0xOPOLeaks Logo by @ines


0xOPOSEC
0xOF - The Virtual Meet

Zezadas

zezadas@sefod.eu

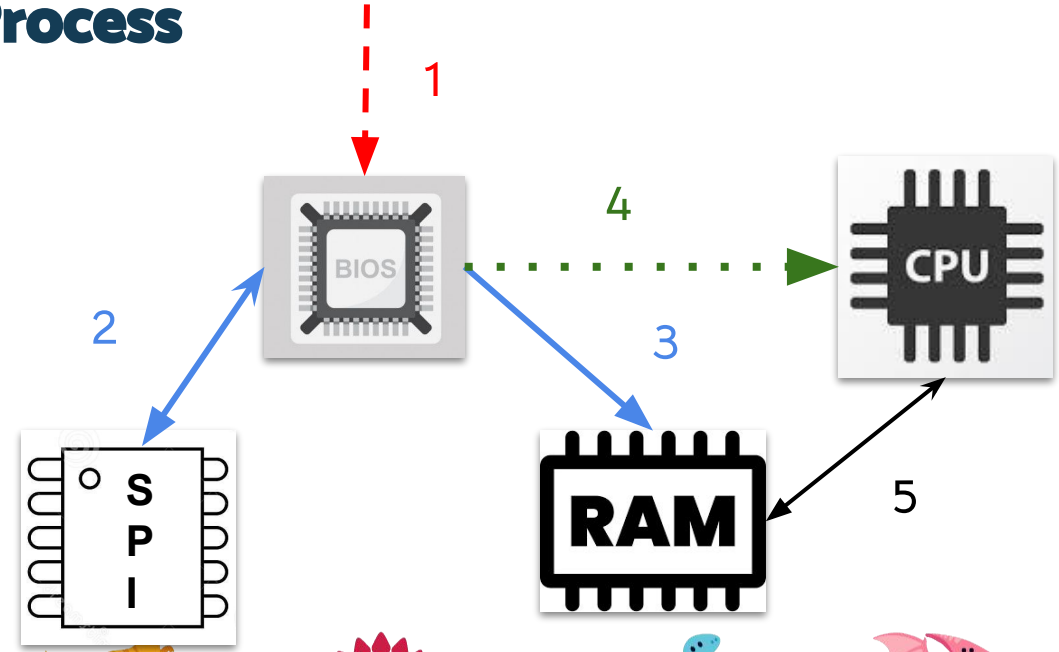
https://sefod.eu

@0xz3z4d45

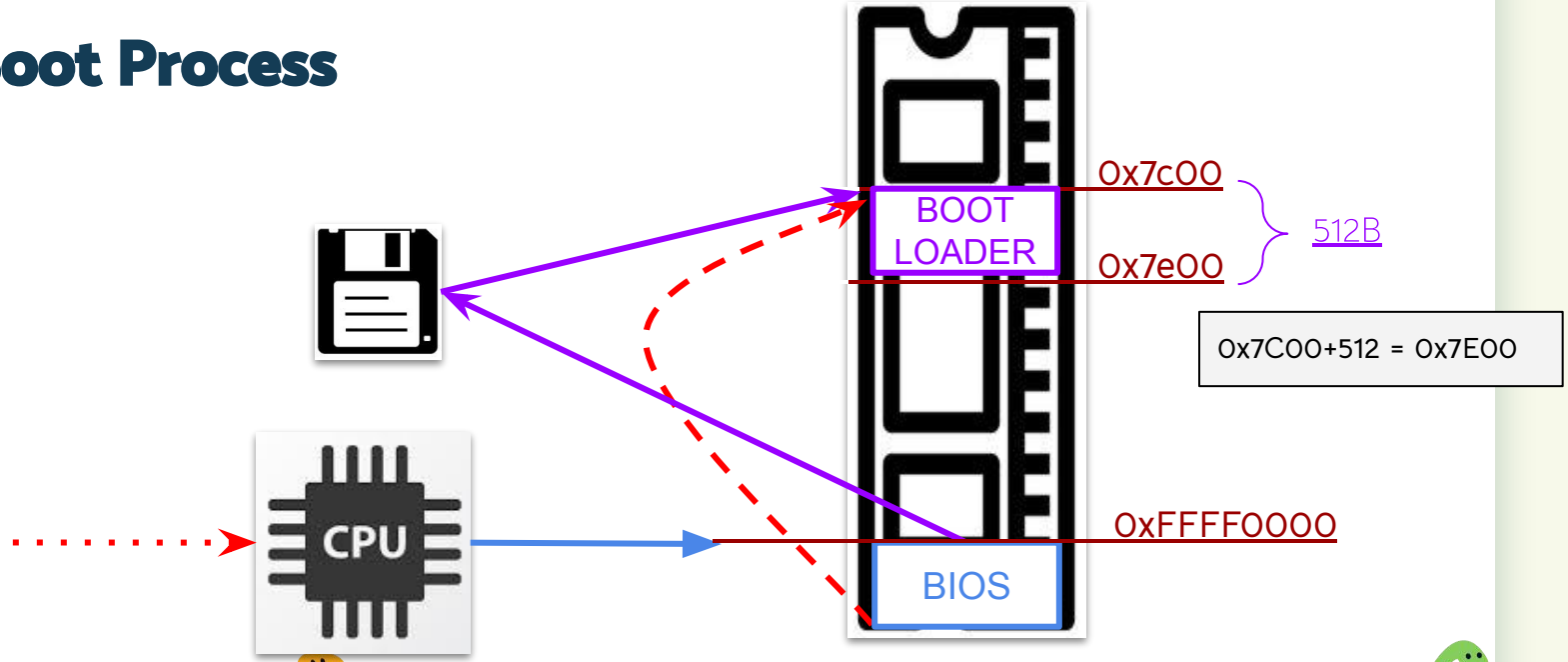
@zezadas

Boot Stages

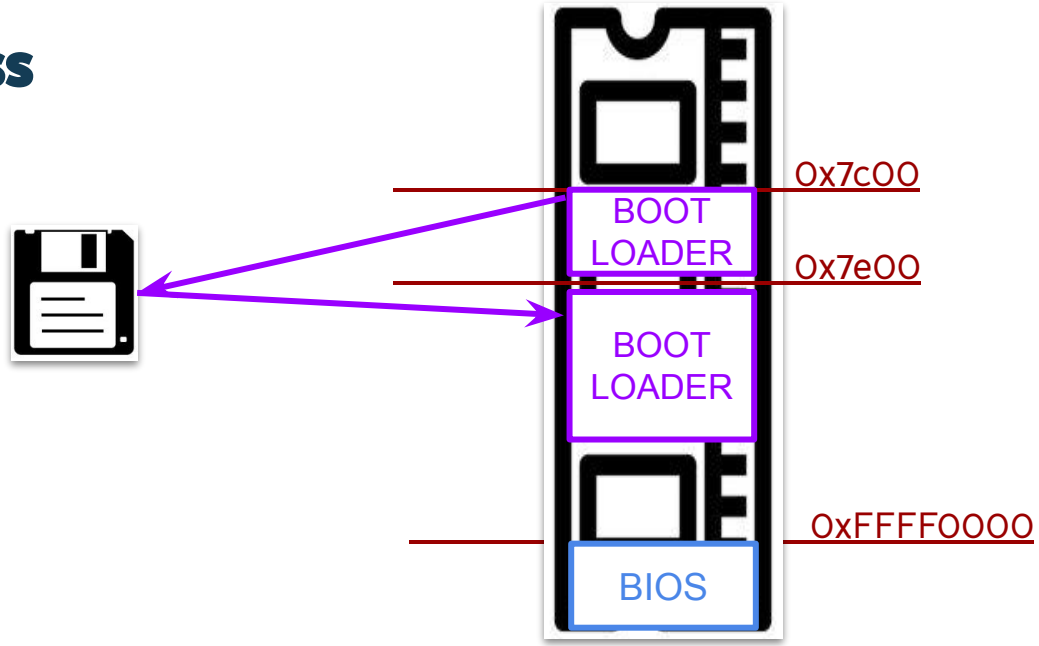
Boot Process



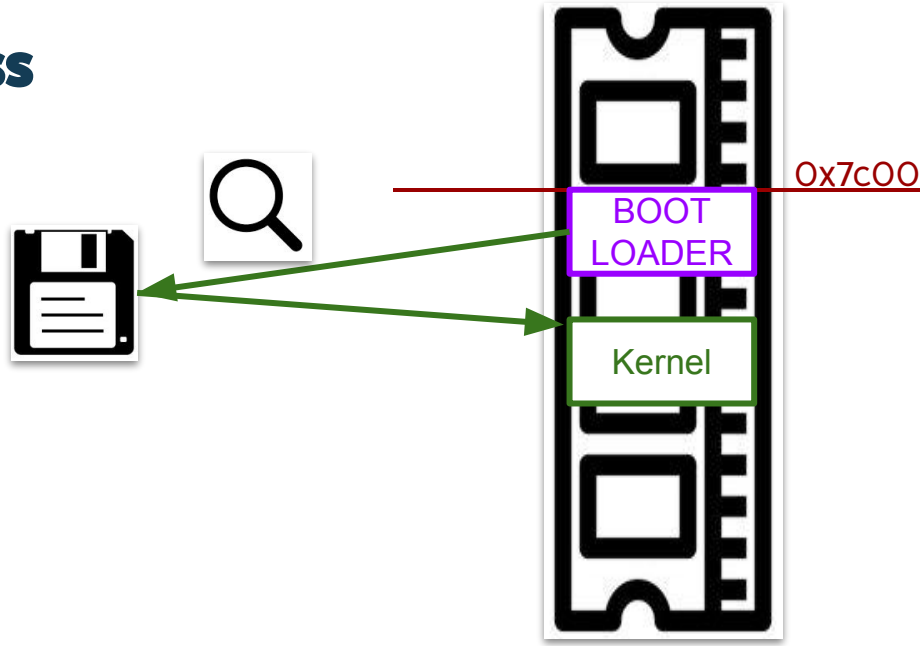
Boot Process



Boot Process

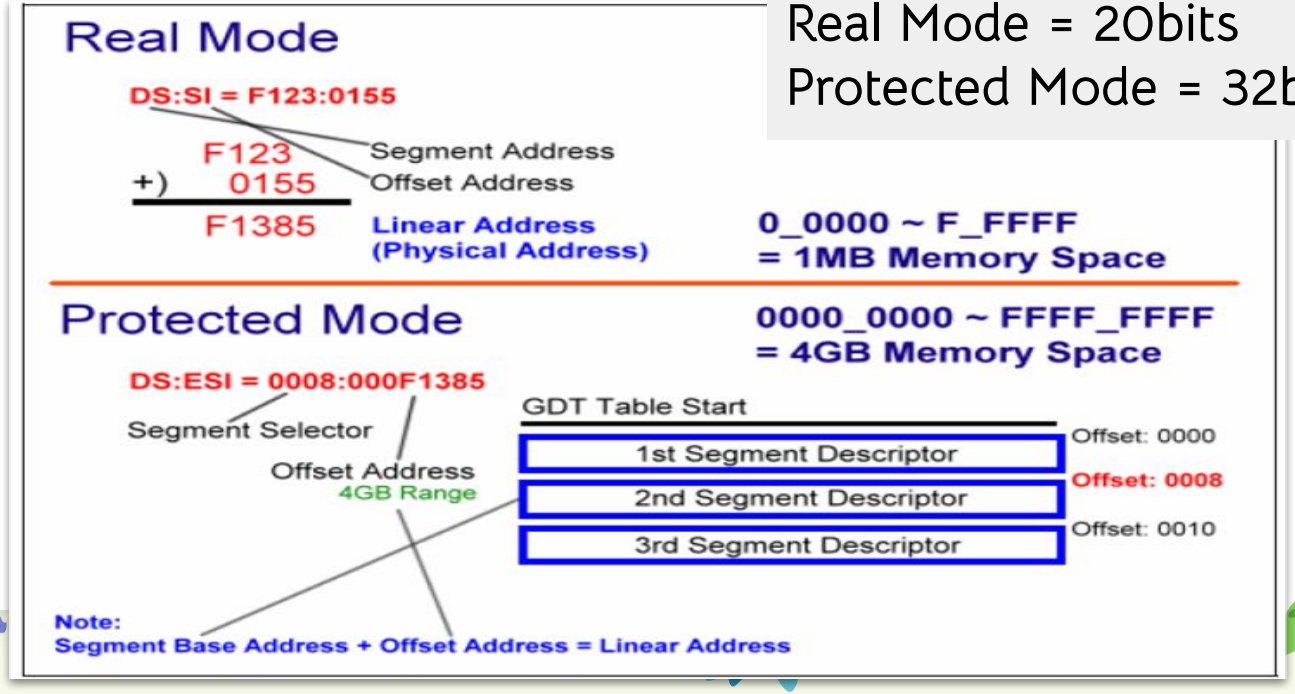


Boot Process



Protected Mode & Real Mode

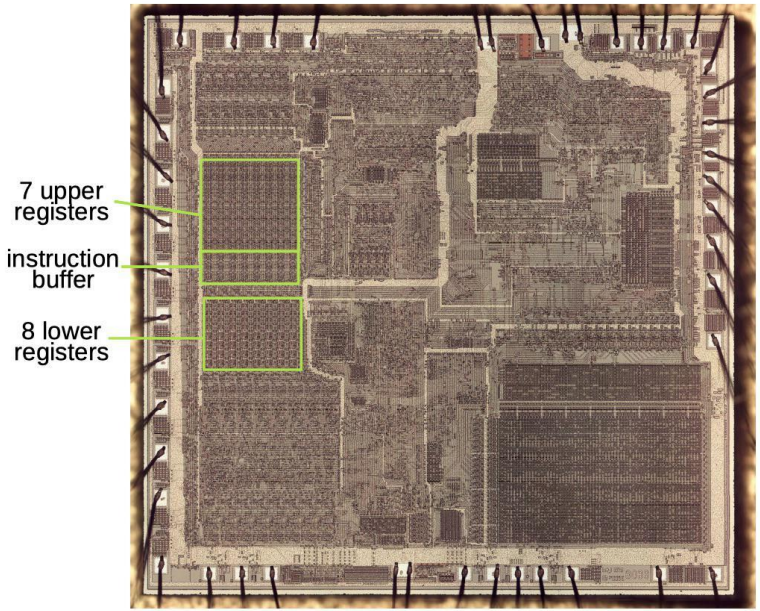
Real Mode = 20bits
Protected Mode = 32bits



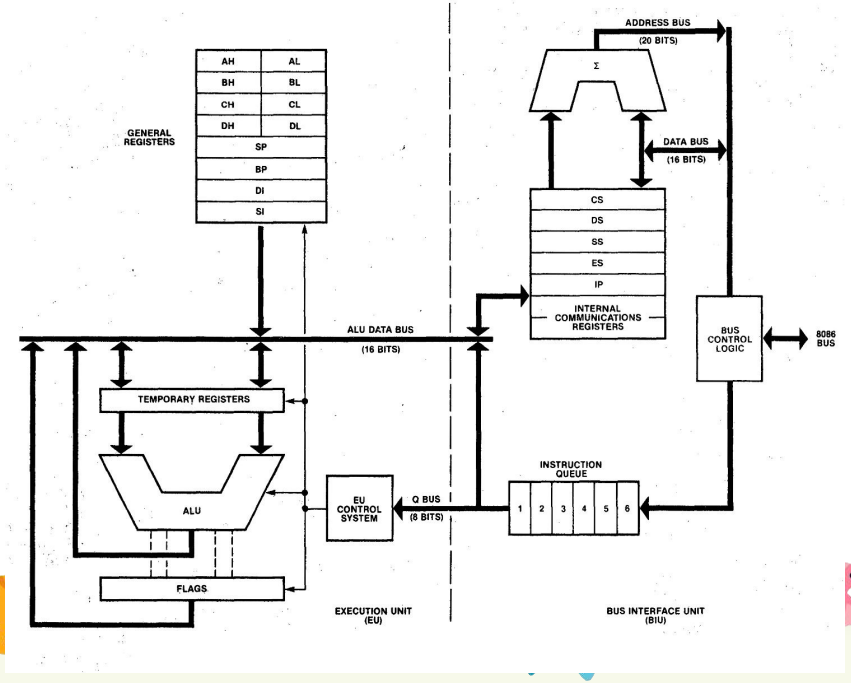
https://2.bp.blogspot.com/-768m37PvfJA/UKtKlFiuOwI/AAAAAAAAADs/CZsQFnfcfnY/s1600/%E5%9C%96%E7%89%871.png

