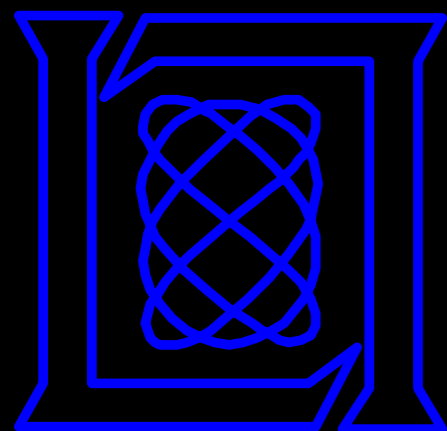


# Dynamic Analysis Kung-Fu with PANDA



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  
Tim Leek  
Josh Hodosh  
Ryan Whelan

Georgia Tech  
MIT Lincoln Lab  
MIT Lincoln Lab  
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!






# What is PANDA?

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






# What is PANDA?

- **P**latform for  **You can write plugins**
- **A**rchitecture
- **N**eutral
- **D**ynamic
- **A**nalysis






# What is PANDA?

- **Platform for**  **You can write plugins**
- **Architecture**  **Supports x86, ARM and MIPS**
- **Neutral** 
- **Dynamic**
- **Analysis**

# What is PANDA?

- **Platform for**  **You can write plugins**
- **Architecture**  **Supports x86, ARM and MIPS**
- **Neutral** 
- **Dynamic** 
- **Analysis**  **Static analysis is hard**

# What is PANDA?

- **Platform for**  **You can write plugins**
- **Architecture**  **Supports x86, ARM and MIPS**
- **Neutral** 
- **Dynamic** 
- **Analysis**  **Static analysis is hard (and often imprecise, slow, hard to scale)**



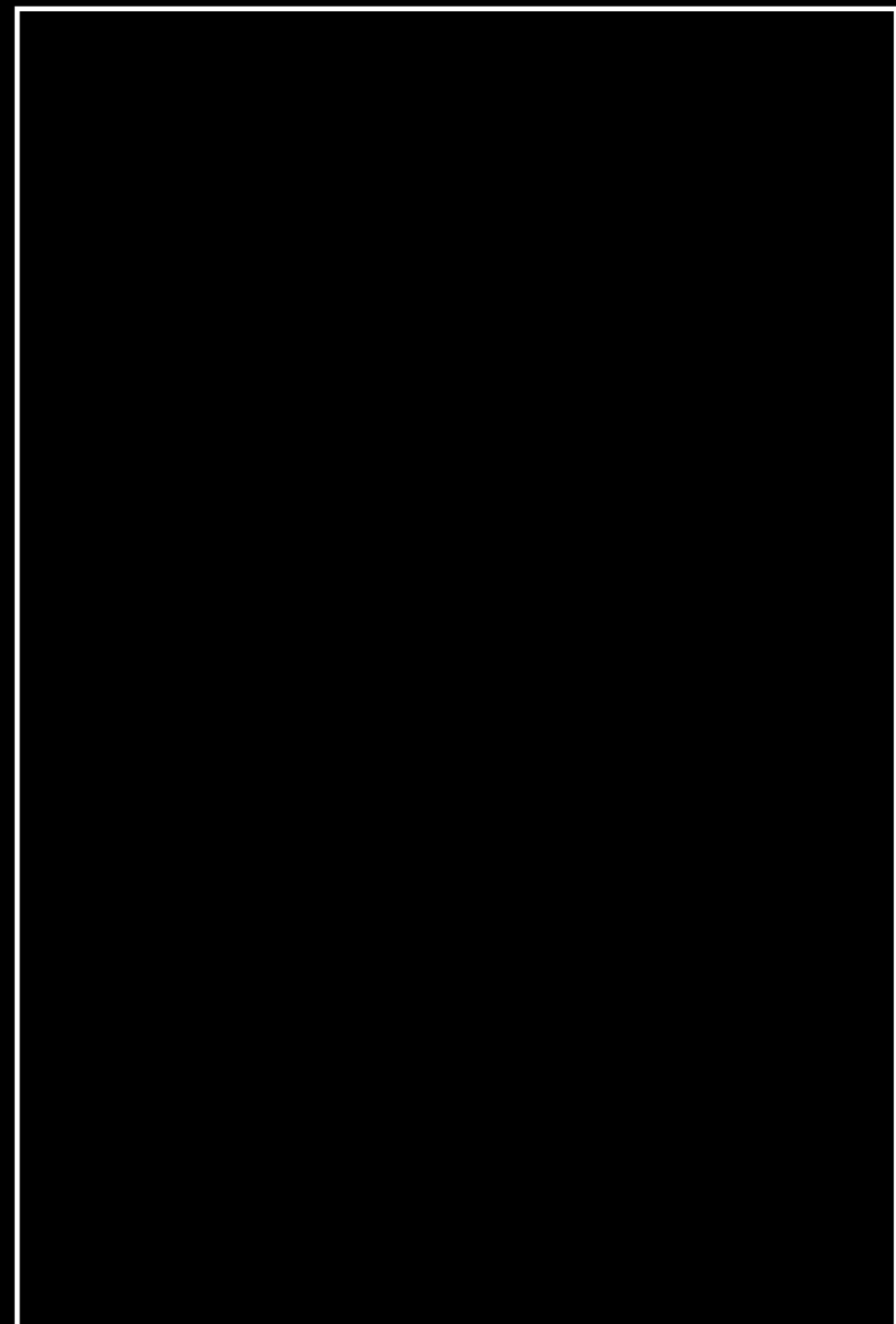
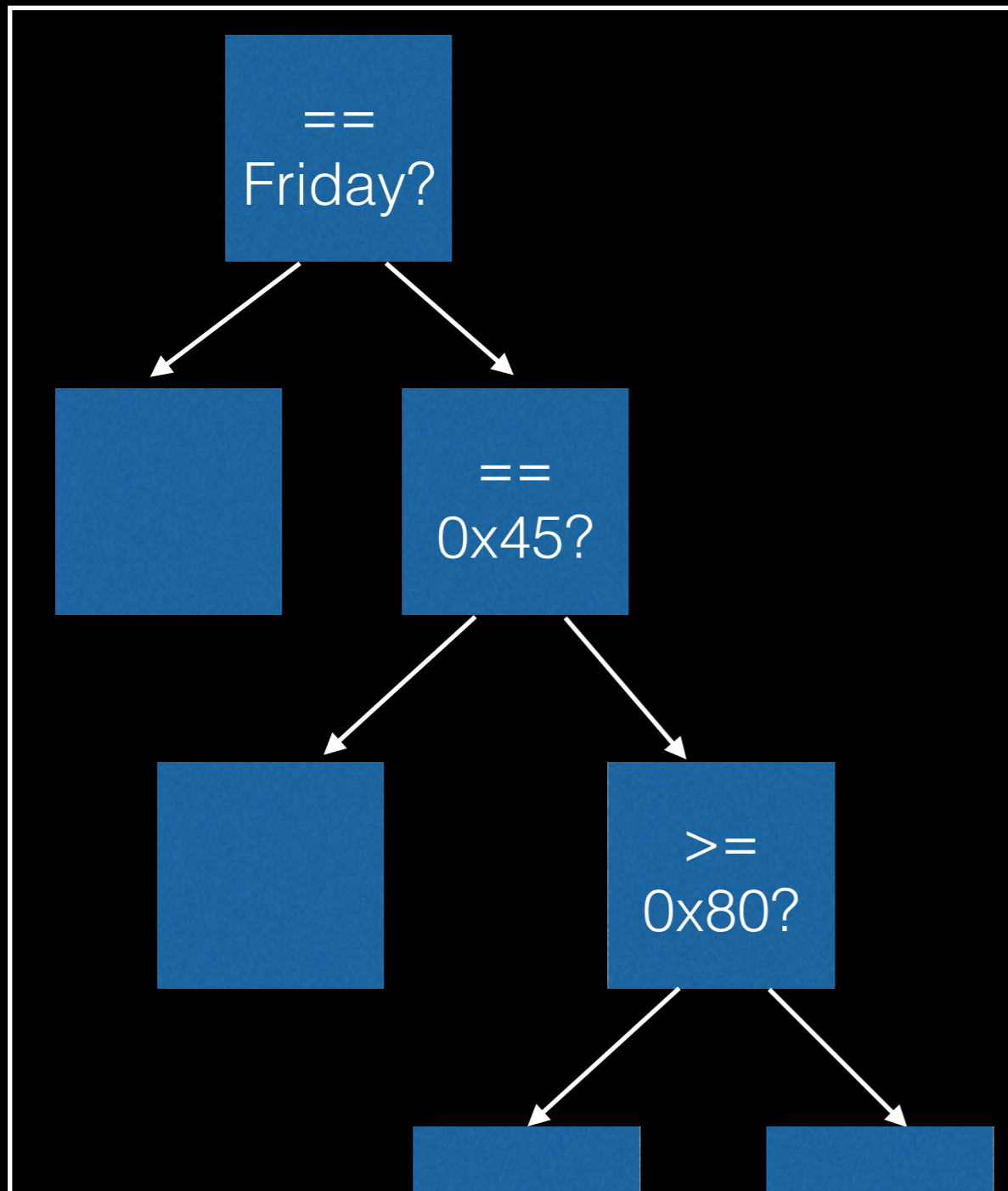
# Features

