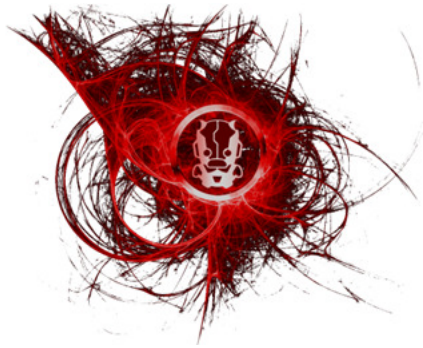
IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

In the lands of corrupted elves:
Breaking ELF software with Melkor fuzzer

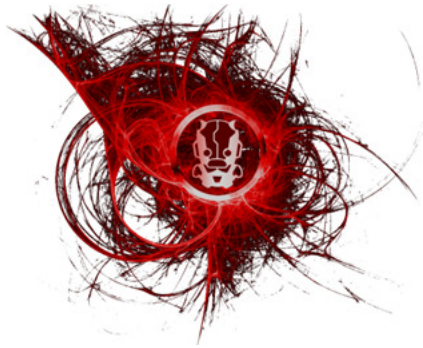
Alejandro Hernández
IOActive



About me

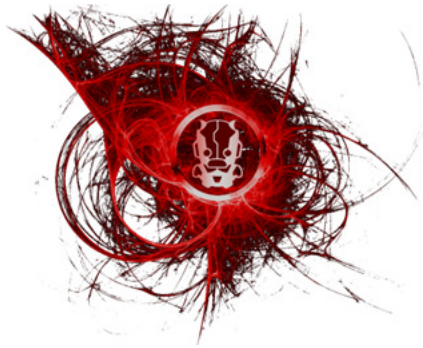
- Senior Security Consultant [IOActive.com]
- ELF, C programming & fuzzing enthusiast
- Passionate about security. ~11 years now.
- From Chiapas, Mexico

- http://www.brainoverflow.org
- @nitr0usmx



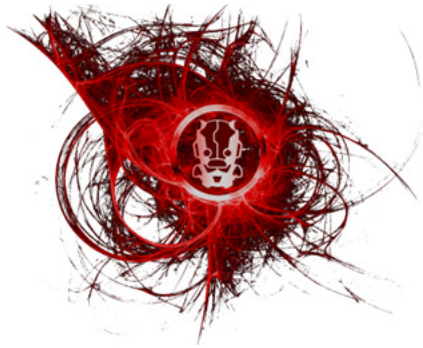
Agenda

- The ELF file format
- ELF parsing
  - Who's is parsing?
  - Security risks in ELF parsing
  - Discovered vulnerabilities in the past
  - ELF parsing (mistakes) nowadays
- ELF Fuzzing
  - Smart vs dumb
  - Code / branch coverage
  - ELF metadata dependencies
- Cont.



Agenda (Cont.)

- Melkor – an ELF file format fuzzer
  - Who's Melkor
  - Design & Implementation
  - Fuzzing rules
  - ELF metadata dependencies
  - Generators and test data
  - Usage
  - Logging
  - Download
- ~~Breaking~~ Fuzzing ELF software
  - DEMOS
- Conclusions

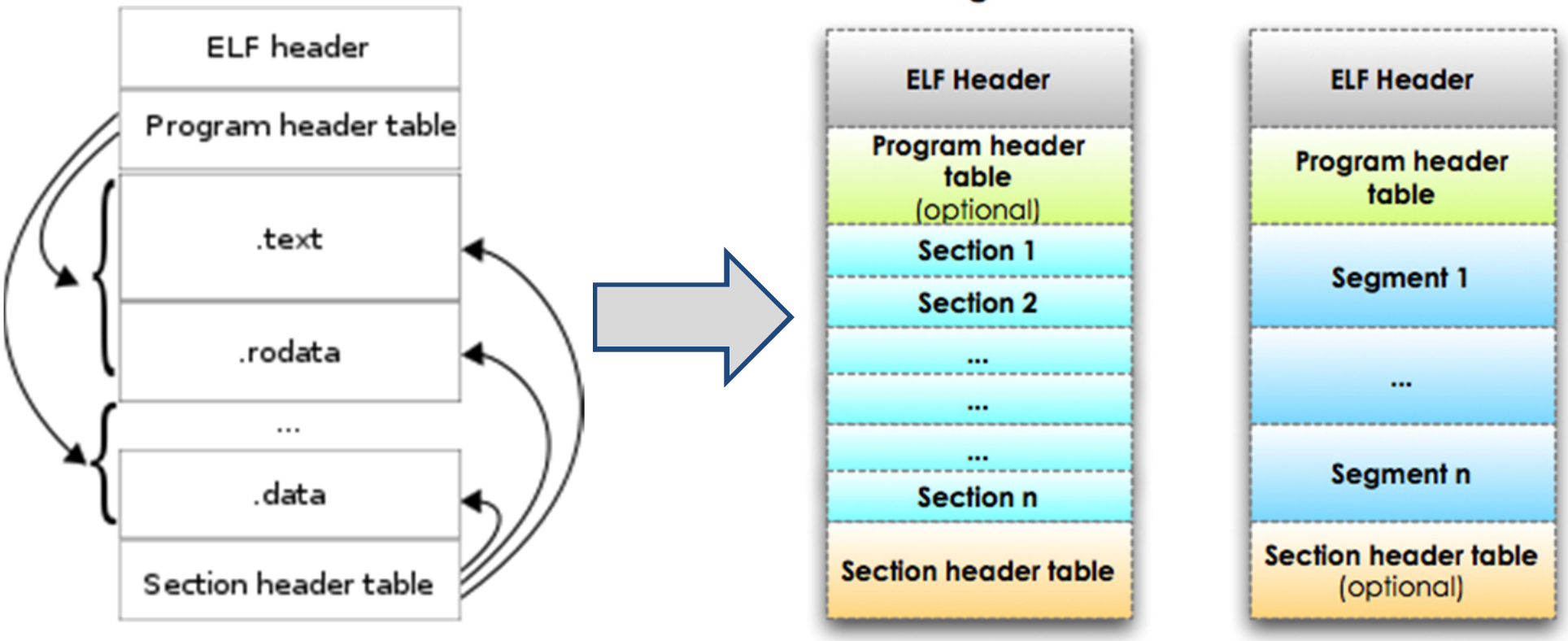


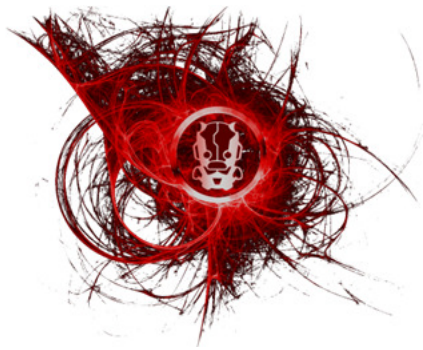
The ELF file format

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- Executable and Linkable Format
- In 1999 was chosen as the standard binary file format for Unix and Unix-like systems on x86
- Adopted by many OS on many different platforms
- Executables, relocatable objects (.o), shared libraries (.so) and core dumps.

The ELF file format





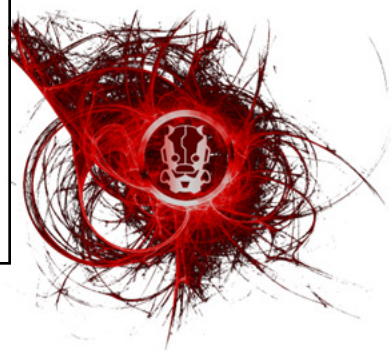
The ELF file format

- **Specification(s)** [10]

Tool Interface Standard (TIS)
Executable and Linking Format (ELF)
Specification
Version 1.2

Contents

Contents

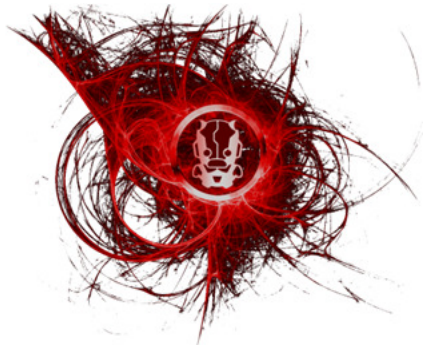


The ELF file format

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Data types** (/usr/include/elf.h)

```
/* Type for a 16-bit quantity.  */
typedef uint16_t Elf32_Half;
typedef uint16_t Elf64_Half;

/* Types for signed and unsigned 32-bit quantities.  */
typedef uint32_t Elf32_Word;
typedef int32_t  Elf32_Sword;
typedef uint32_t Elf64_Word;
typedef int32_t  Elf64_Sword;

/* Types for signed and unsigned 64-bit quantities.  */
typedef uint64_t Elf32_Xword;
typedef int64_t  Elf32_Sxword;
typedef uint64_t Elf64_Xword;
typedef int64_t  Elf64_Sxword;

/* Type of addresses.  */
typedef uint32_t Elf32_Addr;
typedef uint64_t Elf64_Addr;

/* Type of file offsets.  */
typedef uint32_t Elf32_Off;
typedef uint64_t Elf64_Off;

/* Type for section indices, which are 16-bit quantities.  */
typedef uint16_t Elf32_Section;
typedef uint16_t Elf64_Section;

/* Type for version symbol information.  */
typedef Elf32_Half Elf32_Versym;
typedef Elf64_Half Elf64_Versym;
```



The ELF file format

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Data structures** (/usr/include/elf.h)

```
typedef struct
{
  unsigned char e_ident[EI_NIDENT];
  Elf64_Half    e_type;
  Elf64_Half    e_machine;
  Elf64_Word    e_version;
  Elf64_Addr    e_entry;
  Elf64_Off     e_phoff;
  Elf64_Off     e_shoff;
  Elf64_Word    e_flags;
  Elf64_Half    e_ehsize;
  Elf64_Half    e_phentsize;
  Elf64_Half    e_phnum;
  Elf64_Half    e_shentsize;
  Elf64_Half    e_shnum;
  Elf64_Half    e_shstrndx;
} Elf64_Ehdr;
```

```
typedef struct
{
  Elf64_Word    sh_name;
  Elf64_Word    sh_type;
  Elf64_Xword   sh_flags;
  Elf64_Addr    sh_addr;
  Elf64_Off     sh_offset;
  Elf64_Xword   sh_size;
  Elf64_Word    sh_link;
  Elf64_Word    sh_info;
  Elf64_Xword   sh_addralign;
  Elf64_Xword   sh_entsize;
} Elf64_Shdr;
```

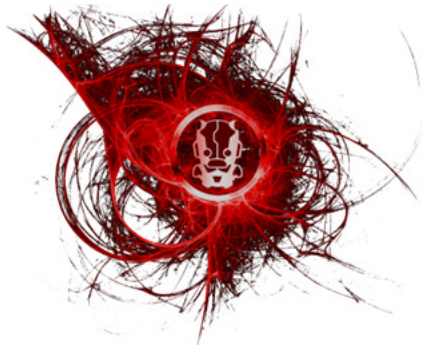
```
typedef struct
{
  Elf64_Word    p_type;
  Elf64_Word    p_flags;
  Elf64_Off     p_offset;
  Elf64_Addr    p_vaddr;
  Elf64_Addr    p_paddr;
  Elf64_Xword   p_filesz;
  Elf64_Xword   p_memsz;
  Elf64_Xword   p_align;
} Elf64_Phdr;
```

```
typedef struct
{
  Elf64_Word    st_name;
  unsigned char st_info;
  unsigned char st_other;
  Elf64_Section st_shndx;
  Elf64_Addr    st_value;
  Elf64_Xword   st_size;
} Elf64_Sym;
```

```
typedef struct
{
  Elf64_Sxword  d_tag;
  union
    {
      Elf64_Xword d_val;
      Elf64_Addr d_ptr;
    } d_un;
} Elf64_Dyn;
```

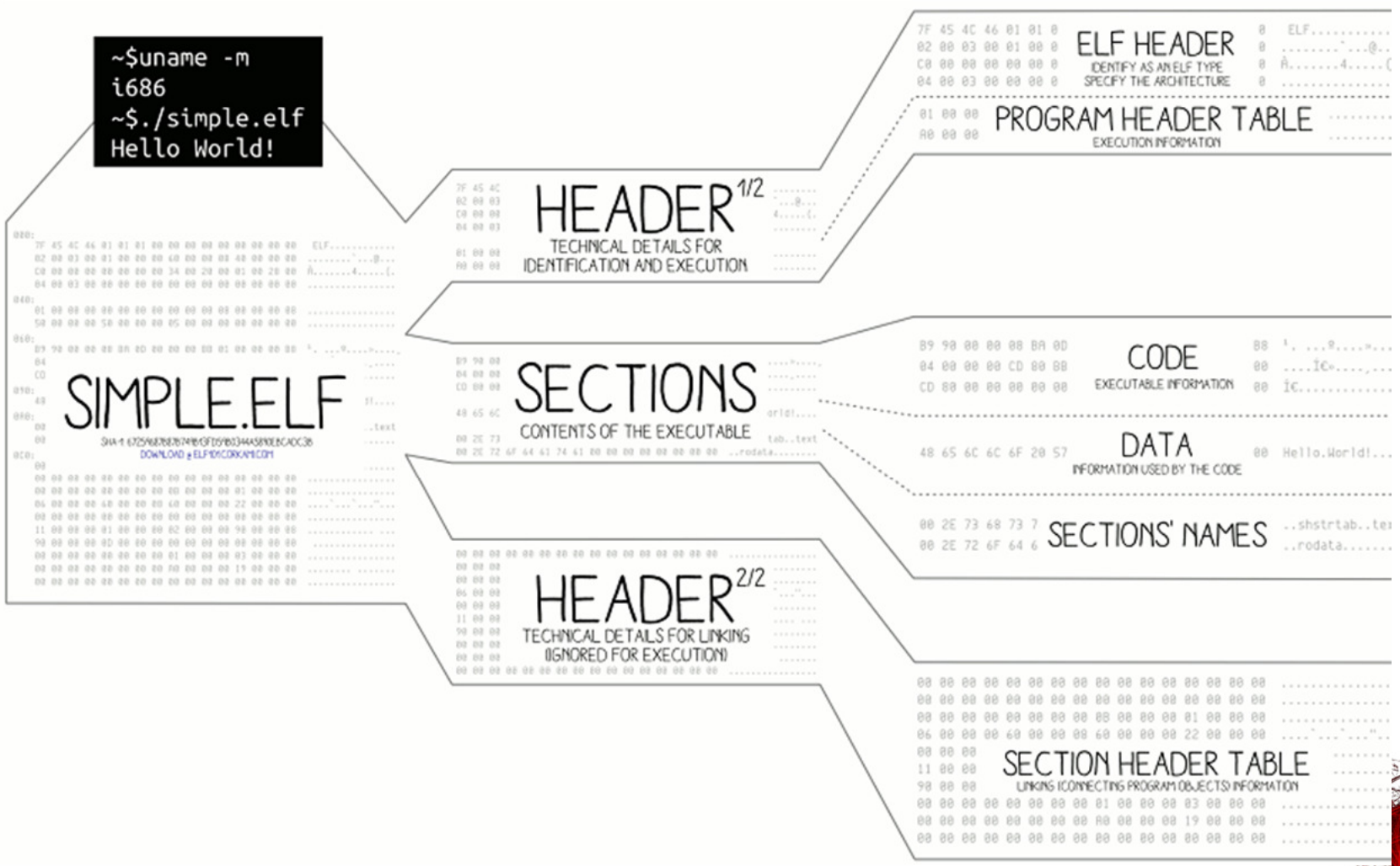
```
typedef struct
{
  Elf64_Addr    r_offset;
  Elf64_Xword   r_info;
  Elf64_Sxword  r_addend;
} Elf64_Rela;
```

```
typedef struct
{
  Elf64_Word n_namesz;
  Elf64_Word n_descsz;
  Elf64_Word n_type;
} Elf64_Nhdr;
```



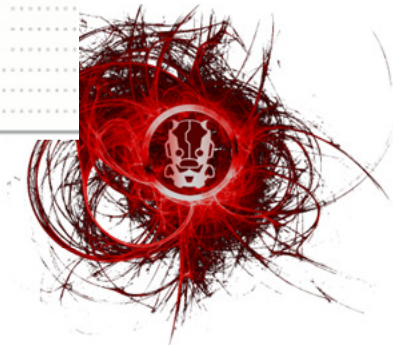
The ELF file format

- **Relationships between metadata**



ELF 101 a Linux executable walk-through

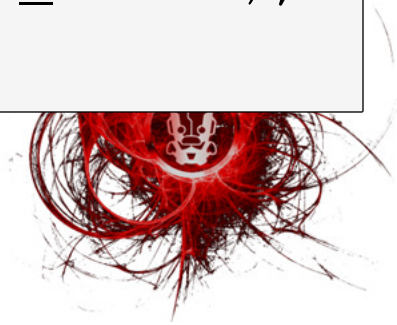
ANGE ALBERTINI
CORKAMI.COM



The ELF file format

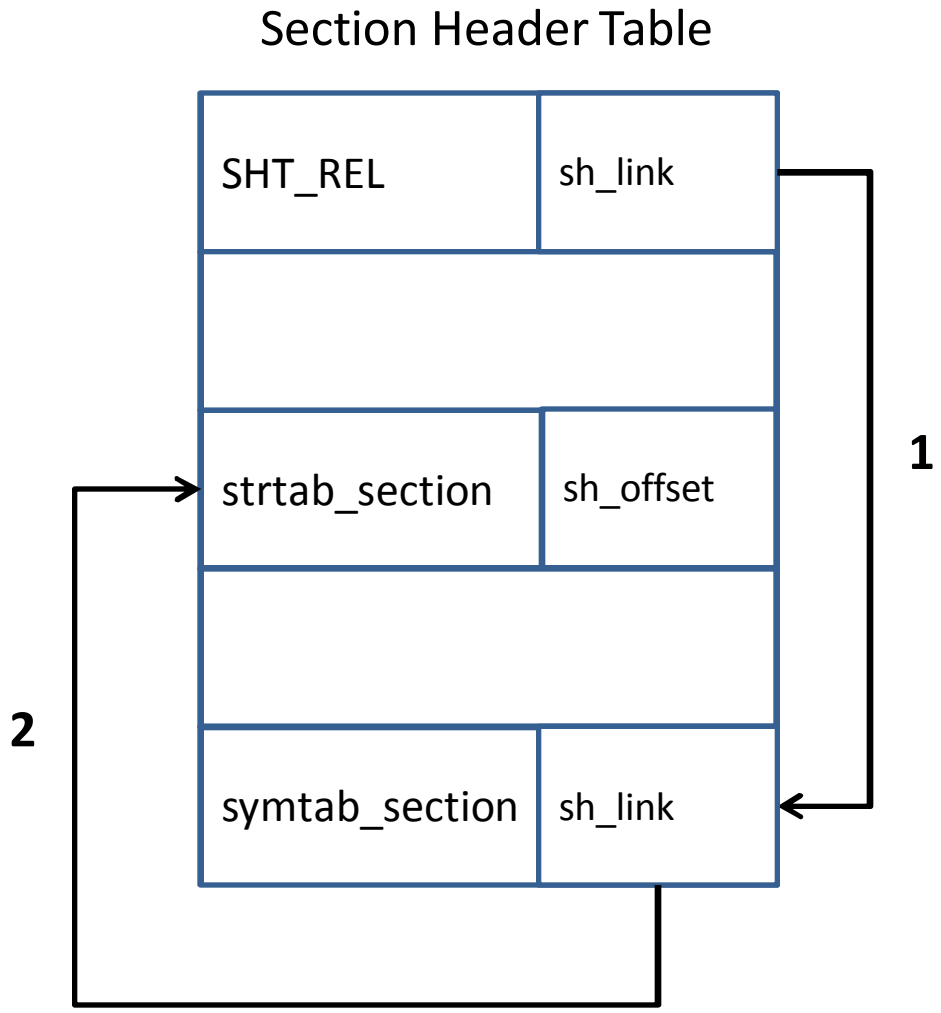
- **Relationships between metadata**
- Example:

```
for(k = 0; k < hdr.e_shnum; k++, shdr++){
    if(shdr->sh_type != SHT_REL)
        continue;

    symtab_section = shdr_table[shdr->sh_link];
    strtab_section = *(Elf64_Shdr *) (mem + hdr.e_shoff +
(symtab_section.sh_link * sizeof(Elf64_Shdr)));
    symstrtab_offset  = strtab_section.sh_offset;

    rela = (Elf64_Rela *) (mem + shdr->sh_offset);
    sym  = (Elf32_Sym  *) (mem + symtab_section.sh_offset);
...
```



