PANDA: Platform for Architecture-Neutral Dynamic Analysis

https://github.com/moyix/panda

Brendan Dolan-Gavitt
NYU

NYU TANDON SCHOOL OF ENGINEERING

Josh Hodosh, Patrick Hulin, Tim Leek, and Ryan Whelan
MIT Lincoln Lab

Who Am I?

- Assistant Professor at NYU School of Engineering
- PhD at Georgia Tech under Wenke Lee
- Things I've done that you may know: Volatility, pdbparse, creddump, PANDA

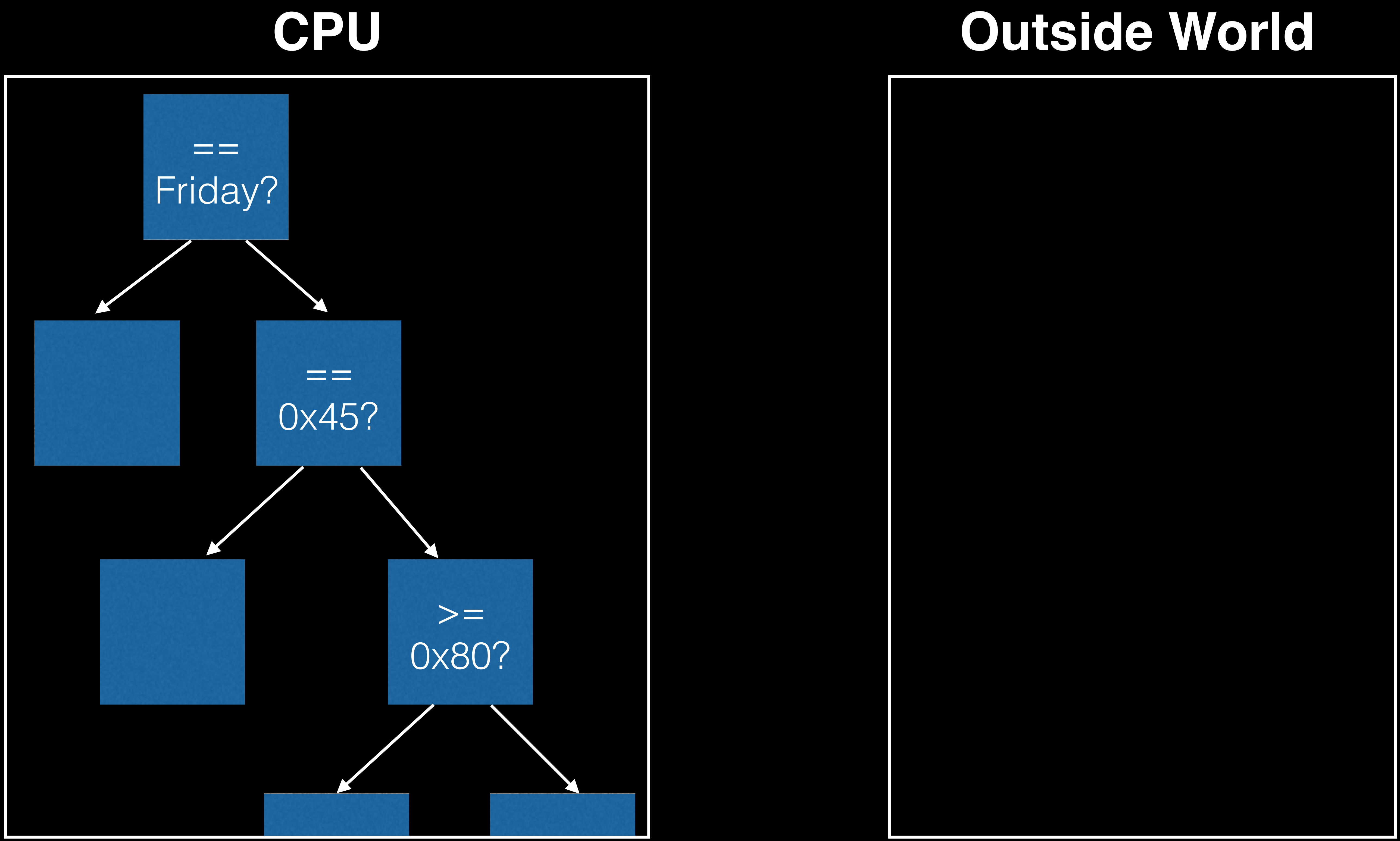
What is This Talk

- An introduction to PANDA: a Platform for Architecture-Neutral Dynamic Analysis
- A demonstration of interesting projects we've used PANDA for
- A transparent attempt to inspire people to collaborate and do something interesting with PANDA!

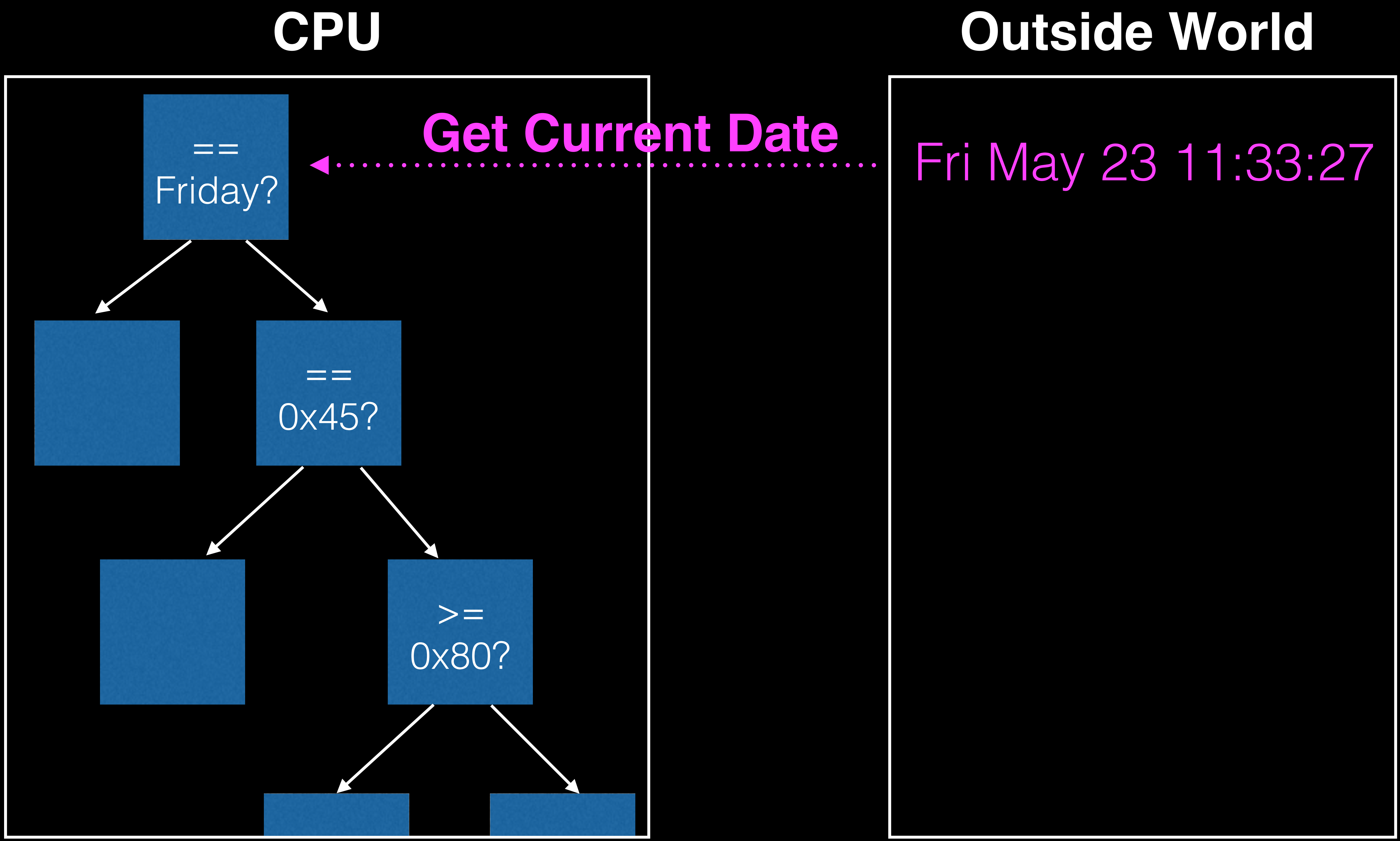
PANDA: Built for Dynamic Analysis

- Based on QEMU 1.0.1 whole-system emulator
- Deterministic record/replay
- Translation from binary to LLVM for all QEMU architectures (extended from S2E code)
- Android (ARM) emulation support
- Plugin architecture – easy to extend to new analyses

