H2HC

Hackers to Hackers Conference

R3MF – R3v3rs1ng on Mach-O File

$Whoami

Ricardo L0gan

Security Specialist with over 15 years of experience, malware research enthusiastic, pentest and reverse engineering. I’ve a solid knowledge on topics like network security, hardening and tuning across multiple platforms such as Windows, Linux, OS X and Cisco. Beginner in programming languages as Python, C and Assembly.

In Brazil, I contribute with some security conferences organizations such as SlackShow Community, bSides SP and Hackers to Hackers Conference (H2HC).

Member # RTFM 💀 Co|\|cL/\V€ #

Long live Open Source - Use Linux (Slackware)

R3v3rs1ng on Mach-O File

Agenda

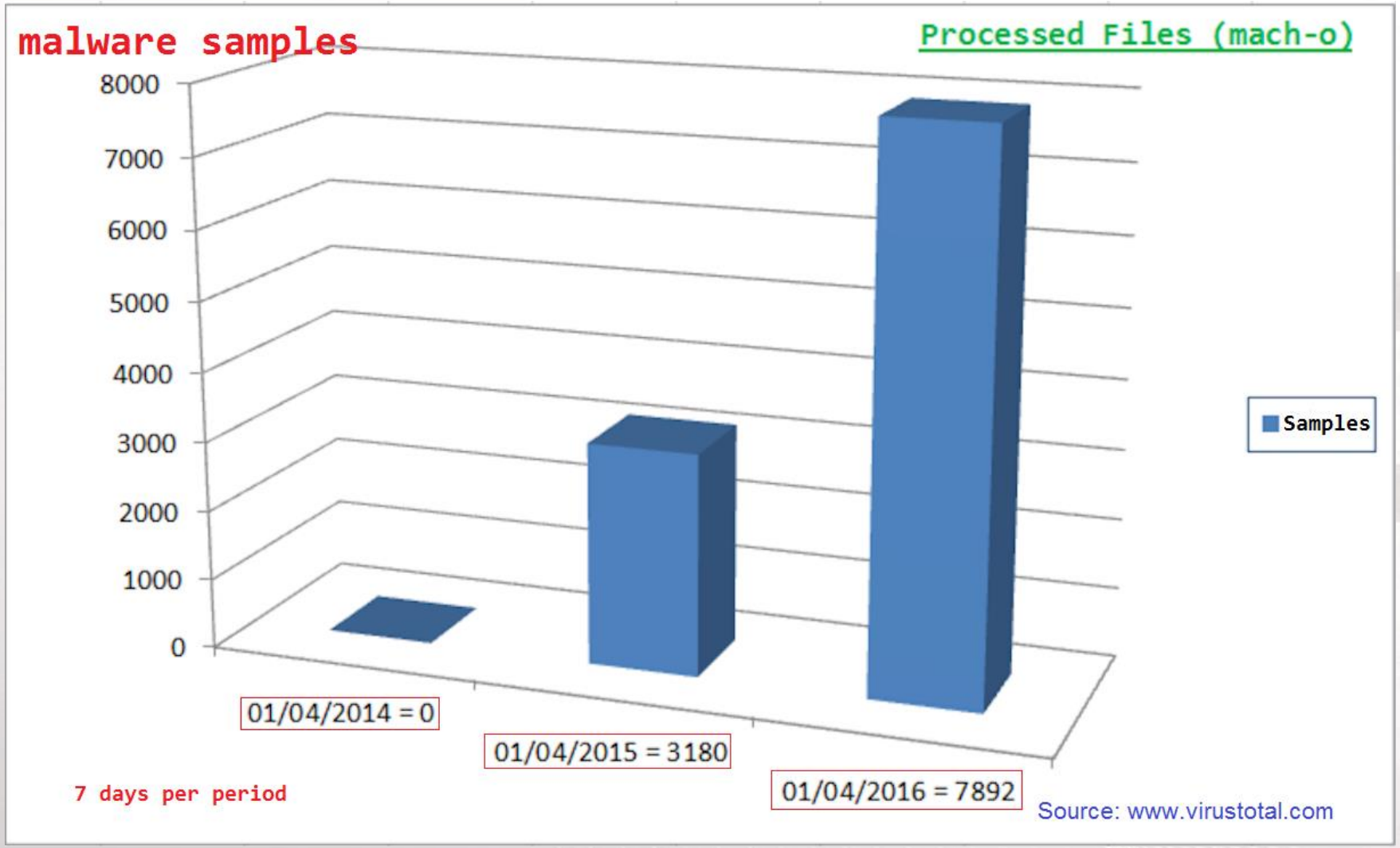
0x00 Motivation of Research
0x01 The Mach-O Format
0x02 Demo I (crackme)
0x03 Tricks for Reversing
0x04 Demo II (malware)
0x05 Conclusions / Q & (MAYBE \0/) A

