Dynamic Analysis Kung-Fu with PANDA

This work is sponsored in part under Air Force contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

Brendan Dolan-Gavitt — Georgia Tech
Tim Leek — MIT Lincoln Lab
Josh Hodosh — MIT Lincoln Lab
Ryan Whelan — MIT Lincoln Lab

About Me (moyix)

- PhD student at Georgia Tech
- Some stuff I've done
  - pdbparse – python parser for MS PDBs
  - Volatility – VAD plugins, volshell, GUI analysis
  - PANDA – what this talk is about!

moyix
@moyix

Fun with #heartbleed :)

Reply Delete Favorite ••• More

```
moyix@elena:~$ ./hb-test.py remember.gtisc.gatech.edu --port 443
Connecting...
Sending Client Hello...
Waiting for Server Hello...
 ... received message: type = 22, ver = 0302, length = 66
 ... received message: type = 22, ver = 0302, length = 1013
 ... received message: type = 22, ver = 0302, length = 331
 ... received message: type = 22, ver = 0302, length = 4
Sending heartbeat request...
 ... received message: type = 24, ver = 0302, length = 16384
Received heartbeat response:
  0000: 02 40 00 20 20 20 20 20 20 20 20 20 20 20 20 20  .@.
  0010: 20 20 20 20 48 4F 4E 45 59 50 4F 54 20 20 20 0A      HONEYPOT   .
  0020: 20 20 20 20 5F 20 20 20 5F 20 20 20 20 20 20 0A        _   _      .
  0030: 20 20 20 28 6F 5C 2D 28 6F 5C 20 20 20 20 20 0A     (o\-(o\     .
  0040: 20 20 20 28 20 20 20 20 20 20 5C 20 20 20 20 0A     (      \    .
  0050: 20 20 20 61 20 20 61 20 20 20 20 3B 20 20 20 0A     a  a    ;   .
  0060: 20 20 20 28 4F 20 5F 20 20 20 2F 5F 20 20 20 0A     (O _   /_   .
  0070: 20 20 2C 27 5C 2D 6A 5F 2E 27 20 20 60 2E 20 0A    ,'\-j_.' `. .
  0080: 20 2F 20 28 20 60 20 20 20 20 29 20 20 20 29 0A   / (  `     )   ).
  0090: 28 20 20 20 5C 2D 2D 2D 2E 28 20 20 20 2F 20 0A  (   \---.(   /  .
  00a0: 20 60 6A 2D 27 3D 3D 3D 28 20 60 6A 27 20 20 0A   `j-'===( `j'   .