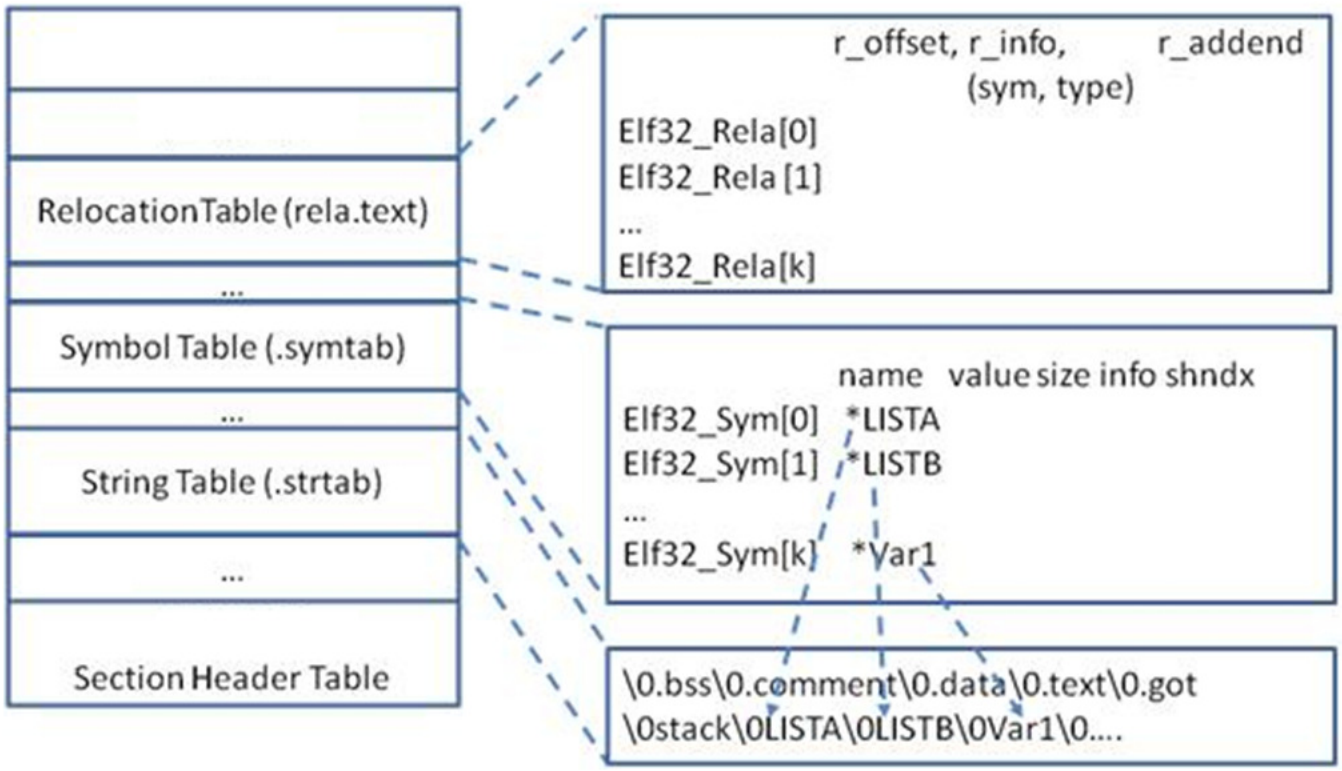
The ELF file format

- **Relationships between metadata**
- Example:



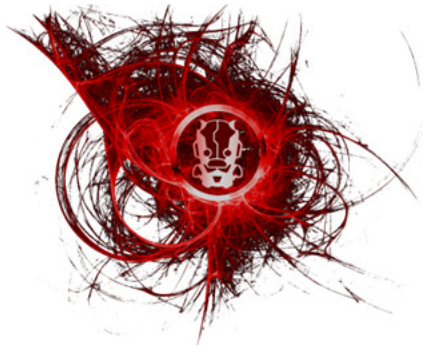
The ELF file format

- **Relationships between metadata**
- Example:



ELF parsing

- **Who's is parsing?**
  - OS kernels
    - Thoroughly audited over the years
  - Debuggers
    - gdb
    - IDA Pro
    - Etc.
  - Reverse Engineering frameworks
    - ERESI
    - radare2
    - Etc.
  - OS utilities
    - binutils
    - #apt-cache search ELF



ELF parsing

- **Who's is parsing?**
  - Malware
  - Antivirus engines?
    - Sophail by Tavis Ormandy [6]

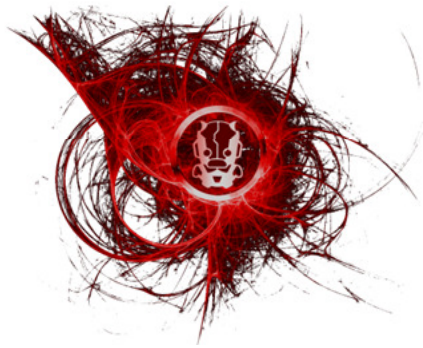
Many of the decoders are simply bizarre nonsense. For example, the ELF decoder specifically excludes Siemens TriCore executables (used in industrial microcontrollers).

ELF defines dozens of esoteric architectures like the Fujitsu FR20 or the Matsushita MN10200, all of which are perfectly valid.

```
// This makes no sense.
if (ElfMachine != EM_TRICORE){
  // Matches ET_NONE, ET_REL, ET_EXEC and ET_DYN
  if ((ElfType - 1) <= 2)
      return CLASS_ELF_STORAGE.Name;
  return NULL;
}
```

Figure 13. Pseudocode for a bizarre architecture exclusion in the ELF decoder.

The most likely explanation is that a customer complained that one of their embedded executables for a Siemens/Infineon TriCore device was triggering a CRC32 collision with one of the static file signatures Sophos distribute. Rather than fix the problem properly, Sophos simply excluded the entire architecture, no longer recognising them as executable.



ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Who's is parsing?**
  - Google dork: "+ELF (parser|parsing)"

- Linux
- Solaris
- IRIX
- FreeBSD
- NetBSD
- OpenBSD
- DragonFly BSD
- Syllable
- HP-UX (except for
- QNX Neutrino
- MINIX[6]

- `readelf` is a Unix binary
- `elfutils` provides alterr
- `elfdump` is a command fc
- `objdump` provides a wide data.
- The Unix `file` utility can

ELF has also seen some adoption in non-Unix operating systems, such as:

- OpenVMS, in its Itanium version
- BeOS Revision 4 and later for x86 based computers (where it replaced
- Haiku, the open source reimplementation of BeOS.
- RISC OS[7]

Other operating systems running on PowerPC using ELF:

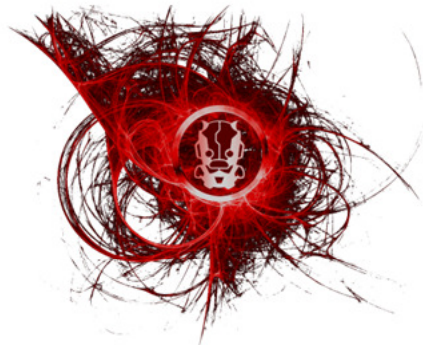
- AmigaOS 4, the ELF executable has replaced the previ
- MorphOS
- AROS

Some game consoles also use ELF:

- PlayStation Portable,[8] PlayStation 2, PlayStation 3
- GP2X.
- Dreamcast
- GameCube, Wii

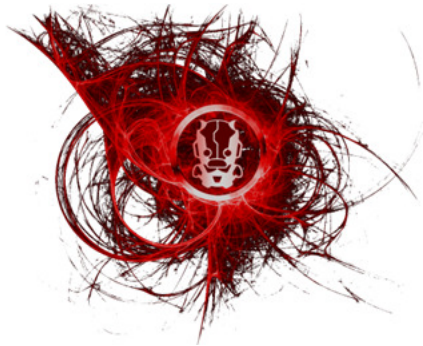
Some operating systems for mobile phones and mobile devices use ELF:

- Symbian OS v9 uses E32Image[9] format that is based on the ELF file format;
- Sony Ericsson, for example, the W800i, W610, W300, etc.
- Siemens, the SGOLD and SGOLD2 platforms: from Siemens C65 to S75 and BenQ-Siemens E71/EL71);
- Motorola, for example, the E398, SLVR L7, v360, v3i (and all phone LTE2 which has the patch applied)
- Bada, for example, the Samsung Wave S8500.
- Nokia phones or tablets running the Maemo or the Meego OS, for example, the Nokia N900.
- Android uses ELF .so libraries for the Java Native Interface.



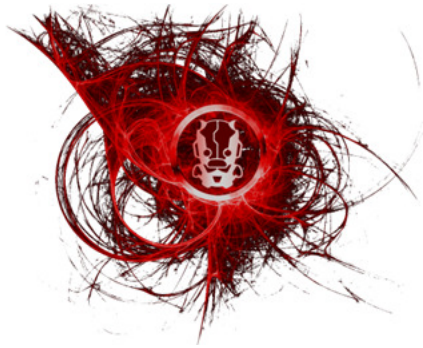
ELF parsing

- **Security risks in ELF parsing**
  - Memory corruption / Buffer overflows
    - Out of bounds array indexes or offsets
    - Loops copying data more times than expected
  - Invalid memory dereferences
    - Out of bounds array indexes or offsets
    - Crashes / DoS
  - Arithmetic / Integer wrap-arounds
    - Calculations with user-controlled data
      - nElements * elementSize
      - nElements * sizeof()
      - totalSize / elementSize
      - arrayIndex * sizeof()



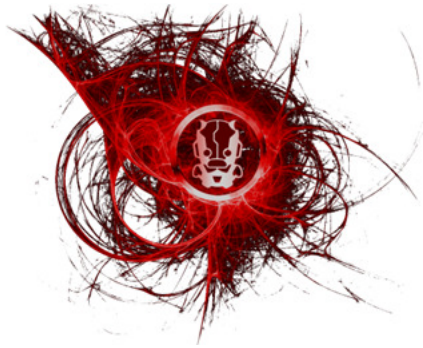
ELF parsing

- **Security risks in ELF parsing**
  - Memory corruption / Buffer overflows
    - Might lead to code execution
    - Undefined behaviors
  - Crashes / DoS
    - In the debugger / reversing tool
      - Anti-reversing technique
      - Binaries harder to debug
    - Protection against malware infections
      - Malware has parsers too
    - OS kernel panic()'s



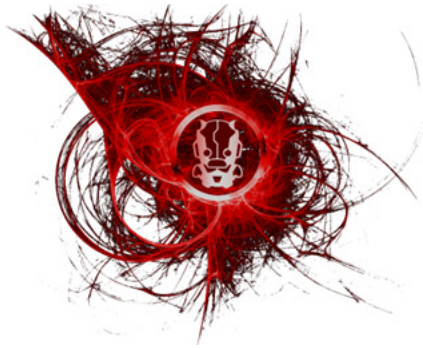
ELF parsing

- **Security risks in ELF parsing**
  - Most data types are unsigned ints. Two of them are signed ints (/usr/include/elf.h):
    - typedef int32_t Elf32_Sword;
    - typedef int32_t Elf64_Sword;
    - typedef int64_t Elf32_Sxword;
    - typedef int64_t Elf64_Sxword;
      - r_addend (Relocations)
      - d_tag (Dynamic information)
  - Harder to trigger integer overflows
  - However, when assigning values to local signed variables, signedness bugs might exist [1]



ELF parsing

- **Discovered vulnerabilities in the past**
  - ELF, unlike PE (Portable Executable), has been less audited
  - Mostly found doing manual testing
    - Code review + Binary modification
  - Google dork: “site:securityfocus.com +ELF”



ELF parsing

- Discovered vulnerabilities in the past

info | discussion | exploit | solution | references

HT Editor ELF Parser Unspecified Remote Heap Overflow Vulnerability

HT Editor is affected by an unspecified heap overflow vulnerability.

Specific details about this issue are not currently available. It is known that this vulnerability affects the ELF parser.

A successful attack may result in arbitrary code execution and allow the attacker to gain unauthorized access to the vulnerable computer.

info | discussion | exploit | solution | references

Linux Kernel ELF File Entry Point Denial of Service Vulnerability

Linux kernel is prone to a denial-of-service vulnerability when processing a malformed ELF file.

info | discussion | exploit | solution | references

FreeBSD Malformed ELF Image Denial of Service Vulnerability

A vulnerability exists in versions 3.x, and 4.x and 5.x prior to August 15, 2000, of FreeBSD. A failure of the ELF image activator to perform sufficient sanity checks on the ELF image header could result in local users being able to perform a denial of service attack against the machine.

ELF parsing

- Discovered vulnerabilities in the past

info | discussion | exploit | solution | references

GNU glibc 'ld.so' ELF Header Parsing Remote Integer Overflow Vulnerability

Bugtraq ID: 40063

Class: Boundary Condition Error

info | discussion | exploit | solution | references

QNX RTOS Malformed ELF Binary File Local Denial Of Service Vulnerability

QNX RTOS is prone to a local denial-of-service vulnerability.

Attackers can exploit this issue to cause a kernel panic, denying service to legitimate users.

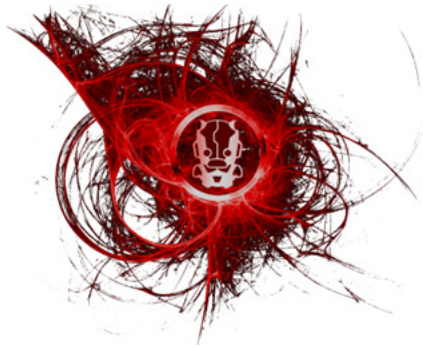
info | discussion | exploit | solution | references

Xen CVE-2013-2195 Pointer Dereference Privilege Escalation Vulnerability

Xen is prone to a privilege-escalation vulnerability.

An attacker with access to a guest operating system can exploit this issue to gain elevated privileges on

Note: This issue was previously discussed in BID 60422 (Xen 'ELF' Parser Multiple Security Vulnerabilities),



ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- Discovered vulnerabilities in the past

info | discussion | exploit | solution | references

Linux Kernel BINFMT_ELF Loader Local Privilege Escalation Vulnerabilities

Multiple vulnerabilities have been identified in the Linux ELF binary loader. These issues can allow local attackers to gain elevated privileges. The source of these issues resides in the 'load_elf_binary' function of the 'binfmt_elf.c' file.

The first issue results from an improper check performed on the return value of the 'kernel_read()' function. An attacker may gain control over execution flow of a setuid binary by modifying the memory layout of a binary.

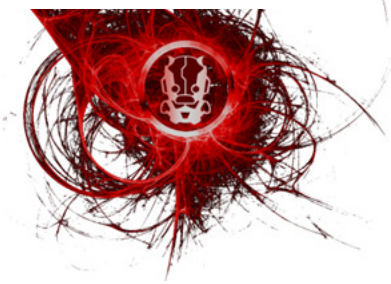
The second issue results from improper error-handling when the 'mmap()' function fails.

The third vulnerability results from a bad return value when the program interpreter (linker) is mapped into memory. It is reported that this issue occurs only in the 2.4.x versions of the Linux kernel.

The fourth issue presents itself because a user can execute a binary with a malformed interpreter name string. This issue can lead to a system crash.

The final issue resides in the 'execve()' code. This issue may allow an attacker to disclose sensitive data that can potentially be used to gain elevated privileges.

These issues are currently undergoing further analysis. This BID will be updated and divided into separate BIDS in the future.



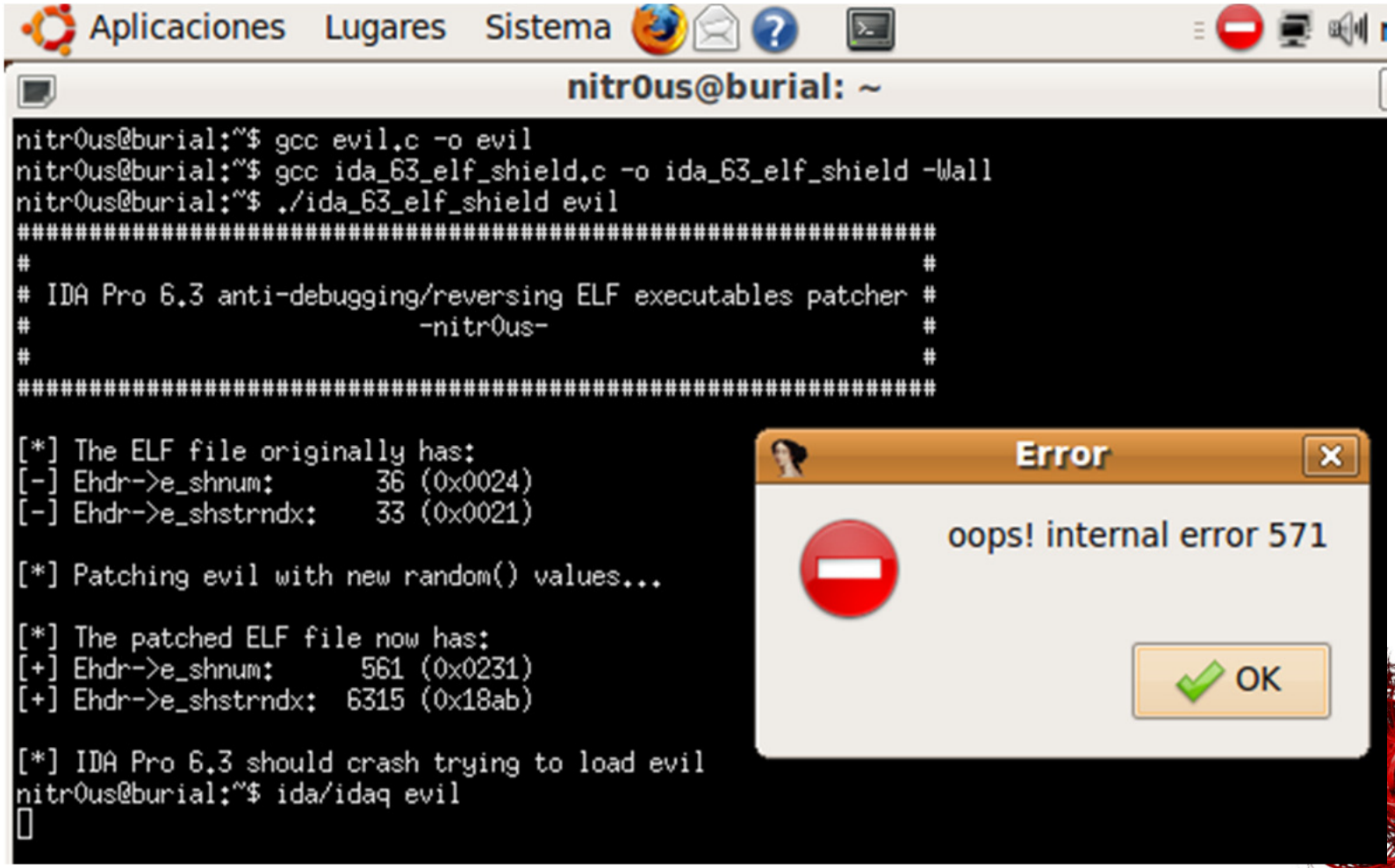
ELF parsing

- **Discovered vulnerabilities in the past**
  - Invalid pointer dereference in **gdb** (reported but **still unpatched**) used as an anti-debugging technique [4]

```
[-] '.debug_line' section was not found in the Section Header Table !
[*] Adding the '.debug_line' with the payload inside

[+] Patching the size of ".shstrtab" section to 0x0c88
[+] Patching the size of ".strtab" section to 0x0c88 as well (gdb also check this size)

[*] The insertion point will be at: 0x15ec

[-] The Program Header Table is before the insertion point.

[+] Injecting the '.debug_line' Elf32_Shdr struct at the end of the Section Header Table...
[+] Injecting the '.debug_info' Elf32_Shdr struct at the end of the Section Header Table...
[+] Injecting the '.debug_str' Elf32_Shdr struct at the end of the Section Header Table...
[+] Injecting the '.debug_abbrev' Elf32_Shdr struct at the end of the Section Header Table...

[+] Injecting the malformed line header structure (payload) into the new created '.debug_line' section...
[+] Injecting the content of '.debug_info', '.debug_str' and '.debug_abbrev' sections...
[+] The '.debug_line' section was added successfully with the payload inside

[*] "./foo" is now completely patched
[*] gdb (GNU debugger) <= 7.5.1 should crash trying to load "./foo"
nitr0us@chatsubo:~$ ./foo
foo
nitr0us@chatsubo:~$ gdb -q ./foo
Reading symbols from /home/nitr0us/foo...Segmentation fault
nitr0us@chatsubo:~$
```

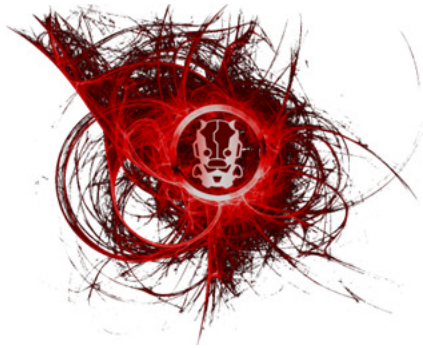
ELF parsing

- **Discovered vulnerabilities in the past**
  - Invalid pointer dereference in **IDA Pro** (patched) used as an anti-debugging technique [4]



ELF parsing

- **ELF parsing nowadays**
  - ~15 years later (adopted in 1999)
- Most ELF analysis tools rely on the SHT (Section Header Table)
- The following bugs have been found with Melkor



ELF parsing

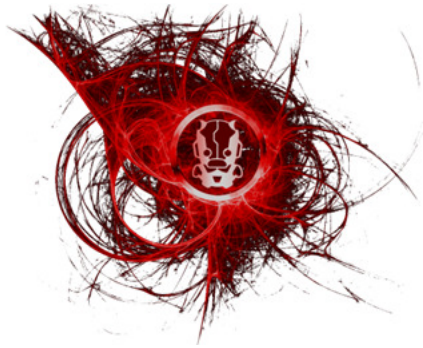
- **ELF parsing (mistakes) nowadays**
  - ***still*** blindly trust in the input:
    - Offsets
    - Indexes
    - Sizes (total_size / struct_size)
    - Addresses
    - Debugging information (DWARF)
      - Not part of ELF but it's very related

```
nitr0us@chatsubo:~/orcs$ objdump --dwarf=decodedline dwarf_AAAAAAAAAAA

dwarf_AAAAAAAAAAA:     file format elf32-i386

Decoded dump of debug contents of section .debug_line:

CU: AAAAAAAAAAA:
File name                                Line number    Starting address
*** glibc detected *** objdump: double free or corruption (!prev): 0x09967258 ***
*** glibc detected *** objdump: malloc(): memory corruption: 0x099672b0 ***
```

ELF parsing

- **ELF parsing (mistakes) nowadays**

*"The **most common mistake applied by a programmer is in trusting a field inside a binary structure** that should not be trusted.*
*"When dealing with **sections that must have subsections,** knowing ahead of time how many sections are embedded within the primary section of a structure is required and again, **a value must be used to instruct the application only to iterate x number of times."** [7]*