Intel 8086

CPU 8086



CPU 8086


ADDRESS BUS
(20 BITS)
Σ
DATA BUS
(16 BITS)
GENERAL REGISTERS
AH
AL
BH
BL
CH
CL
DH
DL
SP
BP
DI
SI
CS
DS
SS
ES
IP
INTERNAL COMMUNICATIONS REGISTERS
BUS CONTROL LOGIC
8086 BUS
ALU DATA BUS
(16 BITS)
TEMPORARY REGISTERS
ALU
FLAGS
EU CONTROL SYSTEM
Q BUS
(8 BITS)
INSTRUCTION QUEUE
1
2
3
4
5
6
EXECUTION UNIT
(EU)
BUS INTERFACE UNIT
(BIU)

Intel 8086

Intel 8086 registers

$^{1}_{9}$ $^{1}_{8}$ $^{1}_{7}$ $^{1}_{6}$ $^{1}_{5}$ $^{1}_{4}$ $^{1}_{3}$ $^{1}_{2}$ $^{1}_{1}$ $^{1}_{0}$ $^{0}_{9}$ $^{0}_{8}$ $^{0}_{7}$ $^{0}_{6}$ $^{0}_{5}$ $^{0}_{4}$ $^{0}_{3}$ $^{0}_{2}$ $^{0}_{1}$ $^{0}_{0}$			*(bit position)*
Main registers			
	AH	AL	**AX** (primary accumulator)
	BH	BL	**BX** (base, accumulator)
	CH	CL	**CX** (counter, accumulator)
	DH	DL	**DX** (accumulator, extended acc.)
Index registers			
0 0 0 0	SI		**S**ource **I**ndex
0 0 0 0	DI		**D**estination **I**ndex
0 0 0 0	BP		**B**ase **P**ointer
0 0 0 0	SP		**S**tack **P**ointer
Program counter			
0 0 0 0	IP		**I**nstruction **P**ointer
Segment registers			
CS		0 0 0 0	**C**ode **S**egment
DS		0 0 0 0	**D**ata **S**egment
ES		0 0 0 0	**E**xtra **S**egment
SS		0 0 0 0	**S**tack **S**egment
Status register			
- - - - O D I T S Z - A - P - C			Flags

Intel 8086

$AX = 0xABCD
$AH = = 0xAB
$AL = = 0xCD

Intel 8086 registers

$^{1}_{9}$	$^{1}_{8}$	$^{1}_{7}$	$^{1}_{6}$	$^{1}_{5}$	$^{1}_{4}$	$^{1}_{3}$	$^{1}_{2}$	$^{1}_{1}$	$^{1}_{0}$	$^{0}_{9}$	$^{0}_{8}$	$^{0}_{7}$	$^{0}_{6}$	$^{0}_{5}$	$^{0}_{4}$	$^{0}_{3}$	$^{0}_{2}$	$^{0}_{1}$	$^{0}_{0}$	*(bit position)*
Main registers																				
	AH									AL										**AX** (primary accumulator)
	BH									BL										**BX** (base, accumulator)
	CH									CL										**CX** (counter, accumulator)
	DH									DL										**DX** (accumulator, extended acc.)

Intel 8086

Program counter		
0 0 0 0	IP	Instruction Pointer
Segment registers		
CS	0 0 0 0	Code Segment
DS	0 0 0 0	Data Segment
ES	0 0 0 0	Extra Segment
SS	0 0 0 0	Stack Segment

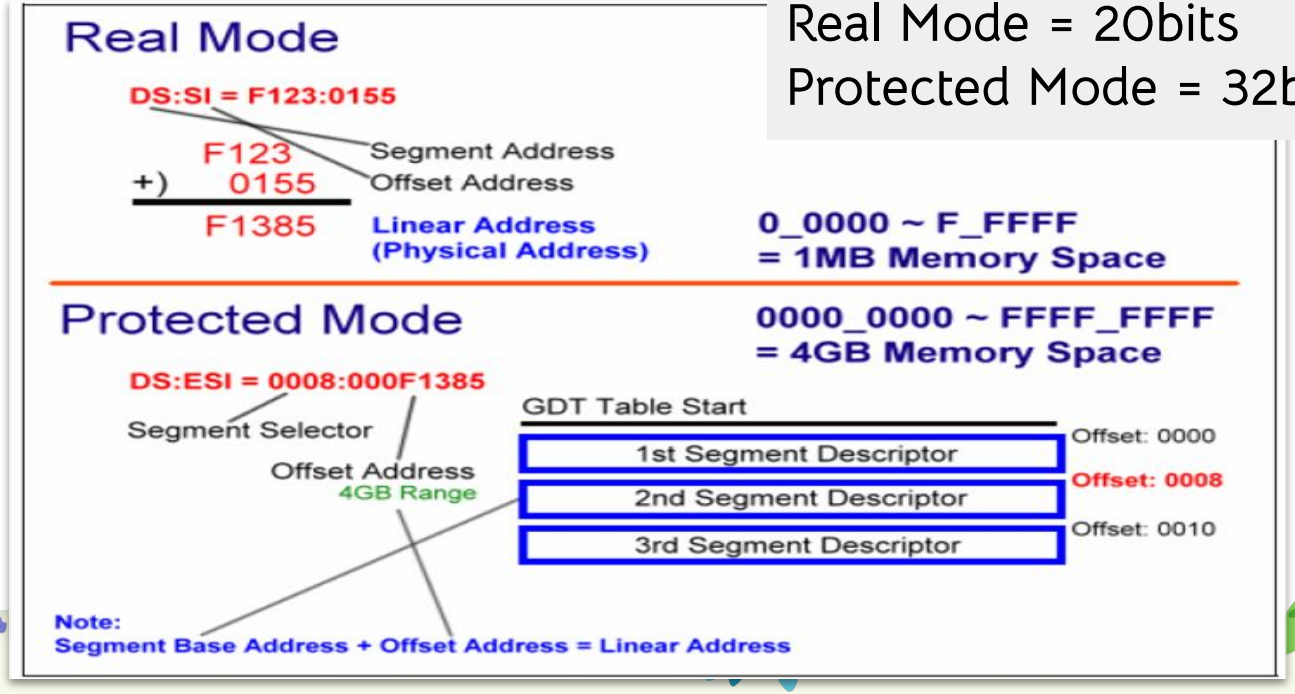
$CS= 0x1234 << 4

$CS= 0x12340
+$IP= 0x0056

$PC==0x12396

Protected Mode & Real Mode

Real Mode = 20bits
Protected Mode = 32bits



https://2.bp.blogspot.com/-768m37PvfJA/UKtKlFiuOwI/AAAAAAAAADs/CZsQFnfcfnY/s1600/%E5%9C%96%E7%89%871.png

Intel 8086

```
gef➤  x/3i 0x7e00
   0x7e00:      push   bp
   0x7e01:      mov    bp,sp
   0x7e03:      push   di
gef➤  print $cs
$6 = 0x7e0
gef➤  print $eip
$7 = (void (*)()) 0x1
gef➤  print ($cs<<4)+$eip
$8 = (void (*)()) 0x7e01
```

MBR

MASTER BOOT RECORD

≥ INVOKE-IR
BY: JARED ATKINSON
TEMPLATE BY: ANGE ALBERTINI

```
000: 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010: 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020: BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030: E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040: B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050: F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060: 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070: 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080: 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090: 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0: 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0: 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0: AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0: E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0: 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0: 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100: 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110: 53 66 55 66 68 00 00 00 00 66 68 00 00 7C 00 00 66
120: 61 68 00 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130: 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140: 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150: 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160: 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170: 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180: 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190: 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0: 65 6D 00 00 00 63 7B 9A 82 D4 BA 7D 00 00 00_20
1C0: 21 00 07 FE FF FF 00 08 00 00 00 90 36 06 80_FE
1D0: FF FF 07 FE FF FF 00 A0 36 06 00 60 09 00 00_00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00_00
1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

PARTITION TYPES

0x00 - EMPTY	0x83 - LINUX
0x01 - FAT12	0x84 - HIBERNATION
0x04 - FAT16	0x85 - LINUX_EXTENDED
0x05 - MS_EXTENDED	0x86 - NTFS_VOLUME_SET
0x06 - FAT16	0x87 - NTFS_VOLUME_SET_1
0x07 - NTFS	0xa0 - HIBERNATION_1
0x0b - FAT32	0xa1 - HIBERNATION_2
0x0c - FAT32	0xa5 - FREEBSD
0x0e - FAT16	0xa6 - OPENBSD
0x0f - MS_EXTENDED	0xa8 - MACOSX
0x11 - HIDDEN_FAT12	0xa9 - NETBSD
0x14 - HIDDEN_FAT16	0xab - MAC_OSX_BOOT
0x16 - HIDDEN_FAT16	0xb7 - BSDI
0x1b - HIDDEN_FAT32	0xb8 - BSDI_SWAP
0x1c - HIDDEN_FAT32	0xee - EFI_GPT_DISK
0x1e - HIDDEN_FAT16	0xef - EFI_SYSTEM_PARTITION
0x42 - MS_MBR_DYNAMIC	0xfb - VMWARE_FILE_SYSTEM
0x82 - SOLARIS_X86	0xfc - VMWARE_SWAP
0x82 - LINUX_SWAP	

BOOT CODE

FIELDS	VALUES
jump to boot program	
disk parameters	
boot program code	
disk signature	82D4BA7D

CHS ADDRESSING

```
00100000 00100001 00000000
00100000 100001 0000000000
  Head - 1st byte
Sector - 2nd byte (0-5 bits)
Cylinder - 2nd byte (6-7 bits)
           3rd byte
```

PARTITION TABLE

FIELDS	VALUES
status	0x00 - Non-Bootable
starting head	0x20
starting sector	0x21
starting cylinder	0x00
partition type	0x07 - NTFS
ending head	0xFE
ending sector	0x3F
ending cylinder	0x3FF
relative start sector	0x800
total sectors	0x6369000
status	0x80 - Bootable
starting head	0xFE
starting sector	0x3F
starting cylinder	0x3FF
partition type	0x07 - NTFS
ending head	0xFE
ending sector	0x3F
ending cylinder	0x3FF
relative start sector	0x636A000
total sectors	0x96000
partition type	0x00 - EMPTY
partition type	0x00 - EMPTY

END OF MBR

marker	0x55AA

https://nixhacker.com/explaining-the-magic-of-mbr-and-its/

MBR Format

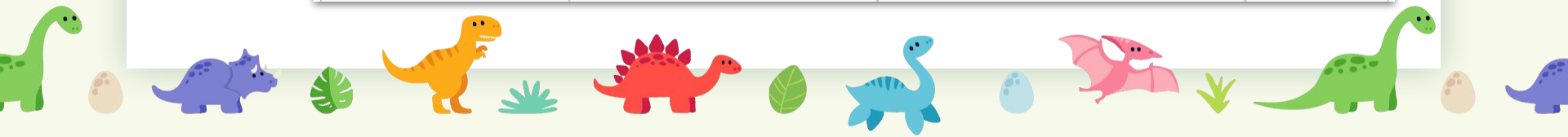
Structure of a master boot record

Address			description		Size in bytes
Hex	Oct	Dec			
0000	0000	0	Code area		440 (max.446)
01B8	0670	440	Disk signature (optional)		4
01BC	0674	444	Usually nulls; 0x0000		2
01BE	0676	446	Table of primary partitions (Four16-byte entries, IBM partition table scheme)		64
01FE	0776	510	55h	MBR signature	2
01FE	0777	511	AAh		
				MBR, total size:446+64+2=	512

https://nixhacker.com/explaining-the-magic-of-mbr-and-its/

MBR Max Partitions == 4

`0x01BE` (446)	Partition entry №1	*Partition table* (for primary partitions)	16
`0x01CE` (462)	Partition entry №2		16
`0x01DE` (478)	Partition entry №3		16
`0x01EE` (494)	Partition entry №4		16



https://en.wikipedia.org/wiki/Master_boot_record

WriteUp

The Intended Solution

Starts Where It Ended - (bit.ly/opoxmas)

From: Hacking Team <hacking.team@sefod.eu>

To: Rui Tinto <rui.tinto@sefod.eu>

Subject: 0xOPOLEAKS

Body:

We confirm the installation of the rootkit.
You know what they say, "Never leave your laptop unattended"

!!HACK THE PLANET!!
FLAG{things.may.not.always.be.what.they.seem}

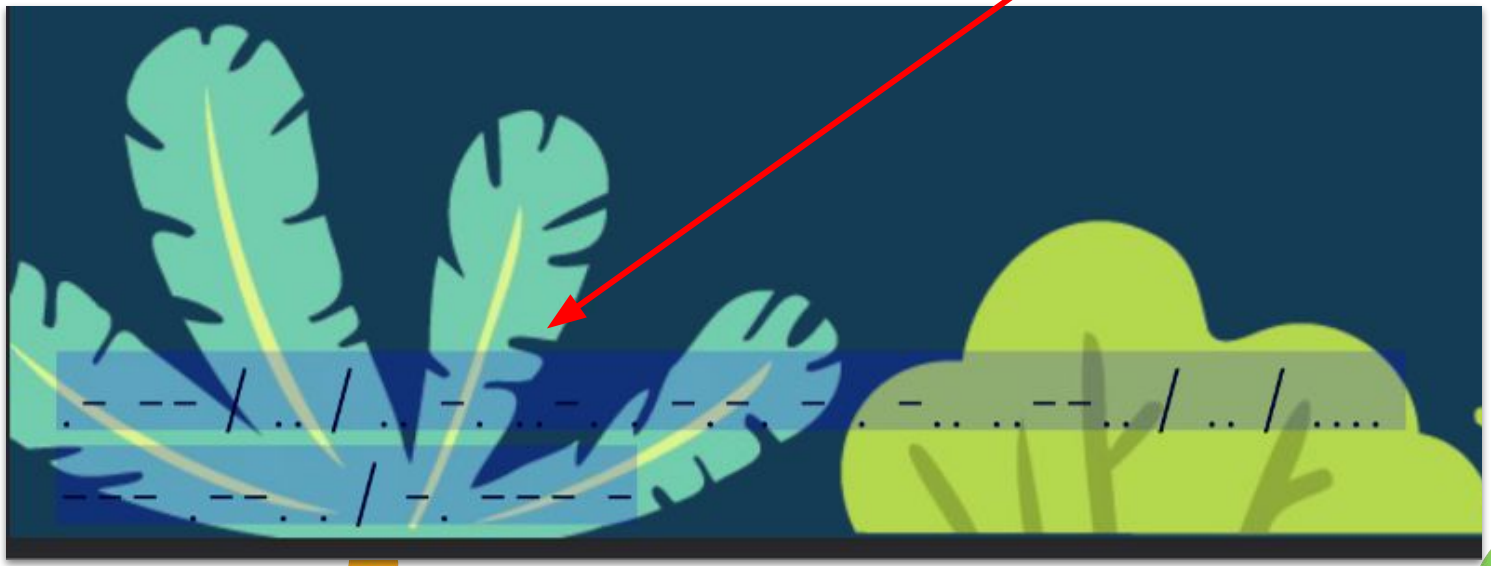
We kept a snippet of the rootkit for you. Have fun!
https://sefod.eu/ctf/snippet.html

Regards,
Hacking team

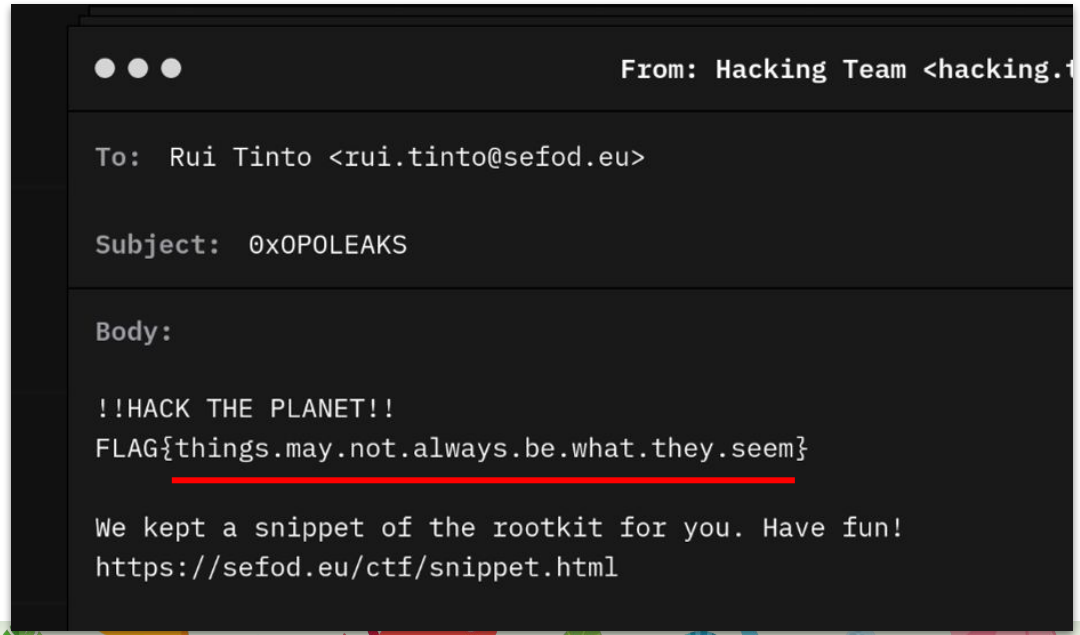
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.6 (MingW32)

0xOPOLEAKS

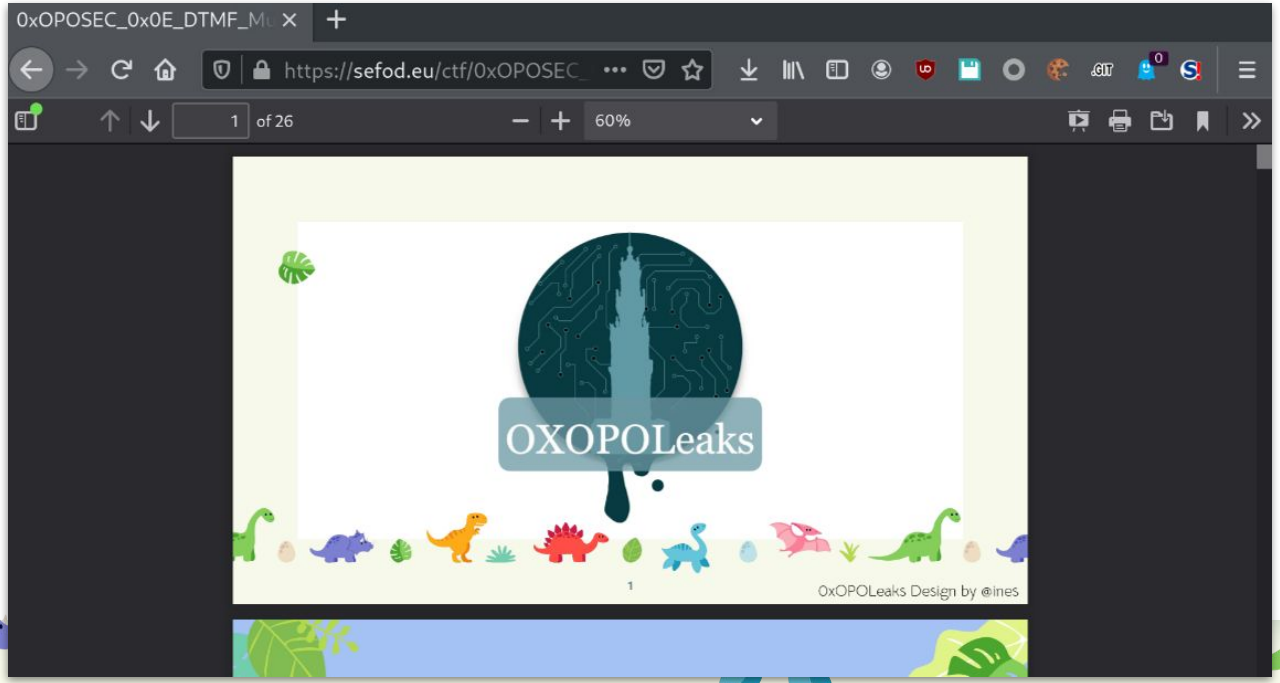
AM I INFECTED? I HOPE NOT



0xOPOLEAKS

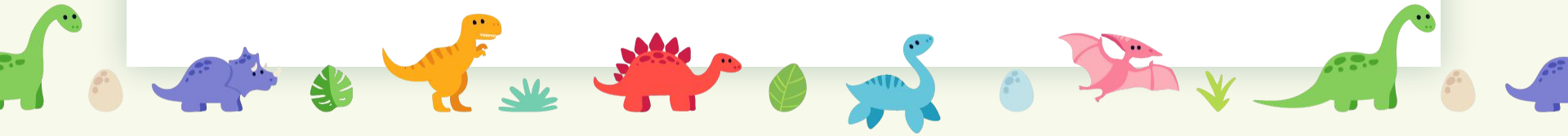

From: Hacking Team <hacking.t
To: Rui Tinto <rui.tinto@sefod.eu>
Subject: 0xOPOLEAKS
Body:
!!HACK THE PLANET!!
FLAG{things.may.not.always.be.what.they.seem}
We kept a snippet of the rootkit for you. Have fun!
https://sefod.eu/ctf/snippet.html

0xOPOLEAKS


0xOPOSEC_0x0E_DTMF_Mu
https://sefod.eu/ctf/0xOPOSEC_
1
of 26
60%
OXOPOLeaks
1
0xOPOLeaks Design by @ines

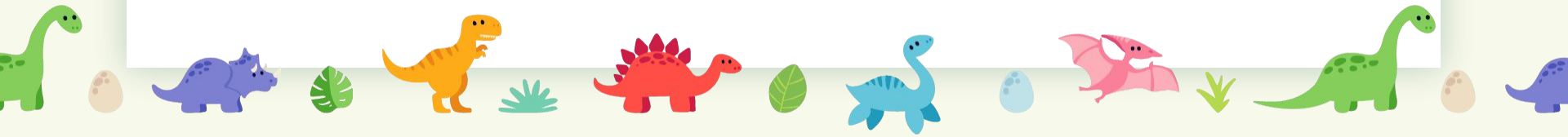
0xOPOLEAKS

