SNAKES ON A PAYLOAD

CORE SECURITY TECHNOLOGIES

CORE LABS RESEARCH & DEVELOPMENT

INTRODUCCIÓN

Fernando Russ

- 2000-2005: Core ST
- 2005-2008: Core Impact
- 2008-2011: CoreLabs

Pedro Varangot

2007-2011: CoreLabs

OUTLINE

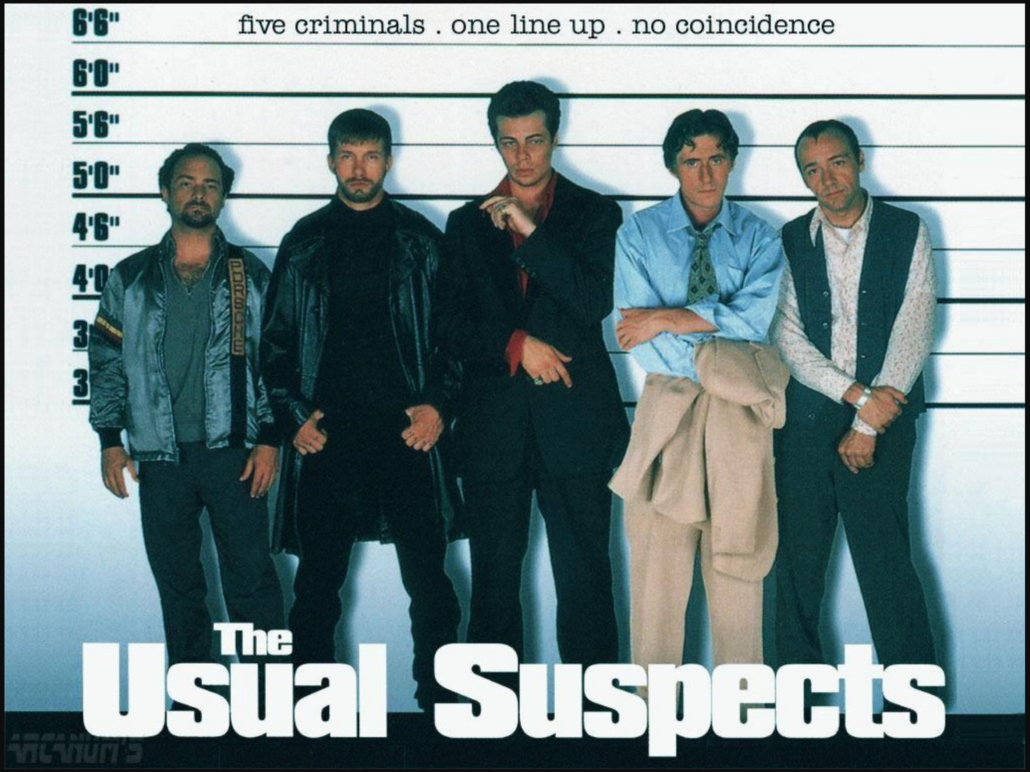
- Exploits y stage 2
- Formato de binario
- Compilando código en C de forma PIC
  - Demo
- Un stage 2 totalmente PIC que corre una VM de Python
  - Demo

POST-EXPLOTACIÓN

- No todo es calc.exe
- Un “stage 2” es el código encargado de “weaponizar” una vulnerabilidad explotada
- Hay varias dificultades propias
  - Shellcode vs. Proxy vs. VM
  - Multiplataforma y Multi-SO

DISEÑANDO UN STAGE2

- Tecnologías que ya hay disponibles
  - Proxycall
  - MOSDEF
  - Mosquito
  - Meterpreter
  - Netifera



PAYLOADS CON VMS, as seen on:

- Mosquito en 2005 (Lua)
- Mosquito en 2006 (Lisp)
- Dai Zovi en 2007 (Lua)
- Netifera en 2008 (Java)
- …
- ¡Nosotros! en 2011 (Python--)

Mosquito 2006: http://www.youtube.com/watch?v=8IAlBjY9nBk
Dai Zovi: http://www.usenix.org/event/woot07/tech/full_papers/daizovi/daizovi_html/
Netifera: http://blog.netifera.com/xcon2008-netifera-presentation/

DISEÑANDO UN STAGE2

- Un escenario de phone home:
  - Detección de proxy para poder salir

```
import os

for file in os.listdir("/etc"):
    try:
        for line in open(file).readlines():
            if "proxy" in line:
                possible_proxy_line(line)
    except:
        pass

def possible_proxy_line(line):
    ...
```

- ¿En assembler?
- ¿Cuantas syscalls son?

