Logo by: Normand Veilleux

Reverse All the Things with PANDA

Brendan Dolan-Gavitt
Columbia University

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

Reverse Engineering

- Common perception:

Reverse Engineering for Good

- At least three major “socially responsible” uses:

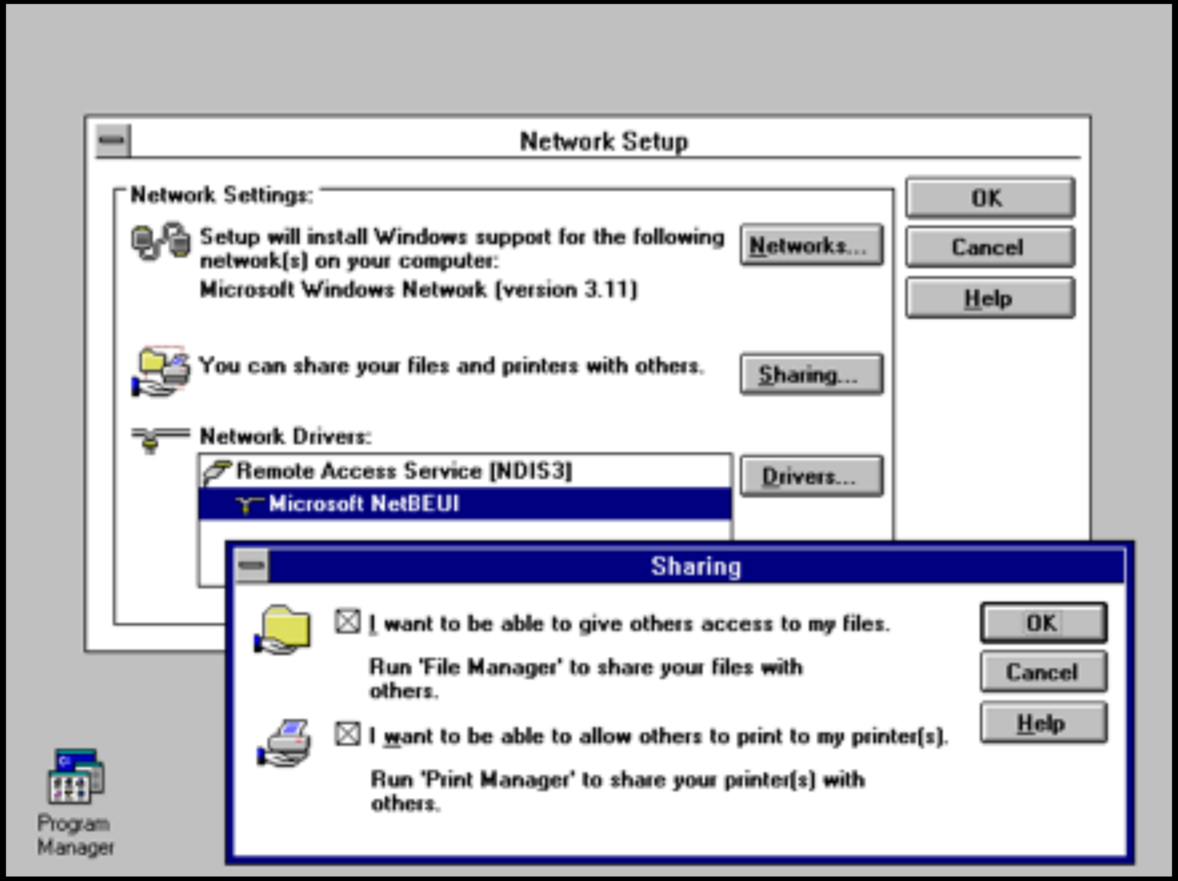
  1. Enable legacy code to continue to function

  2. Identify critical vulnerabilities

  3. Understand the true purpose and actions of code

Legacy Code

- Source lost
- Original vendor defunct
- Lost the CD key
- Need RE to update to modern environment



Network Setup

Network Settings:

Setup will install Windows support for the following network(s) on your computer:
Microsoft Windows Network (version 3.11)

You can share your files and printers with others.

Network Drivers:

Remote Access Service [NDIS3]
Microsoft NetBEUI

Sharing

☒ I want to be able to give others access to my files.

Run 'File Manager' to share your files with others.

☒ I want to be able to allow others to print to my printer(s).

Run 'Print Manager' to share your printer(s) with others.

Understanding Vulnerabilites

- Have a crash, but is it an exploitable vulnerability?
- Often depends on dark, undocumented corners of the software and its libraries
- Reverse engineering necessary to uncover these details

Auditing Software

- Even apparently legitimate programs may not be working in the users' interest
    - E.g., Sony BMG rootkit

SONY BMG
MUSIC ENTERTAINMENT

- Or they may not be working as claimed
    - E.g.: can Apple read your iMessages?
- Can reverse engineer to audit behavior

Case Studies

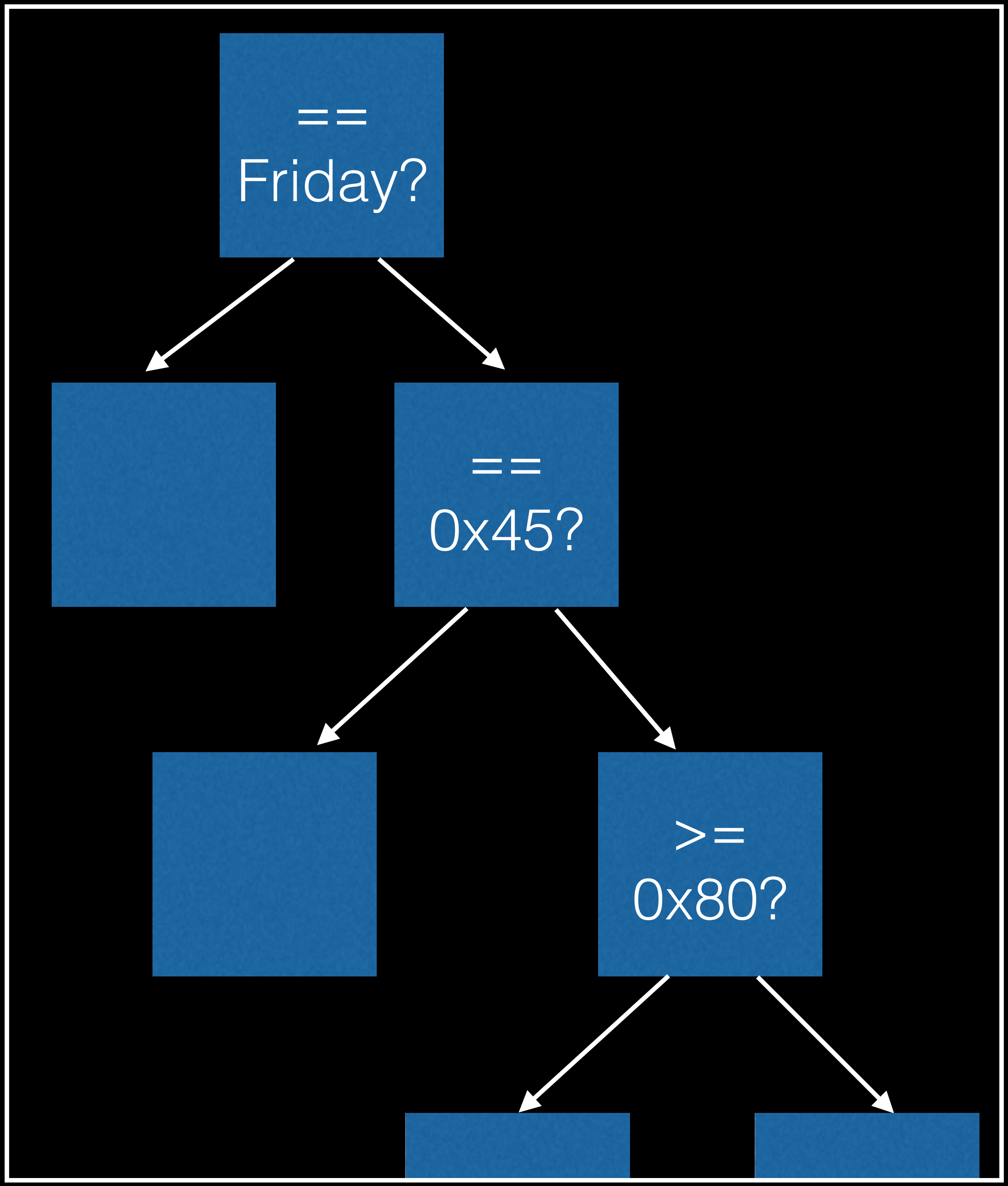
- Reverse engineering the Starcraft CD key check
- Diagnosing a vulnerability in Internet Explorer
- Uncovering censorship in LINE IM client

PANDA: Built for RE

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

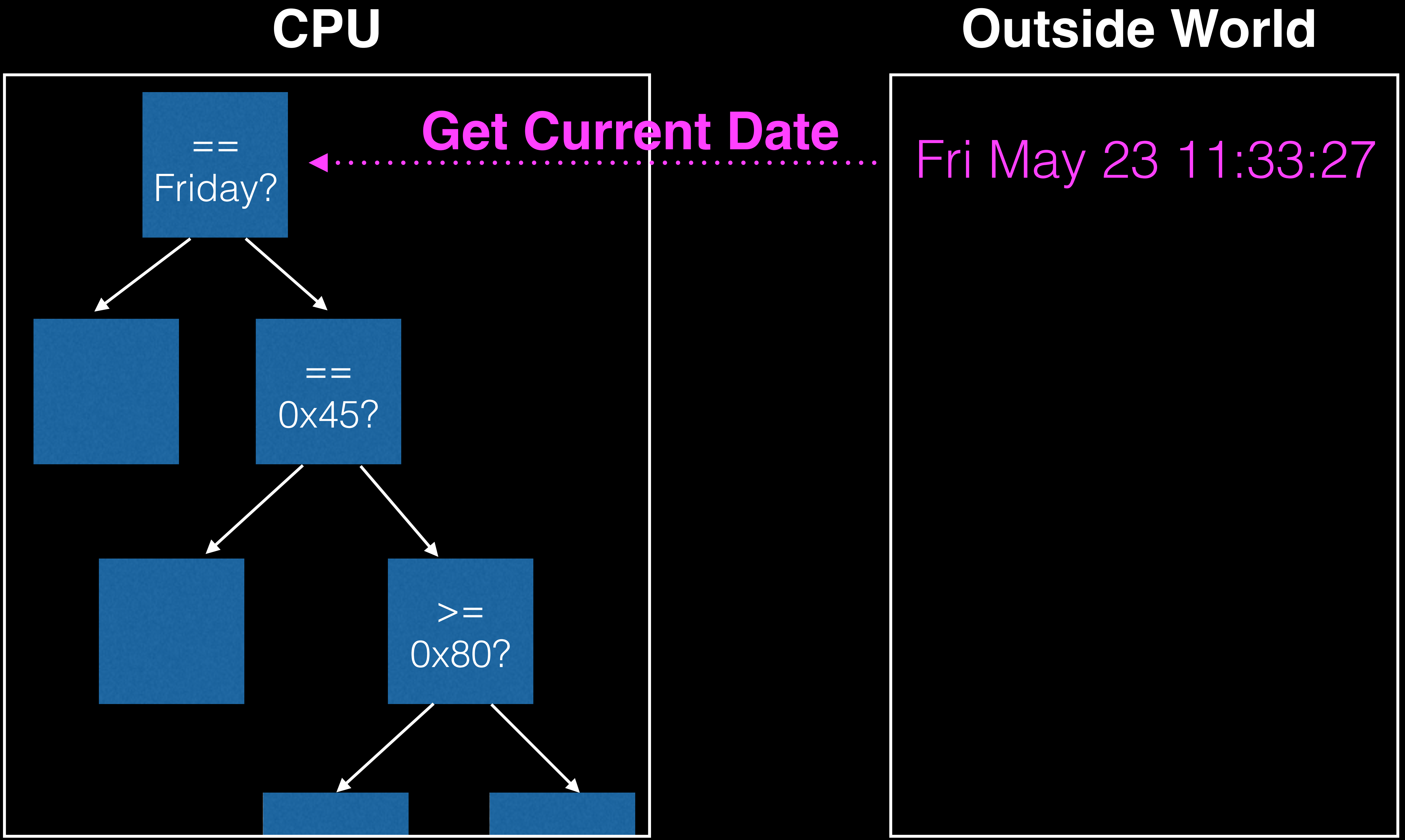
Record/Replay

CPU



Outside World

Record/Replay



CPU

Outside World

== Friday?

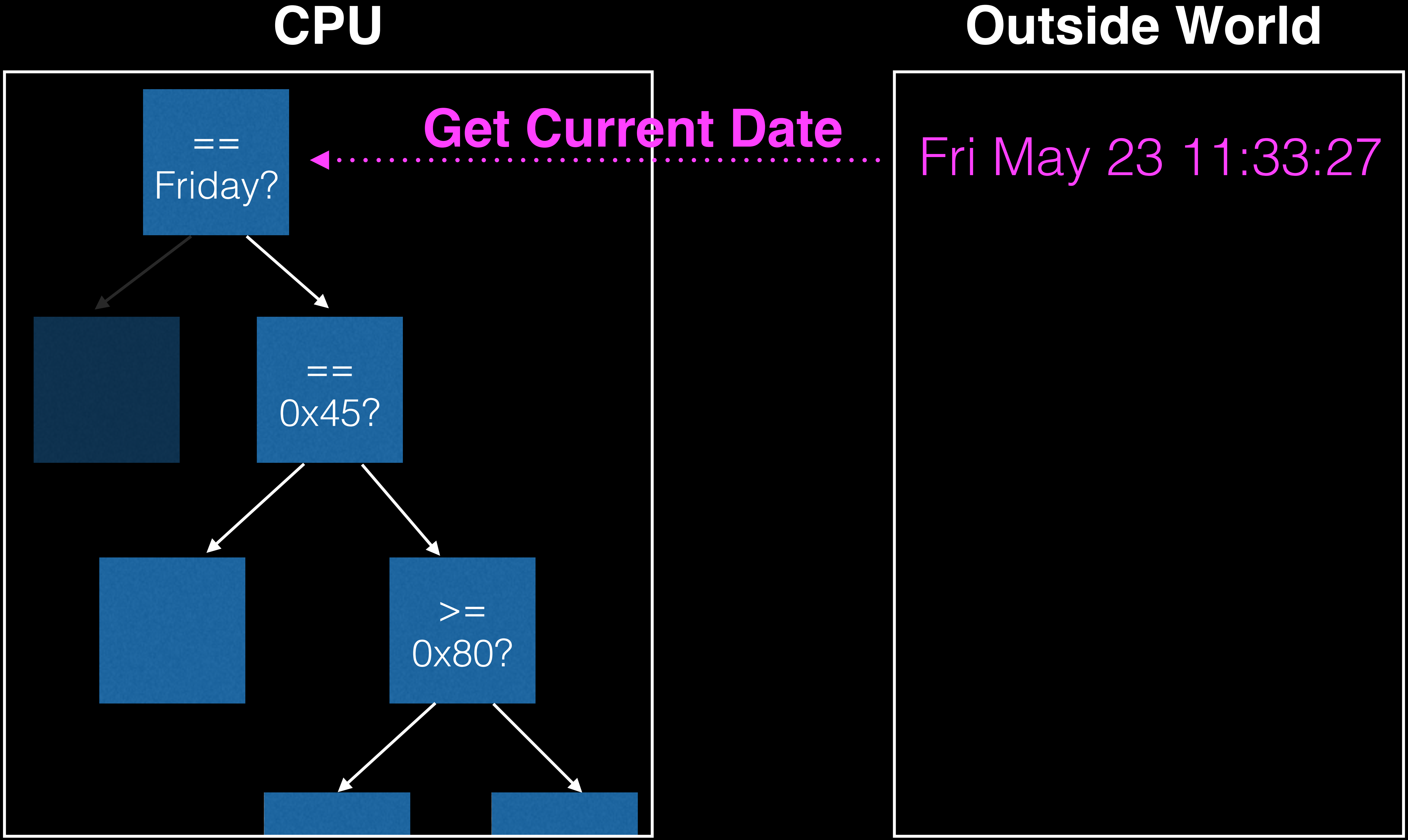
== 0x45?