```
└─ file 0xOPOSEC_0x0E_DTMF_MultiTap.pdf
0xOPOSEC_0x0E_DTMF_MultiTap.pdf: DOS/MBR boot sector
```

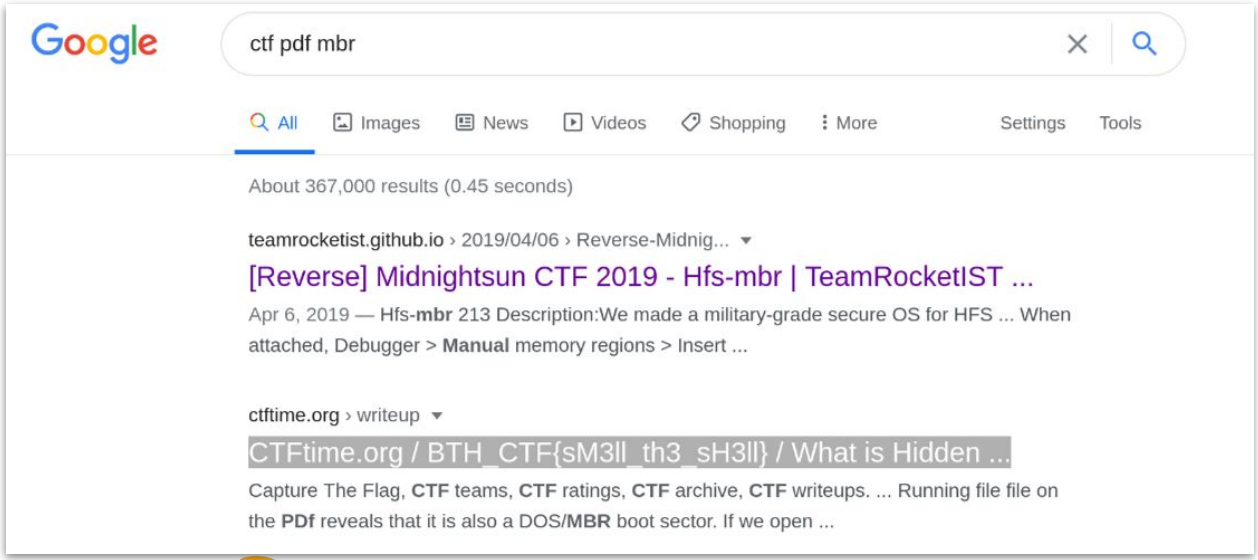


Not That Relevant

```
└─ binwalk 0xOPOSEC_0x0E_DTMF_MultiTap.pdf

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
6             0x6             PDF document, version: "1.5"
11818         0x2E2A          PDF document, version: "1.6"
11889         0x2E71          Zlib compressed data, default compression
12484         0x30C4          JPEG image data, JFIF standard 1.01
76376         0x12A58         Zlib compressed data, default compression
100370        0x18812         JPEG image data, JFIF standard 1.01
106033        0x19E31         Zlib compressed data, default compression
```



OSINT


Google
ctf pdf mbr
All
Images
News
Videos
Shopping
More
Settings
Tools
About 367,000 results (0.45 seconds)
teamrocketist.github.io › 2019/04/06 › Reverse-Midnig...
[Reverse] Midnightsun CTF 2019 - Hfs-mbr | TeamRocketIST ...
Apr 6, 2019 — Hfs-mbr 213 Description:We made a military-grade secure OS for HFS ... When attached, Debugger > Manual memory regions > Insert ...
ctftime.org › writeup
CTFtime.org / BTH_CTF{sM3ll_th3_sH3ll} / What is Hidden ...
Capture The Flag, CTF teams, CTF ratings, CTF archive, CTF writeups. ... Running file file on the PDf reveals that it is also a DOS/MBR boot sector. If we open ...

OSINT

by toblu / LionHack

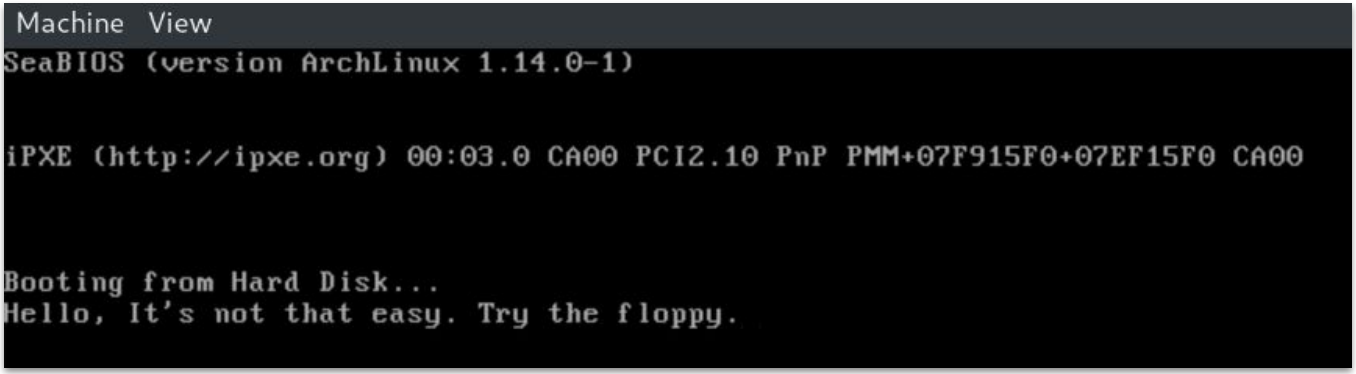
Rating: 5.0

What is Hidden (488 pts) (Misc)

```
I think there is something more with this PDF, can you find it?
Wrap the found flag in BTH_CTF{}

given: a .pdf file named Web.pdf
```

Running `file` file on the PDf reveals that it is also a DOS/MBR boot sector. If we open the file in qemu with `qemu-system-i386 Web.pdf` and p

qemu-system-i386 0xOPOSEC_0x0E_DTMF_MultiTap.pdf



OSINT stackoverflow

1 Answer

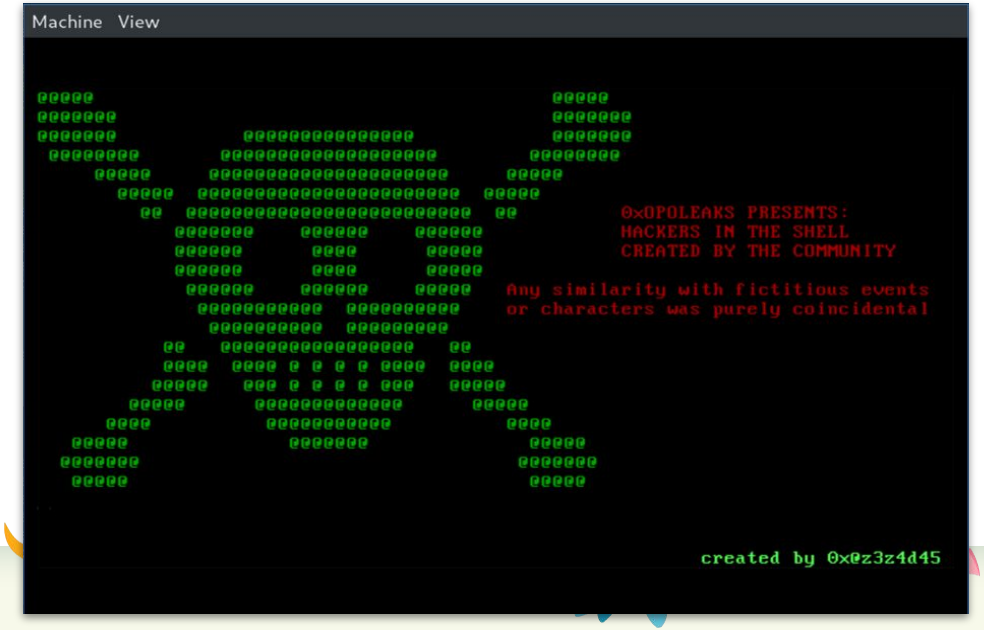
Active | Oldest | Votes

4

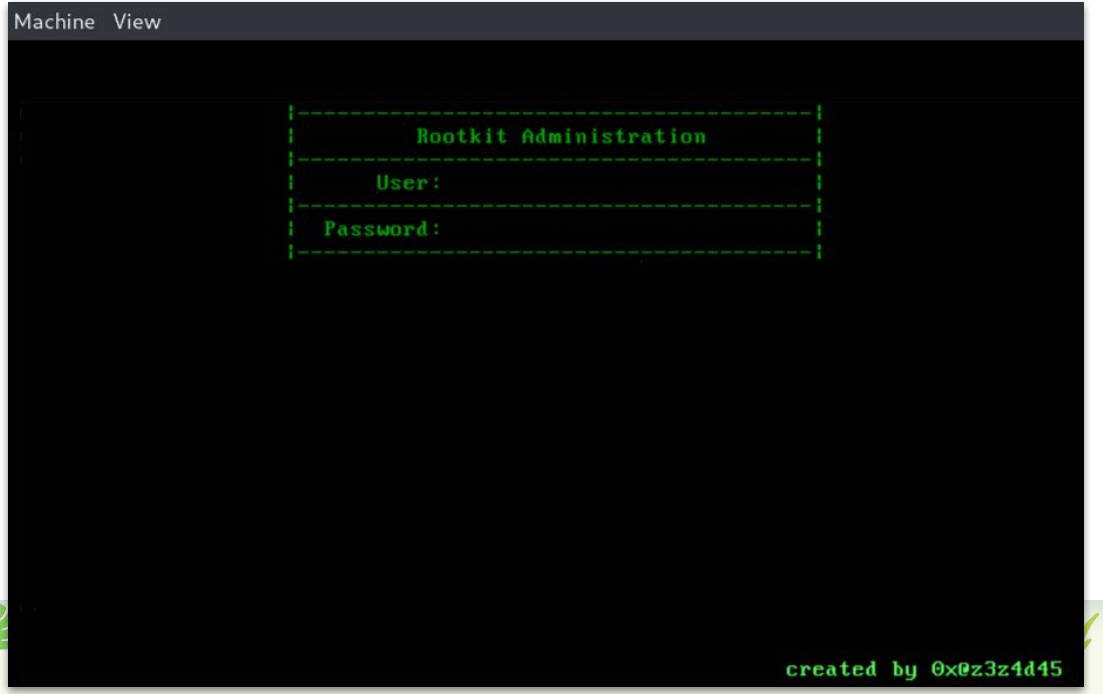
The short way:

```
$ qemu-system-x86_64 -fda /path/to/floppy.img
```

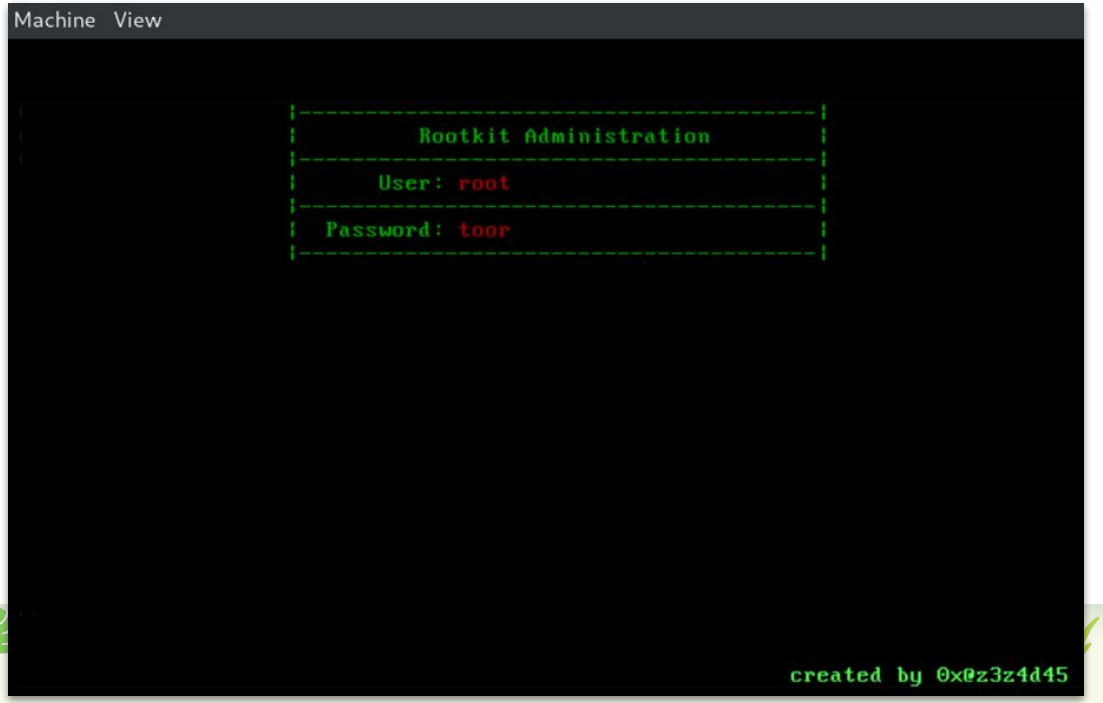
qemu-system-i386 -fda 0xOPOSEC_0x0E_DTMF_MultiTap.pdf



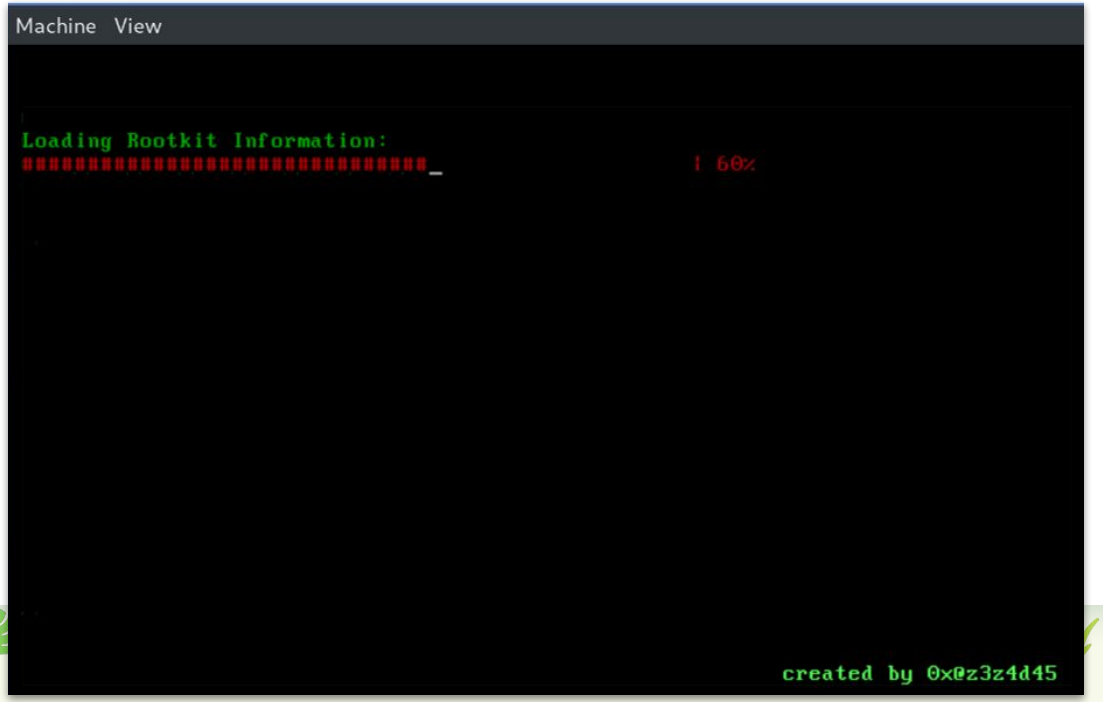
Noobie Login



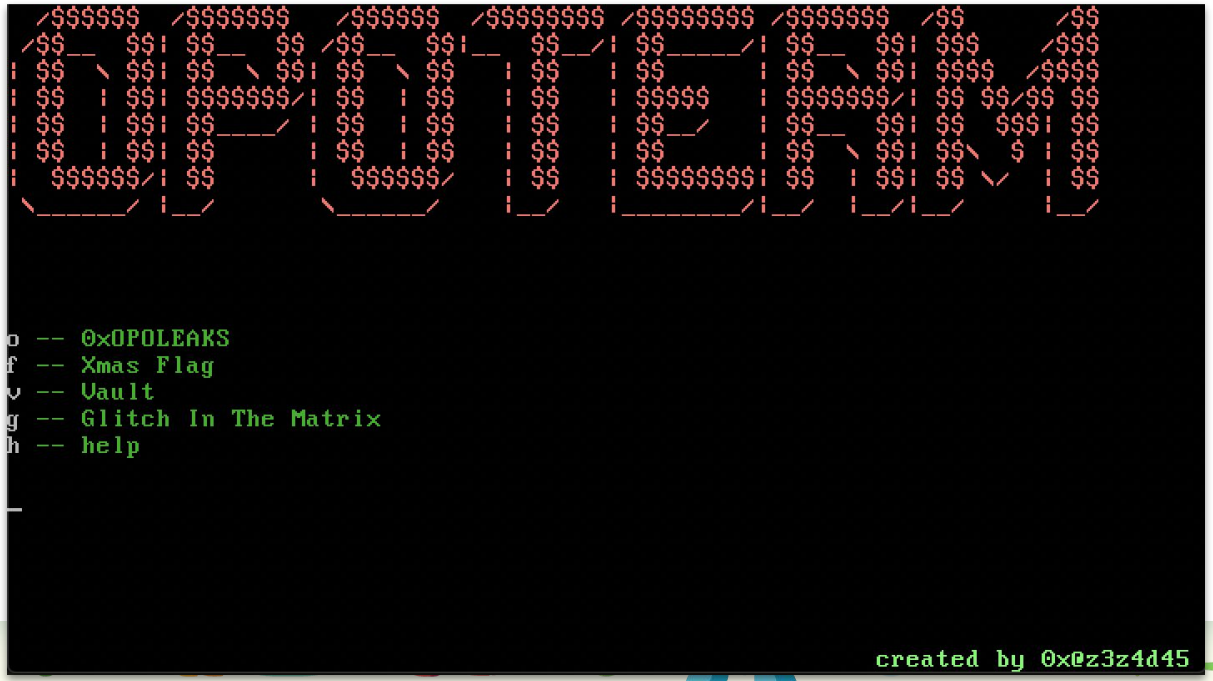
Noobie Login



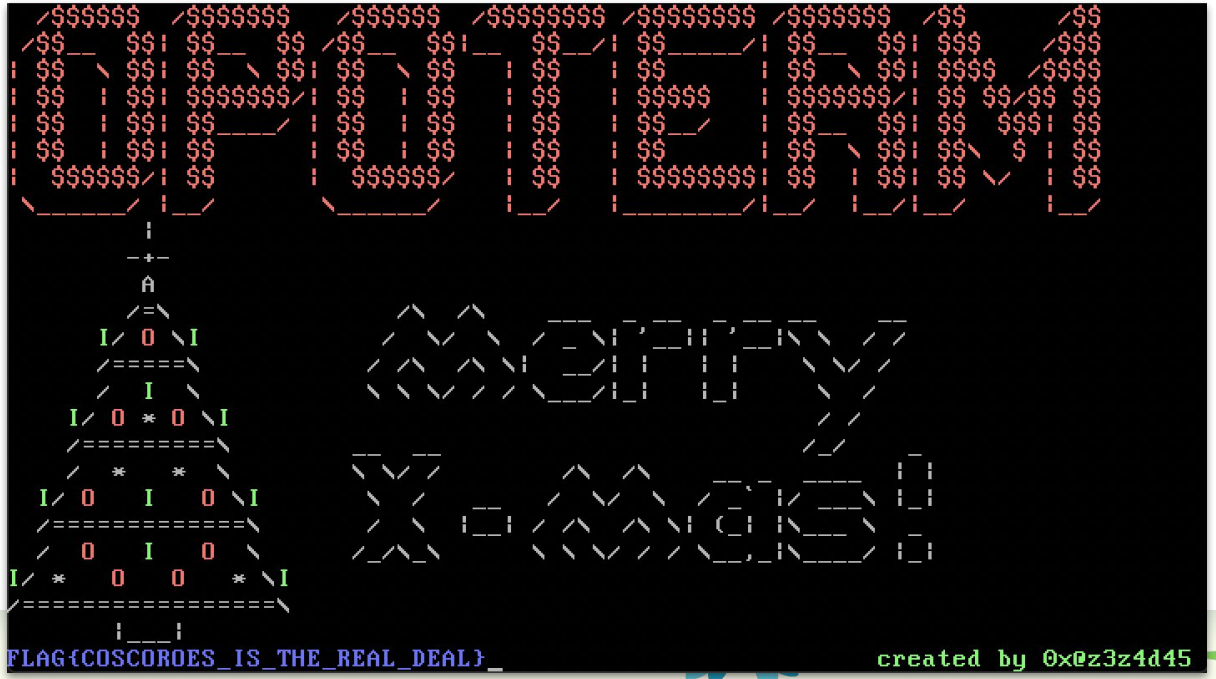
Loading...


Machine View
Loading Rootkit Information:
60%
created by 0x0z3z4d45

Main Menu

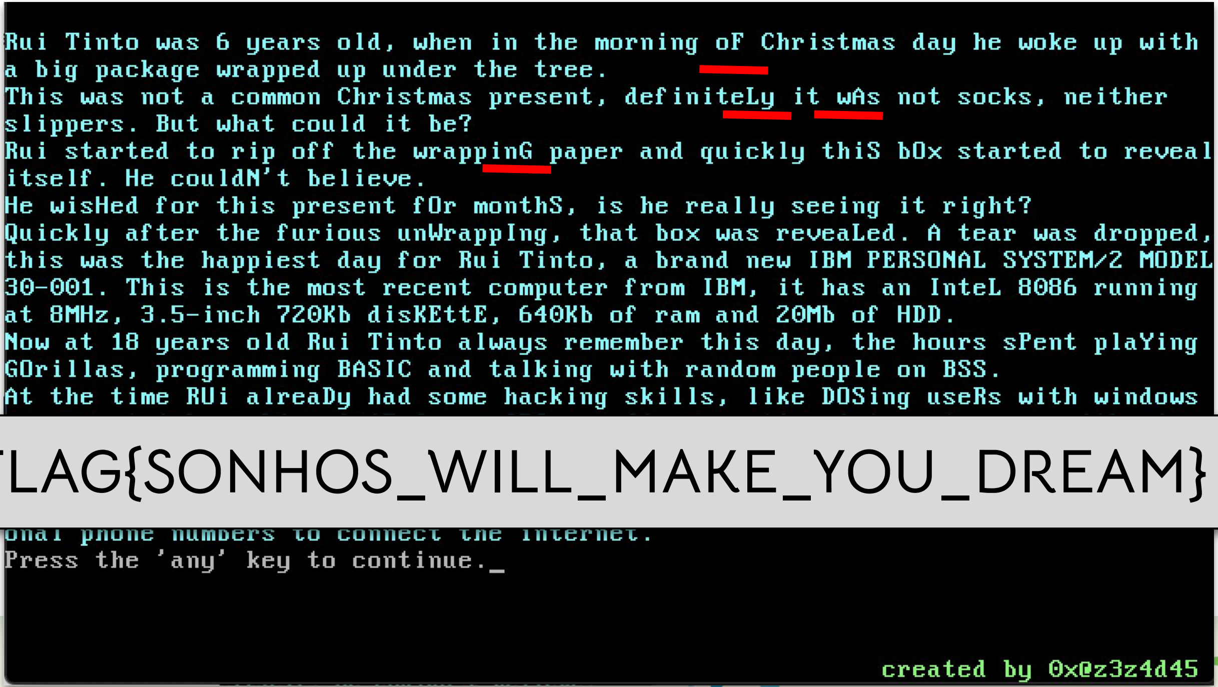

o -- 0xOPOLEAKS
f -- Xmas Flag
v -- Vault
g -- Glitch In The Matrix
h -- help
created by 0x@z3z4d45

XMas Flag




FLAG{COSCOROES_IS_THE_REAL_DEAL}

OxOPOLEAKS

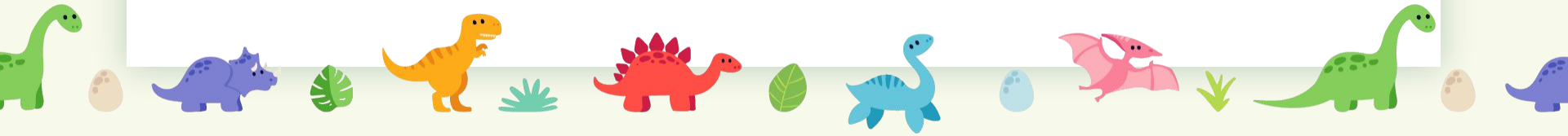


FLAG{SONHOS_WILL_MAKE_YOU_DREAM}

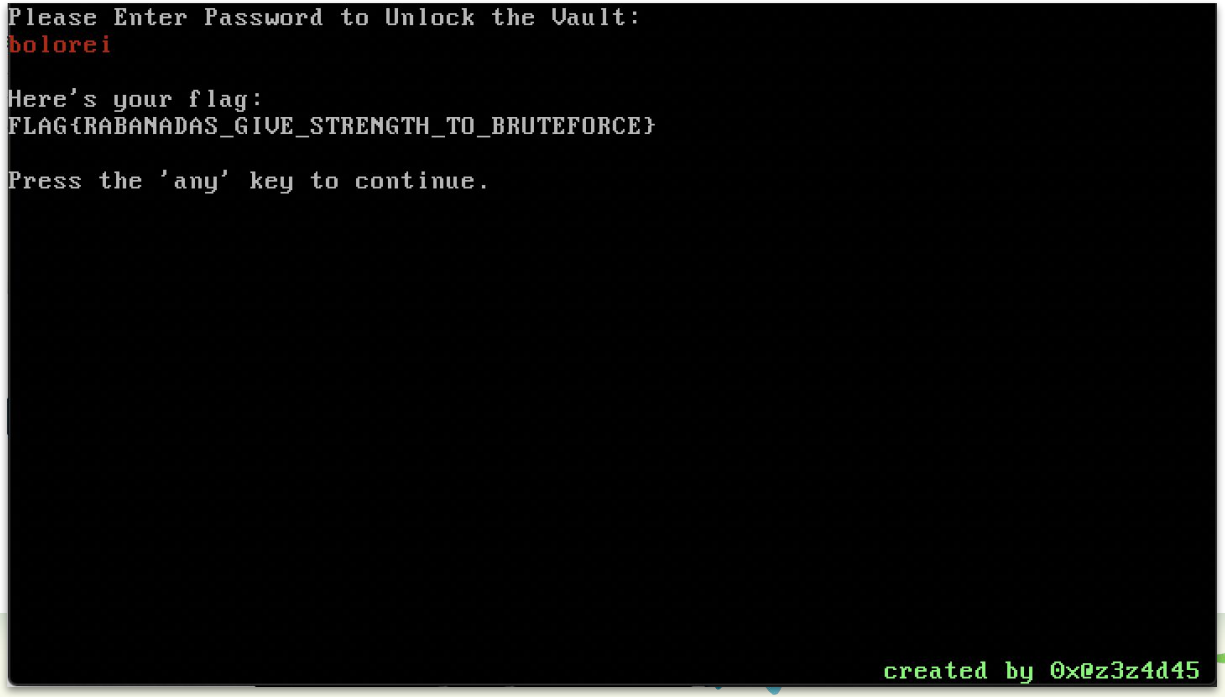

FLAG{SONHOS_WILL_MAKE
_YOU_DREAM}

Vault