DISEÑANDO UN STAGE 2

- Si hay una VM mandamos bytecode

```python
import os

for file in os.listdir("/etc"):
    try:
        for line in open(file).readlines():
            if "proxy" in line:
                possible_proxy_line(line)
    except:
        pass

def possible_proxy_line(line):
    ...
```

```c
unsigned char find_proxy_bc[] = {
44,2,0,0,30,3,0,1,100,101,102,32,104
,111,108,97,40,41,58,0,12,0,0,11,116
,101,115,116,95,115,115,50,46,112,12
1,0,33,0,0,0,12,0,0,1,63,0,0,0,
34,0,0,0,16,0,0,72,44,9,0,0,30,3,0,1
,100,101,102,32,104,111,108,97,40,41
,58,0,12,1,0,11,...}
```

QUE HICIMOS NOSOTROS

- Payload que corre una Virtual Machine
- **Tinypy**, con algunos cambios mínimos
  - No son para que compile… son mejoras
- Podría haber sido Lua
  - No podíamos usar “Snakes On a Payload”
  - Los de Mosquito hacen un balance negativo

http://chargen.matasano.com/chargen/2008/6/2/reminiscences-about-lua-and-mosquito.html
http://www.tinypy.org/

EL AUTOR DE TINYPY

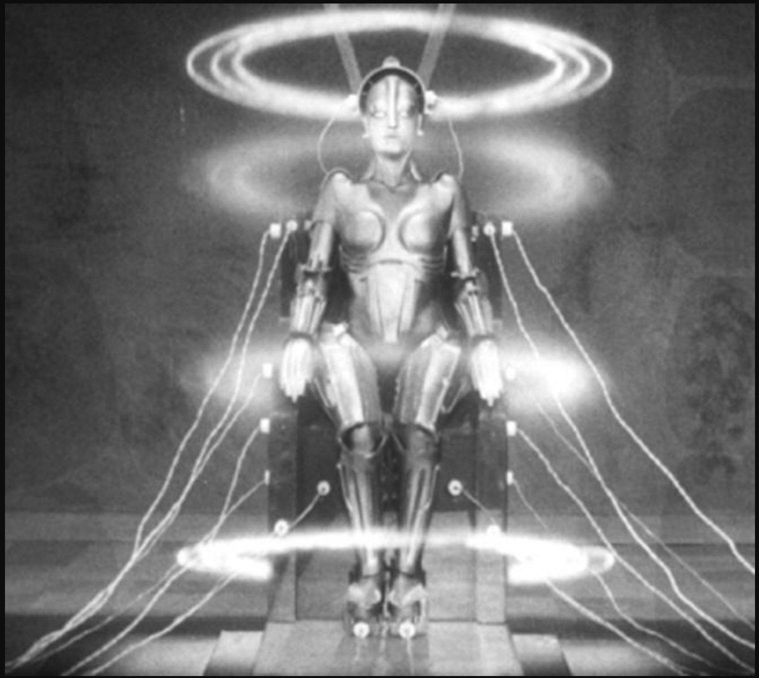
Y SU CABRA

Cuzco

http://www.philhassey.com/

“BATTERIES INCLUDED”

- Se compilan de un directorio de paquetes
- Quedan como módulos “built-in”
- Hay un “main” para poder hacer import en el orden adecuado



Scripts y herramientas para compilar

EN EL PRÓXIMO CAPÍTULO…

ELF WALK WITH ME

- Formatos de binario: ELF
  - Muchas secciones, arbitrarias
  - Relocations
    - Por plataforma: ABI

- .text
- .got
- .rela
- .dynsym
- .note.GNU-stack

"How to Write Shared Libraries", **Ulrich Drepper**: http://www.akkadia.org/drepper/dsohowto.pdf