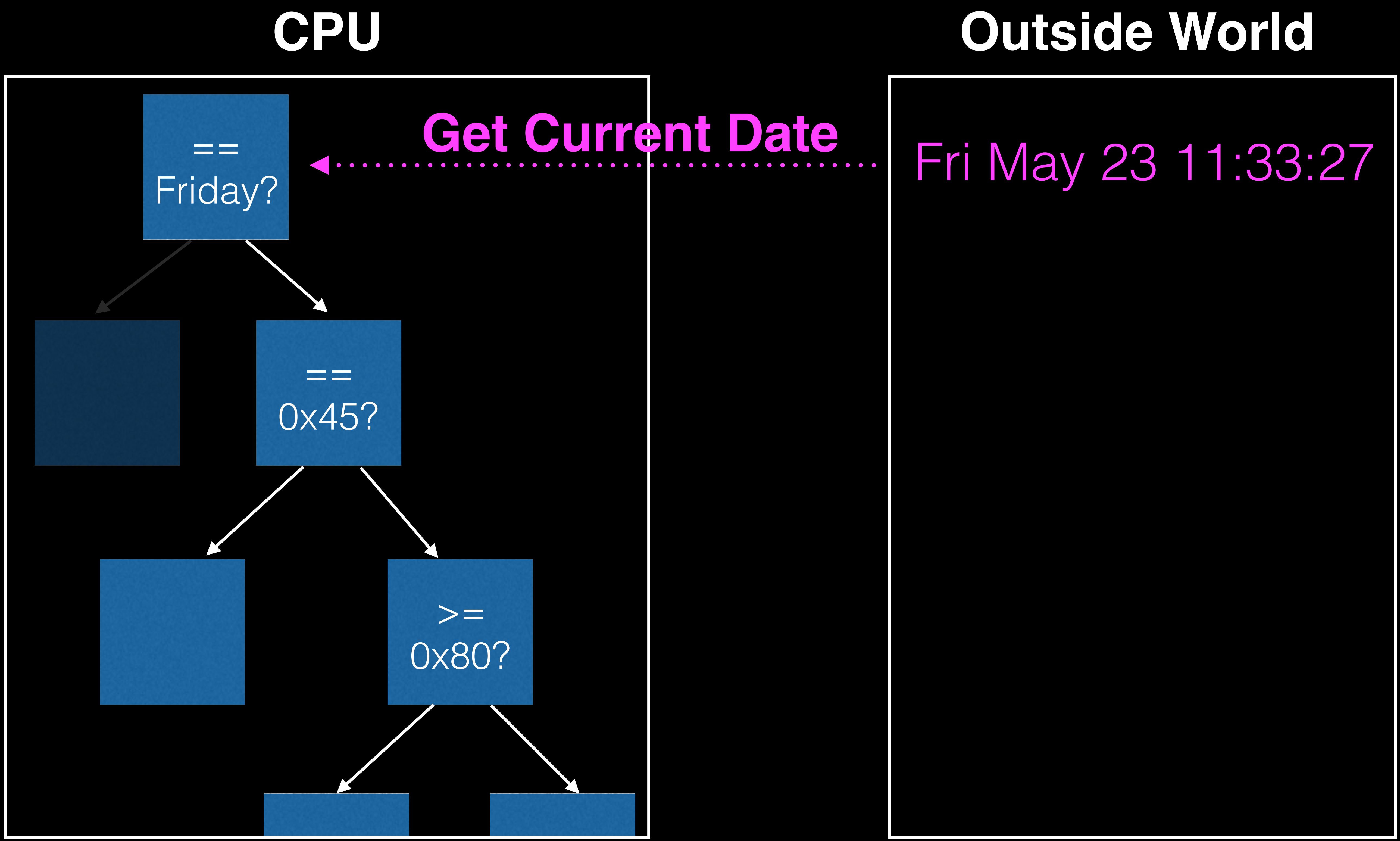
Record/Replay



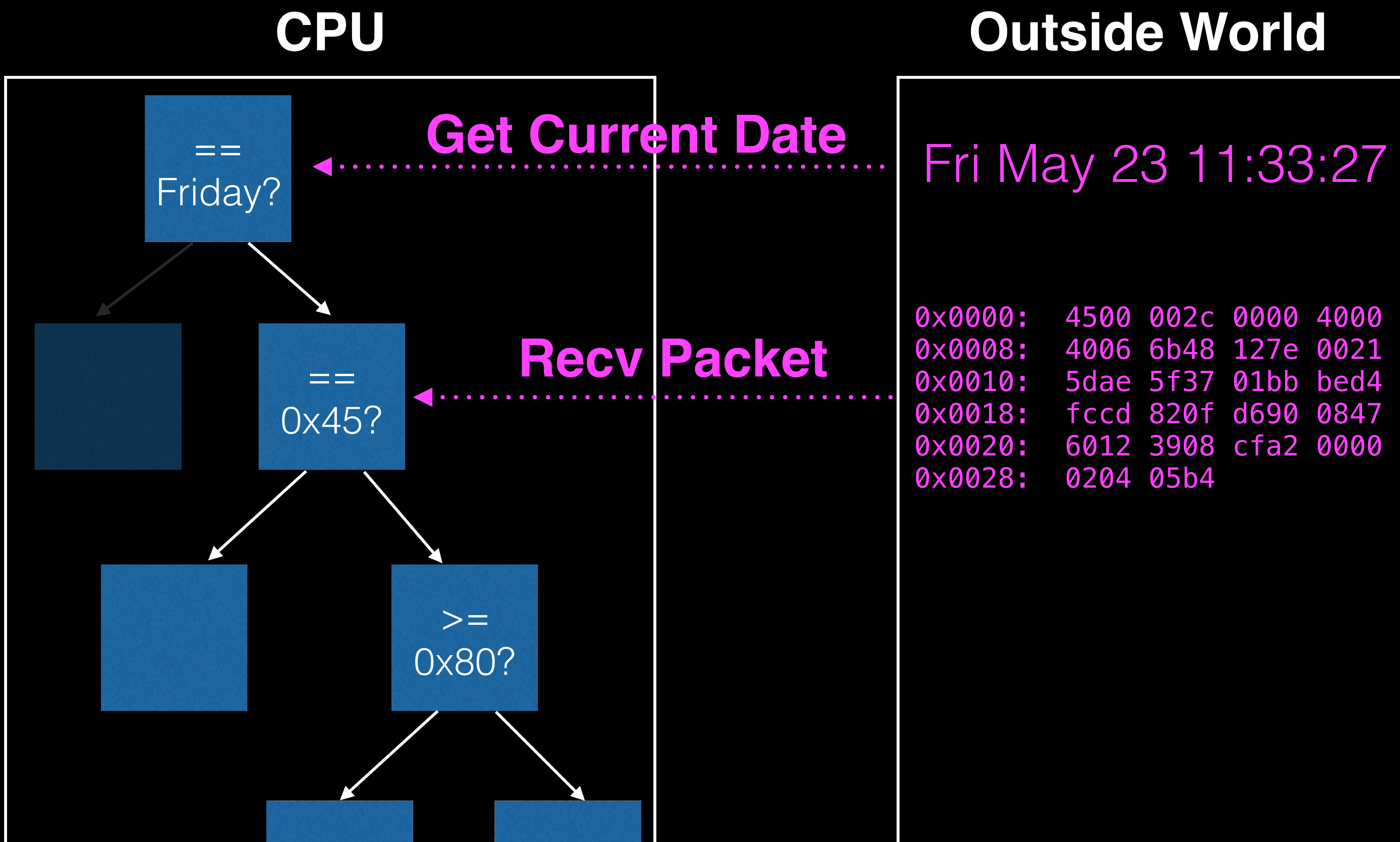
Record/Replay



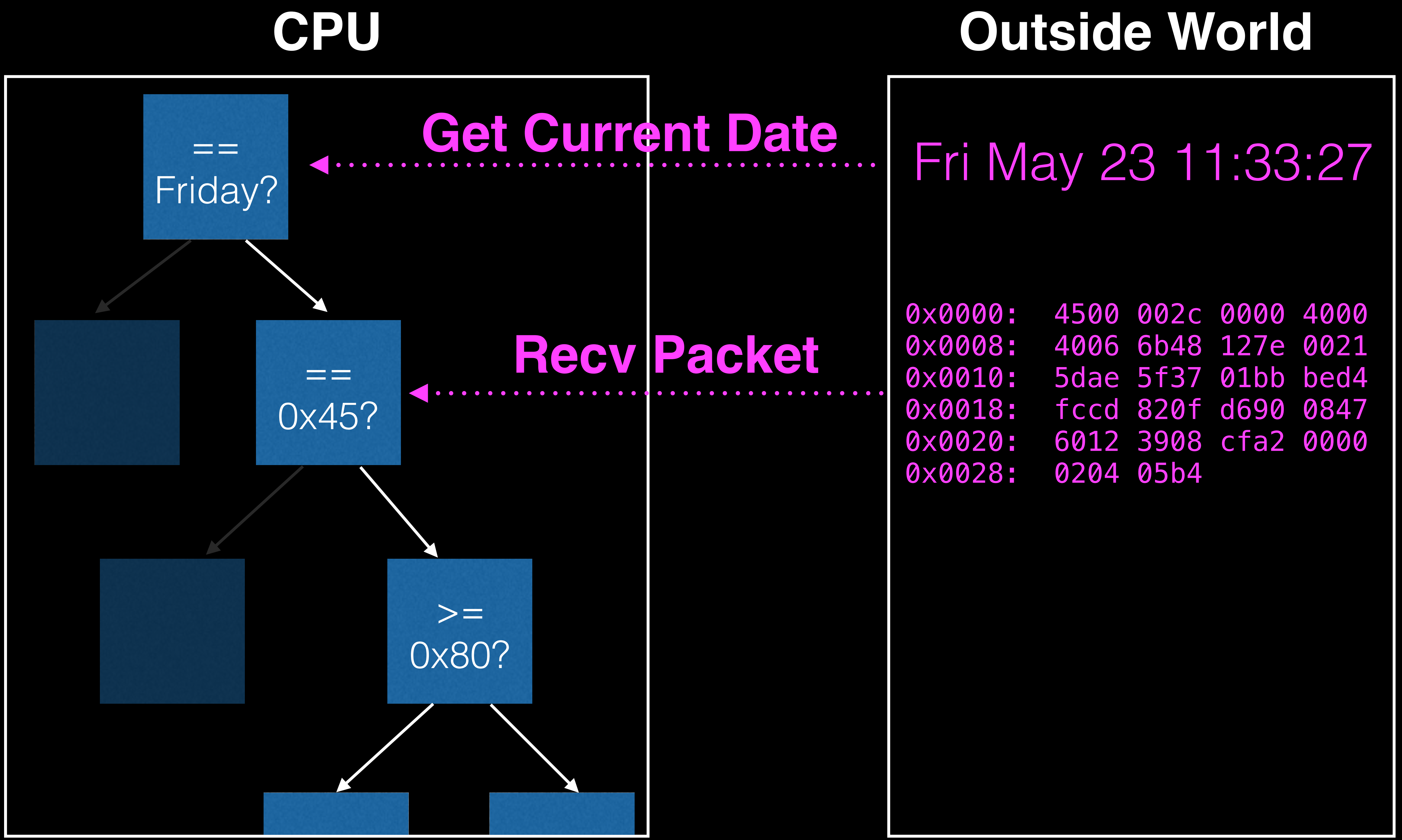
Record/Replay



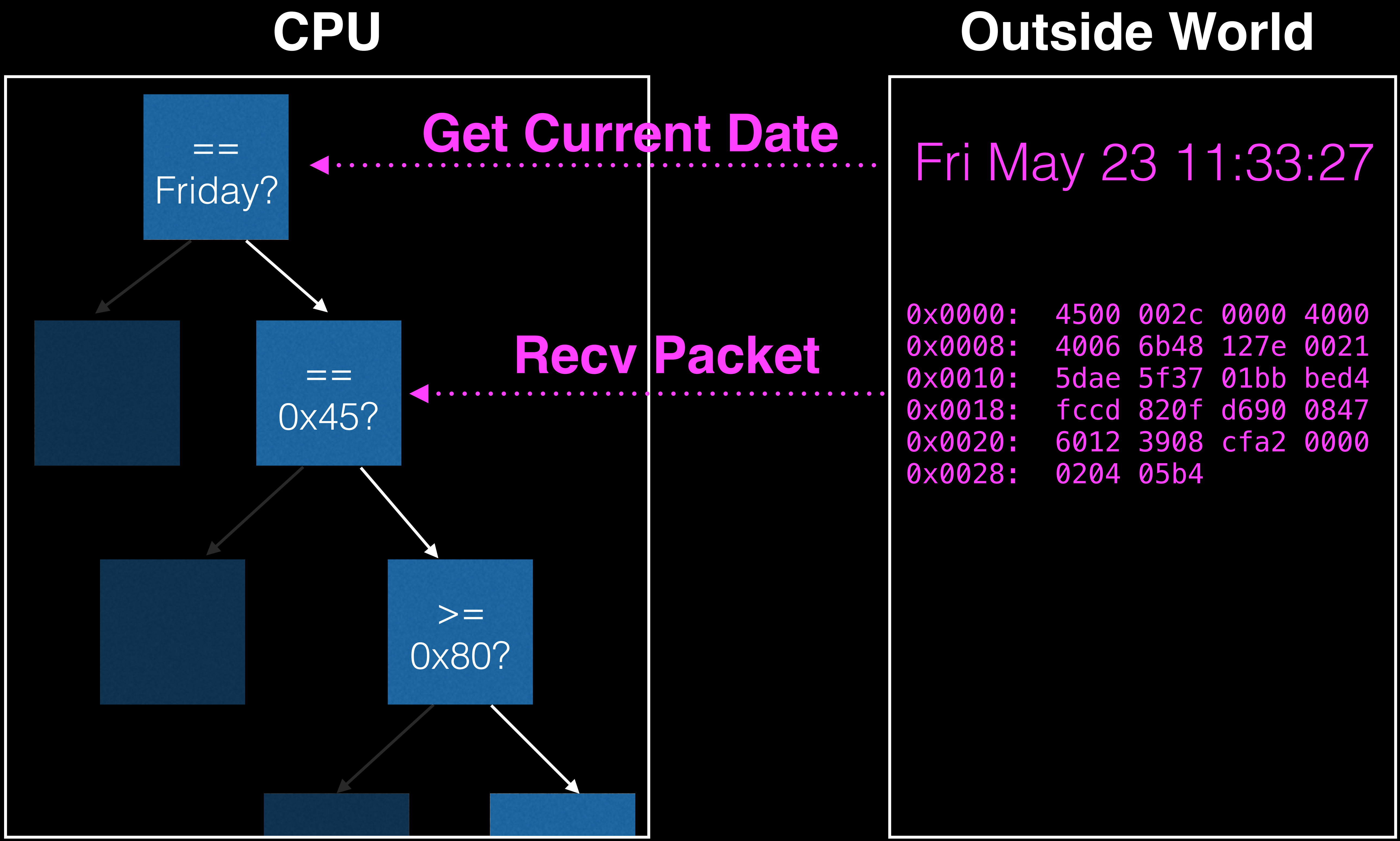
Record/Replay



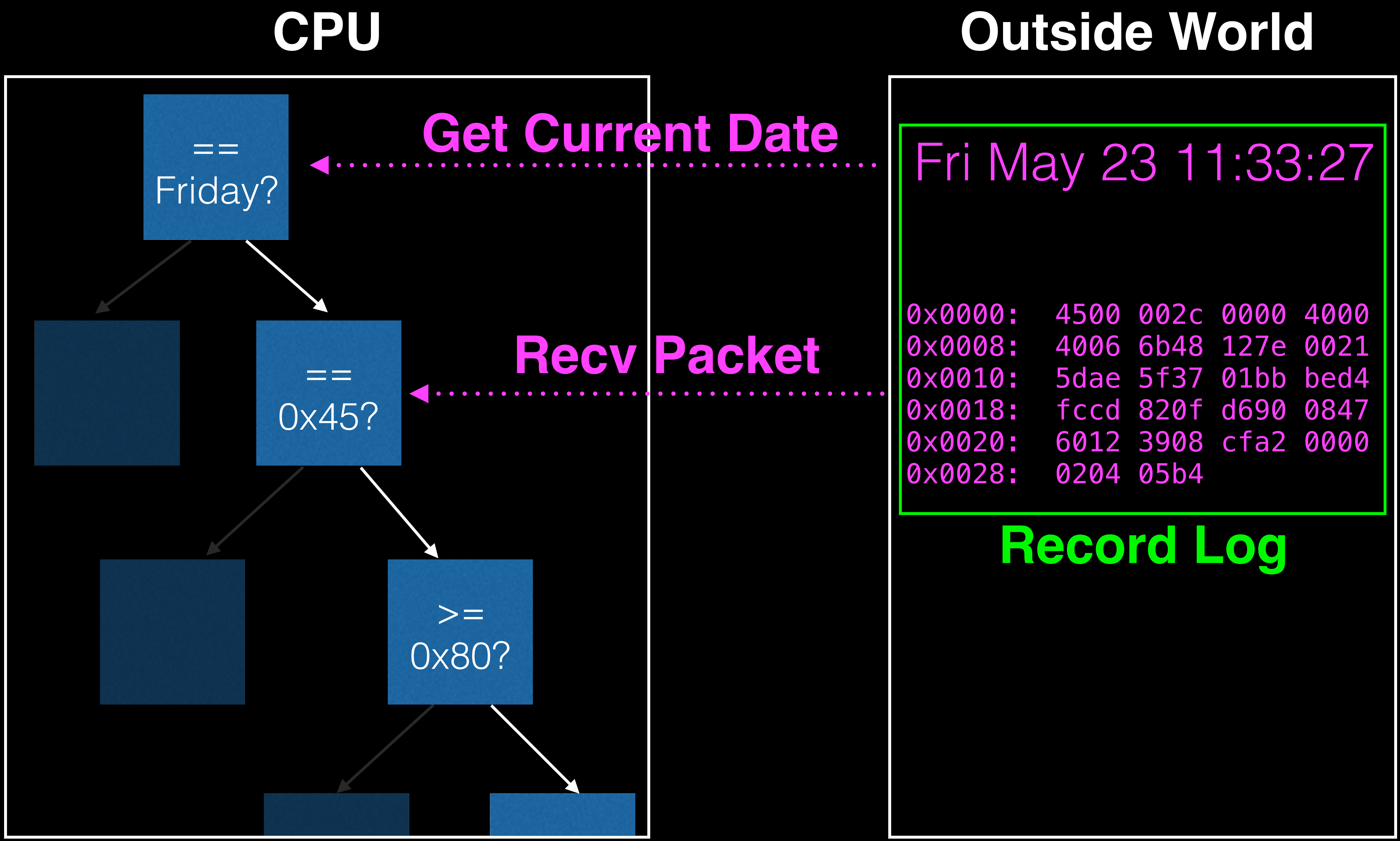
Record/Replay



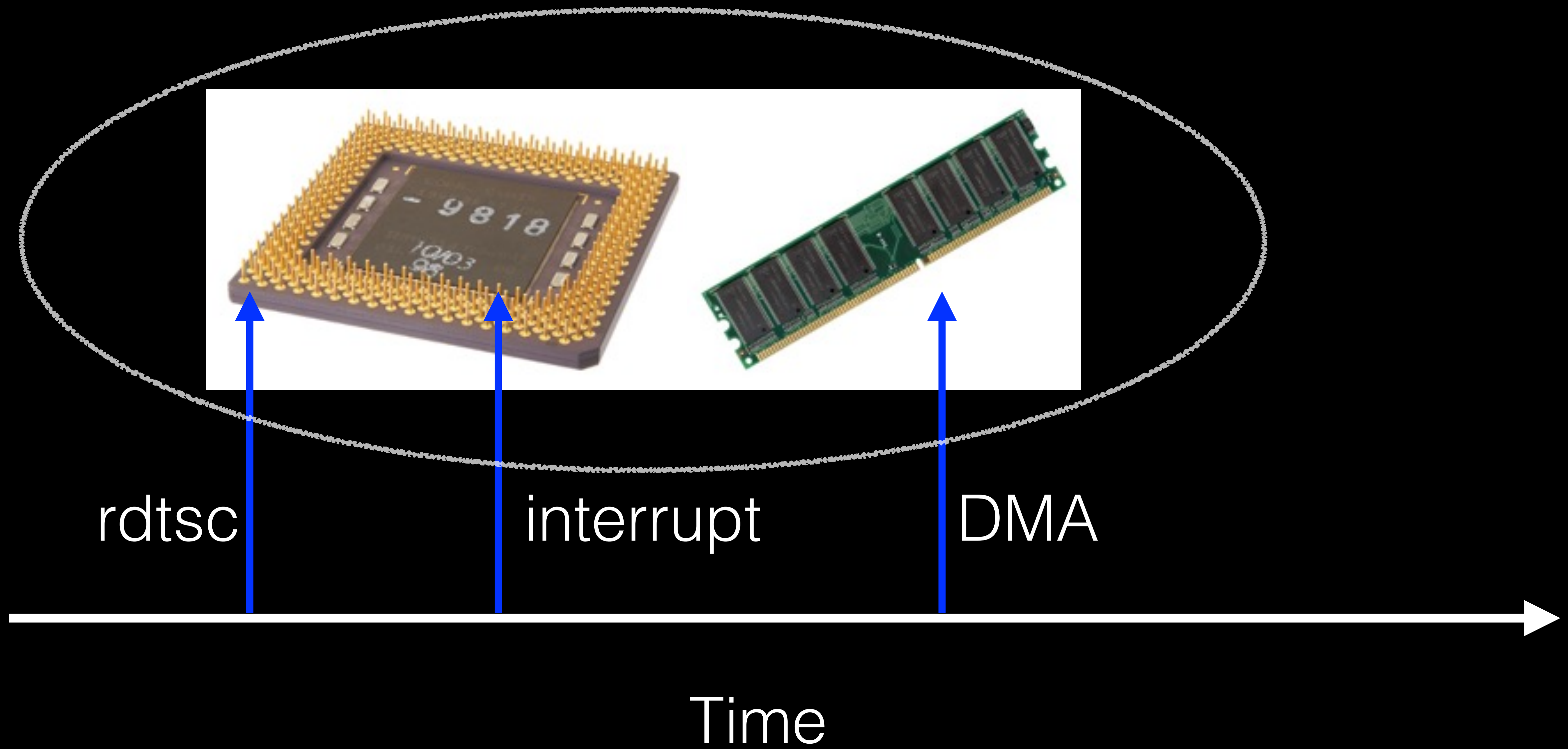
Record/Replay



Record/Replay



Record / Replay



Reproducibility via Replay

- Software execution is ephemeral
  - Environment may change
  - Timings may change
- Shareable replays make dynamic analyses **repeatable** and **reproducible**

www.rrshare.org

PANDA Share - Share PANI ×

www.rrshare.org

Logged in as moyix
Logout

PANDA SHARE

[Home] [About]

This site stores recordings made with the PANDA dynamic analysis platform. To find out more about PANDA's record/replay features, you can peruse the documentation. After downloading, the `.rr` files can be extracted using scripts/rrunpack.py in the PANDA distribution.

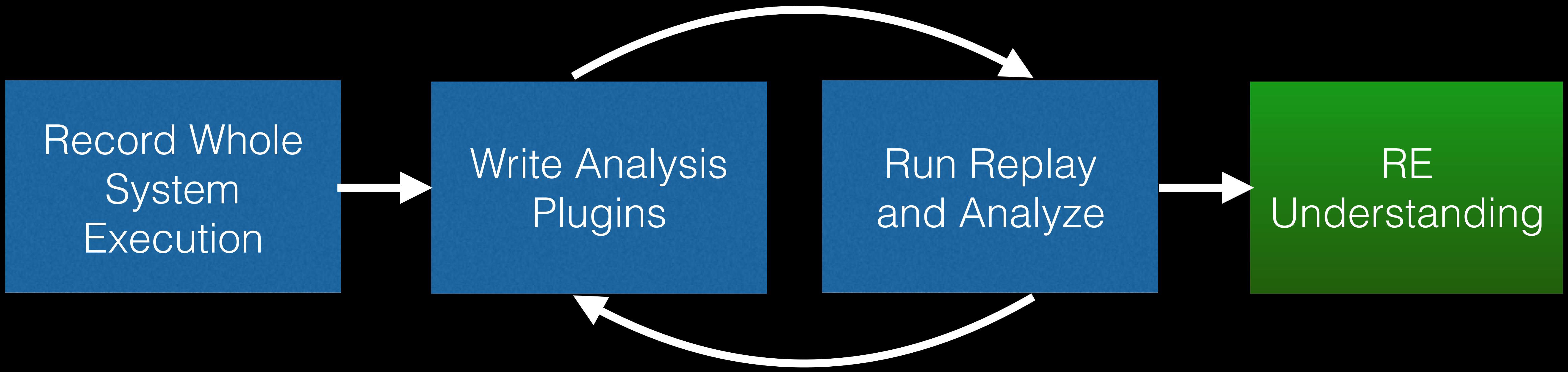
Upload a new record/replay log

Name	Summary	Download	Size	Instructions
cve-2012-4792-exploit	Exploitation of cve-2012-4792	rrlogs/cve-2012-4792-exploit.rr	130.1 MB	968.8 million
cve-2012-4792-crash	Crashing instance of cve-2012-4792	rrlogs/cve-2012-4792-crash.rr	129.9 MB	608.8 million
cve-2011-1255-exploit	Exploitation of cve-2011-1255	rrlogs/cve-2011-1255-exploit.rr	126.6 MB	2.1 billion
cve-2011-1255-crash	Crashing instance of cve-2011-1255	rrlogs/cve-2011-1255-crash.rr	127.1 MB	1.4 billion
cve-2014-1776-crash	Crashing instance of cve-2014-1776	rrlogs/cve-2014-1776-crash.rr	155.9 MB	1.2 billion
dia2dump	Parsing a PDB with dia2dump	rrlogs/dia2dump.rr	190.8 MB	5.4 billion
line2	Sending an IM using LINE for Android	rrlogs/line2.rr	64.6 MB	10.4 billion
win7_64bit_install_STOP_D1	Failure during boot to install CD of Win7 64bit. DRIVER_IRQL_NOT_LESS_OR_EQUAL	rrlogs/win7_64_install_fail.rr	203.3 MB	5.3 billion
carberp2	Running custom RU_Az build of the Carberp malware	rrlogs/carberp2.rr	91.9 MB	2.9 billion
	Running custom Full build of the Carberp			

Log Size

Replay	Instructions	Log Size	Instr/Byte
freebsdboot	9.3 billion	533 MB	17
spotify	12 billion	229 MB	52
haikuurl	8.6 billion	119 MB	72
carberp1	9.1 billion	43 MB	212
win7iessl	8.6 billion	9.4 MB	915
Starcraft	60 million	1.8 MB	33

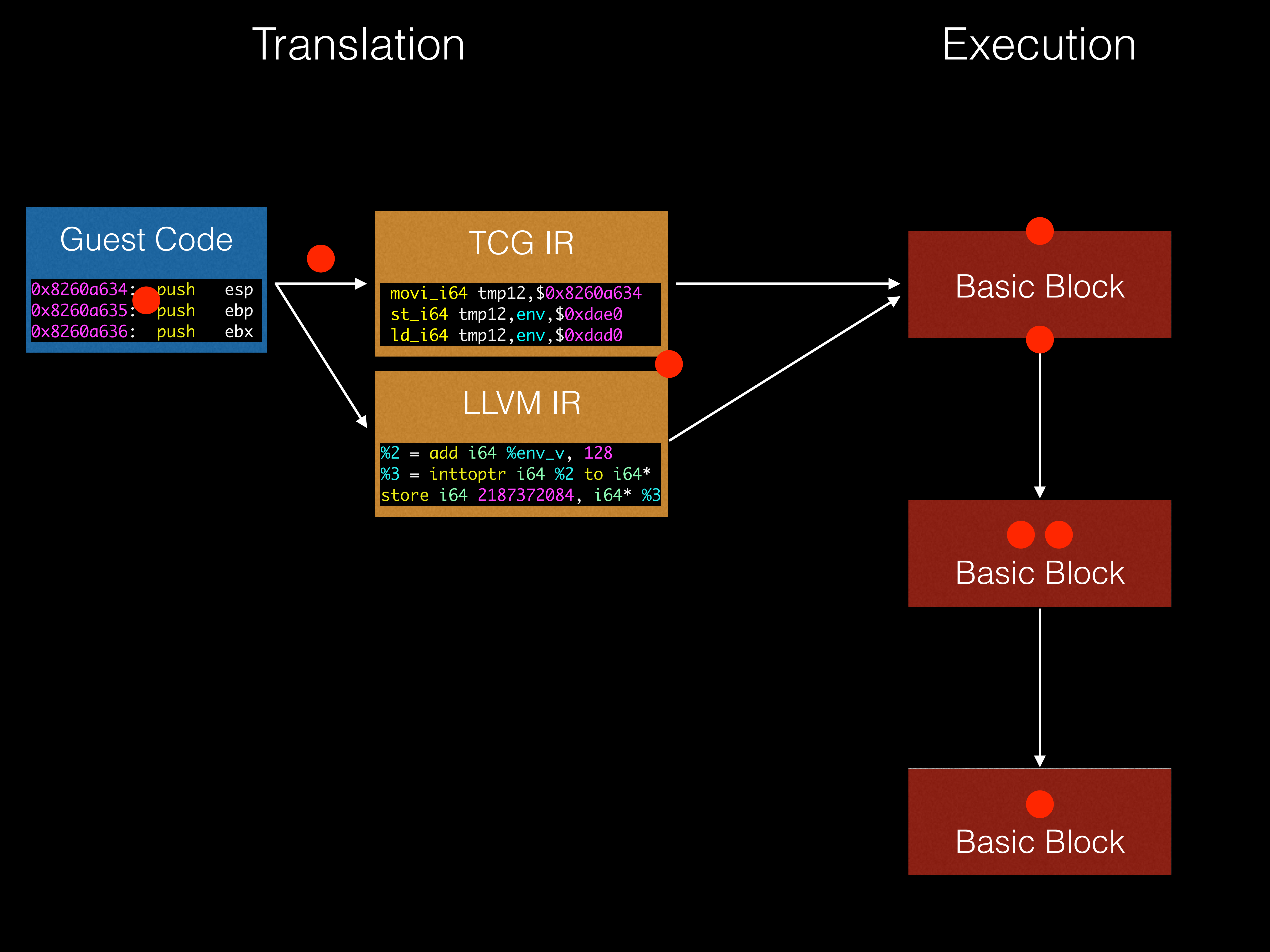
PANDA Model



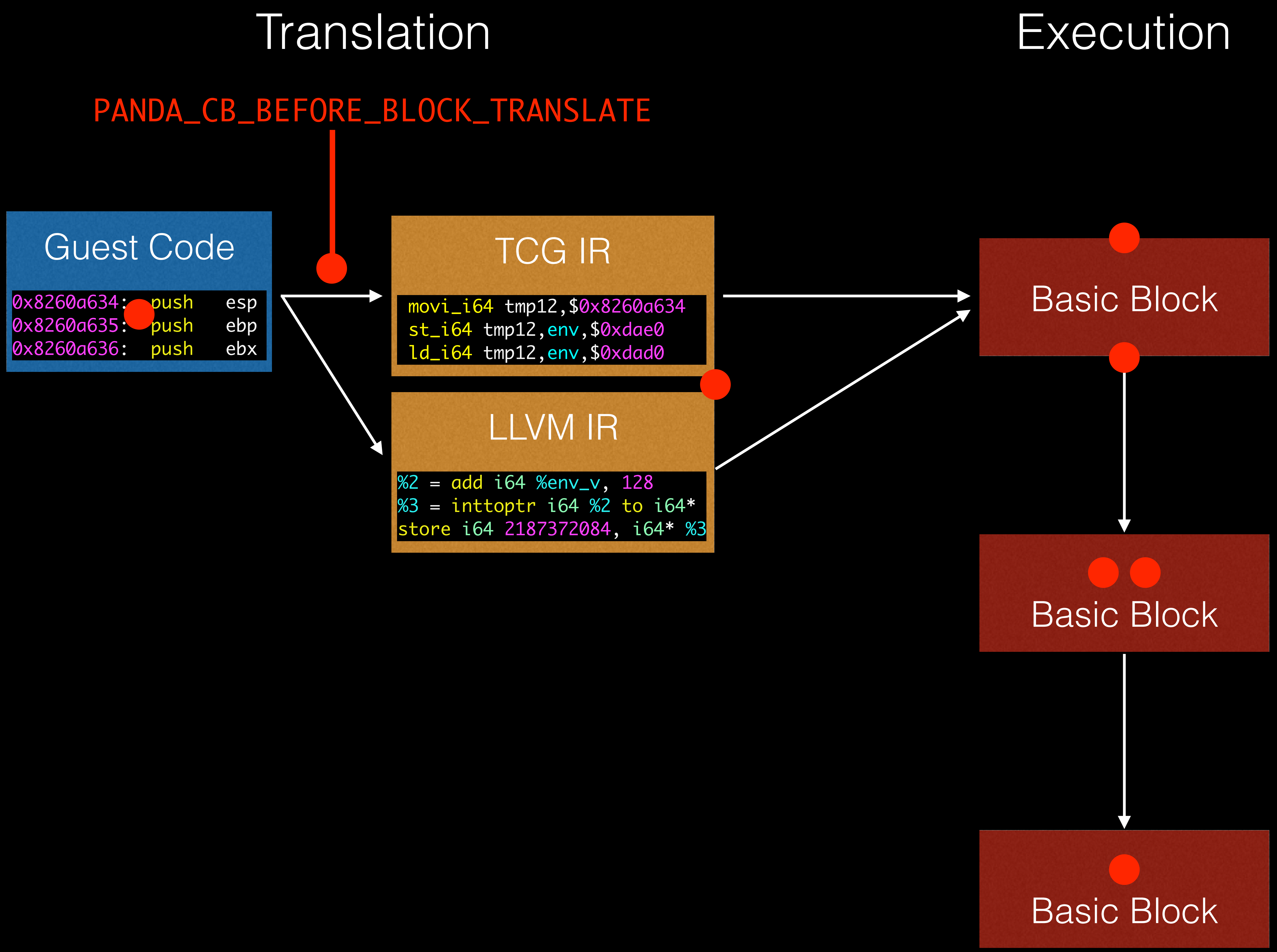
- Record / replay critical:
  - Heavy analyses don’t disrupt execution
  - Analyses don’t have to worry about memory layout changing between runs