ELF parsing

- **ELF parsing (mistakes) nowadays**
  - Example of a dummy program that executes the for() loop based on e_shnum and size/entsize:

```
for(k = 0; k < hdr->e_shnum; k++, shdr++){
   if(shdr->sh_type != SHT_SYMTAB && shdr->sh_type != SHT_DYNSYM)
      continue;
...
   nsyms = shdr->sh_size / shdr->sh_entsize;
   sym = (Elf64_Sym *) (mem + shdr->sh_offset);
...
   for(l = 0; l < nsyms; l++, sym++)
      if(ELF64_ST_TYPE(sym->st_info) != STT_SECTION)
         printf("[%2d] %s\n", mem + strtab + sym->st_name);
}
```

ELF parsing

- **ELF parsing (mistakes) nowadays**

```
nitr0us@exiled:~$ gcc dummy_elf.c -o dummy_elf -Wall
nitr0us@exiled:~$ ./dummy_elf
Usage: ./dummy_elf <elf_file>
nitr0us@exiled:~$ ./dummy_elf foo
Symbol table [.dynsym] with 4 entries:
[NR] Symbol name
[ 0]
[ 1] puts
[ 2] __libc_start_main
[ 3] __gmon_start__
```

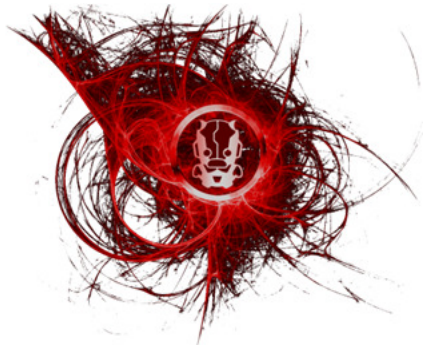
1

```
nitr0us@exiled:~$ ./dummy_elf foo
Symbol table [.dynsym] with 11184810 entries:
[NR] Symbol name
[ 0]
[ 1] puts
[ 2] __libc_start_main
[ 3] __gmon_start__
Segmentation fault (core dumped)
```

3

```
[-] section 5: .dynsym
  name string index                0000004e
  type                             0000000b (dynamic
  flags                            0000000000000002 c
  address                          00000000004002b8
  offset                           00000000000002b8
  size                             00000000ffffffff
  link                             0000000100000006
```

2

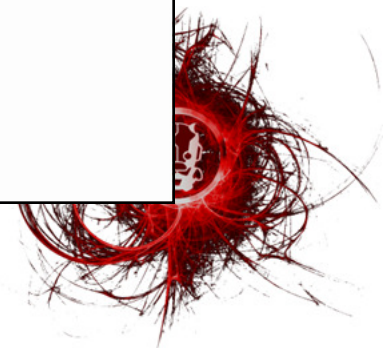


ELF parsing

- **ELF parsing (mistakes) nowadays**
  - Some applications validate if memory addresses (p_vaddr, e_entry, etc.) and/or file offsets (sh_offset, p_offset, etc.) are inside their valid boundaries
  - Some others don't:

```
-bash-4.2$ uname -a
OpenBSD calypso.localdomain 5.2 GENERIC#278 i386
-bash-4.2$ foo/NOTE_p_offset_0x41414141
foo
-bash-4.2$ readelf -lW foo/NOTE_p_offset_0x41414141 | grep NOTE
  NOTE           0x41414141 0x1c0001a8 0x1c0001a8 0x00018 0x00018
-bash-4.2$ ldconfig -vP /home/nitr0us/foo/
loading dir /home/nitr0us/foo/
processing /home/nitr0us/foo//NOTE_p_offset_0x41414141
Segmentation fault
-bash-4.2$
```

(Found with Melkor fuzzer.)



ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF parsing (mistakes) nowadays**

src/libexec/ld.so/ldconfig/prebind.c:

```
elf_check_note(void *buf, Elf_Phdr *phdr)
{
        u_long address;
        u_int *pint;
        char *osname;

        address = phdr->p_offset;
        pint = (u_int *)((char *)buf + address);
        osname = (char *)buf + address + sizeof(*pint) * 3;

        if (pint[0] == 8 /* OpenBSD\0 */ && pint[1] == 4 &&
            pint[2] == 1 /* type_osversion */ &&
            strcmp("OpenBSD", osname) == 0)
                return 1;

        return 0;
}
```

ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

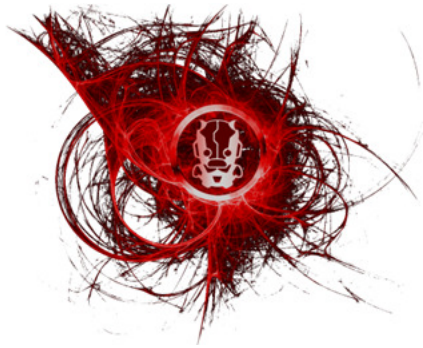
- **ELF parsing (mistakes) nowadays**
  - Some trust in sizeof(*user_input), some others prefer sizeof(Elfx_DataType) and some others perform validations:
    - `if(sizeof(*x) != sizeof(dataType))`
      `return ERROR;`

```
nitr0us@exiled:~$ readelf --relocs ./foo
readelf: Error: Section 9 has invalid sh_entsize of 0
readelf: Error: (Using the expected size of 24 for the rest of this dump)
readelf: Error: Section 10 has invalid sh_entsize of 0
readelf: Error: (Using the expected size of 24 for the rest of this dump)

Relocation section '.rela.dyn' at offset 0x380 contains 1 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000600ff8  000300000006 R_X86_64_GLOB_DAT 0000000000000000 __gmon_start__ + 0

Relocation section '.rela.plt' at offset 0x398 contains 3 entries:
  Offset          Info           Type           Sym. Value    Sym. Name + Addend
000000601018  000100000007 R_X86_64_JUMP_SLO 0000000000000000 puts + 0
000000601020  000200000007 R_X86_64_JUMP_SLO 0000000000000000 __libc_start_main + 0
000000601028  000300000007 R_X86_64_JUMP_SLO 0000000000000000 __gmon_start__ + 0
nitr0us@exiled:~$
```

ELF parsing

- **ELF parsing (mistakes) nowadays**
  - Some do not validate the number of elements before allocate memory:
    - Allocate less memory space than needed
      - Buffer overflows
    - Memory exhaustion

  - `malloc(nElems_user_input * sizeof(Elfx_Struct));`
  - `malloc(nElems_user_input * sizeof(*user_input));`
  - `calloc(nElems_user_input , sizeof(Elfx_Struct));`



ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF parsing (mistakes) nowadays**
  - Process memory exhaustion:

```
nitr0us@exiled:~$ readelf -lW orcs/gdb_mem_exhaustion | grep -A9 "Program Headers"
Program Headers:
  Type           Offset   VirtAddr           PhysAddr           FileSiz  MemSiz
  PHDR           0x000040 0x0000000000409a8f 0x0000000000409a8f 0x80000000c0000000
  NULL           0xfffffffffffff0000 0x0000000000400238 0x0000000000400238 0x00001c
  LOAD           0x000000 0x0000000000400000 0x0000000000400000 0x000d24 0x000d24
  LOAD           0x000e10 0x0000000000600e10 0x0000000000600e10 0x000290 0x0002a0
  DYNAMIC        0x000e28 0x000000000defaced 0x000000000defaced 0x79a1fa74fee1dead
  LOPROC+f00ff01 0x000254 0x0000000000400254 0x0000000000400254 0x313374021cea9 0x
  GNU_EH_FRAME   0x000be4 0xffffffff80000000 0xffffffff80000000 0xff00ff004d33b8cd
  NULL           0x000000 0x0000000044444444 0x0000000044444444 0x000000 0x000000
nitr0us@exiled:~$ gdb -q orcs/gdb_mem_exhaustion
Reading symbols from orcs/gdb_mem_exhaustion...done.
(gdb) r
Starting program: /home/nitr0us/orcs/gdb_mem_exhaustion
utils.c:1073: internal-error: virtual memory exhausted.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Quit this debugging session? (y or n) y
utils.c:1073: internal-error: virtual memory exhausted.
A problem internal to GDB has been detected,
further debugging may prove unreliable.
Create a core file of GDB? (y or n) y
Aborted (core dumped)
nitr0us@exiled:~$
```

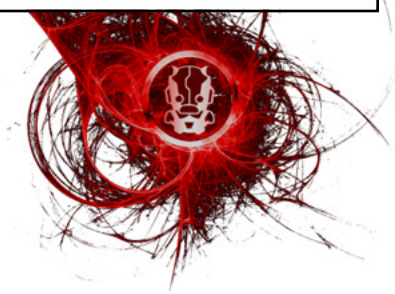
(Found with Melkor fuzzer.)

ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF parsing (mistakes) nowadays**
  - Process memory exhaustion:

```
(gdb) r
Starting program: /home/nitr0us/orcs/gdb_mem_exhaustion

Breakpoint 2, xmalloc (size=18446744072635809792) at ./common/common-utils.c:53
53          malloc_failure (size);
(gdb) p/x 18446744072635809792
$1 = 0xfffffffffc0000000
(gdb) bt
#0  xmalloc (size=18446744072635809792) at ./common/common-utils.c:53
#1  0x0000000000435b1a in read_program_header (type=2, p_sect_size=0x7fffffffdad8, p_arch_size=0x7fffffffdadc)
    at solib-svr4.c:570
#2  0x0000000000436013 in scan_dyntag_auxv (dyntag=1879048214, ptr=0x7fffffffdb40) at solib-svr4.c:729
#3  0x0000000000436180 in elf_locate_base () at solib-svr4.c:797
#4  0x00000000004362c0 in locate_base (info=0xd4ce80) at solib-svr4.c:861
#5  0x0000000000436cd8 in svr4_current_sos_direct (info=0xd4ce80) at solib-svr4.c:1429
#6  0x0000000000436dfe in svr4_current_sos () at solib-svr4.c:1480
#7  0x00000000006a9c4b in update_solib_list (from_tty=0, target=0xc55060 <current_target>) at solib.c:704
#8  0x00000000006aa0ab in solib_add (pattern=0x0, from_tty=0, target=0xc55060 <current_target>, readsyms=1)
    at solib.c:918
#9  0x0000000000437b14 in enable_break (info=0xd4ce80, from_tty=0) at solib-svr4.c:2108
#10 0x00000000004394d6 in svr4_solib_create_inferior_hook (from_tty=0) at solib-svr4.c:2928
#11 0x00000000006aa7d9 in solib_create_inferior_hook (from_tty=0) at solib.c:1200
#12 0x0000000000564341 in post_create_inferior (target=0xc55060 <current_target>, from_tty=0) at infcmd.c:438
#13 0x00000000005646e6 in run_command_1 (args=0x0, from_tty=1, tbreak_at_main=0) at infcmd.c:606
```



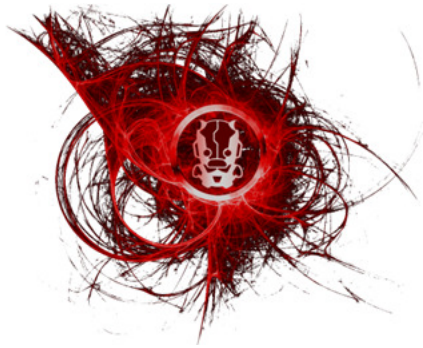
ELF parsing

- **ELF parsing (mistakes) nowadays**
  - More loops based on sh_size/sh_entsize running more times than expected. (**sh_size = 0xbad0c0de**):
    - Temporary DoS (CPU usage) in HT Editor:

```
nitr0us@exiled:~$ readelf -SW foo_rela_plt_sh_size_0xBAD0C0DE | grep rela.plt
  [10] .rela.plt         RELA            0000000000400398 000398 bad0c0de 18   A  5   1
nitr0us@exiled:~$ time /usr/local/bin/ht ./foo_rela_plt_sh_size_0xBAD0C0DE
KilledEdit Windows Help
┌─[x]──────────────────────────────────────────────────────────── log window ──────────
real2.0.0m18.972s) 21:16:16 on Jan 22 2014
user 1990m2.890stefan Weyergraf
sys) 1990m2.034sebastian Biallas <sb@biallas.net>
nitr0us@exiled:~$ cal/bin/ht
config = /home/nitr0us/.htcfg2
loading binary file /home/nitr0us/foo_rela_plt_sh_size_0xBAD0C0DE...
/home/nitr0us/foo_rela_plt_sh_size_0xBAD0C0DE: ELF: found header at 0x00000000
/home/nitr0us/foo_rela_plt_sh_size_0xBAD0C0DE: ELF: loading image (starting analyser)
```

(Found with Melkor fuzzer.)

- The application was killed by the OS after ~19 secs.



ELF parsing

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF parsing (mistakes) nowadays**

ht-2.0.22/htelfrel.cc:

```
rela_size = sizeof (ELF_RELA64);
relnum = elf_shared->sheaders.sheaders64[reloctab_shidx].sh_size /
(reloctab_sh_type == ELF_SHT_REL ? rel_size : rela_size);
...
for (uint i = 0; i < relnum; i++){
  char *tt = t;
  /* dest offset */
  tt = tag_make_edit_qword(tt, tt_end, h+i*rel_size, endianness);
  tt += ht_snprintf(tt, tt_end, " ");

  /* symbol (table idx) */
  tt = tag_make_edit_dword(tt, tt_end, h+i*rel_size+8+4,
endianness);
  tt += ht_snprintf(tt, tt_end, "   ");
...
}
```

ELF parsing

- **ELF parsing (mistakes) nowadays**
  - A common low-hanging fruit crash is through e_shstrndx in the ELF header. It holds a string table index within the Section Header Table:

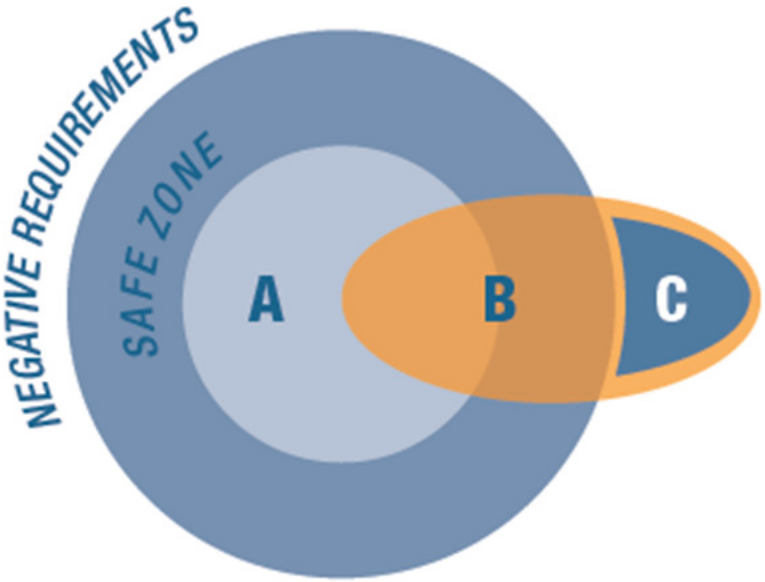
```
nitr0us@chatsubo:~$ cat ./ELFsh_crash.esh
#!/usr/bin/env elfsh
load /bin/ls
set 1.hdr.shstrndx 0xffff
save malformed
# quit never reached
quit
```

```
nitr0us@chatsubo:~$ ./ELFsh_crash.esh


         The ELF shell 0.83 (32 bits built) .::.

         .::. This software is under the General Public License V.2
         .::. Please visit http://www.gnu.org

~quiet
 [*] Set ELFsh default color theme (use nocolor to disable)

 [*] /home/nitr0us/.elfshrc sourcing -OK-
 [*] Type help for regular commands


we are in script mode from revm_loop !
~load  /bin/ls

 [*] Mon Apr 28 11:32:54 2014 - New object loaded : /bin/ls

~set  1.hdr.shstrndx 0xffff
 [*] Expression set succesfully

~save  malformed
Segmentation fault
nitr0us@chatsubo:~$
```

(Found with Melkor fuzzer.)

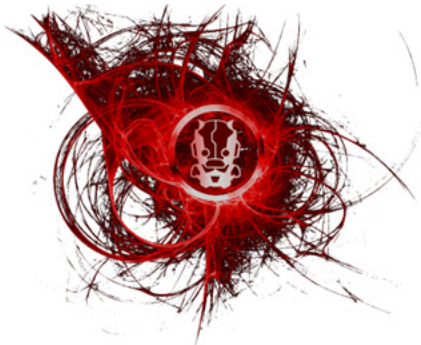
ELF fuzzing

- **Fuzz testing**
  - Automated approach to create invalid / semi-valid data to find bugs that would have often been missed by human eyes
    - If data is too valid, might not cause problems
    - If data is too invalid, might be quickly rejected [9]



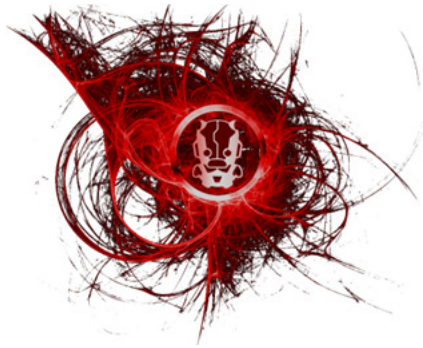
A Desired functionality
B Implemented functionality
C Undesired functionality

Taken from [5]



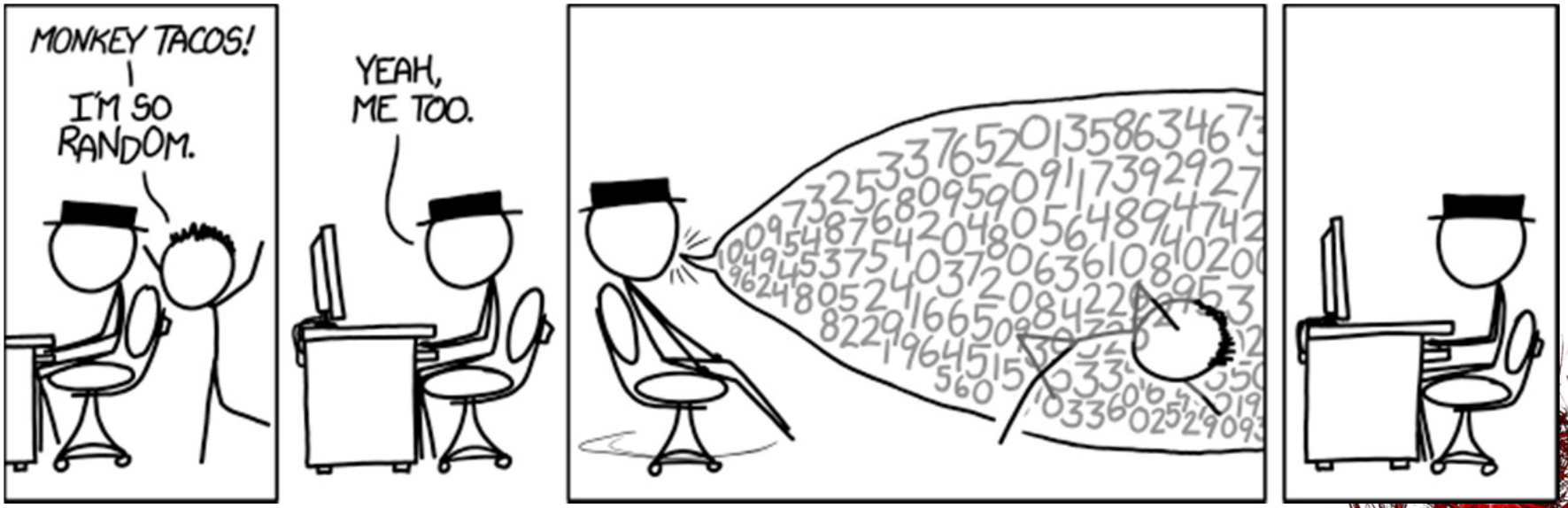
ELF fuzzing

- **Smart vs dumb fuzzing**
  - Two approaches
    - Mutation-based fuzzing (**dumb**)
      - Takes an input and modifies it randomly
    - Generation-based fuzzing (**smart**)
      - Generates the tests with specification knowledge
  - ~~dumb~~ random fuzzing in most cases find less bugs than smart fuzzing



ELF fuzzing

- **Smart vs dumb fuzzing** [2]
  - All paths + all data == infinite problem
  - Notion of randomness (dumbness) and specific knowledge (intelligence)
    - Semi-valid data