0x00 Motivation Of Research

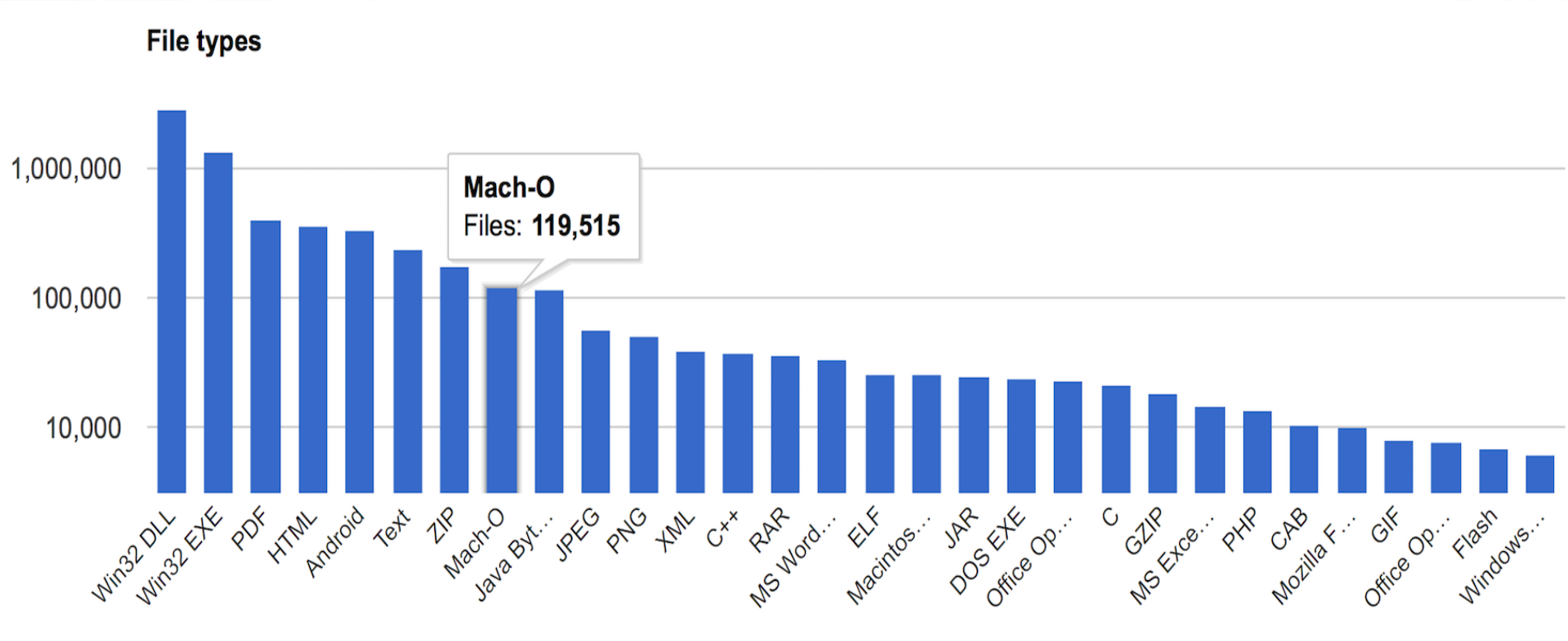
For Fun ;) \0/

- Malware
- Crackmes
- Vulnerability Hunting

0x00 Motivation Of Research



0x00 Motivation Of Research



Source: www.virustotal.com 17/11/2016

0x00 Motivation Of Research

Mac.BackDoor.OpinionSpy.3

Names: MacOS_X/OpinionSpy.A (Microsoft),
Mac.BackDoor.OpinionSpy.3 (F-Secure),
Mac.BackDoor.OpinionSpy.3 (Trend)

.OSA --> ZIP:
- PremierOpinion
- upgrade.xml

OSX_KAITEN.A

Names: MacOS_X/Tsunami.A (Microsoft),
OSX/Tsunami (McAfee),
OSX/Tsunami-Gen (Sophos),
OSX/Tsunami.A (F-Secure),

Binary:
/tmp/.z

OSX_CARETO.A

Names: MacOS:Appetite-A [Trj] (Avast)
OSX/BackDoor.A (AVG)
Trojan.OSX.Melgato.a (Kaspersky)
OSX/Backdoor-BRE (McAfee)
Backdoor:MacOS_X/Appetite.A (Microsoft)
OSX/Appetite-A (Sophos)

itunes212.{BLOCKED}pdt.com

0x01 The Mach-o Format

The mach-o is a Universal (fat) binaries (for i386 x86_64 ppc ppc64 armv6 armv7), this format was adopted as the standard in OS X from version 10.6 on.