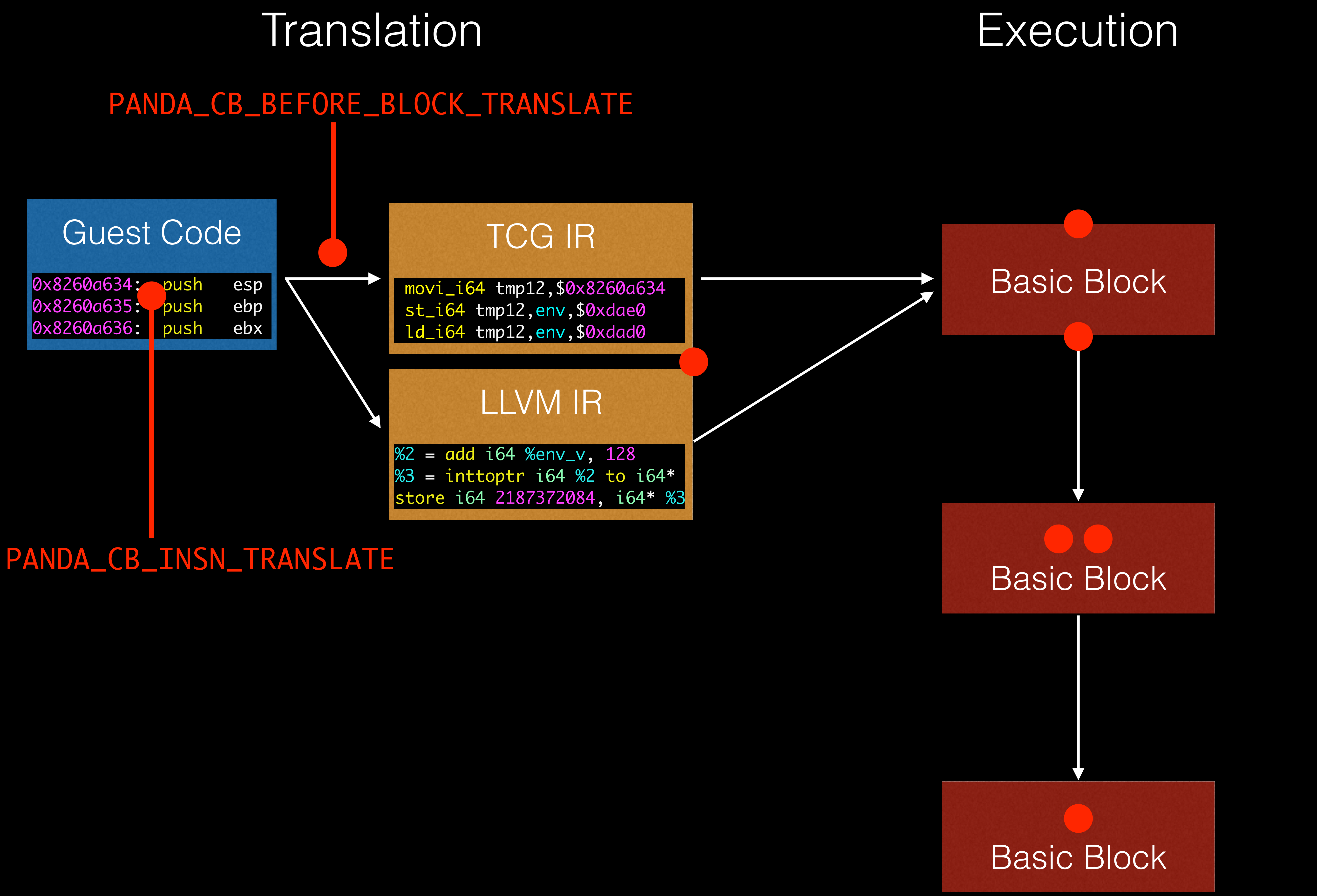
Plugin Architecture

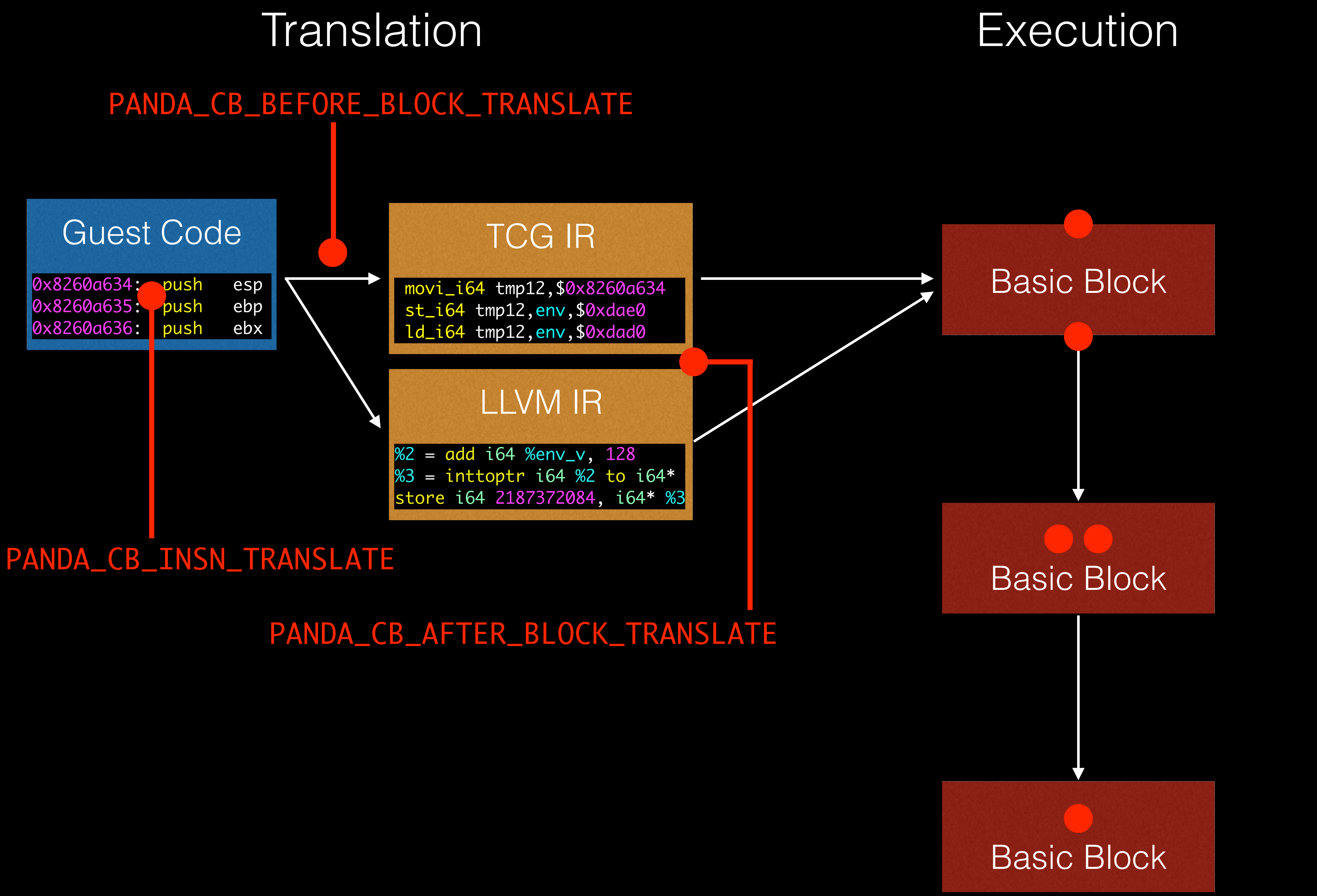
- Extend PANDA by writing plugins
- Implement functions that take action at various *instrumentation points*
- Can also instrument generated code in LLVM mode
- Plugin-plugin interaction: compose simple tools for complex functionality

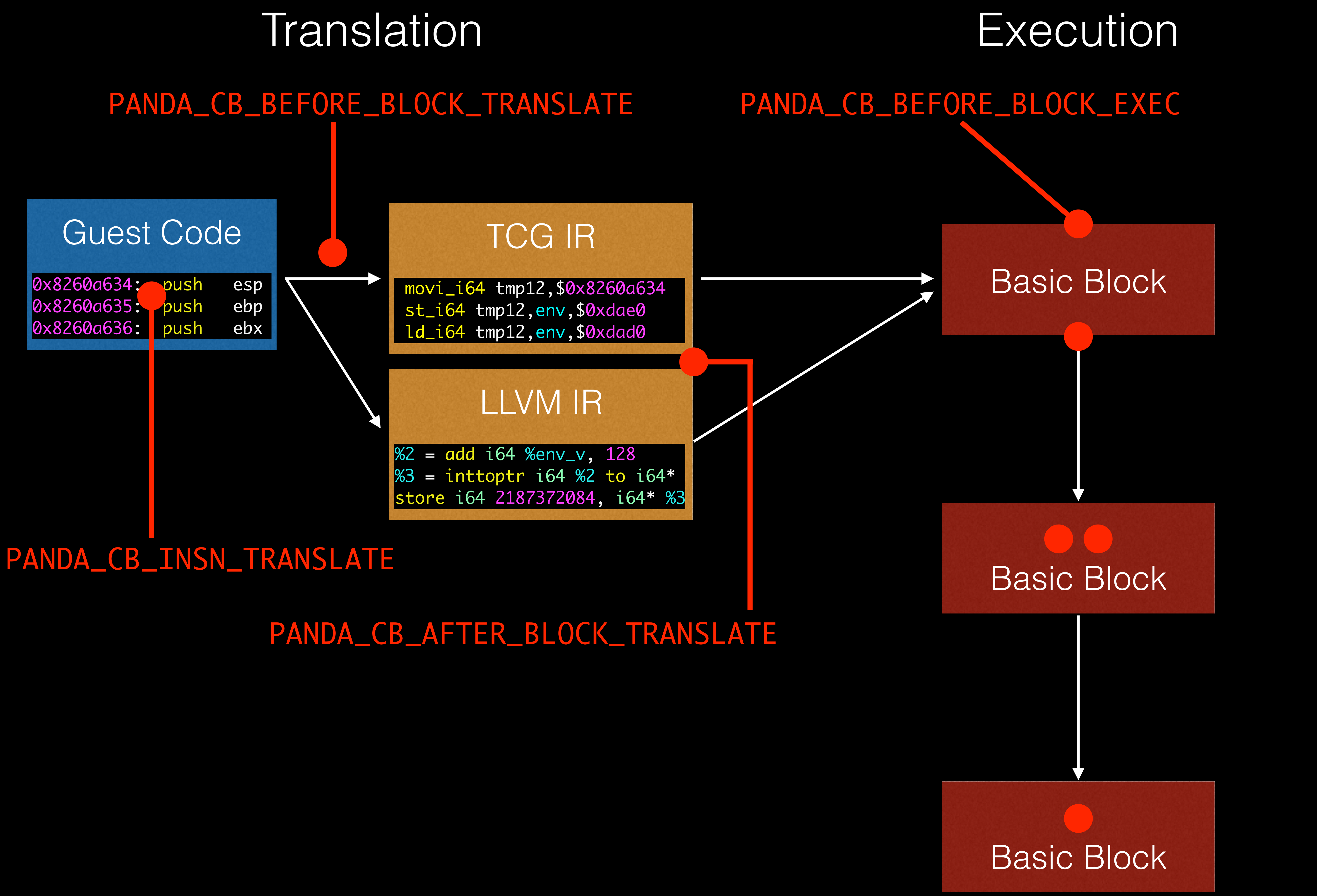


Translation
Execution
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
Basic Block
Basic Block

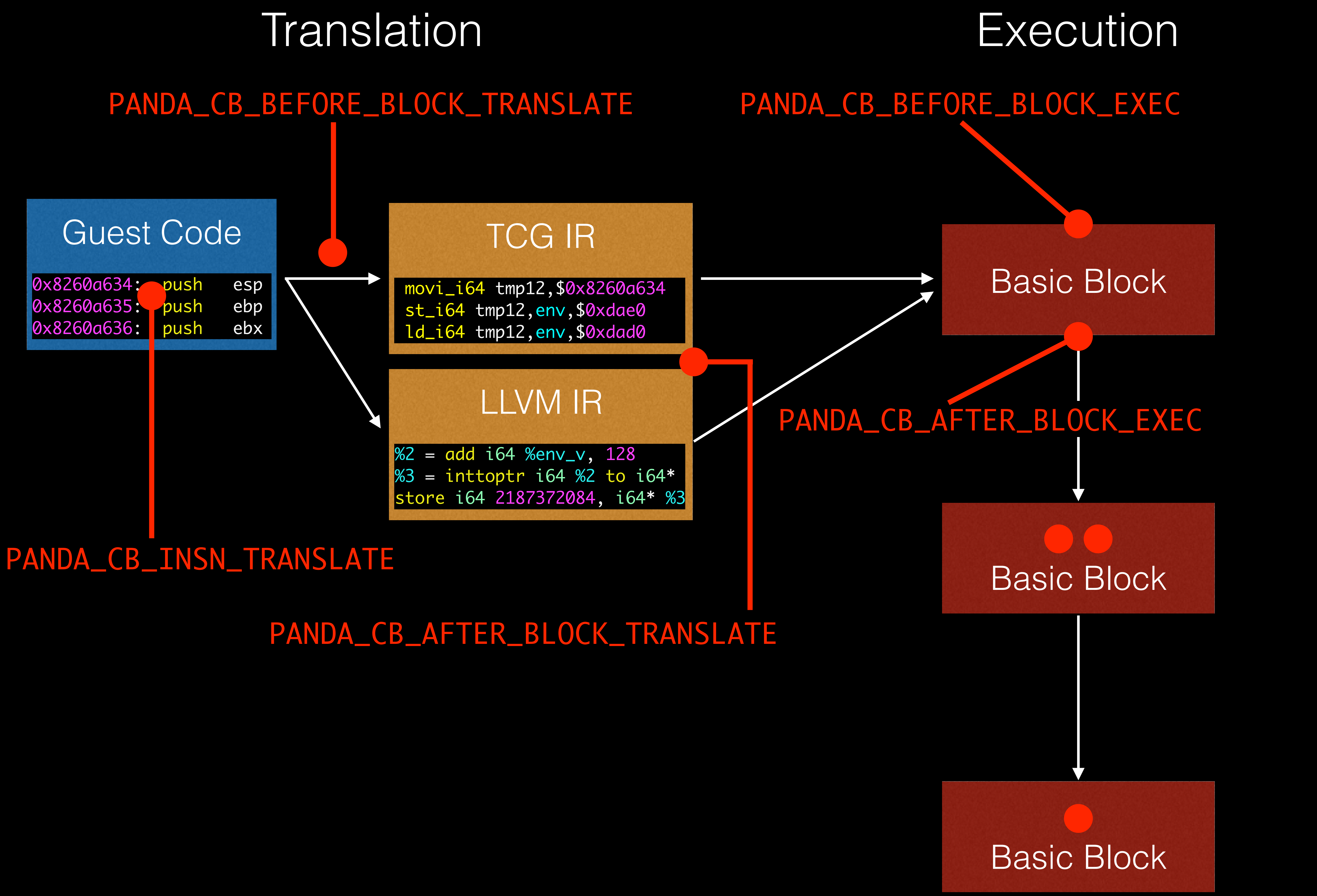

Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
Basic Block
Basic Block


Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
PANDA_CB_INSN_TRANSLATE
Basic Block
Basic Block
Basic Block

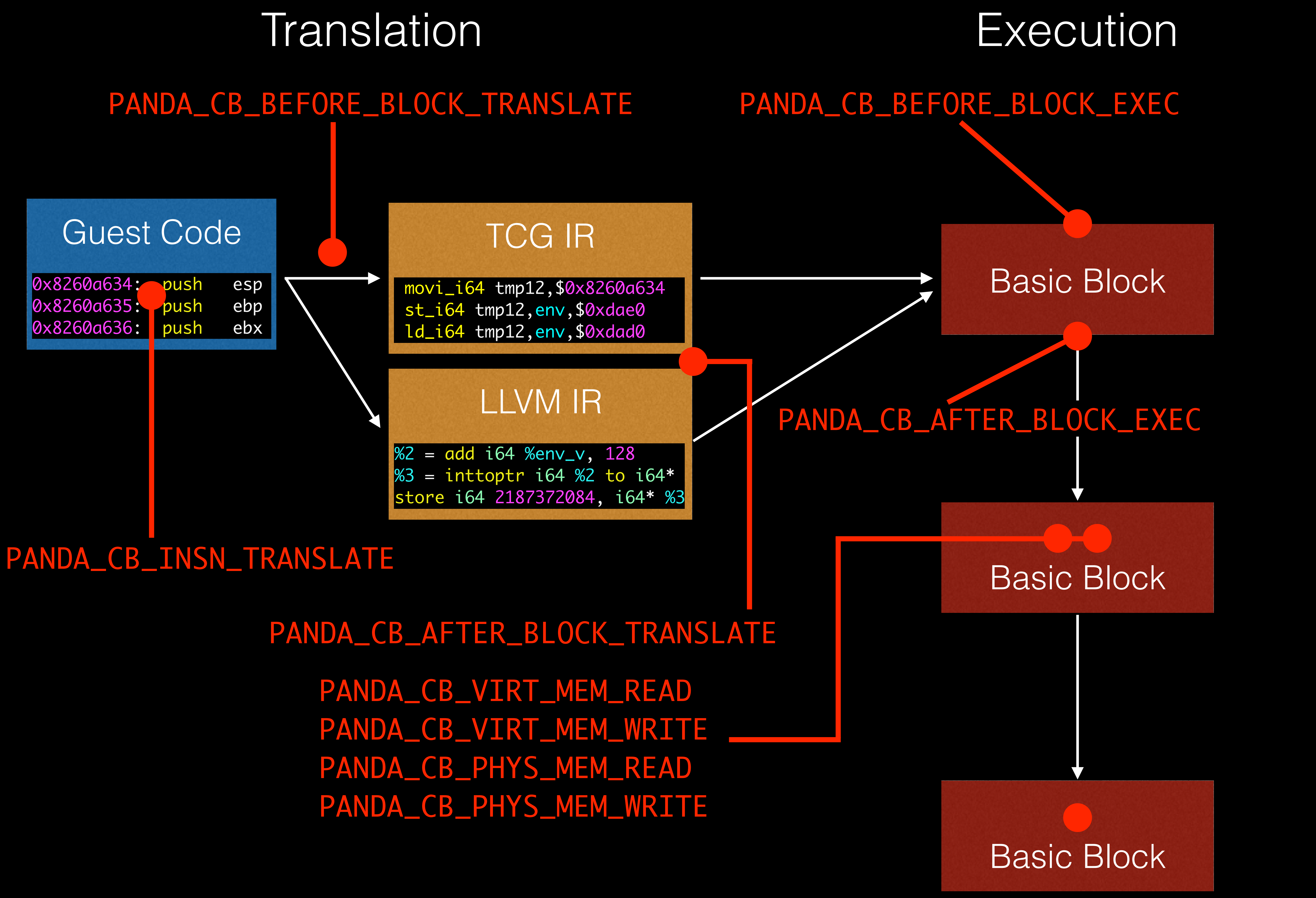

Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
PANDA_CB_INSN_TRANSLATE
PANDA_CB_AFTER_BLOCK_TRANSLATE
Basic Block
Basic Block
Basic Block



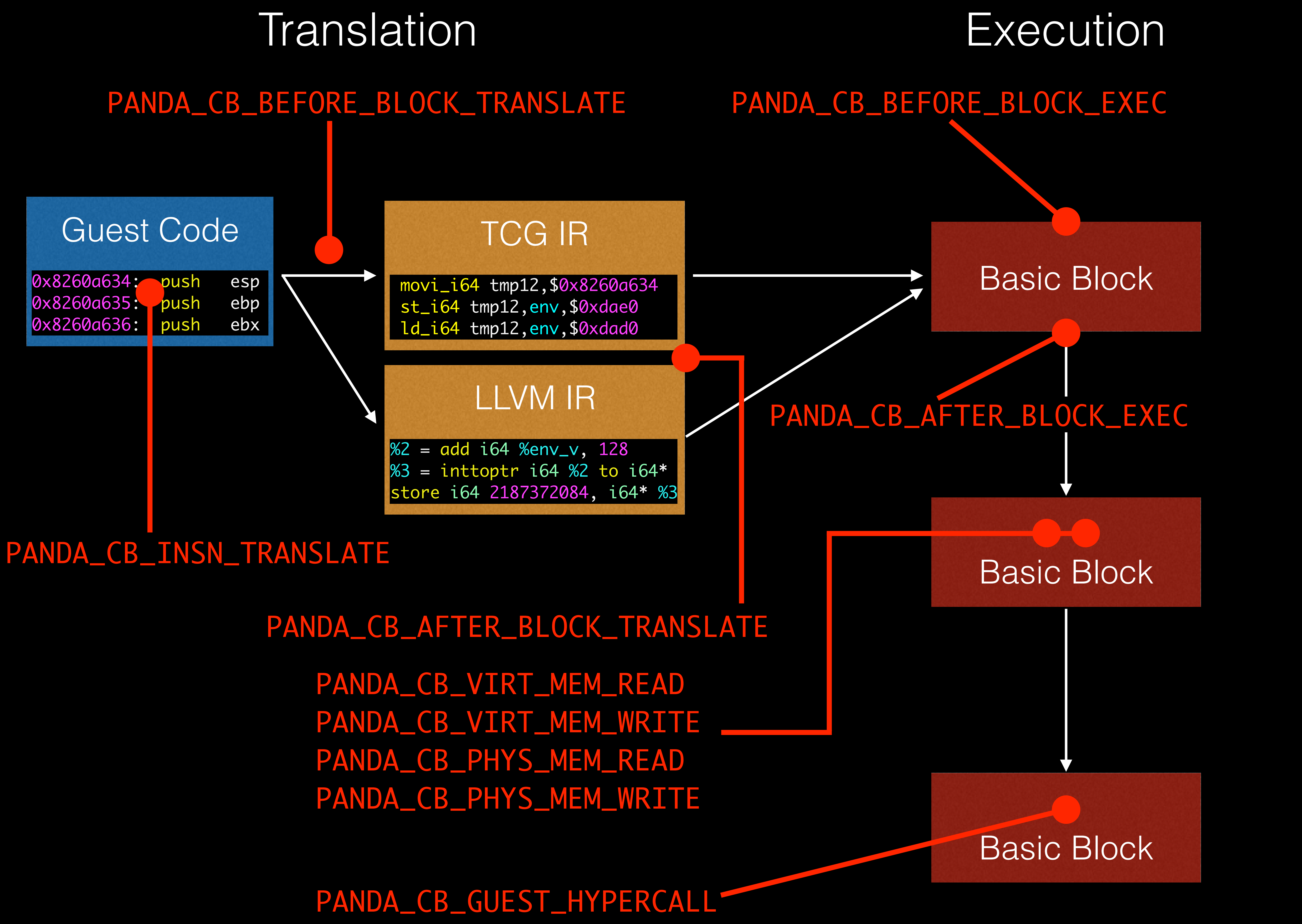
Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
PANDA_CB_BEFORE_BLOCK_EXEC
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
Basic Block
Basic Block
PANDA_CB_INSN_TRANSLATE
PANDA_CB_AFTER_BLOCK_TRANSLATE



Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
PANDA_CB_BEFORE_BLOCK_EXEC
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
PANDA_CB_AFTER_BLOCK_EXEC
Basic Block
Basic Block
PANDA_CB_INSN_TRANSLATE
PANDA_CB_AFTER_BLOCK_TRANSLATE



Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
PANDA_CB_BEFORE_BLOCK_EXEC
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
PANDA_CB_AFTER_BLOCK_EXEC
Basic Block
Basic Block
PANDA_CB_INSN_TRANSLATE
PANDA_CB_AFTER_BLOCK_TRANSLATE
PANDA_CB_VIRT_MEM_READ
PANDA_CB_VIRT_MEM_WRITE
PANDA_CB_PHYS_MEM_READ
PANDA_CB_PHYS_MEM_WRITE


Translation
Execution
PANDA_CB_BEFORE_BLOCK_TRANSLATE
PANDA_CB_BEFORE_BLOCK_EXEC
Guest Code
0x8260a634: push esp
0x8260a635: push ebp
0x8260a636: push ebx
TCG IR
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
LLVM IR
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
Basic Block
PANDA_CB_AFTER_BLOCK_EXEC
Basic Block
PANDA_CB_INSN_TRANSLATE
PANDA_CB_AFTER_BLOCK_TRANSLATE
PANDA_CB_VIRT_MEM_READ
PANDA_CB_VIRT_MEM_WRITE
PANDA_CB_PHYS_MEM_READ
PANDA_CB_PHYS_MEM_WRITE
Basic Block
PANDA_CB_GUEST_HYPERCALL

LLVM Translation

```
0x8260a634:  push   esp
0x8260a635:  push   ebp
0x8260a636:  push   ebx
0x8260a637:  push   esi
0x8260a638:  push   edi
0x8260a639:  sub    esp,0x54
0x8260a63c:  mov    ebp,esp
0x8260a63e:  mov    DWORD PTR [ebp+0x44],eax
0x8260a641:  mov    DWORD PTR [ebp+0x40],ecx
0x8260a644:  mov    DWORD PTR [ebp+0x3c],edx
0x8260a647:  test   DWORD PTR [ebp+0x70],0x20000
0x8260a64e:  jne    0x8260a60c
```

LLVM Translation

```
movi_i64 tmp4,$0x8260a634
st_i64 tmp4,env,$0x80
---- 0x8260a634
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
movi_i64 tmp13,$0x1
add_i64 tmp12,tmp12,tmp13
st_i64 tmp12,env,$0xdad0
mov_i64 tmp0,rsp
mov_i64 tmp2,rsp
movi_i64 tmp12,$0xfffffffffffffffc
add_i64 tmp2,tmp2,tmp12
movi_i64 tmp12,$0xffffffff
and_i64 tmp2,tmp2,tmp12
```

[...]

