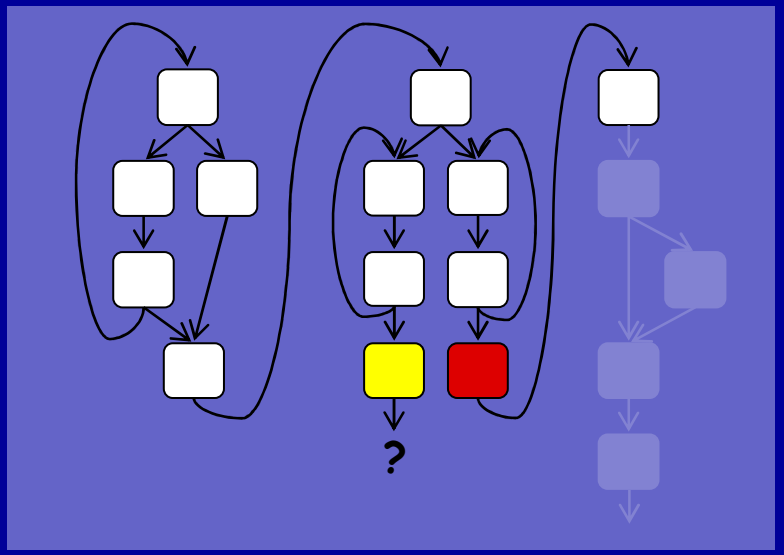
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Code Discovery Algorithm

Hybrid code discovery:

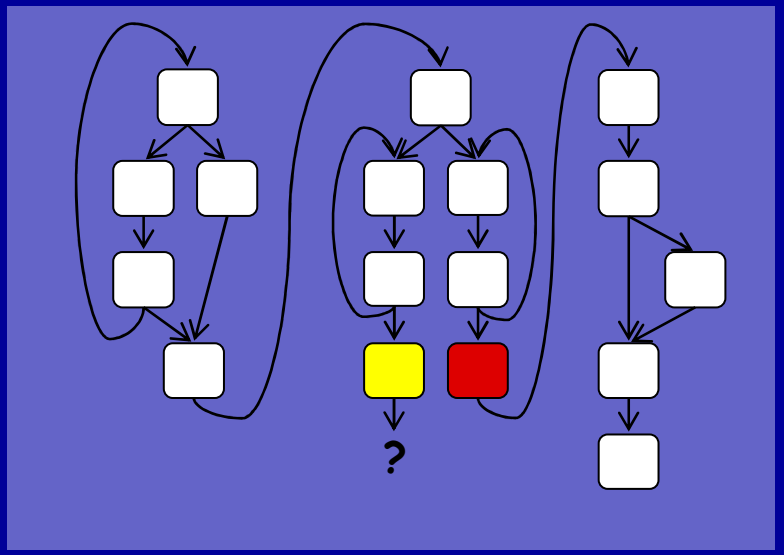
- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



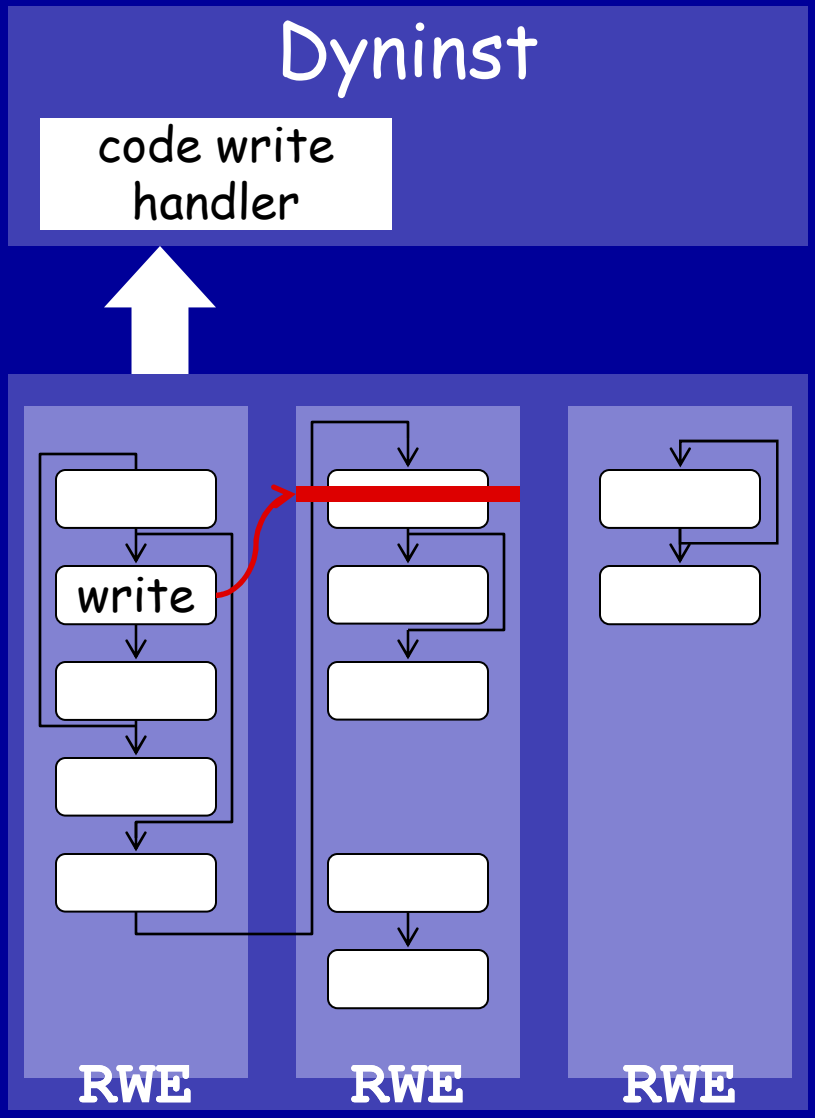
Code Discovery Algorithm

Hybrid code discovery:

- Parse from known entry points
- Instrument control flow that may lead to new code
- Resume execution
- Instrumentation discovers new entry points