We are currently in version ~~10.11.5 (El Capitan)~~ 10.12 MacOS Sierra

0x01 The Mach-o Format

```
bash-3.2$ pwd
/opt/local/lib
bash-3.2$ vim libmagic.dylib
```

```
bash-3.2$ pwd
/usr/share/file/magic
bash-3.2$
```

Binary (Linux)

```
➜ reversing  file cal
cal: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked
 (uses shared libs), for GNU/Linux 2.6.18, stripped
➜ reversing
```

Binary (Windows)

```
➜ reversing  file calc.exe
calc.exe: PE32+ executable for MS Windows (GUI) Mono/.Net assembly
➜ reversing
```

Binary (OS X)

```
➜  reversing  file /usr/bin/cal
/usr/bin/cal: Mach-O 64-bit executable x86_64
➜  reversing
```

0x01 The Mach-o Format

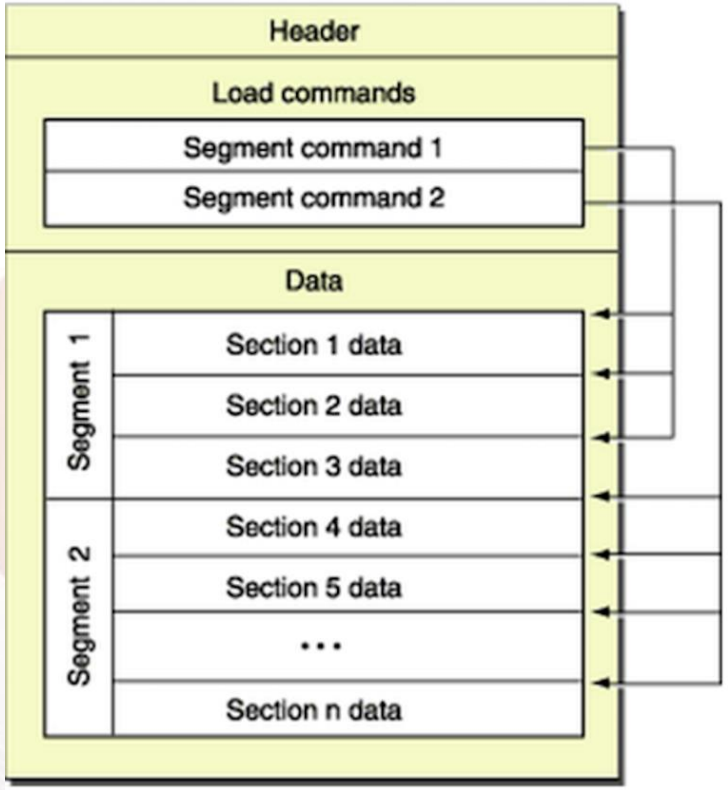
Magic Number: File Signatures

CA FE BA BE - Mach-O Fat Binary
FE ED FA CE - Mach-O binary (32-bit)
FE ED FA CF - Mach-O binary (64-bit)
CE FA ED FE - Mach-O binary (reverse byte 32-bit)
CF FA ED FE - Mach-O binary (reverse byte 64-bit)

0x01 The Mach-o Format

Mach-O (Mach Object)

HEADER
LOAD COMMANDS
SECTIONS



Architecture of object code

ppc ppc64 i386 x86_64 armv6
armv7 armv7s arm64

0x01 The Mach-o Format

Structs on File:

```
loganbr ~ $ vim /usr/include/mach-o/loader.h
```

- Code is located in __TEXT section.
- Linked libraries in LC_LOAD_DYLIB commands.
- The entrypoint is defined at LC_UNIXTHREAD or LC_THREAD.