```
Please Enter Password to Unlock the Vault:
bolorei
```



Vault


Please Enter Password to Unlock the Vault:
bolorei
Here's your flag:
FLAG{RABANADAS_GIVE_STRENGTH_TO_BRUTEFORCE}
Press the 'any' key to continue.
created by 0x@z3z4d45


FLAG{RABANADAS_GIVE_ST
RENGTH_TO_BRUTEFORCE}

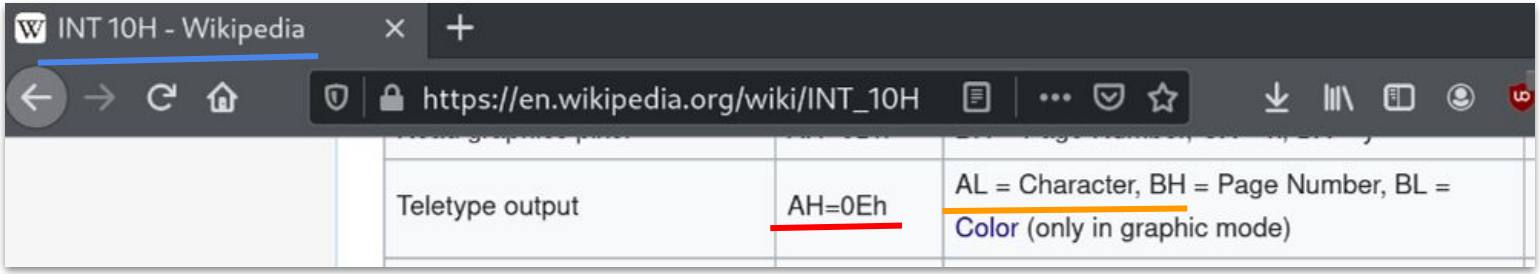
Glitch In The Matrix

```
There is a glitch in the matrix at 0x1337. Can you write shellcode to exfiltrate
 the flag?.
Example: B40EB041CD10 will print char 'A'
Type your shellcode:

B40EB041CD10
Output:
A
Press the 'any' key to continue.
```

created by 0x0z3z4d45

Glitch In The Matrix

Glitch In The Matrix




```
0x0000000000000000:  B4 0E     mov ah, 0xe
0x0000000000000002:  B0 41     mov al, 0x41 'A'
0x0000000000000004:  CD 10     int 0x10
```

Glitch In The Matrix

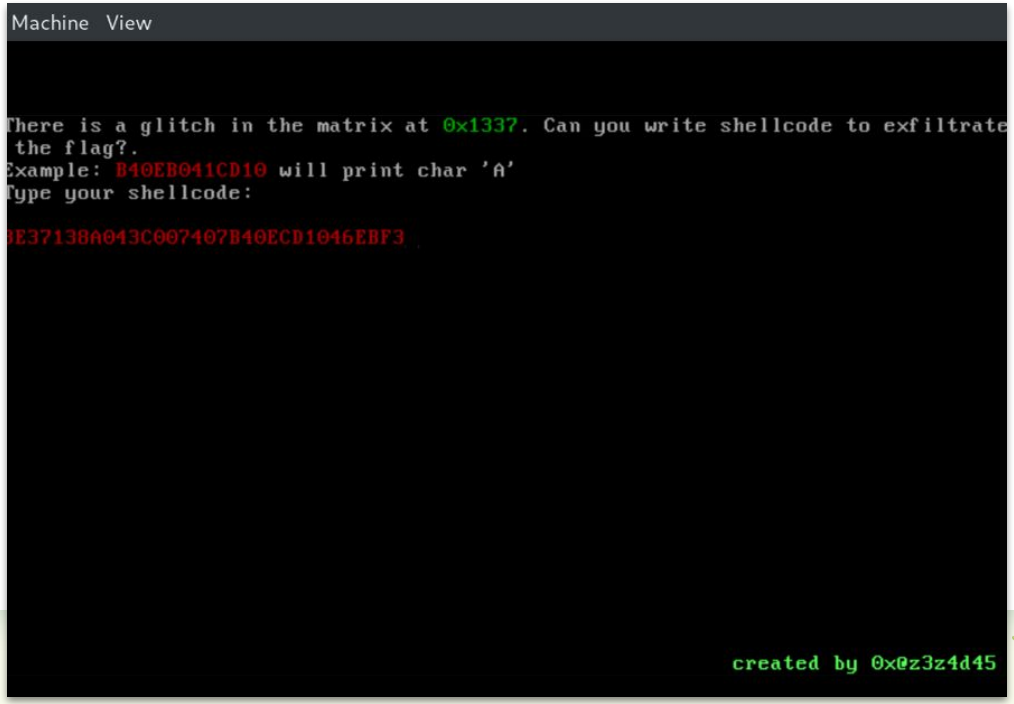
```
mov si,0x1337   //SI contains str addr
printstring:    // label print current addr chr
mov al,[si]     // move content of addr SI to AL
cmp al, 0       // cmp AL == \0
je donezadas    // if AL == \0 goto end
mov ah, 0xE     // AH=0xe means print
int 0x10        // call int to print AL
inc si          // increment mem addr, SI by 1
jmp printstring // jmp to begin of loop
donezadas:
```

ARM ARM (thumb) AArch64 Mips (32) Mips (64) PowerPC (32) PowerPC (64)
Sparc x86 (16) x86 (32) x86 (64)

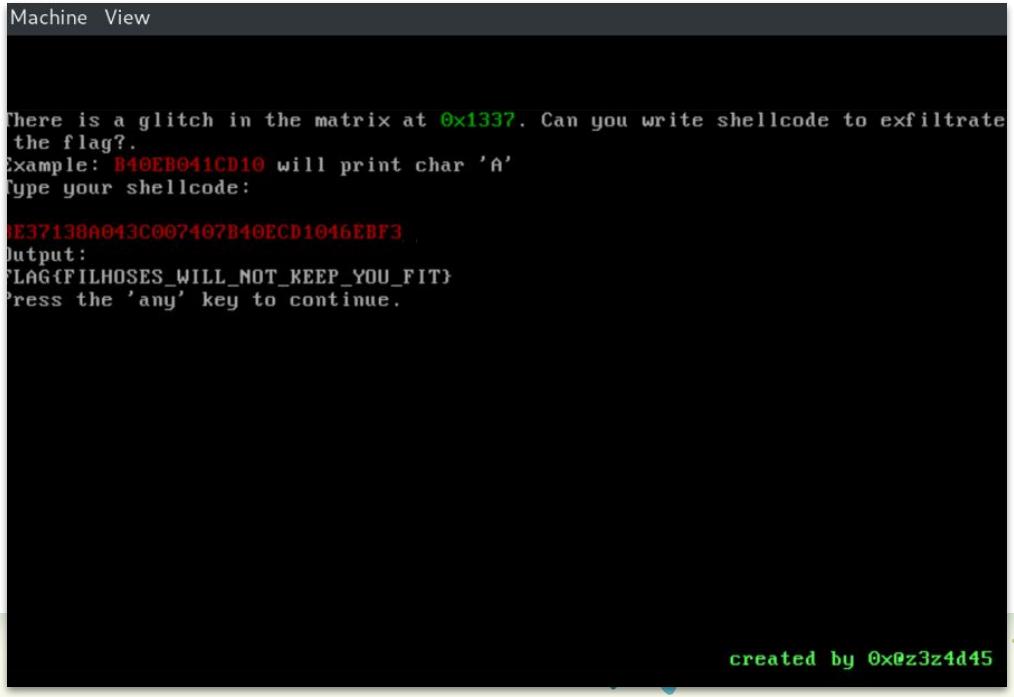
Assembly - Little Endian

```
"\xbe\x37\x13\x8a\x04\x3c\x00\x74\x07\xb4\x0e\xcd\x10\x46\xeb\xf3"
```

Glitch In The Matrix

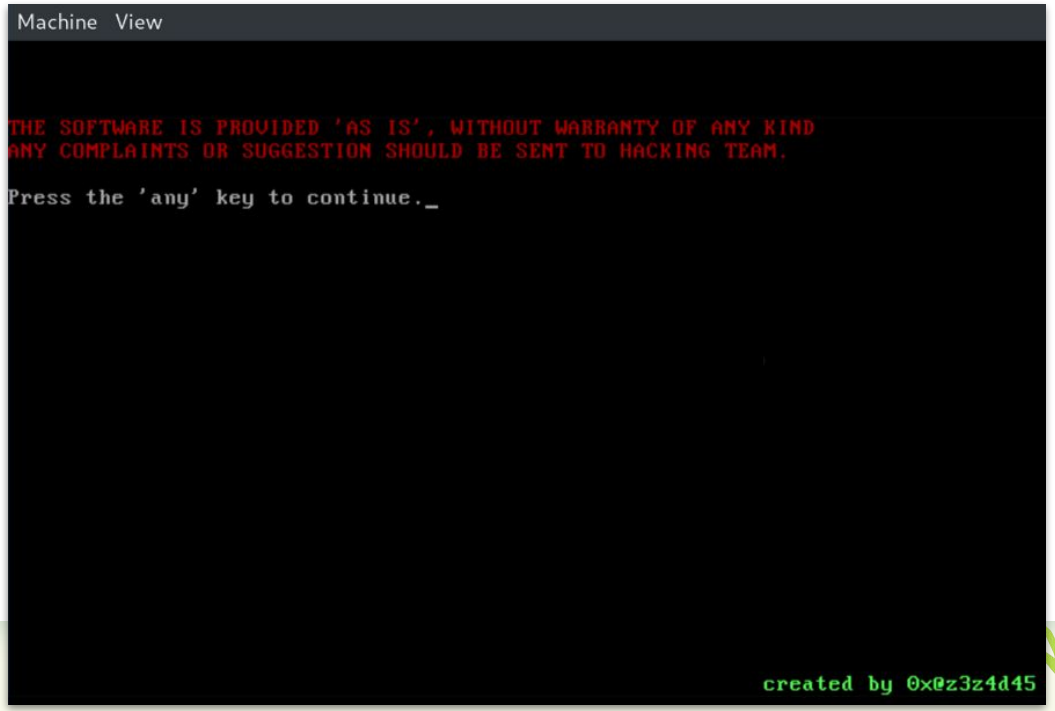


Glitch In The Matrix




FLAG{FILHOSES_WILL_NOT
_KEEP_YOU_FIT}

Help


Machine View
THE SOFTWARE IS PROVIDED 'AS IS', WITHOUT WARRANTY OF ANY KIND
ANY COMPLAINTS OR SUGGESTION SHOULD BE SENT TO HACKING TEAM.
Press the 'any' key to continue._
created by 0x0z3z4d45

Help - Email

From: Hacking Team <hacking.team@sefod.eu>

To: Rui Tinto <rui.tinto@sefod.eu>

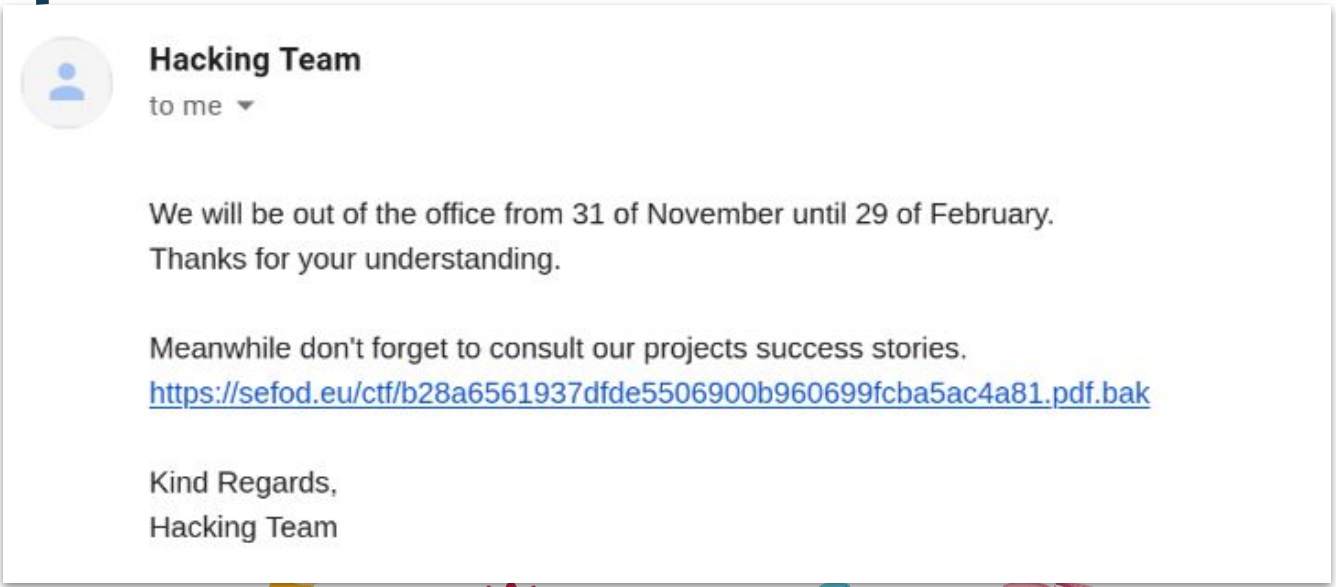
Subject: 0xOPOLEAKS

Body:

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

Hi Rui Tinto,

Help - Email



Hacking Team

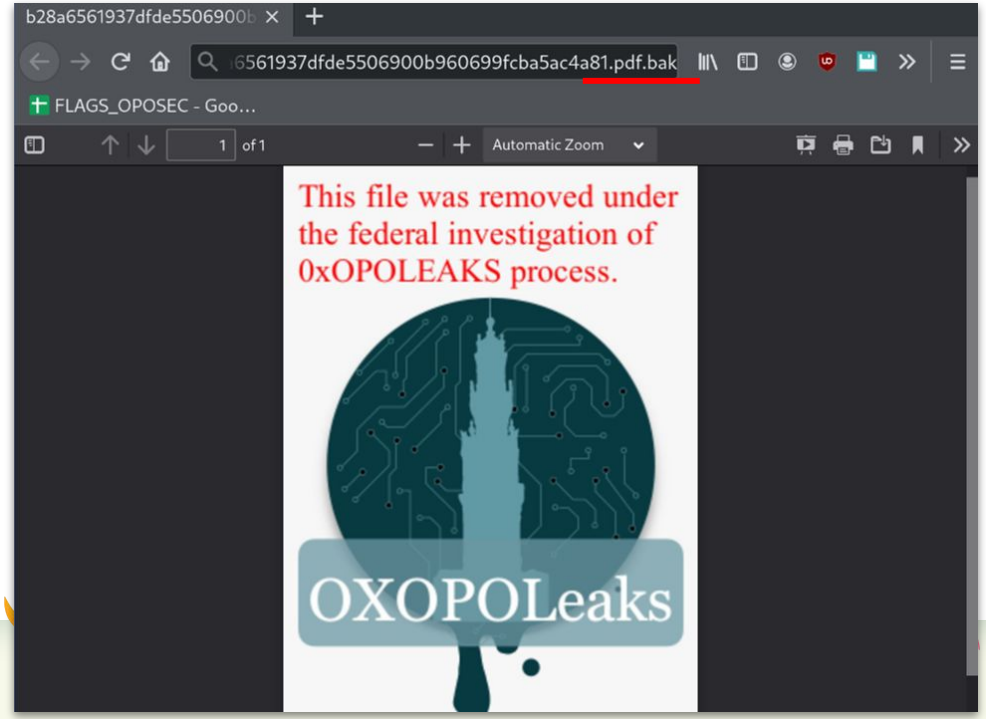
to me

We will be out of the office from 31 of November until 29 of February.
Thanks for your understanding.

Meanwhile don't forget to consult our projects success stories.
https://sefod.eu/ctf/b28a6561937dfde5506900b960699fcba5ac4a81.pdf.bak

Kind Regards,
Hacking Team

Help - Email


b28a6561937dfde5506900b
6561937dfde5506900b960699fcba5ac4a81.pdf.bak
FLAGS_OPOSEC - Goo...
1 of 1
Automatic Zoom
This file was removed under the federal investigation of 0xOPOLEAKS process.
OXOPOLeaks

Help - Email

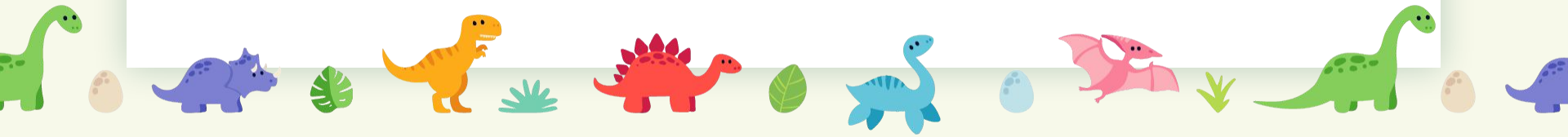

b28a6561937dfde5506900b
e5506900b960699fcba5ac4a81.pdf
FLAGS_OPOSEC - Goo...
1 of 1
Automatic Zoom
This file was removed under the federal investigation of 0xOPOLEAKS process.
OXOPOLeaks

Help - Email