I GOT YOU BABY...

```
; Attributes: bp-based frame

                public main
main            proc near

var_10          = qword ptr -10h
var_4           = dword ptr -4

                push    rbp
                mov     rbp, rsp
                mov     [rbp+var_4], edi
                mov     [rbp+var_10], rsi
                mov     rax, cs:variable_en_GOT_ptr
                mov     dword ptr [rax], 0DEADBEEFh
                mov     eax, 0
                leave
                retn
main            endp
```

gotpcrel.c (~/Documents) - GVIM1

File Edit Tools Syntax Buffers Window DrChip

```c
int variable_en_GOT = 9;

int main(int argc, char **argv)
{
  variable_en_GOT = 0xdeadbeef;

  return 0;
}
```

```
; ===========================================================================

; Segment type: Pure data
; Segment permissions: Read/Write
; Segment alignment 'qword' can not be represented in assembly
_got            segment para public 'DATA' use64
                assume cs:_got
                ;org 600FD8h
__gmon_start___ptr dq offset __gmon_start__
                                        ; DATA XREF: call_gmon_start+4↑r
variable_en_GOT_ptr dq offset variable_en_GOT ; DATA XREF: main+B↑r
_got            ends
```

```
.data:0000000000601018                 public variable_en_GOT
.data:0000000000601018 variable_en_GOT dd 9                    ; DATA XREF: .got:variable_en_GOT_ptr↑o
.data:0000000000601018 _data           ends
.data:0000000000601018
```

GCC VOODOO

- -fpic: Código que se puede mover
- Sacamos cosas de seguridad
- Sacamos cosas de excepciones
- No linkeamos contra nada
- Script de linker propio: rebase a 0

```
gcc –nostartfiles –nostdlib -fno-builtin -fno-asynchronous-unwind-tables -fPIC
gcc -Tscript.ld -Wl,-z,norelro,-z,now,-E,-N,--build-id=none -nostartfiles -
nostdlib -fno-builtin -fno-asynchronous-unwind-tables -fPIC
```

IN SOVIET RUSSIA: ELF RUNS YOU

- Linking “compile time”
  - Muchos .o -> un ejecutable o librería
    - R_X86_64_GOTPCREL
    - R_X86_64_PC32
- Linking “load time”
  - Muchas librerías -> ¿este slideshow?
    - R_X86_64_JMP_SLO
    - R_X86_64_64

“Gold”, **Ian Lance Taylor**, http://airs.com/ian/gold-slides.pdf
“System V Application Binary Interface”, http://www.x86-64.org/documentation/abi.pdf

ELF JUGGLING

- The Binutils
  - readelf
  - objcopy
  - libbfd

- Y bueno, el ld…
  - Primera versión: 1989
    - Está pensado para linkear a.out. ELF es un hack.
    - Todavía había Muro de Berlín.
    - La URSS dejaba Afganistán.
    - ¿$1 = ₳ 10.000? Dos años después, un peso iba a valer un dolar.
  - 13 pasadas en un linkeo común

BINUTILS

gotpcrel.c (~/Documents) - GVIM1

File Edit Tools Syntax Buffers Window DrChip

```c
int variable_en_GOT = 9;

int main(int argc, char **argv)
{
  variable_en_GOT = 0xdeadbeef;

  return 0;
}

```

```
peter@ubuntu:~/Documents$ readelf -r gotpcrel.o

Relocation section '.rela.text' at offset 0x568 contains 1 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
00000000000e  000800000009 R_X86_64_GOTPCREL 0000000000000000 variable_en_GOT - 4

ekoparty@2011:~/Documents$ readelf -r gotpcrel

Relocation section '.rela.dyn' at offset 0x390 contains 2 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000600fe0  000300000006 R_X86_64_GLOB_DAT 0000000000601018 variable_en_GOT + 0
```