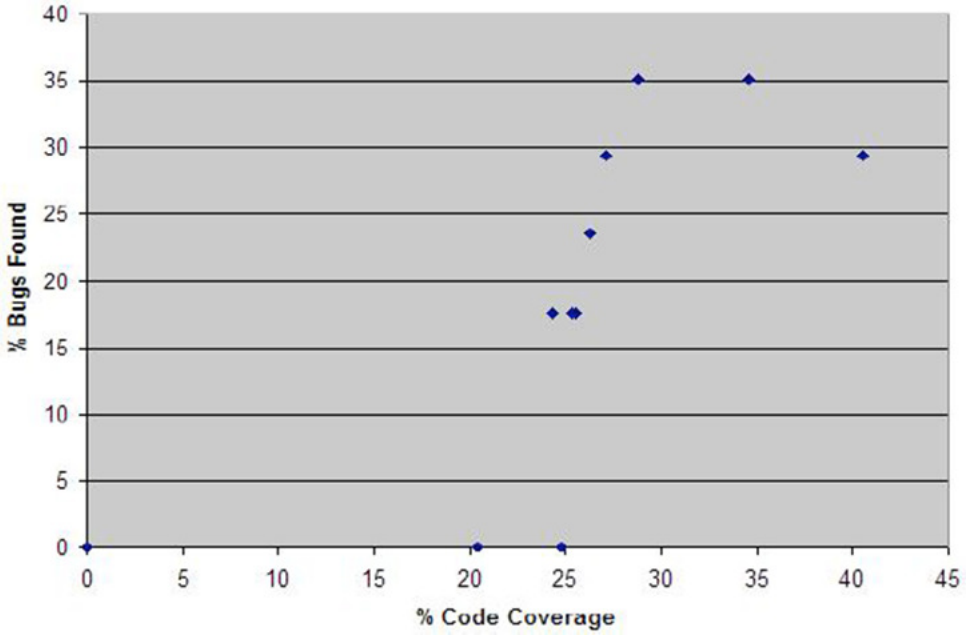


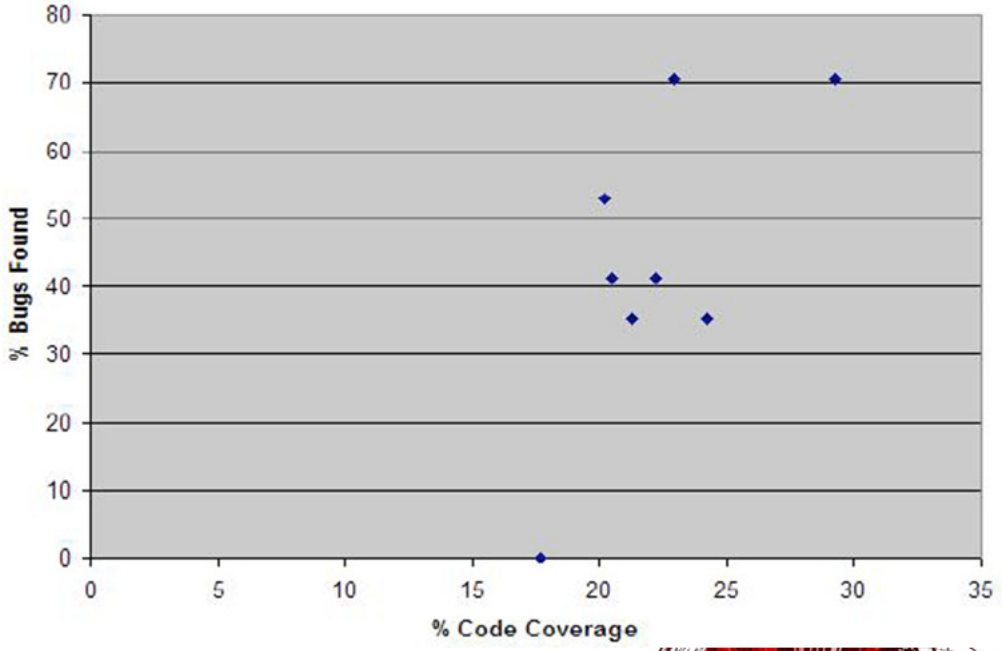
ELF fuzzing

- **Code / branch coverage** [8]
  - Code coverage is a metric which can be used to determine how much code has been executed
  - Branch coverage measures how many branches in code have been taken (conditional jmps)
    - `if( x > 2 )`
      `x = 2;`
  - Specification based test generation achieves better coverage testing

ELF fuzzing

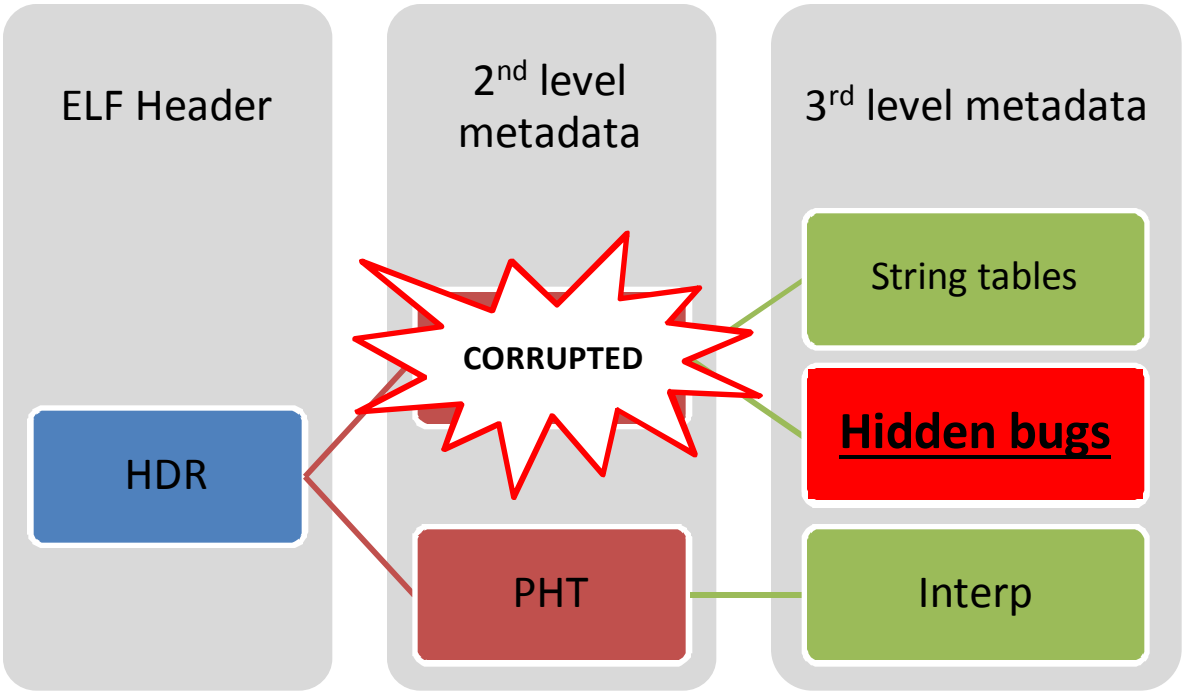
- **Code / branch coverage**
  - Interesting results in [9] show that more bugs are discovered with higher coverage:

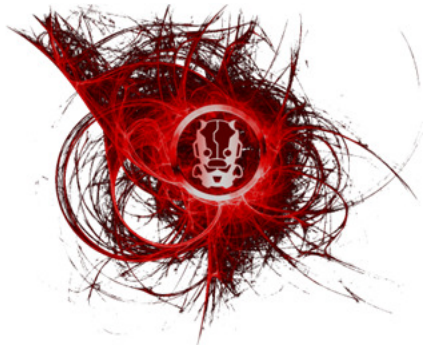




ELF fuzzing

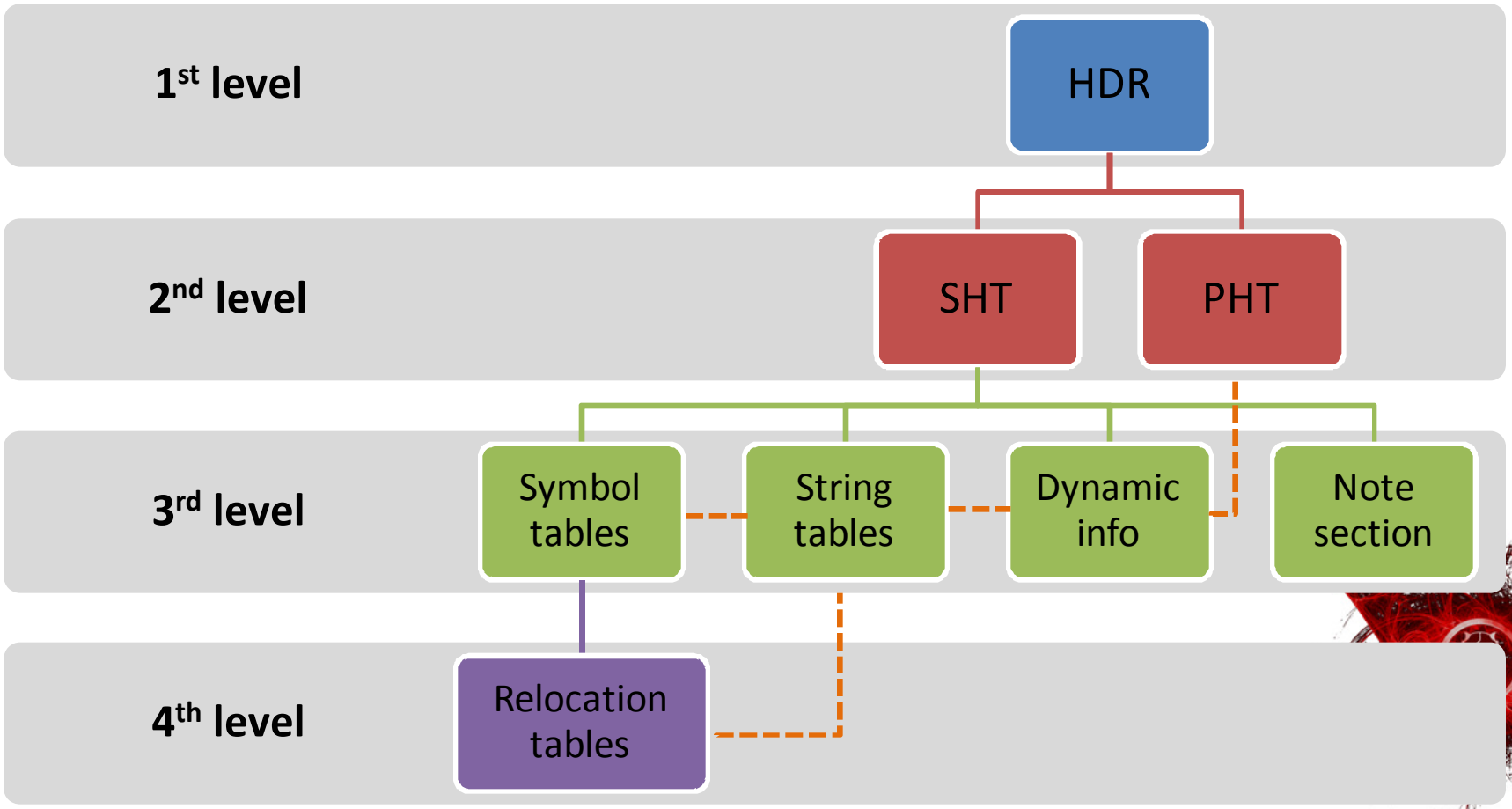
- **ELF metadata dependencies**
  - Some data structures must be fuzzed in the end or not fuzzed at all for higher code / branch coverage:





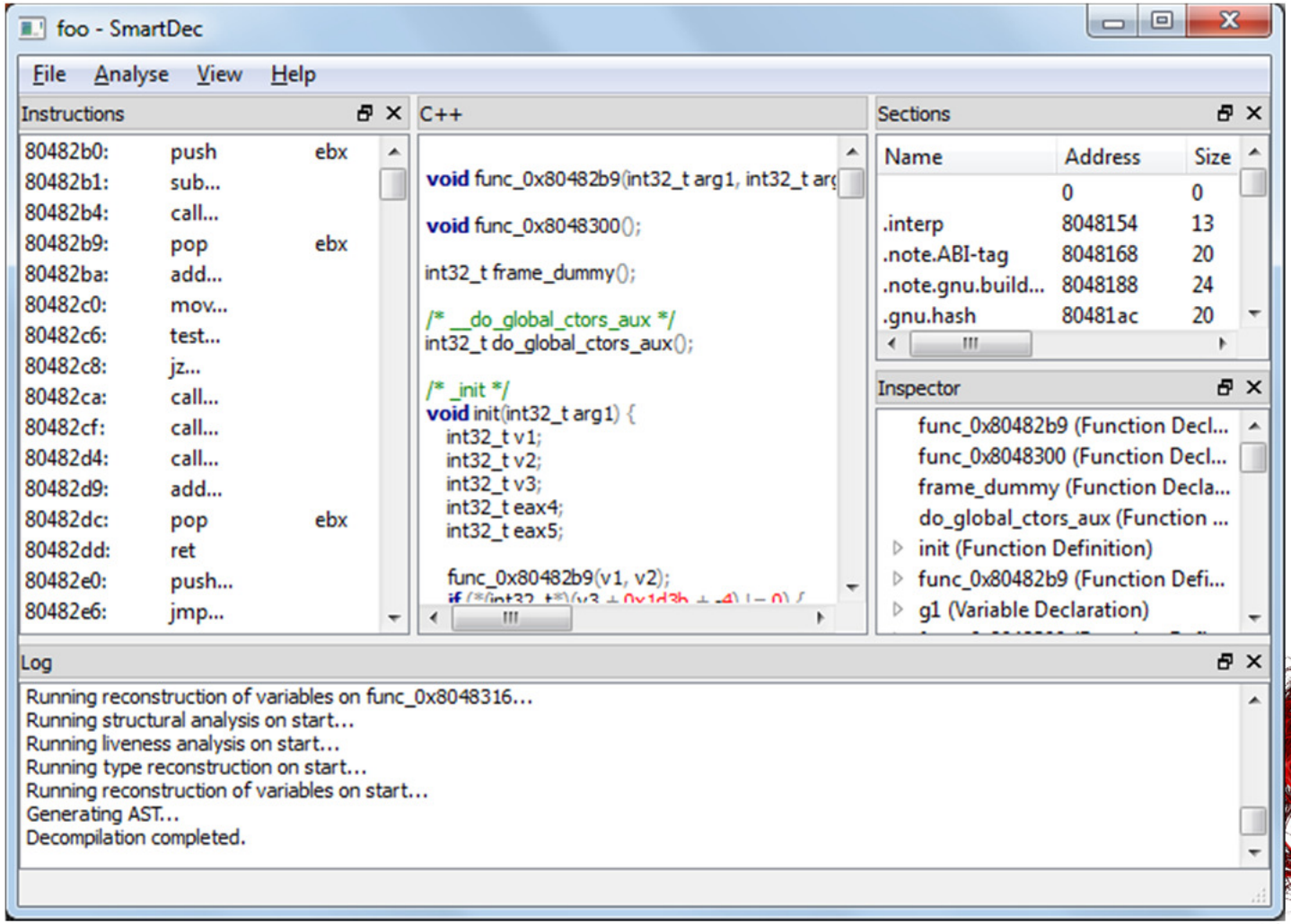
ELF fuzzing

- **ELF metadata dependencies**
  - In normal circumstances, the following ELF metadata dependencies exist while parsing:



ELF fuzzing

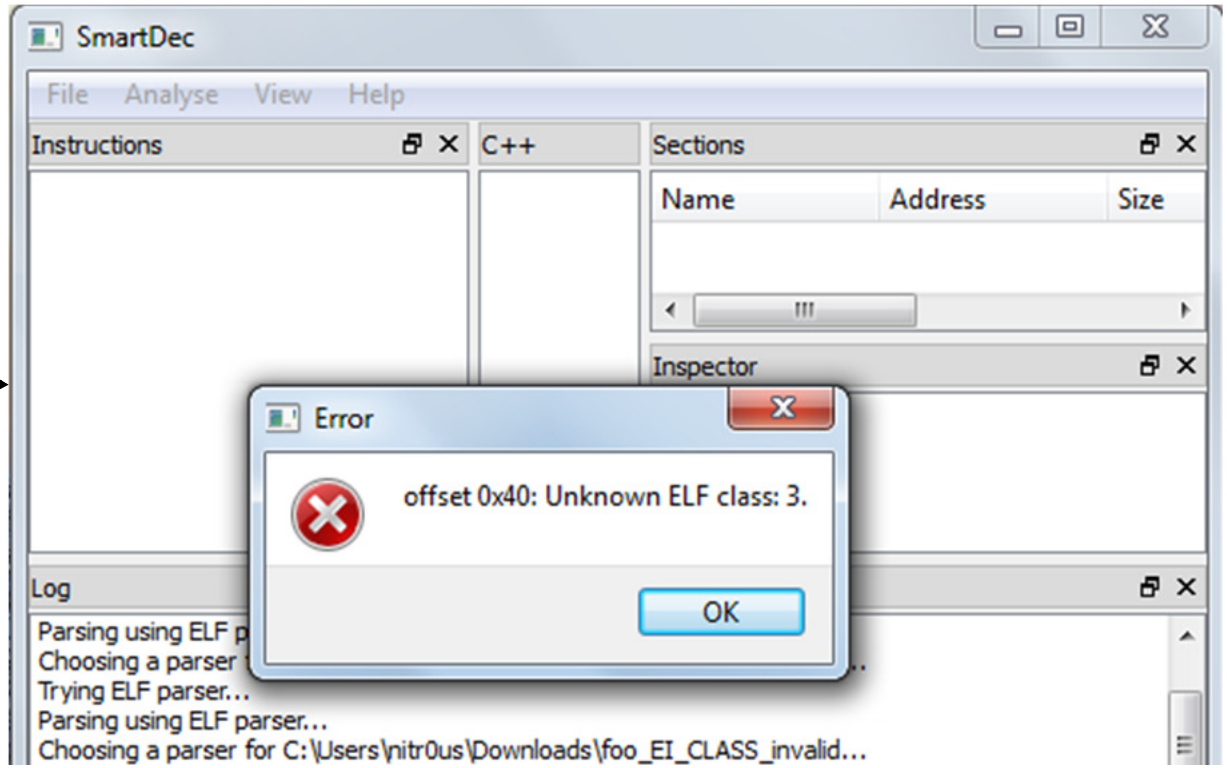
- Example: (SmartDec, Native code to C/C++ Decompiler for Windows)
  - Normal ELF loading

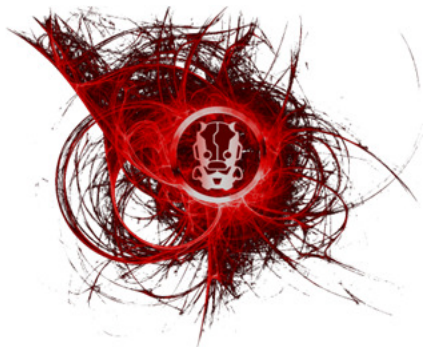


ELF fuzzing

- Example: (SmartDec, Native code to C/C++ Decompiler for Windows)
  - Trying to load the same ELF with an invalid e_ident[EI_CLASS] (a header field), it simply handles the error and doesn't open:

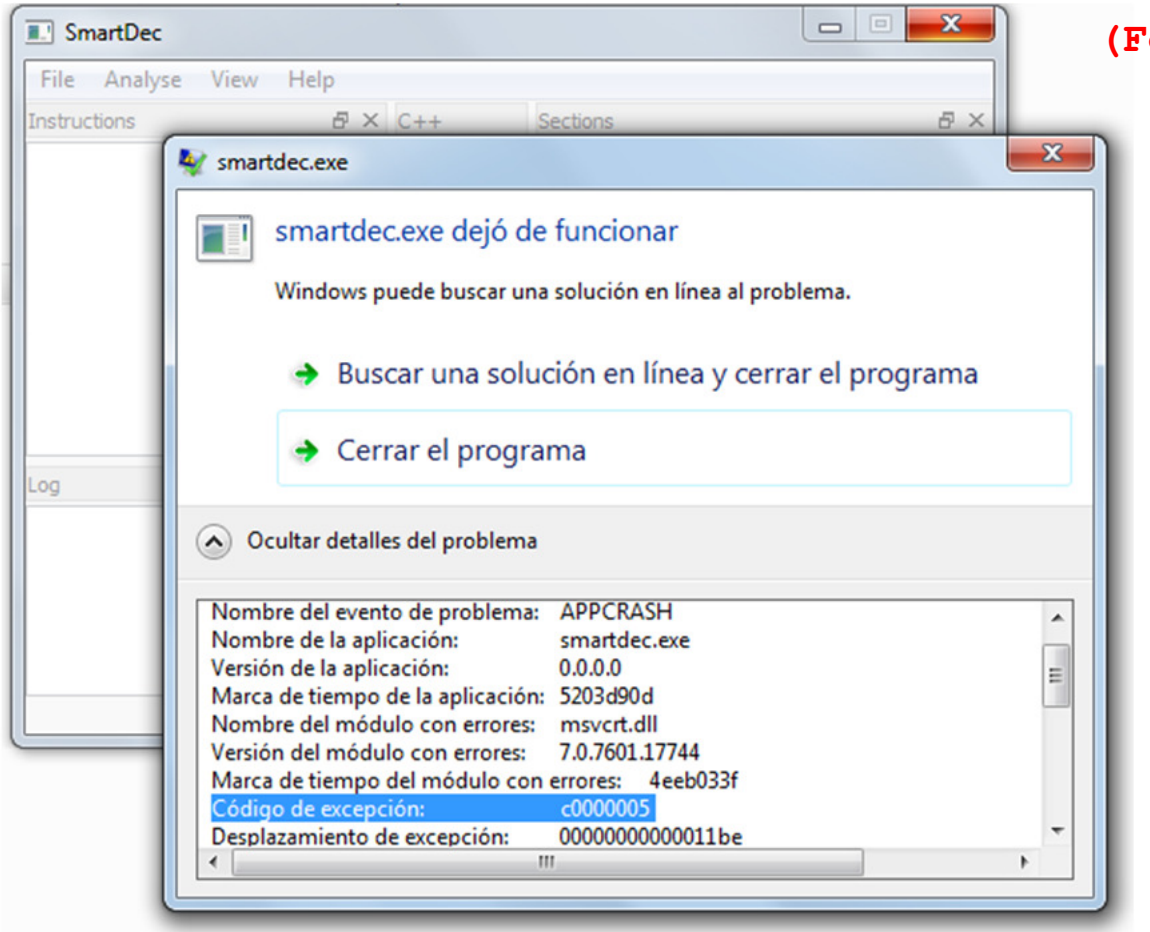
```
nitr0us@chatsubo:~$ readelf -h foo_EI_CLASS_invalid | grep Class
  Class:                             <unknown: 3>
```





ELF fuzzing

- Example: (SmartDec, Native code to C/C++ Decompiler for Windows)
  - However, having an unmodified header, the basic header validations will be bypassed and internal bugs are reached:



(Found with Melkor fuzzer.)