- 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**

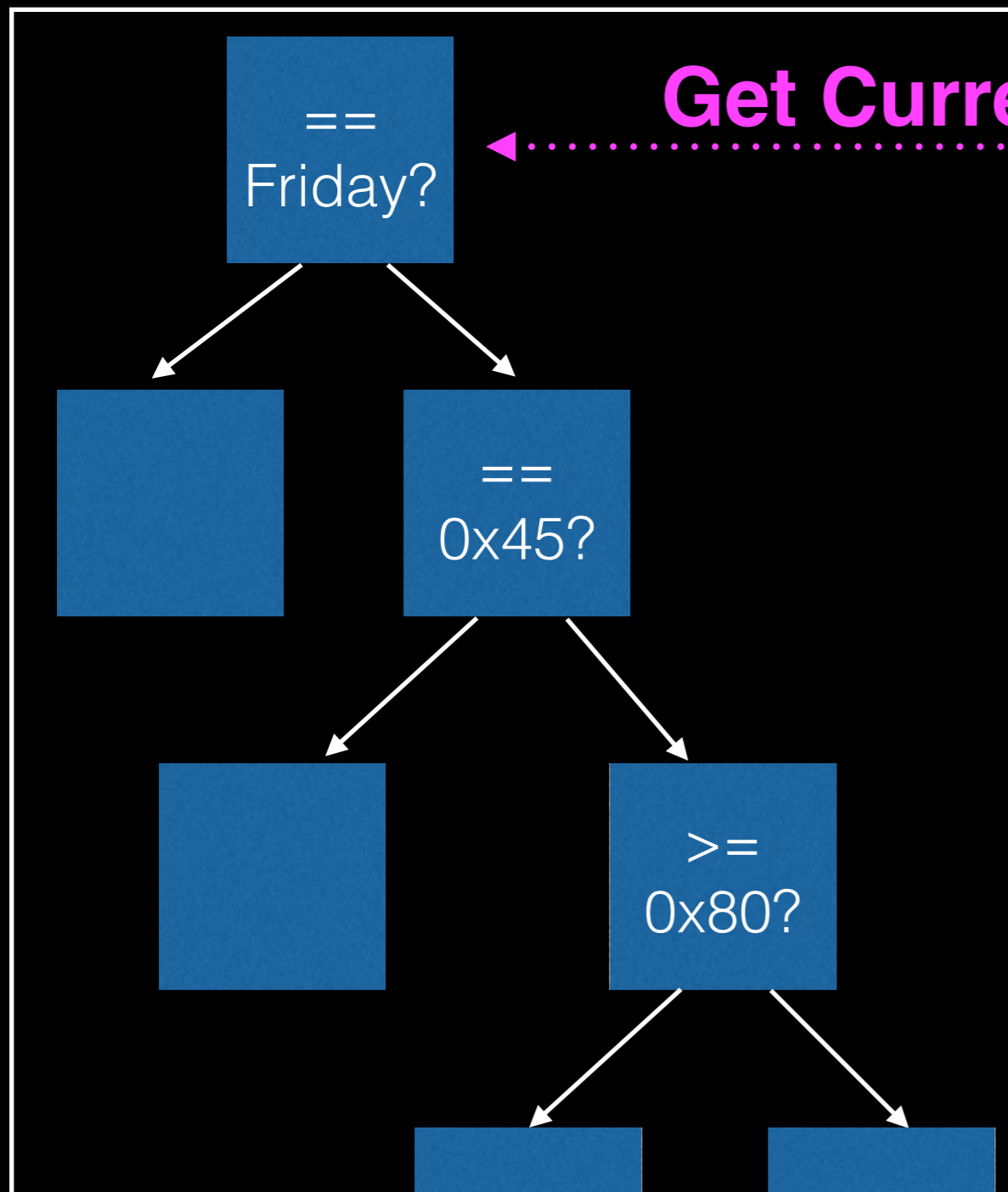
**Outside World**



# Record/Replay

**CPU**

**Outside World**



**Get Current Date**

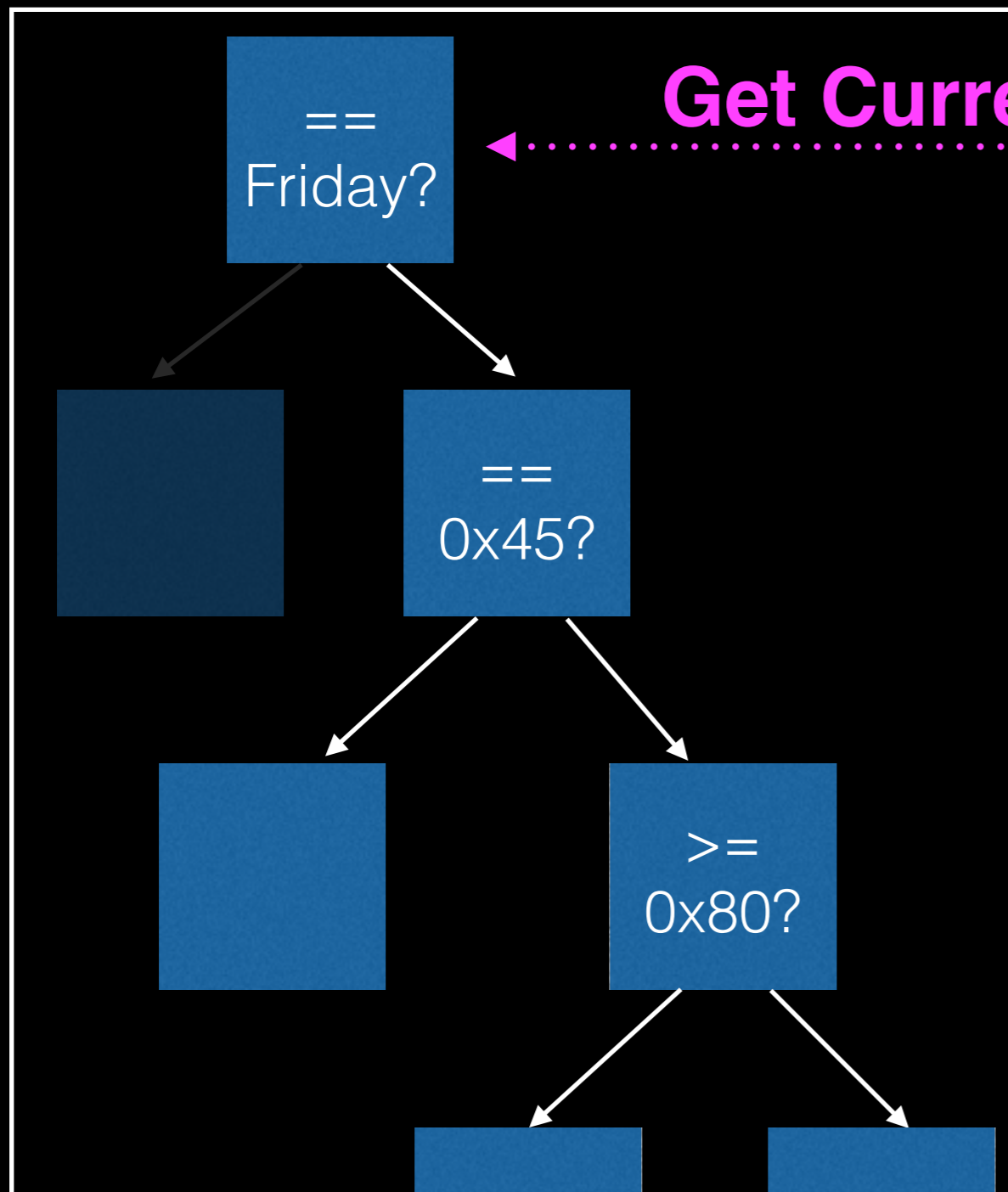


Fri May 23 11:33:27

# Record/Replay

**CPU**

**Outside World**



**Get Current Date**

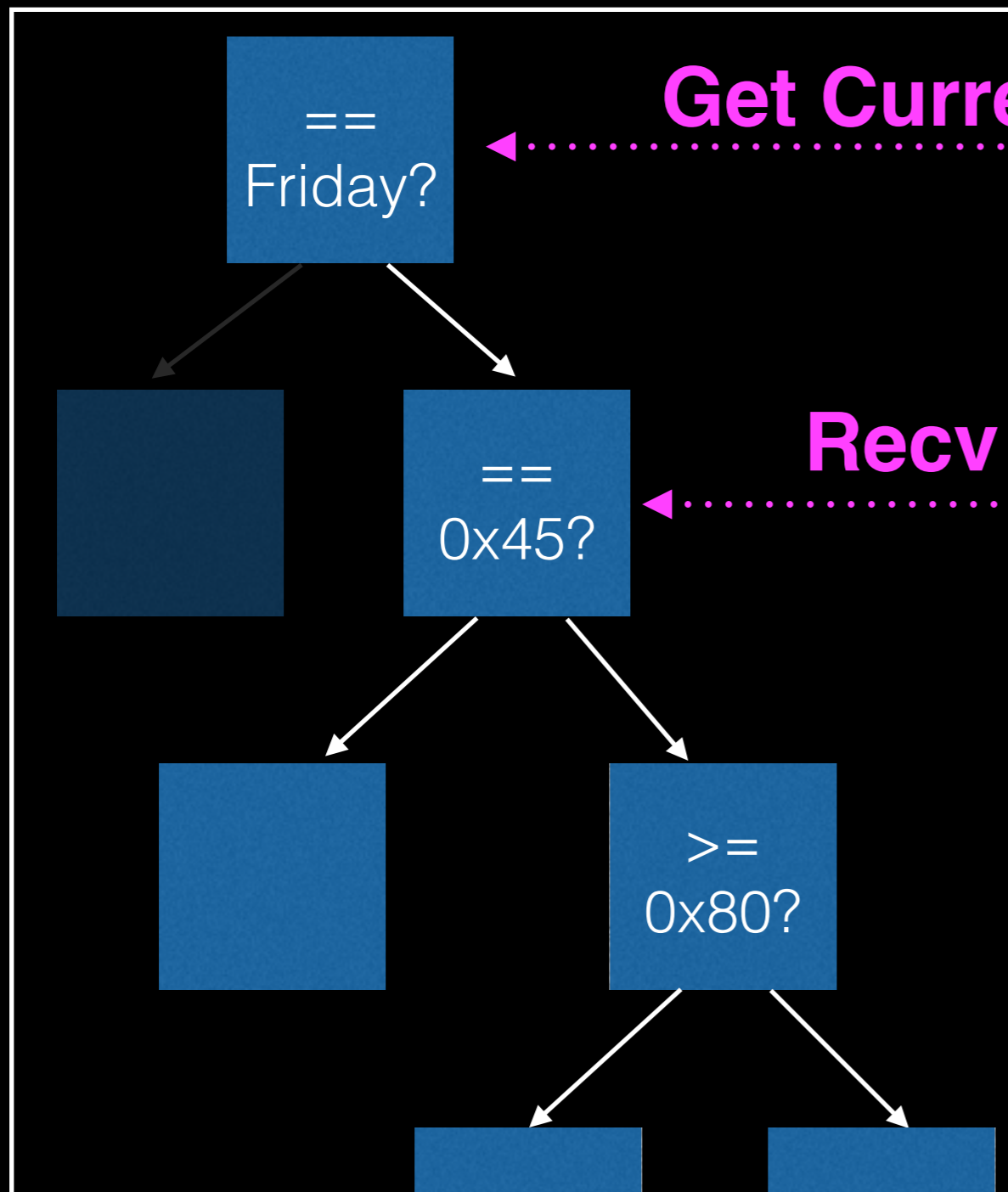


Fri May 23 11:33:27