>= 0x80?

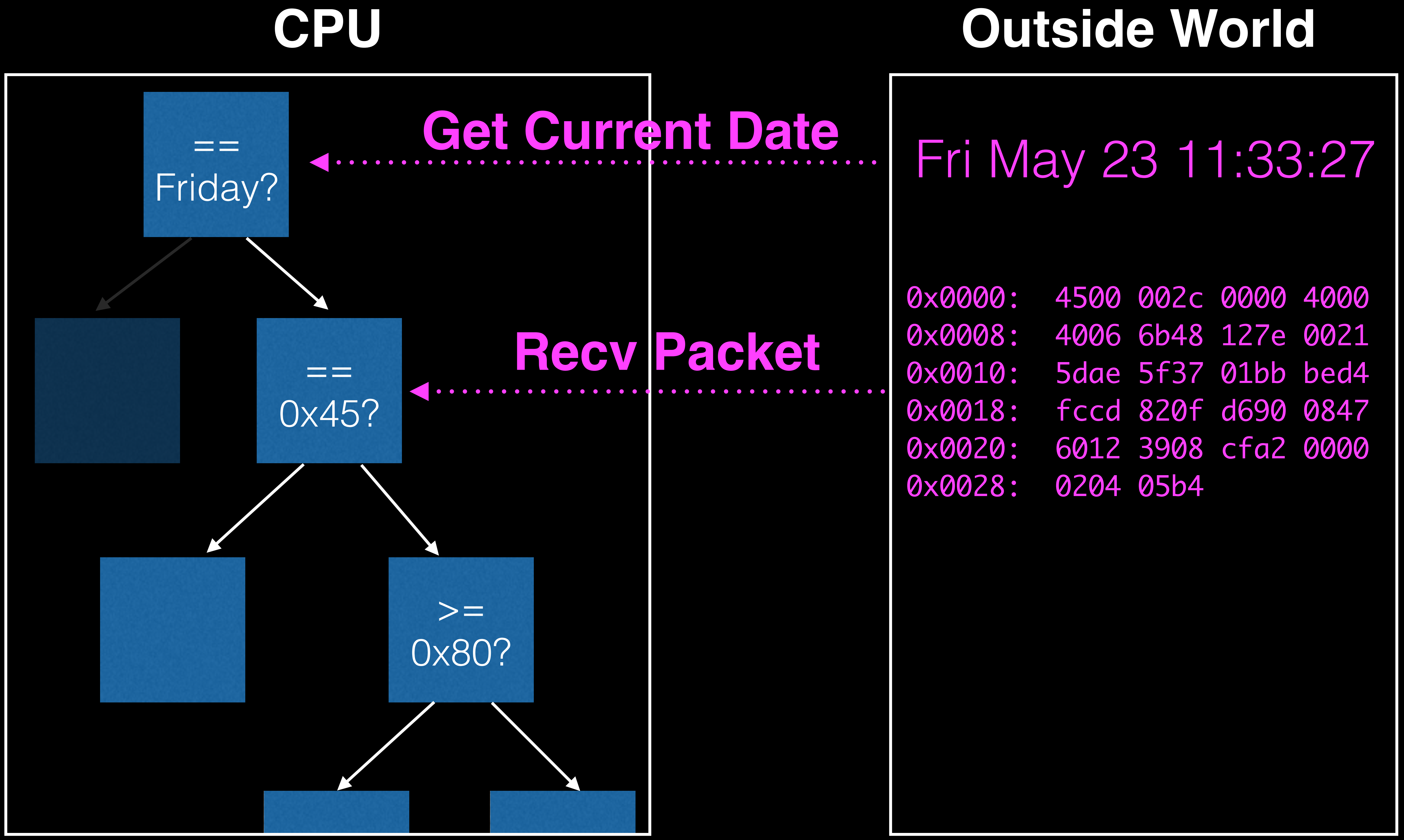
Get Current Date

Fri May 23 11:33:27

Record/Replay



Record/Replay



CPU

Outside World

== Friday?

== 0x45?

>= 0x80?

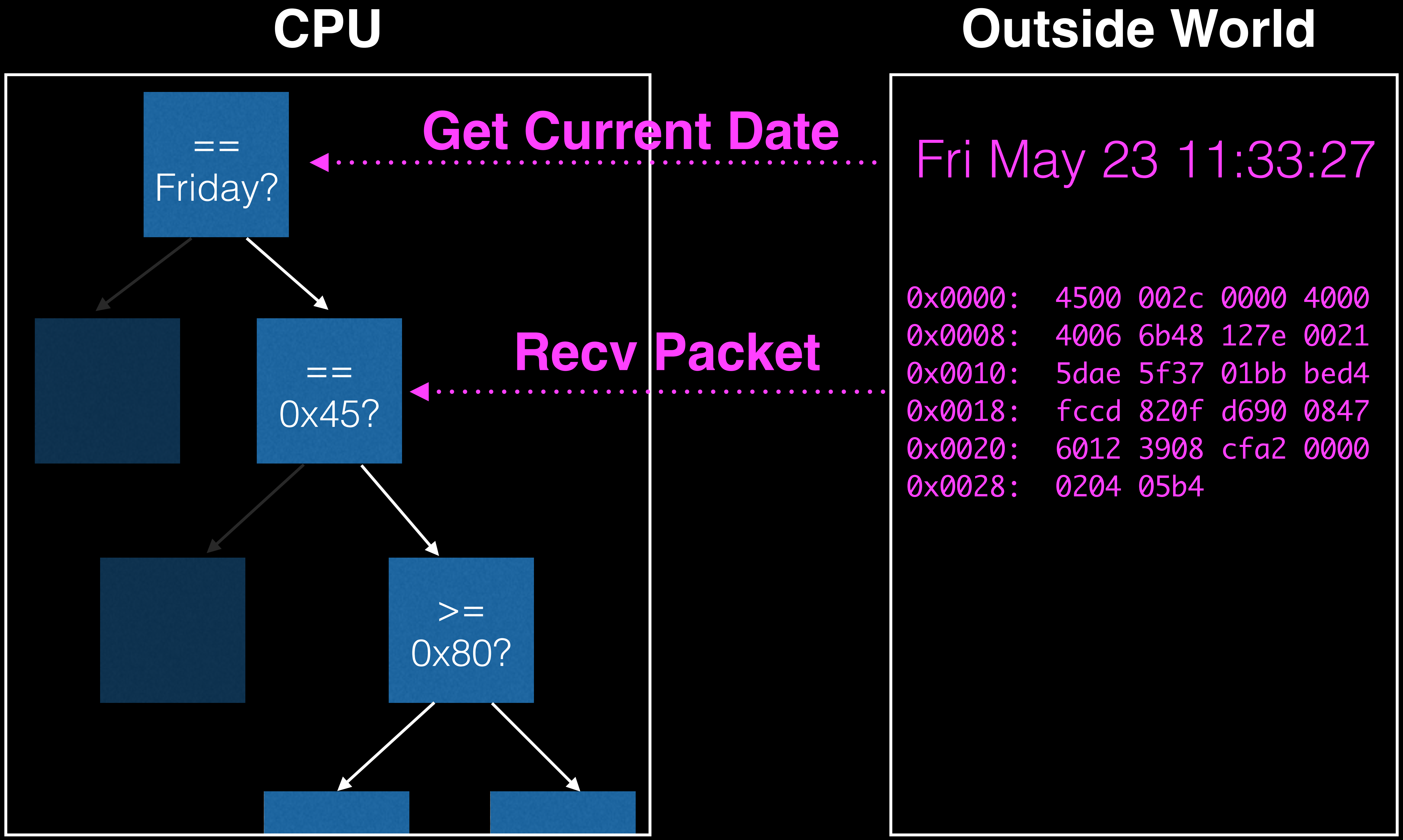
Get Current Date

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

== Friday?

Get Current Date

Fri May 23 11:33:27

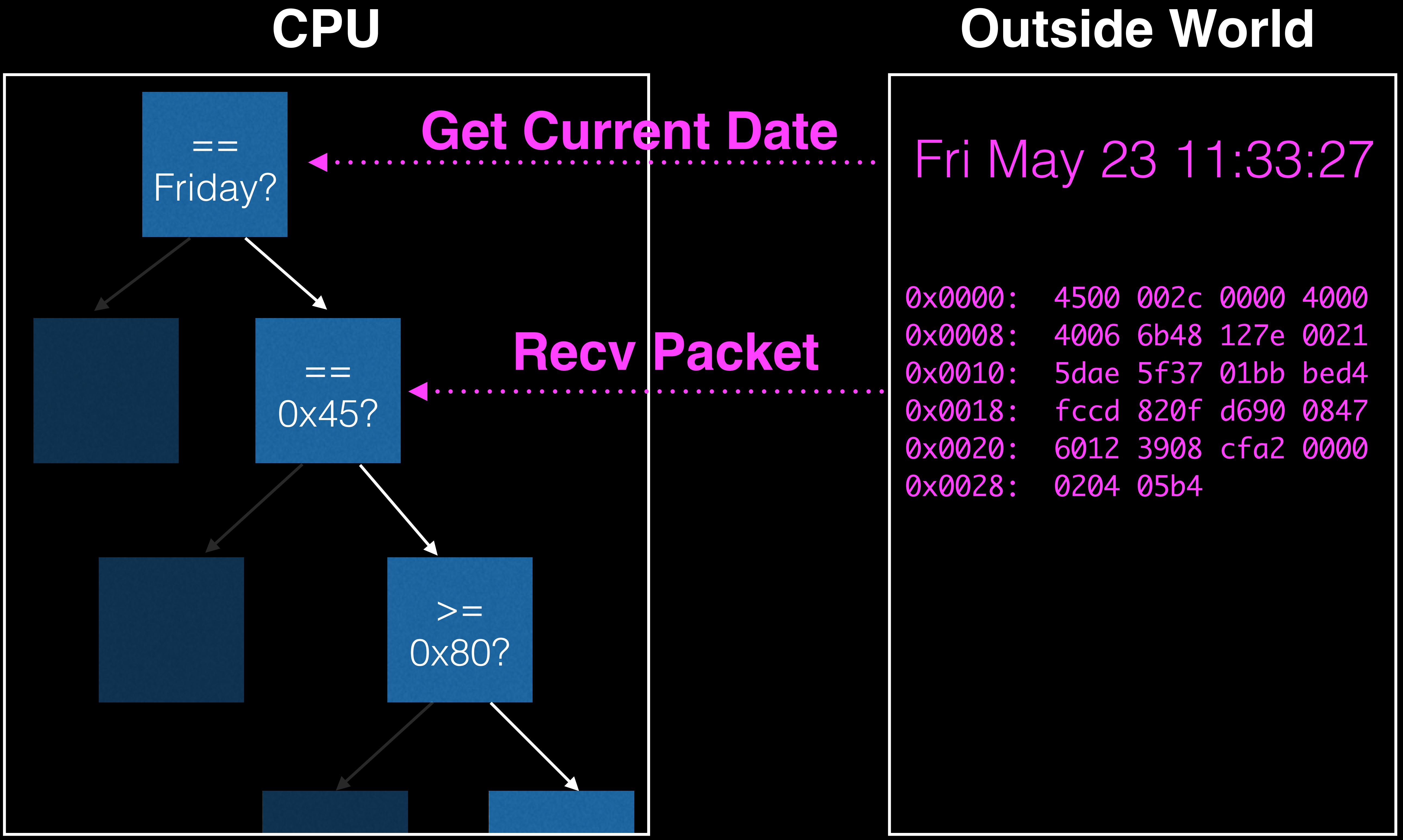
== 0x45?

Recv Packet

>= 0x80?

```
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

==
Friday?

Get Current Date

Fri May 23 11:33:27

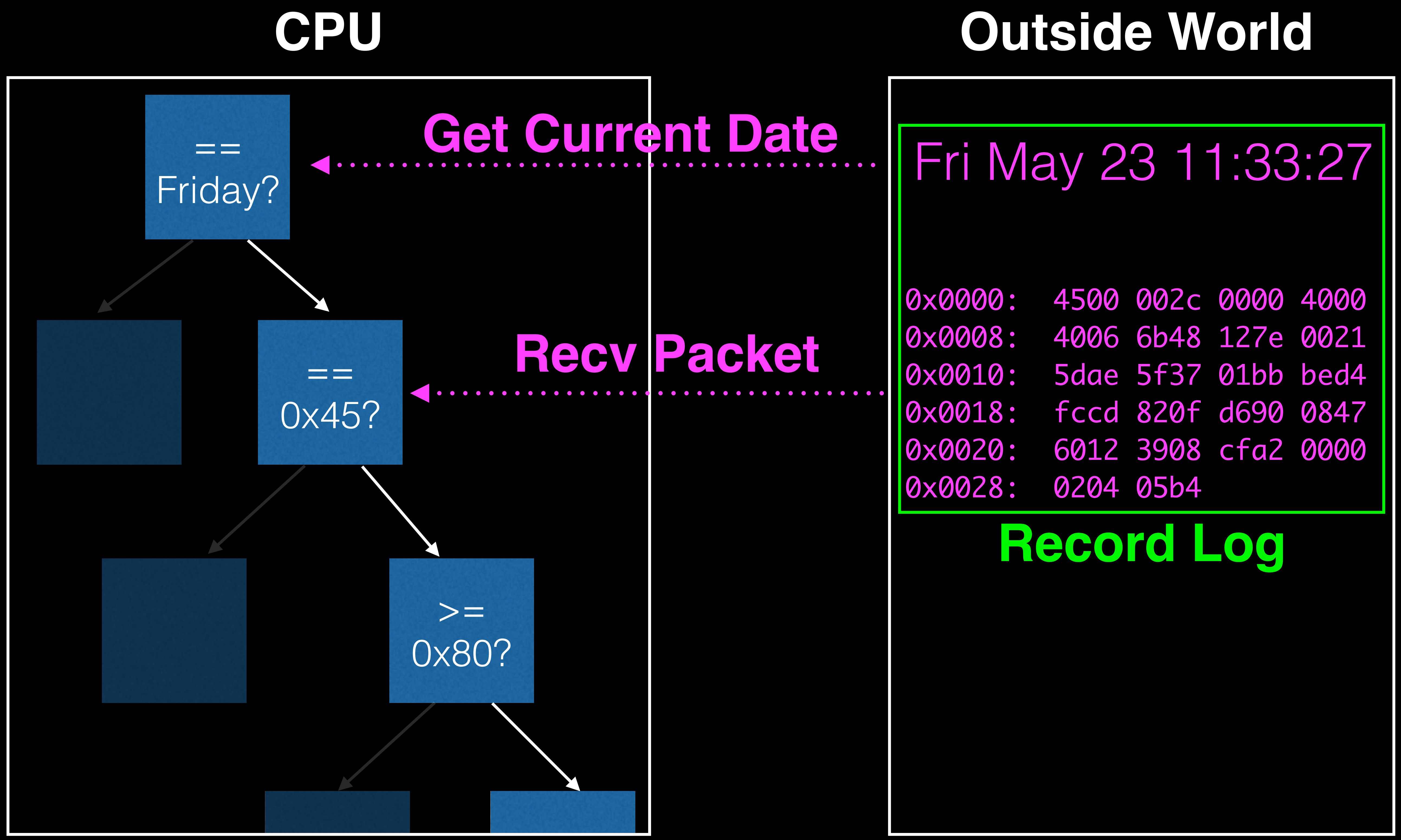
==
0x45?

Recv Packet

>=
0x80?