Melkor fuzzer

- **Who's Melkor**
  - A fictional character from J. R. R. Tolkien's Middle-earth legendarium
  - Was the first Dark Lord and master of Sauron

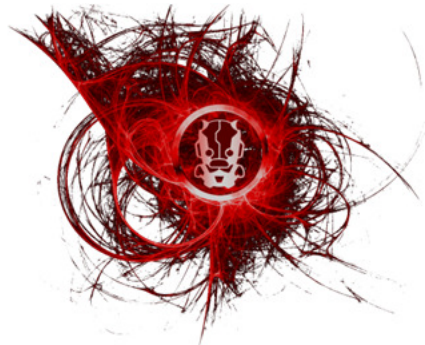
Melkor fuzzer

- **Who's Melkor**
  - Mentioned briefly in The Lord of the Rings and is known for:

"... Melkor had **captured a number of ELVES** before the Valar attacked him, and he **tortured and corrupted them, breeding the first Orcs.**"

"... Melkor was cunning and more filled with malice than ever. Seeing the bliss of the ELVES and remembering that it was for their sake that he was overthrown, **Melkor desired above all things to corrupt them.**"

"Orcs...This has been so from the day they were bred by Melkor from corrupted, tortured and **mutilated ELVES** that may also have been **forced to breed with other unnatural abominations** in the dominion of the Dark Powers."



Melkor fuzzer

- **Hybrid (Mutation-based / Generation-based)**
  - **Mutate** existing data in an ELF sample to create orcs with **knowledge of the file format specification** (fuzzing rules)

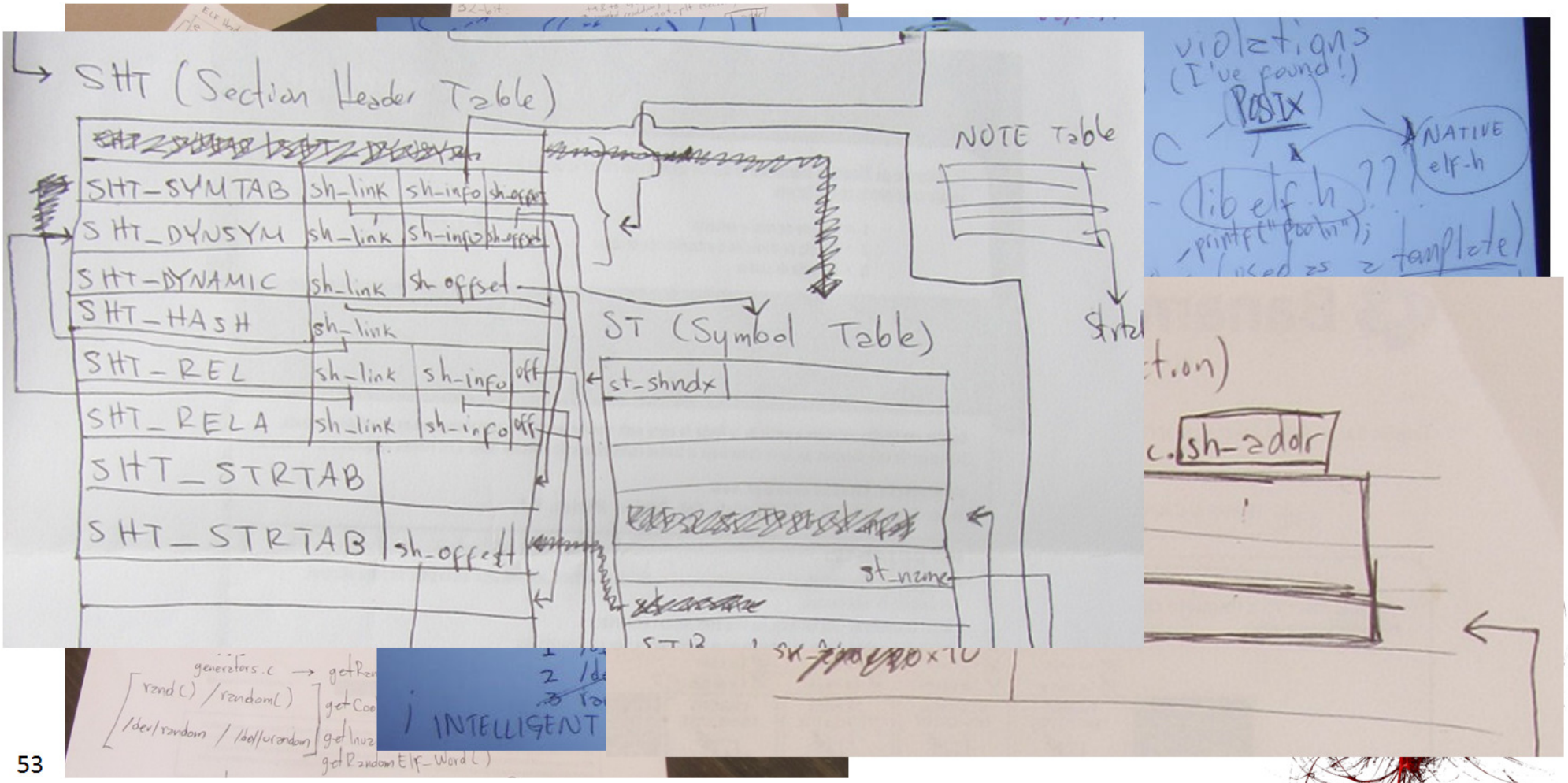
ELF

Melkor fuzzer

Orcs

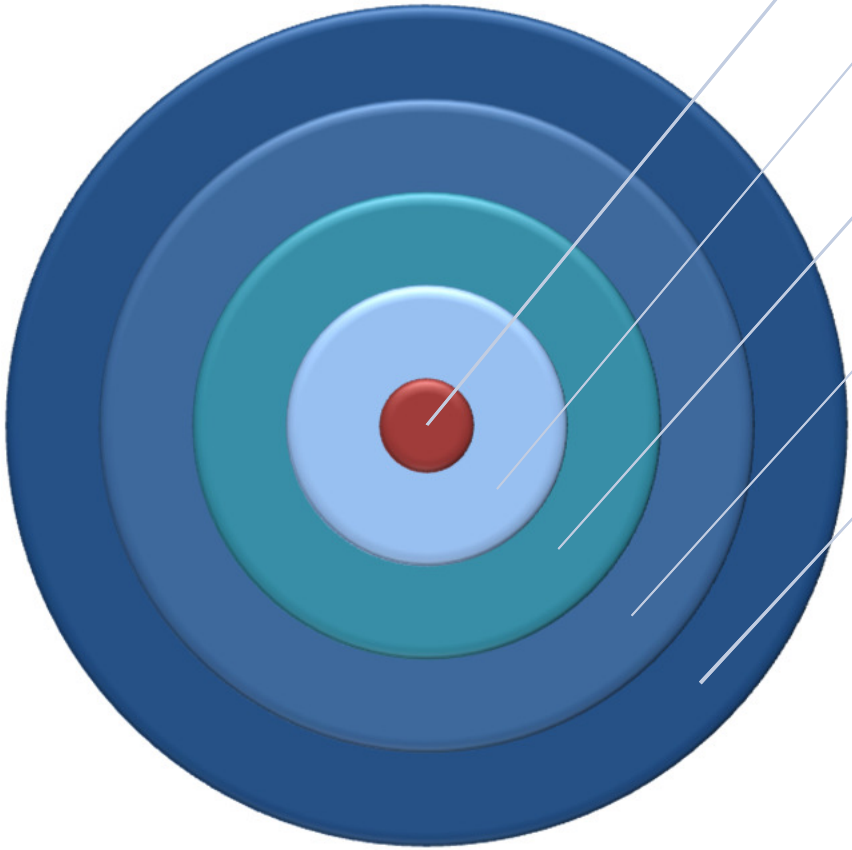
Melkor fuzzer

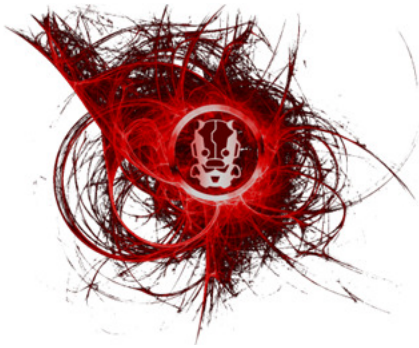
- **Design**
  - I'm not good at software design but... drawing worked



Melkor fuzzer

- Implementation





Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - Three inputs were used:
    - **Specification violations**
      - TIS ELF Specification 1.2 (May, 1995) [10]
      - ELF-64 Object File Format 1.5 (May 1998)
    - Misc. ideas & considerations
    - Parsing patterns seen in ELF software

Metadata		Number of Rules
HDR	Header	19
PHT	Program Header Table	22
SHT	Section Header Table	37
STRS	String Table	3
DYN	Dynamic Section	18
NOTE	Note Section	4
SYM	Symbols Table	15
REL	Relocations Table	3
HASH	Hash Table	2
ENV	OS Environment Variables	3
	Total:	126

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - melkor-v1.0/docs/Melkor_Fuzzing_Rules.pdf

ELF Specification 1.2 Violations
XXX: Value to be fuzzed with semi-valid semantics

Rule	Specification	Violation description	ELF metadata	Page
hdr1	A program header table, if present, tells the system how to create a process image. Files used to build a process image (execute a program) must have a program header table;	Executable ELF without a PHT	- HDR e_type = ET_EXEC | ET_DYN e_phoff = 0 e_phentsize = 0 e_phoff = Valid offset e_phnum = 0 e_phentsize = Valid number	16
hdr2	A section header table contains information describing the file's sections. Every section has an entry in the table; Files used during linking must have a section header table;	Relocatable file without a SHT Empty SHT	- HDR e_type = ET_REL e_shoff = 0 e_shentsize = 0 e_shoff = Valid offset e_shnum = 0 e_shentsize = Valid number	16
hdr3	e_type This member identifies the object file type. Values from ET_LOPROC through ET_HIPROC (inclusive) are reserved for processor-specific semantics. Other values are reserved and will be assigned to new object file types as necessary	ELF type set to normal values (< 5), invalid and uncommon values (>= 5) or zero.	- HDR e_type < 5 e_type >= 5 <= ET_HIPROC e_type = 0	19
hdr4	e_machine This member's value specifies the required architecture for an individual file	ELF machine with invalid / uncommon values	- HDR e_machine > 16 e_machine = 0	19
hdr5	e_entry If the file has no associated entry point, this member holds zero.	Invalid entry point (0 or out of range) Some point in kernel-land	- HDR e_entry = XXX e_entry = 0	19
hdr6	e_phoff This member holds the program header table's file offset in bytes. If the file has no program header table, this member holds zero.	PHT out of bounds	- HDR e_phoff = XXX	19
hdr7	e_ehsize This member holds the ELF header's size in bytes.	Random ELF header size	- HDR e_ehsize = XXX	19
hdr8	e_phentsize This member holds the size in bytes of one entry in the file's program header table; all entries are the same size. e_phnum This member holds the number of entries in the program header table. Thus the product of e_phentsize and e_phnum gives the table's size in bytes. If a file has no program header table, e_phnum holds the value zero.	Combination of low and high values e_phentsize to zero	- HDR e_phentsize = XXX e_phnum = XXX e_phentsize = 0	20

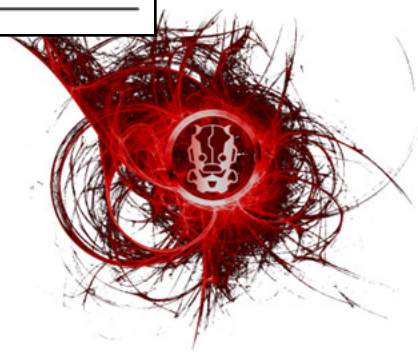
Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - Specification violations (Example 1)
    - ELF Specification:

e_type — This member identifies the object file type.

Name	Value	Meaning
ET_NONE	0	No file type
ET_REL	1	Relocatable file
ET_EXEC	2	Executable file
ET_DYN	3	Shared object file
ET_CORE	4	Core file
ET_LOPROC	0xff00	Processor-specific
ET_HIPROC	0xffff	Processor-specific



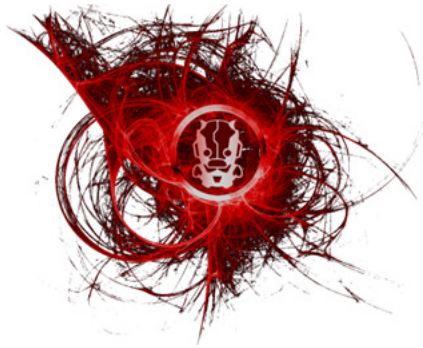
Melkor fuzzer

- **Fuzzing rules**
  - Specification violations (Example 1)
    - Rule definition for that field:

Specification	Violation description
e_type This member identifies the object file type. Values from ET_LOPROC through ET_HIPROC (inclusive) are reserved for processor-specific semantics. Other values are reserved and will be assigned to new object file types as necessary	ELF type set to normal values (< 5), invalid and uncommon values (>= 5) or zero.

ELF metadata	Page	Rule
- HDR e_type < 5 e_type >= 5 <= ET_HIPROC e_type = 0	19	hdr3

Rule name implemented at code level



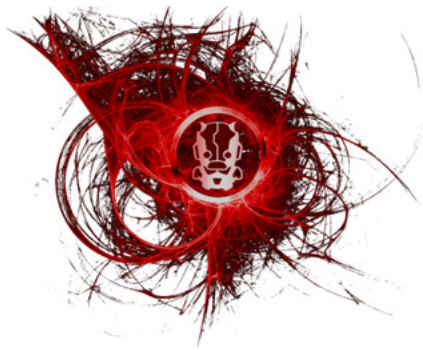
Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - Specification violations (Example 1)
    - That rule at code level:

ELF metadata	Page	Rule
- HDR e_type < 5 e_type >= 5 <= ET_HIPROC e_type = 0	19	hdr3

```
int hdr3(void)
{
        Elf_Half e_type;

        if(rand() % 2) // 50% chance
                e_type = getElf_Half() % ET_NUM;
        else {
                if((rand() % 4) < 3){ // .5 * .75 = 37.5% chance
                        while((e_type = (getElf_Half() % ET_HIPROC)))
                                if(e_type >= 5 && e_type <= ET_HIPROC)
                                        break;
                } else // .5 * .25 = 12.5% chance
                        e_type = 0;
        }

        orcHDR->e_type = e_type;

        fprintf(logfp, "(HDR->e_type = 0x%x)", orcHDR->e_type);

        return 1;
}
```



Melkor fuzzer

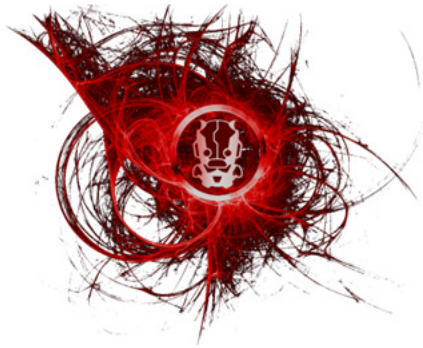
IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - Specification violations (Example 2)

sh_addralign	Some sections have address alignment constraints. For example, if a section holds a doubleword, the system must ensure doubleword alignment for the entire section. That is, the value of sh_addr must be congruent to 0, modulo the value of sh_addralign. Currently, only 0 and positive integral powers of two are allowed. Values 0 and 1 mean the section has no alignment constraints.

Specification	Violation description
sh_addralign That is, the value of sh_addr must be congruent to 0, modulo the value of sh_addralign. Currently, only 0 and positive integral powers of two are allowed. Values 0 and 1 mean the section has no alignment constraints.	sh_addralign not power of two (3,5,7,9,10,11,12,13,14,15,17, etc.) sh_addralign = PAGESIZE +/- 1

ELF metadata	Page	Rule
- SHT sh_addralign = XXX sh_addralign = PAGESIZE - 1 sh_addralign = PAGESIZE + 1	25	sht5



Melkor fuzzer

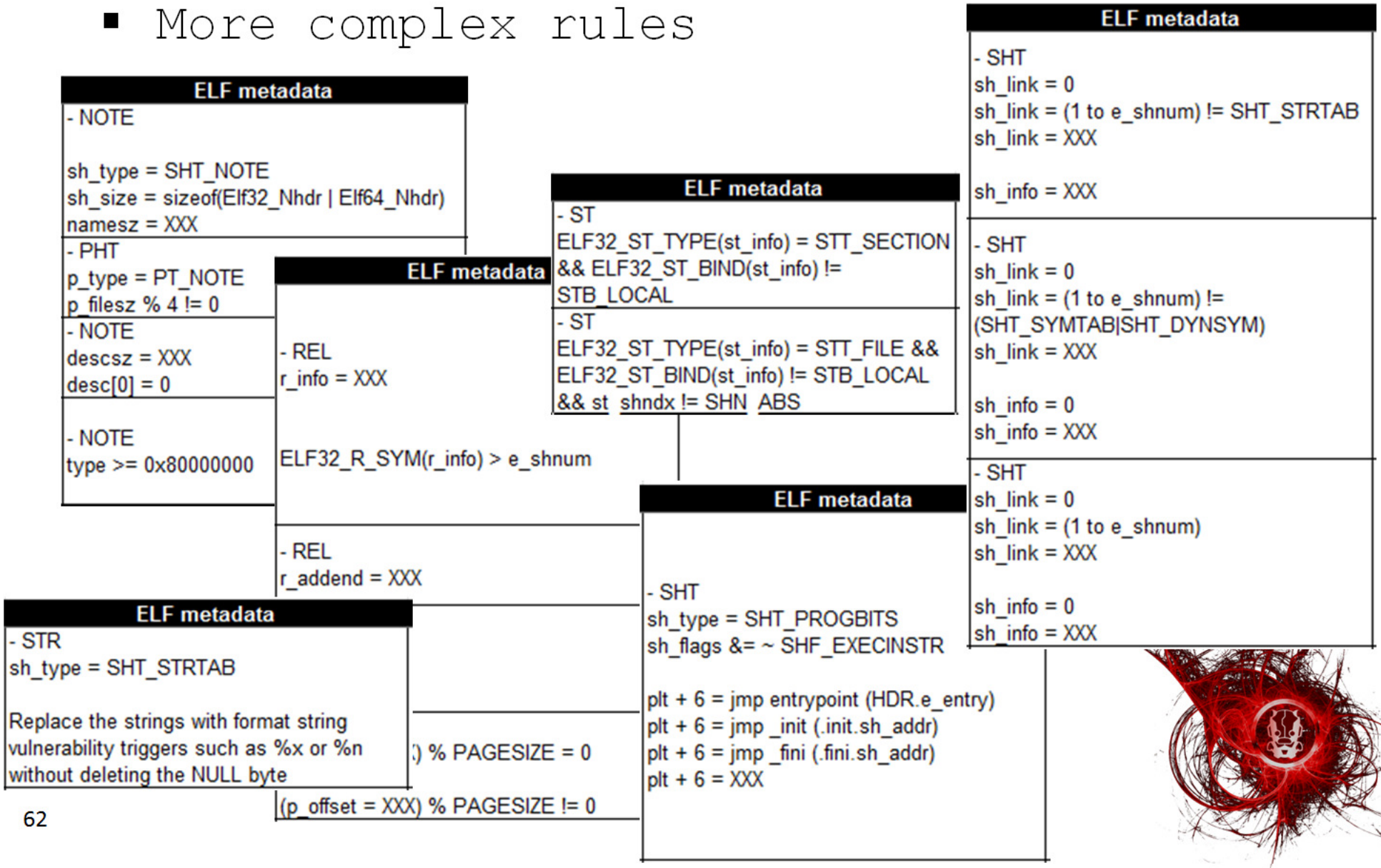
- **Fuzzing rules**
  - Specification violations (Example 2)

ELF metadata	Page	Rule
- SHT sh_addralign = XXX sh_addralign = PAGESIZE - 1 sh_addralign = PAGESIZE + 1	25	sht5

```
int sht5(void)
{
        if(rand() % 2){ // 50% chance
#if defined(__i386__)
                while((orcSHT->sh_addralign = getElf_Word()))
#elif defined(__x86_64__)
                while((orcSHT->sh_addralign = getElf_Xword()))
#endif
                        // Bitwise: x & (x - 1) != 0 if x is NOT a power of 2
                        if((orcSHT->sh_addralign & (orcSHT->sh_addralign - 1)) != 0)
                                break;
        } else {
                if(rand() % 2) // 25%
                        orcSHT->sh_addralign = PAGESIZE - 1;
                else // 25%
                        orcSHT->sh_addralign = PAGESIZE + 1;
        }

        fprintf(logfp, "(SHT[%d]->sh_addralign = 0x"HEX")", sh, orcSHT->sh_addralign);

        return 1;
}
```

Melkor fuzzer

- **Fuzzing rules**
  - More complex rules



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

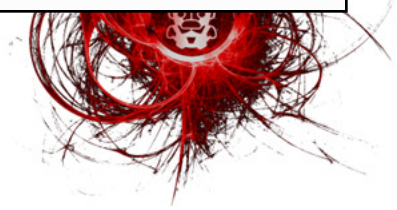
- **Fuzzing rules**
  - More complex rules (Example 1)

```
} else { // Binary patch: the second jmp instruction in the PLT
        unsigned int jmp_asm;
        int r = rand();

        // The 1st jmp in PLT is 6 bytes length in x86 and x86_64
        *(orcptr + elfSHT[plt].sh_offset + 6) = 0xff; // jmp opcode
        *(orcptr + elfSHT[plt].sh_offset + 7) = 0x25; // jmp opcode

        if(r % 4 == 0) // jmp to the original entrypoint
                jmp_asm = (unsigned int) elfHDR->e_entry;
        else if(r % 4 == 1){ // jmp to _init (".init".sh_addr)
                Elf_Section init = findSectionIndexByName(".init");

                if(!init)
                        return 0;

                jmp_asm = (unsigned int) elfSHT[init].sh_addr;
        } else if(r % 4 == 2){ // jmp to _fini (".fini".sh_addr)
                Elf_Section fini = findSectionIndexByName(".fini");

                if(!fini)
                        return 0;

                jmp_asm = (unsigned int) elfSHT[fini].sh_addr;
        } else { // jmp to a semi-random address
                jmp_asm = (unsigned int) getElf_Addr();
                if(rand() % 2)
                        jmp_asm = SWAP32(jmp_asm); // little-endian conversion, just for phun ;D
        }

        memcpy(orcptr + elfSHT[plt].sh_offset + 8, &jmp_asm, sizeof(jmp_asm));
```