# Record/Replay

**CPU**

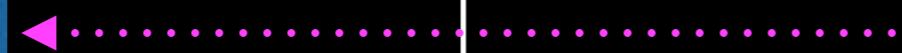
**Outside World**



**Get Current Date**



**Recv Packet**



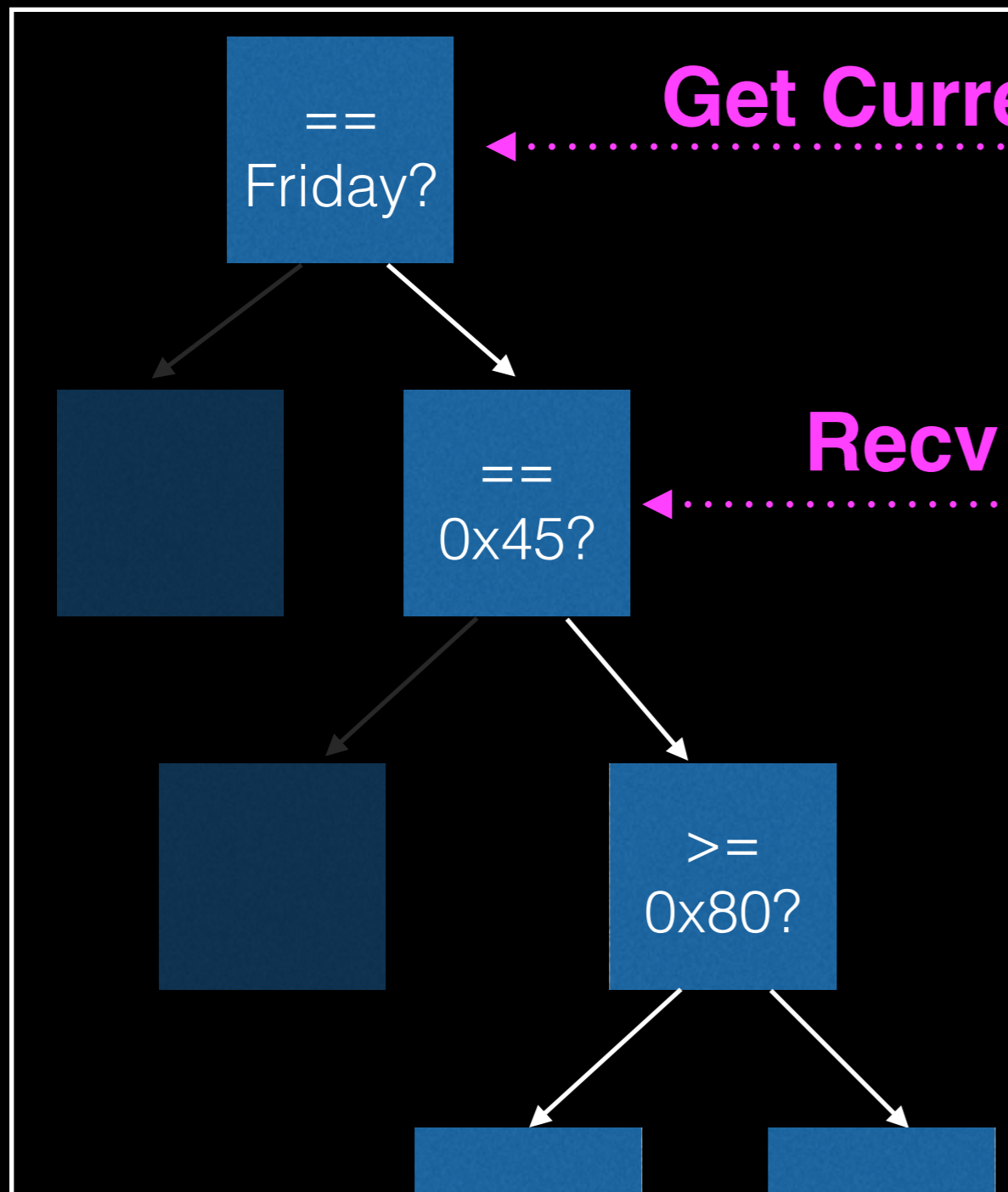
Fri May 23 11:33:27

```
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/Replay

**CPU**

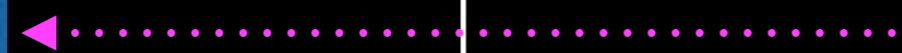
**Outside World**



**Get Current Date**



**Recv Packet**



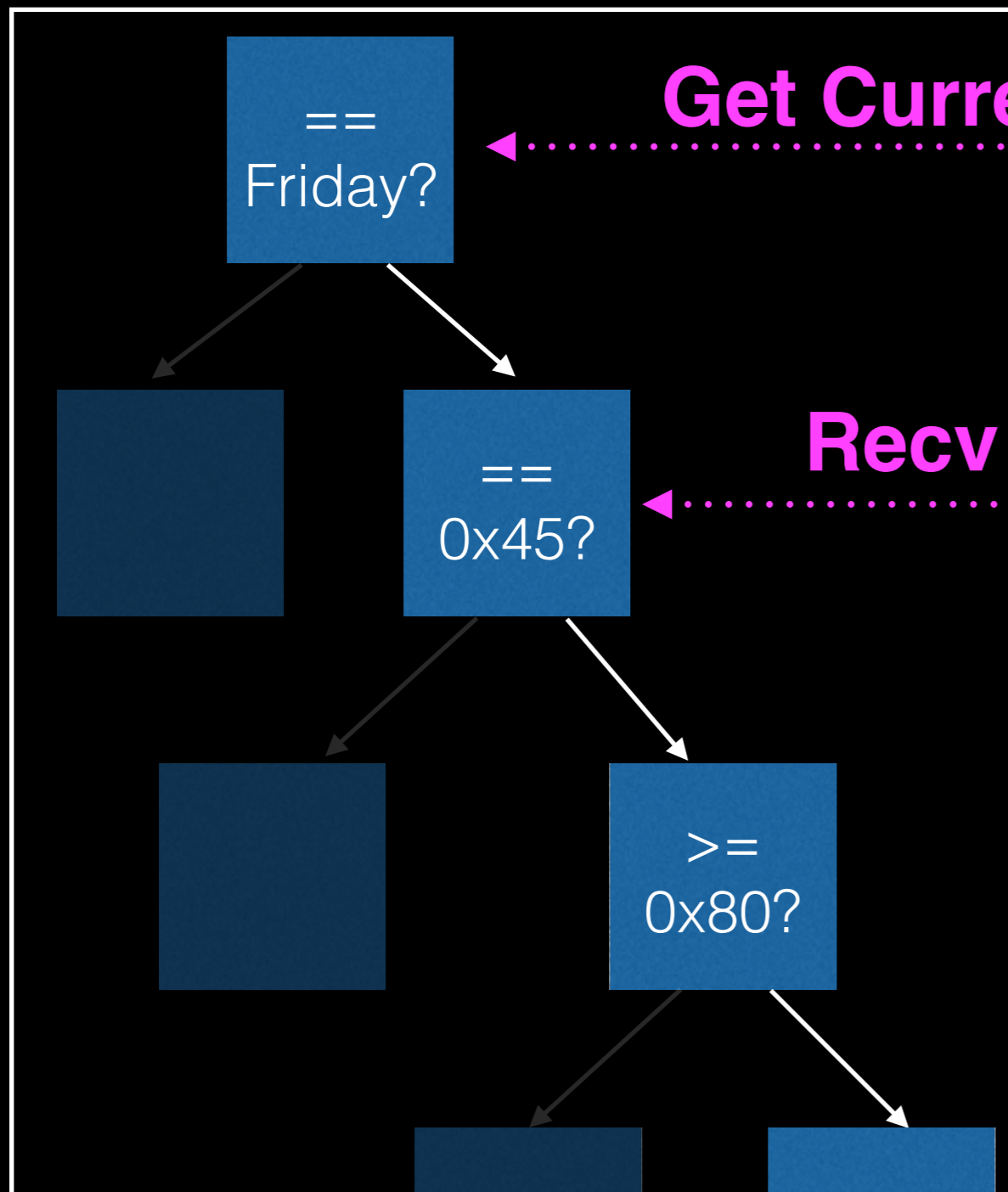
Fri May 23 11:33:27