Overwritten Code Discovery



Overwrite Detection

Possible strategies

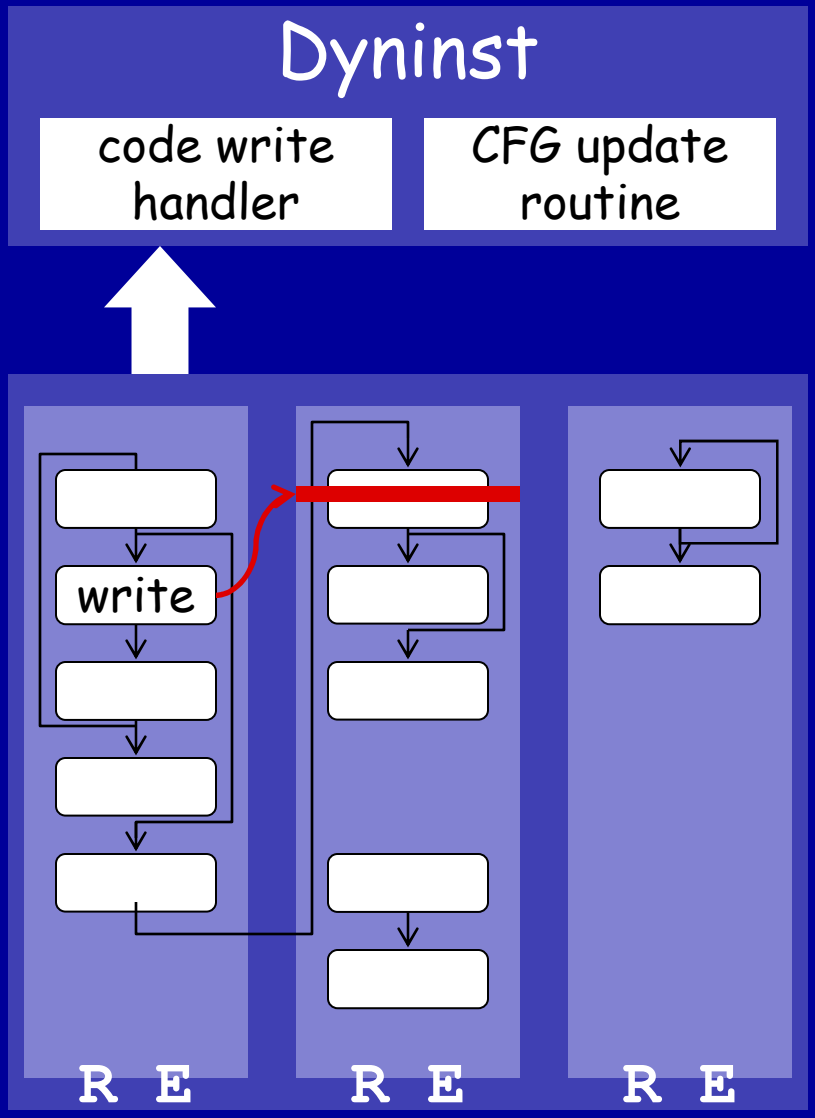
- Check each instruction in the trace for changes [1]
- Monitor writes to code regions

Page-level detection [2]

- Remove write permissions from code pages
- Write to code causes exception
- Handle exception

[1] Royal et al. PolyUnpack. ACSAC '06
[2] Maebe, De Bosschere. AADEBUG '03

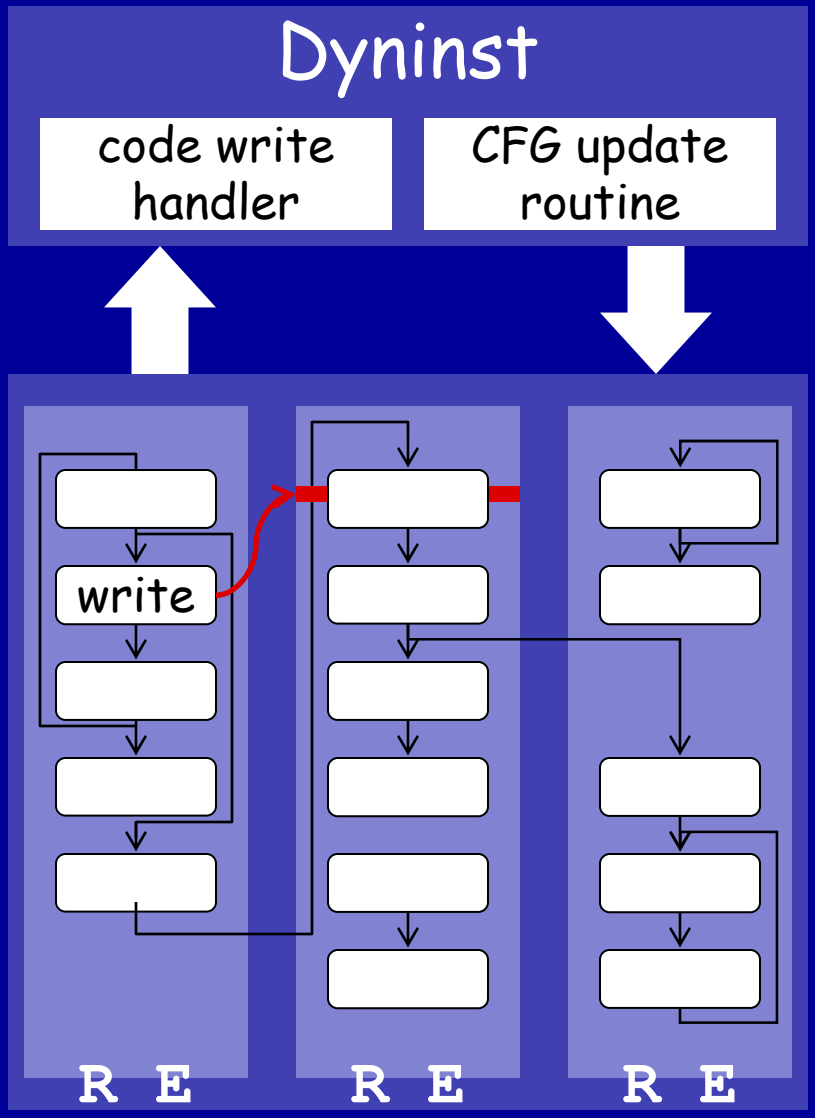
Overwritten Code Discovery



Updating the CFG

- emulate write instruction
- remove overwritten and unreachable blocks
- parse at entry points to overwritten regions