Melkor fuzzer

- **Fuzzing rules**
  - More complex rules (Example 2)

```
int strs2(void)
{
        fprintf(logfp, "(STRS[%d]->sh_offset (0x%x) + ", secnum, (unsigned int) orcSHT->sh_offset);

        if(rand() % 2){
                unsigned int ptr_offset = 1;

                while(ptr_offset < orcSHT->sh_size - 1){
                        if(*(orcSTRS + ptr_offset) != 0){
                                ptr_offset += rand() % 5;
                                continue;
                        }

                        *(orcSTRS + ptr_offset) = (rand() & 0x7f) + 0x80; // > 7-bit ASCII chars
                        fprintf(logfp, "%d = %c (0x%.2x), ", ptr_offset, *(orcSTRS + ptr_offset), *(orcSTRS + ptr_offset) & 0xff);

                        ptr_offset += rand() % 5;
                }
        } else {
                *(orcSTRS) = (rand() & 0x7f) + 0x80; // > 7-bit ASCII chars
                fprintf(logfp, "0 = %c (0x%.2x), ", *(orcSTRS), *(orcSTRS) & 0xff);

                *(orcSTRS + orcSHT->sh_size - 1) = (rand() & 0x7f) + 0x80;
                fprintf(logfp, "%d = %c (0x%.2x), ", (int) orcSHT->sh_size - 1, *(orcSTRS + orcSHT->sh_size - 1), *(orcSTRS + orcSH
0xff);
        }

        fprintf(logfp, ")");

        return 1;
}
```



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Fuzzing rules**
  - More complex rules (Example 3)

```
int pht20(void)
{
        int a, b;
        Elf_Phdr swap;

        // Bubble sort algorithm to put the PT_LOAD segments in decreasing order based in their p_vaddr
        for(a = 0; a < orcHDR->e_phnum - 1; a++){
                if(orcOrigPHT[a].p_type != PT_LOAD)
                        continue;

                for(b = 0; b < orcHDR->e_phnum - 1 - a; b++){
                        if(orcOrigPHT[b].p_type != PT_LOAD)
                                continue;

                        if(orcOrigPHT[b].p_vaddr < orcOrigPHT[a].p_vaddr){
                                memcpy(&swap, &orcOrigPHT[b], sizeof(Elf_Phdr));
                                memcpy(&orcOrigPHT[b], &orcOrigPHT[b + 1], sizeof(Elf_Phdr));
                                memcpy(&orcOrigPHT[b + 1], &swap, sizeof(Elf_Phdr));
                        }
                }
        }

        fprintf(logfp, "(PHT[PT_LOAD].p_vaddr reordered [descending])");

        return 1;
}
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

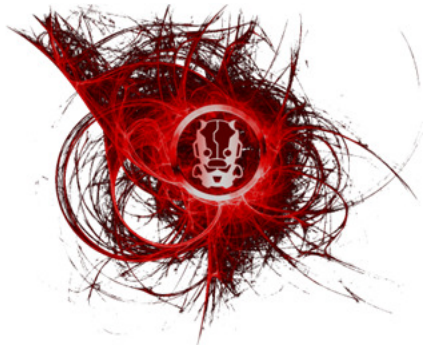
- **Fuzzing rules execution**
  - To iterate through the rules an array of function pointers is created in every fuzzing module and initialized with __attribute__((constructor)

```
/* Function pointer type 'func_ptr'.
   It will be used to create arrays of function pointers in fuzz_*.c
*/
typedef int (*func_ptr)(void);


#define N_RULES_SHT 38 // Total of fuzzing rules defined for this metadata type

// Array of function pointers. Index cero won't be used. The fuzzing rules start from index 1
func_ptr sht[N_RULES_SHT + 1];

void initialize_sht_funcs(void) __attribute__((constructor));

void initialize_sht_funcs(void)
{
        sht[1]  = &sht1;
        sht[2]  = &sht2;
        sht[3]  = &sht3;
        sht[4]  = &sht4;
        sht[5]  = &sht5;
        sht[6]  = &sht6;
        sht[7]  = &sht7;
        sht[8]  = &sht8;
        sht[9]  = &sht9;
        sht[10] = &sht10;
        sht[11] = &sht11;
```

Melkor fuzzer

- **Fuzzing rules execution**
  - Chances of execution given by the -l (likelihood parameter, default 10%) is translated to two variables that'll be used in ...

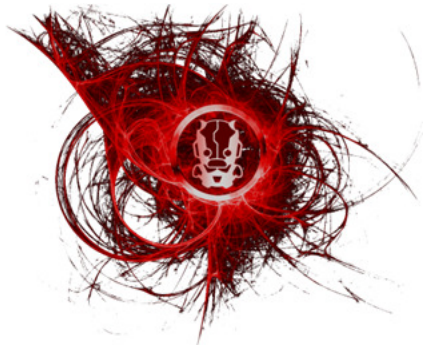
```
case 'l':
        likelihood = atoi(optarg);
        if(likelihood < 1 || likelihood > 100){
                fprintf(stderr, "[!] Likelihood (-l)
                exit(EXIT_FAILURE);
        }

        /*
        rand() % 20 < 1 =   5%
        rand() % 10 < 1 =  10%
        rand() % 5 < 1  =  20%
        rand() % 4 < 1  =  25%
        rand() % 3 < 1  =  33.33%
        rand() % 5 < 2  =  40%
        rand() % 2 < 1  =  50%
        rand() % 5 < 3  =  60%
        rand() % 3 < 2  =  66.66%
        rand() % 4 < 3  =  75%
        rand() % 5 < 4  =  80%
        rand() % 10 < 9 =  90%
        rand() % 1 < 1  =  100%
        */
        if(likelihood <= 5){
                like_a = 20;
                like_b =  1;
        } else if(likelihood <= 10){
                like_a = 10;
                like_b =  1;
        } else if(likelihood <= 20){
                like_a = 5;
```



Melkor fuzzer

- **Fuzzing rules execution**
  - ... Conjunction with rand() in the iteration through the array of pointers

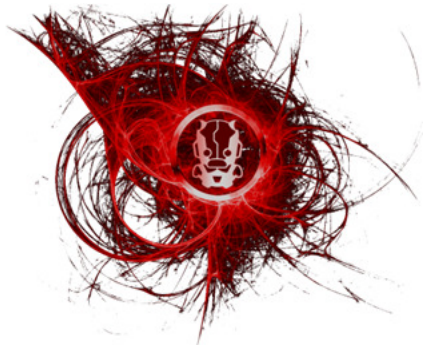
```
void fuzz_sht()
{
        int rule;

        for(sh = 0; sh < orcHDR->e_shnum; sh++, orcSHT++)
                for(rule = 1; rule <= N_RULES_SHT; rule++)
                        if((rand() % like_a) < like_b)
                                if(sht[rule]()){
                                        printf(". ");
                                        debug("SHT[%d] rule [%.2d] executed\n", sh, rule);
                                        fprintf(logfp, " | SHT[%d] rule [%.2d] executed\n",
                                }
}
```



Melkor fuzzer

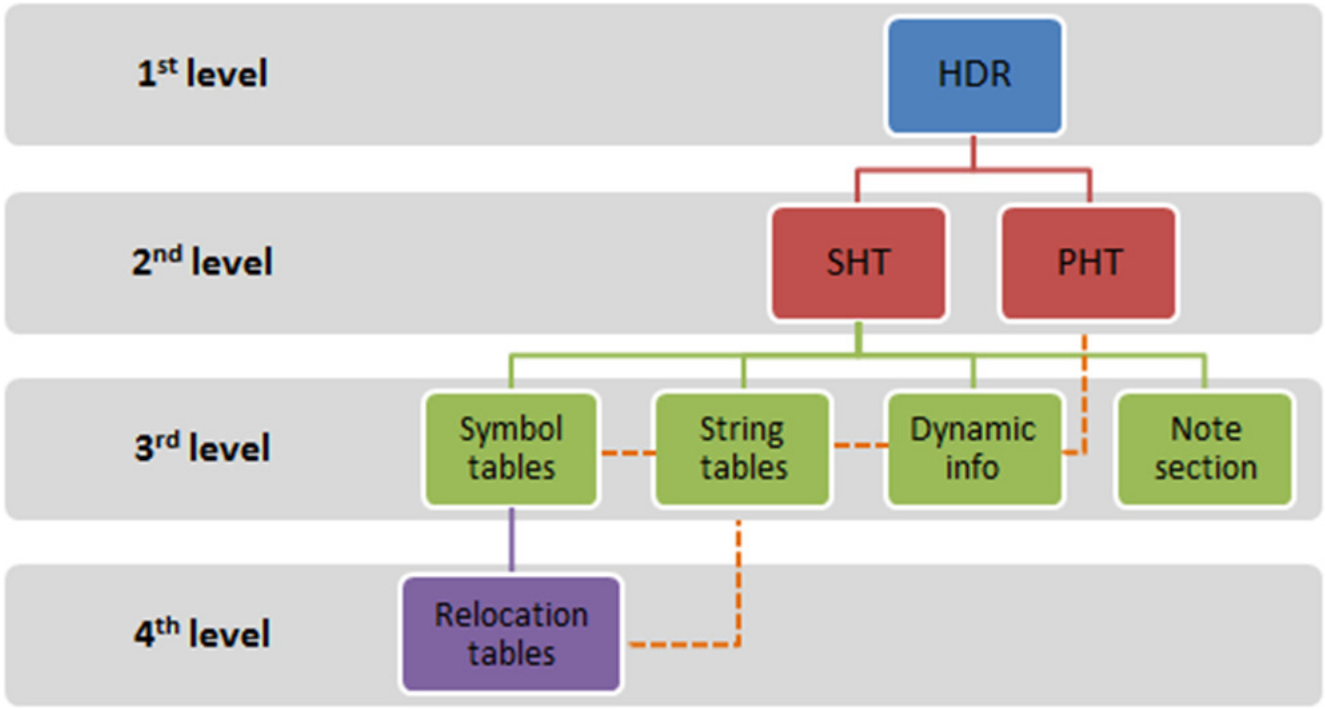
- **Fuzzing rules execution**
  - Some fields are critical and even when the rule is executed, inside the rule function the likelihood is decreased.
  - For example:

```
int pht13(void)
{
        if(rand() % 2) // p_type is a critical field
                return 0;
```

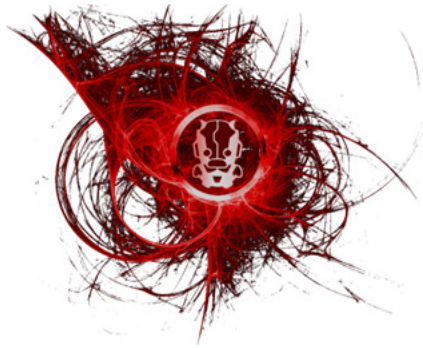


Melkor fuzzer

- **ELF metadata dependencies**



  - These dependencies should not be broken if you want to fuzz deeper levels

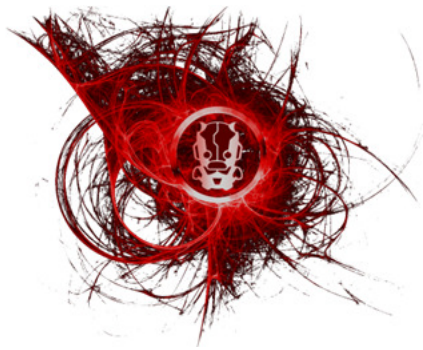


Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF metadata dependencies**
  - Translating them into specific fields:

Level	Metadata	Dependencies	Fuzzable?
1	HDR	-	
2	PHT	e_phoff	N
		e_phnum	N
2	SHT	e_shoff	N
		e_shnum	N
		e_shstrndx	Y (33.33% chance)
3	STRS	e_shstrndx	N
		sh_type (SHT_STRTAB)	N
		sh_offset	N
		sh_name	Y (33.33% chance)
		sh_size	Y (50% chance)
3	NOTE	e_shstrndx	N
		p_type (PT_NOTE)	N
		p_offset	N
		p_vaddr	N
		p_filesz	N
		sh_type (SHT_NOTE)	N
		sh_offset	N
		sh_name	Y (33.33% chance)
		sh_size	Y (50% chance)

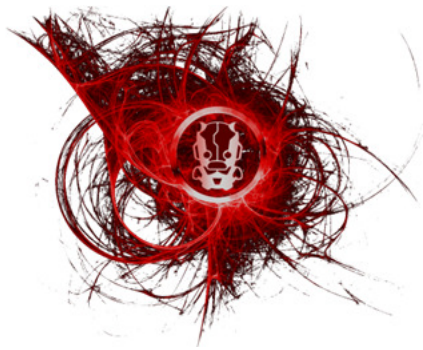


Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF metadata dependencies**
  - Translating them into specific fields:

3	DYN	e_shstrndx	N
		p_type (PT_DYNAMIC)	N
		p_offset	N
		p_vaddr	N
		p_filesz	N
		sh_type (SHT_DYNAMIC)	N
		sh_offset	N
		sh_name	Y (33.33% chance)
		sh_size	Y (25% chance)
		sh_entsize	Y (25% chance)
		sh_link	Y (50% chance)
3	SYM	e_shstrndx	N
		sh_type (SHT_SYMTAB && SHT_DYNSYM)	N
		sh_offset	N
		sh_name	Y (33.33% chance)
		sh_size	Y (25% chance)
		sh_entsize	Y (25% chance)
		sh_link	Y (50% chance)
		sh_info	Y (50% chance)

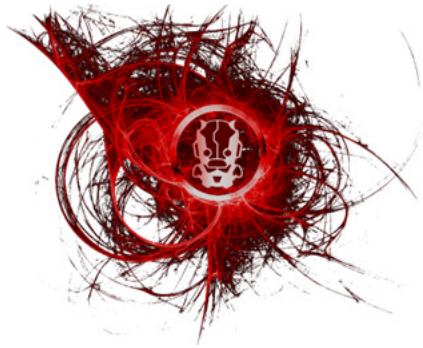


Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF metadata dependencies**
  - Translating them into specific fields:

4	REL	e_shstrndx	N
		sh_type (SHT_REL && SHT_RELA)	N
		sh_offset	N
		sh_type (SHT_SYMTAB && SHT_DYNSYM)	N
		sh_offset (SHT_SYMTAB && SHT_DYNSYM)	N
		sh_link	N
		sh_link (SHT_SYMTAB && SHT_DYNSYM)	Y (33.33% chance)
		sh_offset (SHT_STRTAB)	Y (33.33% chance)
		sh_name	Y (33.33% chance)
		sh_size	Y (25% chance)
		sh_entsize	Y (25% chance)
		sh_info	Y (50% chance)
		SYM (st_info, st_shndx, st_name)	Y (50% chance)



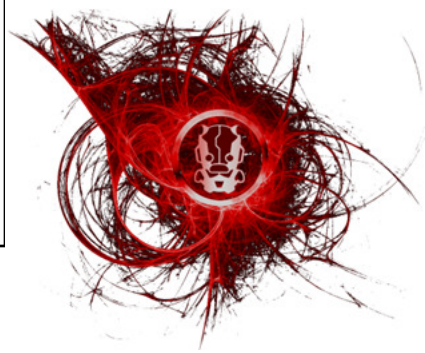
Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **ELF metadata dependencies**
  - And at code level (Example 1):

Level	Metadata	Dependencies	Fuzzable?
1	HDR	-	
2	PHT	e_phoff	N
		e_phnum	N
2	SHT	e_shoff	N
		e_shnum	N
		e_shstrndx	Y (33.33% chance)

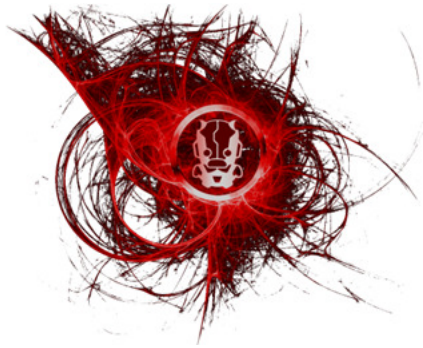
```
int hdr14(void)
{
        if(mode & SHT)
                return 0;

        orcHDR->e_shoff     = getElf_Off();
        orcHDR->e_shnum     = getElf_Half();
        orcHDR->e_shentsize = getElf_Half();

        fprintf(logfp, "(HDR->e_shoff = 0x"HEX",", orcHDR->e_shoff);
        fprintf(logfp, " e_shnum = 0x%x,", orcHDR->e_shnum);
        fprintf(logfp, " e_shentsize = 0x%x)", orcHDR->e_shentsize);

        return 1;
}
```



Melkor fuzzer

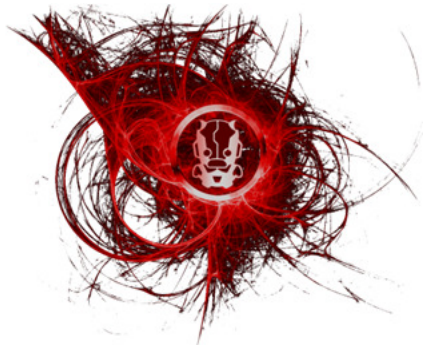
- **ELF metadata dependencies**
  - And at code level (Example 2):

```
int sht4(void)
{
        // Metadata dependencies
        switch(orcSHT->sh_type){
                case SHT_STRTAB:
                        if(mode & STRS)
                                if(rand() % 2) // 50% chance
                                        return 0;
                        break;
                case SHT_NOTE:
                        if(mode & NOTE)
                                if(rand() % 2)
                                        return 0;
                        break;
                case SHT_DYNAMIC:
                        if(mode & DYN)
                                if(rand() % 4 < 3) // 75% chance to return
                                        return 0;
                        break;
                case SHT_SYMTAB:
                case SHT_DYNSYM:
                        if(mode & SYM)
                                if(rand() % 4 < 3)
                                        return 0;
                        break;
                case SHT_RELA:
                case SHT_REL:
                        if(mode & REL)
                                if(rand() % 4 < 3)
                                        return 0;
                        break;
                default:
                        if(rand() % 3 < 2)
                                return 0;
        }

        fuzzSize();

        fprintf(logfp, "(SHT[%d]->sh_size = 0x"HEX")", sh, orcSHT->sh_size);
```



Melkor fuzzer

- **Generators and test data**
  - Semi-valid test data is used in the rules
    - Size fields: common integer bofs values
    - Offsets / addresses: out of bounds values
    - Indexes inside strings: common format strings or non-printable chars
    - Etc.



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Generators and test data**
  - numbers.h

```
// Could be used as short by shifting 16 bits (E.g. int_b0f[0] >> 16 == 0x7fff)
int int_b0f[] = {
        0x7fffffff, // INT_MAX
        0xffffffff, // UINT_MAX (-1 for signed vars)
        0x80000000, // Negative value for signed vars (MSB = 1)
        0xc0000000,
        0xff00ff00,
        0xffff0000,
};

int common_b0f[] = {
        0x41424344, // INC EAX; INC EBX; INC ECX; INC EDX
        0x41414141, // INC EAX * 4
        0x42424242, // INC EBX * 4
        0x43434343, // INC ECX * 4
        0x44444444, // INC EDX * 4
        0x90909090, // NOP
        0xcccccccc, // INT 3
};
```

Melkor fuzzer

- **Generators and test data**
  - numbers.h

```
int int_133t[] = {
        0xB16B00B5, // (  *  ) (  *  )
        0x0DEFACED,
        0XDEADFACE,
        0xCAFED00D,
        0xFEE1DEAD,
        0x0D15EA5E,
        0xDEADC0DE,
        0xBAD0C0DE,
        0xDEFECA7E,
        0xDEFEC8ED,
        0x600DCAFE,
        0x00031337,
};
```

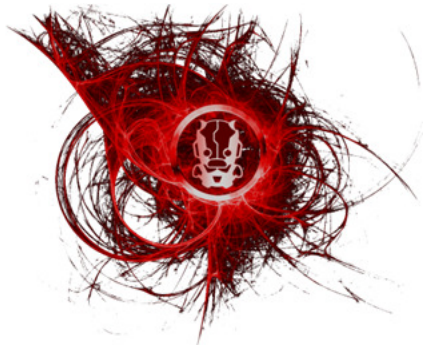
```
short short_133t[] = {
        0xDEAD, 0xBABE,
        0xCAFE, 0xF00D,
        0xBEEF, 0xC0DE,
        0xFACE, 0x0BAD,
        0x1337, 0xD00D,
        0x0FA6, 0xB00B,
};

Elf_Addr key_Addr[] = {
        0x00000000, // Zero page
        0x00400000,
        0x08048000,
        0x40000000, // 1GB
        0x80000000, // 2GB
        0x81000000,
        0xc0000000, // 3GB
        0xc1000000,
        0xd0000000,
};
```

Melkor fuzzer

- **Generators and test data**
  - numbers.h

```
const char *fuzz_paths[] = {
        "orcs_libfoo.so/",
        "./../../../...",
        "/;;/;//..",
        "\\//\\//",
        "~/~~~//",
        "*/!/*",
        "$xX",
        ":::",
        "/#)",
        "$$",
        "//",
        "\\",
        "~",
        "*",
        "/",
};
```