```
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/Replay

**CPU**

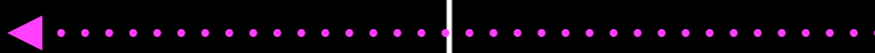
**Outside World**



**Get Current Date**



**Recv Packet**



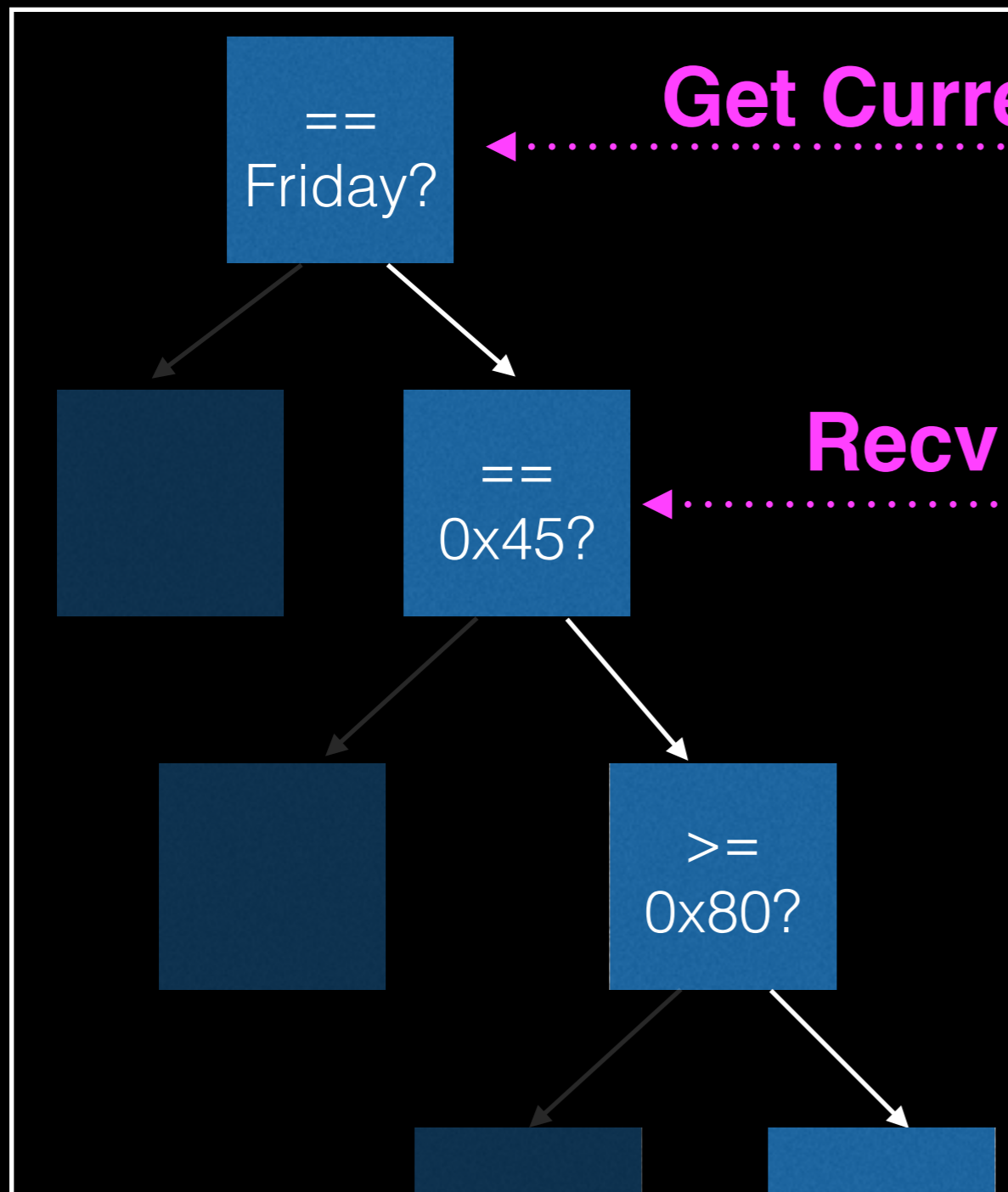
Fri May 23 11:33:27

```
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/Replay

**CPU**

**Outside World**



**Get Current Date**

**Recv Packet**

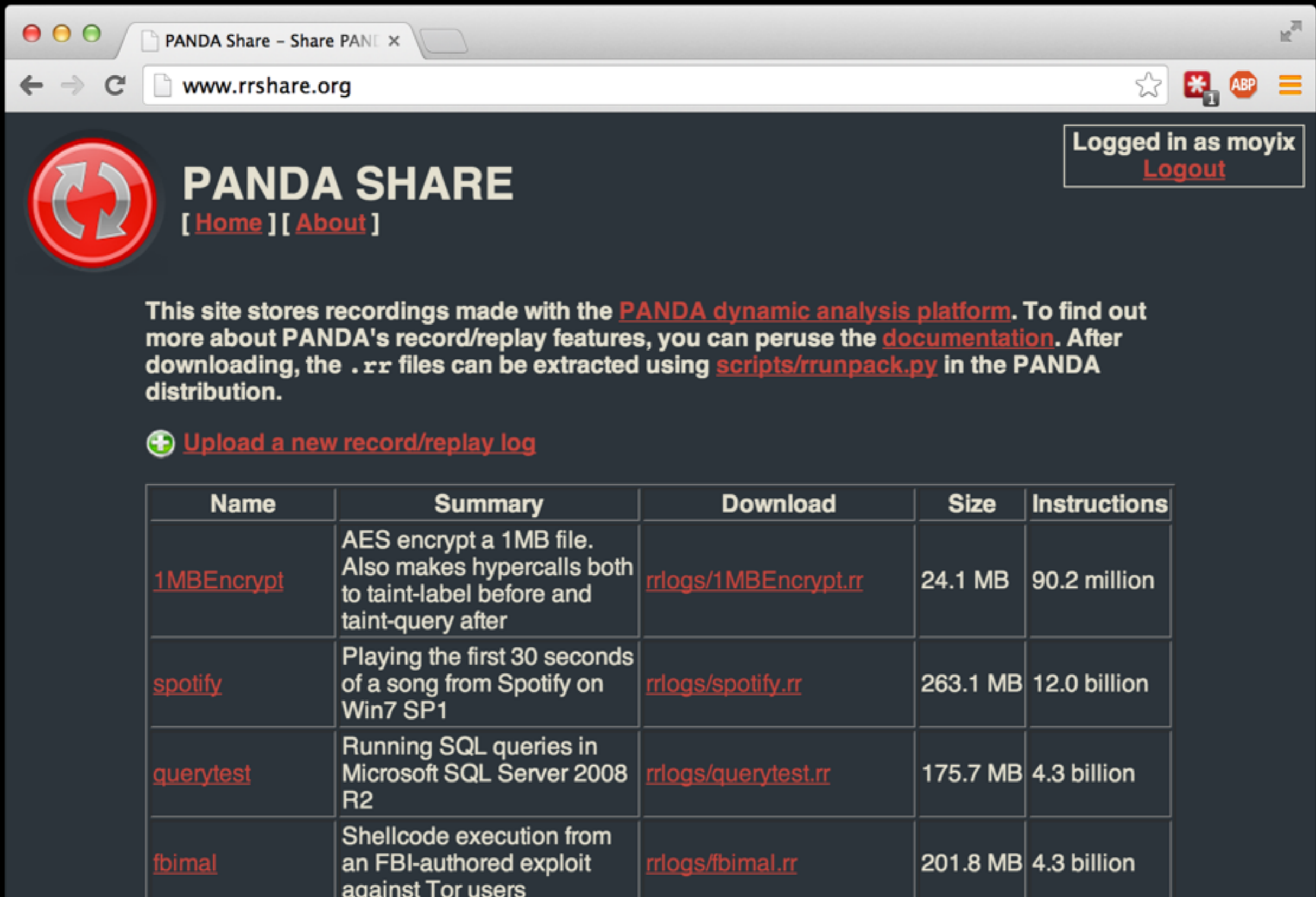
Fri May 23 11:33:27

```
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



The screenshot shows a web browser window with the address bar displaying `www.rrshare.org`. The page title is "PANDA Share - Share PANI x". The main content area features the "PANDA SHARE" logo, which consists of a red circle with two white arrows forming a loop. Below the logo are links for "[ Home ]" and "[ About ]". In the top right corner, a notification box indicates "Logged in as moyix" with a "Logout" link. The main text explains that the site stores recordings from the PANDA dynamic analysis platform and provides instructions on how to extract .rr files. A green plus icon is followed by a link to "Upload a new record/replay log". Below this is a table with four columns: Name, Summary, Download, Size, and Instructions. The table lists four records: 1MBEncrypt, spotify, querytest, and fbimal, each with a brief description, a download link, and its size and instruction count.

**PANDA SHARE**  
[ [Home](#) ] [ [About](#) ]

Logged in as moyix  
[Logout](#)

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
<a href="#">1MBEncrypt</a>	AES encrypt a 1MB file. Also makes hypercalls both to taint-label before and taint-query after	<a href="#">rrlogs/1MBEncrypt.rr</a>	24.1 MB	90.2 million
<a href="#">spotify</a>	Playing the first 30 seconds of a song from Spotify on Win7 SP1	<a href="#">rrlogs/spotify.rr</a>	263.1 MB	12.0 billion
<a href="#">querytest</a>	Running SQL queries in Microsoft SQL Server 2008 R2	<a href="#">rrlogs/querytest.rr</a>	175.7 MB	4.3 billion
<a href="#">fbimal</a>	Shellcode execution from an FBI-authored exploit against Tor users	<a href="#">rrlogs/fbimal.rr</a>	201.8 MB	4.3 billion

# 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,rspace
mov_i64 tmp2,rspace