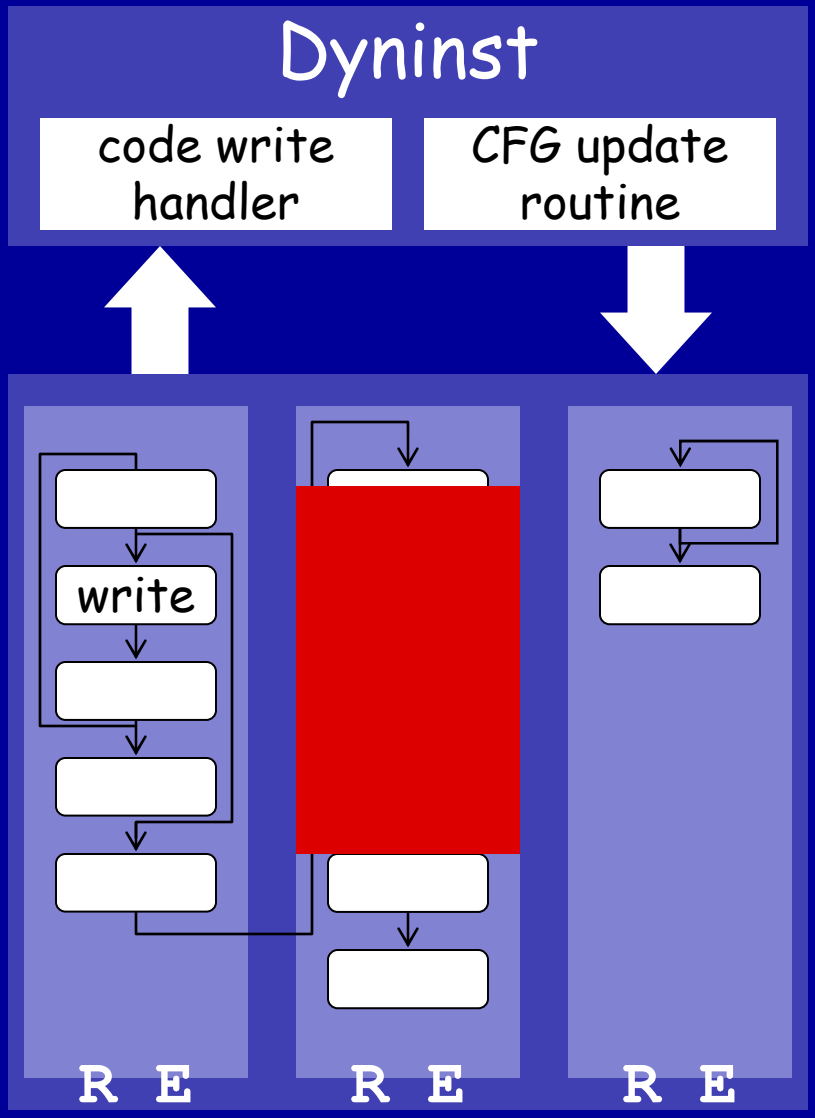
Overwritten Code Discovery



Updating the CFG

- emulate write instruction
- remove overwritten and unreachable blocks
- parse at entry points to overwritten regions