```
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
```

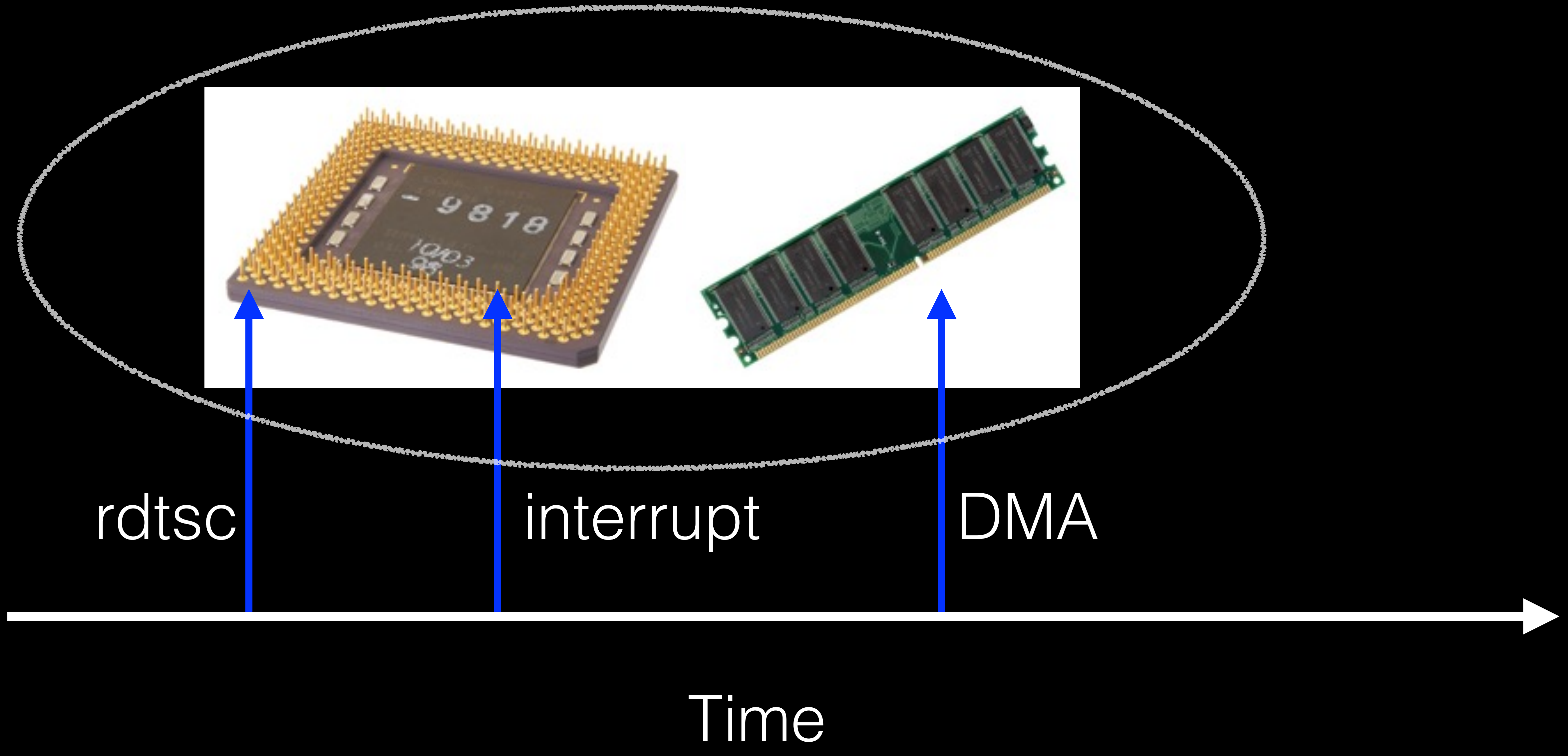
Record/Replay



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



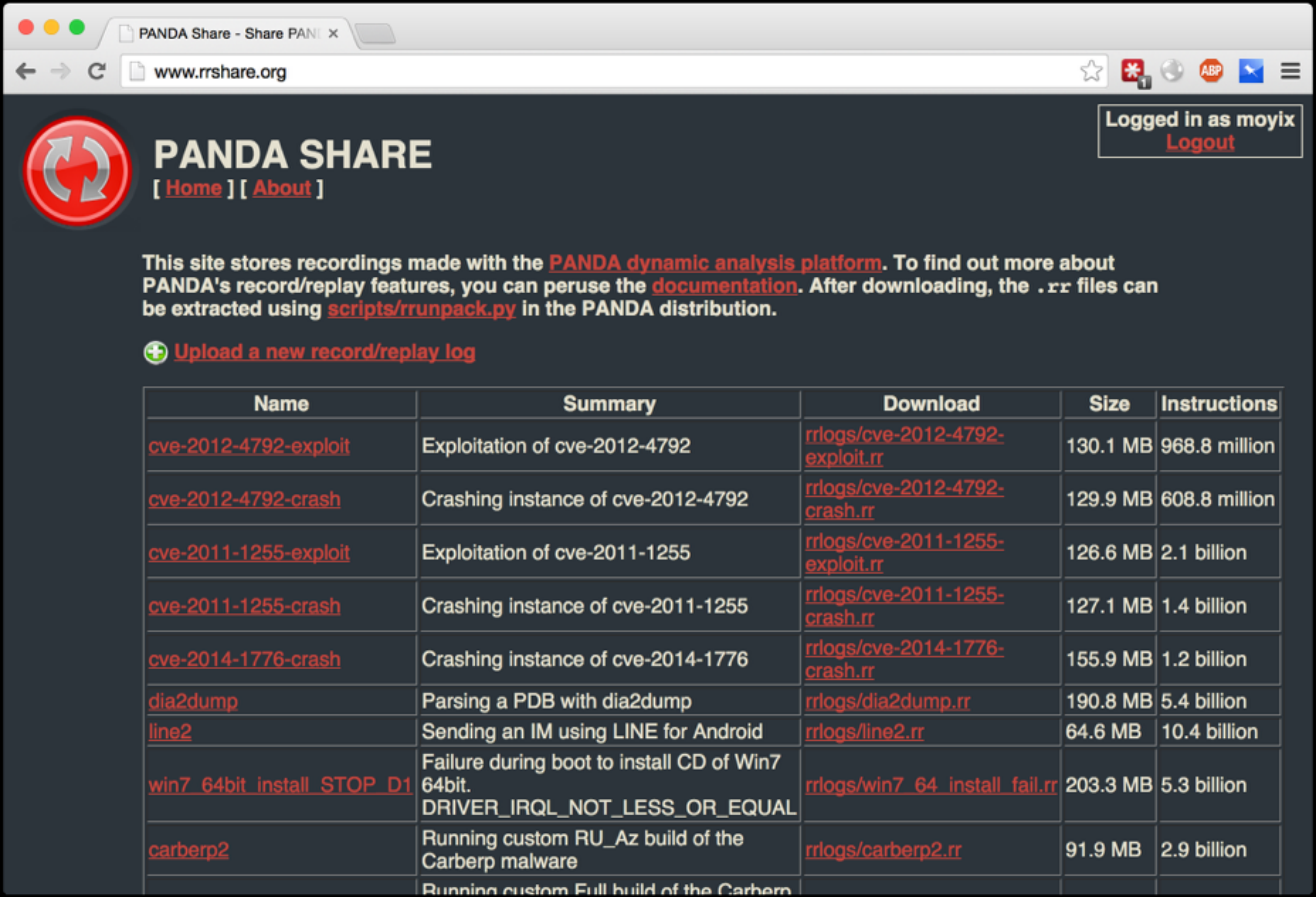
rdtsc

interrupt

DMA

Time

www.rrshare.org



Logged in as moyix
Logout

PANDA SHARE

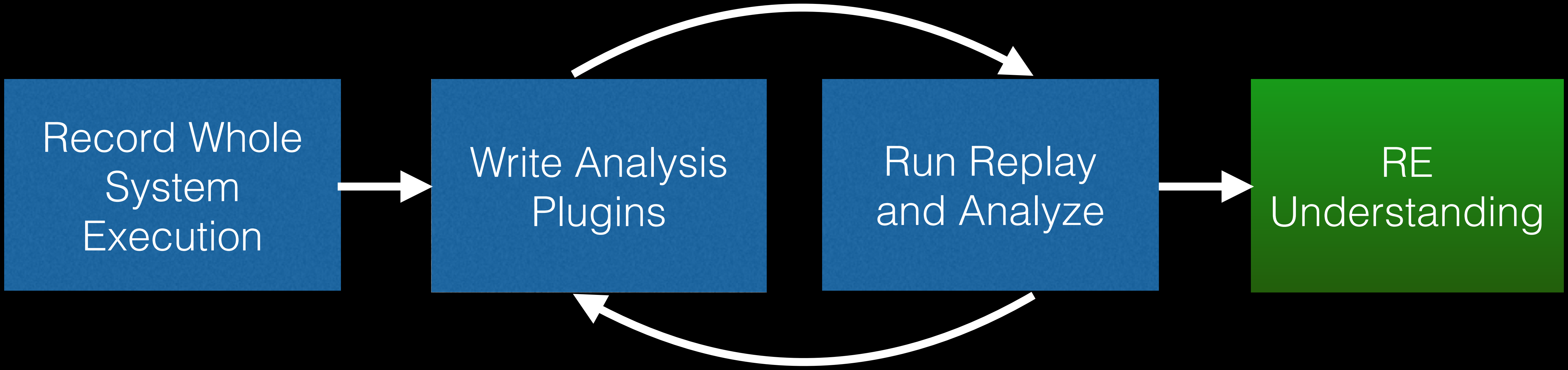
[Home] [About]

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			

PANDA Model



Record Whole System Execution

Write Analysis Plugins

Run Replay and Analyze

RE Understanding

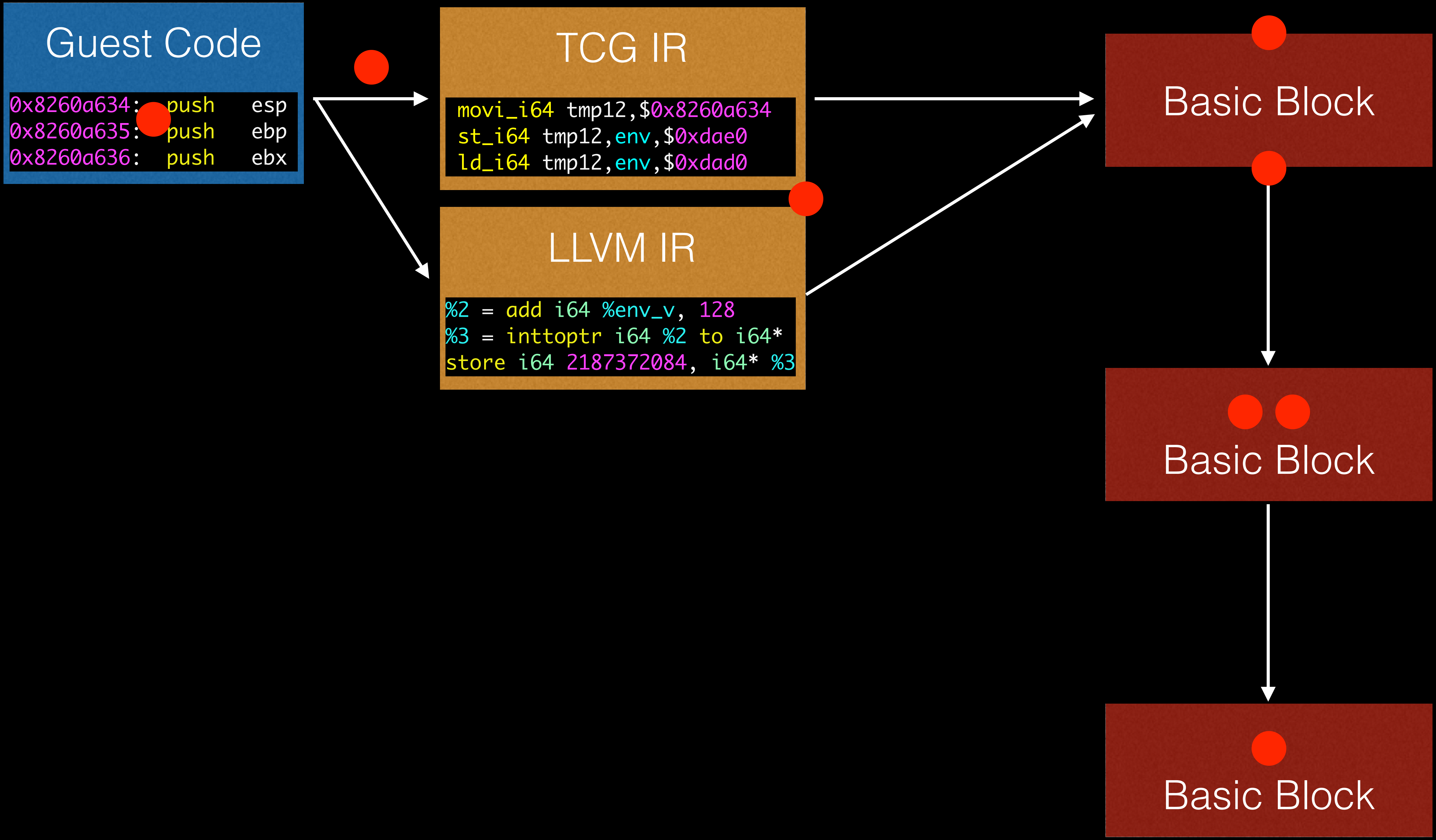
- Record / replay critical:
  - Heavy analyses don’t disrupt execution
  - Analyses don’t have to worry about memory layout changing between runs

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

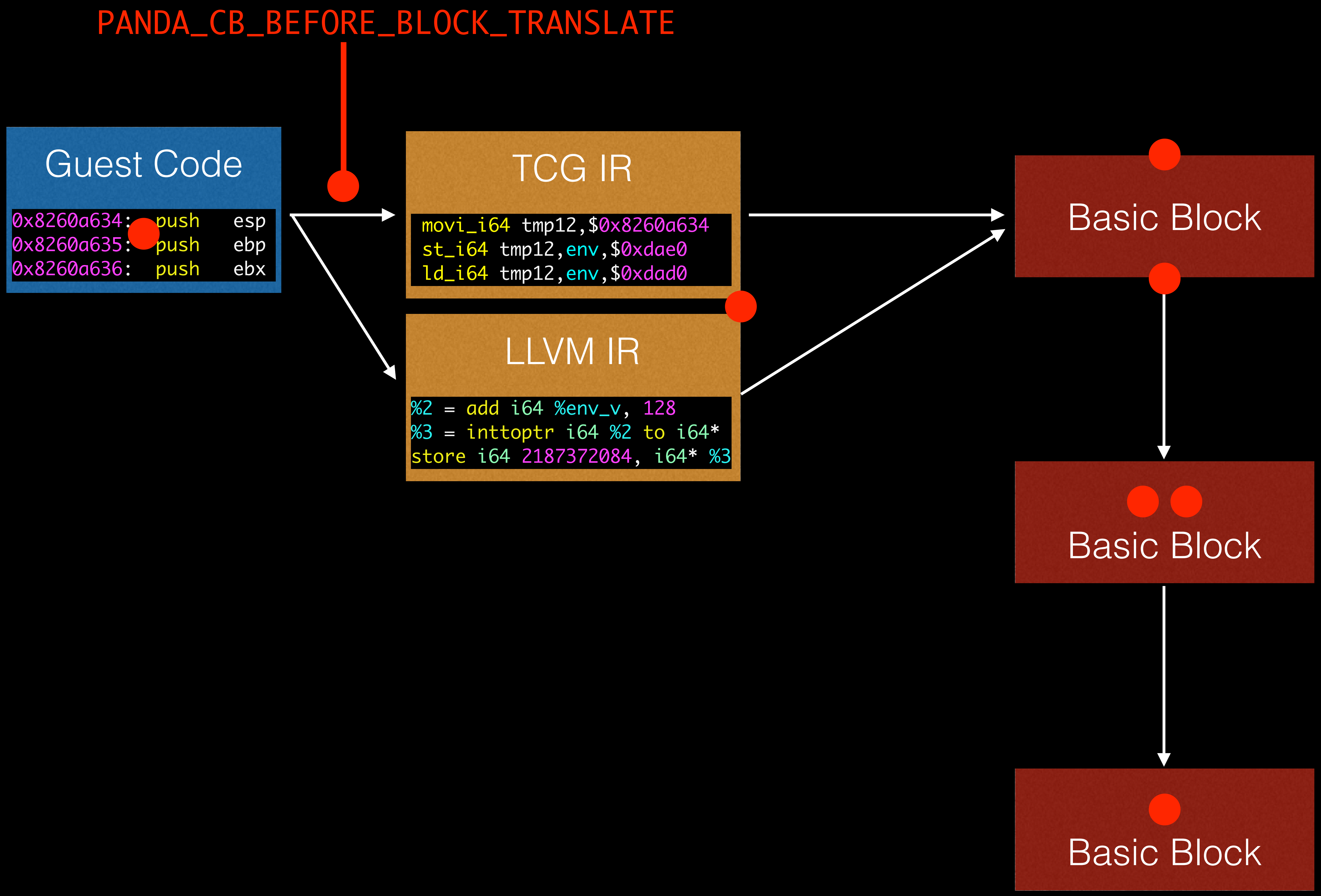
```
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
```

LLVM IR