0x01 The Mach-o Format

HEADER


```
loganbr ~ $ vim /usr/include/mach-o/loader.h
```

```
/*
 * The 64-bit mach header appears at the very beginning of object files for
 * 64-bit architectures.
 */
struct mach_header_64 {
    uint32_t    magic;      /* mach magic number identifier */
    cpu_type_t  cputype;    /* cpu specifier */
    cpu_subtype_t   cpusubtype; /* machine specifier */
    uint32_t    filetype;   /* type of file */
    uint32_t    ncmds;      /* number of load commands */
    uint32_t    sizeofcmds; /* the size of all the load commands */
    uint32_t    flags;      /* flags */
    uint32_t    reserved;   /* reserved */
};
```

0x01 The Mach-o Format

LOAD_COMMANDS


```
loganbr ~ $ vim /usr/include/mach-o/loader.h
```

```
struct load_command {
    uint32_t cmd;       /* type of load command */
    uint32_t cmdsize;   /* total size of command in bytes */
};
```

0x01 The Mach-o Format

SECTIONS


```
loganbr ~ $ vim /usr/include/mach-o/loader.h
```

```
struct section_64 { /* for 64-bit architectures */
    char        sectname[16];   /* name of this section */
    char        segname[16];    /* segment this section goes in */
    uint64_t    addr;       /* memory address of this section */
    uint64_t    size;       /* size in bytes of this section */
    uint32_t    offset;     /* file offset of this section */
    uint32_t    align;      /* section alignment (power of 2) */
    uint32_t    reloff;     /* file offset of relocation entries */
    uint32_t    nreloc;     /* number of relocation entries */
    uint32_t    flags;      /* flags (section type and attributes)*/
    uint32_t    reserved1;  /* reserved (for offset or index) */
    uint32_t    reserved2;  /* reserved (for count or sizeof) */
    uint32_t    reserved3;  /* reserved */
};
```

0x01 The Mach-o Format

Cross Compiling

Building a mach-o file supported
by multiple platforms.

```
logan /opt $
logan /opt $ gcc -arch x86_64 -arch i386 -arch ppc7400 -c hello.c
logan /opt $
```

```
logan /opt $ file hello
hello: Mach-O universal binary with 3 architectures
hello (for architecture x86_64):        Mach-O 64-bit executable x86_64
hello (for architecture i386):  Mach-O executable i386
hello (for architecture ppc7400):       Mach-O executable ppc
```

0x01 The Mach-o Format

```
logan /opt/malware/mach-o $ lipo -detailed_info sample_01
Fat header in: sample_01
fat_magic 0xcafebabe ——— Magic Number: Mach-o Fat Binary
nfat_arch 3
architecture x86_64
    cputype CPU_TYPE_X86_64
    cpusubtype CPU_SUBTYPE_X86_64_ALL
    offset 4096
    size 17136
    align 2^12 (4096)
architecture i386
    cputype CPU_TYPE_I386
    cpusubtype CPU_SUBTYPE_I386_ALL
    offset 24576
    size 16776
    align 2^12 (4096)
architecture ppc7400
    cputype CPU_TYPE_POWERPC
    cpusubtype CPU_SUBTYPE_POWERPC_7400
    offset 45056
    size 20652
    align 2^12 (4096)
```

```
logan /opt/malware/mach-o $ lipo -extract x86_64 -output sample_01-x86_64 sample_01
logan /opt/malware/mach-o $
logan /opt/malware/mach-o $ file sample_01-x86_64
sample_01-x86_64: Mach-O universal binary with 1 architecture
sample_01-x86_64 (for architecture x86_64):    Mach-O 64-bit executable x86_64
logan /opt/malware/mach-o $
logan /opt/malware/mach-o $ lipo -detailed_info sample_01-x86_64
Fat header in: sample_01-x86_64
fat_magic 0xcafebabe
nfat_arch 1
architecture x86_64
    cputype CPU_TYPE_X86_64
    cpusubtype CPU_SUBTYPE_X86_64_ALL
    offset 4096
    size 17136
    align 2^12 (4096)
```

0x01 The Mach-o Format


```
logan /opt/malware/mach-o $ otool -h sample_01 | more
sample_01:
Mach header   Magic Number: Mach-o binary 64-bit
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
 0xfeedfacf 16777223          3  0x80           2    13       1968 0x00000085
```

0x01 The Mach-o Format

```
Load command 0
      cmd LC_SEGMENT_64
  cmdsize 72
  segname __PAGEZERO
   vmaddr 0x0000000000000000
   vmsize 0x0000000100000000
  fileoff 0
 filesize 0
  maxprot 0x00000000
 initprot 0x00000000
   nsects 0
    flags 0x0
Load command 1
      cmd LC_SEGMENT_64
  cmdsize 632
  segname __TEXT
```