Overwritten Code Discovery

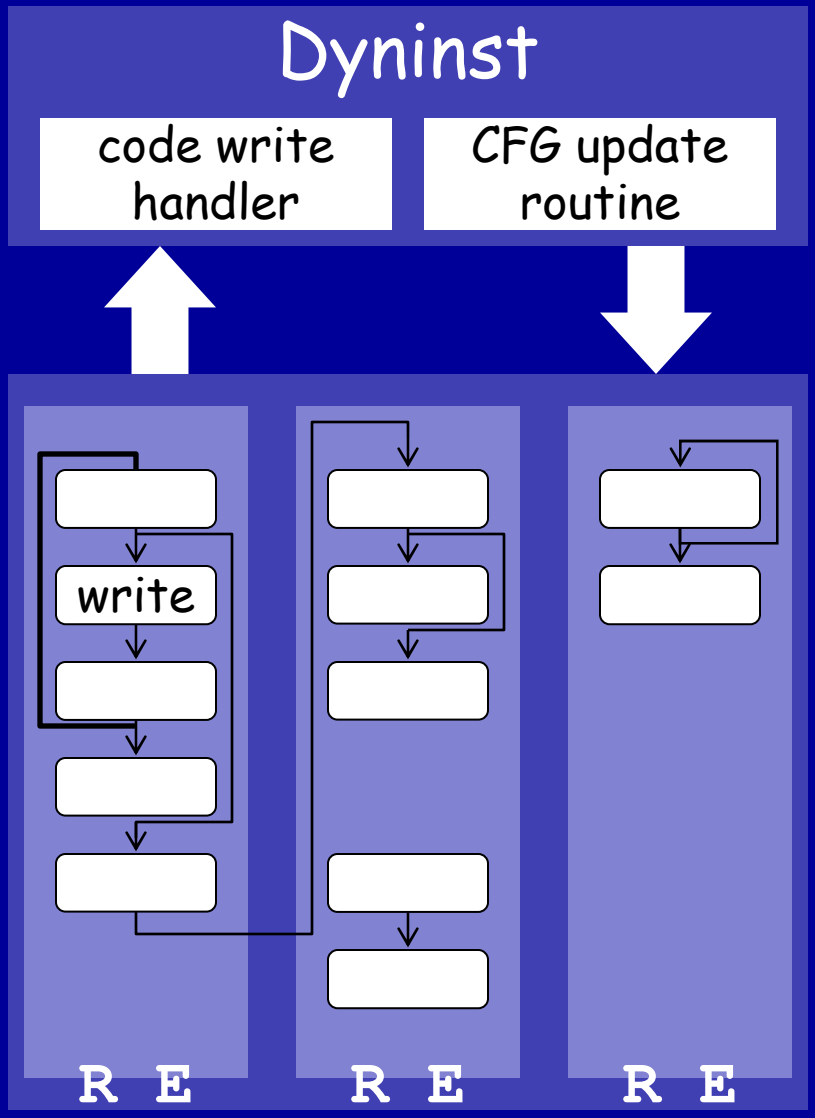


Optimizations

Cases to consider

- large incremental overwrites
- writes to data

Overwritten Code Discovery



Optimizations

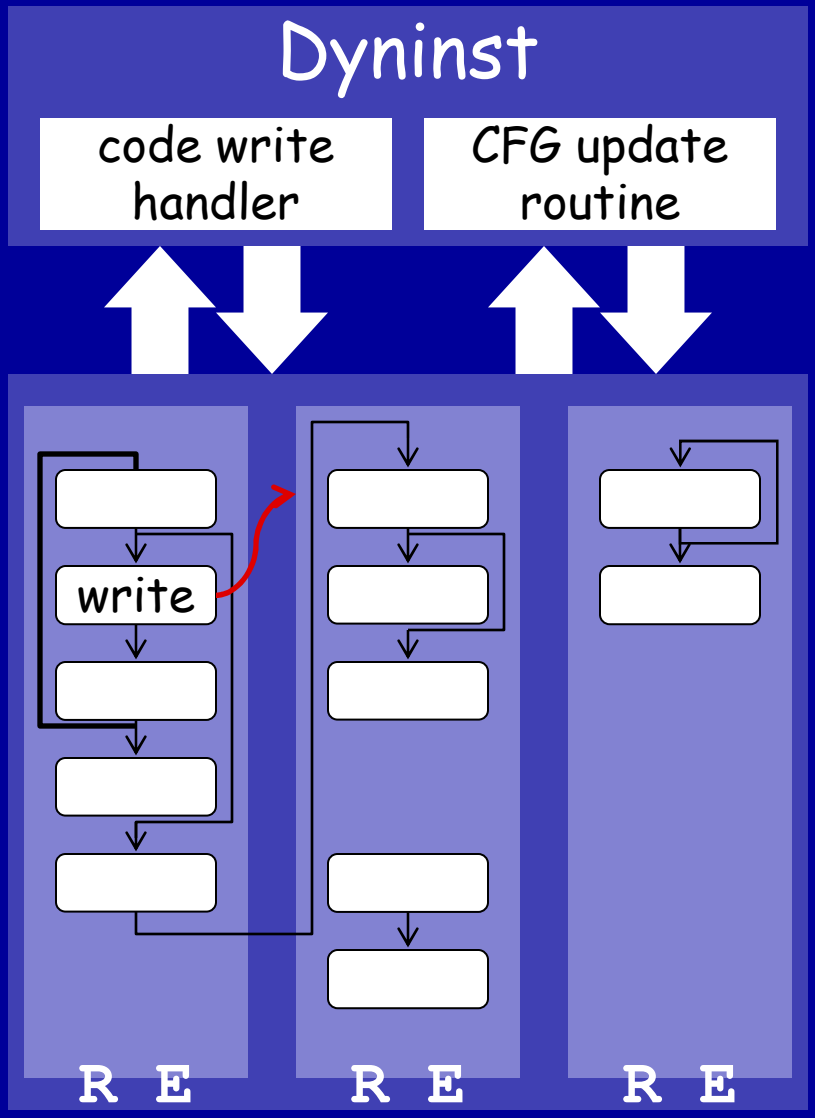
Cases to consider

- large incremental overwrites
- writes to data

Delaying the update

- until write routine terminates

Overwritten Code Discovery



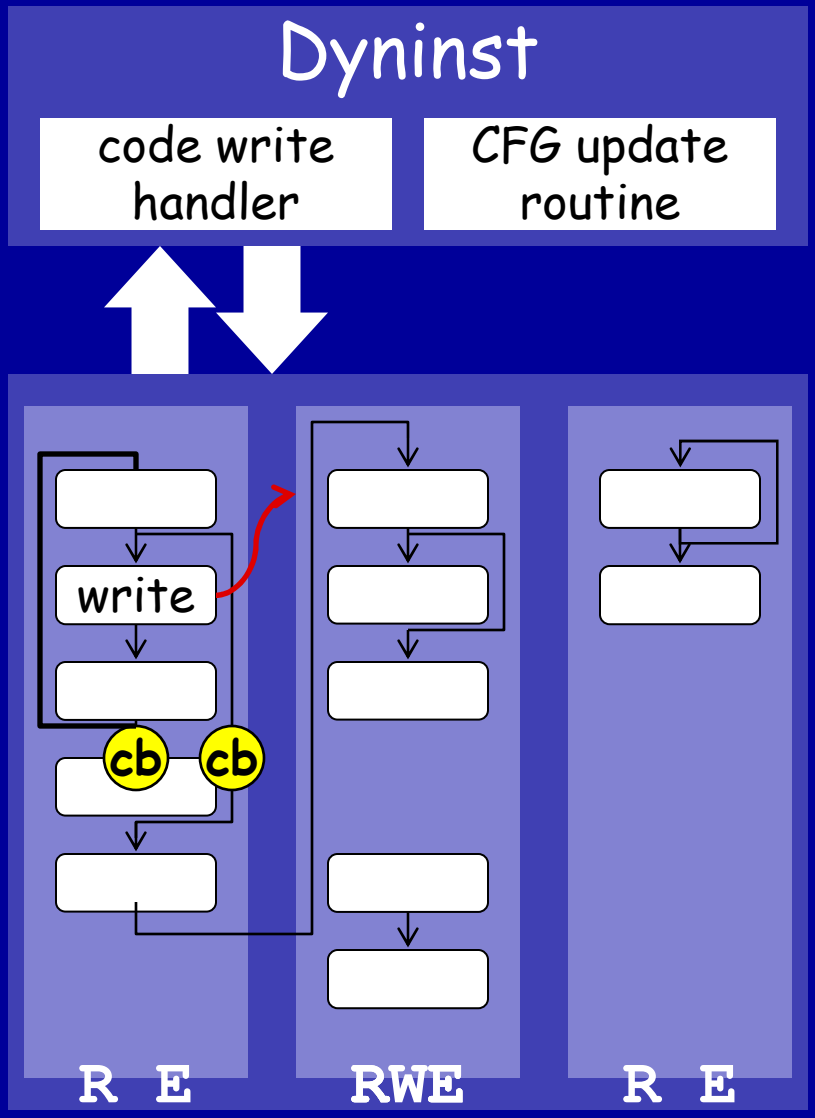
Delayed updates

Two components

1. Handle overwrite signal

2. Update CFG when writes end

Overwritten Code Discovery

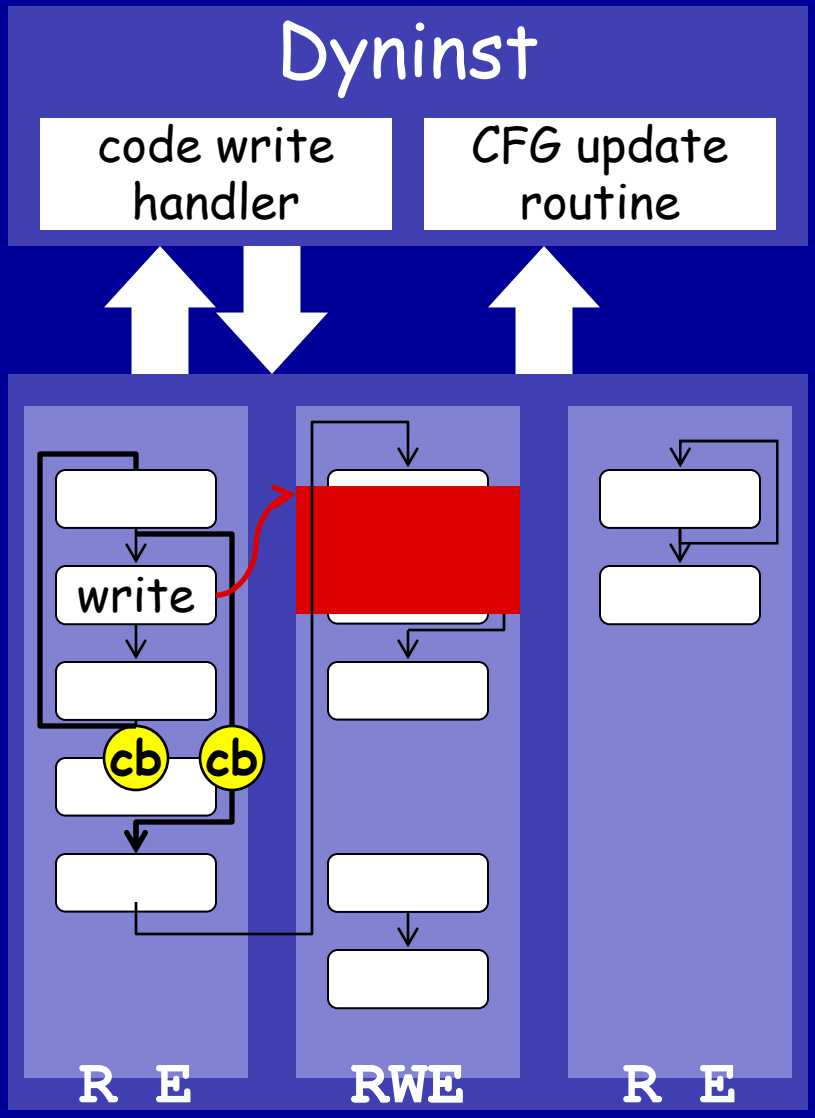


Delayed updates