LLVM Translation

```
define private i64 @tcg-llvm-tb-0-8260a634(i64*) {
entry:
  %1 = getelementptr i64* %0, i32 0
  %env_v = load i64* %1
  %2 = add i64 %env_v, 128
  %3 = inttoptr i64 %2 to i64*
  store i64 2187372084, i64* %3
  store volatile i64 2, i64* inttoptr
      (i64 29543856 to i64*)
  store volatile i64 2187372084, i64* inttoptr
      (i64 29543864 to i64*)
  %4 = add i64 %env_v, 56032
  %5 = inttoptr i64 %4 to i64*
  store i64 2187372084, i64* %5
  %6 = add i64 %env_v, 56016
```

[...]

Android Emulation

- Supports Android 2.x – 4.x
- Can make phone calls, send SMS, run native apps
- Record/replay
- Introspection into Android apps (Dalvik-level) for Android 2.3 (from DroidScope)
- System-level introspection supported on all Android versions

Mining Memory Accesses

- Goal: Find places in system where data of interest (e.g., ssh passphrase) is handled
- Idea: watch every memory access in the system and look for patterns
- Call these points of interest – which we can hook – ***tap points***

More details: *Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection*. B. Dolan-Gavitt, T. Leek, J. Hodosh, W. Lee. ACM CCS. Berlin, Germany, November 2013.

TZB Implementation

- Track calling context with *callstack* plugin
- At every memory access (PANDA_CB_PHYS_MEM_READ/WRITE)
  Get (caller, program counter, address space) – i.e., *tap point*
- Analyze data flowing through tap point (e.g., string matching with *stringsearch* plugin)

Sample Tap Points

Content

Read Write

Tap

Code

```
00646517 0064A423 Kernel   0064A423 push    ebx
00646517 0064A424 Kernel   0064A424 push    [ebp+var_28]
00646517 0064A427 Kernel   0064A427 push    esi
00646517 0064A428 Kernel   0064A428 call    _memcpy

                           _memcpy:
                           [...]
                           00430E08 shr     ecx, 2
                           00430E0B and     edx, 3
                           00430E0E cmp     ecx, 8
                           00430E11 jb      short loc_430E3C
0064A42D 00430E13 Kernel   00430E13 rep movsd
0064A42D 00430E15 Kernel   00430E15 jmp     off_430F2C[edx*4]
```

Sample Tap Points

Content

Read	Write	Tap	Code
	00FFABED	00646517 0064A423 Kernel	0064A423 push ebx
		00646517 0064A424 Kernel	0064A424 push [ebp+var_28]
		00646517 0064A427 Kernel	0064A427 push esi
		00646517 0064A428 Kernel	0064A428 call _memcpy
			_memcpy:
			[...]
			00430E08 shr ecx, 2
			00430E0B and edx, 3
			00430E0E cmp ecx, 8
			00430E11 jb short loc_430E3C
		0064A42D 00430E13 Kernel	00430E13 rep movsd
		0064A42D 00430E15 Kernel	00430E15 jmp off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap			Code		
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
		00646517	0064A427	Kernel	0064A427	push	esi
		00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
		0064A42D	00430E13	Kernel	00430E13	rep movsd	
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap	Code
	00FFABED	00646517 0064A423 Kernel	0064A423 push ebx
00123456	00123456	00646517 0064A424 Kernel	0064A424 push [ebp+var_28]
	00ABCDEF	00646517 0064A427 Kernel	0064A427 push esi
		00646517 0064A428 Kernel	0064A428 call _memcpy
			_memcpy:
			[...]
			00430E08 shr ecx, 2
			00430E0B and edx, 3
			00430E0E cmp ecx, 8
			00430E11 jb short loc_430E3C
		0064A42D 00430E13 Kernel	00430E13 rep movsd
		0064A42D 00430E15 Kernel	00430E15 jmp off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap	Code
	00FFABED	00646517 0064A423 Kernel	0064A423 push ebx
00123456	00123456	00646517 0064A424 Kernel	0064A424 push [ebp+var_28]
	00ABCDEF	00646517 0064A427 Kernel	0064A427 push esi
	0064A42D	00646517 0064A428 Kernel	0064A428 call _memcpy
			_memcpy:
			[...]
			00430E08 shr ecx, 2
			00430E0B and edx, 3
			00430E0E cmp ecx, 8
			00430E11 jb short loc_430E3C
		0064A42D 00430E13 Kernel	00430E13 rep movsd
		0064A42D 00430E15 Kernel	00430E15 jmp off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap			Code		
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Dev	\Dev	0064A42D	00430E13	Kernel	00430E13	rep movsd	
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap			Code		
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\	\Device\	0064A42D	00430E13	Kernel	00430E13	rep movsd	
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap	Code
	00FFABED	00646517 0064A423 Kernel	0064A423 push ebx
00123456	00123456	00646517 0064A424 Kernel	0064A424 push [ebp+var_28]
	00ABCDEF	00646517 0064A427 Kernel	0064A427 push esi
	0064A42D	00646517 0064A428 Kernel	0064A428 call _memcpy
			_memcpy:
			[...]
			00430E08 shr ecx, 2
			00430E0B and edx, 3
			00430E0E cmp ecx, 8
			00430E11 jb short loc_430E3C
\Device\Hard	\Device\Hard	0064A42D 00430E13 Kernel	00430E13 rep movsd
		0064A42D 00430E15 Kernel	00430E15 jmp off_430F2C[edx*4]

Sample Tap Points

Content: Read	Content: Write	Tap			Code		
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\Harddisk	\Device\Harddisk	0064A42D	00430E13	Kernel	00430E13	rep movsd	
		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Sample Tap Points

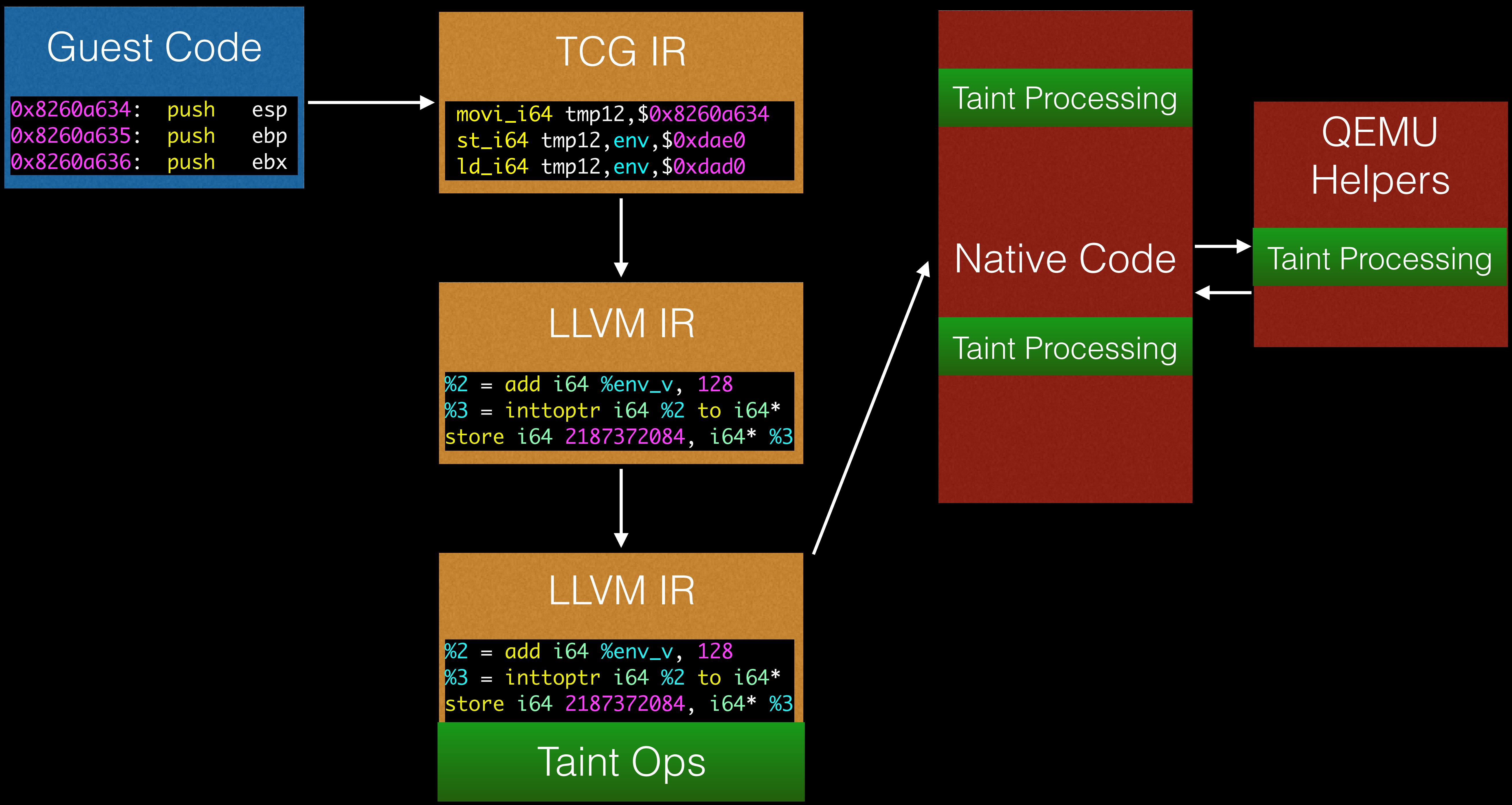
Content: Read	Content: Write	Tap			Code		
	00FFABED	00646517	0064A423	Kernel	0064A423	push	ebx
00123456	00123456	00646517	0064A424	Kernel	0064A424	push	[ebp+var_28]
	00ABCDEF	00646517	0064A427	Kernel	0064A427	push	esi
	0064A42D	00646517	0064A428	Kernel	0064A428	call	_memcpy
					_memcpy:		
					[...]		
					00430E08	shr	ecx, 2
					00430E0B	and	edx, 3
					00430E0E	cmp	ecx, 8
					00430E11	jb	short loc_430E3C
\Device\Harddisk	\Device\Harddisk	0064A42D	00430E13	Kernel	00430E13	rep movsd	
00430F3C		0064A42D	00430E15	Kernel	00430E15	jmp	off_430F2C[edx*4]

Dynamic Taint Analysis

- Follows data flow between *taint source* and *sink*
- Implemented in PANDA as an LLVM pass
  - Allows taint tracking on *all* platforms
  - Can use clang to produce LLVM bitcode for QEMU's C functions and track taint through

More details: *Architecture-Independent Dynamic Information Flow Tracking*. R. Whelan, T. Leek, D. Kaeli. Compiler Construction (CC), Rome, Italy, March 2013.

LLVM Taint2 Instrumentation



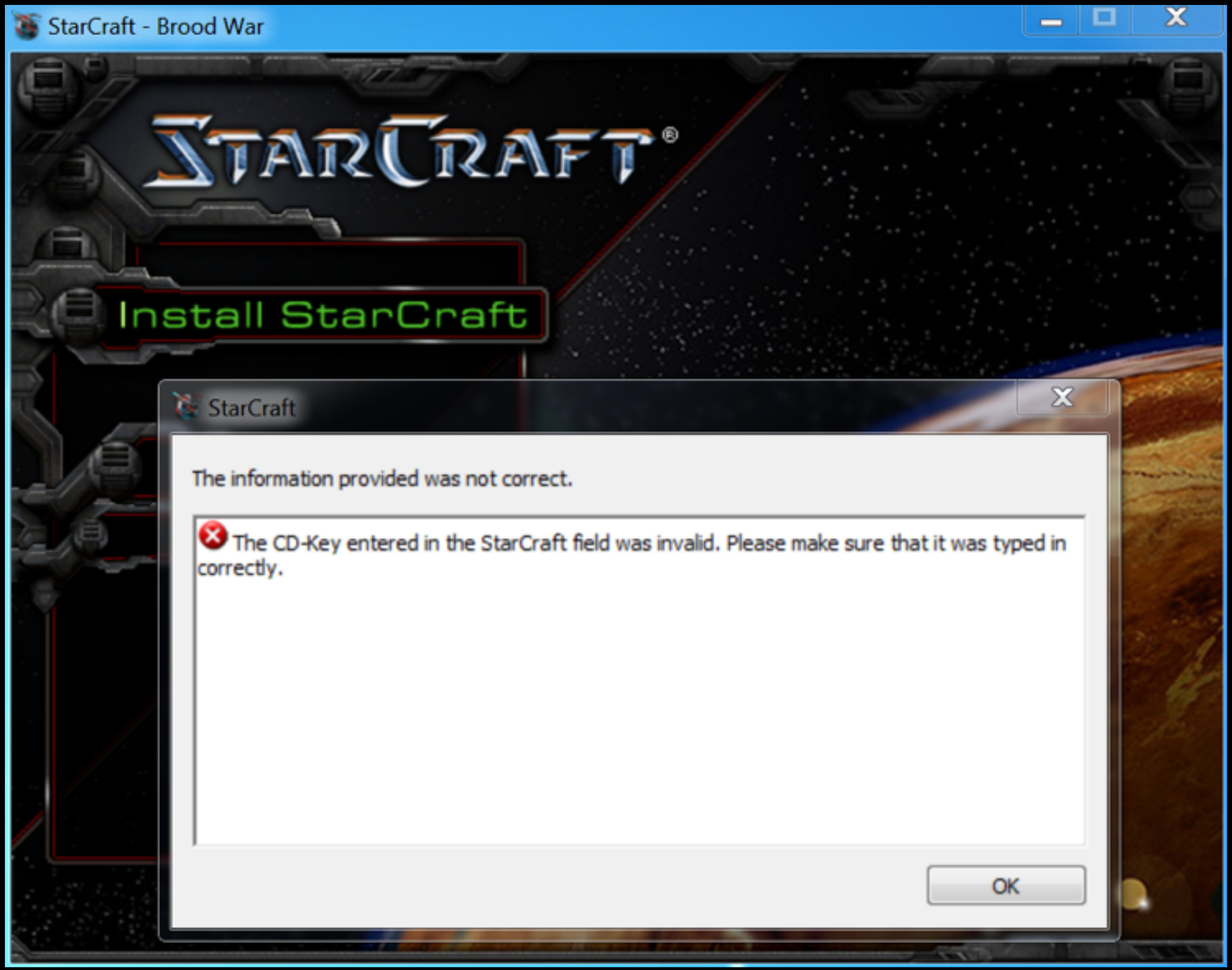
Other Notable Plugins

- *scissors*: extracts out a subset of a replay log
- *replaymovie*: takes frame buffer snapshots during replay and creates a movie
- *syscalls2*: provides callbacks for Linux & Windows system calls and their arguments
- *osi*: OS introspection for Windows 7 & Linux
- *file_taint*, *tstringsearch*: taint labeling based on file contents or in-memory string matches

Case Studies

- Reverse engineering the Starcraft CD key check
- Breaking Spotify DRM
- Understanding a vulnerability in Internet Explorer
- Full-trace malware analysis sandbox
- Automated bug injection

Starcraft CD Key



Creating a Keygen

- Simple approach:
  - Find the key validation code
  - Extract it so you can run it millions of times
  - Feed it random keys until you get a valid one
  - Works when the key space is *dense* – many valid keys relative to total keys

Starcraft RE

- Use TZB to search for code that uses CD key:

Caller 5	Caller 4	Caller 3	Caller 2	Caller 1	PC	CR3	
		0045c252	00428867	004286ff	0044c951	06cba000	1
0045c252	00428867	004286ff	0044c83b	0047d949	0047d4cb	06cba000	1

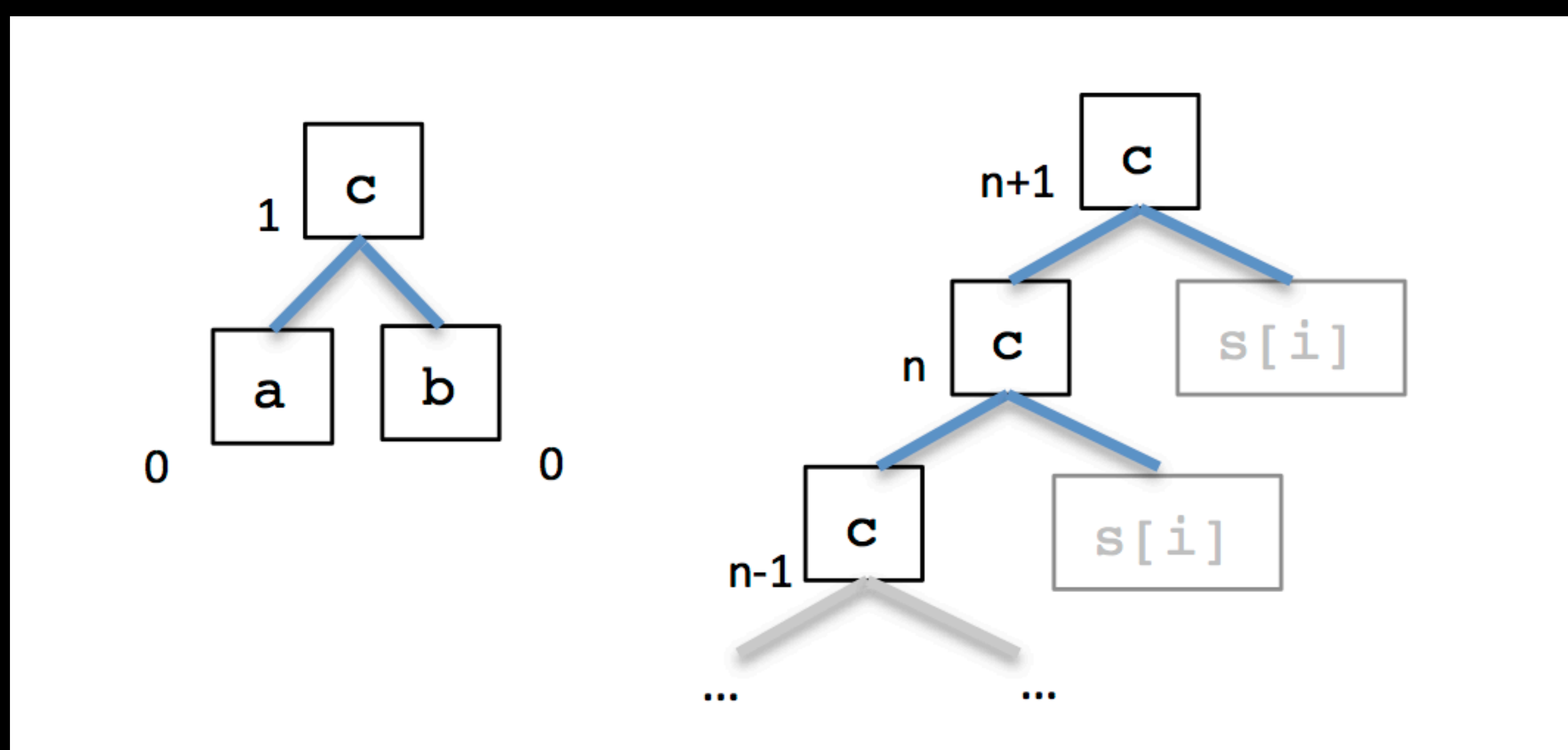
[…]

- Or, taint key and measure computation done on tainted data
  - i.e.: $a = b + c$

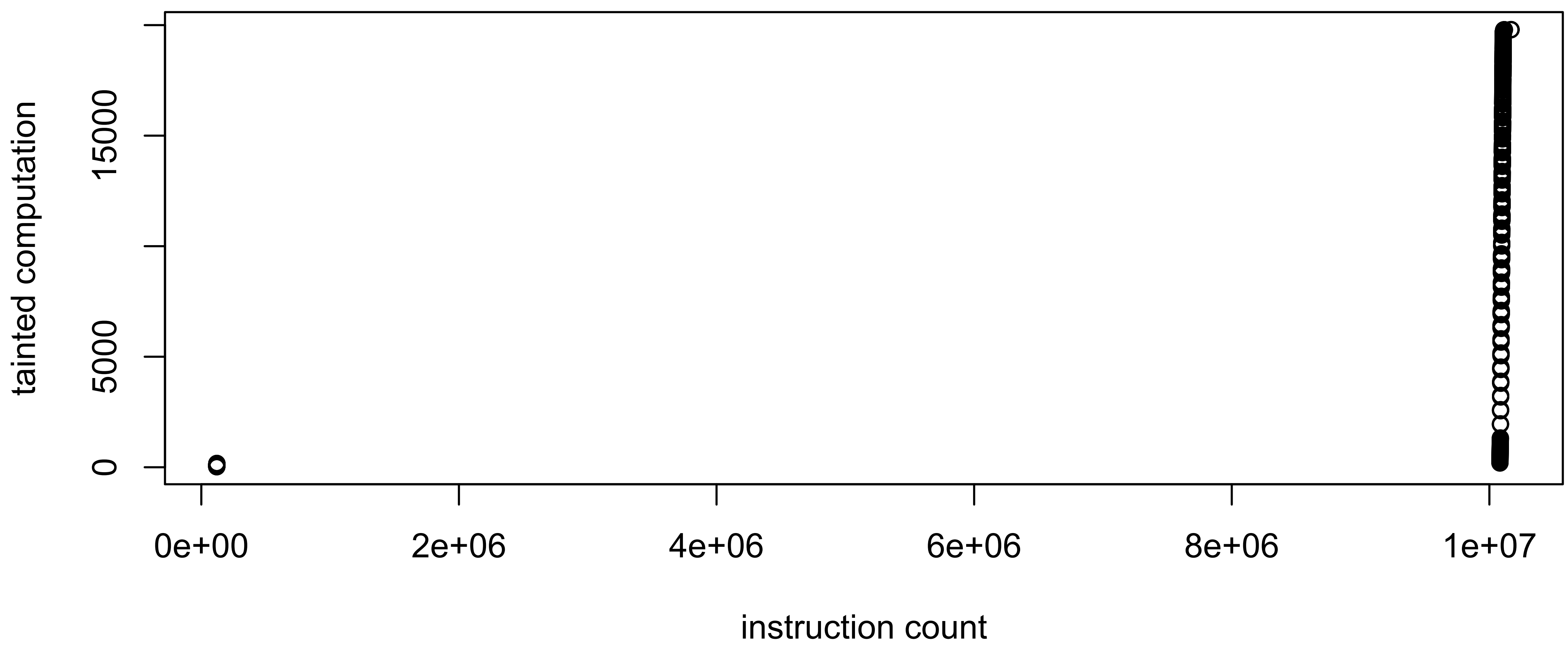
    $tcn(a) = max(tcn(a), tcn(b)) + 1$

Taint Compute Number

```
int c = a+b;
if (a != 0xdeadbeef)
      return;
for (int i=0; i<n; i++)
      c+=s[i];
```



