
# Reverse Engineering with Hardware Debuggers

**11 Mar 10**

**JASON RABER and JASON CHEATHAM**
**ATSPI Assessment Science Team**
**RYTA**
**Air Force Research Laboratory**

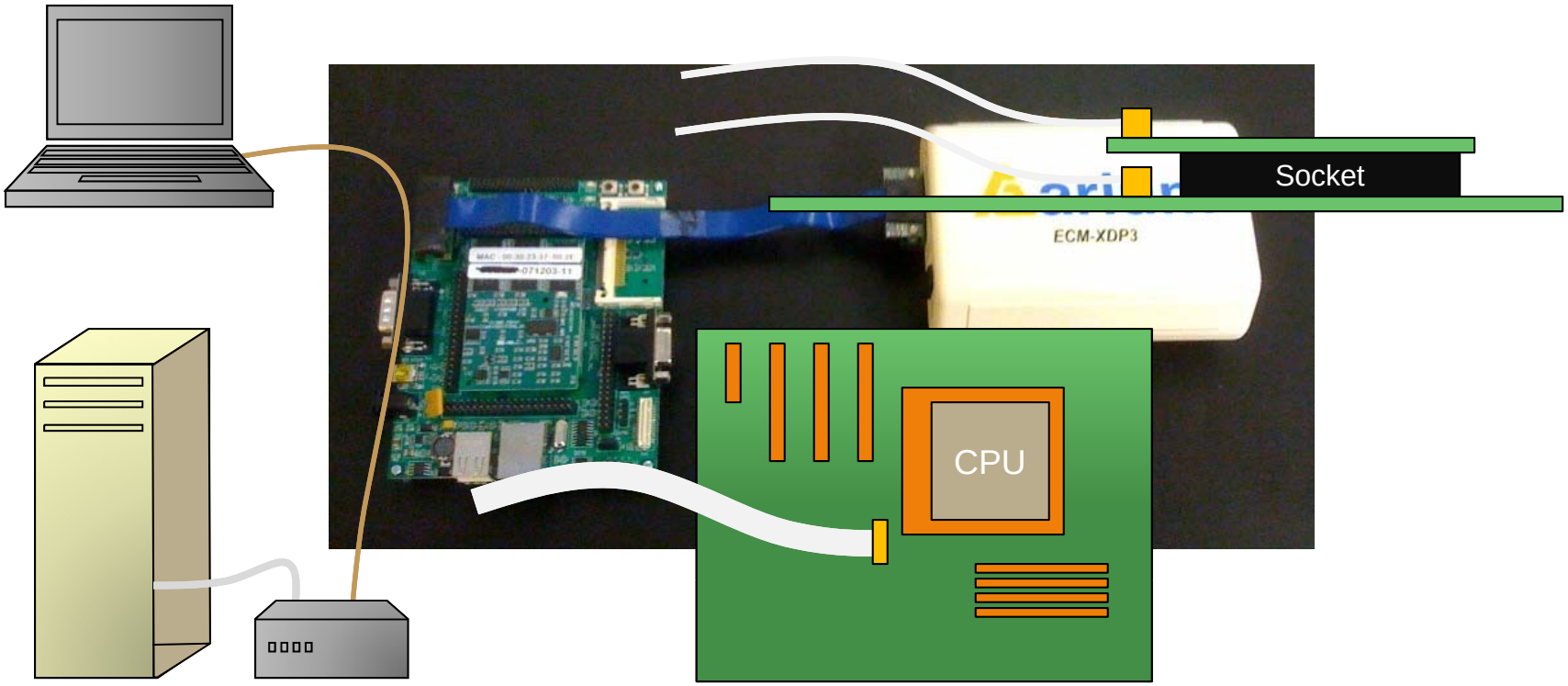
Public release authorization 88 ABW-10-1497