Two components

1. Handle overwrite signal
   a) instrument write loop exits
   b) copy overwritten page
   c) restore write permissions
   d) resume execution
2. Update CFG when writes end

Overwritten Code Discovery

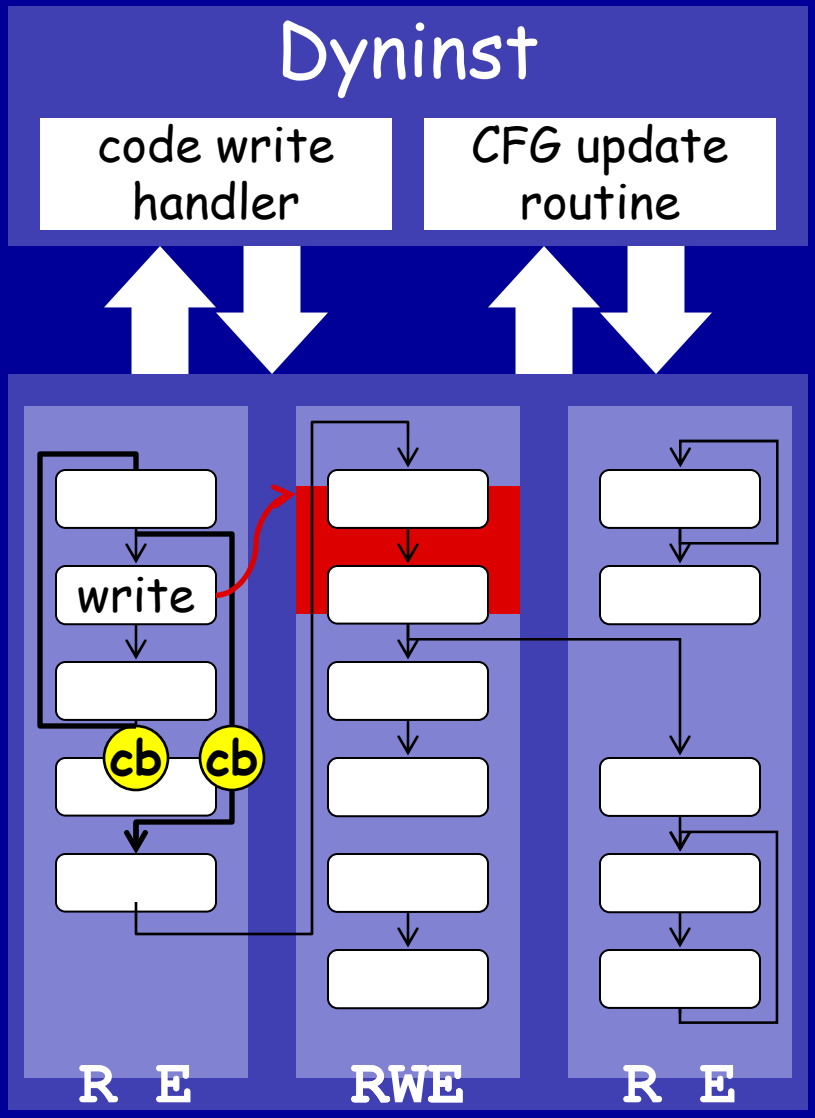


Delayed updates

Two components

1. Handle overwrite signal
   a) instrument write loop exits
   b) copy overwritten page
   c) restore write permissions
   d) resume execution
2. Update CFG when writes end
   a) remove overwritten and unreachable blocks
   b) parse at entry points to overwritten regions
   c) remove write permissions