BINUTILS

```
ekoparty@2011:~$ readelf -r /bin/ls

Relocation section '.rela.dyn' at offset 0x1448 contains 7 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000618fd8  000c00000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0
000000619540  006b00000005 R_X86_64_COPY     0000000000619540 __progname + 0
000000619560  006800000005 R_X86_64_COPY     0000000000619560 __progname_full + 0
000000619568  007100000005 R_X86_64_COPY     0000000000619568 optind + 0
000000619570  007200000005 R_X86_64_COPY     0000000000619570 optarg + 0
000000619578  006e00000005 R_X86_64_COPY     0000000000619578 stderr + 0
000000619580  006600000005 R_X86_64_COPY     0000000000619580 stdout + 0

Relocation section '.rela.plt' at offset 0x14f0 contains 99 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000619000  000100000007 R_X86_64_JUMP_SLO 0000000000000000 strcoll + 0
000000619008  000200000007 R_X86_64_JUMP_SLO 0000000000000000 mktime + 0
000000619010  000300000007 R_X86_64_JUMP_SLO 0000000000000000 memset + 0
000000619018  000400000007 R_X86_64_JUMP_SLO 0000000000000000 mbrtowc + 0
000000619020  000500000007 R_X86_64_JUMP_SLO 0000000000000000 getgrnam + 0
```

BINUTILS

```
ekoparty@2011:~$ readelf -s /lib/x86_64-linux-gnu/libpcre.so.3

Symbol table '.dynsym' contains 70 entries:
   Num:    Value          Size Type    Bind   Vis      Ndx Name
     ...
     4: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND strncmp@GLIBC_2.2.5 (2)
     5: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND malloc@GLIBC_2.2.5 (2)
     ...
     18: 0000000000000000     0 FUNC    GLOBAL DEFAULT  UND memcmp@GLIBC_2.2.5 (2)
     19: 000000000002a820   119 OBJECT  GLOBAL DEFAULT   13 _pcre_OP_lengths
     20: 0000000000025d40    54 FUNC    GLOBAL DEFAULT   11 pcre_refcount
     21: 000000000002a900    64 OBJECT  GLOBAL DEFAULT   13 _pcre_utf8_table4
     22: 0000000000026c70   644 FUNC    GLOBAL DEFAULT   11 pcre_study
     23: 0000000000025380    12 FUNC    GLOBAL DEFAULT   11 pcre_free_substring
     24: 0000000000027290     8 FUNC    GLOBAL DEFAULT   11 pcre_version
     25: 00000000000279b8     0 FUNC    GLOBAL DEFAULT   12 _fini
     26: 0000000000025870   519 FUNC    GLOBAL DEFAULT   11 _pcre_is_newline
     27: 0000000000025a80   553 FUNC    GLOBAL DEFAULT   11 _pcre_was_newline
     28: 000000000023b078     8 OBJECT  GLOBAL DEFAULT   23 pcre_free
```

Ahora sí:

SNAKES ON A PAYLOAD

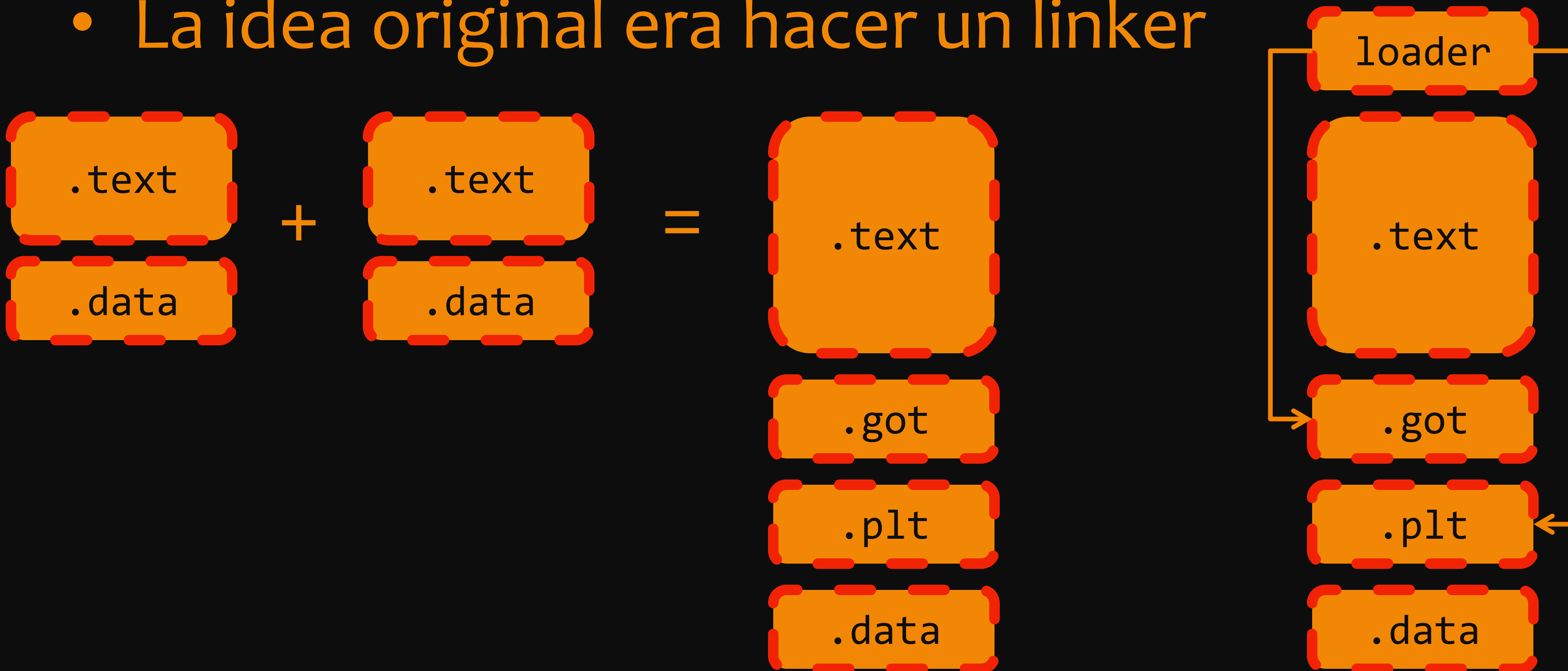
"ESTO NO PUEDE SER TAN DIFÍCIL"