movi_i64 tmp12,$0xfffffffffffffc
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

```
logger: created 256K log 'log_events'
logger: created 64K log 'log_radio'
Netfilter messages via NETLINK v0.30.
nf_conntrack version 0.5.0 (13312 buckets, 53248 max)
CONFIG_NF_CT_ACCT is deprecated and will be removed soon. Please use
nf_conntrack.acct=1 kernel parameter, acct=1 nf_conntrack module or
sysctl net.netfilter.nf_conntrack_acct=1 to enable it.
ctnetlink v0.93: registering with nfnetlink.
NF_IPROXY: Transparent proxy support initialized, version 4.0.0
NF_IPROXY: Copyright (c) 2006-2007 SabaBit IT Ltd.
xt_time: kernel timezone is -0000
ip_tables: (C) 2000-2006 Netfilter Core Team
arp_tables: (C) 2002 David S. Miller
TCP cubic registered
NET: Registered protocol family 10
ip6_tables: (C) 2000-2006 Netfilter Core Team
IPv6 over IPv4 tunneling driver
NET: Registered protocol family 17
NET: Registered protocol family 15
RPC: Registered udp transport module.
RPC: Registered tcp transport module.
802.1Q VLAN Support v1.8 Ben Greear <greearb@candelatech.com>
All bugs added by David S. Miller <davem@redhat.com>
VFP support v0.3: implementor 41 architecture 3 part 40 variant 0
goldfish_rtc goldfish_rtc: setting system clock to 2014-06-20 10:05:00
Freeing init memory: 124K
mmc0: new SDHC card at address e118
mmcblk0: mmc0:e118 SU82G 4.00 GiB
mmcblk0: unknown partition table
init: cannot open '/initlogo.rle'
yaffs: dev is 32505856 name is "mtdblock0"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.0, "mtdblock0"
yaffs_read_super: isCheckpointed 0
save_exit: isCheckpointed 0
yaffs: dev is 32505857 name is "mtdblock1"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.1, "mtdblock1"
yaffs_read_super: isCheckpointed 0
yaffs: dev is 32505858 name is "mtdblock2"
yaffs: passed flags ""
yaffs: Attempting MTD mount on 31.2, "mtdblock2"
yaffs_read_super: isCheckpointed 0
init: cannot find '/system/etc/install-recovery.sh', disabling initramfs
eth0: link up
shell@android:/ $ warning: 'rild' uses 32-bit capabilities (legacy use of mode 0 causes problems on processors
request_suspend_state: wakeup (3->0) at 19726028528 (2014-06-20 10:05:00)
init: sys_prop: permission denied uid:1003 name:service.bootanim.drawable
```