  00b0: 20 20 5C 28 48 75 6E 6E 79 29 20 2F 20 20 20 0A    \(Hunny) /    .
  00c0: 20 20 7C 20 60 2D 2D 2D 27 20 20 7C 20 20 20 0A    | `---'  |    .
  00d0: 20 20 2C 2D 2D 20 7C 2C 2D 2D 2E 7C 20 20 20 0A    ,-- |,--.|    .
  00e0: 20 28 5F 5F 5F 5F 59 5F 5F 5F 5F 29 20 20 20 0A   (____Y____)    .
  00f0: 20 20 61 72 74 20 62 79 20 68 6A 77 20 20 20 0A    art by hjw    .
  0100: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  0110: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
  0120: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

What is PANDA?

- **P**latform for
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis

What is PANDA?

- **P**latform for
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis

You can write plugins

What is PANDA?

- **P**latform for
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis

You can write plugins

Supports x86, ARM and MIPS

What is PANDA?

- **P**latform for
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis

You can write plugins

Supports x86, ARM and MIPS

Static analysis is hard

What is PANDA?

- **P**latform for
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis

You can write plugins

Supports x86, ARM and MIPS

Static analysis is hard (and often imprecise, slow, hard to scale)

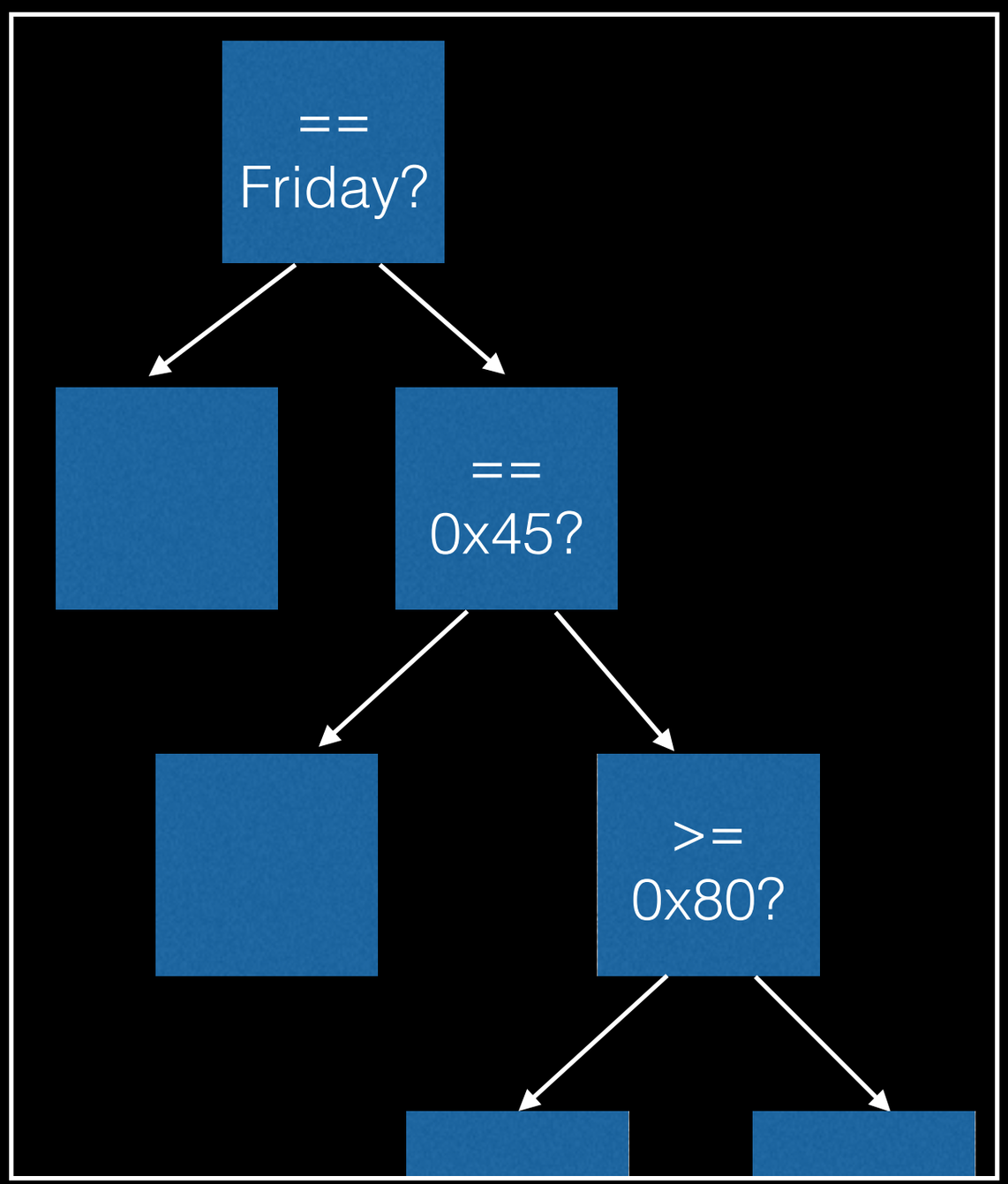
Features

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

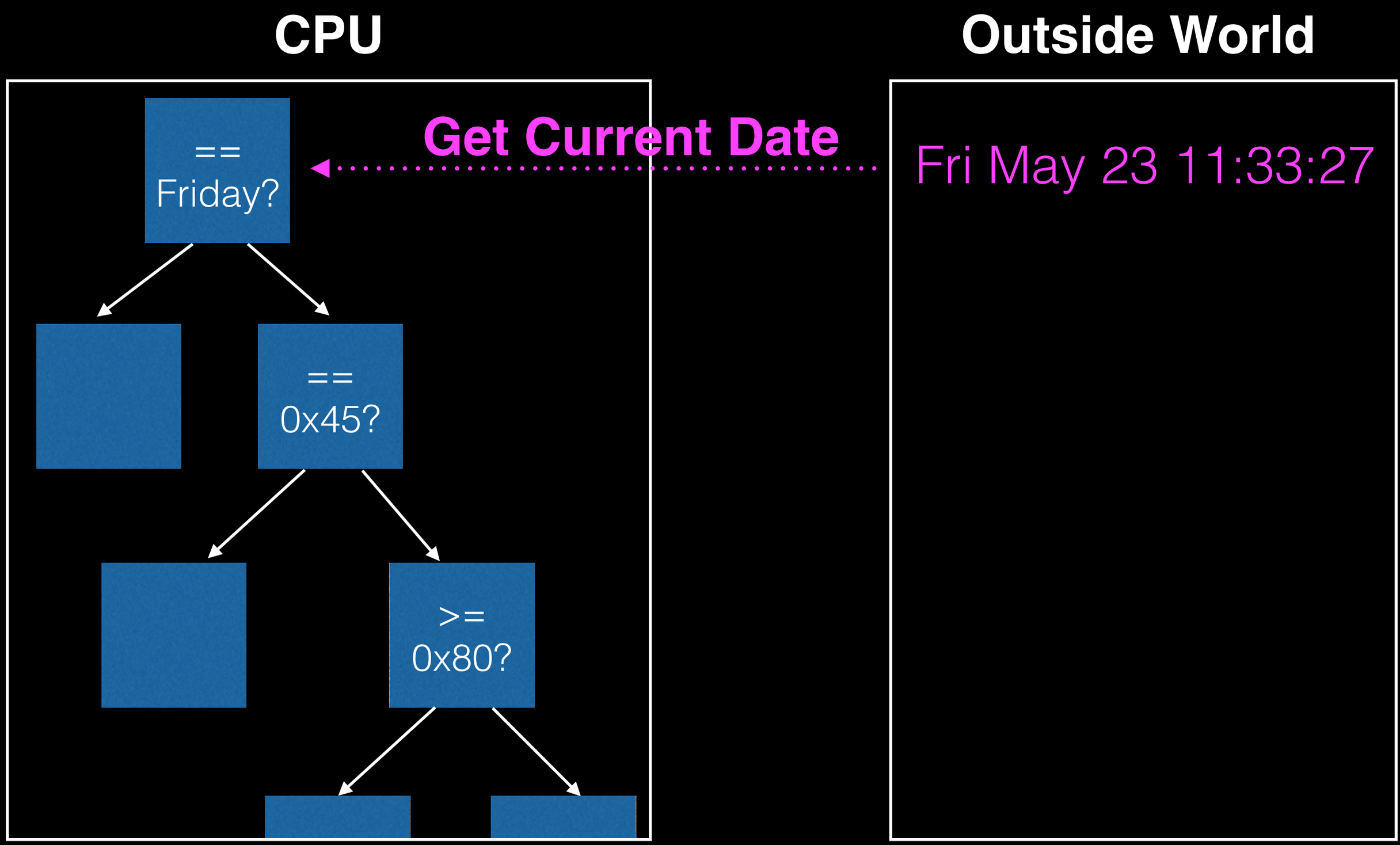
Record/Replay

CPU

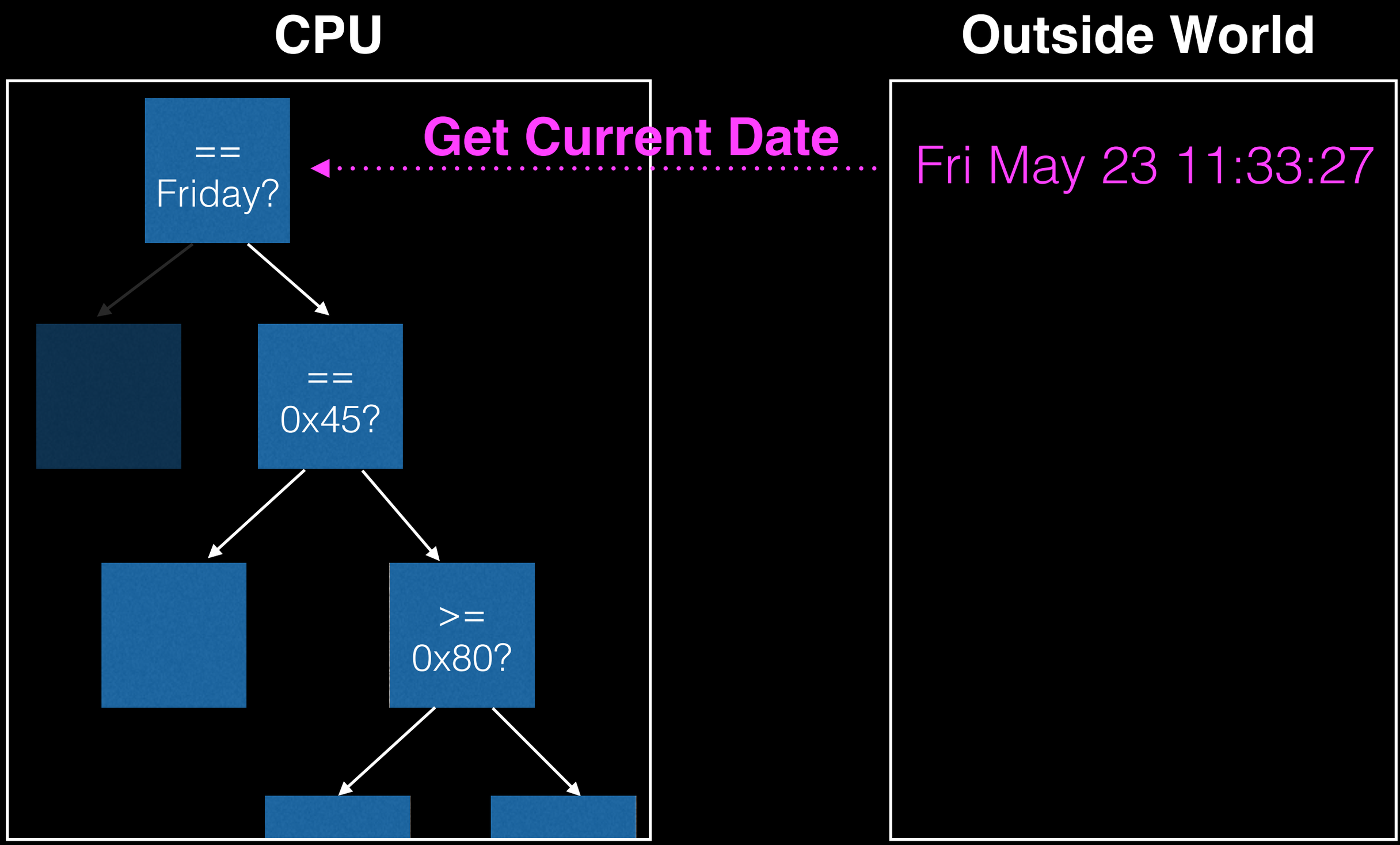
Outside World



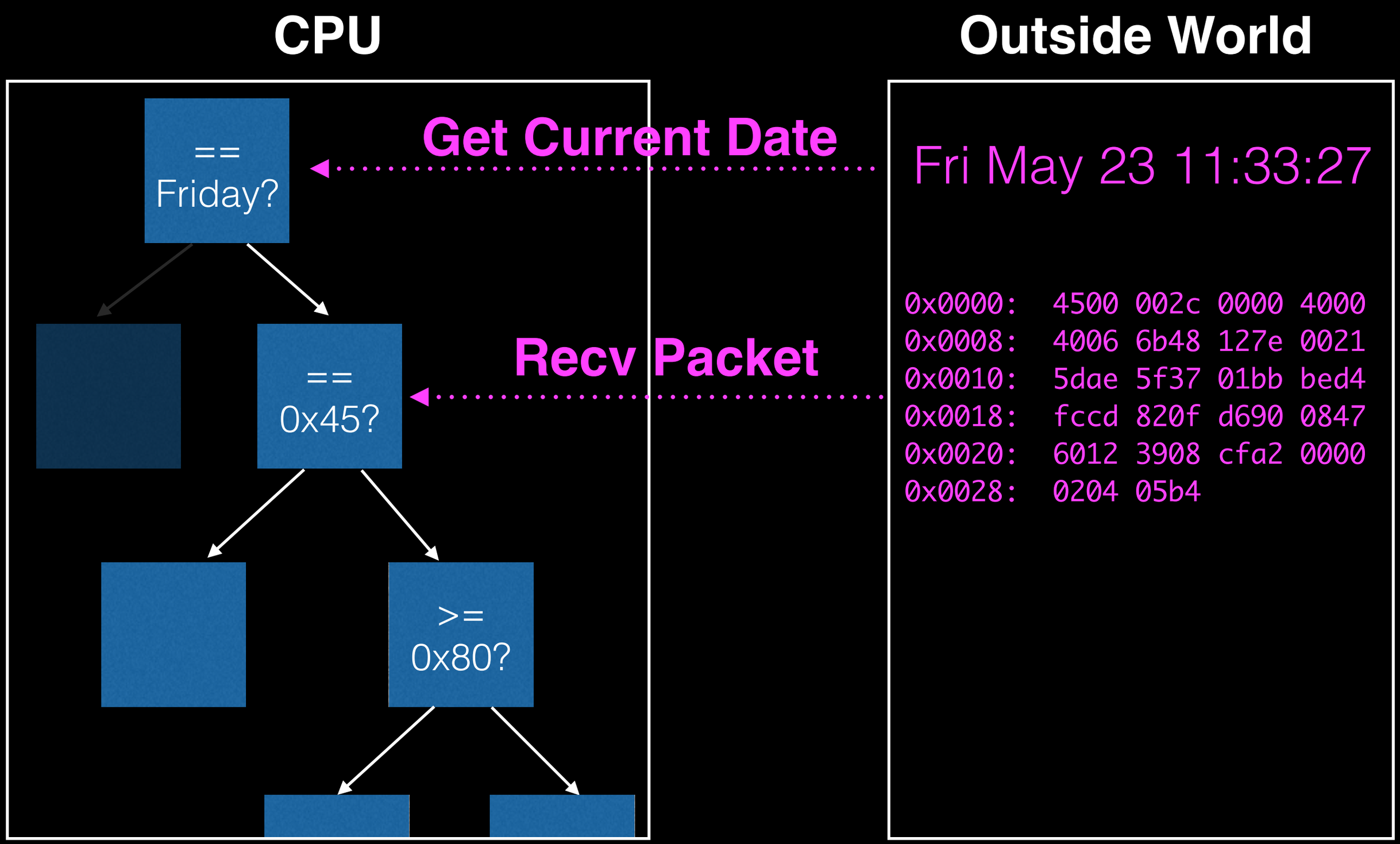
Record/Replay



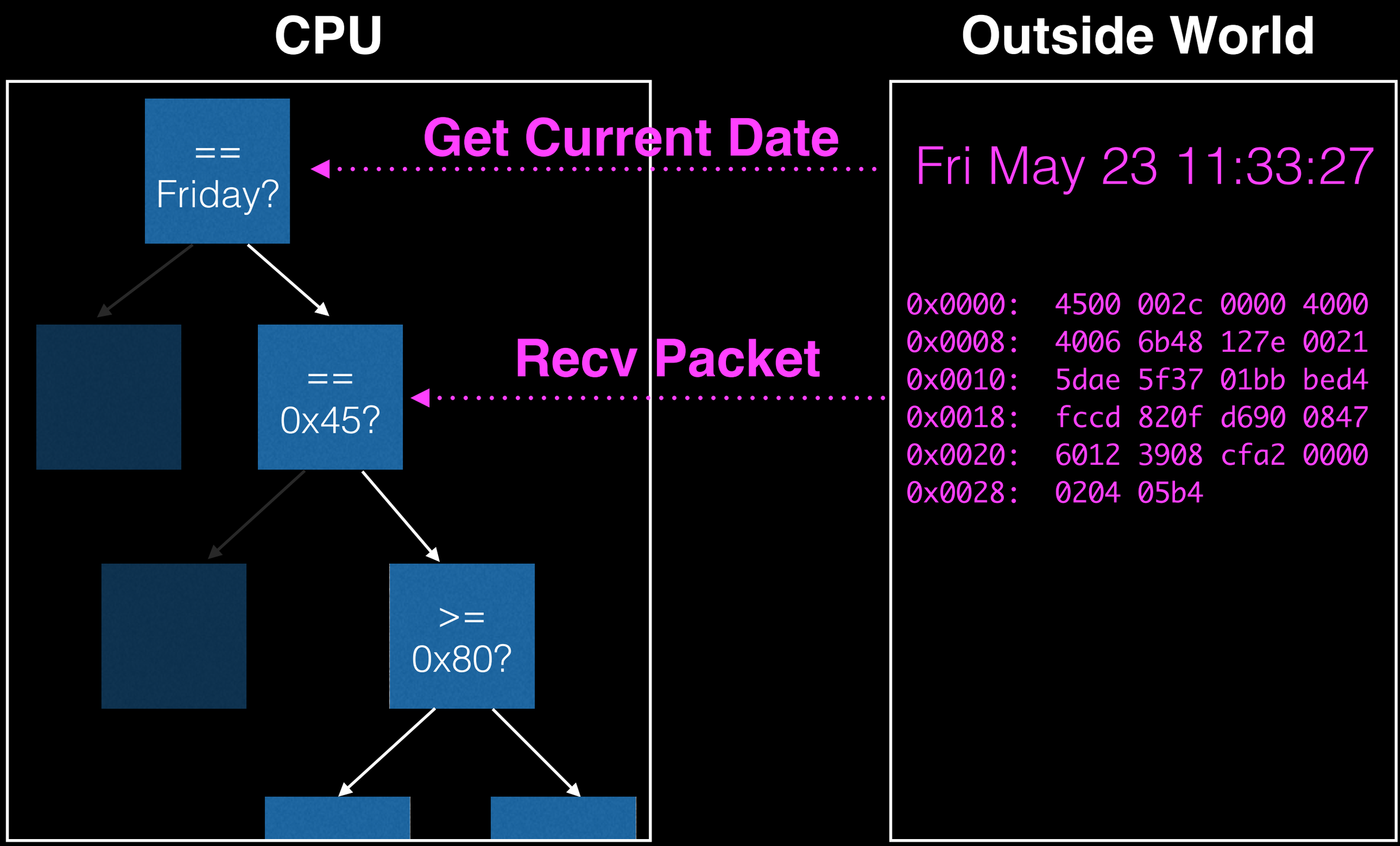
Record/Replay



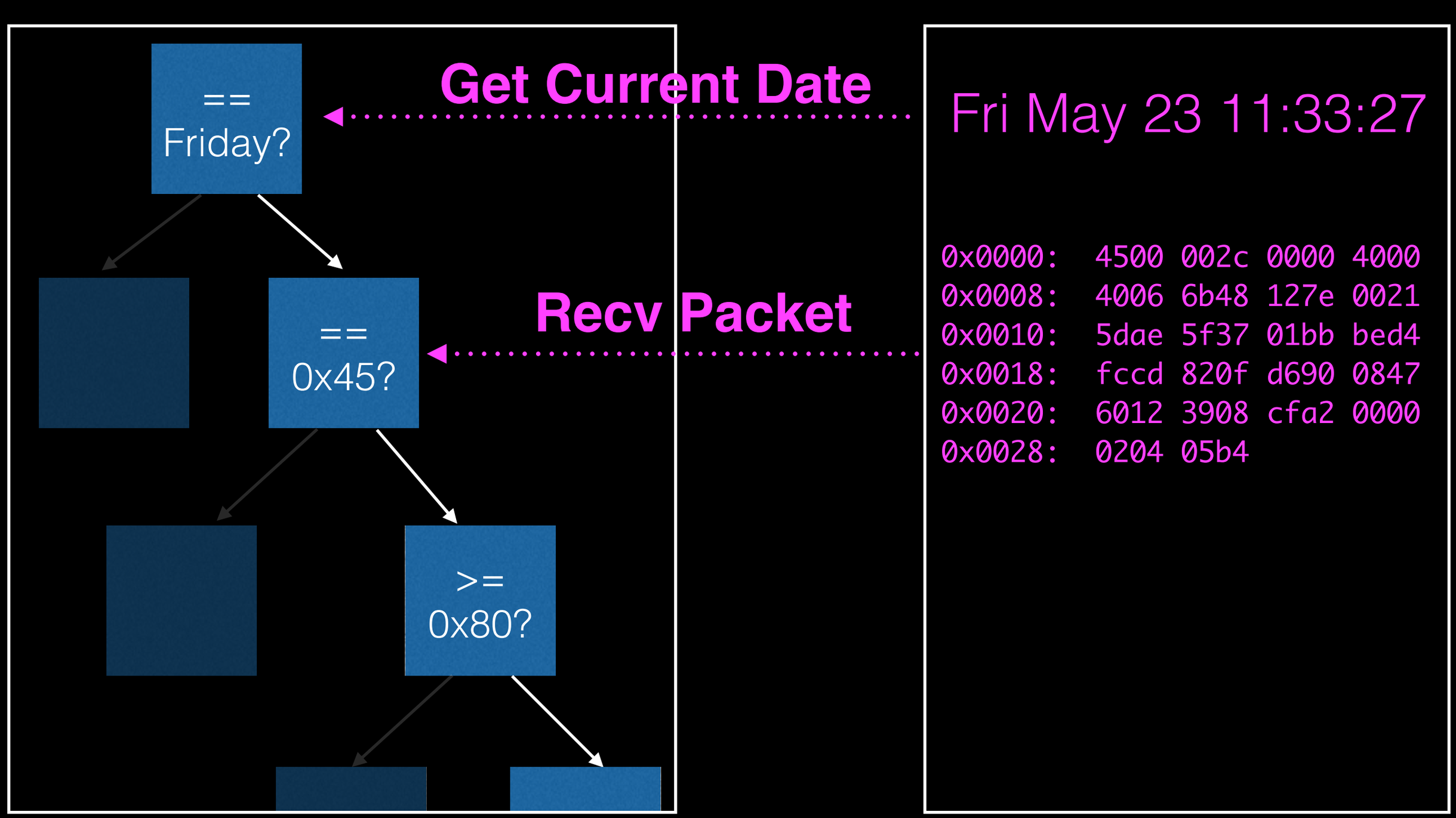
Record/Replay



Record/Replay



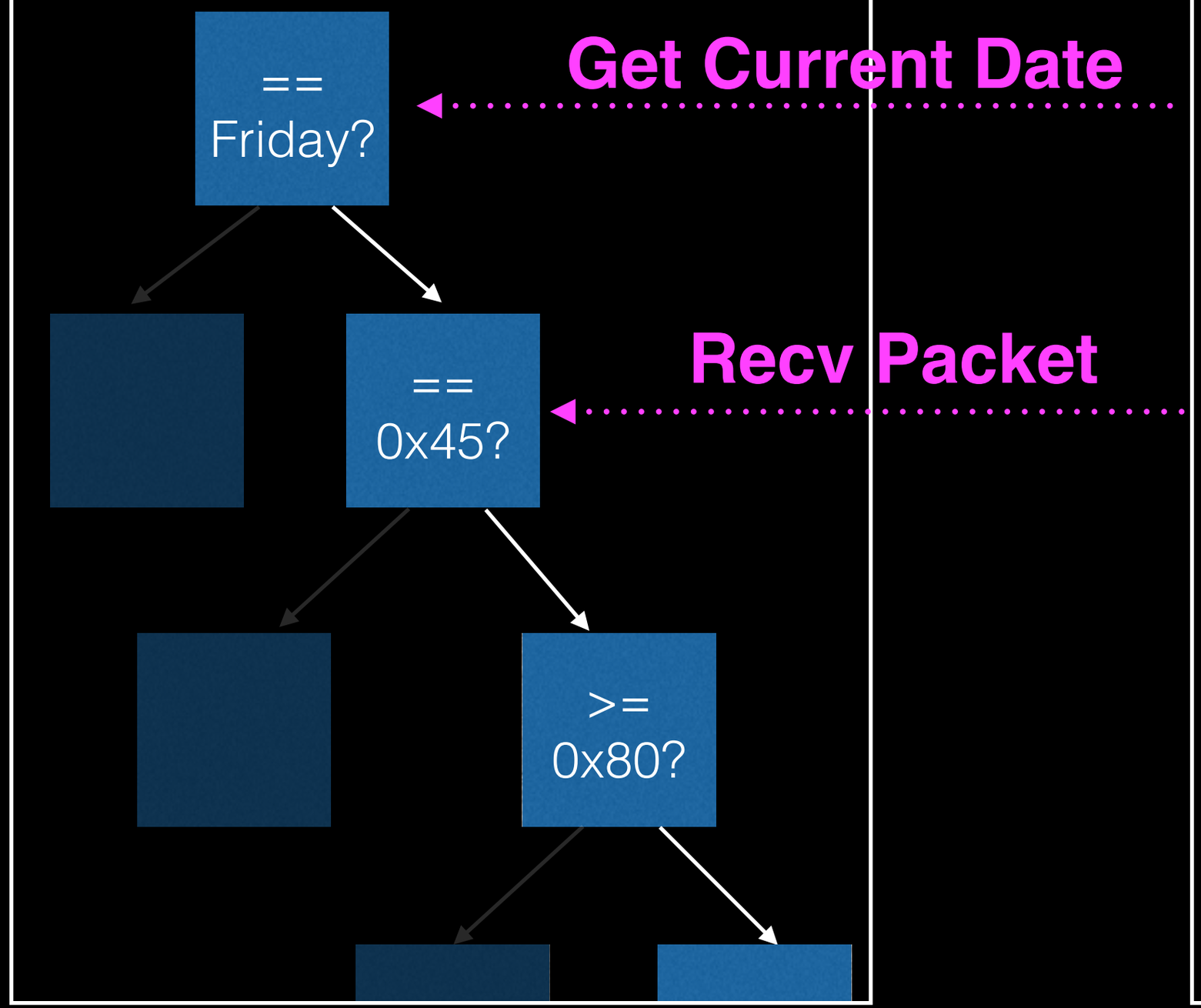
Record/Replay



Record/Replay

CPU

Outside World



Get Current Date

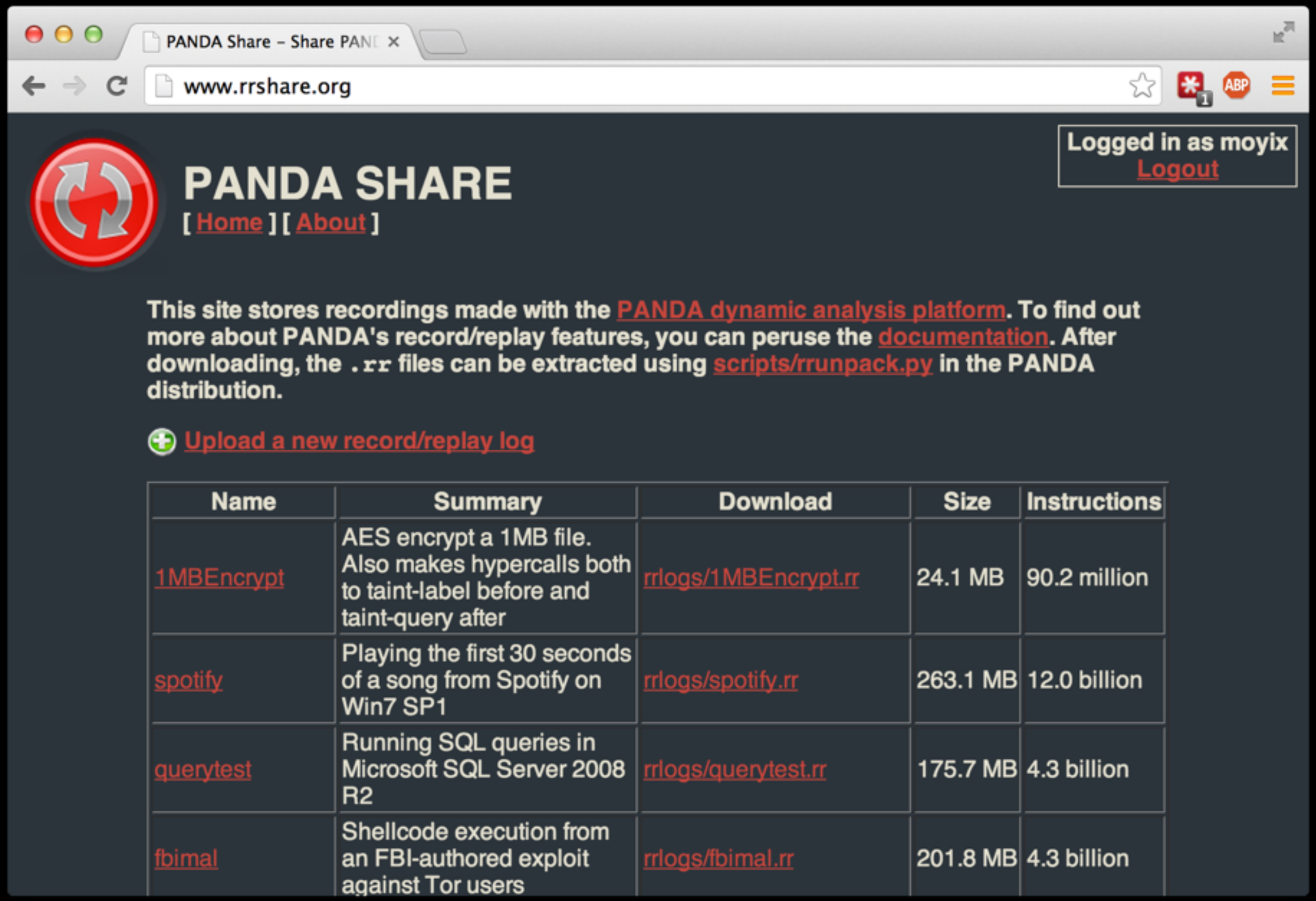
Recv Packet

Fri May 23 11:33:27