Fernando F. Russ (2011)

“ESTO NO PUEDE SER TAN DIFÍCIL”

Fernando F. Russ (2011)

- La idea original era hacer un linker

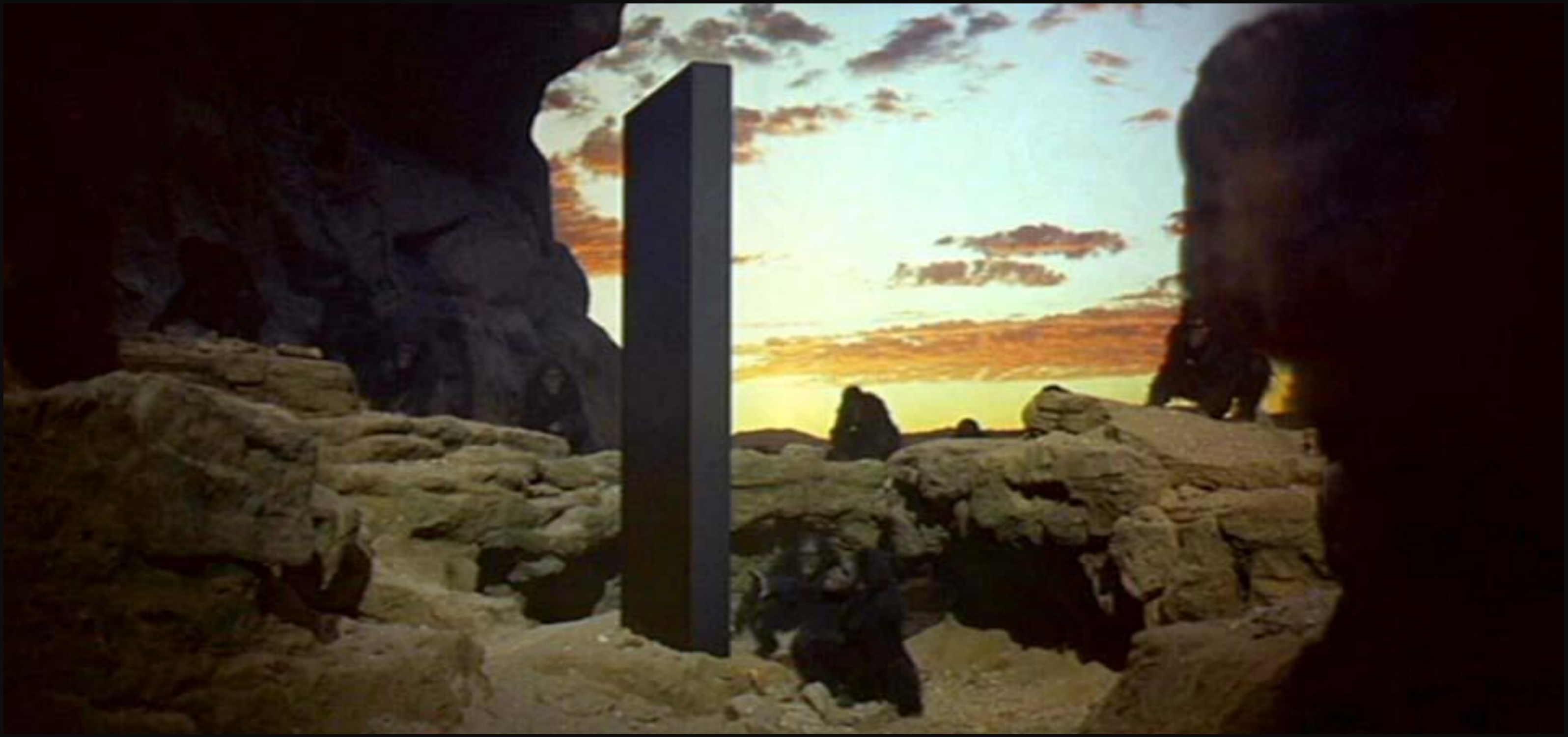


- Un loader “rebaseaba” la GOT y la PLT en runtime

“EL LD ES UN MUNDO DE DOLOR”

Pedro O. Varangot (2011)

- Adquirimos profunda admiración por la gente que hace binutils…