```
const char *fmt_strs[] = {
        " %x ",
        " %n ",
        " %p ",
};
```



Melkor fuzzer

- **Generators and test data**
  - generators.c
    - Functions to return test data based on
      - numbers.h
      - rand()

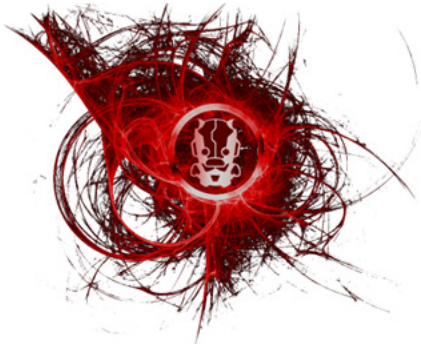
```
Elf_Addr getElf_Addr(void)
{
        Elf_Addr a;

        if(rand() % 2)
                // A key base address + 16 bits random offset
                a = (Elf_Addr) (key_Addr[rand() % (sizeof(key_Addr) / sizeof(Elf_Addr))] + (rand() & 0xffff));
        else {
                if(rand() % 2){
                        int r = rand();

                        if(r % 3 == 0)
                                a = (Elf_Addr) int_133t[rand() % (sizeof(int_133t) / sizeof(int))];
                        else if(r % 3 == 1)
                                a = (Elf_Addr) int_b0f[rand() % (sizeof(int_b0f) / sizeof(int))];
                        else
                                a = (Elf_Addr) common_b0f[rand() % (sizeof(common_b0f) / sizeof(int))];
                } else
                        a = (Elf_Addr) rand();
        }

        return a;
}
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Generators and test data**
  - ELF used as template
    - Some provided in templates/

```
nitrous@mictian:~/melkor-v1.0$ make templ
gcc -ggdb -Wall -DDEBUG templates/foo.c -c -o templates/foo.o
gcc -ggdb -Wall -DDEBUG templates/foo.c templates/libfoo.c -o templates/foo
gcc -ggdb -Wall -DDEBUG templates/foo.c templates/libfoo.c -Wl,-z,relro,-z,now -o templates/foo_full_relro
gcc -ggdb -Wall -DDEBUG templates/foo.c templates/libfoo.c -fstack-protector -z execstack -o templates/foo_s
gcc -ggdb -Wall -DDEBUG templates/foo.c templates/libfoo.c -static -o templates/foo_static
gcc -ggdb -Wall -DDEBUG templates/libfoo.c -c -fPIC -o templates/libfoo.o
gcc -ggdb -Wall -DDEBUG templates/libfoo.o -shared -o templates/libfoo.so
gcc -ggdb -Wall -DDEBUG templates/foo_libfoo.c -L templates -lfoo -o templates/foo_libfoo
gcc -ggdb -Wall -DDEBUG templates/foo_dlopen.c -ldl -o templates/foo_dlopen
gcc -ggdb -Wall -DDEBUG templates/foo_dl_iterate_phdr.c -L templates -lfoo -o templates/foo_dl_iterate_phdr
nitrous@mictian:~/melkor-v1.0$
```

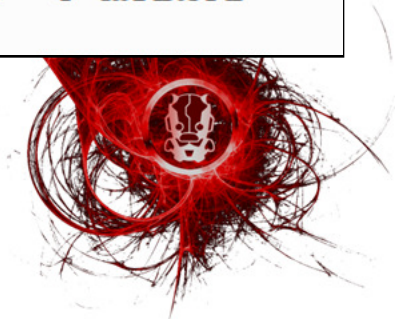


Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Compilation**
  - With a simple $make

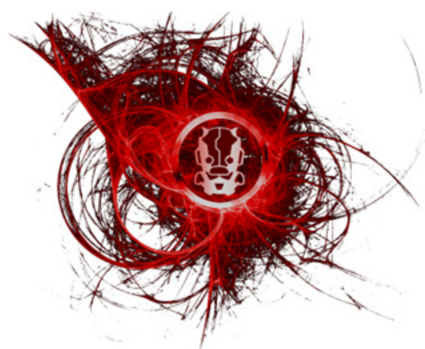
```
gcc -ggdb -Wall -DDEBUG src/print_envp_vars.c -o src/print_envp_vars
gcc -ggdb -Wall -DDEBUG src/env1.c src/generators.c -o src/env1
gcc -ggdb -Wall -DDEBUG src/env2.c src/generators.c -o src/env2
gcc -ggdb -Wall -DDEBUG src/env3.c src/generators.c -o src/env3
gcc -ggdb -Wall -DDEBUG   -c -o src/melkor.o src/melkor.c
gcc -ggdb -Wall -DDEBUG   -c -o src/logger.o src/logger.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_hdr.o src/fuzz_hdr.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_sht.o src/fuzz_sht.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_pht.o src/fuzz_pht.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_sym.o src/fuzz_sym.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_dyn.o src/fuzz_dyn.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_rel.o src/fuzz_rel.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_note.o src/fuzz_note.c
gcc -ggdb -Wall -DDEBUG   -c -o src/fuzz_strs.o src/fuzz_strs.c
gcc -ggdb -Wall -DDEBUG   -c -o src/generators.o src/generators.c
gcc -ggdb -Wall -DDEBUG src/melkor.o src/logger.o src/fuzz_hdr.o src/fuzz_sht.o
.o src/fuzz_rel.o src/fuzz_note.o src/fuzz_strs.o src/generators.o -o melkor
nitrous@mictian:~/melkor-v1.0$
```



Melkor fuzzer

- **Usage**

```
nitrous@mictian:~/melkor-v1.0$ ./melkor
 |
 | M e l k o R v1.0 - An ELF File Format Fuzzer
 | by nitr0us
 |
                        '.       ;'       ';       .'
                         ;l,      x,     lc      ;c'
                           ,x;     k;   ol     :o.
                             dk.   ;0. :0.   ,k:
                             .0k   dO.,;0:  .Ok
                             .0O. cO: c dO. cOx
                             .O0ocOo  l .kk;xOd
                             .00000;  .  ;0000x
                             .cxkkkx:xkolkkkxo.
                               'kkkkkxxxxxx'
                               .xxxxxxxxxxxxx.
                            .;;:xxxxxxxxxxxxo;;;.
                              ..:dxxdo.ddddd:..
                                 ddl:. .ldd:
                                ldd.    .ddl
                                .cdlc. .ldc.
                                  :o.   .o:
                                   ;     ;

              .-.  .              .    _              .-.
    .-.         /|/|             /      /  _          (_) )-.     .-.
    `-'        / `   |    .-.       /      /-./   .-._.     /   \     `-'
   .-.        /    |  ./--'_  /      /   ) (   )      /     )   .-.
   `-'   .-' /     |  (__.'   /_.-  _/     \  `-'      .-/  `--'    `-'
        (__.'      `.                               (_/     `.
                     `._)                                    `._)
```

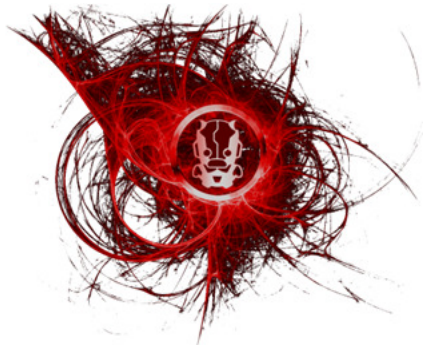


Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Fuzzing options

```
./melkor <ELF metadata to fuzz> <ELF file template> [-n num -l likelihood -q]
 <ELF metadata to fuzz>:
         -a  Autodetect (fuzz according to e_type except -H [the header])
         -H  ELF header
         -S  Section Header Table
         -P  Program Header Table
         -D  Dynamic section
         -s  Symbols Table(s)
         -R  Relocations Table(s)
         -N  Notes section
         -Z  Strings Tables
         -A  All of the above (except -a [Autodetect])
         -B  All of the above (except -a [Autodetect] and -H [ELF Header])
 -n  Number of new fuzzed ELF files (orcs) to create (default: 5000)
 -l  Likelihood (given in % from 1-100) of the execution of each fuzzing rule (default: 10%)
 -q  Quiet mode (doesn't print to STDOUT every executed fuzzing rule)
```



Melkor fuzzer

- **Usage**
  - A simple run (testing preparation):

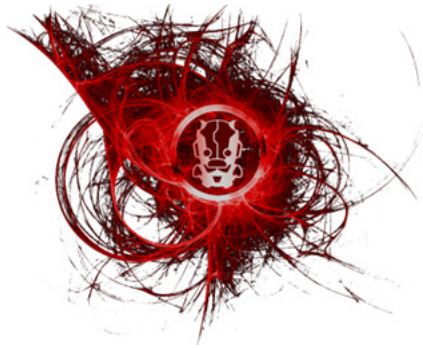
```
nitrous@mictian:~/melkor-v1.0$ ./melkor -a templates/foo -n 1337
          ..-.--..
        ,',',.-`.-.`.
       :.',;'     `.\.
       ||//----,-.--\|    <<--- templates/foo
     \`:|/-----`-'--||'/
      \\|:  <x>  <X>|:'
       `||    " \   |!  _____________________________________
       |!|          ;| / I'll be corrupted  1337 times ! \
       !||:.   --  /|! \_____________________________________/
      /||!||:.___.|!||\
     /|!!!!!|    |!||!\\:.
  ,'//!||!||!`._.||!||,:\\\
: :: |!!!!!|    |!||! |!::
| !! !||!||`---!|!|| ||!|

[+] Automatic mode
[+] ELF type detected: ET_EXEC
[+] Selecting the metadata to fuzz

[+] Detailed log for this session: 'orcs_foo/Report_foo.txt'

[+] The Likelihood of execution of each rule is: Aprox. 10 % (rand() % 10 < 1)

[+] Press any key to start the fuzzing process...
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Malforming ELFs:

```
[+] Fuzzing the Dynamic section .dynamic with 26 entries
. DYN[22] rule [02] executed
. DYN[22] rule [04] executed

[+] Fuzzing the Note section .note.ABI-tag with 32 bytes

[+] Fuzzing the Note section .note.gnu.build-id with 36 bytes
. NOTE[3] rule [03] executed

[+] Fuzzing the String Table .dynstr with 135 bytes

[+] Fuzzing the String Table .shstrtab with 345 bytes
. STRS[35] rule [02] executed

[+] Fuzzing the String Table .strtab with 762 bytes
. STRS[37] rule [01] executed

[+] Fuzzing the Section Header Table with 38 entries
. SHT[0] rule [32] executed
. SHT[1] rule [03] executed
. SHT[1] rule [38] executed
. SHT[6] rule [05] executed
. SHT[6] rule [10] executed
. SHT[6] rule [22] executed
. SHT[6] rule [36] executed
. SHT[7] rule [38] executed
. SHT[8] rule [05] executed
```



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Malformed ELFs (orcs):

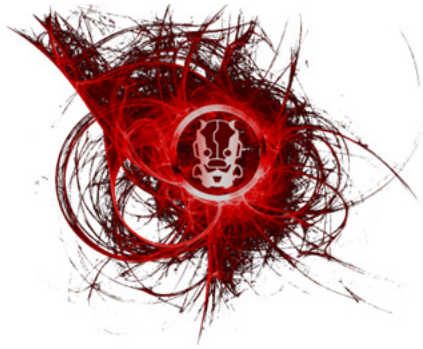
```
nitrous@mictian:~/melkor-v1.0$ md5sum orcs_foo/orc_* | head -15
97d57d21aee9f3adf58611da9e95536e  orcs_foo/orc_0001
089ddca704e7d0c33128e835d2f05dbb  orcs_foo/orc_0002
79428765c2a7acbfe8147a10dcdb5aae  orcs_foo/orc_0003
b73b78a7f4584ce2d291093da93725e3  orcs_foo/orc_0004
f7a3377dba6f797607173a537eb9c989  orcs_foo/orc_0005
b38ad076f89874ebd64744204445b7e1  orcs_foo/orc_0006
e5ace2451a9c706e6dd93fda4efeeef2  orcs_foo/orc_0007
17087172f53f6c4fdd60bb46c3422c60  orcs_foo/orc_0008
8273d618edafd3b10bcb7b331c8c1142  orcs_foo/orc_0009
9b1ddcf4c5fc1aa86898afae7380c726  orcs_foo/orc_0010
f5c244145fc3a6c38f3f50218af6ea68  orcs_foo/orc_0011
5af9da0e1af5bd36bd3a2b862874a318  orcs_foo/orc_0012
e81403344eecea64070aa65f452d013e  orcs_foo/orc_0013
c92b7a329e2e89d1563c70c0bd4a82e4  orcs_foo/orc_0014
ead4f174c1f1a9dc3d09de2c40c91fd7  orcs_foo/orc_0015
nitrous@mictian:~/melkor-v1.0$ md5sum orcs_foo/orc_* | wc -l
1337
nitrous@mictian:~/melkor-v1.0$ md5sum orcs_foo/orc_* | uniq | wc -l
1337
nitrous@mictian:~/melkor-v1.0$
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Malformed ELFs (Default 10%):

```
nitrous@mictian:~/melkor-v1.0$ readelf -SW orcs_foo/orc_112
There are 38 section headers, starting at offset 0x1d78:

Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            0000000000000000 000000 000000 00      0   0  0
  [ 1] .interp           LOUSER+fffffff  0000000000400200 000200 007fff 00   A  0   0 6921411395954937992
  [ 2] .note.ABI-tag     NOTE            000000000040021c 00021c 00d2be 00   A  0   0  4
  [ 3] .note.gnu.build-id NOTE            000000000040023c 00023c 000024 00   A  0   0  4
  [ 4] .hash             HASH            0000000000400260 000260 deadface64795ff3 00  AT 47806 1111638594
  [ 5] .gnu.hash         GNU_HASH        00000000004002b8 0002b8 000024 00   A  6   0  8
  [ 6] .dynsym           DYNSYM          00000000004002e0 0002e0 000198 18   A  7   1 18446462600691036503
  [ 7] .dynstr           STRTAB          0000000000400478 000478 000087 00   A  0   0  1
  [ 8] .gnu.version      VERSYM          0000000000400500 000500 000022 02   A  6   0  2
  [ 9] .gnu.version_r    VERNEED         0000000000400528 000528 000020 00   A  7   1  8
  [10] .rela.dyn         RELA            0000000000400548 000548 000030 18   A  6   0 1068708530351182647
  [11] .rela.plt         RELA            0000000000400578 000578 000150 18   A  6  13 5429305985745437508
  [12] <corrupt>         SYMTAB SECTION INDICIES 00000000004006c8 0006c8 000018 00 WAXxMxxxolp  0   0  4
  [13]                   PROGBITS        00000000004006e0 0006e0 0000f0 10  AX  0   0 4097
  [14] .text             HASH            00000000004007d0 0007d0 000388 00  AX  0   0 16
  [15] .fini             PROGBITS        0000000000400b58 000b58 00000e 00  AX  0   0  4
```



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Malformed ELFs (Aggressive 70%):

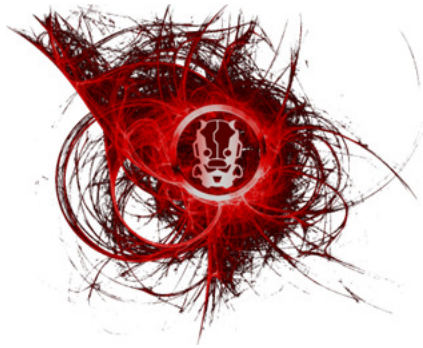
```
nitrous@mictian:~/melkor-v1.0$ readelf -SW orcs_foo/orc_118
There are 38 section headers, starting at offset 0x1d78:
readelf: Error: Unable to read in 0xcb26 bytes of symbols
readelf: Error: Section 10 has invalid sh_entsize ffff (expected 18)
readelf: Error: Section 11 has invalid sh_entsize 17 (expected 18)
readelf: Error: Section 36 has invalid sh_entsize 17 (expected 18)

Section Headers:
  [Nr] Name              Type            Address          Off    Size   ES Flg Lk Inf Al
  [ 0] .pltÃ.dta.bs %x omíentÑ.debug_a¿anges NULL            00000000054d31fe 81006acb ffff0000cafed00d 444444440defa
95
  [ 1] .it %n .noÓe.ABI-tag PROGBITS        0000000000400200 000200 00001c 00 WXTolp  0   0 4097
  [ 2]                   NOTE            0000000000408527 00021c 000020 00  AT  0   0 1449838155202951886
  [ 3] .g %n goè.pltÃ.dta.bs %x omíentÑ.debug_a¿anges NOTE            000000000040023c 00023c 000024 ff00 WAIOGxxxop
  [ 4] n ext             GNU_HASH        0000000000400260 18881556 000058 04  Ao  6   0 4097
  [ 5] .gnu.ªash.d %n m GNU_HASH        00000000553b07a7 0002b8 defec8ed90909090 00 XxMILxxxxxolp  6   0 4095
  [ 6] .d %n m           DYNSYM          ffffffff80000000 0002e0 00cb26 18 MLxxolp  7   1  8
  [ 7] .dta.bs %x omíentÑ.debug_a¿anges STRTAB          000000007be4a715 000478 000000 c753   A  0   0  1
  [ 8] .gnu.version      VERSYM          00000000c10072db 000500 000000 4630  Ao  6   0 14757395257081045377
  [ 9]                   VERNEED         000000005a1c1d0d 000528 000020 00 XxMILxxxxxolp  7   1 805679780492921069
  [10] ¶rela.d %n rea.plt RELA            ffffffffdeadc0de 000548 000030 18   o  6   0 16045756813802392190
  [11]                   RELA            0000000000400578 000578 000151 18  Ao  6  13 16068503166153441502
  [12] .ini %n ext       <unknown>: c    00000000004006c8 0006c8 000018 00 XxMILxxolp  0   0 4774451406420961886
  [13] .dta.bs %x omíentÑ.debug_a¿anges FILTER          00000000004006e0 0006e0 000000 10 WXTlp  0   0 98738103721065
  [14]                   GNU_LIBLIST     00000000287f1fc8 0007d0 00cafe 00   o  0   0 3735873939742654463
  [15]  goè.pltÃ.dta.bs %x omíentÑ.debug_a¿anges X86_64_UNWIND   0000000000400b58 000b58 00000e 00 OGTxxxxxolp  0   0
  [16] .rodçta %n _fram %x r.eh_¨rame.ctors PROGBITS        0000000000400b68 000b68 000001 ffffffffffffffff XxILTxxxx
  [17] u.õu %n id.gnu.ªash.d %n m PROGBITS        0000000000400bac 000bac 00002d ffffffffffffffff  AT  0   0 48467915
  [18] .bs %x omíentÑ.debug_a¿anges PROGBITS        0000000000400bd8 000bd8 00009c 00  Ao  0   0  8
  [19] .ctors            LOOS+ffffff5    ffffffff80000000 031337 000010 00  Wo  0   0  8
  [20] . %n s            GNU_LIBLIST     000000003e7929d3 000c88 000011 ffffffffffffffff xxxxolp  0   0 4097
  [21] .jc».dynaêic     LOOS+ffffff5    0000000044444444 43434343 000008 c0000000ffffffff WIOxxop  0   0  8
  [22] n ext             DYNAMIC         0000000000600ca0 000ca0 bad0c0de0d15ea5f ffffffffffffffff   o 34 1111638594
96
  [23] .g %n goè.pltÃ.dta.bs %x omíentÑ.debug_a¿anges INIT_ARRAY      0000000000600e40 000e40 000000 08   A  0   0 16
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Testing the malformed ELFs (**test_fuzzed.sh**)

```
nitrous@mictian:~/melkor-v1.0$ ./test_fuzzed.sh
Usage: ./test_fuzzed.sh [options] <dir_with_malformed_ELFs_aka_orcs> [program_and_parameters] > OUT.txt
        [options]:
                -B Fuzz $LD_BIND_NOW     (fuzzing rule env1)
                -L Fuzz $LD_LIBRARY_PATH (fuzzing rule env2)
                -P Fuzz $LD_PRELOAD      (fuzzing rule env3)

        [program_and_parameters]:
                If no program given, the malformed ELF will be executed ./orc (OS' ELF loader testing)

Examples:
./test_fuzzed.sh orcs_foo/
./test_fuzzed.sh -BL orcs_foo/ > EXECUTION_FOO_WITH_FUZZED_LD.txt
./test_fuzzed.sh orcs_libfoo.so/ "readelf -SW"
./test_fuzzed.sh -P orcs_foo/
./test_fuzzed.sh orcs_foo_standalone/ "readelf -S" > READELF_FUZZING_RESULTS.txt
./test_fuzzed.sh orcs_foo.o/ "gcc -o foo"
./test_fuzzed.sh -L orcs_foo_static/ "gdb -q"
```



Melkor fuzzer

- **Usage**
  - No, Melkor will not send your orcs to the shopping mall

http://mashable.com/2014/06/12/orc-mall-prank/

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - OS kernel / dynamic loader testing:

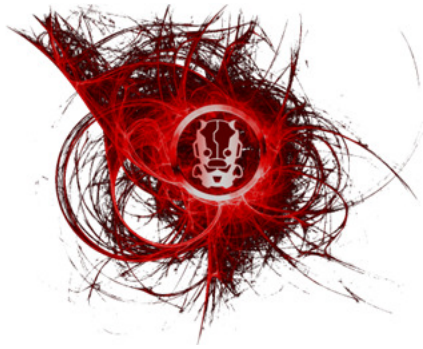
```
nitrous@mictian:~/melkor-v1.0$ ./test_fuzzed.sh -BP orcs_foo/

==================================================

Exporting fuzzed $LD_BIND_NOW ... DONE
Exporting fuzzed $LD_PRELOAD ... DONE

==================================================

LD_* vars in environment:

$LD_BIND_NOW=VFDLSTWTMNNGBWLKXJHAHOOXBCJKPRUOWBCPUALHPCNTYBGWLNASCQSGSCSKTOYSRELMGYTXE
BJWBMCFDTRSFCMCHHMNDSHERWUDOADADMAGBENGYFYGJMLTVXHBSQFMODQFFVFKKHQLMGRLNST

$LD_PRELOAD=;/;;/orc_0001.so.so:../orcs_libfoo.so/orc_0053.so:orcs_libfoo.so/orc_0060.
orc_0028.so:./..../orc_0038.so:orcs_libfoo.so/orc_0002.so:orcs_libfoo.so/orc_0023.so:.
.../orc_0006.so:\orc_0056.so:\orc_0082.so:./..../orc_0100.so:../orcs_libfoo.so/orc_000
rc_0036.so:\orc_0020.so:orcs_libfoo.so/orc_0084.so:./..../orc_0094.so:\orc_0044.so:;/;
_libfoo.so/orc_0040.so:$/orc_0014.so:orcs_libfoo.so/orc_0013.so:~~//orc_0076.so:orcs_l
so:\orc_0072.so.so:./..../orc_0072.so:$/orc_0092.so:/.so/orc_0047.so:orcs_libfoo.so/or
o:orcs_libfoo.so/orc_0065.so:../orcs_libfoo.so/orc_0045.so:orcs_libfoo.so/orc_0053.so:
../orc_0049.so.so:;/;;/orc_0074.so:orcs_libfoo.so/orc_0073.so:orcs_libfoo.so/orc_0022.