0x01 The Mach-o Format

```
Section
    sectname __symbol_stub1
     segname __TEXT
        addr 0x0000000100001402
        size 0x00000000000000c6
      offset 5122
       align 2^1 (2)
      reloff 0
      nreloc 0
       flags 0x80000408
  reserved1 0 (index into indirect symbol table)
  reserved2 6 (size of stubs)
Section
    sectname __cstring
     segname __TEXT
        addr 0x0000000100000f14c8
```

0x01 The Mach-o Format

```
Load command 7
          cmd LC_LOAD_DYLINKER
      cmdsize 32
         name /usr/lib/dyld (offset 12)
Load command 8
      cmd LC_UUID
  cmdsize 24
     uuid 853B1C68-5F39-DF57-87AA-96A3D27064DD
Load command 9
        cmd LC_UNIXTHREAD
    cmdsize 184
     flavor x86_THREAD_STATE64
      count x86_THREAD_STATE64_COUNT
    rax  0x0000000000000000 rbx 0x0000000000000000 rcx  0x0000000000000000
```

0x02 Demo I

Demo 01 Crackme

Binary: cryptorev
Level: easy
Detail: cryptorev is a binary mach-o the goal is run binary and found the flag.

This demo don’t have protection in this code just UPX (packer) and DEADBEEF in file ;)

0x03 Tricks for Reversing

When talking about malwares we have a lot of techniques to make it difficult to analyze (reversing the sample) like:

- Anti-Disassembly
- Anti-Debugging
- Obfuscation
- Anti-Vm

Good Research about this:
https://www.blackhat.com/docs/us-14/materials/us-14-Branco-Prevalent-Characteristics-In-Modern-Malware.pdf

0x03 Tricks for Reversing

VMWARE FUSION / PARALLELS / VIRTUALBOX

- Keep Virtualization Software Updated
- Use System Tools Installed in VM
- Network Host-Only mode
- If you use Shared Folder(Host) leave it as “read-only”
- Disable Gatekeeper (Allow apps downloaded from: Anywhere)

0x03 Tricks for Reversing

Static Analysis

file	->	determine file type
upx / binwalk	->	compress or expand executable files
strings	->	find the printable strings in a object, or other binary, file
strip	->	remove symbols
hexEdit	->	hex editor
Lipo	->	create or operate on universal files
otool	->	object file displaying tool like a objdump and ldd
nm	->	display name list (symbol table)
codesign	->	create and manipulate code signatures
machOView	->	visual Mach-O file browser
class-dump	->	utility for examining the Objective-C runtime information stored in Mach-O files.
dtrace	->	generic front-end to the DTrace facility
fs_usage	->	report system calls and page faults related to filesystem activity in real-time
xattr	->	display and manipulate extended attributes

0x03 Tricks for Reversing

Dynamic Analysis

Xcode	->	xcode is an (IDE) containing a suite of software development.
iDA Pro	->	disassembler and debugger.
hopper	->	tool used for disassemble, and decompile your 32/64bits mach-o file.
lldb	->	debugger
fseventer	->	disk activity tool with a good graphical representation and solid filter tool.
open snoop	->	snoop file opens as they occur. Uses DTrace.
activity Monitor	->	tool to help you keep your system in good shape.
procoxp	->	It's a simple tool like a top get information accessible by proc_info
tcpdump	->	for dump and analisys traffic on a network
wireshark	->	for dump and analisys traffic on a network
lsock	->	based on PF_SYSTEM provider, you can get real time notifications of socket activity like TCPView from SysInternals.
little Snitch	->	network traffic monitoring and control.

0x04 Demo II

Demo 02 Ransomware Keranger

On March 2016 appear the first Ransomware writing for mach-o file on OSX System (KeRanger), Distributed by client BitTorrent Transmission (v.2.90) This threat has been fixed in version v.2.91 the client.

The latest version Gatekeeper OSX already block this ransomware since the first sample published \0/!!!

Reference

Hacking is a way of life

REVERSE Engineering Mac Malware - Defcon 22

https://www.defcon.org/images/defcon-22/dc-22-presentations/Edwards/DEFCON-22-Sarah-Edwards-Reverse-Engineering-Mac-Malware.pdf

OS X ABI Mach-O File Format Reference

https://developer.apple.com/library/mac/documentation/DeveloperTools/Conceptual/MachORuntime/index.html

Calling Convention

http://www.agner.org/optimize/calling_conventions.pdf

Thanks for my wife and brothers (Rubira, Gabriel Negreiros, C00ler, Slayer, Ygor-r0gy)

0x05 Conclusions

Question &(MAYBE ;) \0/)Answer

Contact: ricardologanbr@gmail.com
@l0ganbr