```
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
```

Translation

Execution



PANDA_CB_BEFORE_BLOCK_TRANSLATE

Guest Code

```
0x8260a634:  push    esp
0x8260a635:  push    ebp
0x8260a636:  push    ebx
```

TCG IR

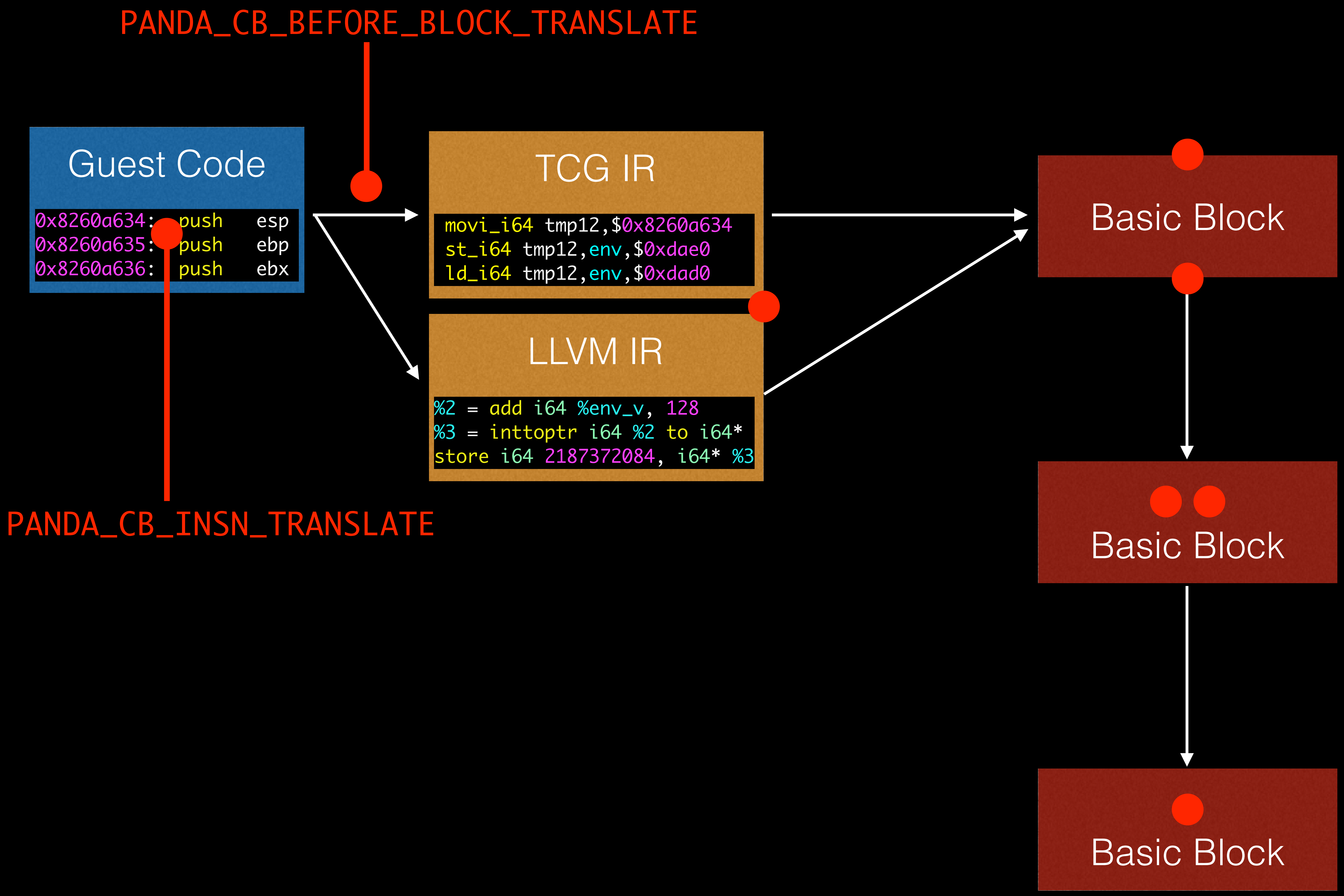
```
movi_i64 tmp12,$0x8260a634
st_i64 tmp12,env,$0xdae0
ld_i64 tmp12,env,$0xdad0
```

LLVM IR

```
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
```

Translation

Execution



PANDA_CB_BEFORE_BLOCK_TRANSLATE

Guest Code

```
0x8260a634:  push   esp
0x8260a635:  push   ebp
0x8260a636:  push   ebx
```

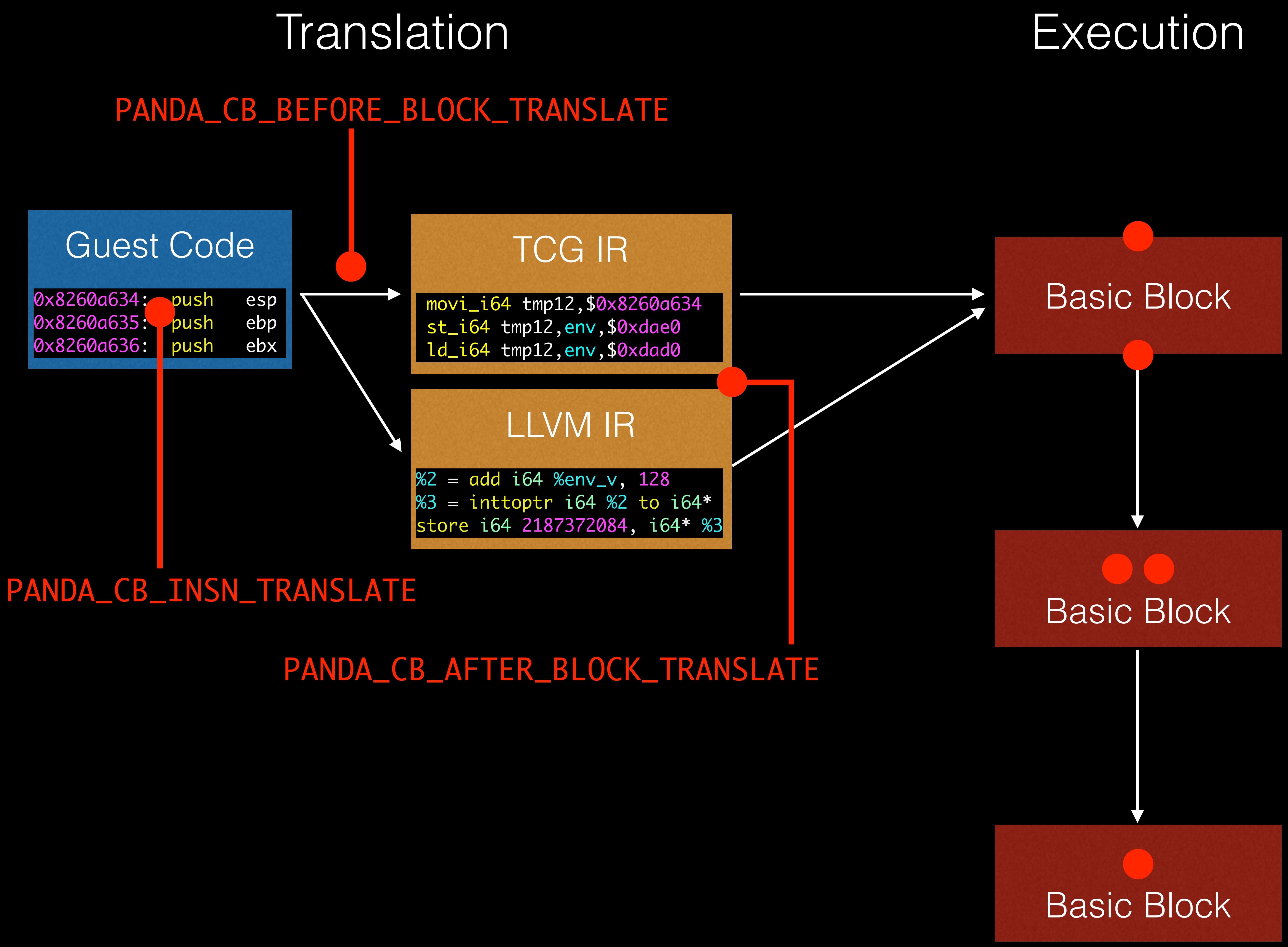
PANDA_CB_INSN_TRANSLATE

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
```