# Outline

- Architecture
- Breakpoints
- Macros
- Hypervisors

# The Hard Way

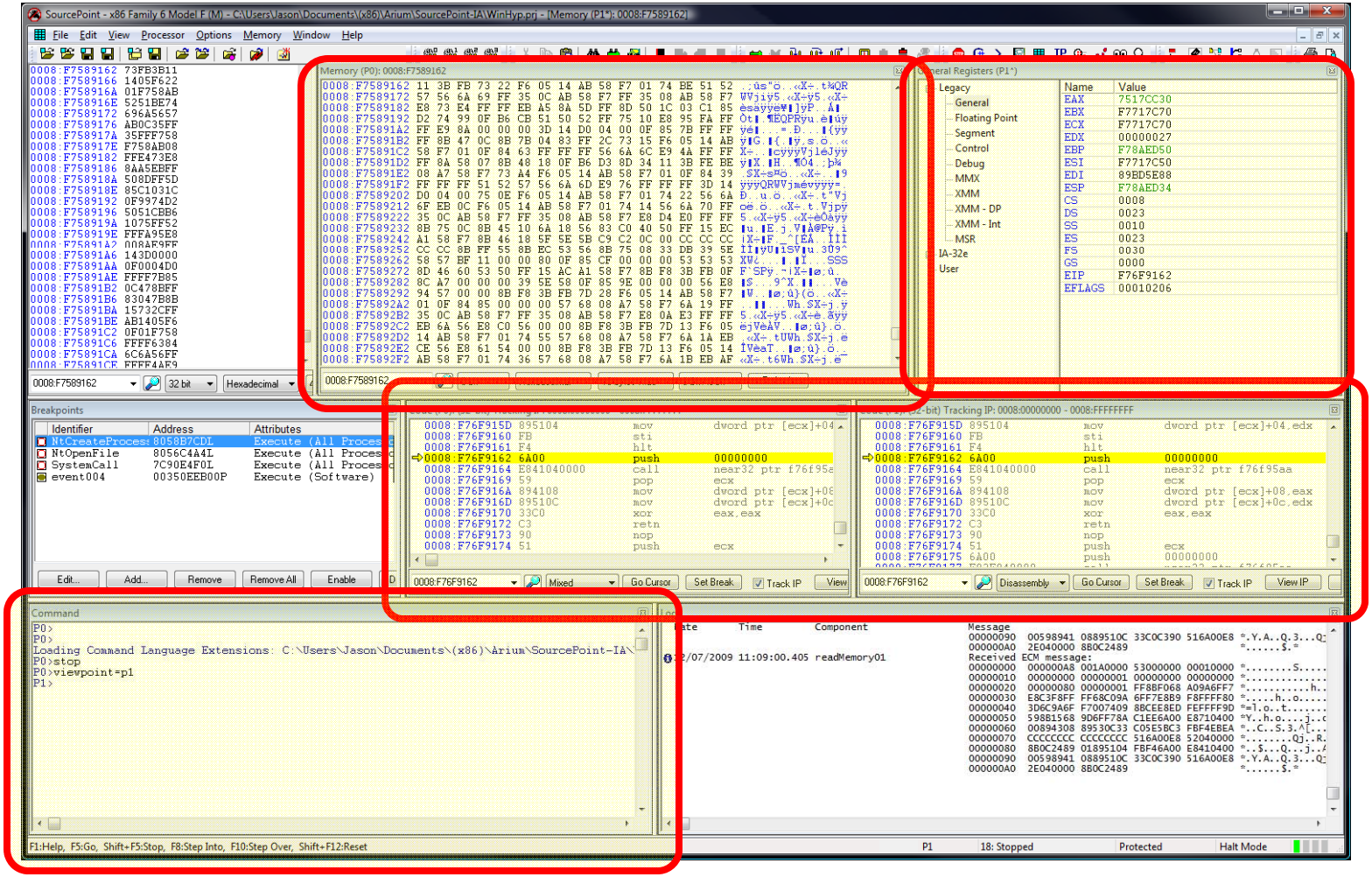
**Hardware debugger / in-target probe (ITP) / in-circuit emulator (ICE)**