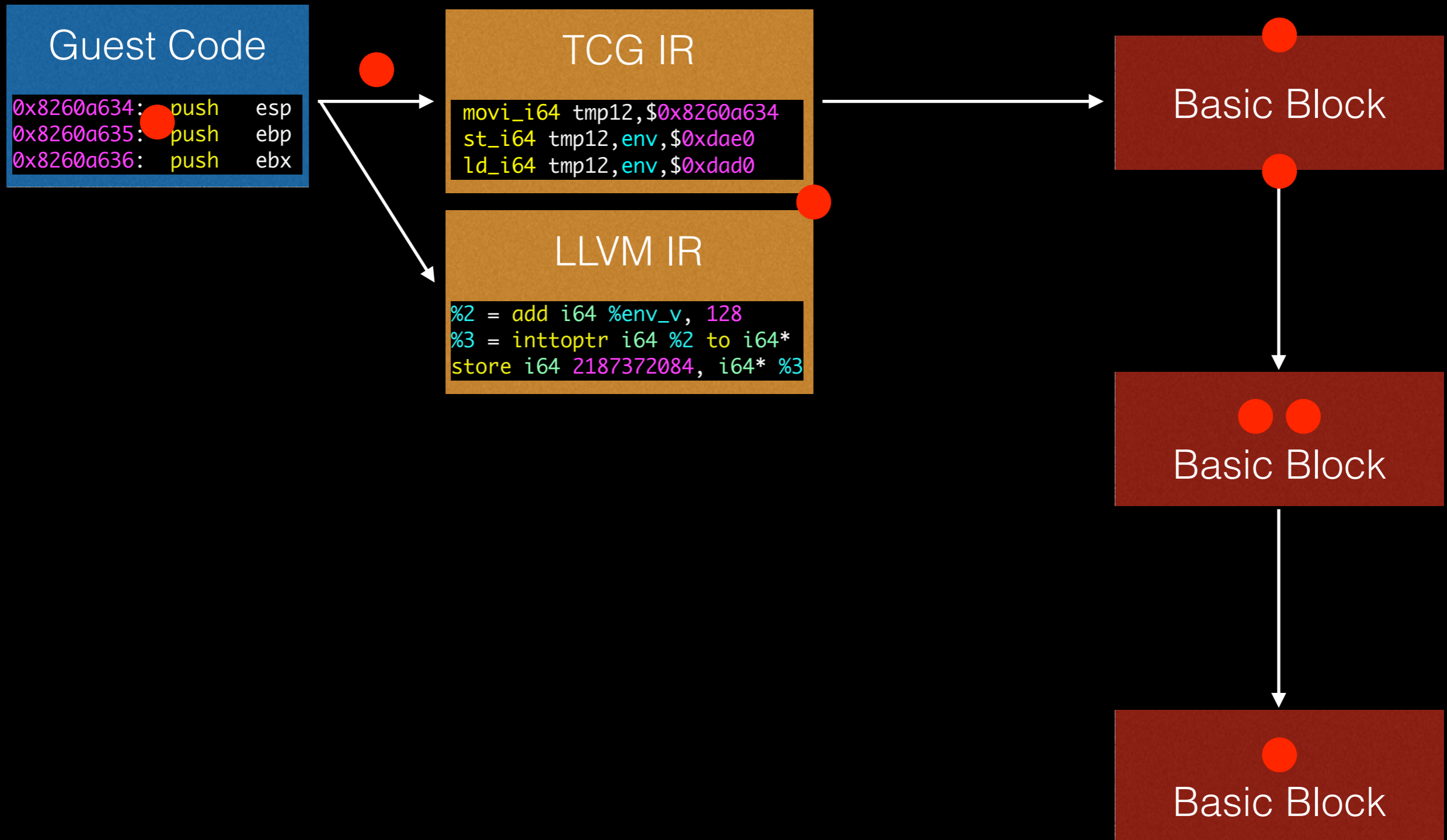
- Supports Android 2.x – 4.2
- Can make phone calls, send SMS, run native apps
- Record/replay
- Introspection into Android apps (Dalvik-level) for Android 2.3 (from DroidScope)
- System-level introspection supported on all Android versions