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
```

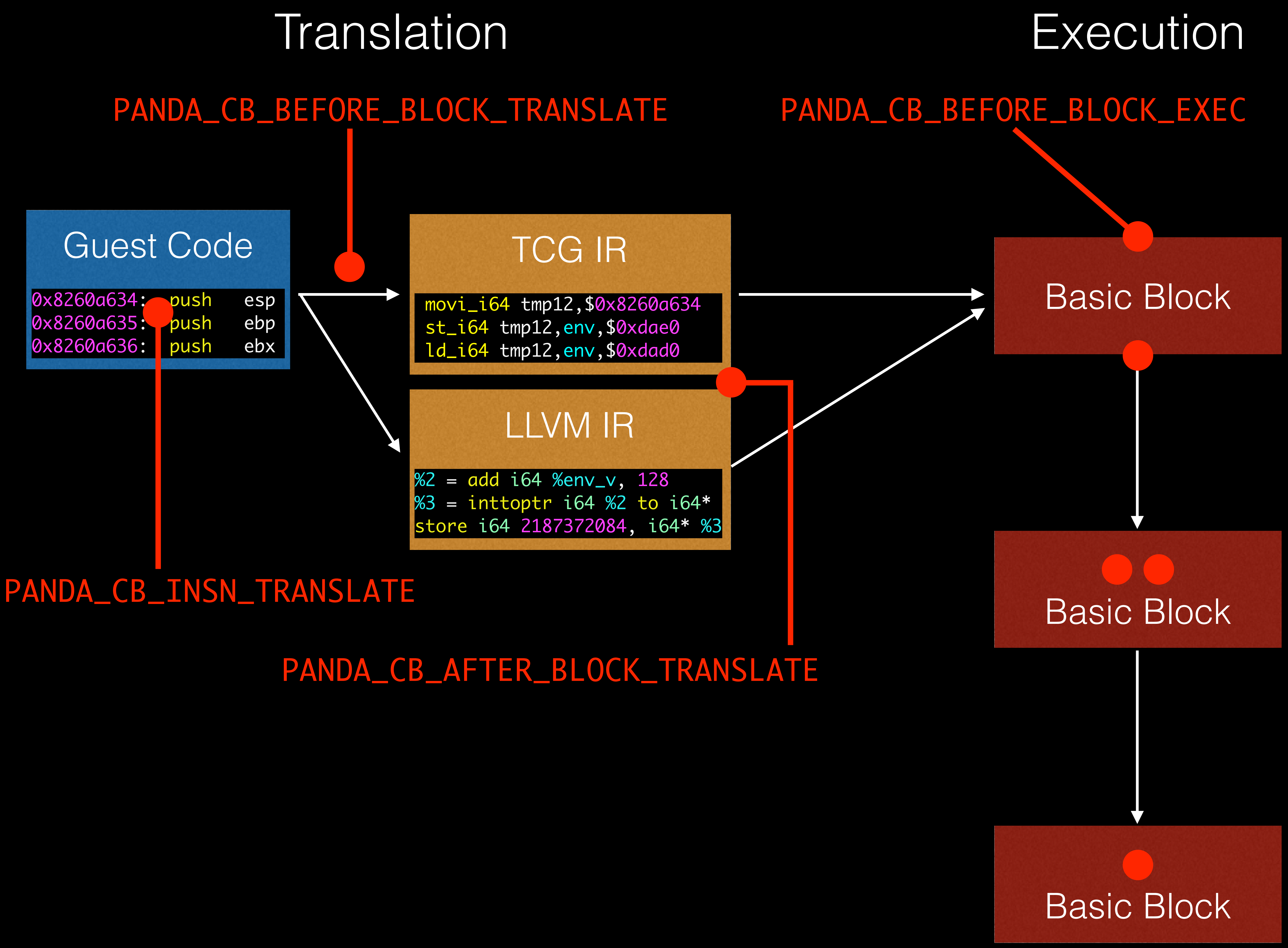
PANDA_CB_INSN_TRANSLATE

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

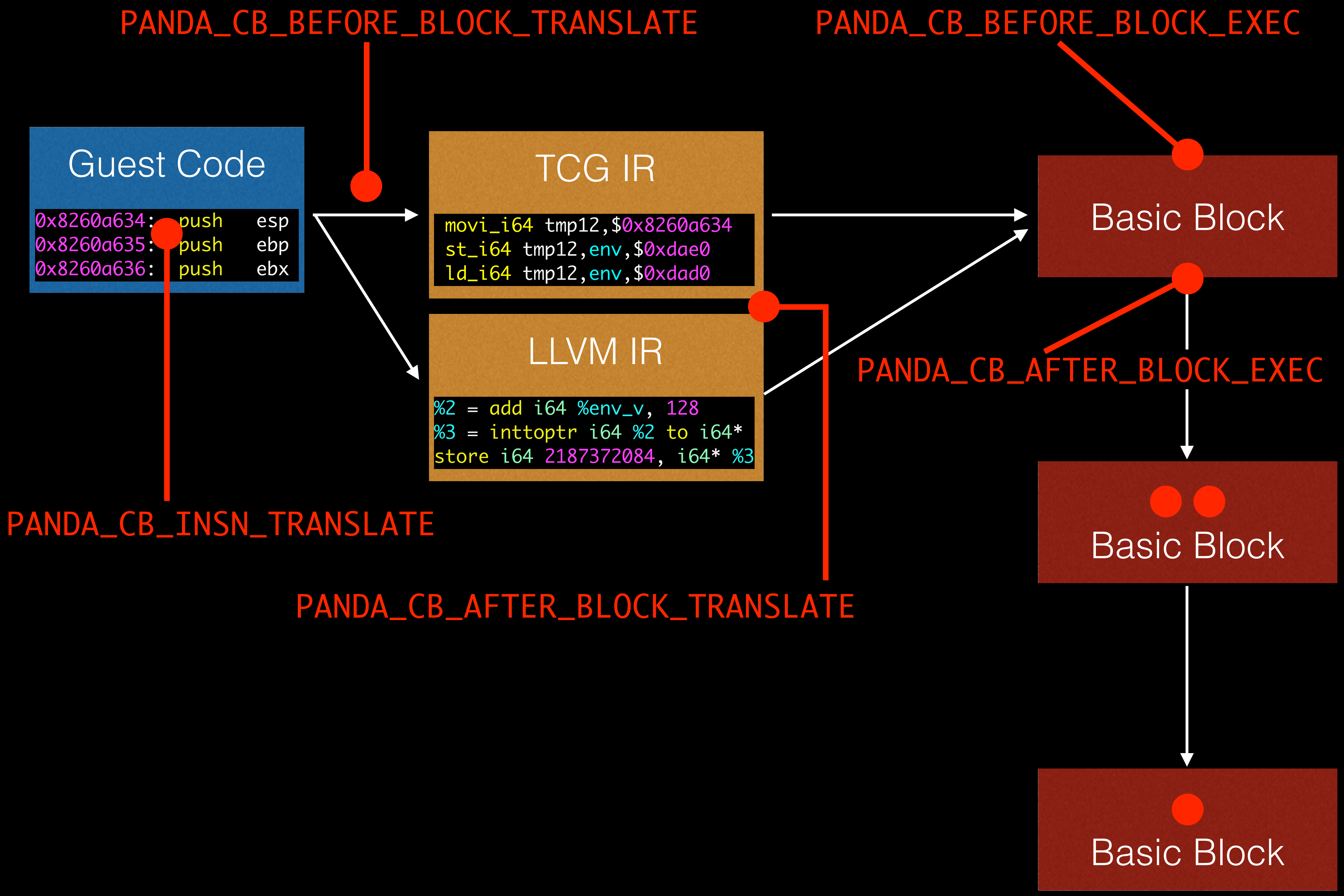
```
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
```

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

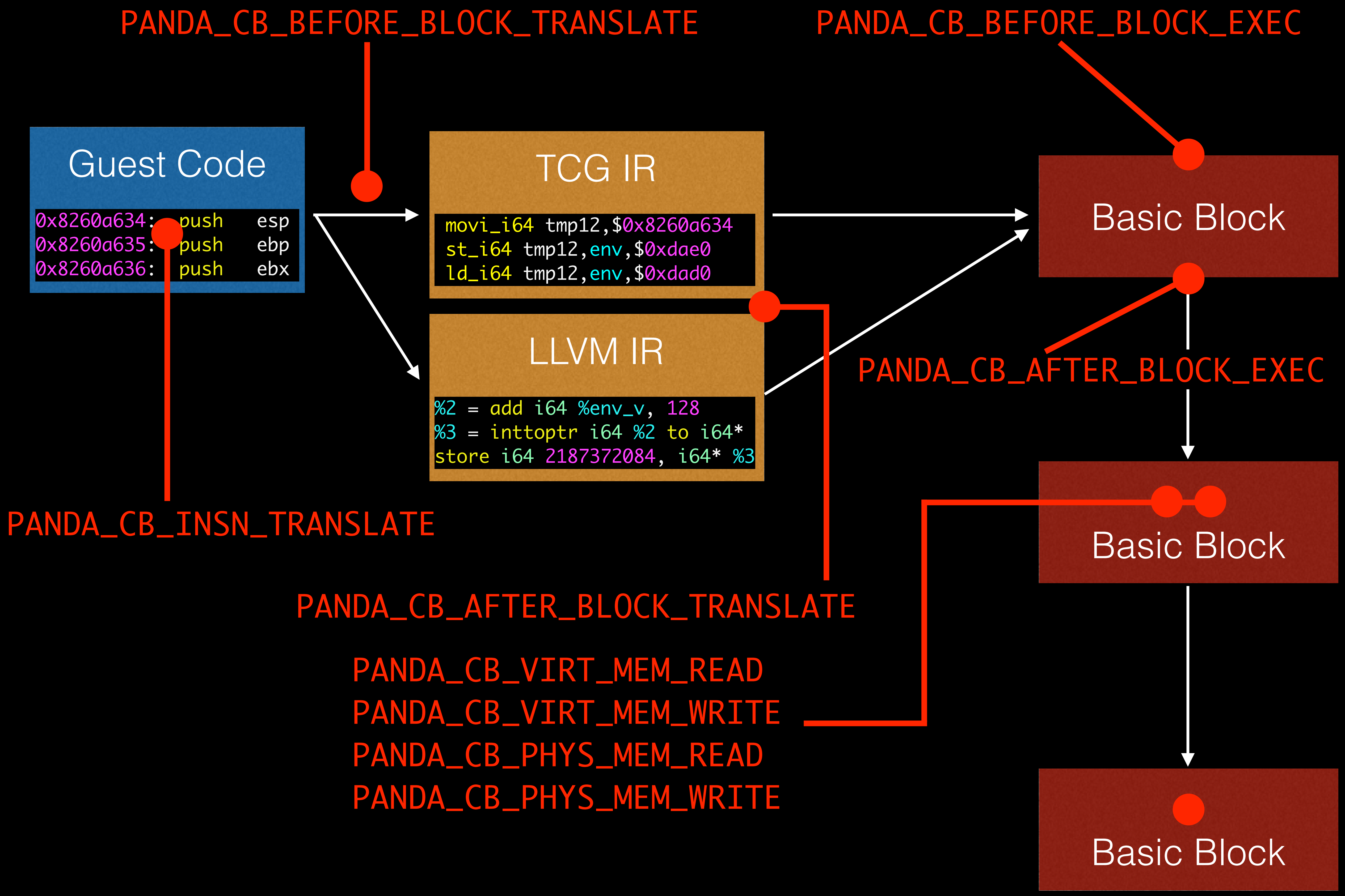
Basic Block

PANDA_CB_AFTER_BLOCK_TRANSLATE

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

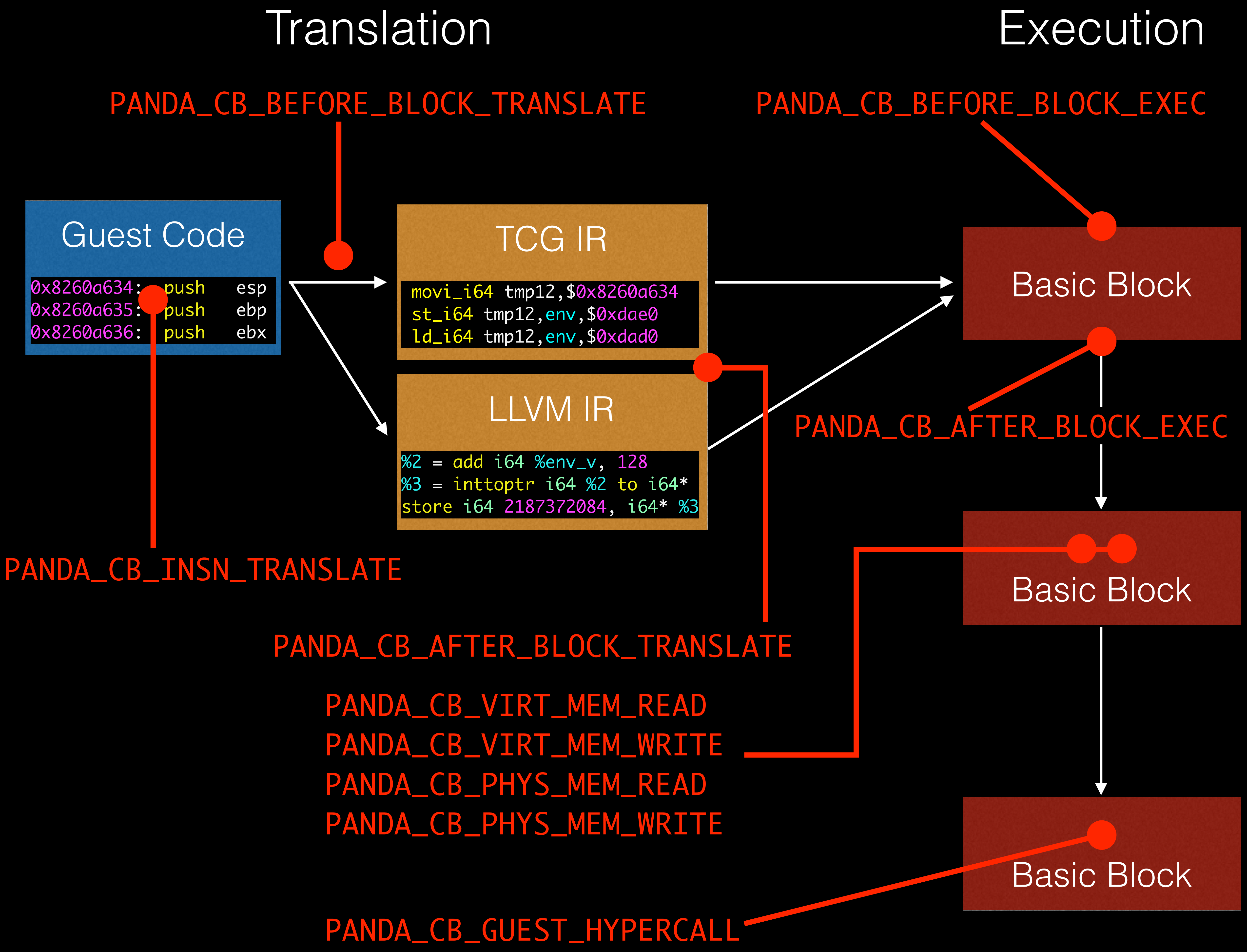
Basic Block

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



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
```