“ESTÁS HACIENDO TODO MAL”

Sourceware (2007)

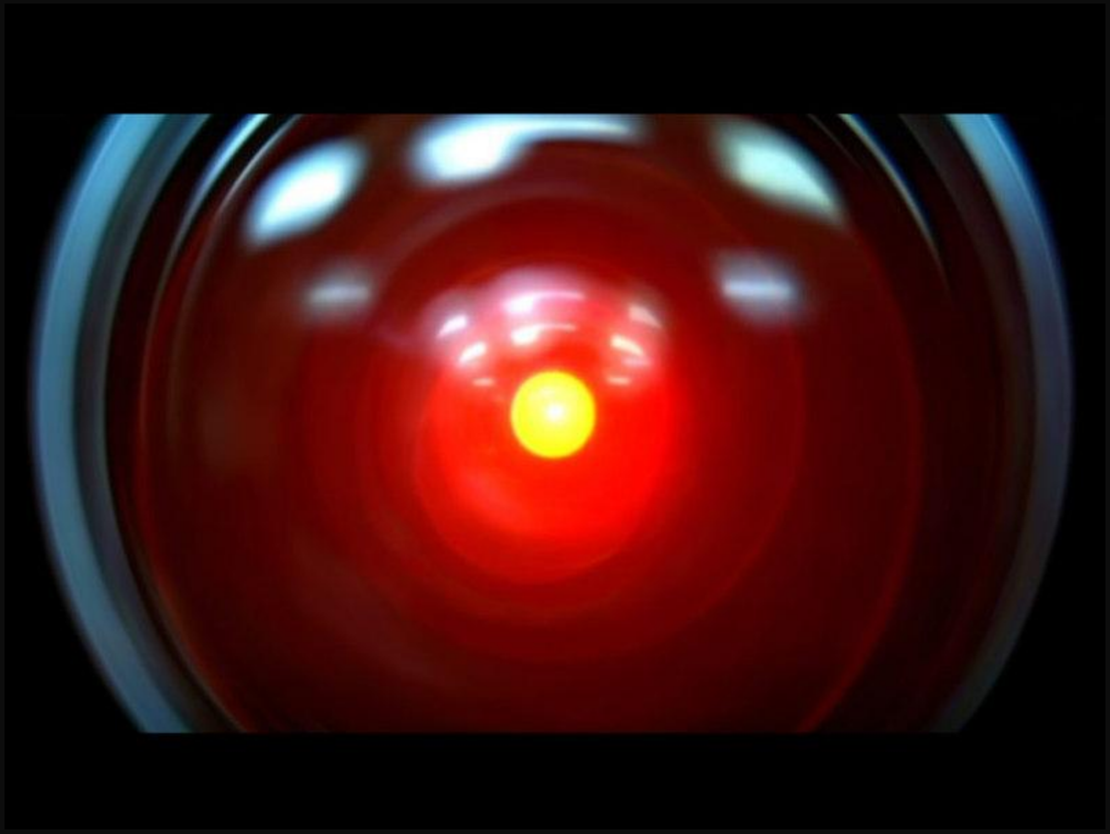
- Las relocations en .data.rel no son tan fáciles de calcular cuando hay structs
  - Si se rebasea toda la sección, se rompen campos que ya están bien calculados
- Al día de hoy no sabemos como el ld calcula esto bien
- No nos importa por ahora, porque lo hace