Starcraft
Tainted Computation



Key Load

```
.text:0047D4A0 loc_47D4A0:                                ; CODE XREF: unpack_key+53↓j
.text:0047D4A0                 xor     edx, edx
.text:0047D4A2                 lea     eax, [esi+7B5h]
.text:0047D4A8                 mov     ecx, 34h
.text:0047D4AD                 div     ecx
.text:0047D4AF                 mov     esi, 34h
.text:0047D4B4                 mov     ebp, 5
.text:0047D4B9                 mov     ecx, edx
.text:0047D4BB                 xor     edx, edx
.text:0047D4BD                 lea     eax, [ecx+7B5h]
.text:0047D4C3                 div     esi
.text:0047D4C5                 mov     esi, edx
.text:0047D4C7                 mov     edx, [esp+10h+arg_0]
.text:0047D4CB                 movzx   eax, byte ptr [edi+edx]
.text:0047D4CF                 movzx   eax, ds:byte_51EA70[eax]
.text:0047D4D6                 cdq
.text:0047D4D7                 idiv    ebp
.text:0047D4D9                 inc     edi
.text:0047D4DA                 cmp     edi, 1Ah
.text:0047D4DD                 mov     [ecx+ebx], al
.text:0047D4E0                 mov     [esi+ebx], dl
.text:0047D4E3                 jb      short loc_47D4A0
.text:0047D4E5                 pop     edi
.text:0047D4E6                 pop     esi
.text:0047D4E7                 pop     ebp
.text:0047D4E8                 pop     ebx
.text:0047D4E9                 retn
.text:0047D4E9 unpack_key      endp
```

Key Validation

```
.text:0044C82C                 lea     ecx, [esp+104h+var_EC]
.text:0044C830                 push    ecx             ; int
.text:0044C831                 push    edi             ; key
.text:0044C832                 mov     [esp+10Ch+var_EC], ebx
.text:0044C836                 call    decrypt_key     ; decrypt_key(k(@9cb68c) = N68KTDHEKMHEV89N74GKEDNYKD,
.text:0044C83B                 mov     edx, [esp+10Ch+var_EC]
.text:0044C83F                 add     esp, 10h
.text:0044C842                 push    edx
.text:0044C843                 mov     ecx, esi
.text:0044C845                 call    test_key
.text:0044C84A                 test    al, al
.text:0044C84C                 jnz     loc_44C94C      ; jumptable 0044C6EA default case
.text:0044C852                 cmp     dword ptr [esi+70h], 4
```

Key Comparison

```
.text:0044C120 test_key        proc near               ; CODE XREF: sub_44C6B0+11B↓p
.text:0044C120                                         ; sub_44C6B0+195↓p
.text:0044C120
.text:0044C120 arg_0           = dword ptr  4
.text:0044C120
.text:0044C120                 mov     edx, [ecx+68h]
.text:0044C123                 mov     eax, [ecx+64h]
.text:0044C126                 cmp     eax, edx
.text:0044C128                 jz      short loc_44C13B
.text:0044C12A                 mov     ecx, [esp+arg_0]
.text:0044C12E                 mov     edi, edi
.text:0044C130
.text:0044C130 loc_44C130:                             ; CODE XREF: test_key+19↓j
.text:0044C130                 cmp     [eax], ecx
.text:0044C132                 jz      short loc_44C13B
.text:0044C134                 add     eax, 4
.text:0044C137                 cmp     eax, edx
.text:0044C139                 jnz     short loc_44C130
.text:0044C13B
.text:0044C13B loc_44C13B:                             ; CODE XREF: test_key+8↑j
.text:0044C13B                                         ; test_key+12↑j
.text:0044C13B                 xor     ecx, ecx
.text:0044C13D                 cmp     eax, edx
.text:0044C13F                 setnz   cl
.text:0044C142                 mov     al, cl
.text:0044C144                 retn    4
.text:0044C144 test_key        endp
```

Key Valid Test

```
.text:0044C130:        cmp       [eax], ecx
```

Panda Plugin

```
bool translate_callback(CPUState *env, target_ulong pc) {
    return env->cr[3] == 0x06cba000 && pc == 0x0044C130;
}

int exec_callback(CPUState *env, target_ulong pc) {
    printf("Inside test_key: \n");

    target_ulong x = 0;
    panda_virtual_memory_rw(env, EAX, (uint8_t *)&x, 4, 0);

    printf("  Expected=" TARGET_FMT_lx " calculated="
        TARGET_FMT_lx "\n", x, ECX);
    return 1;
}
```

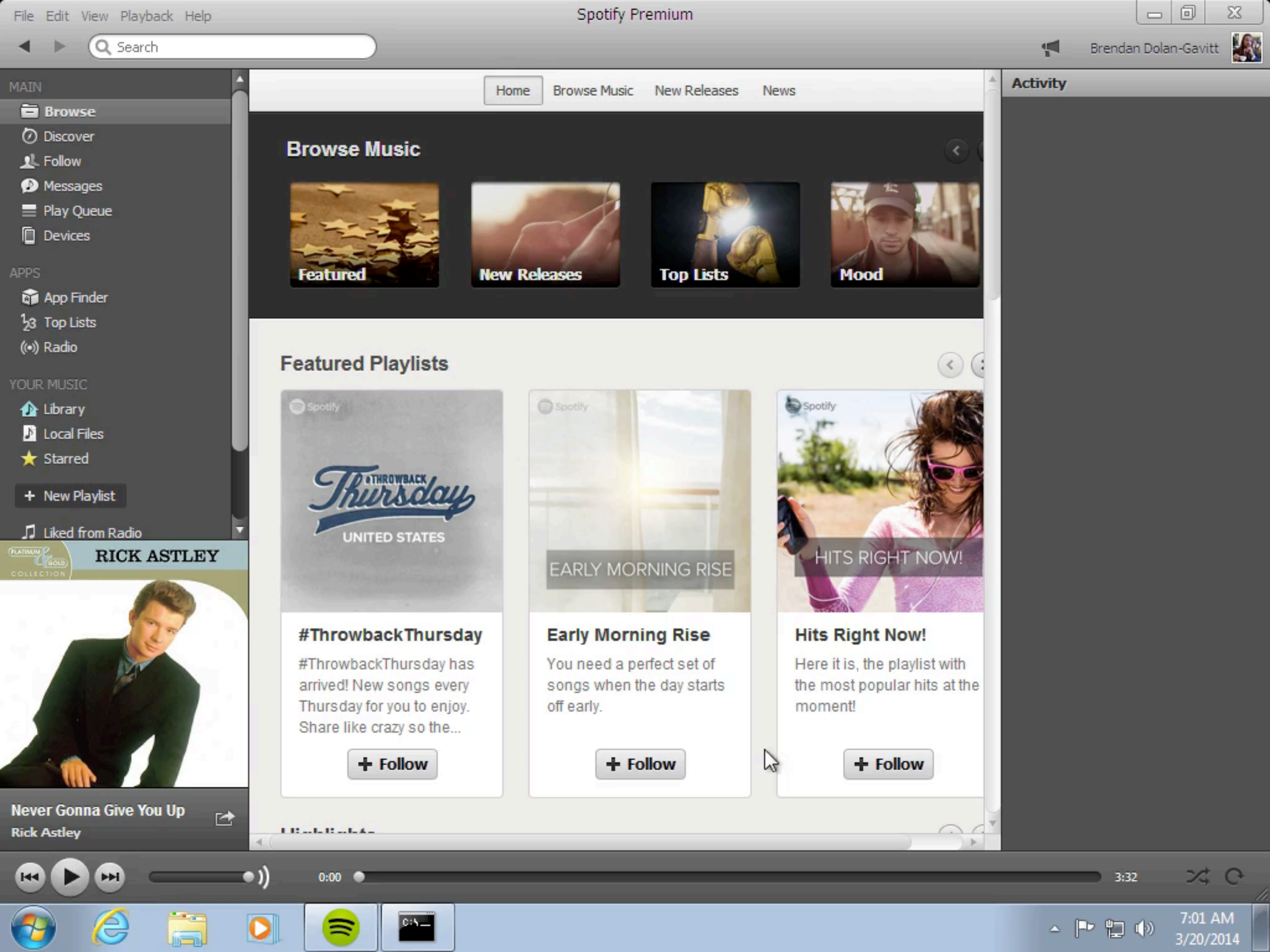
Output

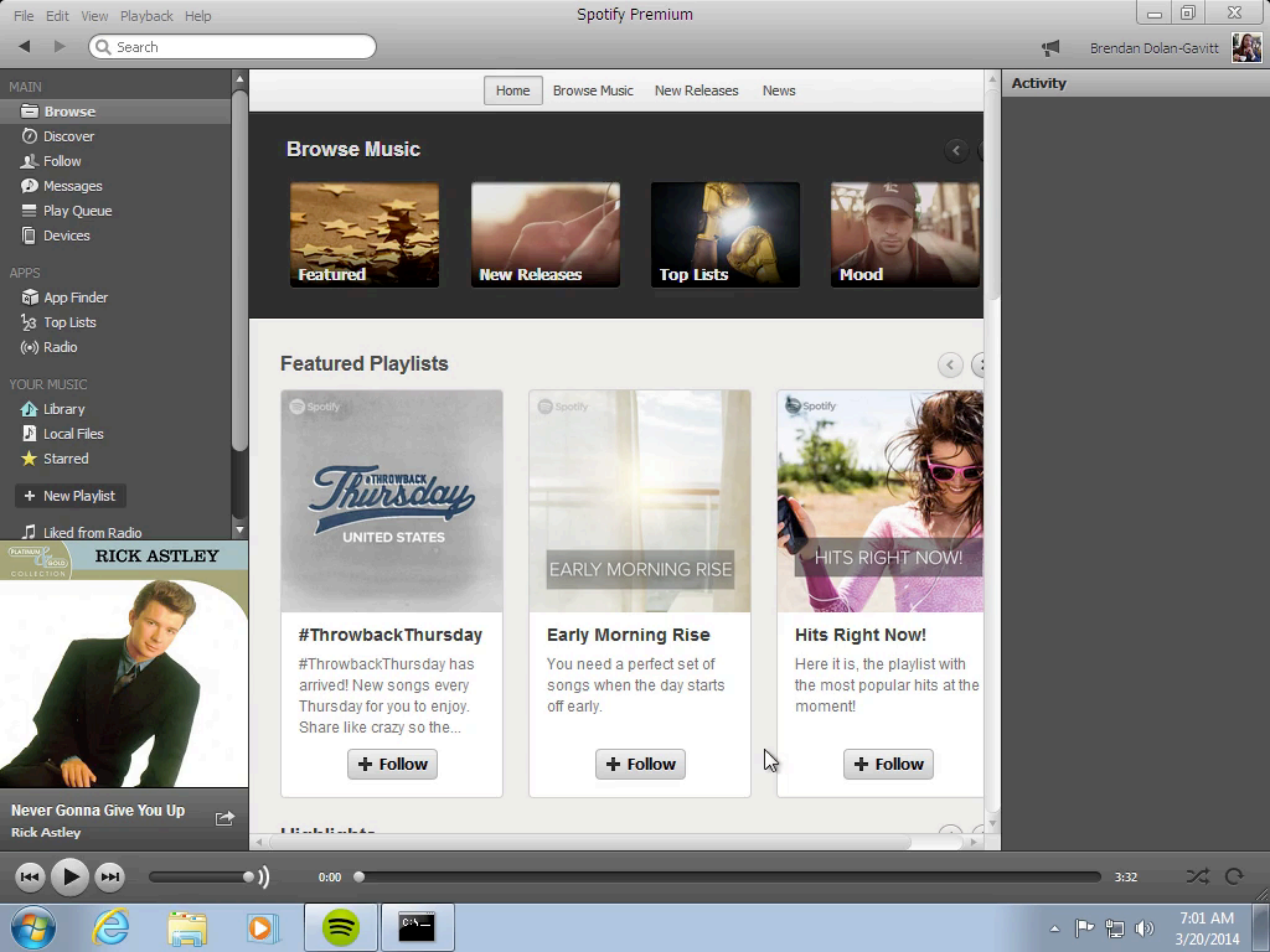
```
Inside test_key:
  Expected=00000017 calculated=000006e1
```

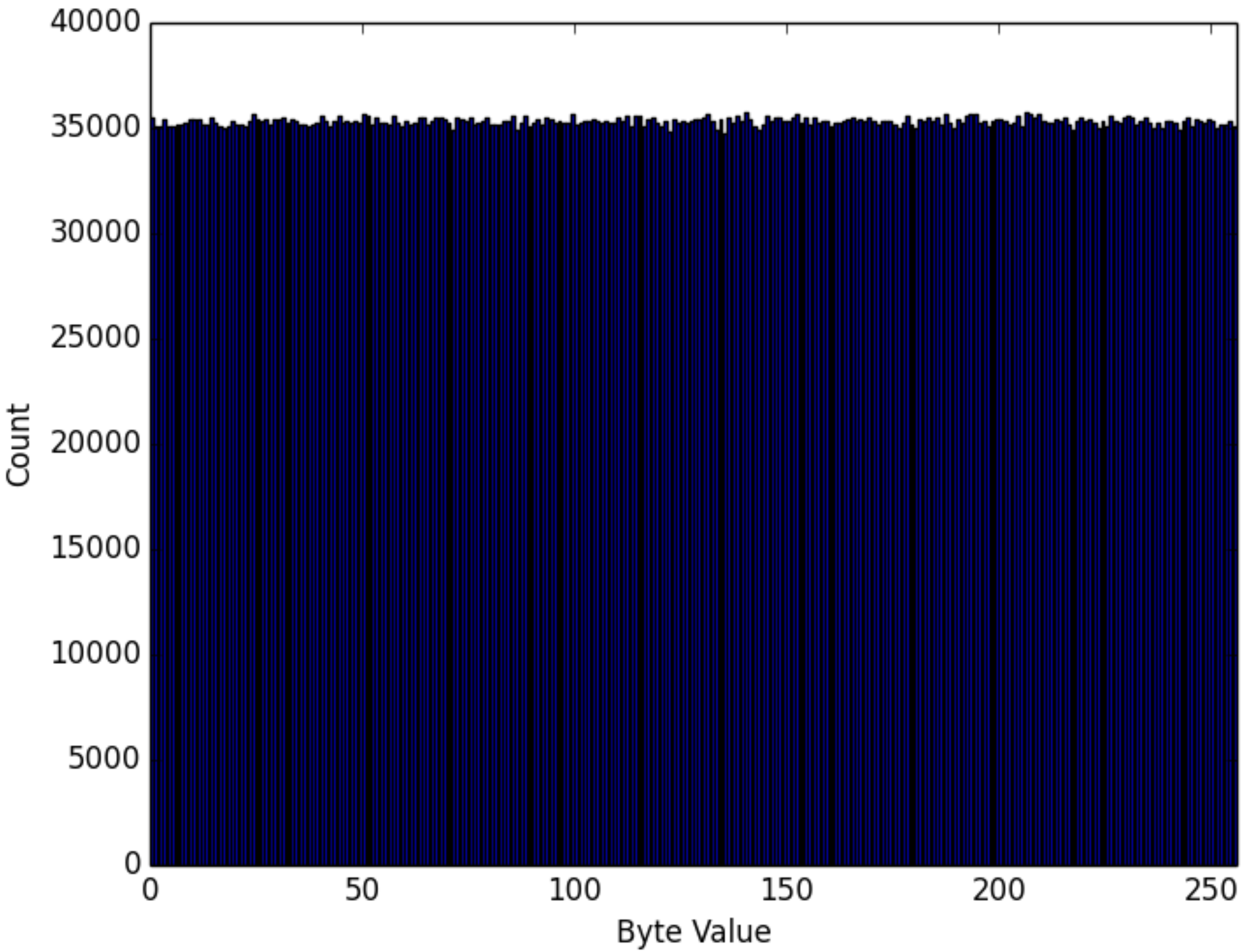
Breaking Spotify DRM

- DRM has a strong “signature”
  - **High** entropy, **high** randomness (χ^2) input
  - **High** entropy, **low** randomness (χ^2) output
- We can look for functions that match this description

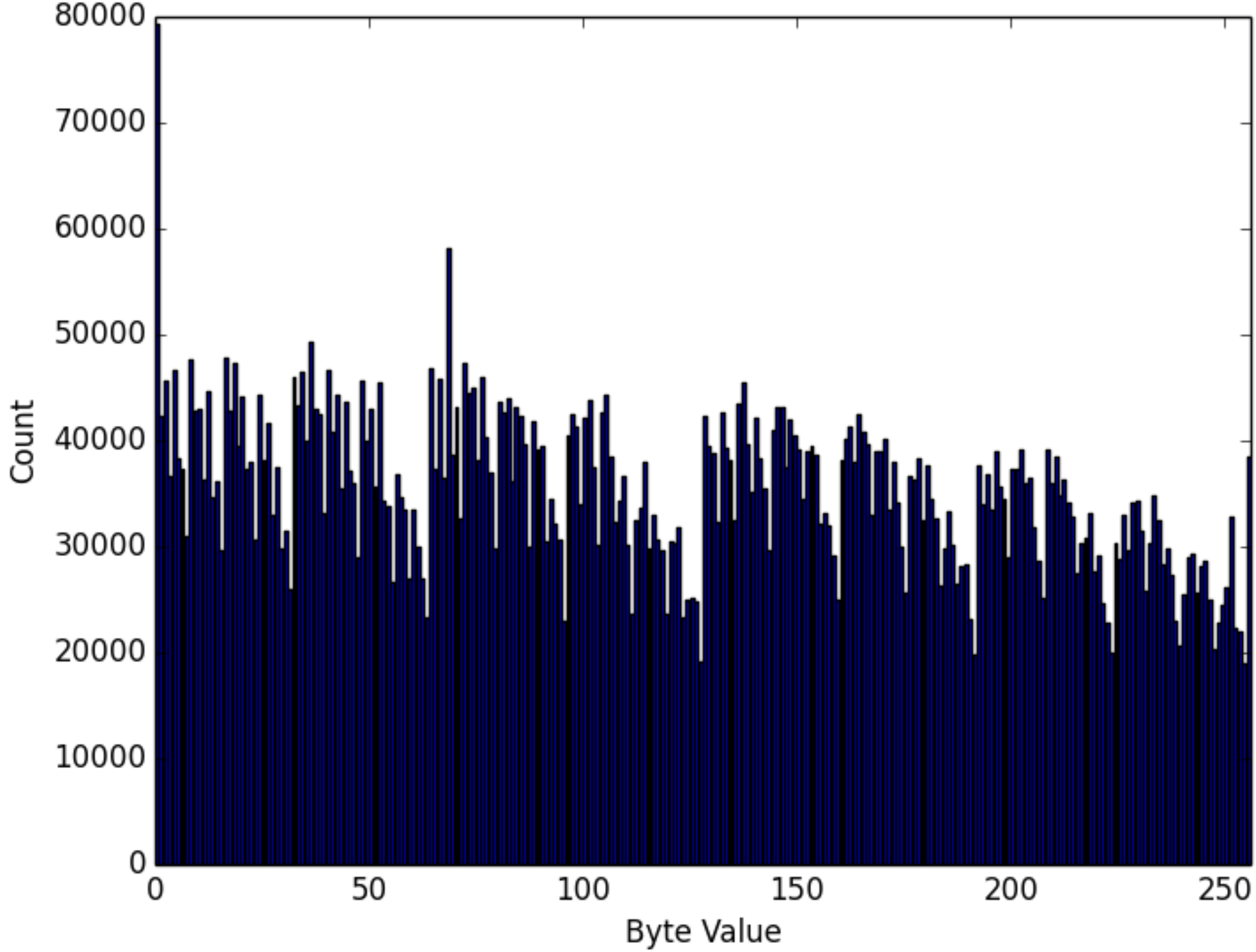
From: *Steal This Movie - Automatically Bypassing DRM Protection in Streaming Media Services* by Wang et al., USENIX Security 2013


File Edit View Playback Help
Spotify Premium
Search
Brendan Dolan-Gavitt
MAIN
Browse
Discover
Follow
Messages
Play Queue
Devices
APPS
App Finder
Top Lists
Radio
YOUR MUSIC
Library
Local Files
Starred
+ New Playlist
Liked from Radio
RICK ASTLEY
Never Gonna Give You Up
Rick Astley
Home
Browse Music
New Releases
News
Browse Music
Featured
New Releases
Top Lists
Mood
Featured Playlists
#THROWBACK Thursday
UNITED STATES
EARLY MORNING RISE
HITS RIGHT NOW!
#ThrowbackThursday
#ThrowbackThursday has arrived! New songs every Thursday for you to enjoy. Share like crazy so the...
+ Follow
Early Morning Rise
You need a perfect set of songs when the day starts off early.
+ Follow
Hits Right Now!
Here it is, the playlist with the most popular hits at the moment!
+ Follow
Activity
0:00
3:32
7:01 AM
3/20/2014


File Edit View Playback Help
Spotify Premium
Search
Brendan Dolan-Gavitt
MAIN
Browse
Discover
Follow
Messages
Play Queue
Devices
APPS
App Finder
Top Lists
Radio
YOUR MUSIC
Library
Local Files
Starred
New Playlist
Liked from Radio
RICK ASTLEY
Never Gonna Give You Up
Rick Astley
Home
Browse Music
New Releases
News
Browse Music
Featured
New Releases
Top Lists
Mood
Featured Playlists
#ThrowbackThursday
#ThrowbackThursday has arrived! New songs every Thursday for you to enjoy. Share like crazy so the...
+ Follow
Early Morning Rise
You need a perfect set of songs when the day starts off early.
+ Follow
Hits Right Now!
Here it is, the playlist with the most popular hits at the moment!
+ Follow
Activity
0:00
3:32
7:01 AM
3/20/2014



40000
35000
30000
25000
20000
15000
10000
5000
0
Count
0
50
100
150
200
250
Byte Value


80000
70000
60000
50000
40000
30000
20000
10000
0
Count
0
50
100
150
200
250
Byte Value

Extracting Audio

- Ok, so we find the function that decrypts DRM
- Now what?
- Write a plugin that waits until that function is called and then saves its output
    - Left as an exercise for the reader...