PANDA_CB_AFTER_BLOCK_EXEC

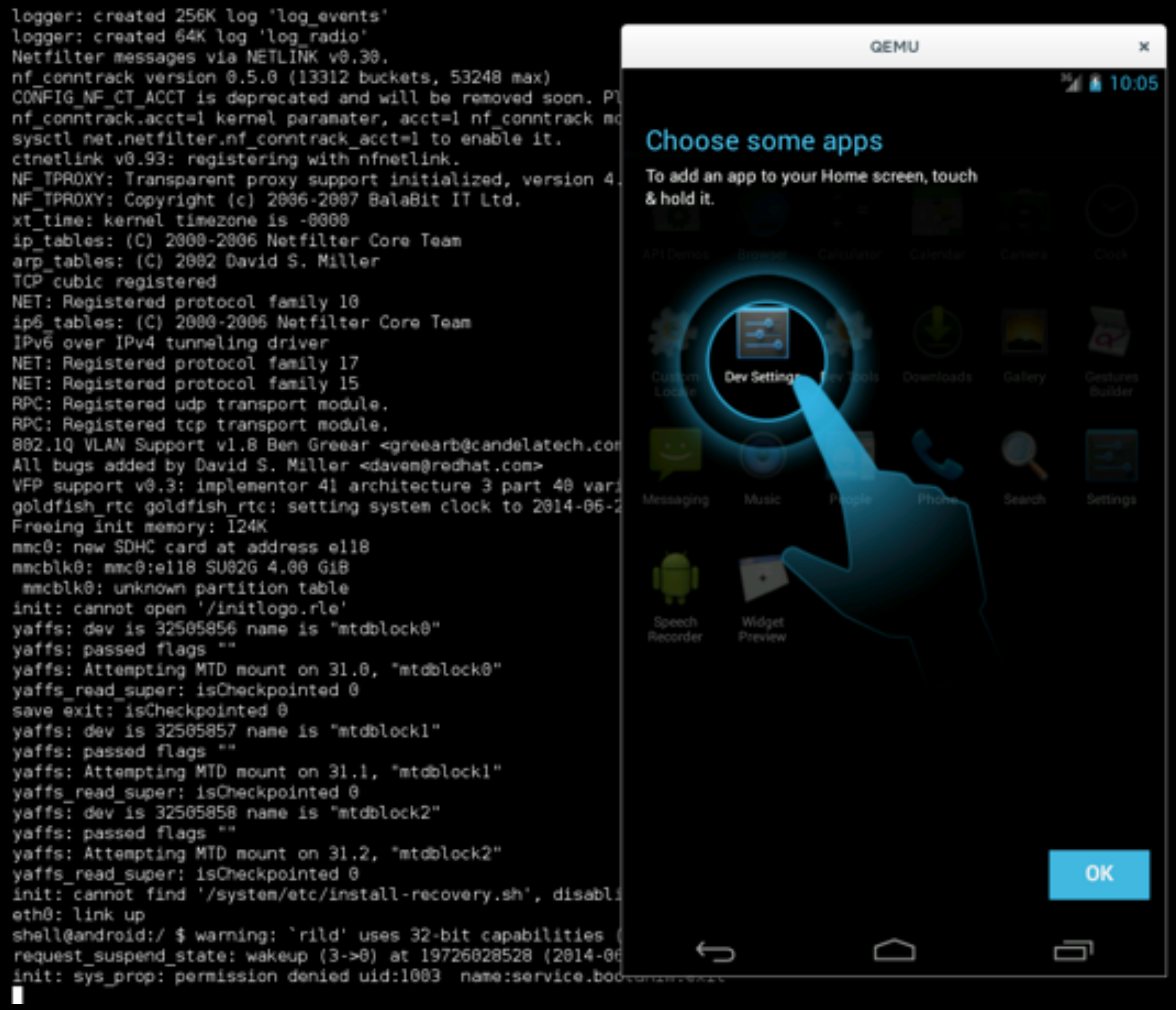
PANDA_CB_INSN_TRANSLATE

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

PANDA_CB_GUEST_HYPERCALL

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. Pl
nf_conntrack.acct=1 kernel paramater, acct=1 nf_conntrack mo
sysctl net.netfilter.nf_conntrack_acct=1 to enable it.
ctnetlink v0.93: registering with nfnetlink.
NF_TPROXY: Transparent proxy support initialized, version 4.
NF_TPROXY: Copyright (c) 2006-2007 BalaBit 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 vari
goldfish_rtc goldfish_rtc: setting system clock to 2014-06-2
Freeing init memory: 124K
mmc0: new SDHC card at address e118
mmcblk0: mmc0:e118 SU02G 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', disabli
eth0: link up
shell@android:/ $ warning: `rild' uses 32-bit capabilities (
request_suspend_state: wakeup (3->0) at 19726028528 (2014-06
init: sys_prop: permission denied uid:1003  name:service.bo
█
```

Choose some apps

To add an app to your Home screen, touch & hold it.

- 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

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

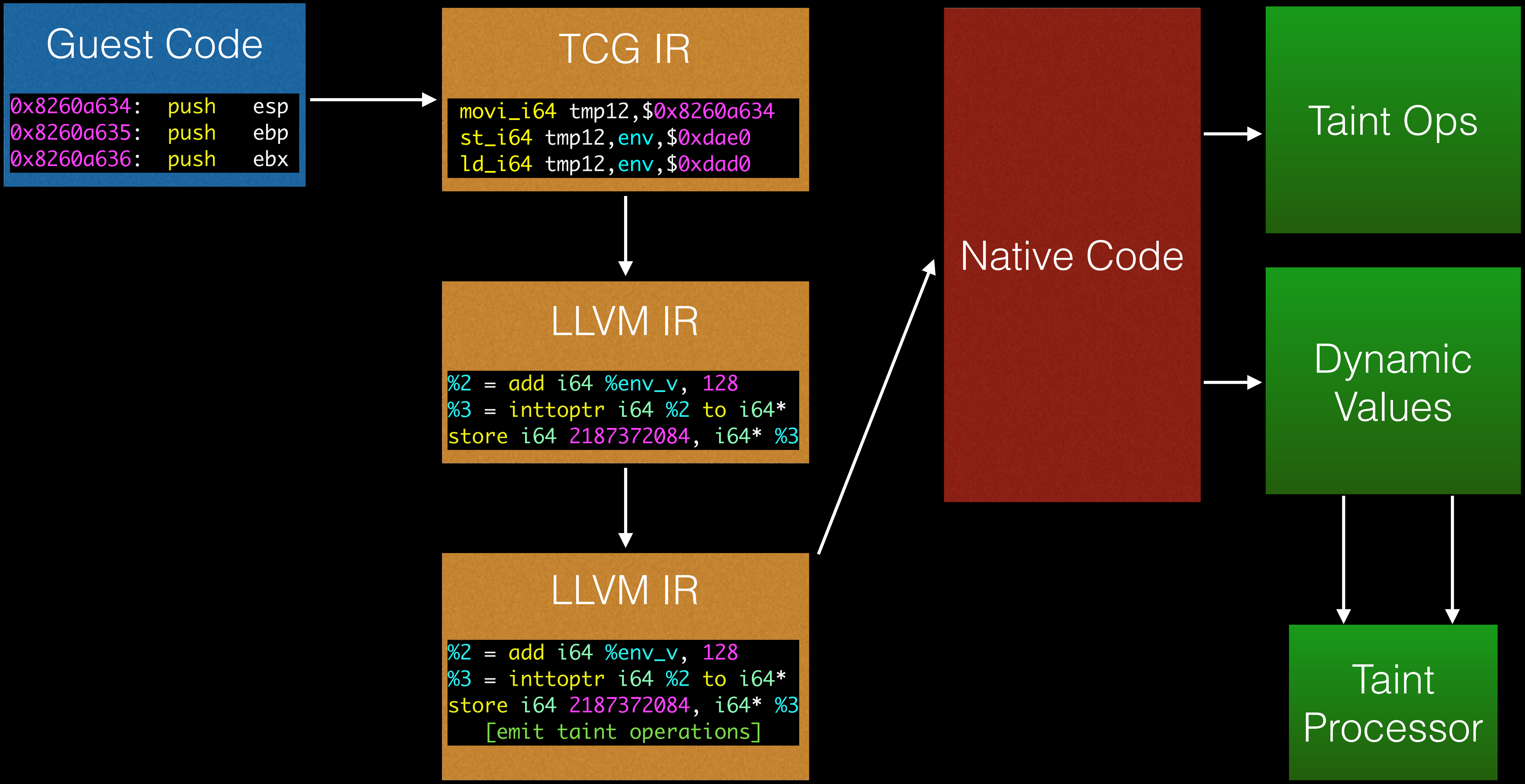
- Track calling context with *callstack* plugin
- At every memory access (PANDA_CB_PHYS_MEM_READ/WRITE)
  Get (caller, program counter, address space) – i.e., *tap point*
- Analyze data flowing through tap point (e.g., string matching with *stringsearch* plugin)

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



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
```

LLVM IR

```
%2 = add i64 %env_v, 128
%3 = inttoptr i64 %2 to i64*
store i64 2187372084, i64* %3
   [emit taint operations]
```

Native Code

Taint Ops

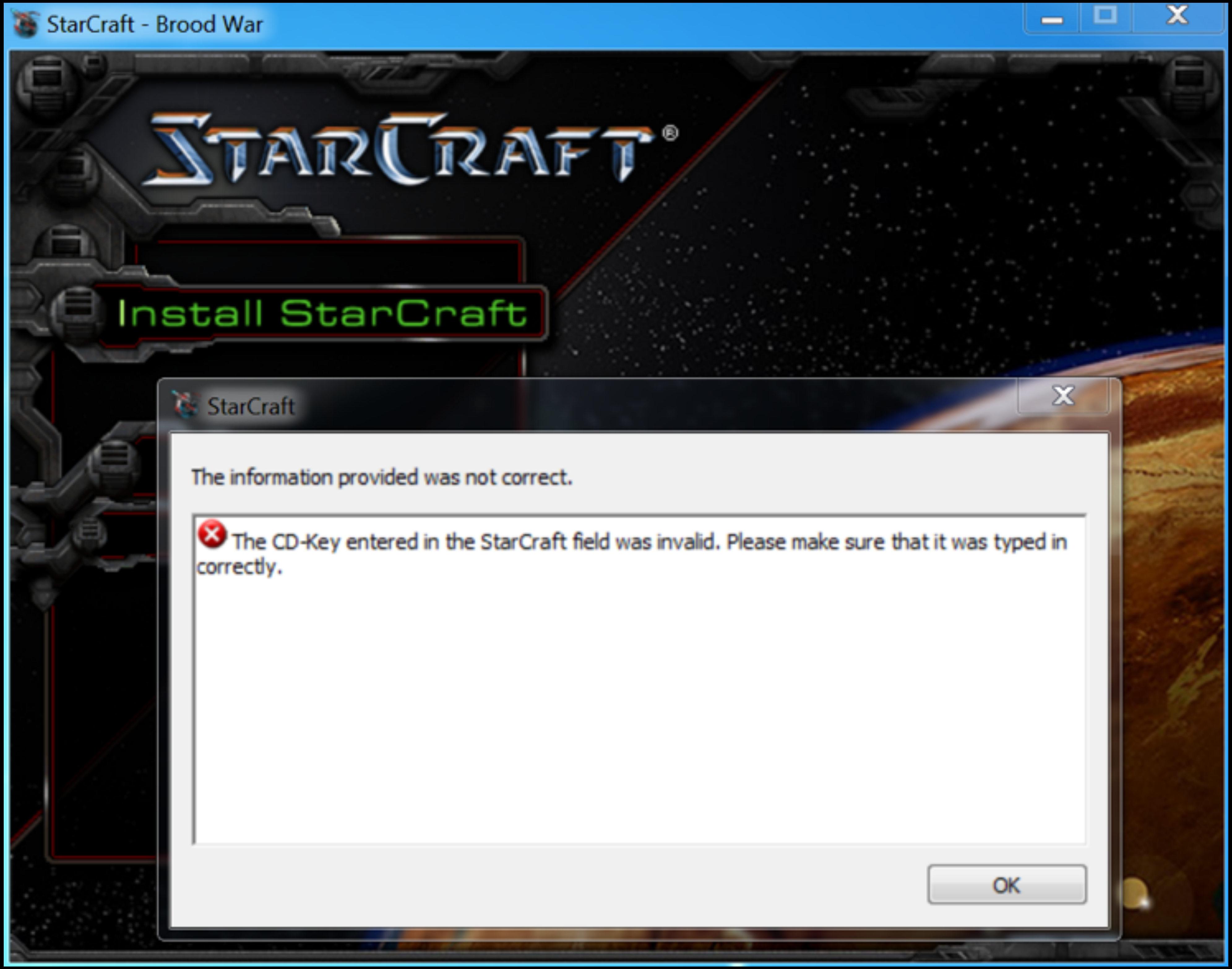
Dynamic Values

Taint Processor

Other Notable Plugins

- *scissors*: extracts out a subset of a replay log
- *callstack*: maintains a shadow callstack
- *replaymovie*: takes frame buffer snapshots during replay and creates a movie
- *syscalls*: provides callbacks for Linux system calls and their arguments

Starcraft CD Key



The information provided was not correct.

The CD-Key entered in the StarCraft field was invalid. Please make sure that it was typed in correctly.