http://sourceware.org/ml/binutils/2007-03/msg00422.html

CONFIEMOS EN EL LINKER

"total, que puede salir mal…"

- -shared y -fpic ya arman la GOT y la PLT bien



PRIMER PASO .C A CÓDIGO “PIC”

- .so que nos da el gcc + script.ld:

```
Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .prelude      0000004f  0000000000000000  0000000000000000  000000e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         0000003c  0000000000000050  0000000000000050  00000138  2**2
                  CONTENTS, ALLOC, LOAD, CODE
  2 .got          00000008  0000000000000090  0000000000000090  00000178  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  3 .got.plt      00000018  0000000000000098  0000000000000098  00000180  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  4 .data         00000004  00000000000000b0  00000000000000b0  00000198  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .rebase       00000107  00000000000000b4  00000000000000b4  0000019c  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  6 .endoftheworld 00000008  00000000000001c0  00000000000001c0  000002a8  2**3
                  CONTENTS, ALLOC, LOAD, CODE
```

PRIMER PASO .C A CÓDIGO “PIC”

```
Sections:
Idx Name          Size      VMA               LMA               File off  Algn
  0 .prelude      0000004f  0000000000000000  0000000000000000  000000e8  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .text         0000003c  0000000000000050  0000000000000050  00000138  2**2
                  CONTENTS, ALLOC, LOAD, CODE
  2 .got          00000008  0000000000000090  0000000000000090  00000178  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  3 .got.plt      00000018  0000000000000098  0000000000000098  00000180  2**3
                  CONTENTS, ALLOC, LOAD, DATA
  4 .data         00000004  00000000000000b0  00000000000000b0  00000198  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  5 .rebase       00000107  00000000000000b4  00000000000000b4  0000019c  2**0
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  6 .endoftheworld 00000008  00000000000001c0  00000000000001c0  000002a8  2**3
                  CONTENTS, ALLOC, LOAD, CODE
```

- El loader en .rebase lee del final de binario.
- Esos datos los emitimos ahí nosotros
  - Procesamos el .so con python llamando a readelf y objcopy

IT'S ALIVE !!!

- s2l.py emite un ".bin"
  - El loader tiene que ser 100% PIC. "Chicken and Egg"
  - ¡Código 100% **assembler free**!
    - Lo que no quiere decir que sea portable…

Loader SOAP
69 LoC

.text
.got
.data
...