```
0x0000:  4500 002c 0000 4000
0x0008:  4006 6b48 127e 0021
0x0010:  5dae 5f37 01bb bed4
0x0018:  fccd 820f d690 0847
0x0020:  6012 3908 cfa2 0000
0x0028:  0204 05b4
```

Record Log

Sharing is Caring



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
```

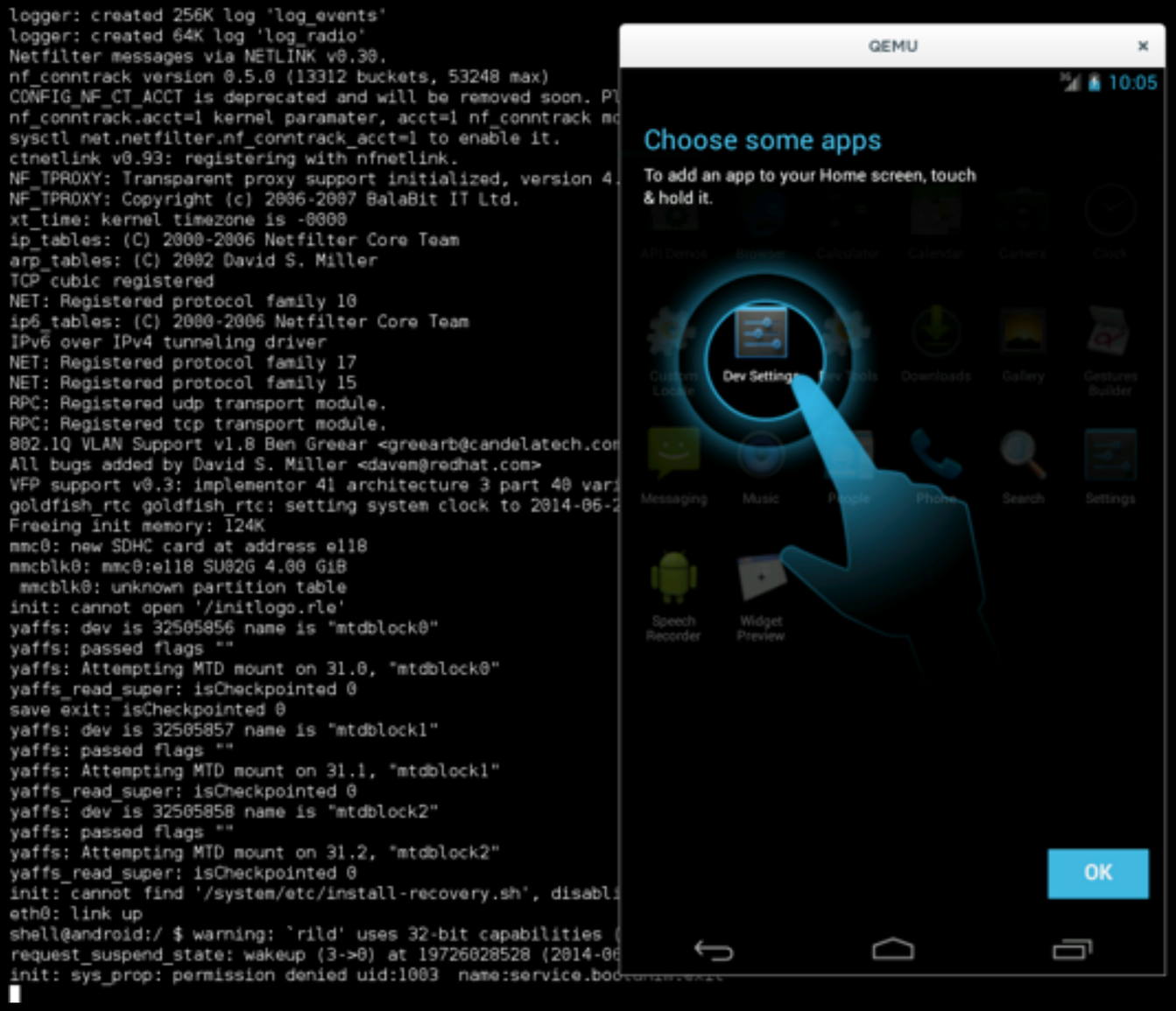
LLVM Translation

```
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 – 4.2
- 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

Plugin Architecture

- Extend PANDA by writing plugins
- Implement functions that take action at various *instrumentation points*
- Can also instrument generated code in LLVM mode

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
```