```
anon /tmp/oposec 24.01.21 18:01 48% 1.29
sha1sum b28a6561937dfde5506900b960699fcba5ac4a81.pdf b28a6561937dfde5506900b960699fcba5ac4a81.pdf.bak
b28a6561937dfde5506900b960699fcba5ac4a81  b28a6561937dfde5506900b960699fcba5ac4a81.pdf
b28a6561937dfde5506900b960699fcba5ac4a81  b28a6561937dfde5506900b960699fcba5ac4a81.pdf.bak
anon /tmp/oposec 24.01.21 18:01 48% 1.24
sha256sum b28a6561937dfde5506900b960699fcba5ac4a81.pdf b28a6561937dfde5506900b960699fcba5ac4a81.pdf.bak
eba62361c9e6bdade9e850c03e3a4f86af4b2d2b3d1fff20f3b3bf3f16bd6c76  b28a6561937dfde5506900b960699fcba5ac4a81.pdf
2e79757998811bef0c8e4efc85254ceae5984c88caa2d71ef05f7e7729e4134f  b28a6561937dfde5506900b960699fcba5ac4a81.pdf.bak
anon /tmp/oposec 24.01.21 18:01 48% 1.24
```

Help - Email

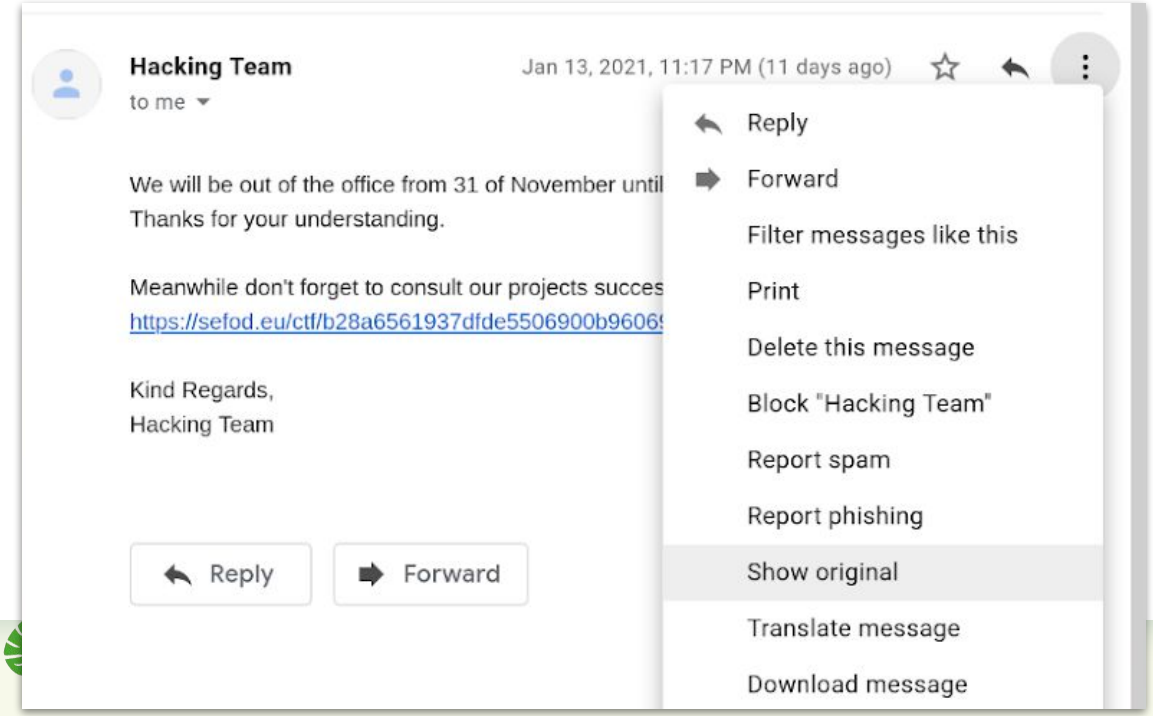
```
anon /tmp/oposec
pdfimages b28a6561937dfde5506900b960699fcba5ac4a81.pdf imagem -png
anon /tmp/oposec
file imagem-000.png
imagem-000.png: PNG image data, 834 x 1236, 8-bit/color RGB, non-interlaced
```



Help - Email - StegSolve

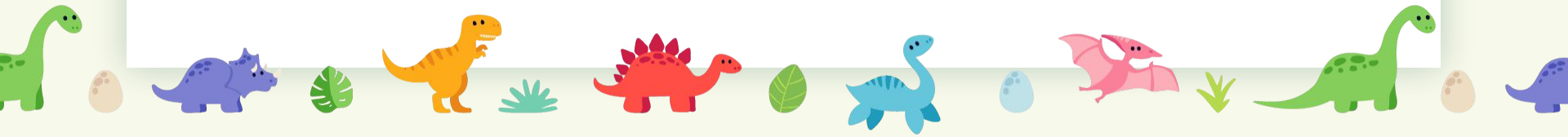

flag{LampPR3I@_De_OvoS_DO
3SN7_5WIm_ON_the_OcE@n}
Credits @Tarrinho

Help - Email


Hacking Team
Jan 13, 2021, 11:17 PM (11 days ago)
to me
We will be out of the office from 31 of November until
Thanks for your understanding.
Meanwhile don't forget to consult our projects succes
https://sefod.eu/ctf/b28a6561937dfde5506900b9606
Kind Regards,
Hacking Team
Reply
Forward
Reply
Forward
Filter messages like this
Print
Delete this message
Block "Hacking Team"
Report spam
Report phishing
Show original
Translate message
Download message

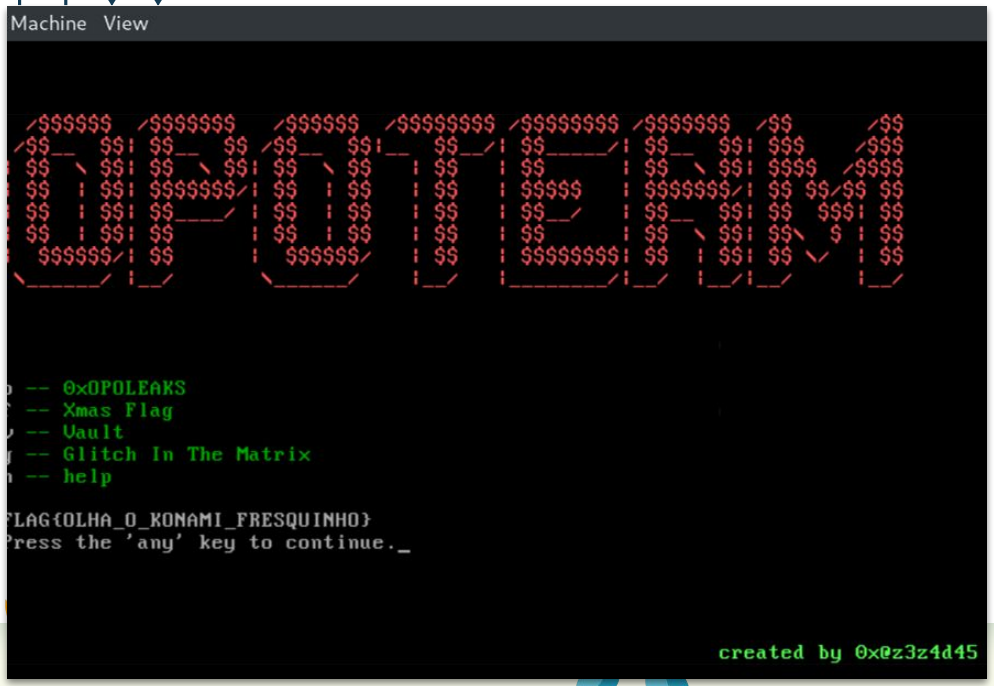
Help - Email