Overwritten Code Discovery

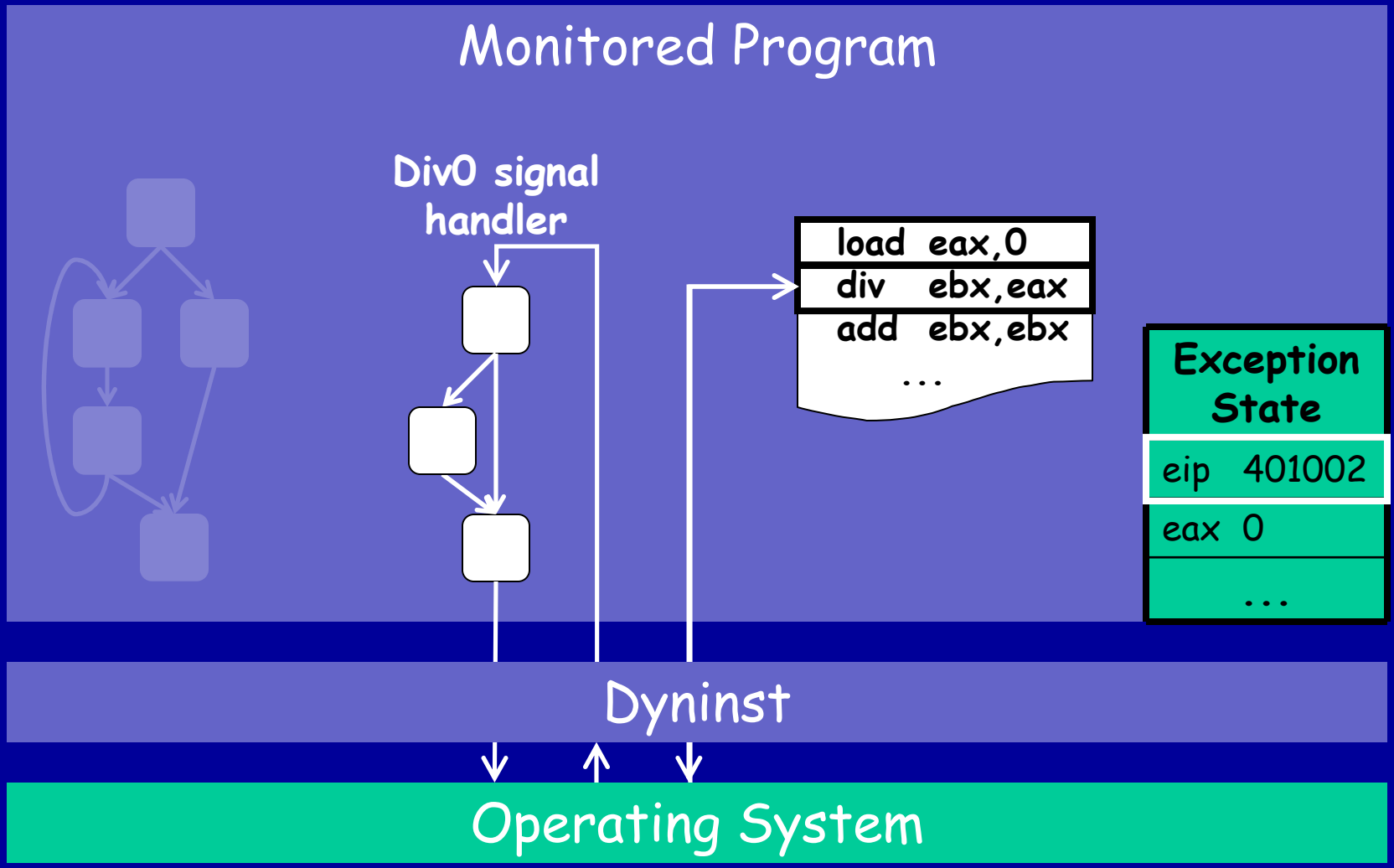


Delayed updates

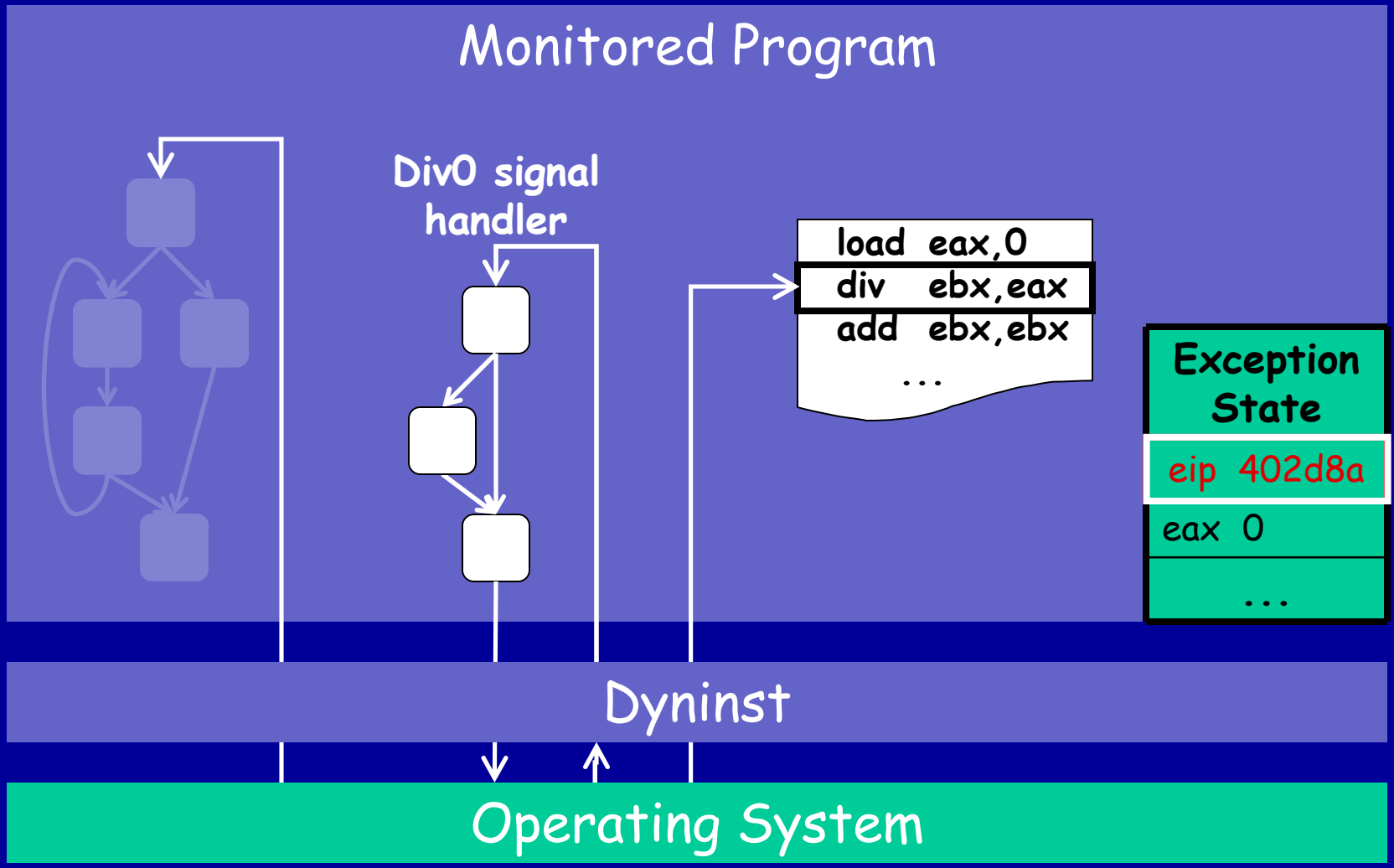
Two components

1. Handle overwrite signal
   a) instrument write loop exits
   b) copy overwritten page
   c) restore write permissions
   d) resume execution
2. Update CFG when writes end
   a) remove overwritten and unreachable blocks
   b) parse at entry points to overwritten regions
   c) remove write permissions