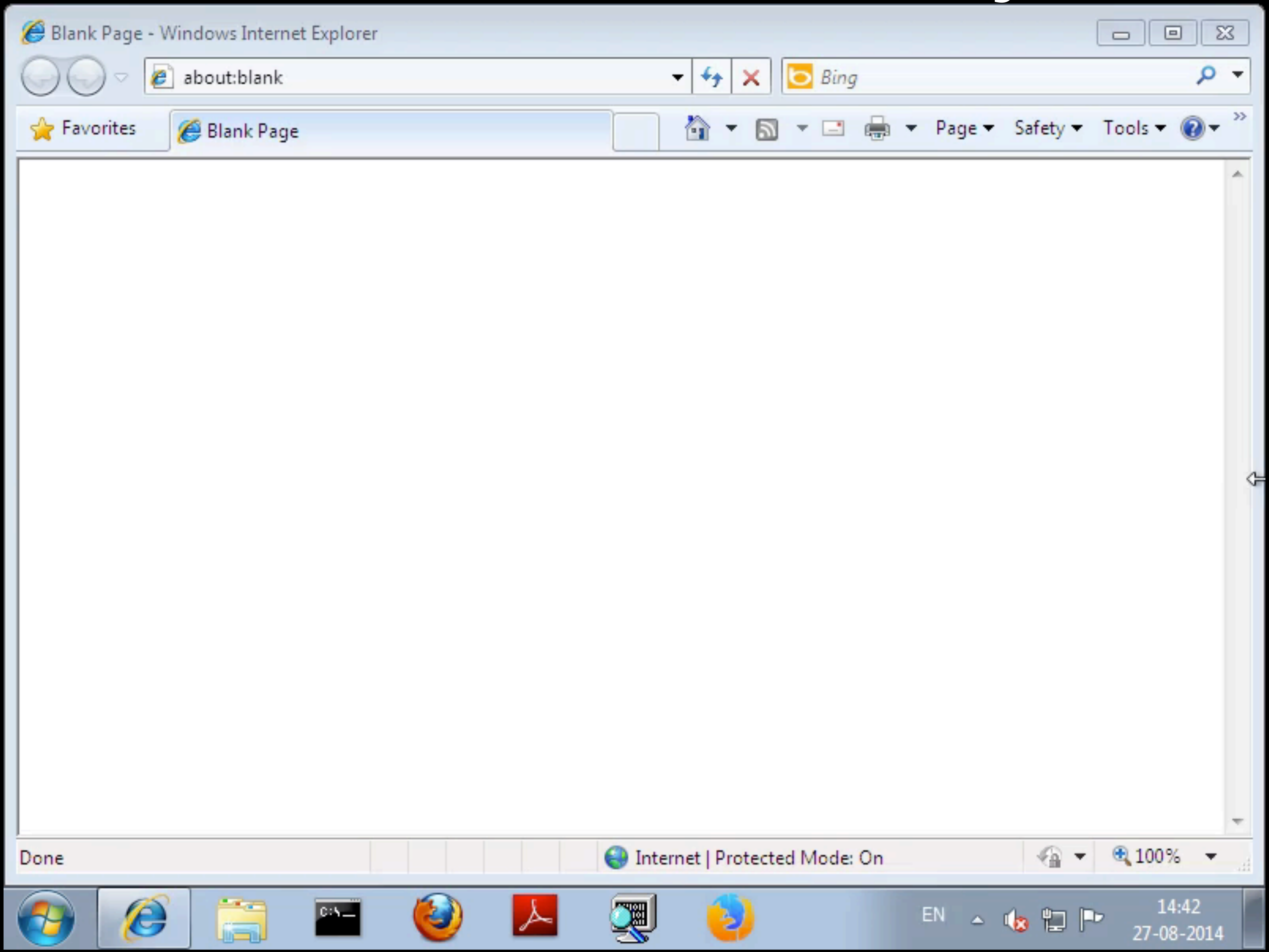
Extracting Audio

- Ok, so we find the function that decrypts DRM
- Now what?
- Write a plugin that waits until that function is called and then saves its output
  - Left as an exercise for the reader...

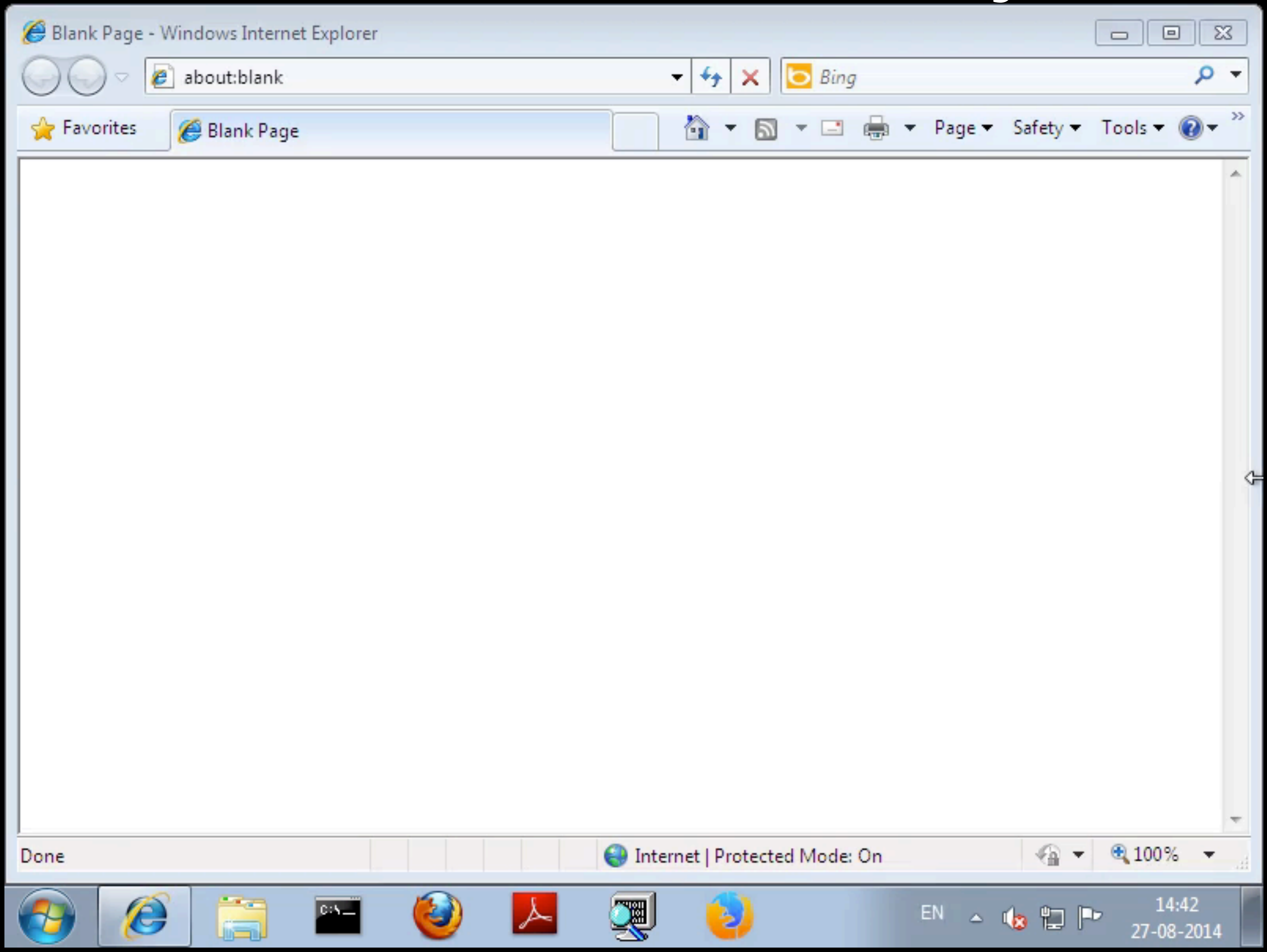
Future Work

- It is interesting that DRM decryption has a strong "dynamic signature"
- Are there other kinds of functions that can be identified by statistical properties of their inputs and outputs? Or intermediate states?
  - Compression / decompression
  - Cryptographic hash functions
  - Numerical computation?

IE Vulnerability



IE Vulnerability



Determining Root Cause

- We want to understand what caused the crash
- Can get bounds on the crash for use with *scissors* with two search strings in TZB:
    - “<html”
    - “has stopped working”
- Once found, can extract HTML for diagnosis

HTML Trigger

```
<HTML XMLNS:t="urn:schemas-microsoft-com:time">
<?IMPORT namespace="t"
implementation="#default#time2">
<body>
<div id="x" contenteditable="true">
HELLOWORLD
<t:TRANSITIONFILTER></t:TRANSITIONFILTER>
<script>
   document.getElementById("x").innerHTML = "";
   CollectGarbage();
   window.onclick;
   document.location.reload();
</script>
</div>
</body>
</HTML>
```

Use After Free Detector

- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

Heap:

Use After Free Detector

- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

Heap:

Use After Free Detector

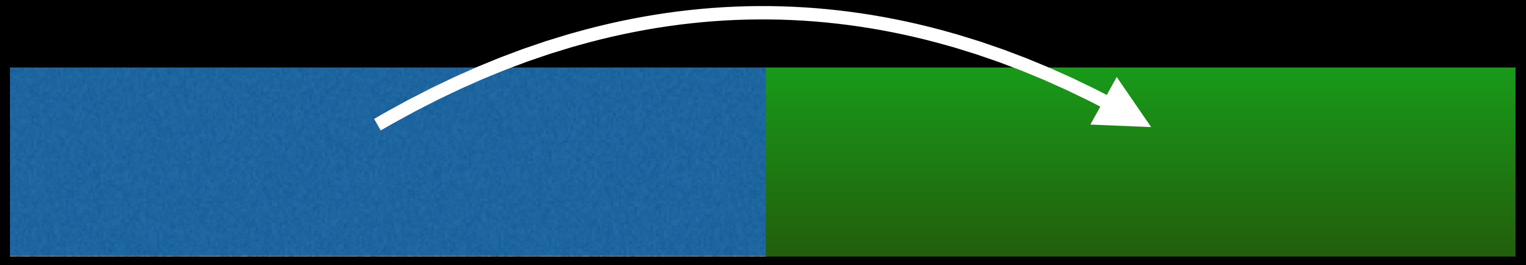
- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

Heap:

Use After Free Detector

- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

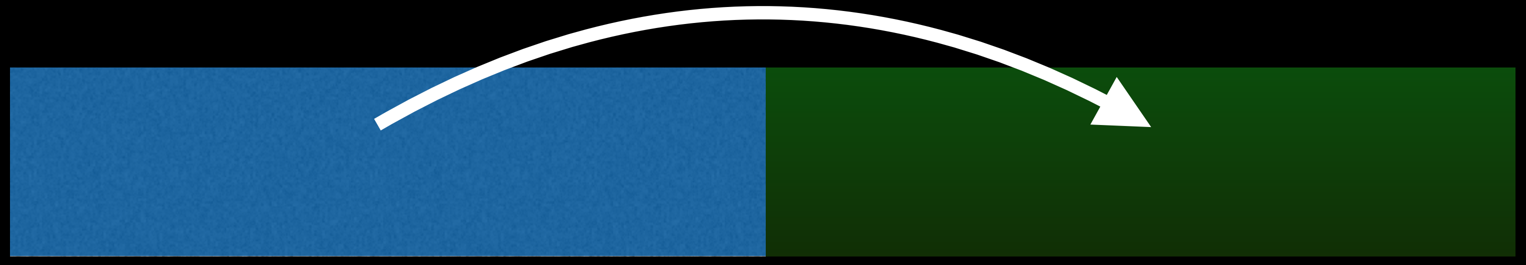
Heap:



Use After Free Detector

- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

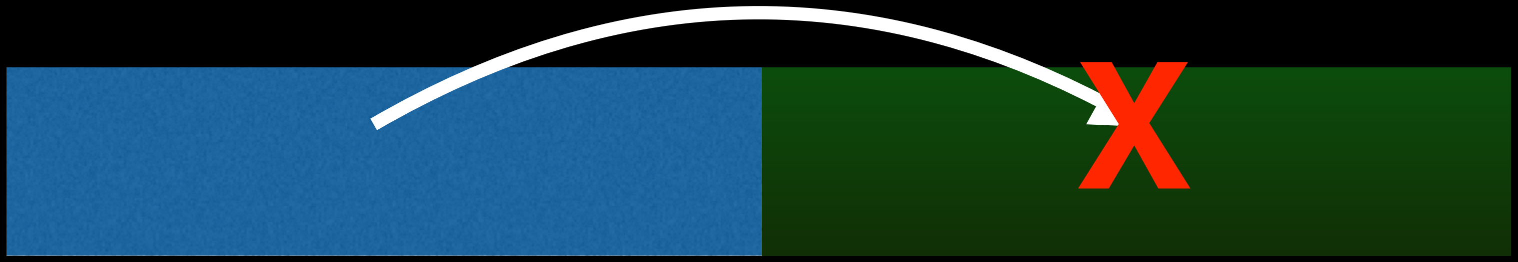
Heap:



Use After Free Detector

- Watch mallocs/frees and keep a map of allocated intervals
- Look for accesses to freed intervals
- Note: not necessarily complete!

Heap:



Use After Free Results

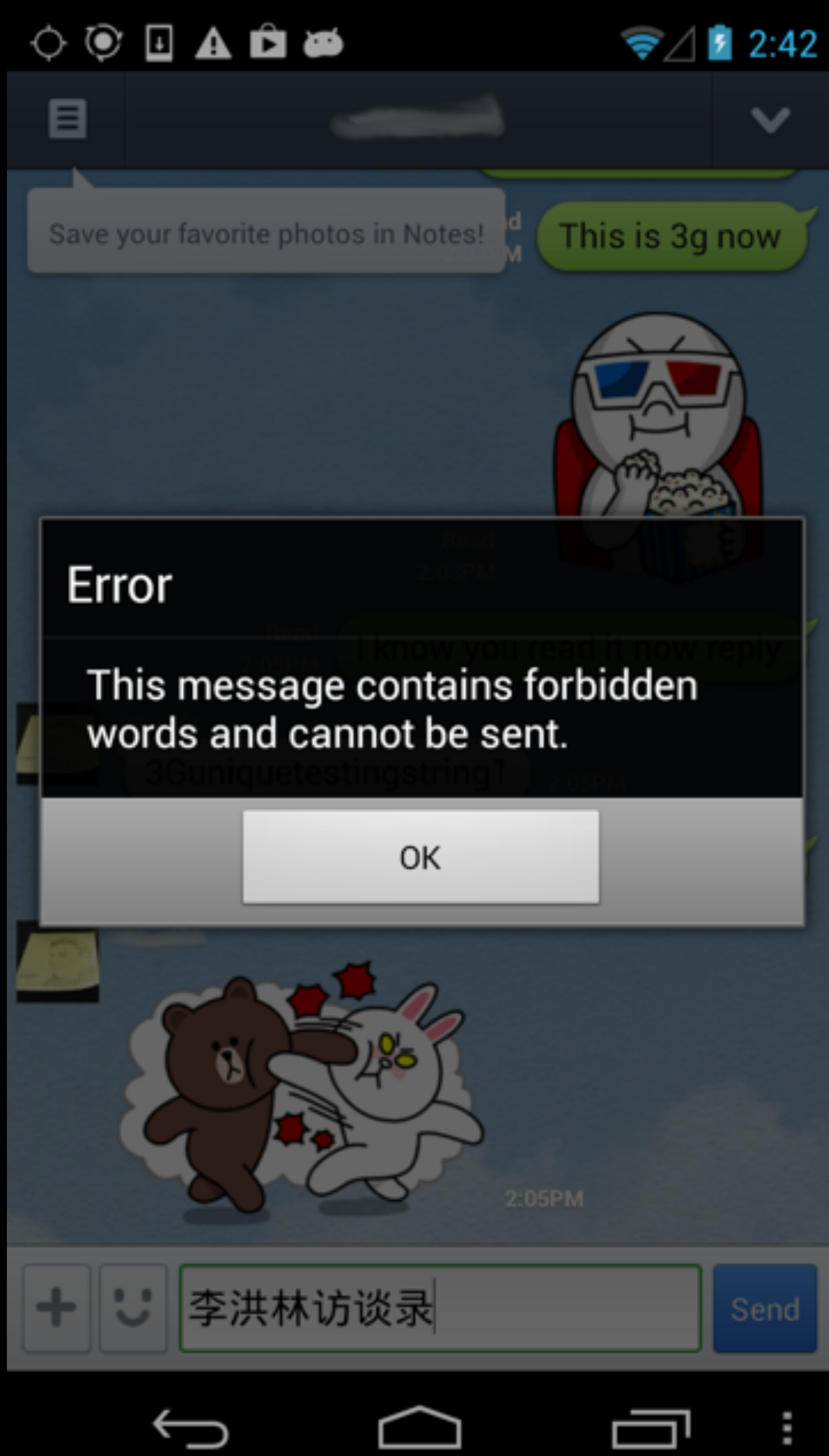
- UAF detector finds exactly one match:

```
USE AFTER FREE READ @ {3f98b320, 5556f0}! PC 6dc996f5
```

- Pinpoints exact location in code where dangling pointer is used
- Bug is CVE-2012-4792
- Could easily be extended for vulnerability *discovery* as well – see, e.g. Undangle by Caballero et al.

Censorship Blacklist Extraction

- LINE is a Japanese-made IM app for Android with ~560M users worldwide
- Found by CitizenLab to censor some words for Chinese users
- We want to find out which ones


2:42
Save your favorite photos in Notes!
This is 3g now
Error
This message contains forbidden words and cannot be sent.
OK
2:05PM
李洪林访谈录
Send


2:46
Tester
Read
1:58PM
Tester
Test
1:58PM
Tester
李. 洪. 林. 访. 谈. 录
1:58PM
Tester
******
1:59PM
Tester
******
2:01PM
Tester
*****
2:42PM
Tester
*****
2:43PM
Tester
*****
2:45PM

LINE Methodology

- Very simple strategy: use TZB to find usage of strings likely to be in “bad words” list:
  - 法轮 (Falun)
  - 天安门 (Tiananmen)
- Dump out the other data accessed at that same program point to get the full list

Censorship Blacklist (sample)

198964	共党	彭博	政变
FLG	共匪	天朝	周斌
GCD	共贼	兲朝	祖莹
GFW	胡温	屠城	共C档
18大	江派	屠杀	08宪章
38军	江系	团派	89事件
八九	江贼	退党	艾未未
半羽	近平	汪洋	薄瓜瓜
鲍彤	九评	瘟神	薄熙来
暴政	军警	晓波	曹建明
柴玲	六四	学潮	曾庆红
赤匪	马凯	学运	陈光诚
	民运	余杰	大纪元

For translations & context see https://china-chats.net/

Future Work

- What if we don't have a good idea of what words may be blacklisted?
- Instead, we may be able to use *taint analysis* combined with *dynamic slicing*
- Use taint analysis to find areas where user's typed input is compared against some value
- Then use dynamic slicing to trace the compared value back to its source

Toy Example

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

Taint user input

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

User input used in comparison

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

Look at data on the other side of the comparison

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

Follow data back to its origin (dynamic slicing)

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

Follow data back
to its origin
(dynamic slicing)

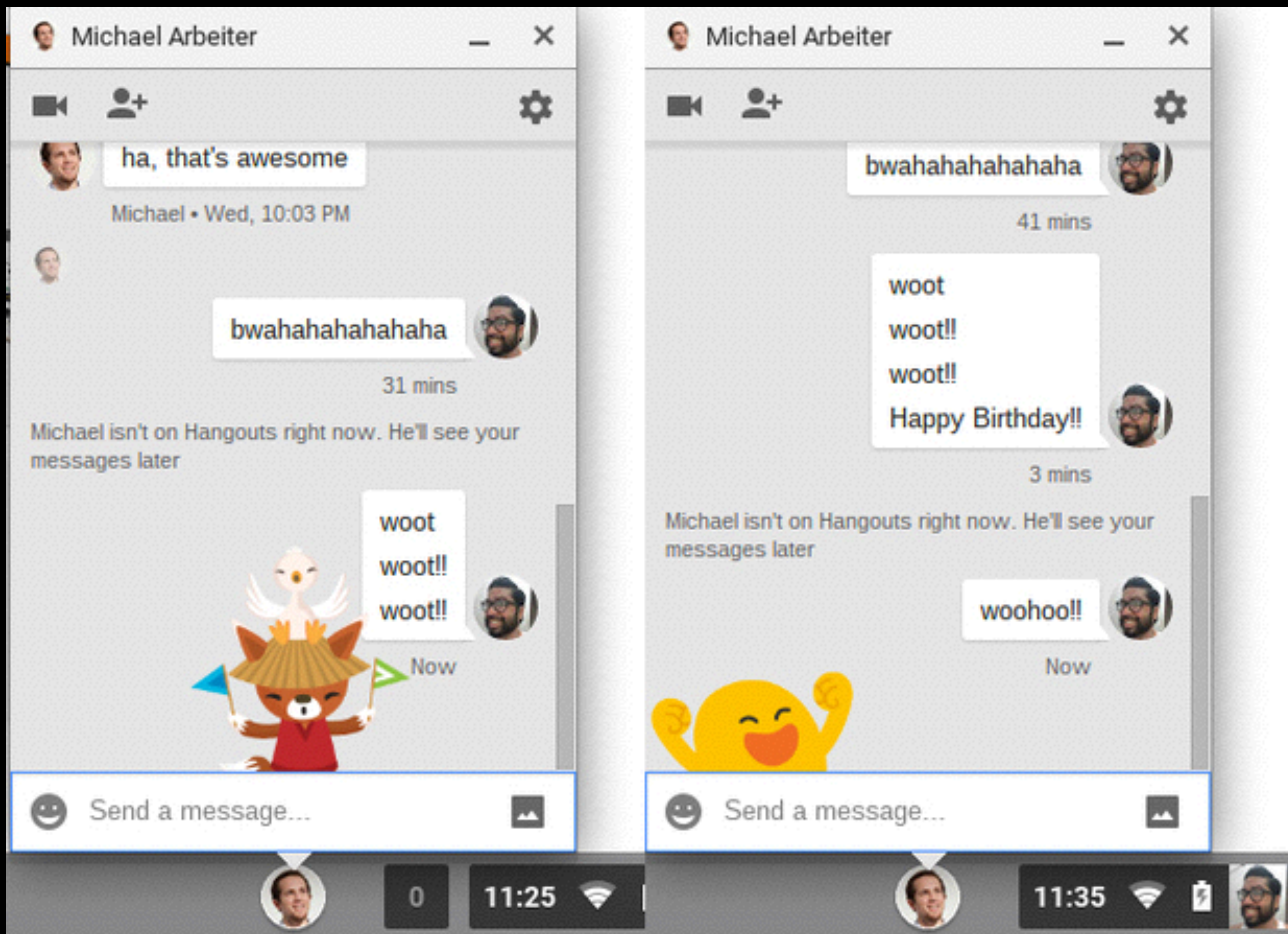
```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

Toy Example

Follow data back to its origin (dynamic slicing)