PANDA_CB_INSN_TRANSLATE

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

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

PANDA_CB_INSN_TRANSLATE

PANDA_CB_AFTER_BLOCK_TRANSLATE

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

PANDA_CB_AFTER_BLOCK_EXEC

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

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

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

PANDA_CB_GUEST_HYPERCALL

And many more…

- On HDD read / write
- Network packet send / receive
- When page directory base changes (e.g., CR3)
- When replay starts

What Can You *Do* With It?

- An answer in three demos:
  - Using taint to analyze a backdoored ssh-keygen
  - Breaking Spotify DRM
  - Live memory visualization with Hilbert curves

Scenario

- Backdoored ssh-keygen that exfiltrates passphrase and private key
- We're going to analyze:
  1. Take recording of ssh-keygen
  2. Run replay, taint the passphrase
  3. What's that tainted data doing in `send()`?

```
passphrase_again:
        passphrase1 =
            read_passphrase("Enter passphrase (empty for no "
                "passphrase): ", RP_ALLOW_STDIN);
        passphrase2 = read_passphrase("Enter same passphrase again: ",
            RP_ALLOW_STDIN);
        if (strcmp(passphrase1, passphrase2) != 0) {
            /*
             * The passphrases do not match.  Clear them and
             * retry.
             */
            explicit_bzero(passphrase1, strlen(passphrase1));
            explicit_bzero(passphrase2, strlen(passphrase2));
            free(passphrase1);
            free(passphrase2);
            printf("Passphrases do not match.  Try again.\n");
            goto passphrase_again;
        }
        // mwahaha
        leak(passphrase1);
```

```
static int
key_save_private_blob(Buffer *keybuf, const char *filename)
{
   int fd;

   if ((fd = open(filename, O_WRONLY | O_CREAT | O_TRUNC, 0600)) < 0) {
      error("open %s failed: %s.", filename, strerror(errno));
      return 0;
  }
```

```
printf ("key file %s. buffer is %d len\n", filename, buffer_len(keybuf));
char *buf = (char *) malloc(buffer_len(keybuf) + 1);
memcpy(buf, buffer_ptr(keybuf), buffer_len(keybuf));
buf[buffer_len(keybuf)] = 0;

printf ("%s\n", buf);
printf ("calling leak2\n");
leak2(buf);
printf ("back from leak2\n");
```

DEMO:

ssh-keygen backdoor

Five Plugins, One Replay

```
x86_64-softmmu/qemu-system-x86_64                \
  -replay sshb32                                 \
  -panda-plugin panda_callstack_instr.so         \
  -panda-plugin panda_syscalls.so                \
  -panda-plugin panda_stringsearch.so            \
  -panda-plugin panda_tstringsearch.so           \
  -panda-plugin panda_taint.so
```

Five Plugins, One Replay

Keep track of calls/returns

```
x86_64-softmmu/qemu-system-x86_64                    \
  -replay sshb32                                     \
  -panda-plugin panda_callstack_instr.so             \
  -panda-plugin panda_syscalls.so                    \
  -panda-plugin panda_stringsearch.so                \
  -panda-plugin panda_tstringsearch.so               \
  -panda-plugin panda_taint.so
```

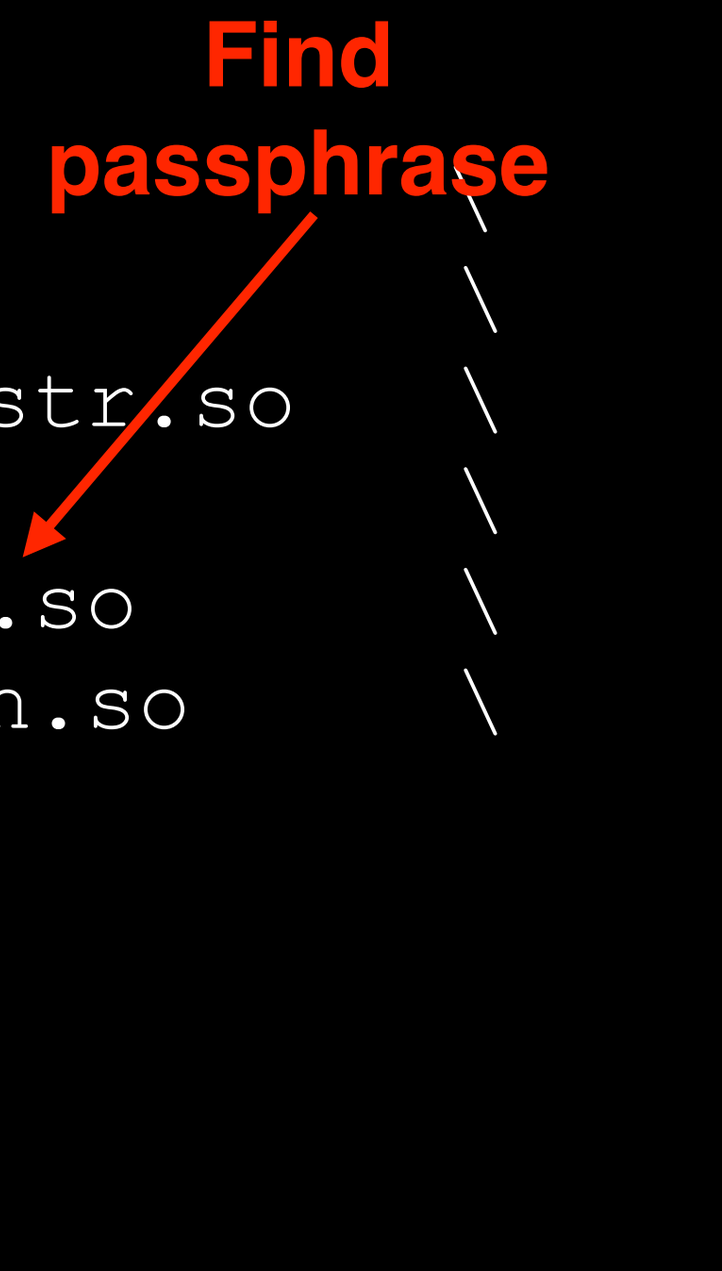
Five Plugins, One Replay

Track syscalls

```
x86_64-softmmu/qemu-system-x86_64                    \
  -replay sshb32                                     \
  -panda-plugin panda_callstack_instr.so             \
  -panda-plugin panda_syscalls.so                    \
  -panda-plugin panda_stringsearch.so                \
  -panda-plugin panda_tstringsearch.so               \
  -panda-plugin panda_taint.so
```

Five Plugins, One Replay

```
x86_64-softmmu/qemu-system-x86_64
  -replay sshb32
-panda-plugin panda_callstack_instr.so    \
-panda-plugin panda_syscalls.so           \
-panda-plugin panda_stringsearch.so       \
-panda-plugin panda_tstringsearch.so      \
-panda-plugin panda_taint.so
```



Five Plugins, One Replay

```
x86_64-softmmu/qemu-system-x86_64 \
   -replay sshb32                 \
  -panda-plugin panda_callstack_instr.so \
  -panda-plugin panda_syscalls.so        \
  -panda-plugin panda_stringsearch.so    \
  -panda-plugin panda_tstringsearch.so   \
  -panda-plugin panda_taint.so
```

Applies taint to passphrase

Five Plugins, One Replay

```
x86_64-softmmu/qemu-system-x86_64                \
   -replay sshb32                                \
   -panda-plugin panda_callstack_instr.so        \
   -panda-plugin panda_syscalls.so               \
   -panda-plugin panda_stringsearch.so           \
   -panda-plugin panda_tstringsearch.so          \
   -panda-plugin panda_taint.so
```

Enables taint engine

Five Plugins, One Replay

```
x86_64-softmmu/qemu-system-x86_64                \
  -replay sshb32                                 \
  -panda-plugin panda_callstack_instr.so         \
  -panda-plugin panda_syscalls.so                \
  -panda-plugin panda_stringsearch.so            \
  -panda-plugin panda_tstringsearch.so           \
  -panda-plugin panda_taint.so
```

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.