# Friends with Benefits

- **Hardware debuggers can see almost everything**
- **They live outside the OS, so even kernel mode rootkits can't hide**
- **Rewriting firmware**
- **They're OS independent**
  - Just needs a compatible x86 processor

# The Softer Side



# Outline

- Architecture
- Breakpoints
- Macros
- Hypervisors

# Ways to Break Things

- **Hardware Breakpoints**
  - DR registers
- **Software Breakpoints**
  - ICEBP (0xF1) instruction + DR7 bit 12
- **Infinite loops**
  - Steal a couple bytes and replace with 0xEBFE

# Infinite Breakpoints

- **EB FE is a jump to the same address (jmp $)**
  - Inject the infinite loop (0xEBFE) into the application or driver
  - Halt the CPU when the system freezes, and there you are
- **They're very easy to use with an ICE**
  - No worries about freezing the system
  - Don't have to deal with virtual memory
- **Checksums can detect these, so place them carefully**

# Outline

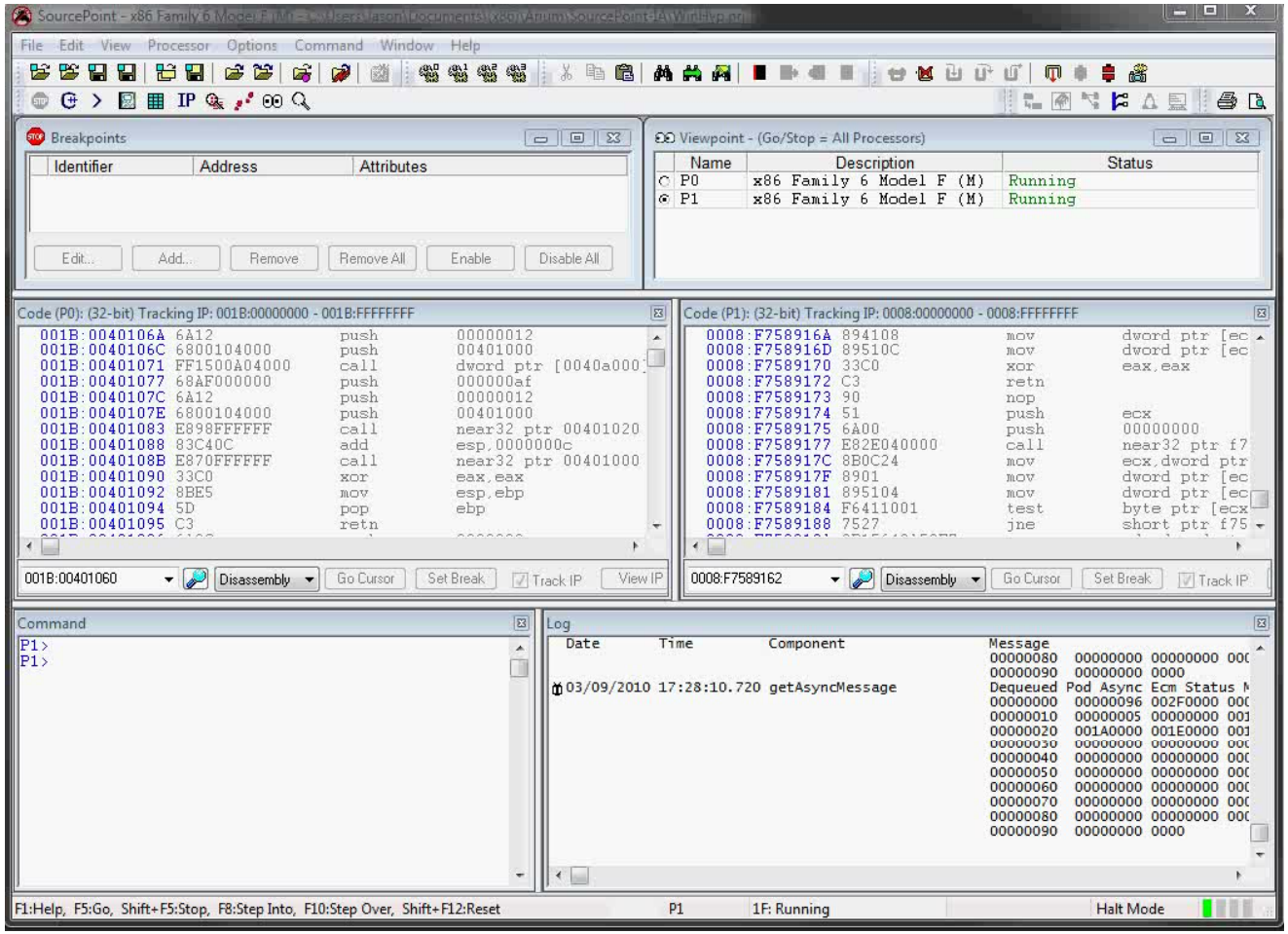
- Architecture
- Breakpoints
- Macros
- Hypervisors