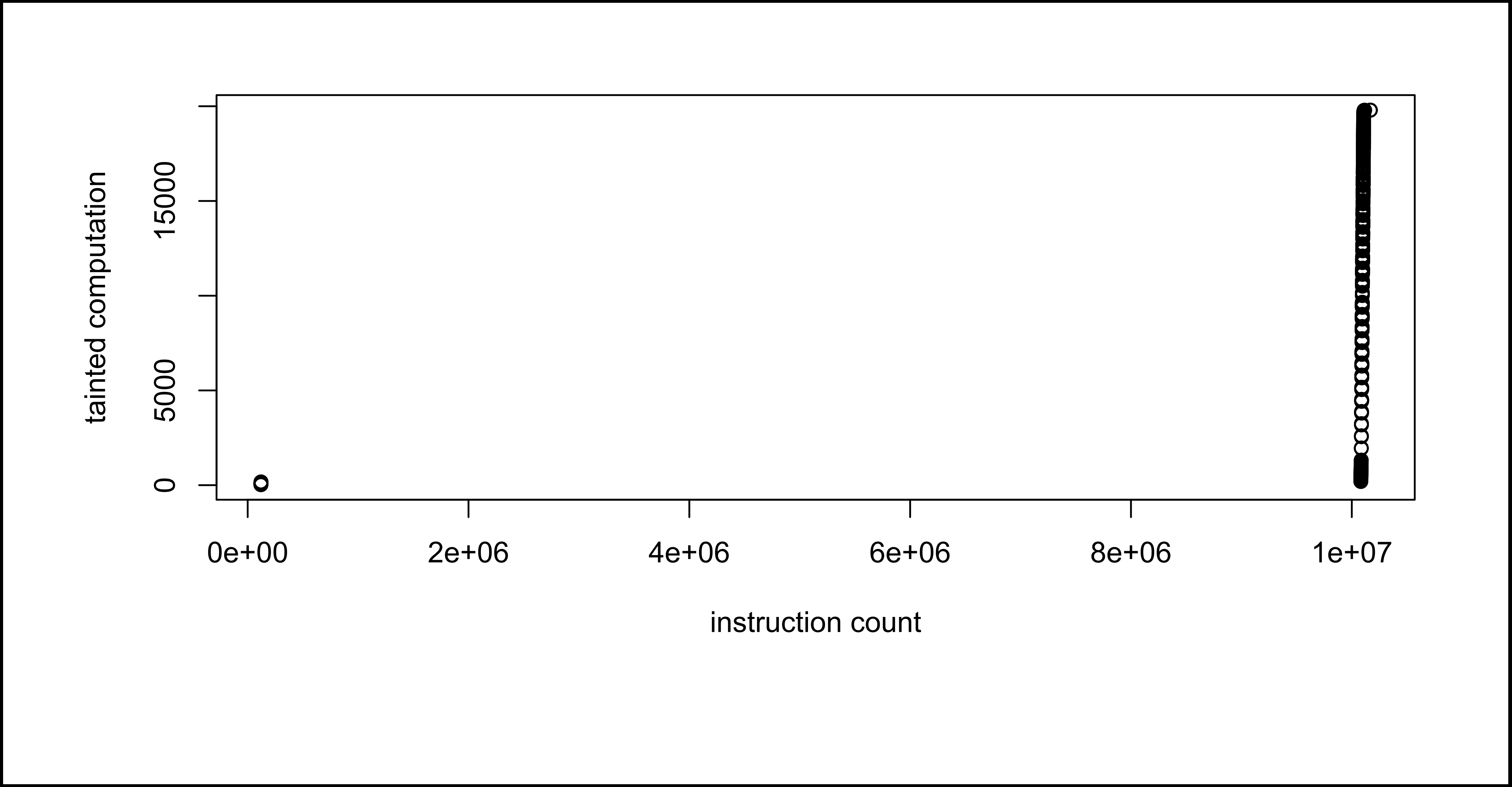
Starcraft RE

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

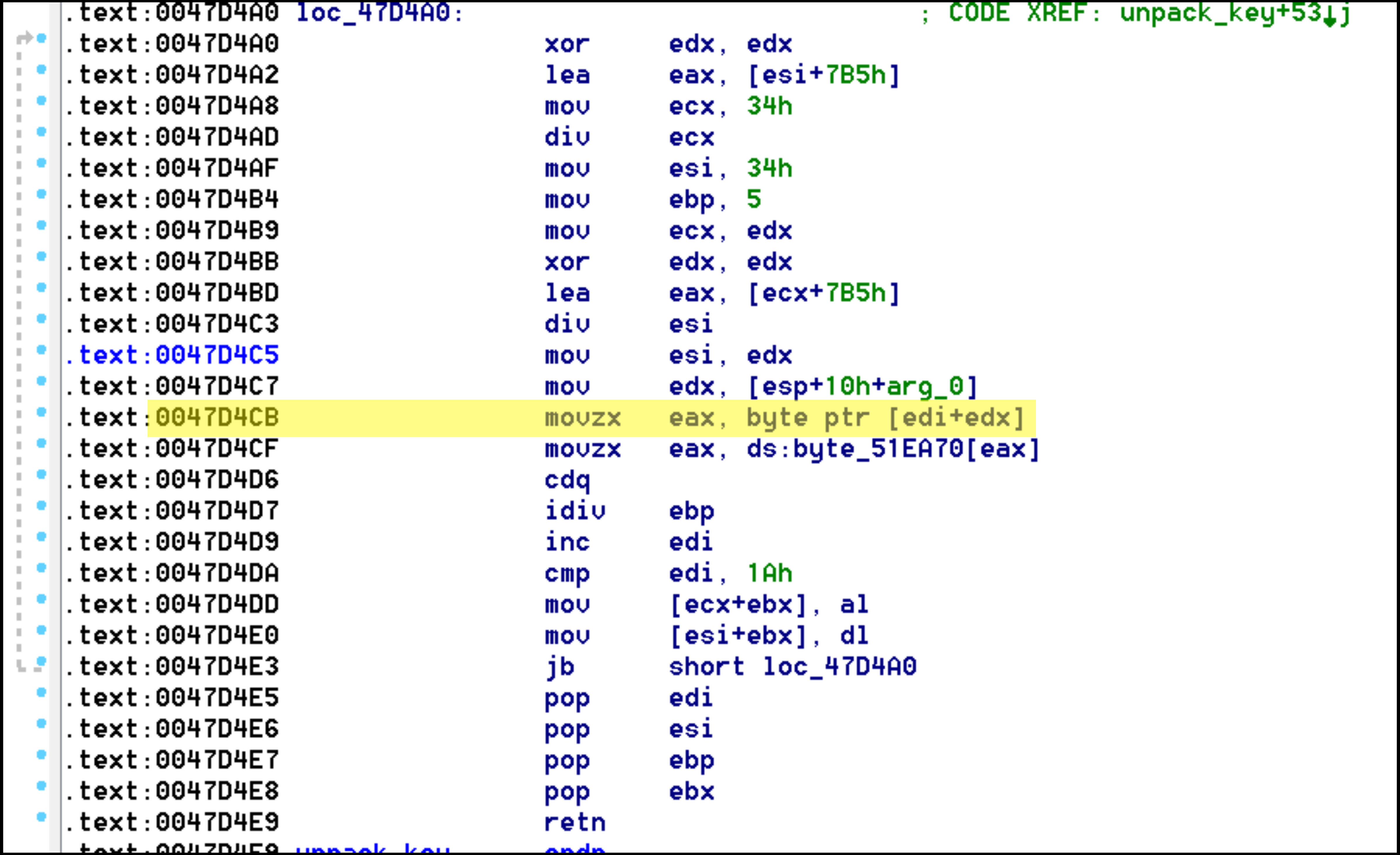
Caller 5	Caller 4	Caller 3	Caller 2	Caller 1	PC	CR3	
		0045c252	00428867	004286ff	0044c951	06cba000	1
0045c252	00428867	004286ff	0044c83b	0047d949	0047d4cb	06cba000	1

- Or, taint key and measure computation done on tainted data
    - i.e.: a = b + c
      tcn(a) = max(tcn(a), tcn(b)) + 1

Starcraft
Tainted Computation

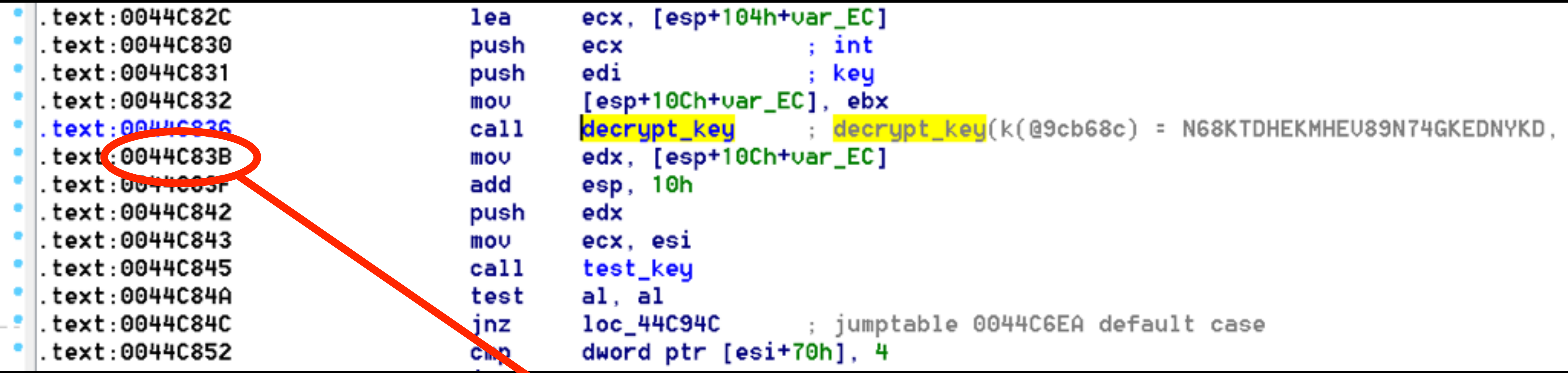


Key Load



```
.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
```

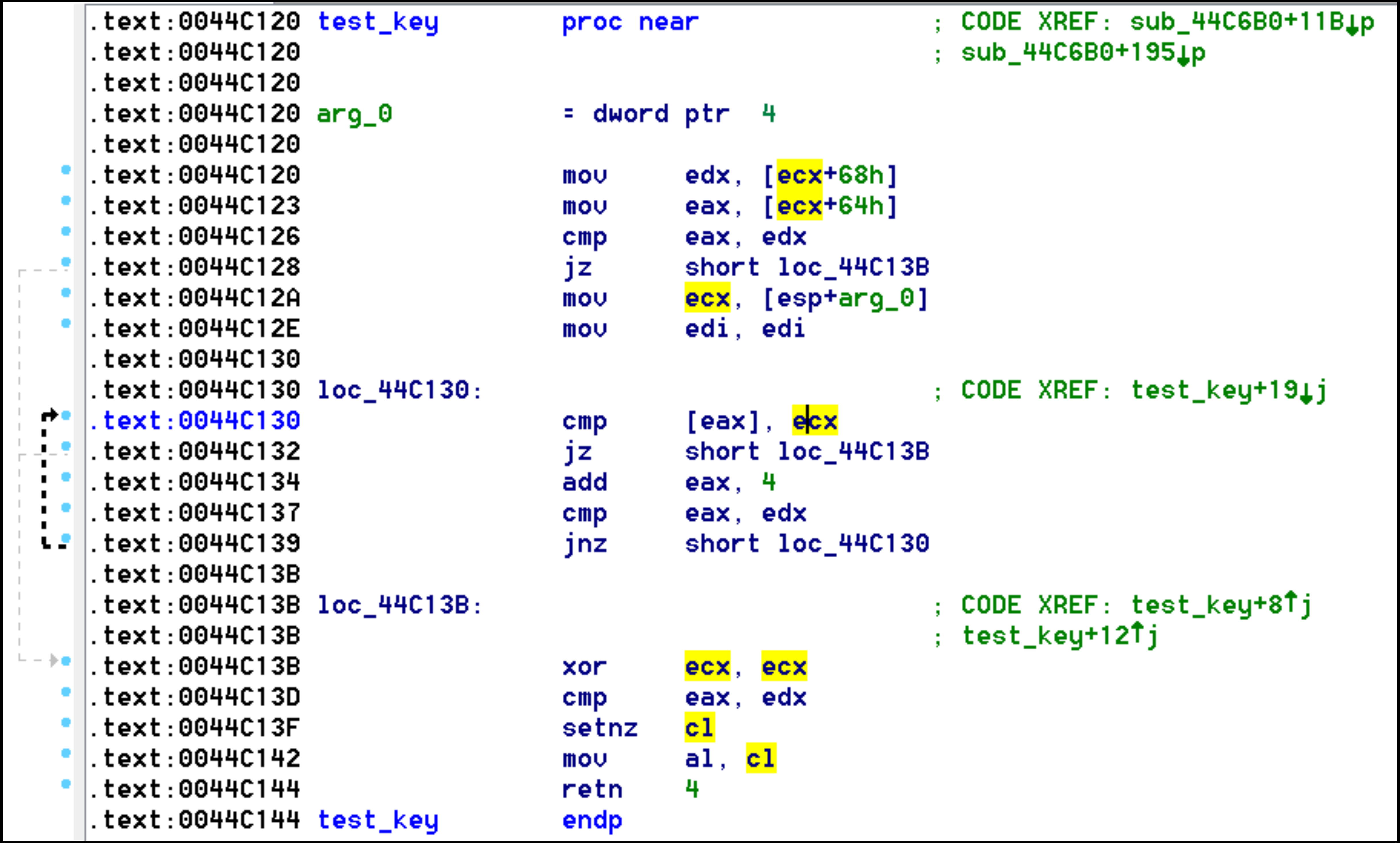
Stepping Out



```
.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
```

```
0045c252 00428867 004286ff 0044c83b 0047d949 0047d4cb 06cba000   1
```

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
```

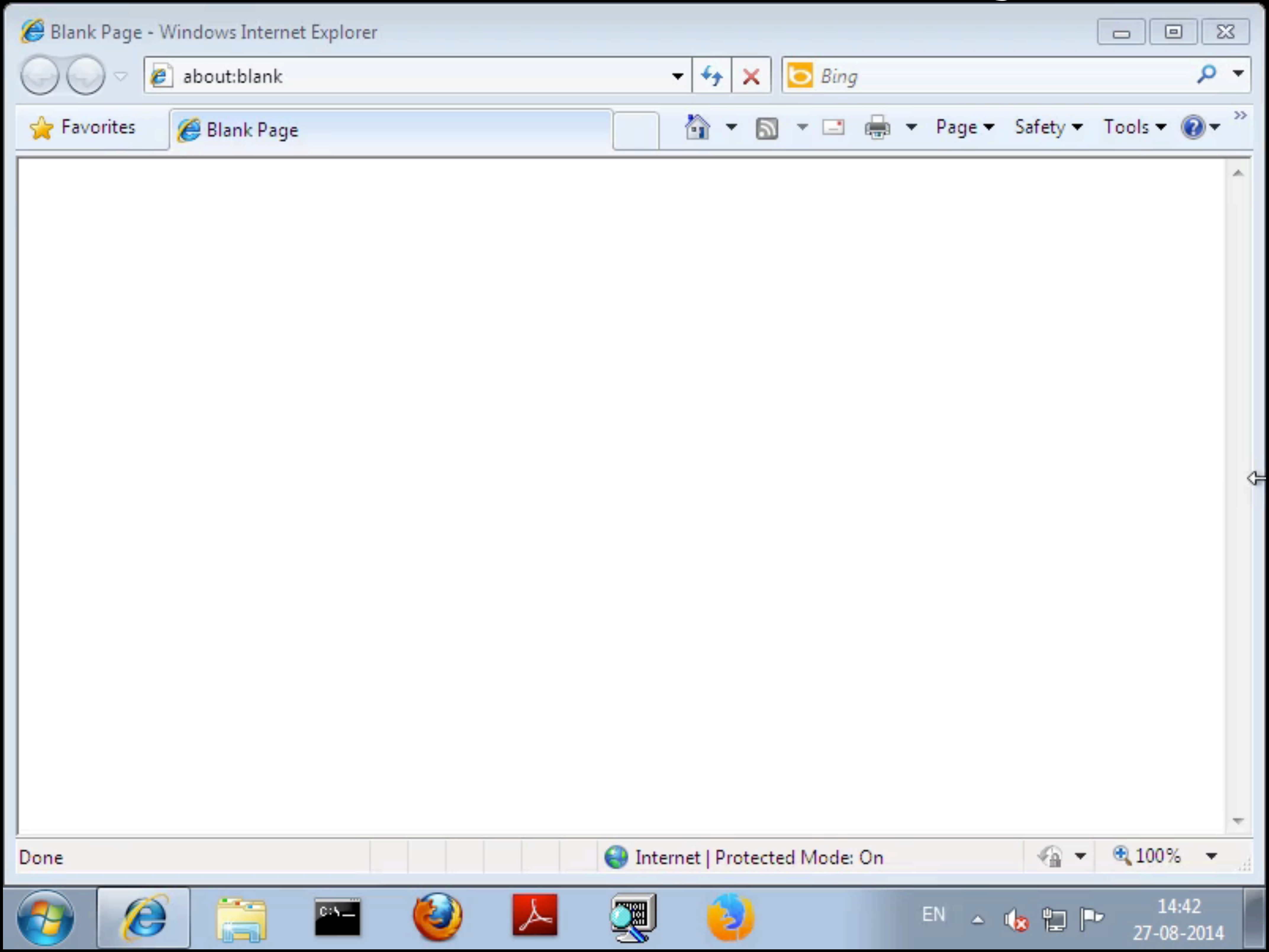
Panda Plugin