```
Subject: bbbb
In-Reply-To: : <CAEs0HZCBsgPQ+NaPzPLqb96ruY-+X8NCVhmk1fLQC4nNxRMm6w@mail.gmail.com>
References: <CAEs0HZCBsgPQ+NaPzPLqb96ruY-+X8NCVhmk1fLQC4nNxRMm6w@mail.gmail.com>
Flag: FLAG{ARROZ_DOCE_WARMS_YOUR_HEART}
Message-Id: <20210113231732.47BC02244E@ja.sefod.eu>
Date: Thu, 14 Jan 2021 00:17:32 +0100 (CET)
```



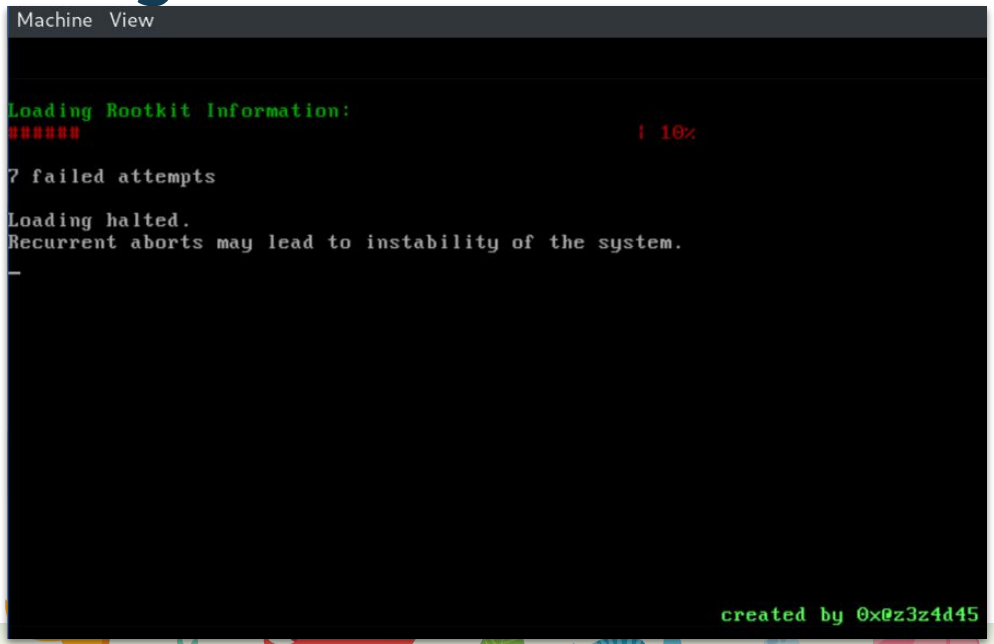

FLAG{ARROZ_DOCE_WARMS_YOUR_HEART}

Konami ↑ ↑ ↓ ↓ ← → ← → B A <ENTER>

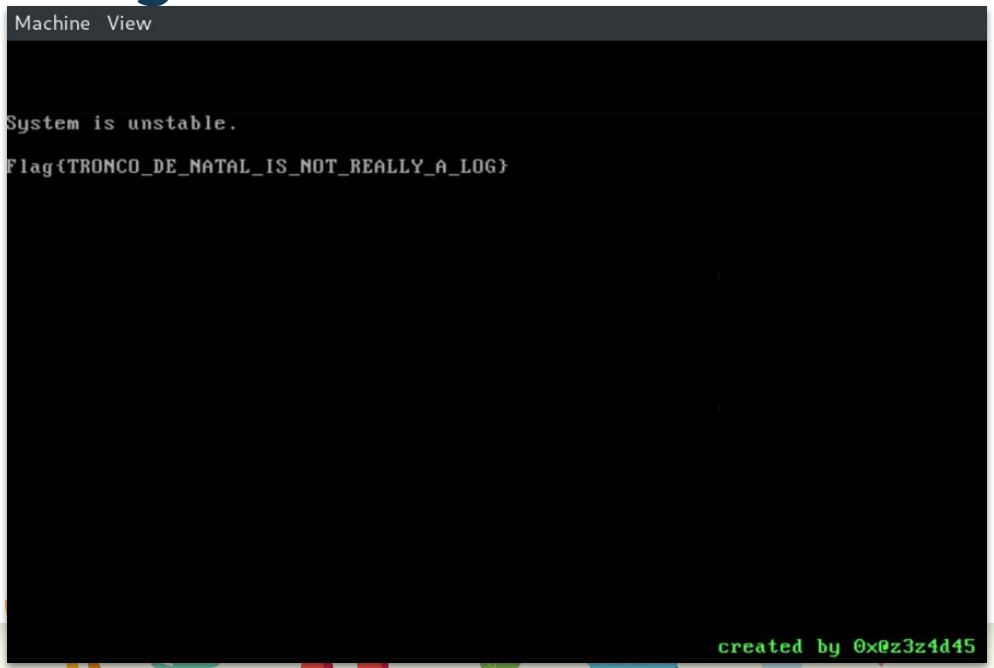



FLAG{OLHA_O_KONAMI_FR
ESQUINHO}

Crash Loading



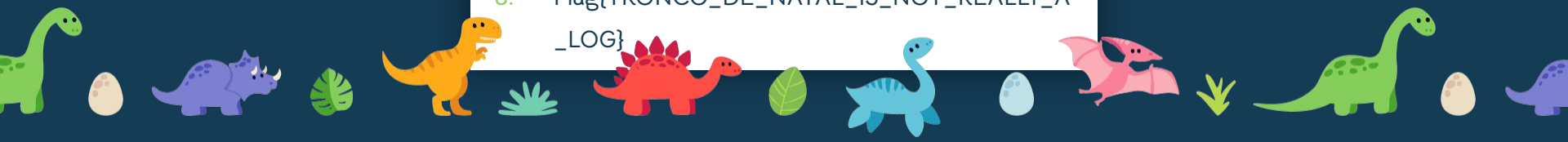
Crash Loading




Flag{TRONCO_DE_NATAL_IS_NOT_REALLY_A_LOG}

Flags

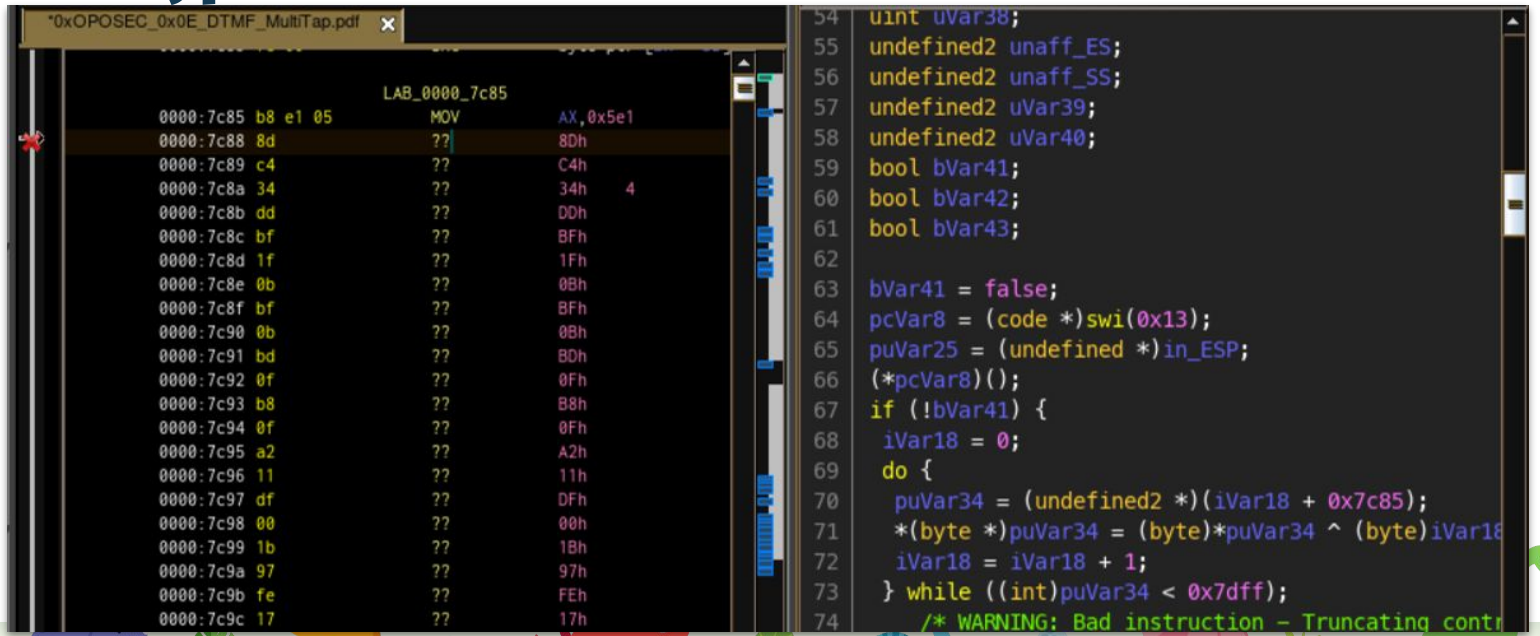
1. FLAG{COSCOROES_IS_THE_REAL_DEAL}
2. FLAG{SONHOS_WILL_MAKE_YOU_DREAM}
3. FLAG{RABANADAS_GIVE_STRENGTH_TO_BRUTEFORCE}
4. FLAG{FILHOSES_WILL_NOT_KEEP_YOU_FIT}
5. ßlag{LampPR3I@_De_OvoS_DO3SN7_5WIm_ON_the_OcE@n}
6. FLAG{ARROZ_DOCE_WARMS_YOUR_HEART}
7. FLAG{OLHA_O_KONAMI_FRESQUINHO}
8. Flag{TRONCO_DE_NATAL_IS_NOT_REALLY_A_LOG}



Unintended Way

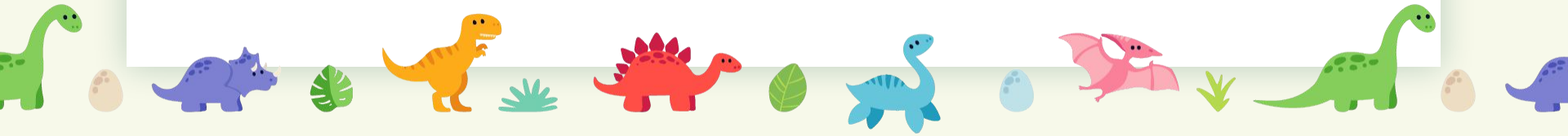
Reverse

Encrypted Bootloader



GDB + QEMU

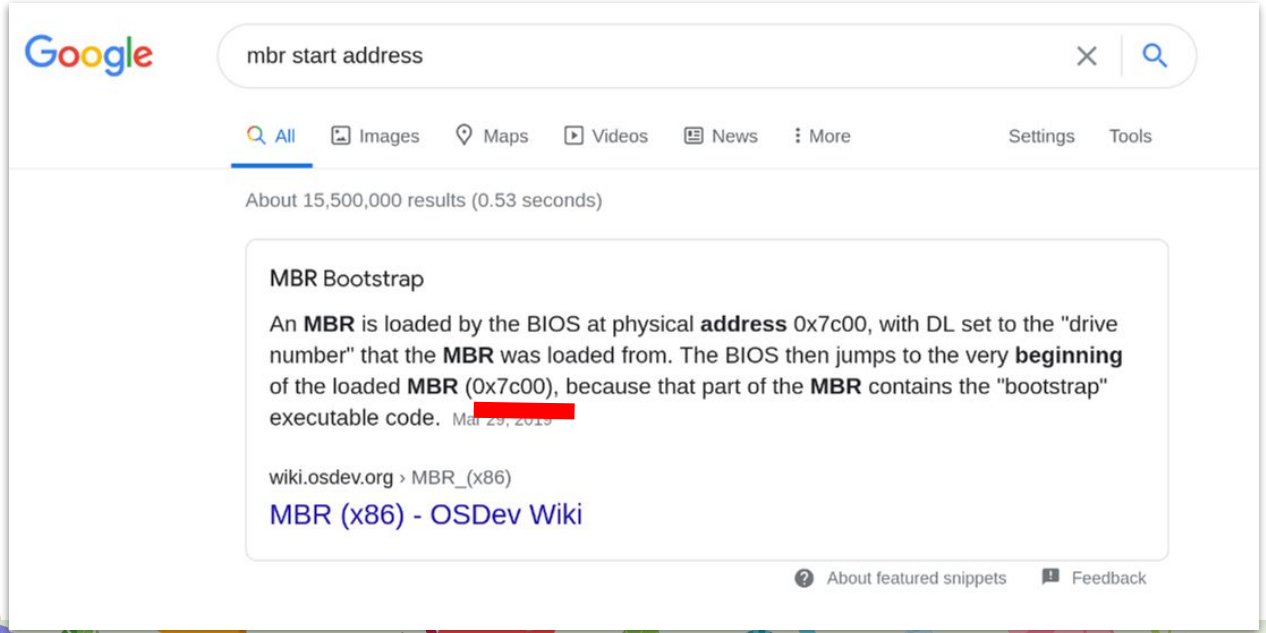
```
└─ qemu-system-i386 -fda BootableSecClinics.pdf -s -S
WARNING: Image format was not specified for 'BootableSecClinics.pdf' and prob
ing guessed raw.
         Automatically detecting the format is dangerous for raw images, writ
e operations on block 0 will be restricted.
         Specify the 'raw' format explicitly to remove the restrictions.
```



GDB + QEMU

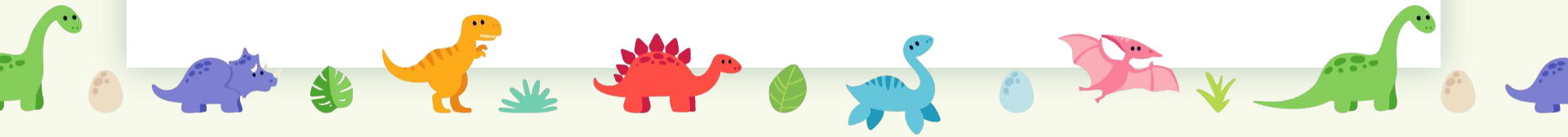
```
└─ gdb-multiarch -quiet
Exception caught while booting Guile.
Error in function "make_objcode_from_file":
bad header on object file: "\x7fELF\x02\x01\x01ÿ\x00\x00\x00\x00\x00\x00\x00\x00"
gdb-multiarch: warning: Could not complete Guile gdb module initialization from:
/usr/share/gdb/guile/gdb/boot.scm.
Limited Guile support is available.
Suggest passing --data-directory=/path/to/gdb/data-directory.
GEF for linux ready, type `gef' to start, `gef config' to configure
80 commands loaded for GDB 10.1 using Python engine 3.9
[+] Configuration from '/home/anon/.gef.rc' restored
[+] 16 extra commands added from '/home/anon/git/gef-extras/scripts'
gef➤  target remote localhost:1234
```

Reverse


Google
mbr start address
All
Images
Maps
Videos
News
More
Settings
Tools
About 15,500,000 results (0.53 seconds)
MBR Bootstrap
An MBR is loaded by the BIOS at physical address 0x7c00, with DL set to the "drive number" that the MBR was loaded from. The BIOS then jumps to the very beginning of the loaded MBR (0x7c00), because that part of the MBR contains the "bootstrap" executable code.
wiki.osdev.org › MBR_(x86)
MBR (x86) - OSDev Wiki
About featured snippets
Feedback

GDB - Breakpoint Start Address

```
gef➤  b *0x7c00
Breakpoint 1 at 0x7c00
gef➤  c
```

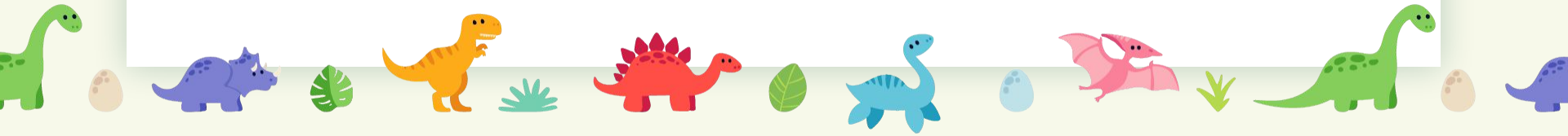


GDB - Bootloader Unencrypted Stub

```
gef➤  x/2i 0x7c00
=> 0x7c00:      and    ax,0xffff
   0x7c03:      jmp    0x7c5c
gef➤  x/10i 0x7c5c
   0x7c5c:      xor    ax,ax
   0x7c5e:      mov    ds,ax
   0x7c60:      xor    dl,dl
   0x7c62:      int    0x13
   0x7c64:      jb     0x7d85
   0x7c68:      xor    cx,cx
   0x7c6a:      mov    bx,0x7c85
   0x7c6d:      add    bx,cx
   0x7c6f:      mov    ax,WORD PTR [bx]
   0x7c71:      xor    al,cl
gef➤
```

GDB - Dump Bootloader

```
gef➤  dump binary memory bootloader.bin 0x7c00 0x7f00
gef➤  !file bootloader.bin
bootloader.bin: DOS/MBR boot sector
```

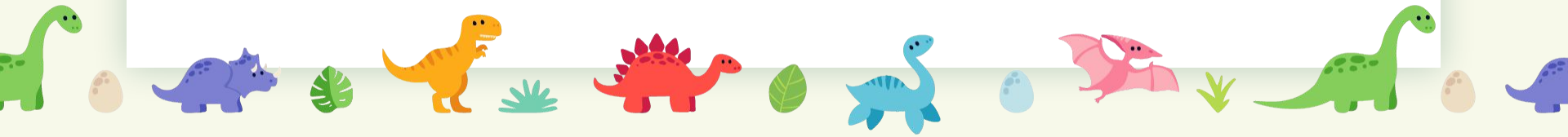


Reverse - Decryption Routine

```
                    LAB_0000_7c5c                                   XREF[1]:    0000:7c03(j)
0000:7c5c 31 c0         XOR         AX,AX
0000:7c5e 8e d8         MOV         DS,AX
0000:7c60 30 d2         XOR         DL,DL
0000:7c62 cd 13         INT         0x13
0000:7c64 0f 82 1d 01   JC          reboot
0000:7c68 31 c9         XOR         CX,CX                                   DEBOFUSCATE

                    DEOBFUSCATE
                    loop                                            XREF[1]:    0000:7c7c(j)
0000:7c6a bb 85 7c      MOV         BX,0x7c85                           start addr obfuscated
0000:7c6d 01 cb         ADD         BX,CX                               add increment to BX
0000:7c6f 8b 07         MOV         AX,word ptr [BX]=>OBFUSCATED        get obfuscated flag
0000:7c71 30 c8         XOR         AL,CL                               xor obfuscated byte with xor key
0000:7c73 88 07         MOV         byte ptr [BX]=>OBFUSCATED,AL        store deobfuscated byte on mem
0000:7c75 83 c1 01      ADD         CX,0x1                              add 1 to increment var
0000:7c78 81 fb fe 7d   CMP         BX,0x7dfe                           compare if reached end of obfuscat
0000:7c7c 7e ec         JLE         loop                                goto begin of loop
0000:7c7e eb 05         JMP         OBFUSCATED                          jump to deobsfucated memory
0000:7c80 ba be ca fe   ds          BA,BE,CA,FE                         marker for external obfuscator
          00
                    END DEOBFUSCATE

                    OBFUSCATED                                      XREF[3]:    0000:7c6f(R), 0000:7c73(W),
                                                                                0000:7c7e(j)
0000:7c85 b8            ??          B8h
0000:7c86 e1            ??          E1h
0000:7c87 05            ??          05h
```

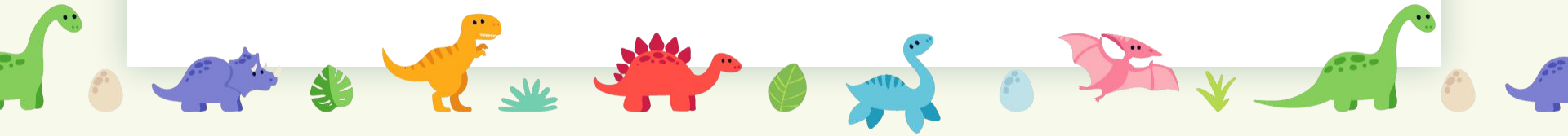
GDB - Breakpoint 1st Decrypted Stub

```
gef➤  b *0x7c85
Breakpoint 4 at 0x7c85
gef➤  c
```



GDB - Dump Bootloader

```
gef➤  dump binary memory bootloader2.bin 0x7c00 0x7f00
gef➤  !file bootloader2.bin
bootloader2.bin: data
gef➤
```



Reverse - Decrypted Bootloader

```
          00

                    LAB_0000_7c85                                   XREF[1]:     0000:7c7e(j)
0000:7c85 b8 e0 07        MOV         AX,0x7e0
0000:7c88 8e c0           MOV         ES,AX
0000:7c8a 31 db           XOR         BX,BX
                    LAB_0000_7c8c+2                                 XREF[0,1]:   0000:7d07(W)
0000:7c8c b8 17 02        MOV         AX,0x217
0000:7c8f b5 00           MOV         CH,0x0
0000:7c91 b1 02           MOV         CL,0x2
0000:7c93 b6 00           MOV         DH,0x0
0000:7c95 b2 00           MOV         DL,0x0
0000:7c97 cd 13           INT         0x13
0000:7c99 0f 82 e8 00     JC          LAB_0000_7d85
0000:7c9d b9 57 7c        MOV         CX,0x7c57
0000:7ca0 b8 a0 7c        MOV         AX,offset DAT_0000_7c1b                    = 3Ch     <

                    LAB_0000_7ca3                                   XREF[1]:     0000:7cac(j)
0000:7ca3 b2 90           MOV         DL,0x90
0000:7ca5 89 cb           MOV         BX,CX
0000:7ca7 88 17           MOV         byte ptr [BX]=>DAT_0000_7c57,DL            = 0Bh
0000:7ca9 41              INC         CX
0000:7caa 39 c1           CMP         CX,AX
0000:7cac 7c f5           JL          LAB_0000_7ca3
```

Reverse - LongJump To Kernel Address

```
                    LAB_0000_7d03                                   XREF[1]:     0000:7d0c(j)
0000:7d03 b2 90         MOV     DL,0x90
0000:7d05 89 cb         MOV     BX,CX
0000:7d07 88 17         MOV     byte ptr [BX]=>LAB_0000_7c8c+2,DL
0000:7d09 41            INC     CX
0000:7d0a 39 c1         CMP     CX,AX
0000:7d0c 7c f5         JL      LAB_0000_7d03
0000:7d0e ea 00 00 e0   JMPF    LAB_0000_7e00                                   KERNEL ADDR
          07
0000:7d13 e8            ??      E8h
```

GDB - Encrypted Kernel

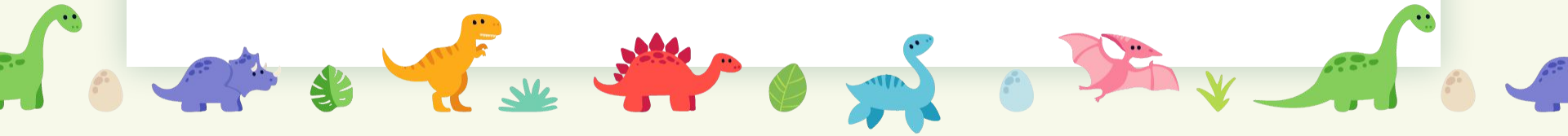
```
gef➤  x/20i 0x7e00
   0x7e00:      add    BYTE PTR [bx+si],al
   0x7e02:      add    BYTE PTR [bx+si],al
   0x7e04:      add    BYTE PTR [bx+si],al
   0x7e06:      add    BYTE PTR [bx+si],al
   0x7e08:      add    BYTE PTR [bx+si],al
   0x7e0a:      add    BYTE PTR [bx+si],al
   0x7e0c:      add    BYTE PTR [bx+si],al
   0x7e0e:      add    BYTE PTR [bx+si],al
   0x7e10:      add    BYTE PTR [bx+si],al
```

Reverse - Kernel Decryption Stub