# 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

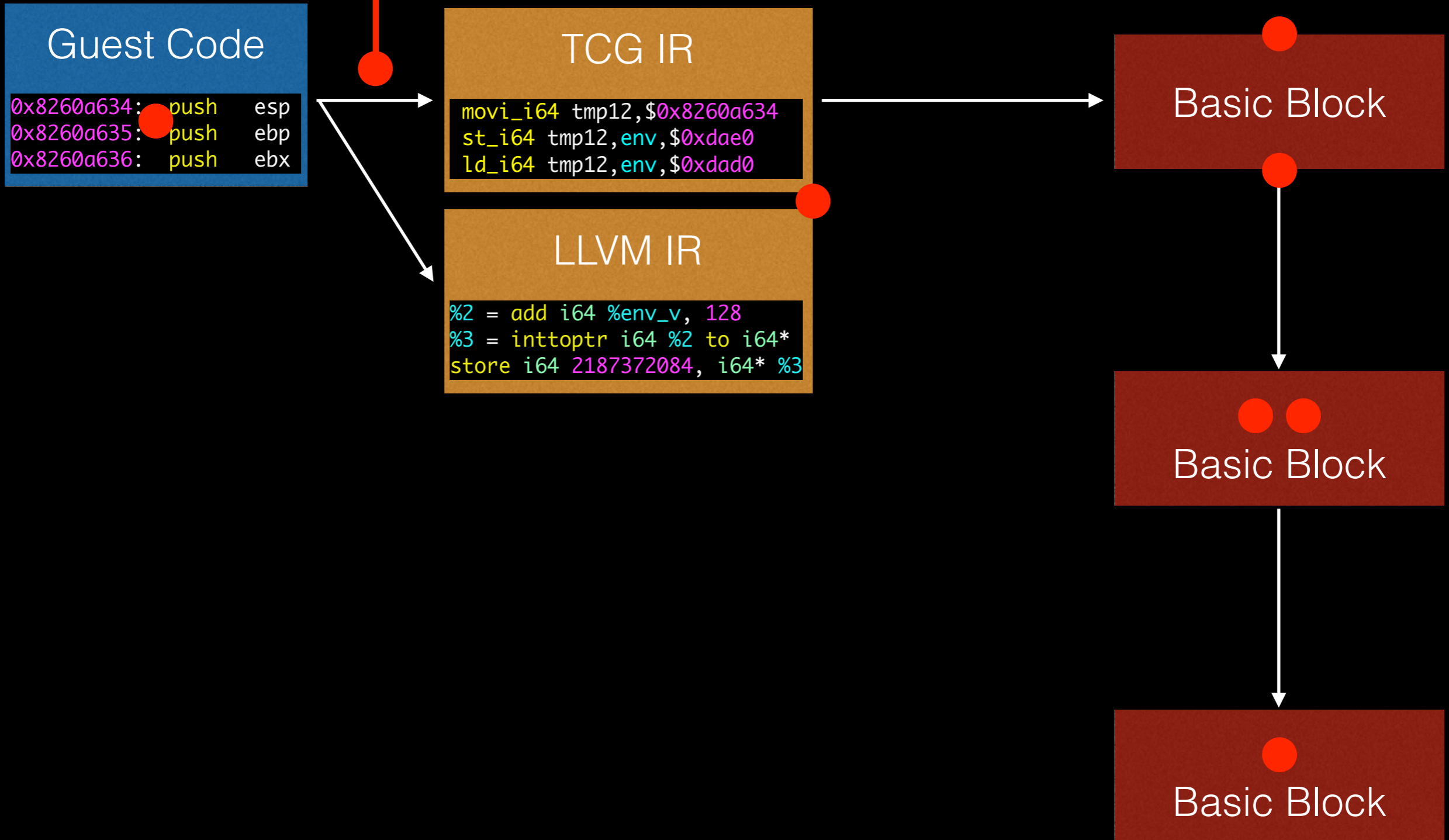
# Execution



# Translation

# Execution

PANDA\_CB\_BEFORE\_BLOCK\_TRANSLATE





# 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
```

PANDA\_CB\_INSN\_TRANSLATE

# 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
```

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
```

```
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
```

```
Basic Block
```

```
Basic Block
```

PANDA\_CB\_INSN\_TRANSLATE

PANDA\_CB\_AFTER\_BLOCK\_TRANSLATE

PANDA\_CB\_AFTER\_BLOCK\_EXEC

# 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
```

```
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

PANDA\_CB\_AFTER\_BLOCK\_EXEC

# 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

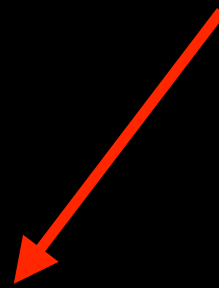
# Five Plugins, One Replay

```
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 Find  
-replay sshb32 passphrase \  
-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		Tap	Code
Read	Write		
		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
		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

Read

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

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

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]

# TZB Implementation

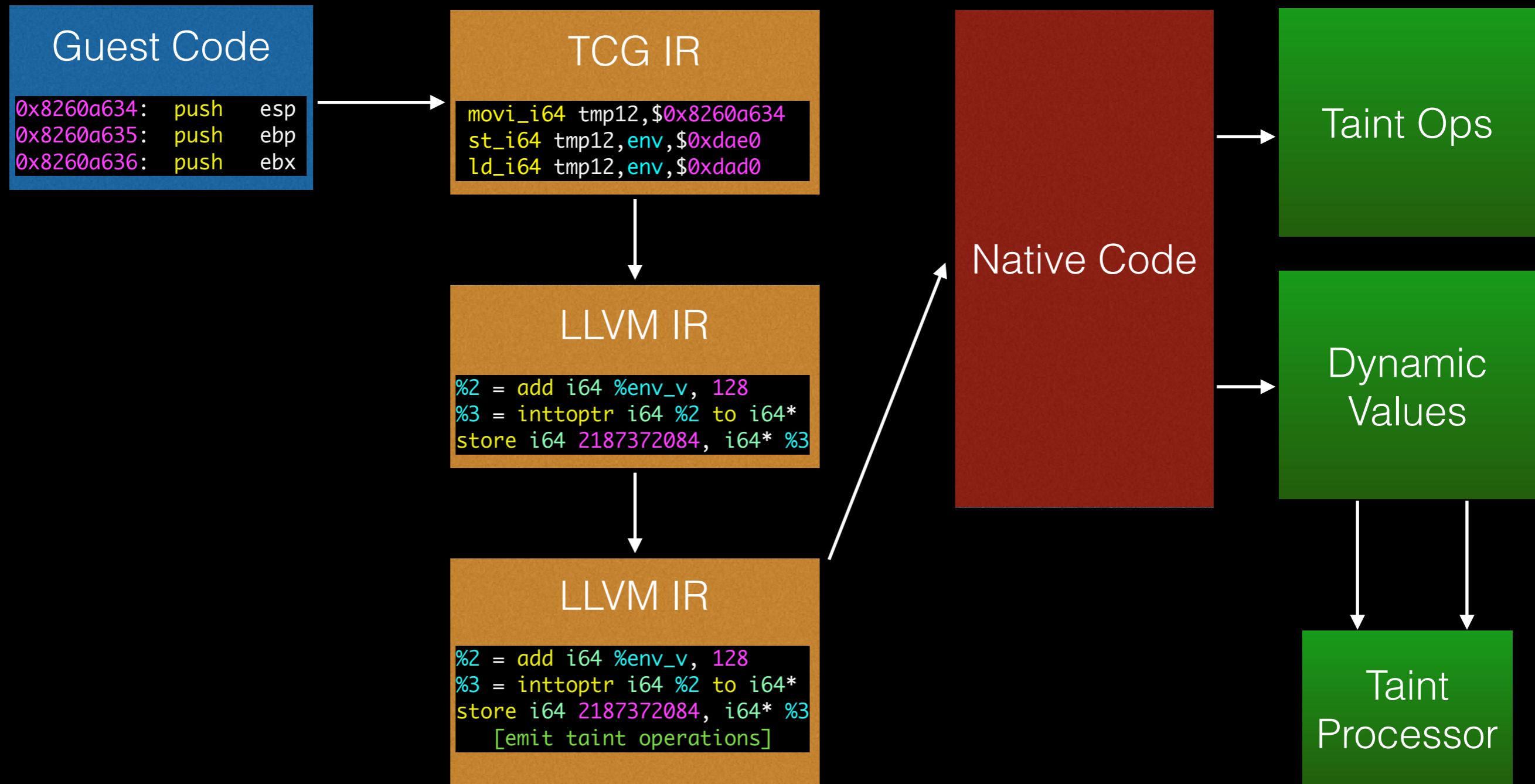
- Track calling context with `callstack_instr` plugin
- At every memory access  
(`PANDA_CB_PHYS_MEM_READ/WRITE`)  
Get (caller, program counter, address space) –  
*i.e., tap point*
- Analyze data flowing through tap point (e.g.,  
string matching with `stringsearch` plugin)