# More Power…

- **Macros can make an emulator very powerful**
  - Implement complex or repetitive tasks
  - Detailed control of ICE
- **SourcePoint uses a C like scripting language**
  - Variables, functions, control flow
  - Types have well-defined widths
    - ord1, int4, real8, …
  - Control statements for ICE

```
define proc pcrange(startaddr,
   endaddr)
  define ord4 startaddr
  define ord4 endaddr
{
  while (1) {
    if (EIP >= startaddr &&
        EIP <= endaddr) {
      break
    } else {
      step 4
    }
  }
}
```

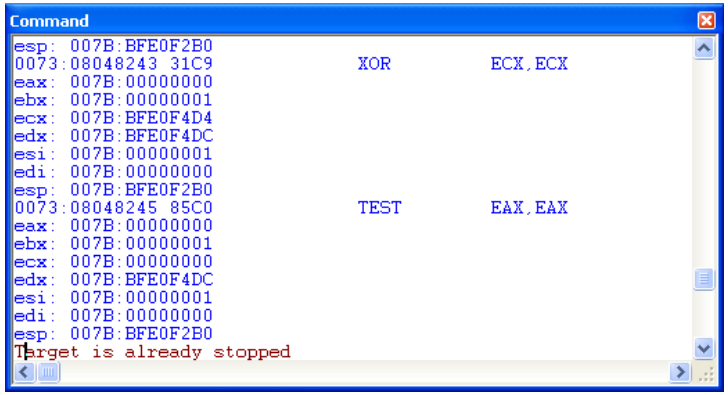
# Range Breakpoint



# Run Trace

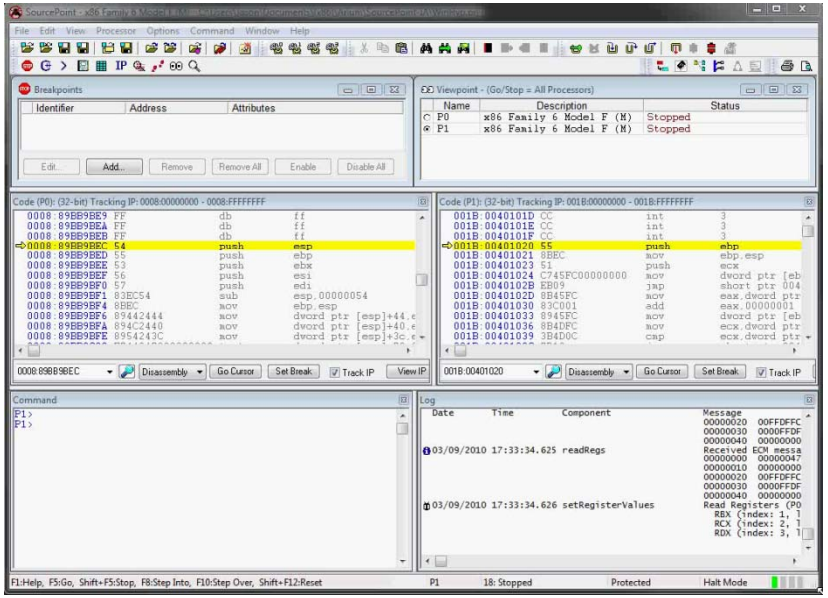
```
flist (“tracelog.txt”, 1)
define ord4 lasteip = EIP
softremove
while (1) {
    if (EIP >= 0xC0000000 && lasteip < 0xC0000000) {
        softbreak = location=lasteip+2
        softbreak = location=lasteip+3
        softbreak = location=lasteip+4
        softbreak = location=lasteip+5
        softbreak = location=lasteip+6
        lasteip = EIP
        go
    }

    if (EIP != endaddr) {
        asm eip
        printf("eax: “); eval eax virtual
        printf("ebx: “); eval ebx virtual
        printf("ecx: "); eval ecx virtual
        printf("edx: "); eval edx virtual
        printf("esi: "); eval esi virtual
        printf("edi: "); eval edi virtual
        printf("esp: "); eval esp virtual
        lasteip = EIP
        step
    } else {
        nolog
