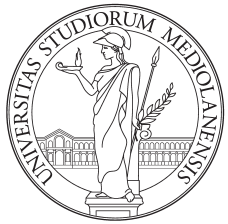

Università degli Studi di Milano
Facoltà di Scienze e Tecnologie
Dipartimento di Informatica

Reverse engineering: *obfuscation*

Srdjan Matic <srdjan@security.di.unimi.it>
Aristide Fattori <joystick@security.di.unimi.it>

A.A. 2013–2014

Hands on!

`http://security.di.unimi.it/sicurezza1314/obfuscation.tar.gz`

Stripping {ex1}

Not really obfuscation...

```
sicurezza@sicurezza2: /obfuscation$ gcc $(cat ../FLAGS ) -o ex1 ex1.c
sicurezza@sicurezza2: /obfuscation$ strip ex1
```

Strip removes unnecessary symbols from an ELF. This is not considered an obfuscation technique *per se*, but it can make things nasty, expecially when mixed with other techniques.

What is the first function executed in an ELF?

How to locate `<main>`?

Junk code {ex2}
Thwarting linear sweep

```
jmp $+5
.byte 0x33, 0xe4, 0xc7, 0x41, 0x42
```

```
8048457: eb 05                   jmp 804845e <main+0xa>
8048459: 33 e4                   xor %esp,%esp
804845b: c7 41 42 e8 b5 ff ff    movl $0xffffb5e8,0x42(%ecx)
8048462: ff eb                   ljmp *<internal disassembler error>
8048464: ...
```

Linear sweep does **not** follow CTIs, so it starts disassembling junk.

How does recursive traversal handle it?

Let's try IDA!

Junk code {ex2}
Thwarting linear sweep

```
jmp $+5
.byte 0x33, 0xe4, 0xc7, 0x41, 0x42
```

```
8048457: eb 05                    jmp 804845e <main+0xa>
8048459: 33 e4                    xor %esp,%esp
804845b: c7 41 42 e8 b5 ff ff     movl $0xffffb5e8,0x42(%ecx)
8048462: ff eb                    ljmp *<internal disassembler error>
8048464: ...
```

Linear sweep does **not** follow CTIs, so it starts disassembling junk.

How does recursive traversal handle it?

Let's try IDA!

How could you modify it to thwart IDA too?

Junk code {ex3}

Thwarting recursive traversal too

```
cmpb %al, %al
je $+5
.byte 0x33, 0xe4, 0xc7, 0x41, 0x42
```

```
8048459: 38 c0                  cmp %al,%al
804845b: 74 05                  je 8048462 <main+0xc>
804845d: 33 e4                  xor %esp,%esp
804845f: c7 41 42 e8 b3 ff ff   movl $0xffffb3e8,0x42(%ecx)
8048466: ff                     (bad)
8048467: ...
```

Using a conditional `jmp` forces recursive traversal to follow both paths, leading to confusion.

Branch functions {ex4}

Altering branches and/or calls to either:

- obscure the target
- mislead the disassembler

```
jmp 0x8048340c
```

```
804844d: 68 fc 83 04 08          push $0x804840c
8048452: c3                      ret
8048453: ...
```

Can be made worse, by inserting junk code right after the `ret` instruction.

Virtually an infinite number of variations can be implemented.

Branch functions {ex4}

Altering branches and/or calls to either:

- obscure the target
- mislead the disassembler

```
call 0x8048340c
```

```
804844b: eb 0a                  jmp 8048457 <main+0xf>
804844d: 68 fc 83 04 08         push $0x80483fc
8048452: 83 04 24 10            addl $0x10,(%esp)
8048456: c3                     ret
8048457: e8 f1 ff ff ff         call 804844d <main+0x5>
804845c: ...
```

Can be made worse, by inserting junk code right after the `ret` instruction.

Virtually an infinite number of variations can be implemented.

Packing {ex5}

- The code is hidden by 1+ layers of encryption/compression
- Decryption/decompression at run-time only

Countermeasures?

- Manual analysis of the packing routine
- Dynamic analysis