```
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;
}
```

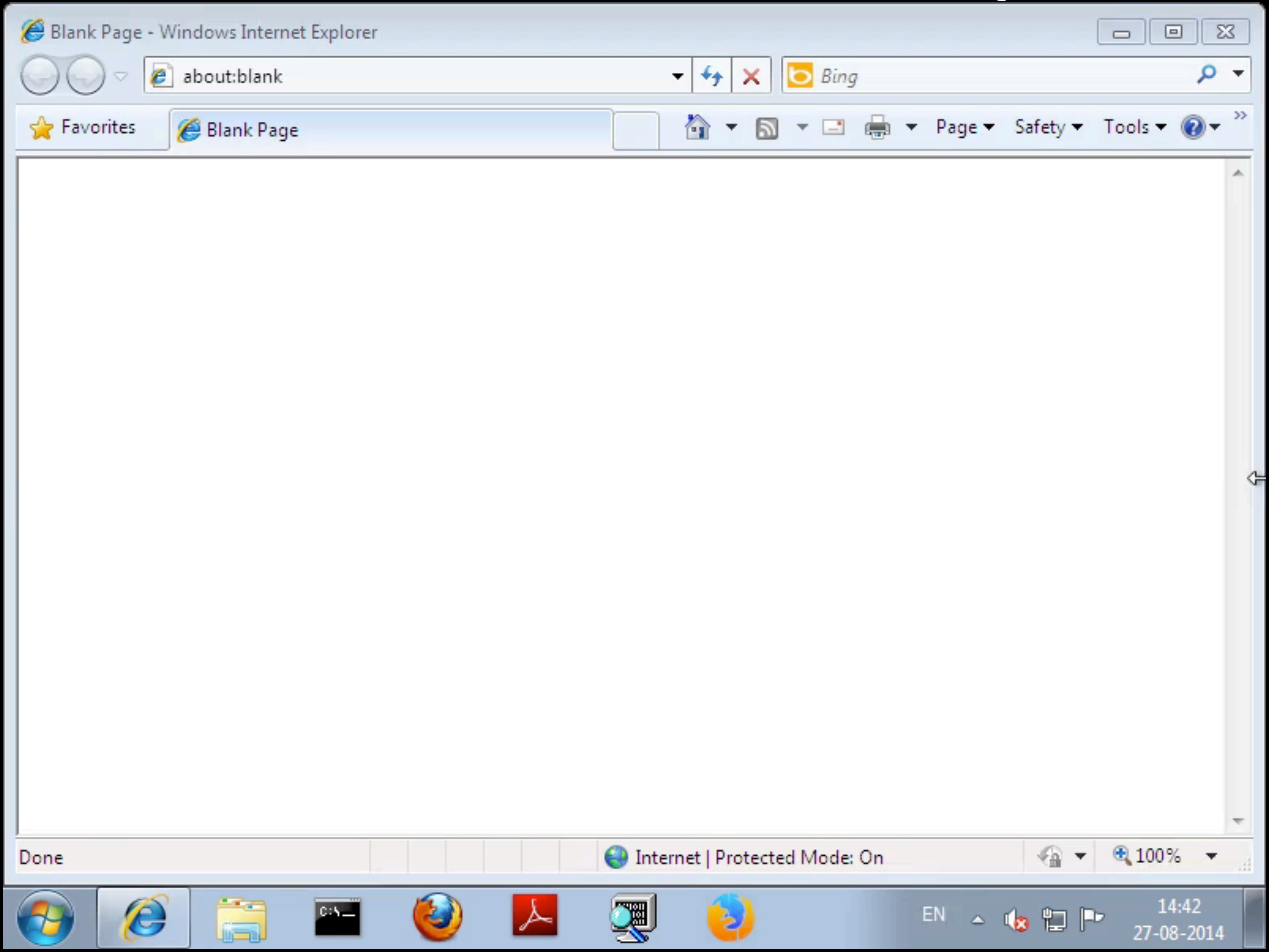
Output

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

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:

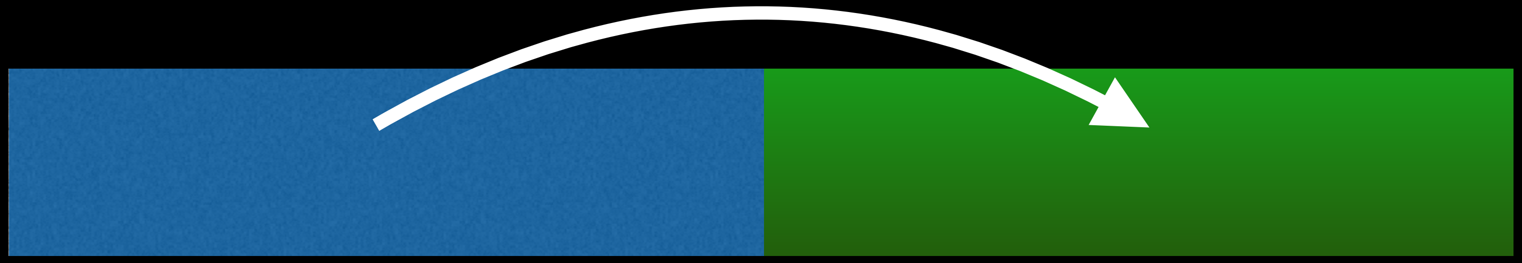
Use After Free Detector

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

Use After Free Detector

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

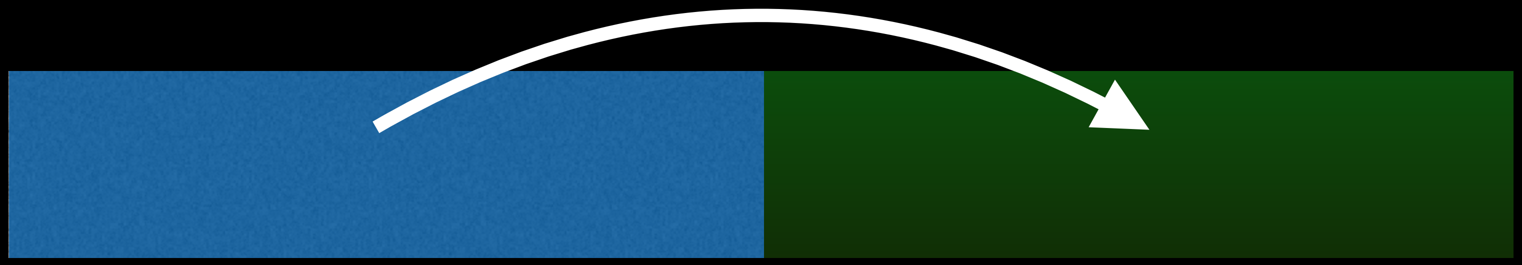
Heap:



Use After Free Detector

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

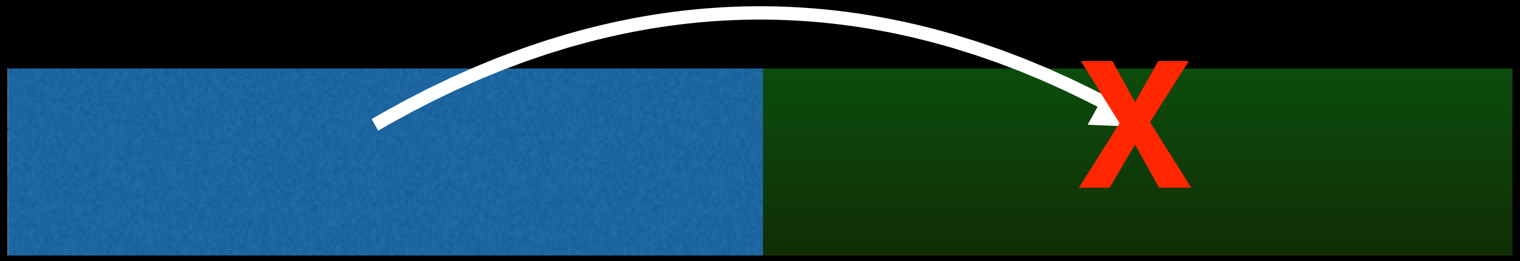
Heap:



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

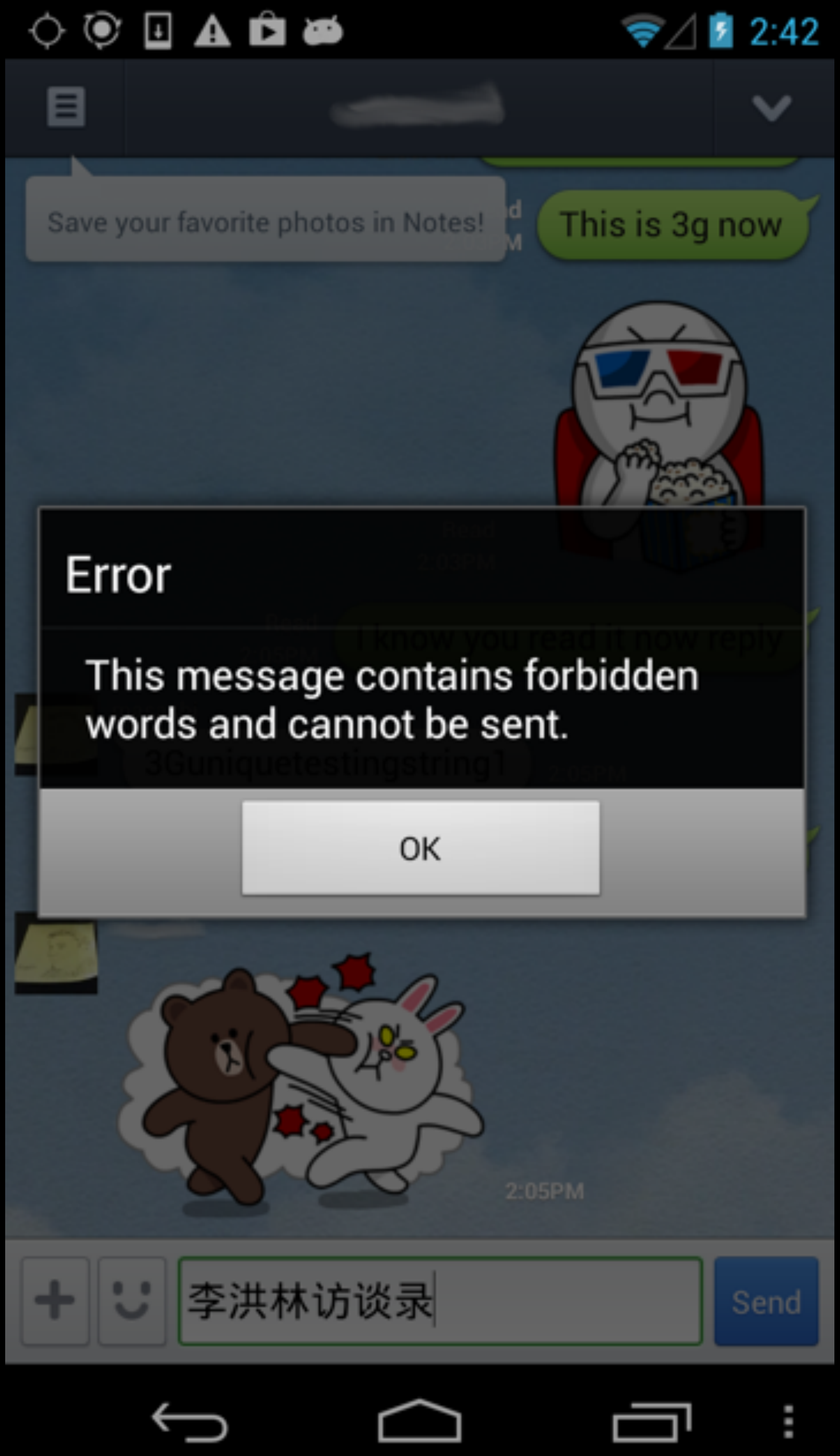
- UAF detector finds exactly one match:

```
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

[redacted]

Save your favorite photos in Notes!

This is 3g now

I know you read it now reply

3Guniquetestingstring1

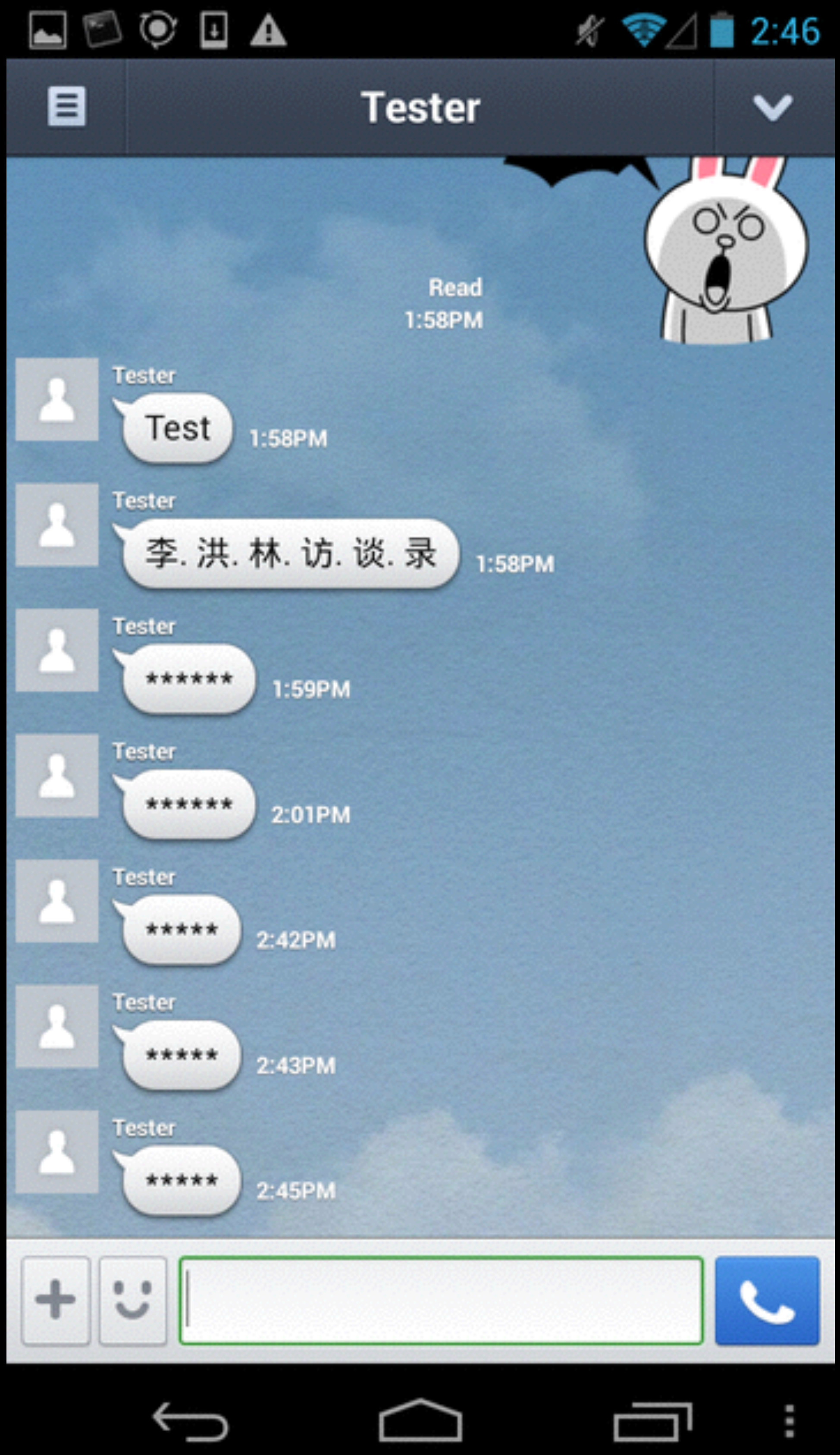
Error

This message contains forbidden words and cannot be sent.

OK

2:05PM

李洪林访谈录



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
FLG
GCD
GFW
18大
38军
八九
半羽
鲍彤
暴政
柴玲
赤匪

共党
共匪
共贼
胡温
江派
江系
江贼
近平
九评
军警
六四
马凯
民运

彭博
天朝
兲朝
屠城
屠杀
团派
退党
汪洋
瘟神
晓波
学潮
学运
余杰

政变
周斌
祖莹
共C档
08宪章
89事件
艾未未
薄瓜瓜
薄熙来
曹建明
曾庆红
陈光诚
大纪元

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

Conclusion

- Reverse engineering is a useful, legitimate technique that deserves more academic study!

- In order to have confidence in closed-source software we must be able to RE it

- PANDA can help dramatically speed up RE tasks through dynamic analysis

Credits

- PANDA devs
  - Tim Leek (MIT Lincoln Lab)
  - Patrick Hulin (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 on Twitter
  brendan@cs.columbia.edu
- Join the mailing list: panda-users@mit.edu
- IRC Channel: #panda-re on Freenode
- Contribute code:
  https://github.com/moyix/panda