```
def censor_message():
    user_msg = input()
    blacklist = open('blacklist.txt').readlines()
    for word in blacklist:
        if user_msg == word:
            return "Censored"
    return user_msg
```

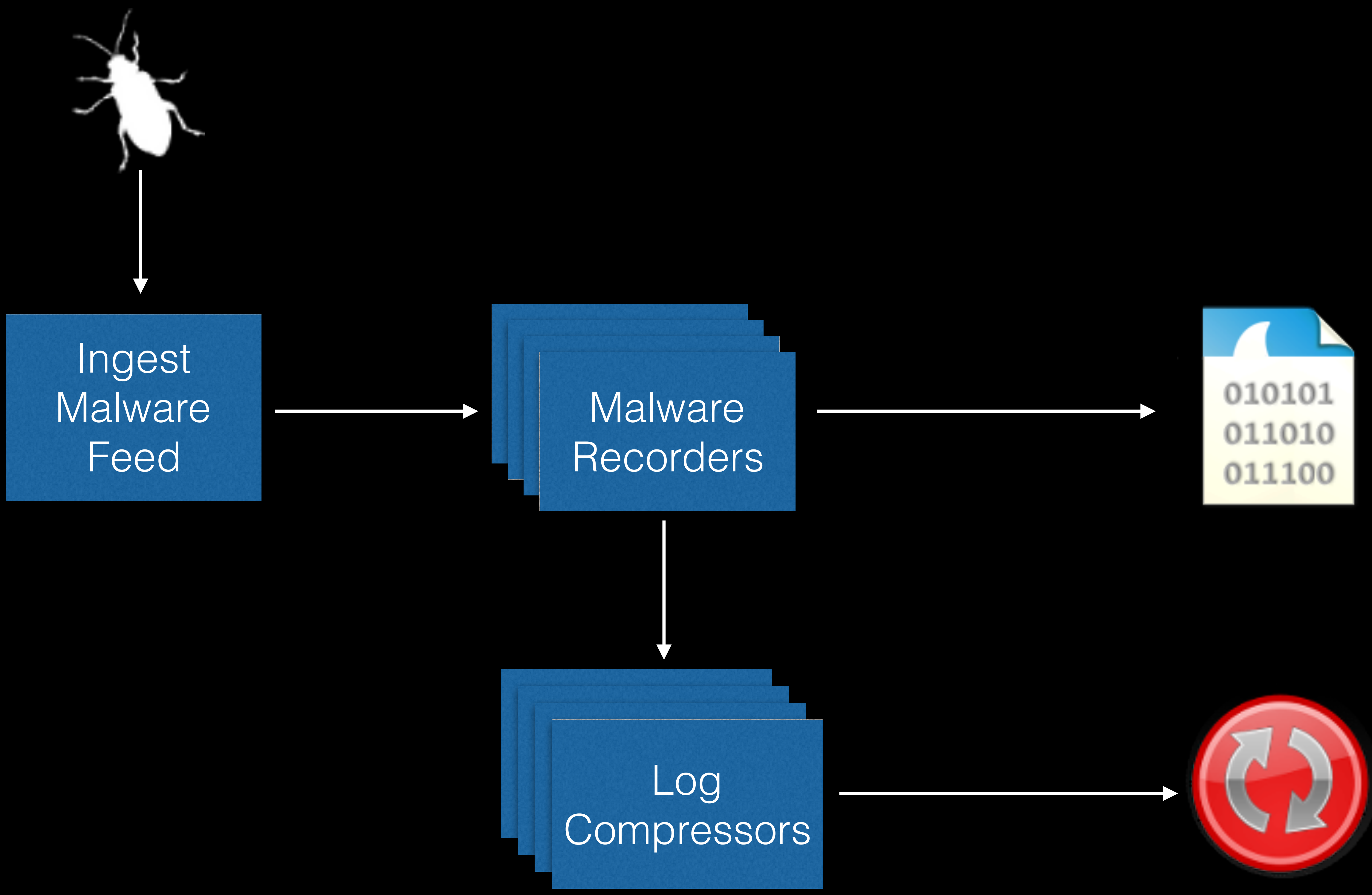
Applications Beyond Censorship



MalRec: A Malware Recording Platform

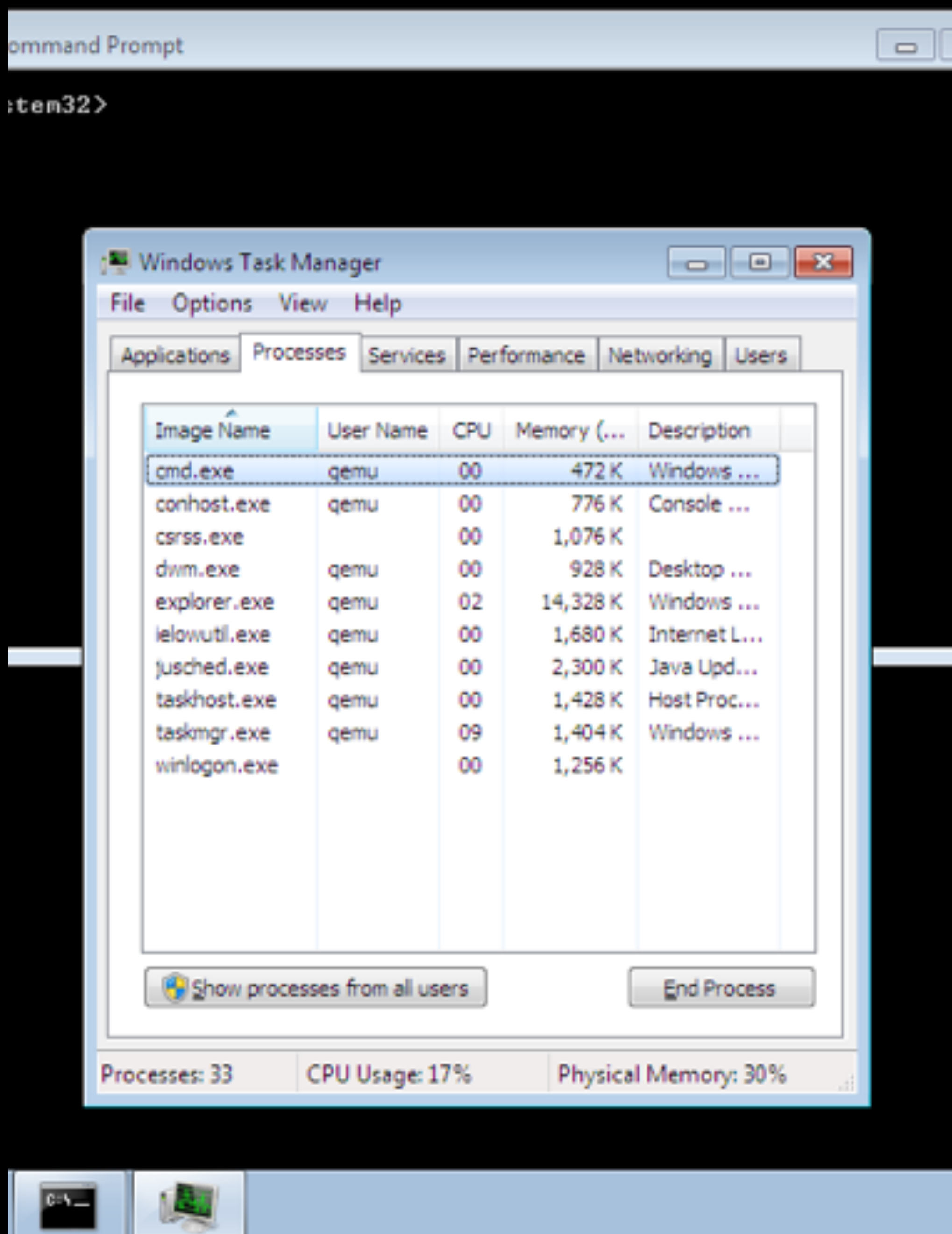
- Based on PANDA dynamic analysis platform
- Simple agentless setup:
  - Malware loaded via CD image
  - Started by sending keystrokes to VM
  - No in-guest monitoring utilities (reports can be generated from replays)

Malware Pipeline



GUI Actuation

GUI View



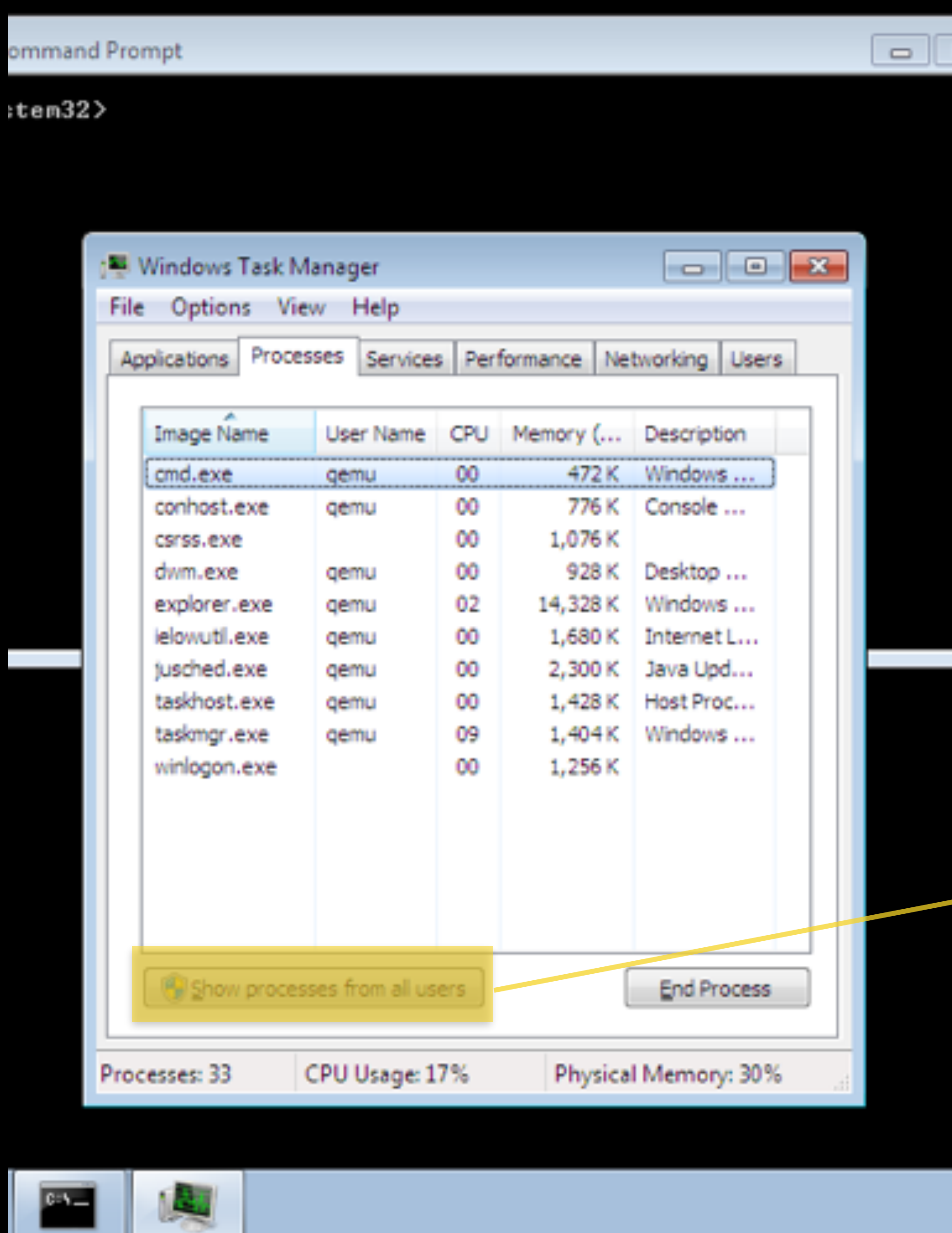
Volatility View (wintree)

```
./vol.py -f mem.dd --profile=Win7SP1x86 wintree
```

```
.Windows Task Manager (visible) taskmgr.exe:1516 -
..Users  taskmgr.exe:1516 -
...&Send Message... (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Logoff (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Disconnect (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...Users (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#70038 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
..Networking  taskmgr.exe:1516 -
...#40156 (visible) taskmgr.exe:1516 6.0.7601.17514!ScrollBar
...No Active Network Adapters Found. (visible) taskmgr.exe:1516 6.0.7601.
...Totals (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#201a8 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
..Performance  taskmgr.exe:1516 -
...&Resource Monitor... (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...Kernel Memory (MB) (visible) taskmgr.exe:1516 DavesFrameClass
...Physical Memory (MB) (visible) taskmgr.exe:1516 DavesFrameClass
..Tab1 (visible) taskmgr.exe:1516 6.0.7601.17514!SysTabControl32
...#50162  taskmgr.exe:1516 6.0.7601.17514!msctls_updown32
..Processes (visible) taskmgr.exe:1516 -
...&End Process (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Show processes from all users (visible) taskmgr.exe:1516 6.0.7601.175
...Processes (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#801a4 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
```

GUI Actuation

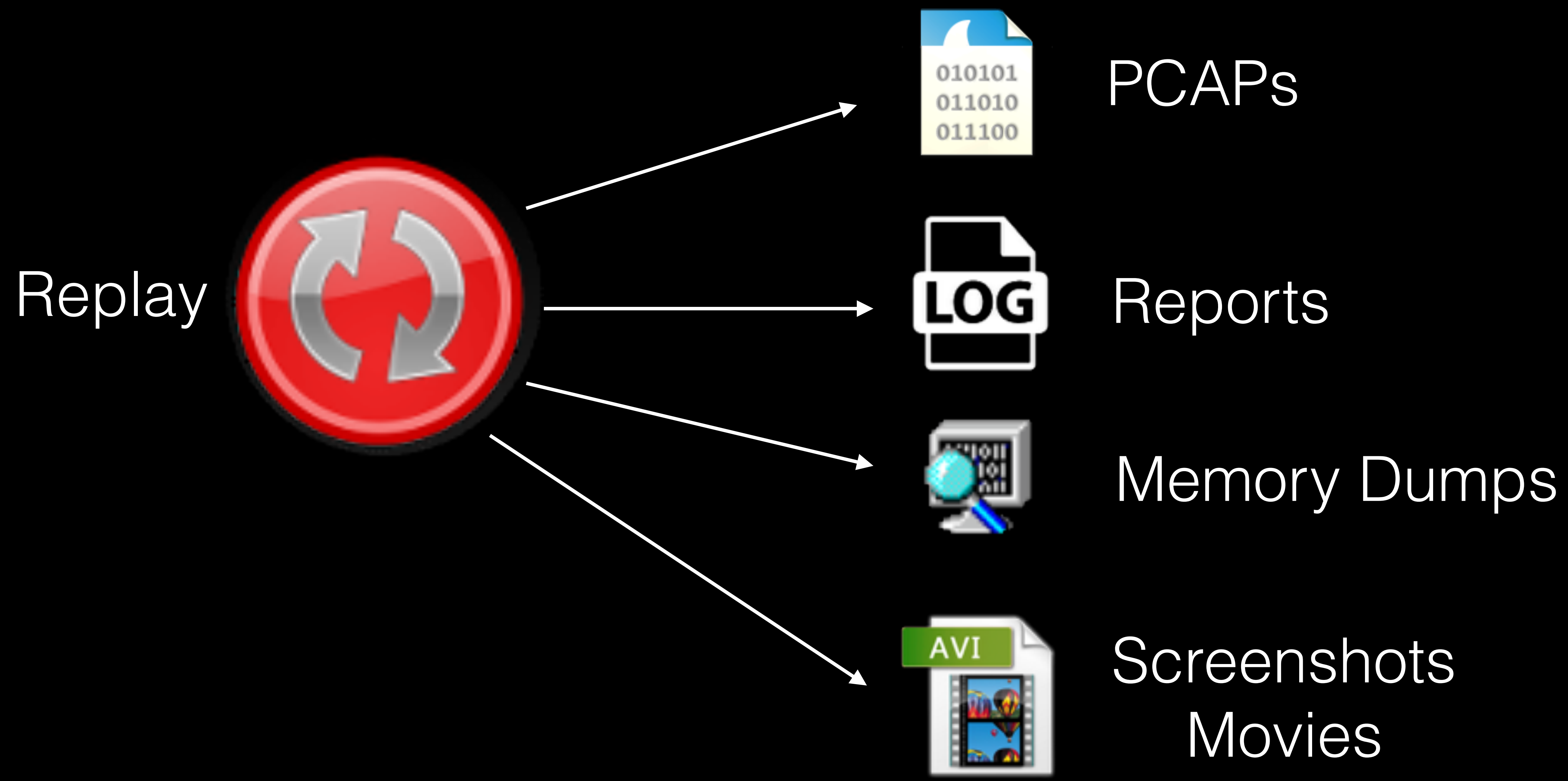
GUI View

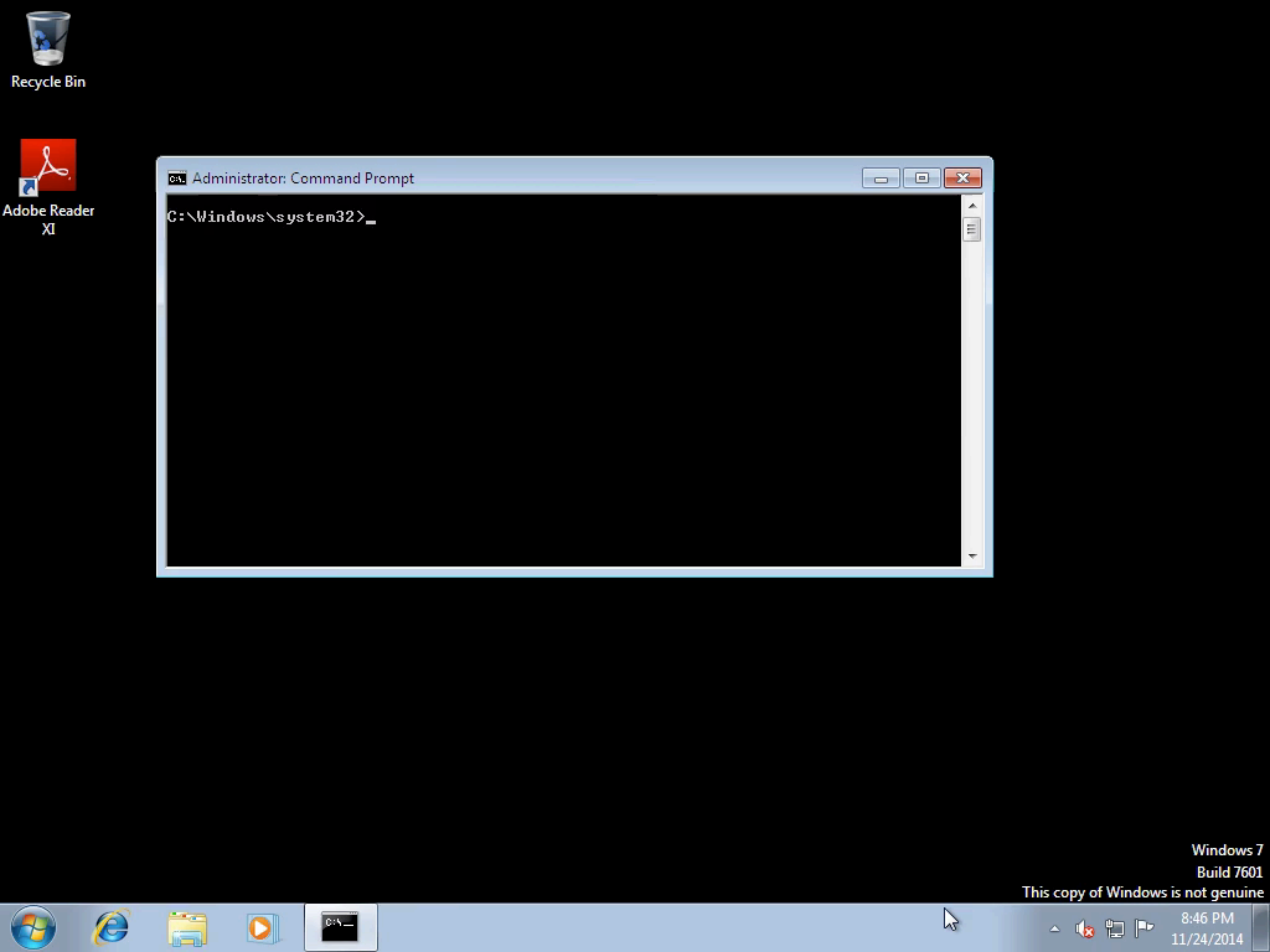


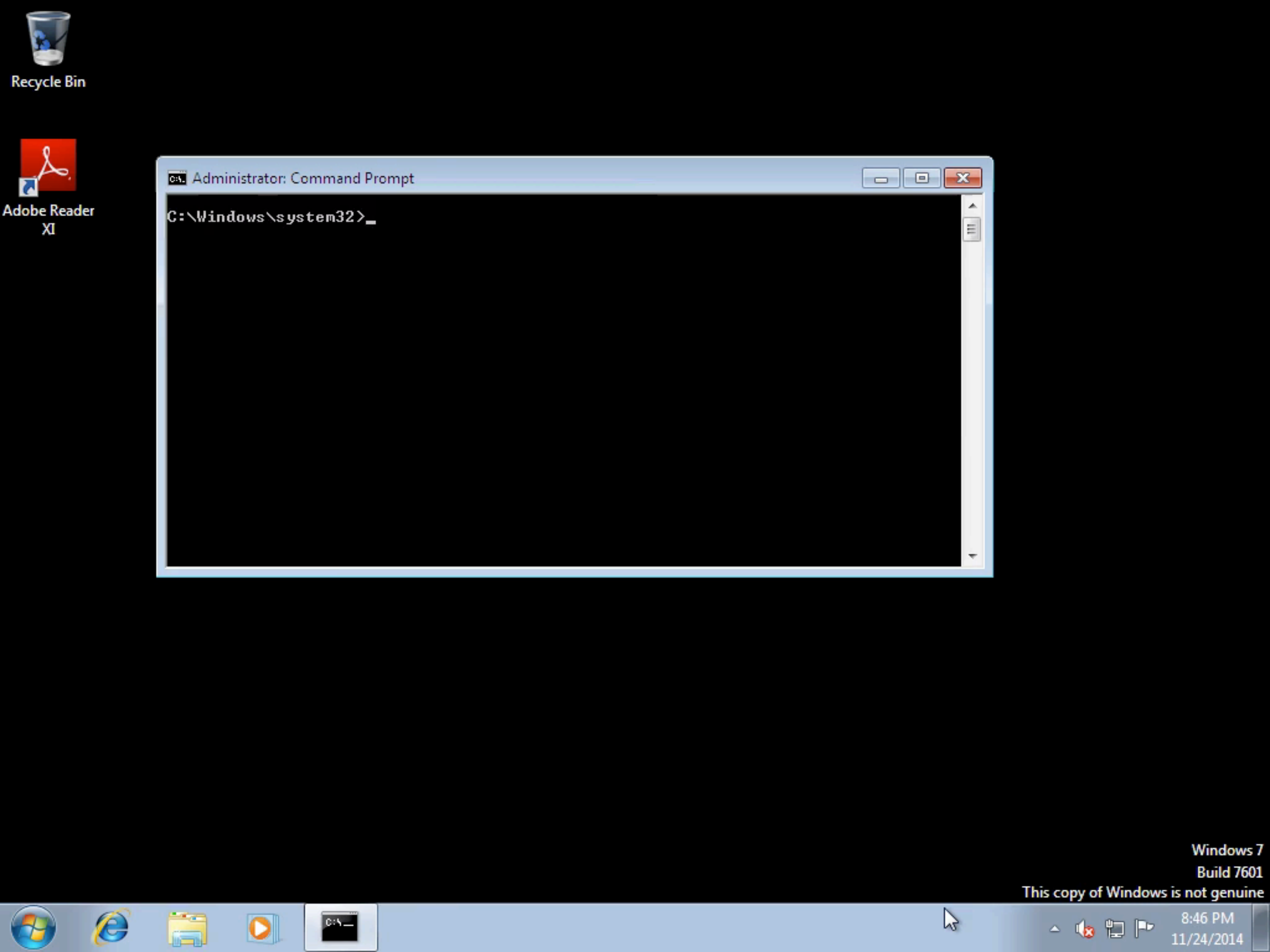
Volatility View (wintree)