Signal and Exception Handler Discovery

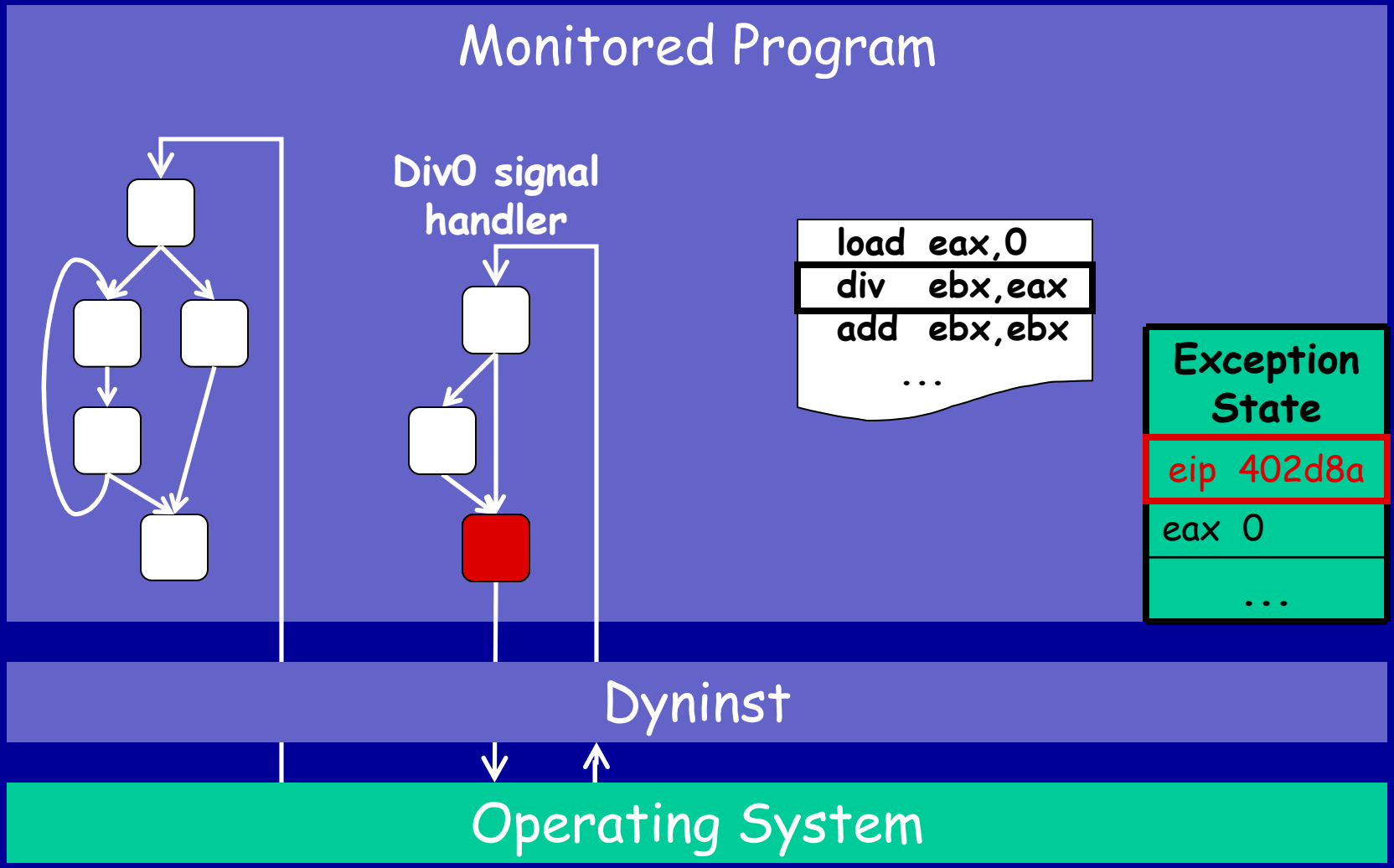


Handler-Based Redirections [1]



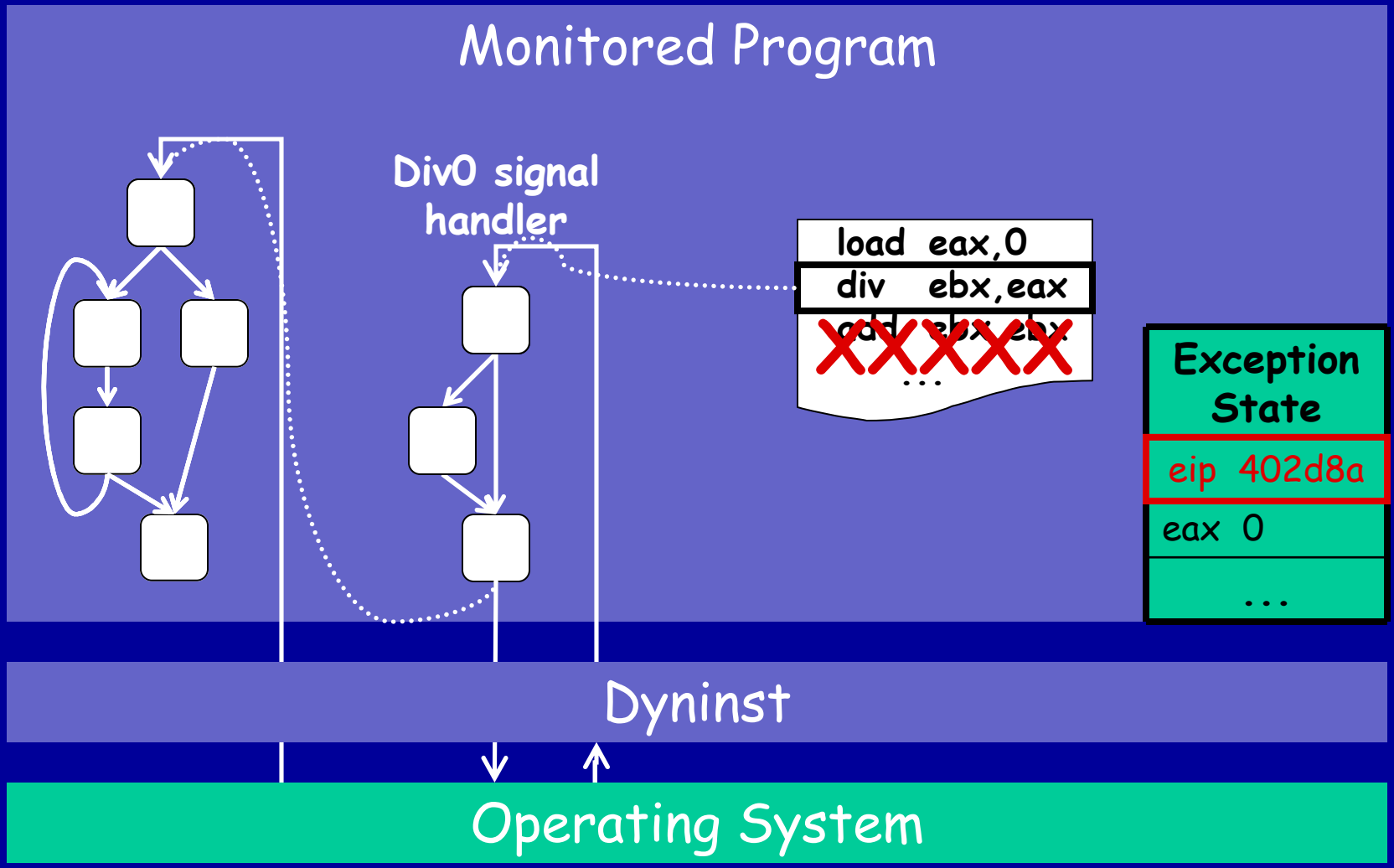
[1] Popov, Debray, Andrews. Usenix 2007
Danekhar. http://www.codeproject.com/KB/system/inject2exe.aspx 2005