==================================================

Press any key to start the testing...
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - OS kernel / dynamic loader testing:

```
nitr0us@exiled:~/melkor-v1.0$ sudo dmesg | grep orc_ | tail -50
[  565.462210] orc_548[2806]: segfault at 600e28 ip 00007f22d775910a sp 00007fffbd707860 error 4 in ld-2.17.so[7f22d7757000+21000]
[  568.899192] orc_597[2890]: segfault at 601058 ip 0000000000400790 sp 00007fffd52b80b8 error 4 in orc_597[400000+1000]
[  569.141578] orc_600[2896]: segfault at 0 ip           (null) sp 00007fff6b585828 error 14 in orc_600[400000+1000]
[  569.346299] orc_604[2901]: segfault at 600e28 ip 00007f4fe607210a sp 00007fff62bc5790 error 4 in ld-2.17.so[7f4fe6070000+21000]
[  569.562912] orc_605[2903]: segfault at 800e28 ip 00007f6f74dc310a sp 00007fff1ec77970 error 4 in ld-2.17.so[7f6f74dc1000+21000]
[  569.746875] orc_606[2905]: segfault at 800e28 ip 00007f688d84f10a sp 00007fff1e5f3850 error 4 in ld-2.17.so[7f688d84d000+21000]
[  569.947429] orc_610[2911]: segfault at 410040 ip 00007ff6ad0effa0 sp 00007fffede3c420 error 4 in ld-2.17.so[7ff6ad0ee000+21000]
[  570.154591] orc_614[2916]: segfault at 600e28 ip 00007fb8cfeec10a sp 00007fff60824d50 error 4 in ld-2.17.so[7fb8cfeea000+21000]
[  570.361689] orc_617[2920]: segfault at 400800 ip 0000000000400800 sp 00007fff66953000 error 14 in orc_617[600000+2000]
[  570.572409] orc_618[2922]: segfault at 3ff468 ip 00007fed47d53860 sp 00007fffe494c5e8 error 4 in ld-2.17.so[7fed47d3c000+21000]
[  570.793673] orc_621[2927]: segfault at 1ff4c8 ip 00007fa54a51f7c0 sp 00007fffbf267e68 error 4 in ld-2.17.so[7fa54a515000+21000]
[  574.308832] orc_671[3003]: segfault at 800e28 ip 00007f944ddd210a sp 00007fff5c210310 error 4 in ld-2.17.so[7f944ddd0000+21000]
[  574.482102] orc_677[3010]: segfault at 601058 ip 0000000000400790 sp 00007fffc97489e8 error 4 in orc_677[400000+1000]
[  574.650128] orc_679[3013]: segfault at 800e28 ip 00007f581c39310a sp 00007fff08830010 error 4 in ld-2.17.so[7f581c391000+21000]
[  574.819186] orc_680[3016]: segfault at 600e28 ip 00007fa2c8c7c10a sp 00007fffa51ce6c0 error 4 in ld-2.17.so[7fa2c8c7a000+21000]
[  574.993202] orc_683[3022]: segfault at 40400040 ip 00007f550da6efa0 sp 00007fff83586920 error 4 in ld-2.17.so[7f550da6d000+21000]
[  575.157605] orc_684[3024]: segfault at 0 ip           (null) sp 00007fffe6d56458 error 14 in orc_684[400000+1000]
[  575.371189] orc_698[3042]: segfault at 800e28 ip 00007fa6422b910a sp 00007fff393e5150 error 4 in ld-2.17.so[7fa6422b7000+21000]
[  575.564195] orc_7[3045]: segfault at 400800 ip 0000000000400800 sp 00007ffff1a964b0 error 14 in orc_7[600000+2000]
[  575.729089] orc_700[3048]: segfault at 800e28 ip 00007fe41b1d910a sp 00007fff30229260 error 4 in ld-2.17.so[7fe41b1d7000+21000]
[  575.892790] orc_701[3050]: segfault at 600e28 ip 00007f0f7097210a sp 00007fffae8fd4f0 error 4 in ld-2.17.so[7f0f70970000+21000]
[  579.485929] orc_79[3169]: segfault at 800e28 ip 00007f7edc01010a sp 00007fff8bb03310 error 4 in ld-2.17.so[7f7edc00e000+21000]
[  579.683801] orc_792[3173]: segfault at 800e28 ip 00007ffc7196110a sp 00007fff4bdefb40 error 4 in ld-2.17.so[7ffc7195f000+21000]
```



Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Application testing (Example: dumpelf)

```
nitr0us@exiled:~/melkor-v1.0$ ./test_fuzzed.sh orcs_foo/ dumpelf | egrep "Tes

Testing program: dumpelf orcs_foo/orc_1
Testing program: dumpelf orcs_foo/orc_10
Testing program: dumpelf orcs_foo/orc_100
Testing program: dumpelf orcs_foo/orc_1000
        .sh_addralign = 4 Segmentation fault (core dumped)
Testing program: dumpelf orcs_foo/orc_101
        .sh_type      = 1           , /* [SHTSegmentation fault (core dumped)
Testing program: dumpelf orcs_foo/orc_102
Testing program: dumpelf orcs_foo/orc_103
Testing program: dumpelf orcs_foo/orc_104
Testing program: dumpelf orcs_foo/orc_105
        .sSegmentation fault (core dumped)
Testing program: dumpelf orcs_foo/orc_106
        .sh_addralign = 4     Segmentation fault (core dumped)
Testing program: dumpelf orcs_foo/orc_107
Testing program: dumpelf orcs_foo/orc_108
Testing program: dumpelf orcs_foo/orc_109
Testing program: dumpelf orcs_foo/orc_11
Segmentation fault (core dumped)
Testing program: dumpelf orcs_foo/orc_110
Testing program: dumpelf orcs_foo/orc_111
```

Melkor fuzzer

- **Usage**
  - Application testing (Example: gcc)

```
------------------------------------------------
Testing program: gcc -o test orcs_foo.o/orc_0008.o
/usr/bin/ld: orcs_foo.o/orc_0008.o: invalid string offset 3405697037 >= 77 for section `.strtab
/usr/bin/ld: BFD (GNU Binutils for Debian) 2.20.1-system.20100303 internal error, aborting at .
fd/reloc.c line 5558 in bfd_generic_get_relocated_section_contents

/usr/bin/ld: Please report this bug.

collect2: ld returned 1 exit status
------------------------------------------------
Testing program: gcc -o test orcs_foo.o/orc_0009.o
/usr/bin/ld: Dwarf Error: Offset (401826251) greater than or equal to .debug_str size (586).
orcs_foo.o/orc_0009.o: In function `main':
/home/nitrous/melkor-v1.0/templates/foo.c:14: relocation truncated to fit: R_X86_64_32 against
ta'
/home/nitrous/melkor-v1.0/templates/foo.c:24: undefined reference to `help'
/usr/bin/ld: orcs_foo.o/orc_0009.o: access beyond end of merged section (401826251)
collect2: ld returned 1 exit status
------------------------------------------------
Testing program: gcc -o test orcs_foo.o/orc_0010.o
/usr/bin/ld: orcs_foo.o/orc_0010.o: bad reloc symbol index (0xff00ff00 >= 0x1c) for offset 0x16
ection `.debug_info'
orcs_foo.o/orc_0010.o: could not read symbols: Bad value
collect2: ld returned 1 exit status
------------------------------------------------
```

Melkor fuzzer

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

- **Usage**
  - Application testing (Example: gcc)

```
Testing program: gcc -o test orcs_foo.o/orc_0170.o
Testing program: gcc -o test orcs_foo.o/orc_0171.o
collect2: ld terminated with signal 11 [Segmentation fault]
Testing program: gcc -o test orcs_foo.o/orc_0172.o
Testing program: gcc -o test orcs_foo.o/orc_0173.o
Testing program: gcc -o test orcs_foo.o/orc_0174.o
Testing program: gcc -o test orcs_foo.o/orc_0175.o
Testing program: gcc -o test orcs_foo.o/orc_0176.o
Testing program: gcc -o test orcs_foo.o/orc_0177.o
Testing program: gcc -o test orcs_foo.o/orc_0178.o
Testing program: gcc -o test orcs_foo.o/orc_0179.o
collect2: ld terminated with signal 11 [Segmentation fault]
Testing program: gcc -o test orcs_foo.o/orc_0180.o
Testing program: gcc -o test orcs_foo.o/orc_0181.o
Testing program: gcc -o test orcs_foo.o/orc_0182.o
Testing program: gcc -o test orcs_foo.o/orc_0183.o
Testing program: gcc -o test orcs_foo.o/orc_0184.o
Testing program: gcc -o test orcs_foo.o/orc_0185.o
Testing program: gcc -o test orcs_foo.o/orc_0186.o
Testing program: gcc -o test orcs_foo.o/orc_0187.o
Testing program: gcc -o test orcs_foo.o/orc_0188.o
collect2: ld terminated with signal 11 [Segmentation fault]
Testing program: gcc -o test orcs_foo.o/orc_0189.o
```

Melkor fuzzer

- **Logging**
  - A simple logging facility implemented to identify the fuzzed metadata in detail

```
[+] Fuzzing process finished
[+] Orcs (malformed ELFs) saved in 'orcs_foo/'
[+] Detailed fuzzing report: 'orcs_foo/Report_foo.txt'
```

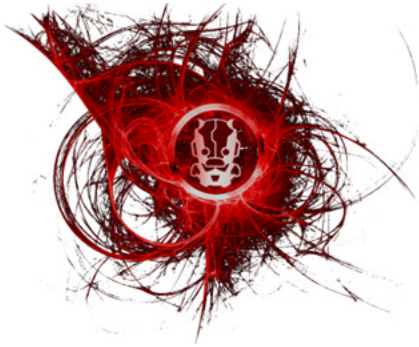
```
 ~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-
| Log report for fuzzed files based on foo                                          |
 ~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-~-

How to read this report:

(Fuzzed Metadata) | Corresponding fuzzing rule (docs/Melkor_Fuzzing_Rules.pdf)
```

```
[+] Fuzzing the String Table .strtab with 762 bytes
(STRS[37]->sh_offset (0x2f20) + 13 = %x , 27 = %n , 47 = %n , 68 = %x , 78 = %n , 100 = %n , 128 = %
 , 194 = %x , 223 = %n , 242 = %n , 253 = %n , 280 = %n , 299 = %x , 317 = %x , 337 = %n , 352 = %x
 440 = %x , 451 = %n , 469 = %x , 488 = %x , 517 = %x , 542 = %n , 552 = %x , 564 = %n , 581 = %x ,
14 = %x , 631 = %x , 633 = %x , 659 = %n , 684 = %n , 701 = %x , 719 = %n , 747 = %x , ) | STRS[37]

[+] Fuzzing the Section Header Table with 38 entries
(SHT[2]->sh_entsize = 0x0000000000000000) | SHT[2] rule [06] executed
(SHT[3]->sh_size = 0x0000000000000025, sh_entsize = 0xffffffffffffffff) | SHT[3] rule [36] executed
(SHT[4]->sh_size = 0x0000000000000059, sh_entsize = 0x0000000000000003) | SHT[4] rule [36] executed
(SHT[7]->sh_addralign = 0x474d0db3ffffffff) | SHT[7] rule [05] executed
(SHT[8]->sh_flags = 0x00000000ff00002) | SHT[8] rule [32] executed
(SHT[9]->sh_flags = 0x00000000ff00002) | SHT[9] rule [32] executed
(SHT[11]->sh_size = 0x0000000000000000) | SHT[11] rule [04] executed
```

Melkor fuzzer

- **Download**

http://www.brainoverflow.org/code/melkor-v1.0.tar.gz

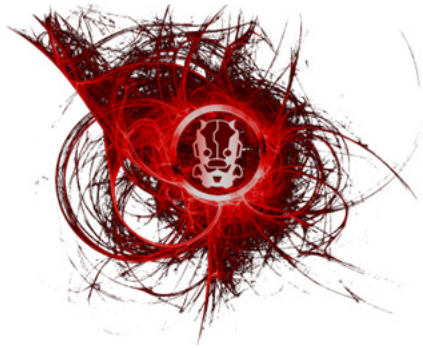


Fuzzing ELF software

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

DEMOS

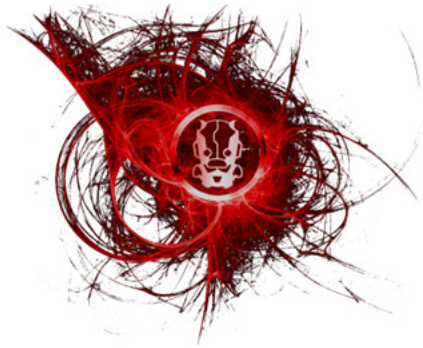
- Homework: **fuzz Melkor fuzzer** ;-)
  - Yes, inception fuzzing
  - Read BUGS.txt
  - It could be used as a test subject



Conclusions

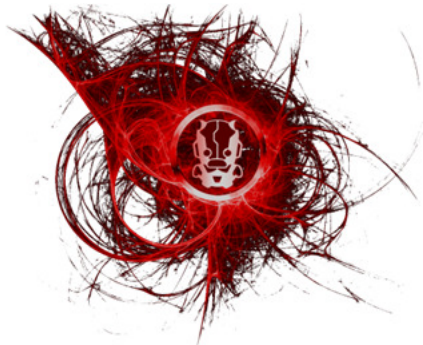
- ELF is just another file format where common parsing mistakes are still used

- ELF parsers are not just in the OS kernels, readelf and objdump. Many new software are parsing and supporting 32 & 64-bit ELF files



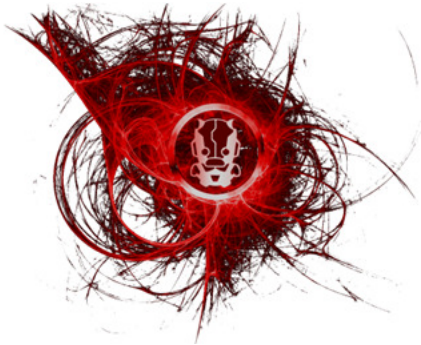
Conclusions

- Fuzzing discover defects that normally are harder to find in less time than manual testing
- Fuzzing is much better having knowledge of the semantics (specifications)
- A single crash could be an exploitable security bug or could be used as an anti-reversing or anti-infection technique



Conclusions

- **Melkor fuzzer** will help you to find functional (and security) bugs in your ELF parsers.



References

[1] blexim. (2002). “Basic Integer Overflows”. PHRACK Magazine, Release 60. Retrieved from http://phrack.org/issues/60/10.htm

[2] DeMott, J., Enbody, R. (Aug 5th, 2006). “The Evolving Art of Fuzzing”. DEFCON 14. Retrieved from https://www.defcon.org/images/defcon-14/dc-14-presentations/DC-14-DeMott.pdf

[3] Dietz, W., Li, P., Regehr, J., and Adve, V. (June 2012). “Understanding Integer Overflow in C/C++”. Proceedings of the 34th International Conference on Software Engineering (ICSE), Zurich, Switzerland. Retrieved from http://www.cs.utah.edu/~regehr/papers/overflow12.pdf

[4] Hernández, A. (Dec 18th, 2012). “Striking Back GDB and IDA debuggers through malformed ELF executables”. Retrieved from http://blog.ioactive.com/2012/12/striking-back-gdb-and-ida-debuggers.html

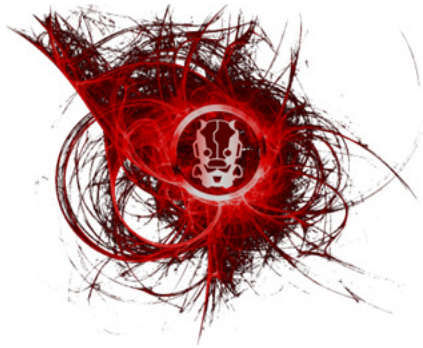
References

[5] Jarkko, L., Rauli, K., and Heikki, K. (2006). "Codenomicon Robustness Testing – Handling the Infinite Space while breaking Software". Retrieved from http://www.codenomicon.com/resources/whitepapers/code-whitepaper-2006-06.pdf

[6] Ormandy, T. "Sophail: A Critical Analysis of Sophos Antivirus". Retrieved from https://lock.cmpxchg8b.com/sophail.pdf

[7] Padilla, O. (May 12th, 2005). "Analyzing Common Binary Parser Mistakes". Uninformed e-zine Vol. 3. Retrieved from http://uninformed.org/index.cgi?v=3&a=1&t=pdf

[8] Miller, C. (Oct 20th, 2007). "Fuzzing with Code Coverage By Example". ToorCon 2007. Retrieved from http://fuzzinginfo.files.wordpress.com/2012/05/cmiller_toorcon2007.pdf



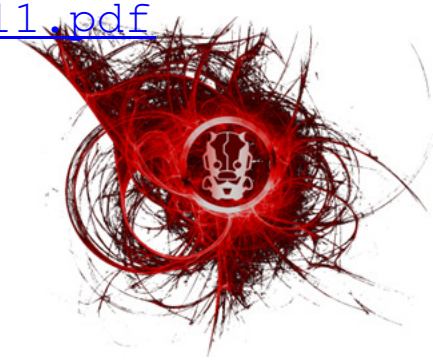
References

[9] Miller, C. (Mar 28th, 2008. "Fuzz By Number: More Data About Fuzzing Than You Ever Wanted To Know". CanSecWest 2008. Retrieved from https://cansecwest.com/csw08/csw08-miller.pdf

[10] TIS Committee. (1995). "Executable and Linking Format (ELF) Specification v 1.2". Retrieved from http://www.uclibc.org/docs/elf.pdf

[11] van Sprundel, I. (Dec 8th, 2005). "Fuzzing: Breaking software in an automated fashion". Retrieved from http://events.ccc.de/congress/2005/fahrplan/attachments/582-paper_fuzzing.pdf

[12] Wysopal, C., Nelson, L., Zovi, D. D., and Dustin, E. (2007). "Local Fault Injection" (Ch. 11) in The Art of Software Security Testing, pp. 201-229. Retrieved from http://www.securityfocus.com/images/infocus/Wysopal_CH11.pdf



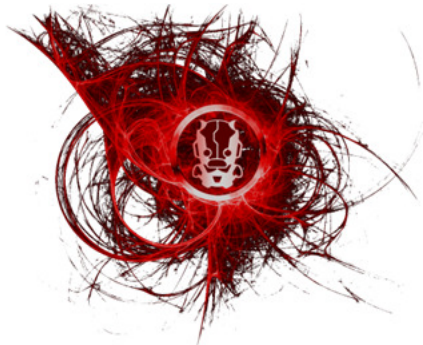
Special Thanks

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

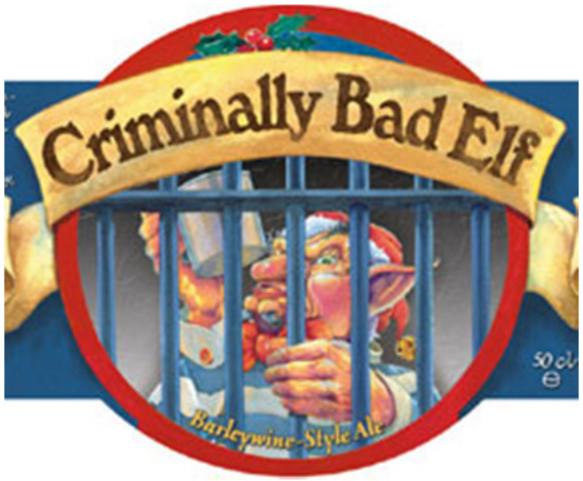
- To all the researchers / coders from whom I've learned cool ELF stuff and fuzzing:

Silvio Cesare, the ELF shell crew, mayhem, scut, the grugq, Itzik Kotler, Electronic Souls, Julien Vanegue, Andrew Griffiths, dex, beck, Brian Raiter, Rakan El-Khalil, Jared DeMott, skape, JunkCode, Sebastian Krahmer, Paul Starzetz, Charlie Miller, Ilja van Sprundel, Chris Rohlf, aczid, Rebecca Shapiro, Sergey Bratus and some others I forgot to mention

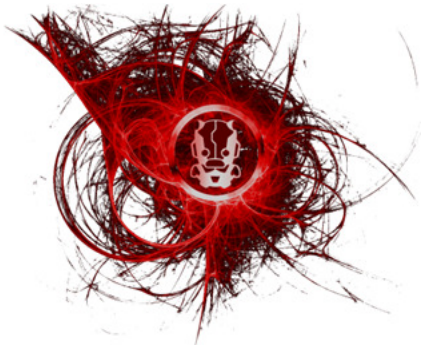
... **Thanks !**



Thanks !



/ Alejandro Hernández
/ @nitr0usmx



In Memorial †

IOActive
COMPREHENSIVE INFORMATION SECURITY SERVICES

Dedicated to the memory of one of my best friends,
Aaron Alba.