```
./vol.py -f mem.dd --profile=Win7SP1x86 wintree
.Windows Task Manager (visible) taskmgr.exe:1516 -
..Users  taskmgr.exe:1516 -
...&Send Message... (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Logoff (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Disconnect (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...Users (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#70038 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
..Networking  taskmgr.exe:1516 -
...#40156 (visible) taskmgr.exe:1516 6.0.7601.17514!ScrollBar
...No Active Network Adapters Found. (visible) taskmgr.exe:1516 6.0.7601.
...Totals (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#201a8 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
..Performance  taskmgr.exe:1516 -
...&Resource Monitor... (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...Kernel Memory (MB) (visible) taskmgr.exe:1516 DavesFrameClass
...Physical Memory (MB) (visible) taskmgr.exe:1516 DavesFrameClass
..Tab1 (visible) taskmgr.exe:1516 6.0.7601.17514!SysTabControl32
...#50162  taskmgr.exe:1516 6.0.7601.17514!msctls_updown32
..Processes (visible) taskmgr.exe:1516 -
...&End Process (visible) taskmgr.exe:1516 6.0.7601.17514!Button
...&Show processes from all users (visible) taskmgr.exe:1516 6.0.7601.175
...Processes (visible) taskmgr.exe:1516 6.0.7601.17514!SysListView32
....#801a4 (visible) taskmgr.exe:1516 6.0.7601.17514!SysHeader32
```

Replay Subsumes Other Artifacts




Recycle Bin
Adobe Reader XI
Administrator: Command Prompt
C:\Windows\system32>_
Windows 7
Build 7601
This copy of Windows is not genuine
8:46 PM
11/24/2014


Recycle Bin
Adobe Reader XI
Administrator: Command Prompt
C:\Windows\system32>_
Windows 7
Build 7601
This copy of Windows is not genuine
8:46 PM
11/24/2014

Stats

- 50,176 traces available for download
- More than 1 quadrillion instructions' worth of execution
- Each trace has:
  - VirusTotal report (JSON)
  - Full record/replay log
  - Movie (originally 5 seconds, now 25s)
  - PCAP of network traffic

Log Size (redux)

- In the case of malware we can get additional savings: all recordings start from same snapshot
  - So, just save a diff of the snapshot for each new recording
- 24,189 full system executions
  - Before snapshot optimization: 2.4 terabytes
  - After: 387GB
- 1147.5 instructions/byte (!)

Open Data

- Full dataset available via HTTP, BitTorrent
  - http://moyix.blogspot.com/
- Also as individual files at:
  - http://panda.gtisc.gatech.edu/malrec/

panda.gtisc.gatech.edu/malrec/

UUID	Filename	MD5	PCAP	RR Log	Added
4fc89505-75a0-4734-ac6d-1ebbdca28caa	005b80688b590435b7aab13342a00c6e.exe	005b80688b590435b7aab13342a00c6e	pcap	rrlog	2014-12-08 01:32:51.913522107 +0000
a64339ce-5fcb-415e-99f4-aa639c635805	02b955cf0d29e46502cb5dafd4244082.exe	02b955cf0d29e46502cb5dafd4244082	pcap	rrlog	2014-12-08 01:36:18.581528091 +0000
9a5cfaee-a478-444f-8dca-7f401f8f0df5	00b68dc33cd0a7122ffc8f1a237528c7.exe	00b68dc33cd0a7122ffc8f1a237528c7	pcap	rrlog	2014-12-08 01:32:58.649522302 +0000
92b72e3a-917c-4792-91aa-1d9950739d99	005de27b207285e70dea705feff8a4e7.exe	005de27b207285e70dea705feff8a4e7	pcap	rrlog	2014-12-08 01:33:11.061522661 +0000
e2152d26-73ff-4953-907d-8d6e9e32a4f3	03627679800f9540633a0a338e2d1930.exe	03627679800f9540633a0a338e2d1930	pcap	rrlog	2014-12-08 01:36:51.157529034 +0000
bc2581aa-85e8-4012-9e27-c728a00f3ff8	02b9a077e3c373089f0624a8bb66ec8d.exe	02b9a077e3c373089f0624a8bb66ec8d	pcap	rrlog	2014-12-08 01:36:04.829527693 +0000
3be52156-4f93-4a37-9af1-d1d45b526825	03d33743572fa24494582f24137e0d89.exe	03d33743572fa24494582f24137e0d89	pcap	rrlog	2014-12-08 01:32:47.105521968 +0000
f8b6036a-40d8-486b-af0c-8ec2840960f4	03d78a0f036ea665b8147a584584b179.exe	03d78a0f036ea665b8147a584584b179	pcap	rrlog	2014-12-08 01:36:24.365528258 +0000
f2d1662f-1079-4f45-b542-8b1cf8fdb1a9	079e0f2a6d817d8c88b1587f352d7cd0.exe	079e0f2a6d817d8c88b1587f352d7cd0	pcap	rrlog	2014-12-08 01:40:47.225535869 +0000
5d5eb4f6-13b0-44ed-bfd6-73b5aa0d284f	0995d976f26730007596d14fcce219a0.exe	0995d976f26730007596d14fcce219a0	pcap	rrlog	2014-12-08 01:40:24.841535221 +0000
a6d2a1e0-027c-4f80-90c6-9e9f84de53da	0c5fd363447293ac308e8079d532192c.exe	0c5fd363447293ac308e8079d532192c	pcap	rrlog	2014-12-08 01:40:09.885534788 +0000
d4ec17b9-90ec-4e96-b40b-f6e77f5ca1a7	0e1d93833d3909e454b79c9ccf82c698.exe	0e1d93833d3909e454b79c9ccf82c698	pcap	rrlog	2014-12-08 01:40:10.293534800 +0000
781b95ff-943f-4590-877e-442d31991320	0f3f08e54ac62879b8ac4873e4be58e9.exe	0f3f08e54ac62879b8ac4873e4be58e9	pcap	rrlog	2014-12-08 01:43:53.813541271 +0000
0c413017-1c47-4d48-b90e-5d21e5407b52	101357b66a53eb86cab6c69fc48df3b7.exe	101357b66a53eb86cab6c69fc48df3b7	pcap	rrlog	2014-12-08 01:42:42.201539198 +0000
60a022d2-2287-4814-8d0d-676e215c0db1	10146d57a77bd3008e7f789b2a1b2540.exe	10146d57a77bd3008e7f789b2a1b2540	pcap	rrlog	2014-12-08 01:42:54.657539558 +0000
8edbd0f0-9d0f-41d9-9148-bc92966e949b	12b5501c2f30e8c3b7a8475da1c8e05e.exe	12b5501c2f30e8c3b7a8475da1c8e05e	pcap	rrlog	2014-12-08 01:57:43.441565292 +0000
537a5f48-7233-4996-af8e-20e3df1e99aa	11b64c44a79fc463d1c46c9faf1856ca.exe	11b64c44a79fc463d1c46c9faf1856ca	pcap	rrlog	2014-12-08 01:57:39.601565180 +0000
f481da2e-5ca1-4e60-a7d9-45a3a410f758	11225eec69d383c79fb6d4bff180ca7d.exe	11225eec69d383c79fb6d4bff180ca7d	pcap	rrlog	2014-12-08 01:57:41.937565248 +0000
f2298ba9-af24-473b-b14e-b564445741c8	17af4487d844314a20f03c866d3d5fa2.exe	17af4487d844314a20f03c866d3d5fa2	pcap	rrlog	2014-12-08 01:57:43.593565296 +0000
f220daf4-eaff-4626-b935-6938e5fd5c2f	257db161cbcf9d820b00c51b6d7d18e7.exe	257db161cbcf9d820b00c51b6d7d18e7	pcap	rrlog	2014-12-08 02:03:16.933574947 +0000
b437845a-6c4e-48c2-b1cf-db8e18e369df	1e888f5b607899b50c09f1840b474d0c.exe	1e888f5b607899b50c09f1840b474d0c	pcap	rrlog	2014-12-08 02:04:05.617576357 +0000
9f5b9ff9-957f-4b4d-8c50-6f028ab134e2	1c012c325a06e52b1e56b1a3420620e2.exe	1c012c325a06e52b1e56b1a3420620e2	pcap	rrlog	2014-12-08 02:03:25.617575199 +0000
0d8cf2c9-b9c0-468b-8b55-9a9c2f7b0459	267c351d05b28db0c06620536bf4f010.exe	267c351d05b28db0c06620536bf4f010	pcap	rrlog	2014-12-08 02:03:42.361575683 +0000
8cba72d5-9f8d-446d-a9fe-7abf85d025fc	26e7f238b29cdc9c9ca06b35332f0c77.exe	26e7f238b29cdc9c9ca06b35332f0c77	pcap	rrlog	2014-12-08 02:08:39.733584293 +0000
464d62fe-20e9-43a7-afb1-ae730e571163	29cc460c9fa5c6b7edea77eaf91102c9.exe	29cc460c9fa5c6b7edea77eaf91102c9	pcap	rrlog	2014-12-08 02:08:59.489584865 +0000
813e63fc-43aa-498a-8af2-d8088384b874	289510340cc1396f995bf20ee4ea9bb3.exe	289510340cc1396f995bf20ee4ea9bb3	pcap	rrlog	2014-12-08 02:09:10.321585179 +0000
ce28db56-a5d3-4a28-ba69-3f603192e3ce	29dc3212b5fae469ecffa8ed1a1a1599.exe	29dc3212b5fae469ecffa8ed1a1a1599	pcap	rrlog	2014-12-08 02:09:11.157585203 +0000
974dbfac-4017-441e-8471-f84c81c7a818	2b4c8a076d21ccaf82e6e60b05d9f033.exe	2b4c8a076d21ccaf82e6e60b05d9f033	pcap	rrlog	2014-12-08 02:14:00.745593588 +0000
7c8801fc-c29f-49c5-8412-dce75dea3fa0	3b5c8f00989260c51395cd0d09aa0cb1.exe	3b5c8f00989260c51395cd0d09aa0cb1	pcap	rrlog	2014-12-08 02:14:06.501593754 +0000
e5f6f3d3-29e4-42fd-9011-522054fee9f3	2db49478ce69cb1beaa3e96471cdf4e2.exe	2db49478ce69cb1beaa3e96471cdf4e2	pcap	rrlog	2014-12-08 02:14:05.069593713 +0000
9fc52909-6fa1-468f-b5dc-280b7d0c2e17	3ba61a3efa0227bd4d7e0a3e2d6e415c.exe	3ba61a3efa0227bd4d7e0a3e2d6e415c	pcap	rrlog	2014-12-08 02:14:19.137594120 +0000
5bc23607-cc4c-468c-b25c-3351920bb6ba	3d3f5e93b5386db5fdc8e637a5ed0480.exe	3d3f5e93b5386db5fdc8e637a5ed0480	pcap	rrlog	2014-12-08 02:19:19.849602827 +0000

MalRec Limitations

- Analysis time is fixed (10 minutes)
- Only one path through malware
- PANDA is based on QEMU 1.0.1 & non-virtualized – very detectable
- Currently do not accept submissions from the public

Future Work

- Extract interesting data from these traces
  - Printable strings passing through memory
  - Instruction mnemonic histograms
- Go big on our big data
  - Visualization
  - Machine learning
  - Searching / information retrieval

LAVA: Large-Scale Automated Vulnerability Addition

Thesis

Major problem in computer science: programs just don't have enough bugs

LAVA: Large-Scale Automated Vulnerability Addition

Thesis

~~Major problem in computer science: programs just don't have enough bugs~~

Jedi Truth

What I told you was true... from a certain point of view

LAVA: Large-Scale Automated Vulnerability Addition

Thesis

We don't know where the bugs are in programs or how they are distributed.

Debugging the Bug Finders

- Many companies have products that claim to find bugs in programs
- Lack of ground truth makes it very difficult to evaluate these claims made
- If Coverity finds 22 bugs in my program, is that good or bad?
- **Do these tools work?**

Debugging the Bug Finders

- Existing corpora are fixed size and static – it's easy for vendors to optimize to the benchmark
- Instead we would like to automatically create corpora!
- Take an existing program and *automatically* add new bugs into it
- Now we can measure how many of our bugs they find, giving some indication of their performance

Goals

- We want to produce bugs that are:
  - Plentiful (can put 1000s into a program easily)
  - Distributed throughout the program
  - Come with a triggering input
  - Only manifest for a tiny fraction of inputs

Sounds Simple... But Not

- Why not just change all the strncpys to strcpys?
  - Turns out this breaks most programs for *every* input – trivial to find the bugs
  - We won't know how to trigger the bugs – hard to prove they're "real" and security-relevant
  - This applies to most local, random mutations

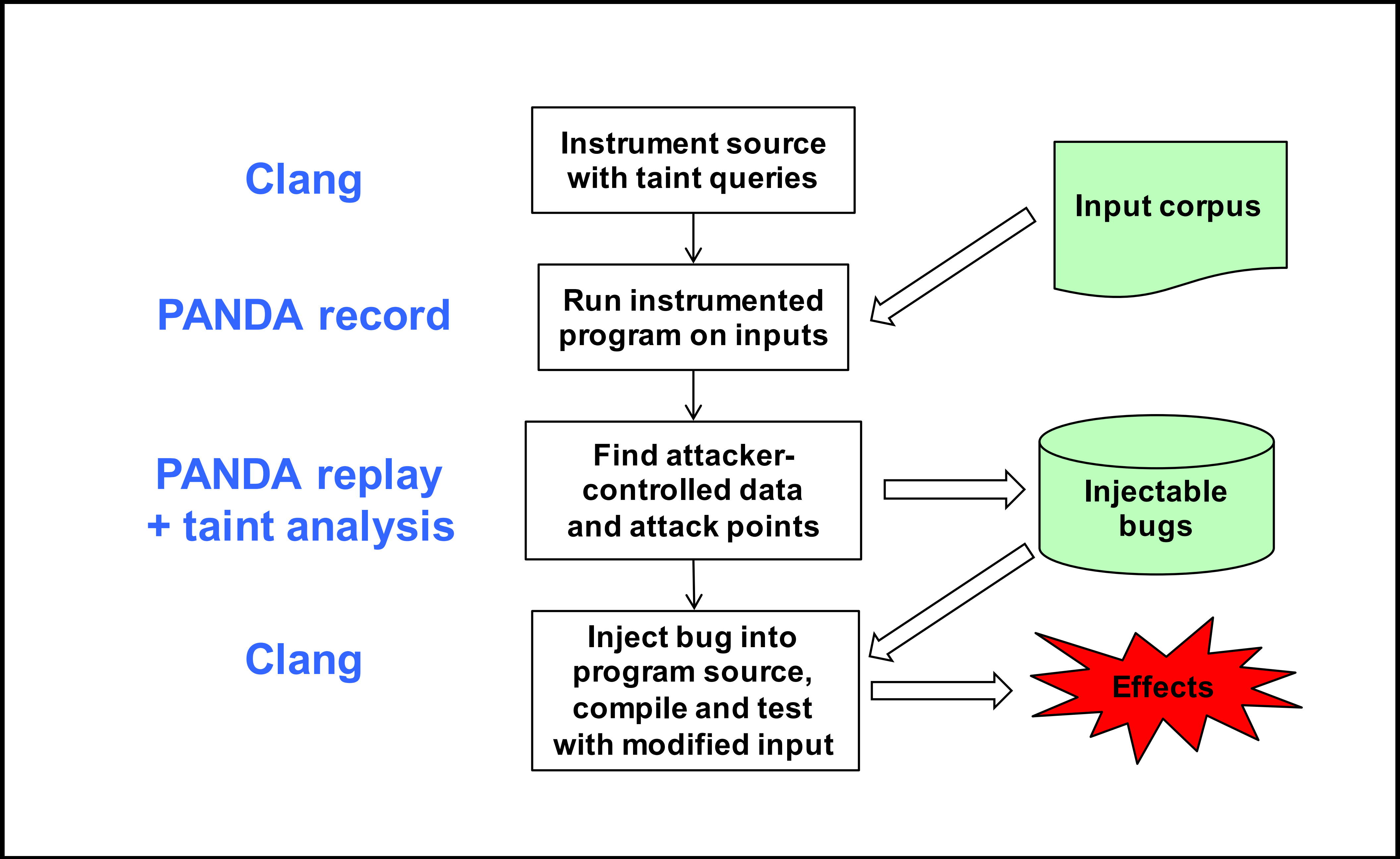
Our Approach

- We want to find parts of the program's input data that are:
  - **Dead:** not currently used much in the program (i.e., we can set to arbitrary values)
  - **Uncomplicated:** not altered very much (i.e., we can predict their value throughout the program's lifetime)
  - **Available** in some program variables
- If we can find these, we will be able to add code to the program that uses such data to trigger a bug

New Measures

- How do we find out what data is **dead** and **uncomplicated**?
- Two new taint-based measures:
  - *Liveness*: a count of how many times some input byte is used to decide a branch
  - *Taint compute number*: a measure of how much computation been done on some data

Approach

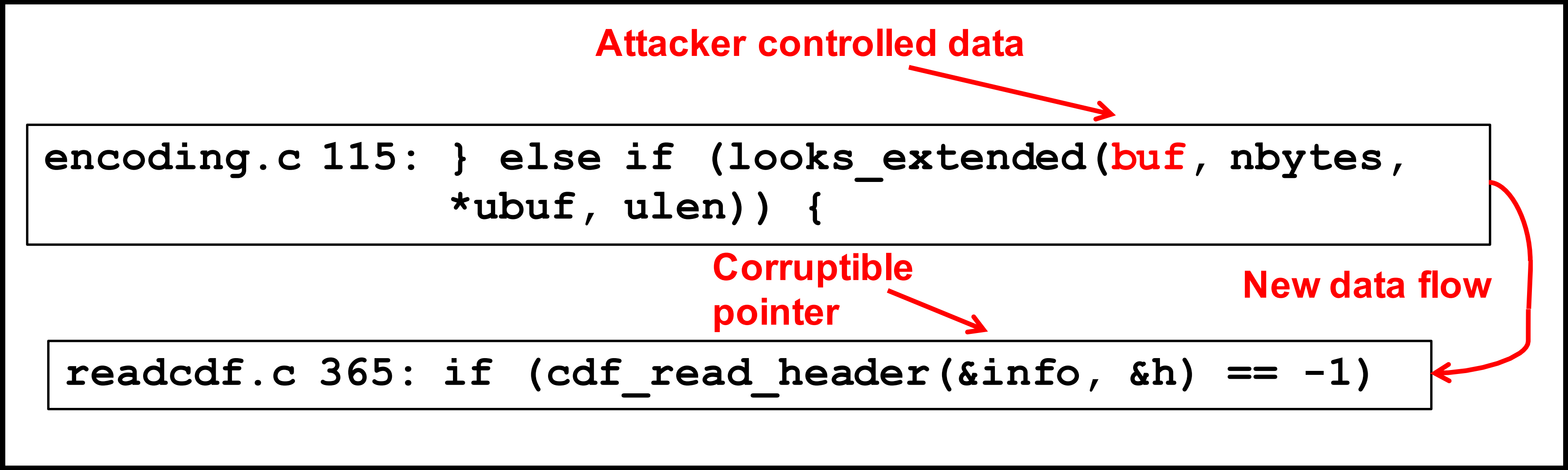


Clang
PANDA record
PANDA replay
+ taint analysis
Clang
Instrument source with taint queries
Run instrumented program on inputs
Find attacker-controlled data and attack points
Inject bug into program source, compile and test with modified input
Input corpus
Injectable bugs
Effects

LAVA Bug Example

- PANDA taint analysis shows that bytes 0-3 of `buf` on line 115 of `src/encoding.c` is attacker-controlled (dead & uncomplicated)

- From PANDA we also see that in `readcdf.c` line 365 there is a read from a pointer – if we modify the pointer value we will likely cause a bug in the program



LAVA Bug Example

```
// encoding.c:
} else if
  (({int rv =
       looks_extended(buf, nbytes, *ubuf, ulen);
     if (buf) {
       int lava = 0;
       lava |= ((unsigned char *)buf)[0];
       lava |= ((unsigned char *)buf)[1] << 8;
       lava |= ((unsigned char *)buf)[2] << 16;
       lava |= ((unsigned char *)buf)[3] << 24;
       lava_set(lava);
    }; rv; })) {
```

```
// readcdf.c:
if (cdf_read_header
     ((&info) + (lava_get()) *
       (0x6c617661 == (lava_get()) || 0x6176616c == (lava_get())),
        &h) == -1)
```

Results: Specific Value

Program	Total Bugs	Unique Bugs Found		
		FUZZER	SES	Combined
`uniq`	28	7	0	7
`base64`	44	7	9	14
`md5sum`	57	2	0	2
`who`	2136	0	18	18
Total	2265	16	27	41

Only 2% of injected bugs found

Results: Range-Triggered Bugs

Tool	Bug Type					
	Range					
	2^0	2^7	2^{14}	2^{21}	2^{28}	KT
FUZZER	0	0	9%	79%	75%	20%
SES	8%	0	9%	21%	0	10%

Conclusion

- PANDA is a mature platform capable of many interesting dynamic analyses
- Many projects not mentioned here:
  - Transparent SSL/TLS interception by reading out keys from memory
  - Offline provenance tracing ("how was this document derived?")
  - Live visualizations of memory accesses

Credits

- PANDA devs
  - Tim Leek (MIT Lincoln Lab)
  - Patrick Hulin (MIT Lincoln Lab)
  - Josh Hodosh (MIT Lincoln Lab)
  - Ryan Whelan (MIT Lincoln Lab)
- Contributors
  - Manolis Stamatogiannakis (VU University Amsterdam)
  - Federico Scrinzi (EIT ICTLabs Master School / Google)
  - Evan Downing (Georgia Tech)

Contact

- Get in touch! @moyix on Twitter
  brendandg@nyu.edu
- Join the mailing list: panda-users@mit.edu
- IRC Channel: #panda-re on Freenode
- Contribute code:
  https://github.com/moyix/panda