# 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



# LLVM Taint Instrumentation



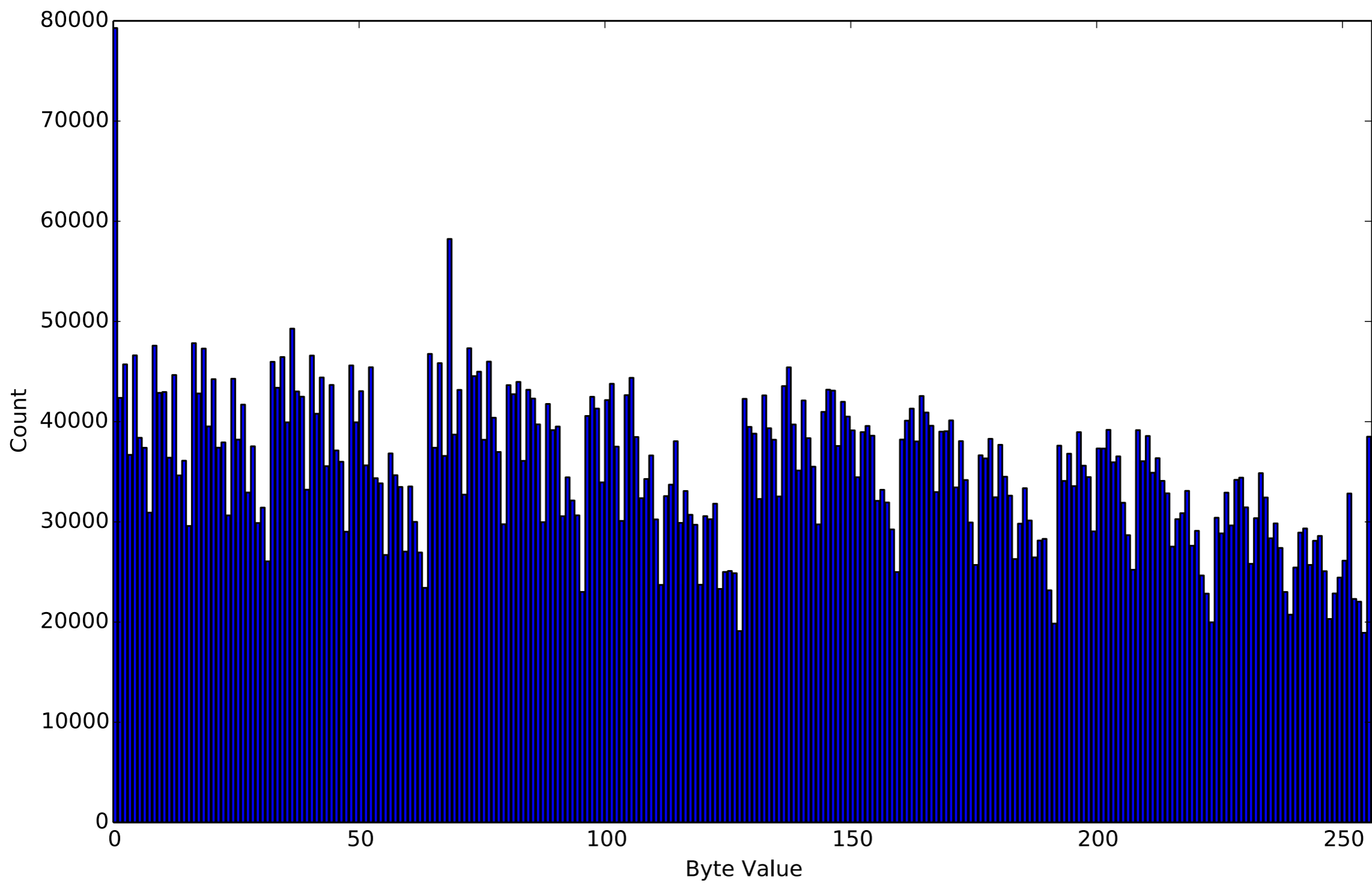
DEMO:  
ssh-keygen backdoor

# Breaking Spotify DRM

- DRM has a strong “signature”
  - **High** entropy, **high** randomness ( $\chi^2$ ) input
  - **High** entropy, **low** randomness ( $\chi^2$ ) 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



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

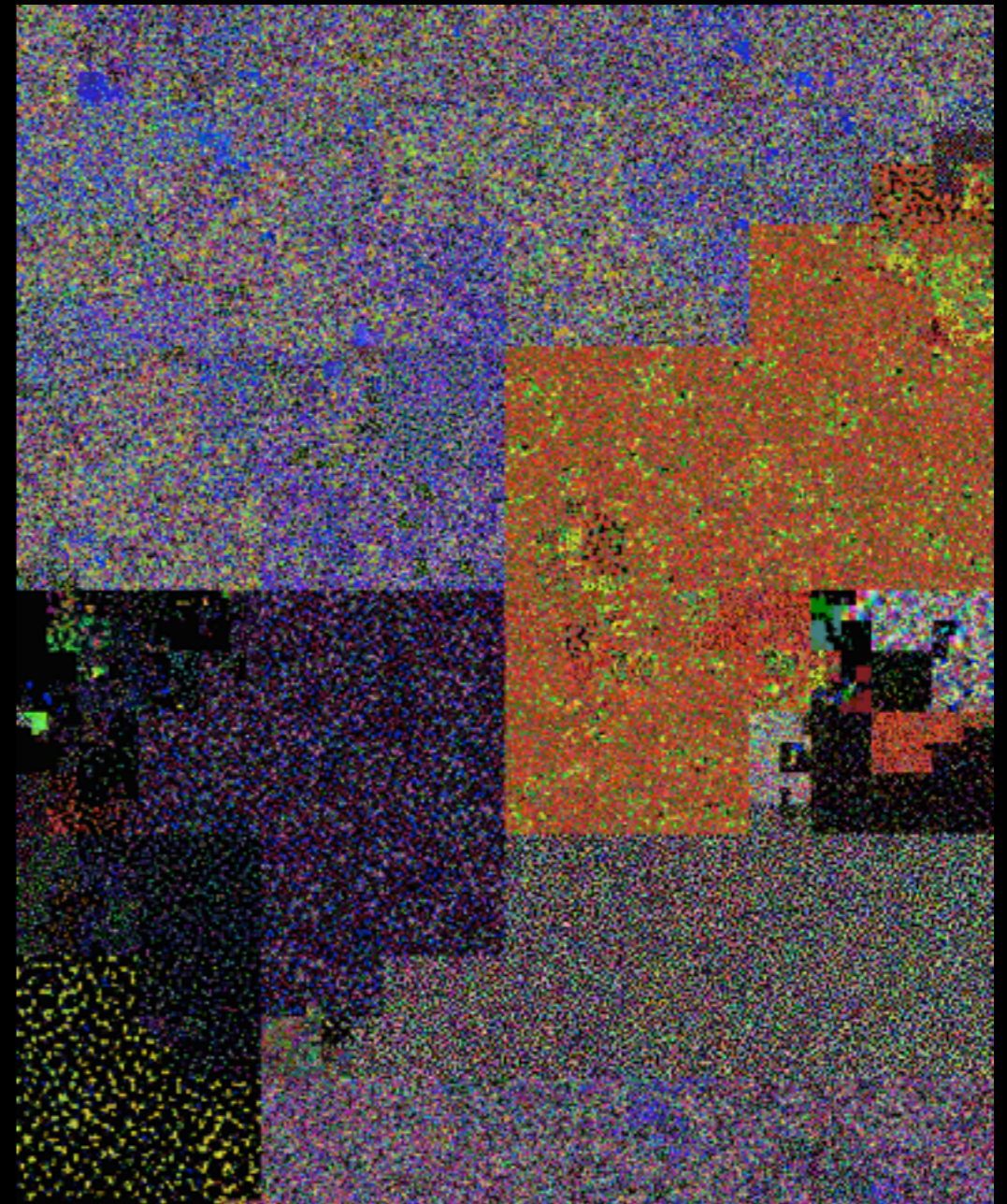
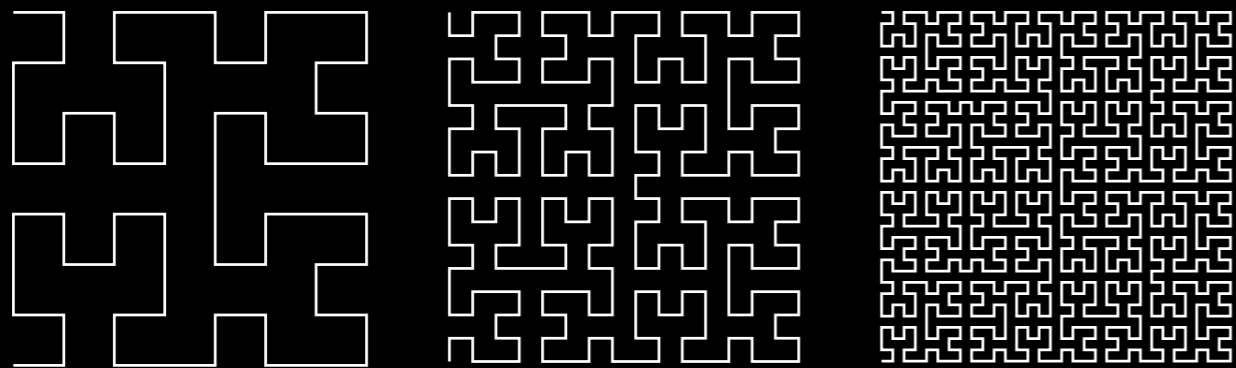


Image from Aldo Cortesi,  
*Visualizing binaries with space-filling curves*

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](mailto:brendan@cc.gatech.edu)
- Join the mailing list: [panda-users@mit.edu](mailto:panda-users@mit.edu)
- Contribute code:  
<https://github.com/moyix/panda>