        stop
    }
}
```



# From the Outside In

- **Emulator access is pretty raw, so we wrote some simple forensic macros**
  - list_procs()
  - get_ssdt()
  - get_syscall_addr(name)
  - hook_syscall(name)
  - list_mods()



# get_syscall_addr

```
define proc pointer get_syscall_addr(syscall_name)
  define nstring syscall_name
{
  if (syscall_name == "NtCreateProcess
    return ssdt_index_to_addr(0x30, 0)
  } else if (syscall_name == "NtCreateSe
    return ssdt_index_to_addr(0x32, 0)
  } else {
    printf("Unkr
    return 0
  }
}
```

```
define proc pointer ssd
  define
{
  define
  define
  return
}
```

```
define proc po
{
  define pointe
  define pointe
  define pointe
  define pointe
  return ssdt
}
```

```
define proc pointer find_gdi_proc() {
  define pointer PsActiveProcessHead = get_head_proc()

  define pointer first_proc = flink_to_proc_addr(PsActiveProcessHead)

  define pointer current_flink = next_proc_flink(first_proc)
  define pointer current_proc = flink_to_proc_addr(current_flink)

  while (current_flink != PsActiveProcessHead)  {
    define ord1 type = proc_type(current_proc)
    if (type != 0x03) { printf("Non-process type: %x\n", type); break; }

    define ord4 w32proc = proc_win32process(current_proc)

    if (w32proc != 0x0) return current_proc

    current_flink = next_proc_flink(current_proc)
    current_proc = flink_to_proc_addr(current_flink)
  }
}
```