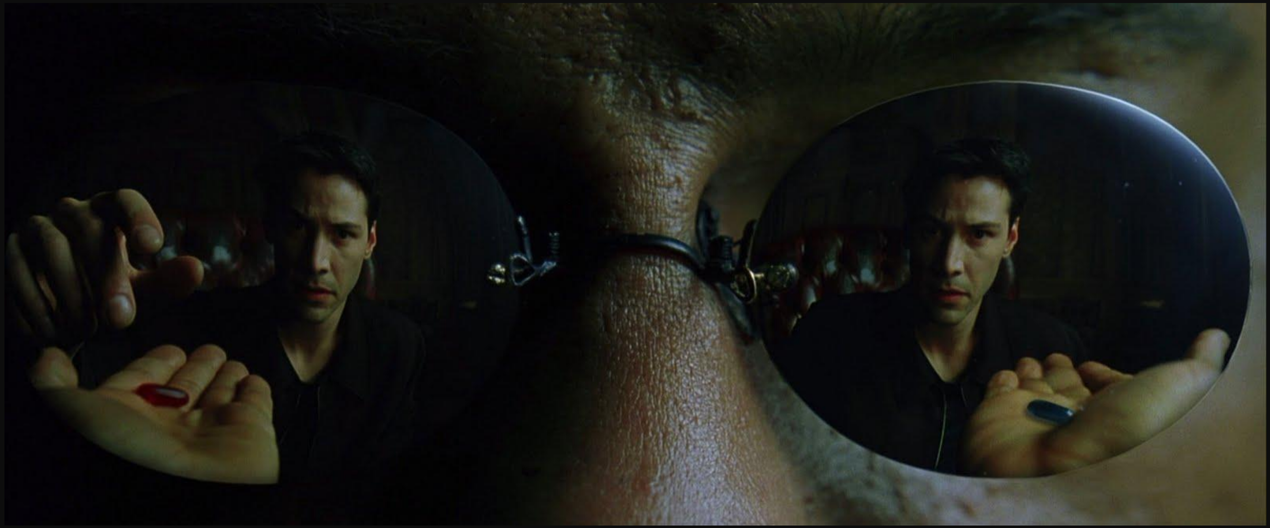
Relocations
SOAP

De ejecutar código PIC con nuestro framework

MINI DEMO

SEGUNDO PASO, LIBRERÍAS DEL SO

- Hay dos opciones para salir



  - Hacer la libc de nuevo.
  - Reutilizar el loader del sistema operativo.

http://www.nologin.org/Downloads/Papers/remote-library-injection.pdf
http://www.hick.org/code/skape/papers/win32-shellcode.pdf

SEGUNDO PASO, LIBRERÍAS DEL SO

- Que hacemos con las libs y las relocations:

```c
#define RELENTRY_TYPE_END 0
#define RELENTRY_TYPE_REBASE 1
#define RELENTRY_TYPE_JMPSLOT 2
#define RELENTRY_TYPE_LOADLIB 3

#define PACKED __attribute__ ((__packed__))

typedef union PACKED {
    struct PACKED {
      relentry_type type;
      unsigned long offset;
    } rebase_reloc;
    struct PACKED {
      relentry_type type;
      unsigned long offset;
      char symbol_name[128];
    } jmpslot_reloc;
    struct PACKED {
      relentry_type type;
    } end;
} reloc_entry;
```

- Hay un tipo nuevo de “relocation”
- Puede cargar una lib
- Puede resolver un símbolo

SEGUNDO PASO, LIBRERÍAS DEL SO

- El s2l emite los nombres de las funciones
- El loader hace dlopen y dlsym

```
case RELENTRY_TYPE_JMPSLOT:
  *relocation = (size_t)dlsym(handle, relocs[idx].jmpslot_reloc.symbol_name);
  break;

case RELENTRY_TYPE_LOADLIB:
  handle = dlopen(relocs[idx].jmpslot_reloc.symbol_name, 2);
  break;
```

SEGUNDO PASO, LIBRERÍAS DEL SO

- El s2l emite los nombres de las funciones
- El loader hace dlopen y dlsym

```
jump_slots = [relocation for relocation in elf.relocations if relocation.NAME == "R_X86_64_JUMP_SLO"
symtable = {}
for libname, symbols in symbol_table.items():
    symtable[libname] = [reloc for reloc in jump_slots if reloc.symbol.name in symbols]

for libname, relocations in symtable.items():
    if len(relocations) > 0: # we have relocations to emit for this lib...
        # load the library
        Relocation.emitEntry( _f, type = Relocation.RelocType.LOADLIB,
                                  name = libname)
        for relocs in relocations:
            Relocation.emitEntry( _f, type = Relocation.RelocType.JMPSLOT,
                                      offset = relocs.offset,
                                      name = relocs.symbol.name)
```

ALGUNOS PROBLEMAS

- Relocations raras... _COPY
- Threads / TLS
- Signals
- @@GLIBC...

Corriendo Python

DEMO CON LA VM

Code.google.....

RELIS

CRÉDITOS / AGRADECIMIENTOS

- Andrés Luksenberg
- Leandro Skladnik
- Adrían Manrique
- Anibal Sacco
- Oren Isacson
- Iván Arce
- Todos los tipos que citamos
- "Google"
- ¡Ekoparty!

Los que hicieron esto antes que nosotros. A los que no lo publicaron, también les agradecemos.

THE END?

(Ningún ELF fue maltratado durante la producción de esta charla)