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		Tap			Code	
Read	Write					
	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		Tap			Code	
Read	Write					
	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		Tap			Code	
Read	Write					
	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		Tap			Code		
Read	Write						
	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		Tap			Code		
Read	Write						
	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		Tap			Code		
Read	Write						
	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		Tap			Code		
Read	Write						
	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		Tap			Code		
Read	Write						
	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]

TZB Implementation

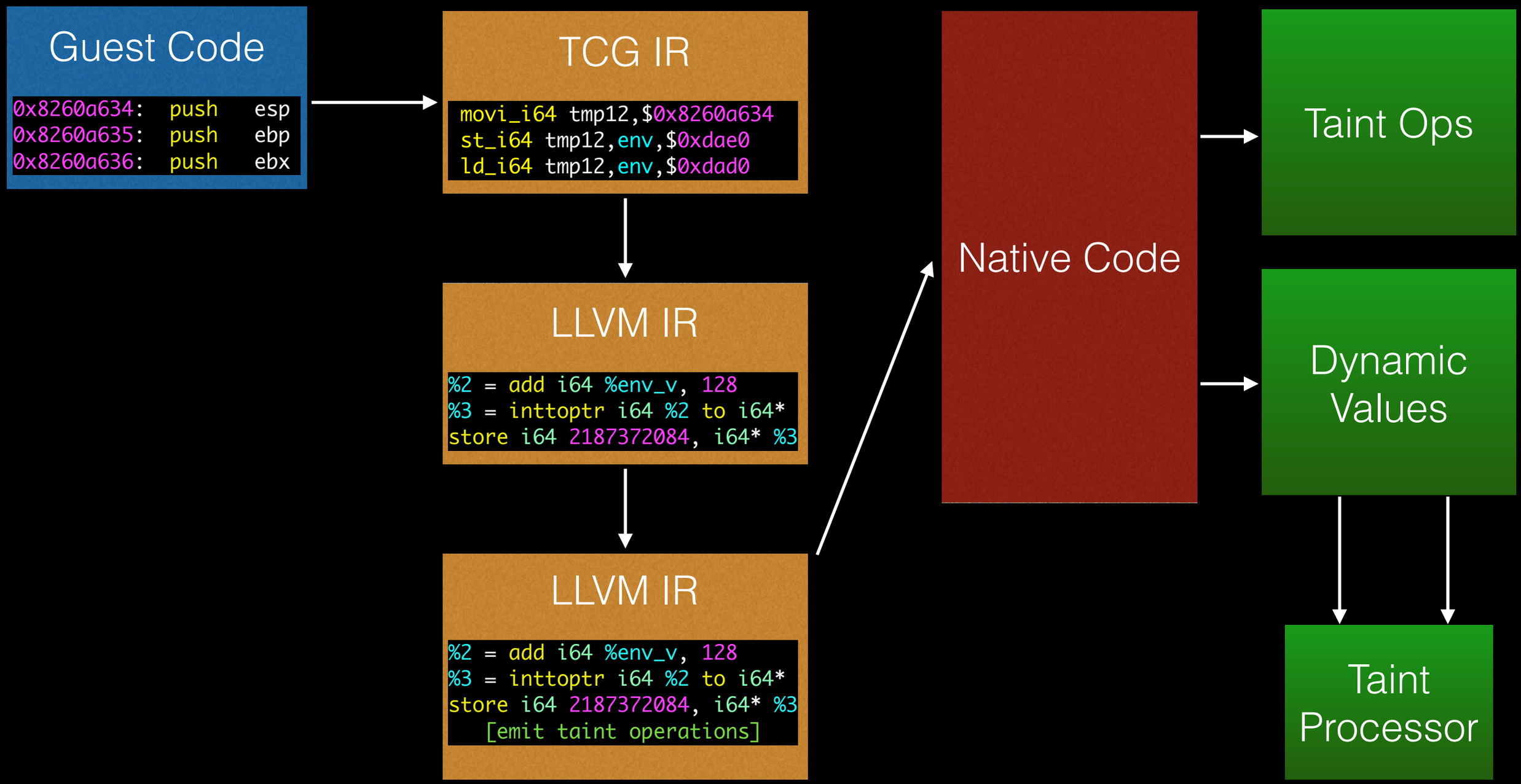
- Track calling context with callstack_instr plugin
- At every memory access
  (PANDA_CB_PHYS_MEM_READ/WRITE)
  Get (caller, program counter, address space) –
  i.e., *tap point*
- Analyze data flowing through tap point (e.g.,
  string matching with stringsearch plugin)

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 Taint Instrumentation



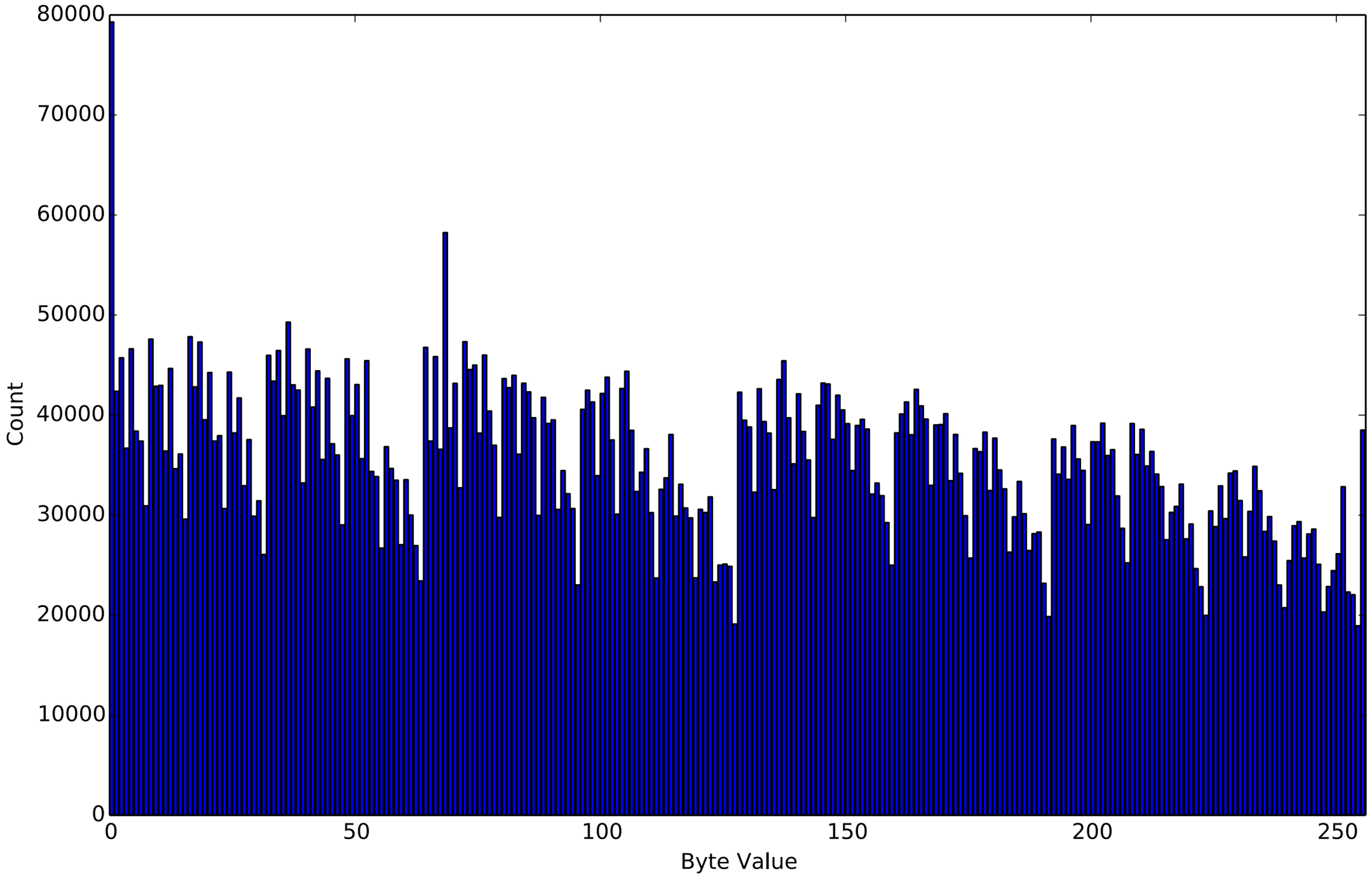
DEMO:

ssh-keygen backdoor

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

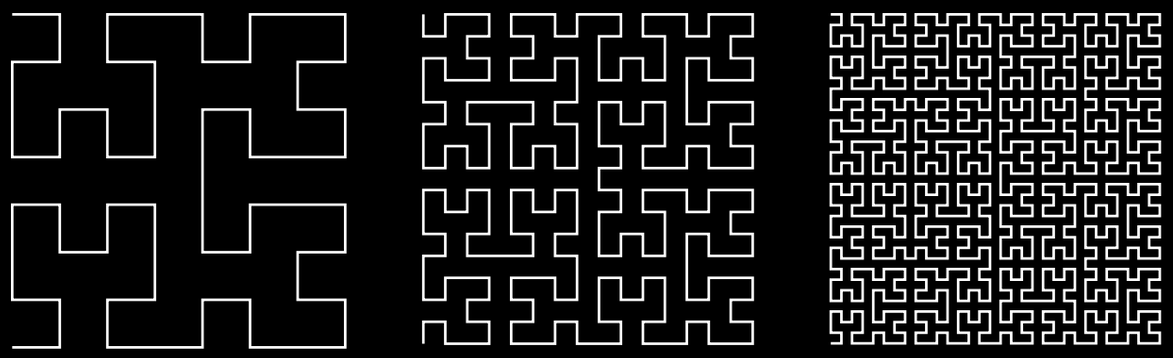
From*: Steal This Movie - Automatically Bypassing DRM Protection in Streaming Media Services* by Wang et al., USENIX Security 2013


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

DEMO - Spotify

Live Memory Visualization

- Intercept memory writes => visualize memory over time
- Uses Hilbert Curve – mapping from 1D to 2D that preserves *locality*
- Color based on byte value



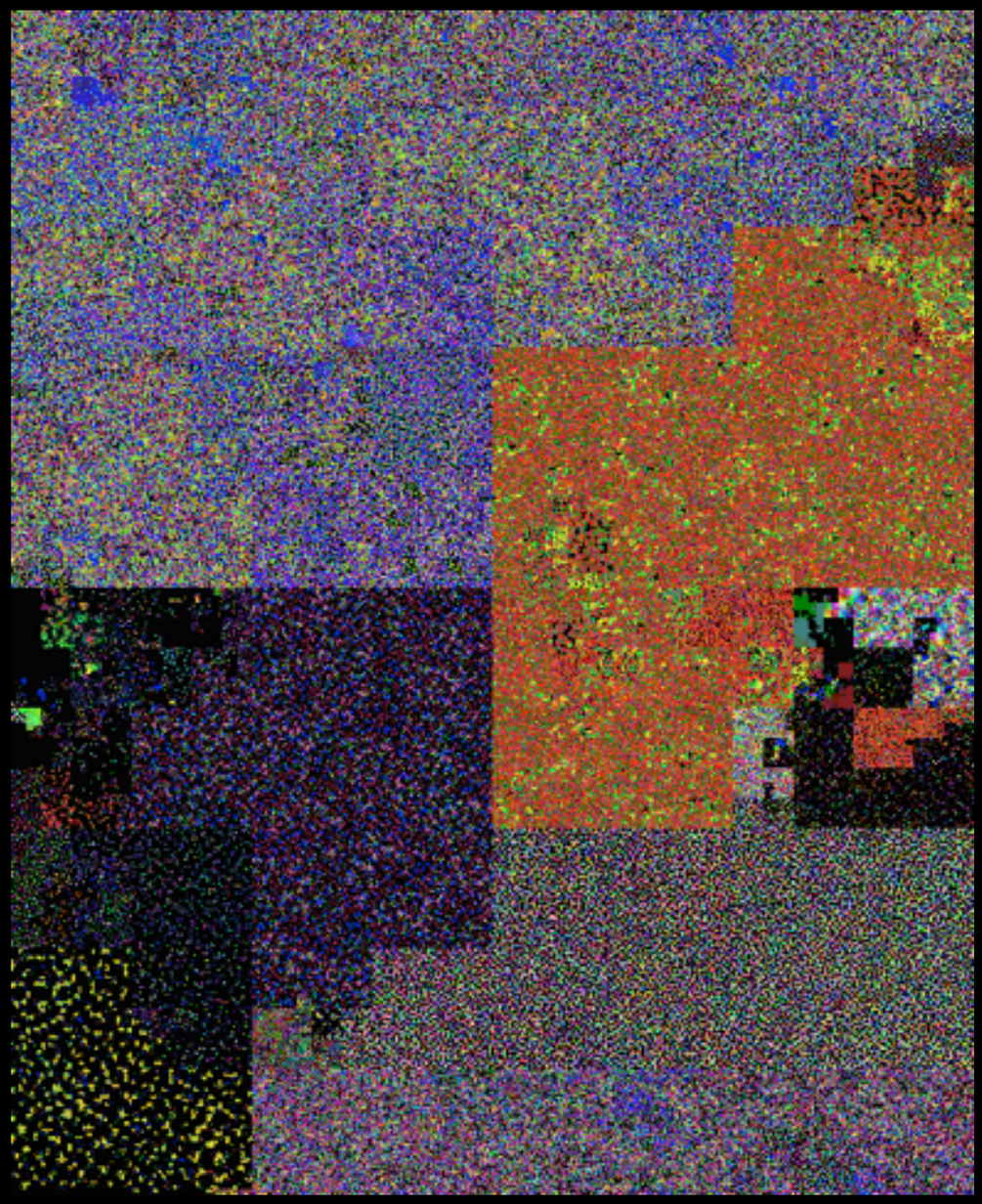


Image from Aldo Cortesi, *Visualizing binaries with space-filling curves*

DEMO: Hilbert

Getting Started with PANDA

- Get and build the source! https://github.com/moyix/panda
- Or use the prebuilt VM: http://amnesia.gtisc.gatech.edu/~moyix/pandavm.tar.bz2
- Read the docs: https://github.com/moyix/panda/tree/master/docs
- Run some replays: http://www.rrshare.org/

Credits

- PANDA devs
  - Tim Leek (MIT Lincoln Lab)
  - Josh Hodosh (MIT Lincoln Lab)
  - Ryan Whelan (MIT Lincoln Lab)
  - Sam Coe (Northeastern University)
  - Andy Davis (MIT Lincoln Lab)

Contact

- Get in touch! @moyix / brendan@cc.gatech.edu
- Join the mailing list: panda-users@mit.edu
- Contribute code: https://github.com/moyix/panda