# Outline

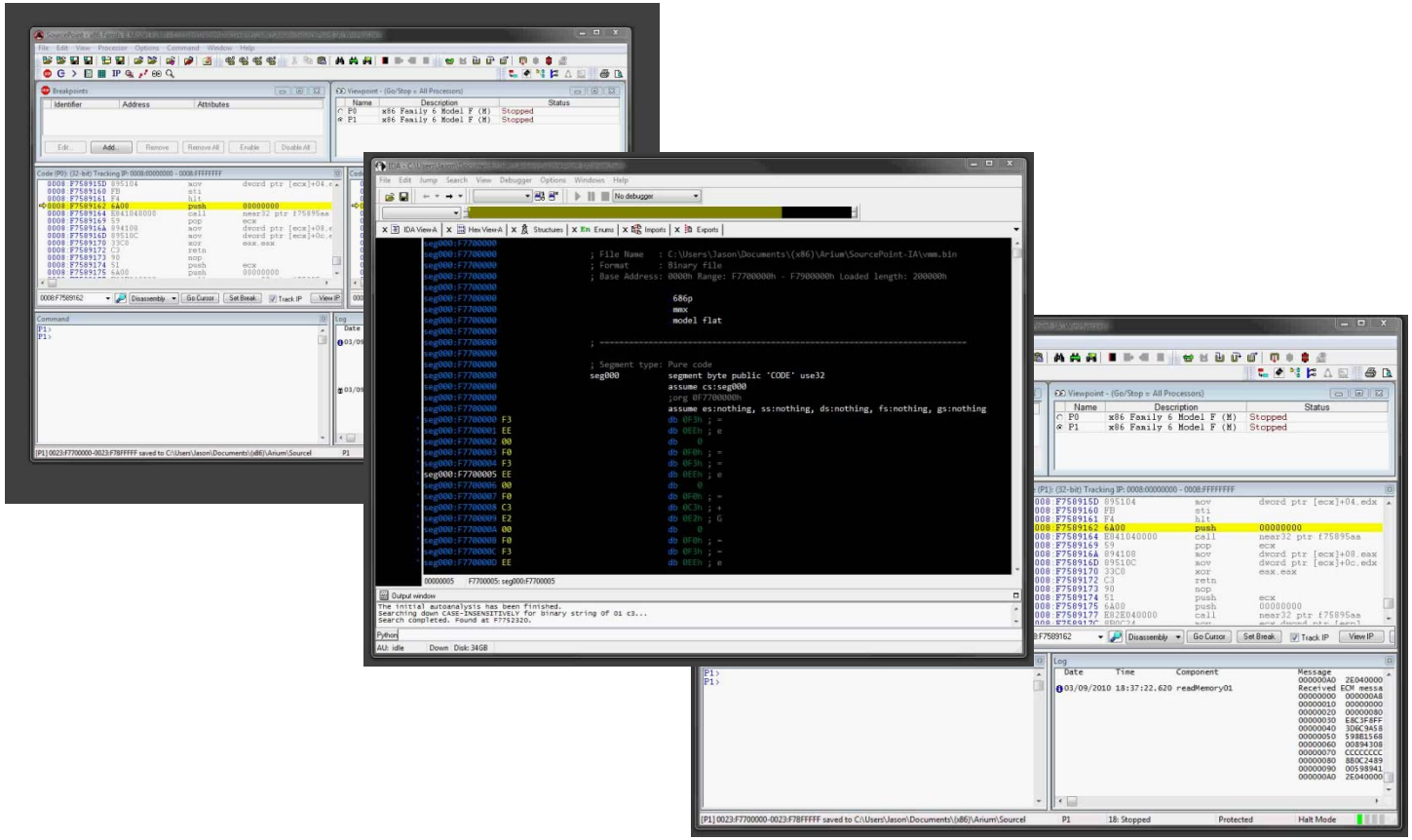
- Architecture
- Breakpoints
- Macros
- Hypervisors

# Blue…Something


- **ICE runs below hypervisors**

- **You can go far with a few simple macros**
  - vmx_is_enabled
  - vmx_goto_resume
  - vmx_guest_eip
  - vmx_exit_reason

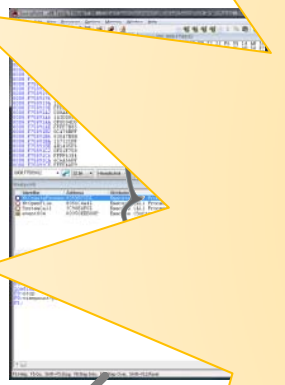
# Down We Go…



# Nobody's Perfect

- **You can't single step or trace across the ring -1/0 boundary**
- **Finding the hypervisor is tricky**
- **Newer VM instructions can cause problems**

# Summary




The Debuginator