Instrumenting the Handlers



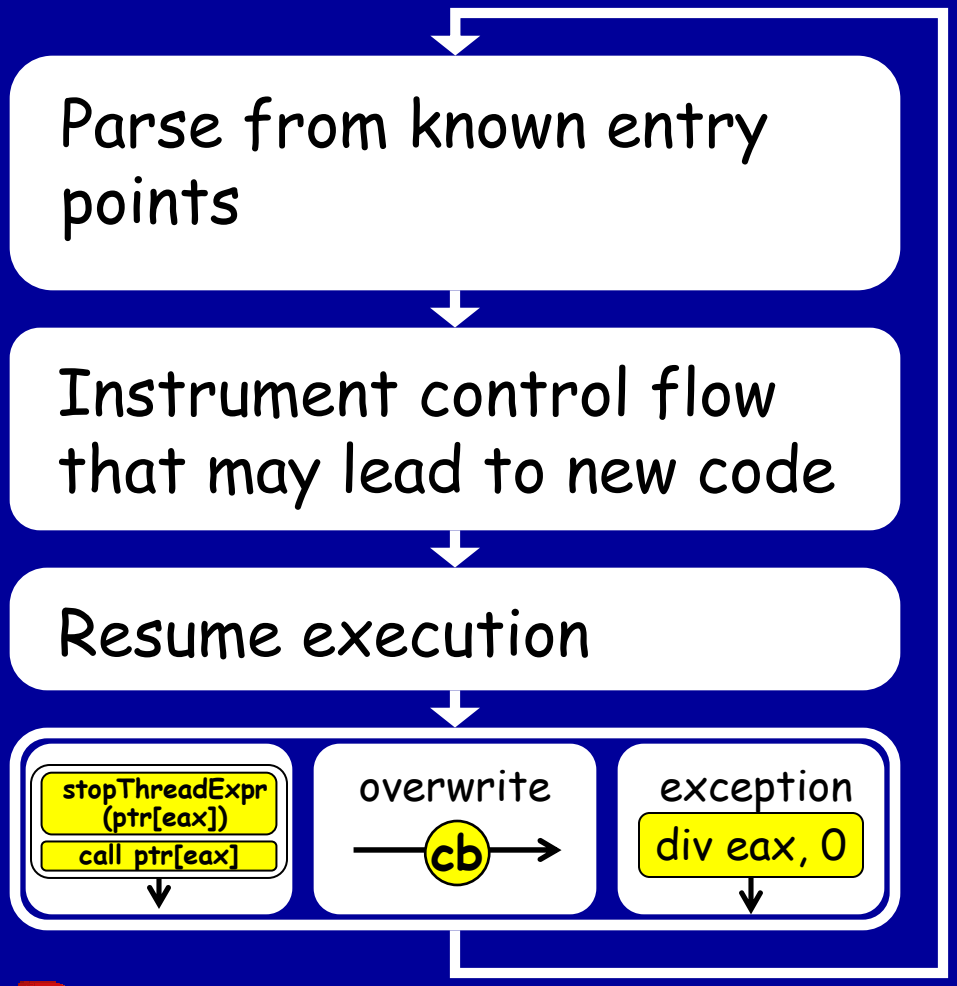
[1] Popov, Debray, Andrews. Usenix 2007
Danekhar. http://www.codeproject.com/KB/system/inject2exe.aspx 2005

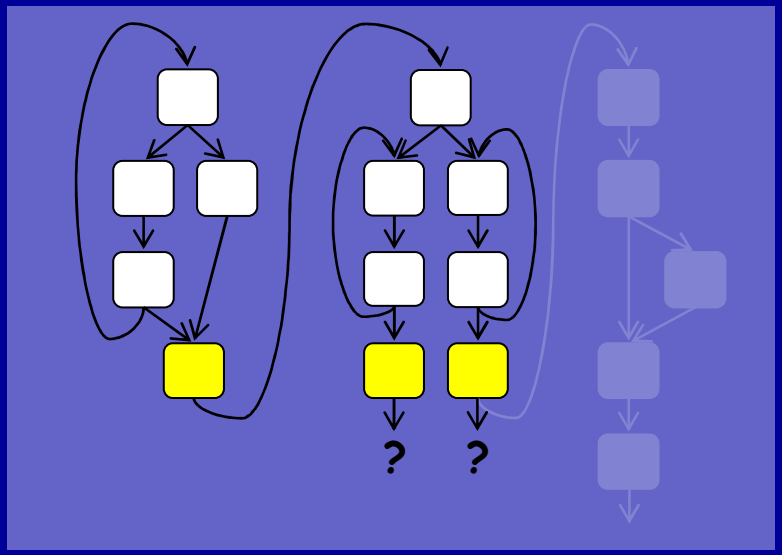
Handler-Based Redirections [1]



[1] Popov, Debray, Andrews. Usenix 2007
Danekhar. http://www.codeproject.com/KB/system/inject2exe.aspx 2005

Overall Algorithm





Fully Analyzed Packed Programs

Packer	malware market share[1]	Dyninst
UPX	9.45%	yes
PolyEnE	6.21%	yes
EXECryptor	4.06%	
Themida	2.95%	
PECompact	2.59%	
Upack	2.08%	yes
nPack	1.74%	yes
Aspack	1.29%	yes
FSG	1.26%	yes
Asprotect	0.43%	
Armadillo	0.37%	
Yoda's Protector	0.33%	
WinUPack	0.17%	yes
MEW	0.13%	yes

Tamper-resistance techniques (e.g., self-checksumming)

[1] Packer (r)evolution. Panda Research, 2008. Two-month average Feb-March 2008.