```
                    LAB_0000_7cb0                                   XREF[1]:
0000:7cb0 89 cb           MOV         BX,CX
0000:7cb2 81 c3 00 7e     ADD         BX,offset LAB_0000_7d7b
0000:7cb6 a1 52 7c        MOV         AX,[s__0000_7c52]
0000:7cb9 8a 17           MOV         DL,byte ptr [BX]=>LAB_0000_7d7b
0000:7cbb 30 c2           XOR         DL,AL
0000:7cbd 30 ca           XOR         DL,CL
0000:7cbf 88 17           MOV         byte ptr [BX]=>LAB_0000_7d7b,DL
0000:7cc1 41              INC         CX
0000:7cc2 8a 57 01        MOV         DL,byte ptr [BX + 0x1]=>LAB_0000_7d7b+1
0000:7cc5 30 e2           XOR         DL,AH
0000:7cc7 30 ca           XOR         DL,CL
0000:7cc9 88 57 01        MOV         byte ptr [BX + 0x1]=>LAB_0000_7d7b+1,DL
0000:7ccc 41              INC         CX
0000:7ccd a1 54 7c        MOV         AX,[s__0000_7c52+2]
0000:7cd0 8a 57 02        MOV         DL,byte ptr [BX + 0x2]=>LAB_0000_7d7d
0000:7cd3 30 c2           XOR         DL,AL
0000:7cd5 30 ca           XOR         DL,CL
0000:7cd7 88 57 02        MOV         byte ptr [BX + 0x2]=>LAB_0000_7d7d,DL
0000:7cda 41              INC         CX
0000:7cdb 8a 57 03        MOV         DL,byte ptr [BX + 0x3]=>LAB_0000_7d7e
0000:7cde 30 e2           XOR         DL,AH
0000:7ce0 30 ca           XOR         DL,CL
0000:7ce2 88 57 03        MOV         byte ptr [BX + 0x3]=>LAB_0000_7d7e,DL
0000:7ce5 41              INC         CX
0000:7ce6 81 f9 00 30     CMP         CX,0x3000
0000:7cea 7e c4           JLE         LAB_0000_7cb0
0000:7cec b9 00 7c        MOV         CX,0x7c00
0000:7cef b8 ef 7c        MOV         AX,offset LAB_0000_7c6a
```

GDB - Breakpoint kernel

```
gef➤  b *0x7e00
Breakpoint 3 at 0x7e00
gef➤  c
```



GDB - Unencrypted Kernel

```
gef➤  x/20i 0x7e00
   0x7e00:      push   bp
   0x7e01:      mov    bp,sp
   0x7e03:      push   di
   0x7e04:      push   si
   0x7e05:      call   0x7e0c
   0x7e08:      pop    si
   0x7e09:      pop    di
   0x7e0a:      pop    bp
   0x7e0b:      ret
   0x7e0c:      push   bp
   0x7e0d:      mov    bp,sp
```

GDB - Dump Kernel

```
gef➤  dump binary memory kernel.bin 0x7e00 0xf000
gef➤  !file kernel.bin
kernel.bin: data
gef➤  !binwalk kernel.bin --opcodes

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
------
0             0x0             Intel x86 instructions, function prologue
12            0xC             Intel x86 instructions, function prologue
64            0x40            Intel x86 instructions, function prologue
196           0xC4            Intel x86 instructions, function prologue
405           0x195           Intel x86 instructions, function prologue
430           0x1AE           Intel x86 instructions, function prologue
487           0x1E7           Intel x86 instructions, function prologue
543           0x21F           Intel x86 instructions, function prologue
585           0x249           Intel x86 instructions, function prologue
613           0x265           Intel x86 instructions, function prologue
```

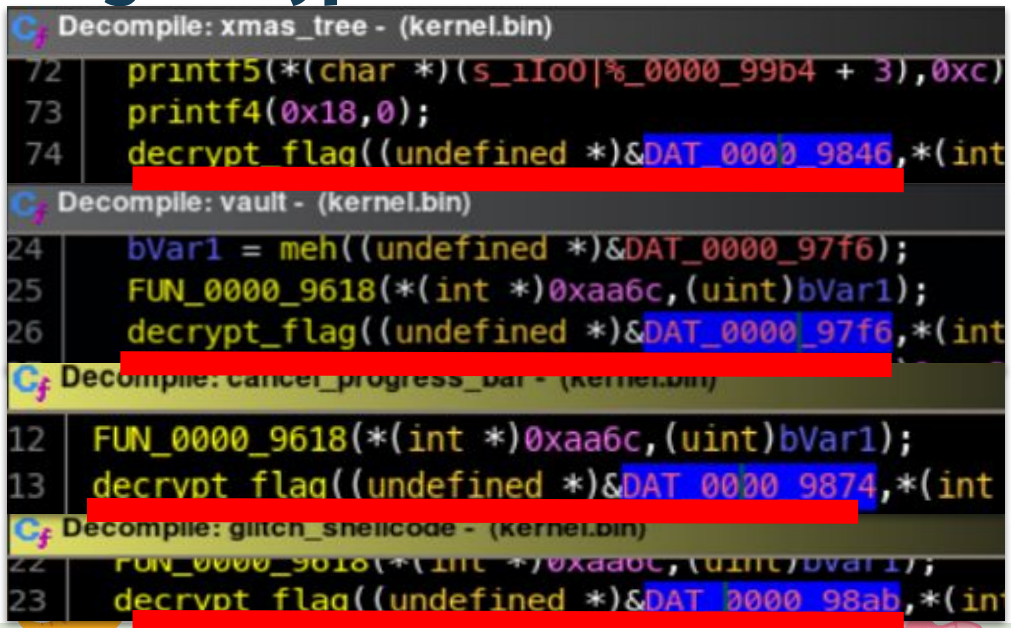
Reverse - Kernel

```
                             //
     assume DF = 0x0  (Default)
0000:7c00 55                 PUSH        BP
0000:7c01 89 e5              MOV         BP,SP
0000:7c03 57                 PUSH        DI
0000:7c04 56                 PUSH        SI
0000:7c05 e8 04 00           CALL        FUN_0000_7c0c                                    undefined
0000:7c08 5e                 POP         SI
0000:7c09 5f                 POP         DI
0000:7c0a 5d                 POP         BP
0000:7c0b c3                 RET

                             *************************************************************
                             *                          FUNCTION                         *
                             *************************************************************
                             undefined __stdcall16near FUN_0000_7c0c(void)
     undefined                   AL:1           <RETURN>
                             FUN_0000_7c0c                                 XREF[1]:     0000:7c05(c)
0000:7c0c 55                 PUSH        BP
0000:7c0d 89 e5              MOV         BP,SP
0000:7c0f 57                 PUSH        DI
0000:7c10 56                 PUSH        SI
0000:7c11 e8 af 18           CALL        SUB_0000_94c3
0000:7c14 e8 97 01           CALL        FUN_0000_7dae                                    undefined
0000:7c17 e8 02 08           CALL        FUN_0000_841c                                    undefined
0000:7c1a e8 9a 16           CALL        SUB_0000_92b7
0000:7c1d e8 97 16           CALL        SUB_0000_92b7
0000:7c20 e8 94 16           CALL        SUB_0000_92b7
0000:7c23 e8 91 16           CALL        SUB_0000_92b7
```

Reverse Flag Decryption

```
*************************************************************************
undefined __cdecl16near decrypt_flag(undefined * param_1, int para...
    AL:1            <RETURN>
    Stack[0x2]:2    param_1
    Stack[0x4]:2    param_2
decrypt_flag                                    XREF[4]:    xmas_tree:0000:8c84(c),
                                                            vault:0000:8f93(c),
                                                            cancel_progress_bar:0000:92de(c),
                                                            glitch_shellcode:0000:93a1(c)
    PUSH        BP
    MOV         BP,SP
```

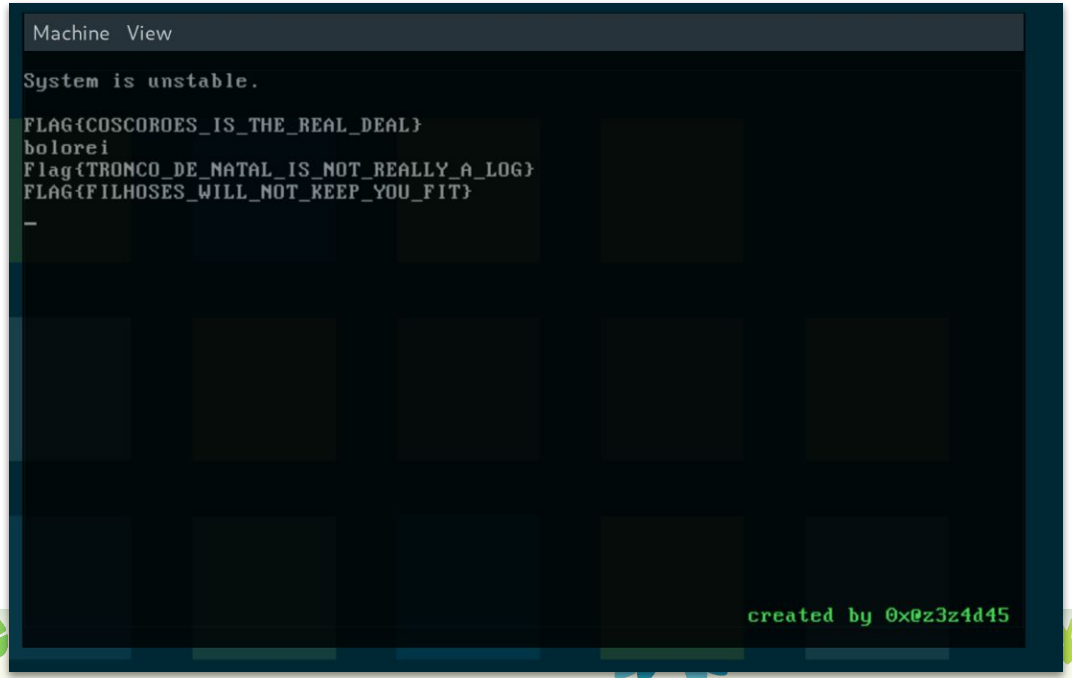
Reverse Flag Decryption References



Scripting With GDB

```
target remote localhost:1234
set tdesc filename /home/anon/git/MBRHelloWorld/bsos/target.xml
# break on banner
b *0x861c
c
#bypass banner to rootkit login
set $eip=0x880a-$cs*0x10
b *0x8839
#skip login
set $eip=0x883b-$cs*0x10
#skip boot counter
b *0x96ab
c
#skip count verification to print flags and stuff
set $eip=0x96ad-$cs*0x10
#intercept decrypt method and change args
b *0x92de
c
########set $flag=*(short *)($sp)
#replace obfs flag addr
set {short}($sp) =0x9846
#last method replace to begin of decrypt
del
b *0x9313
c
#jump again to decrypt flag
set $eip=0x92d3-$cs*0x10
b *0x92de
c
```

Reverse

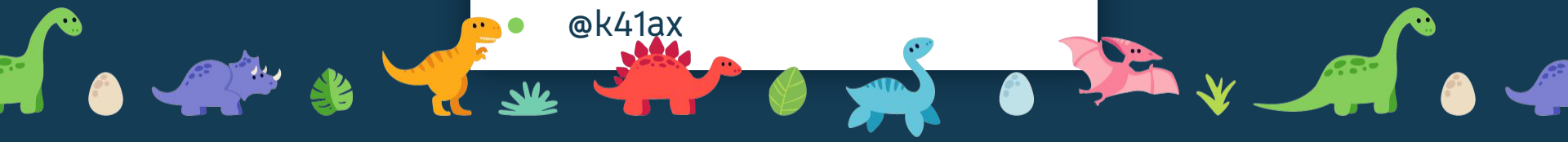

Machine View
System is unstable.
FLAG{COSCOROES_IS_THE_REAL_DEAL}
bolorei
Flag{TRONCO_DE_NATAL_IS_NOT_REALLY_A_LOG}
FLAG{FILHOSES_WILL_NOT_KEEP_YOU_FIT}
created by 0x0z3z4d45

First Blood

Flags	Solvers
COSCOROES	@norwat
SONHOS	@nunohumberto
RABANADAS	@nunohumberto
FILHOSES	@nunohumberto
LampPR3I@	@nunohumberto
ARROZ_DOCE	@nunohumberto
KONAMI	@norwat
TRONCO	@nunohumberto

Solvers

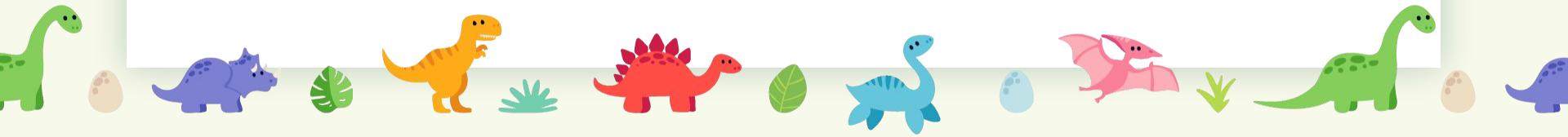
- @nunohumberto (First Blood!)
- @jp
- @mluis
- @norwat
- @ines
- @miguelpduarte
- @ArmySick
- @aap
- @jpdias
- @k41ax



Developing Bootloader

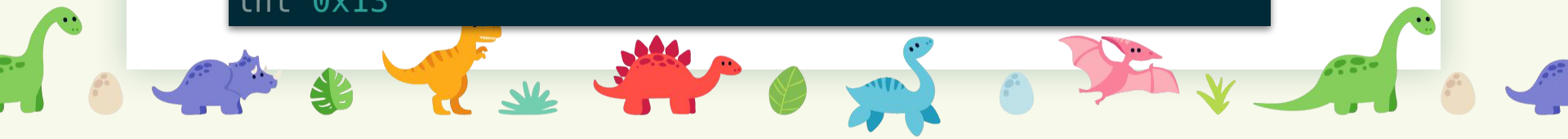
Probing Hw

```
start:

  xor ax, ax» » » ;Zero AX
  mov ds, ax» » » ;Zero DS
  xor dl, dl» » » ;Zero DL
  int 0x13        ;Reset Disk System
  jc reboot       ;jump if condition match
```



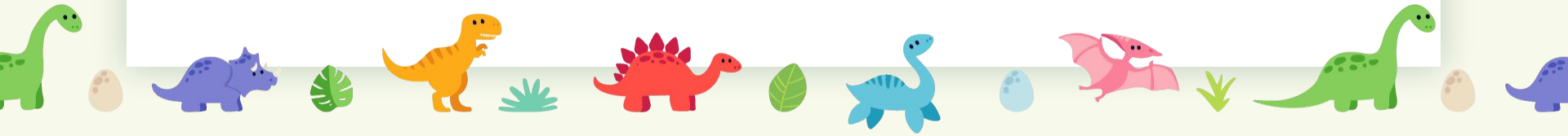
Read Disk

```
mov ax, 0x7e0»» ;Load to 07E0:0000 -> 0x7E00

mov es, ax» » »
xor bx, bx

;; mov ax, 0x0201  ;Read (02) One (01) sector.
mov ax, 0x0217  ;Read (02) 0x17=23 sectors.
mov ch, 0       ;Cylinder 0
mov cl, 2       ;Sector   2 (These begin at 1 not 0.)
mov dh, 0       ;Head     0 (Top side of a floppy.)
mov dl, 0       ;Lecteur de disquettes
int 0x13
```



LongJump

```
;;;;;;;;;;;;;;;;;;;;;;;;
;;JUMP TO KERNELIO
db 0xea                   ;JMP to kernel
dw 0, 0x07E0» » ;at 0x7E00
```



Kernel - C

```
void main();
void entrypoint() {
  main();
}

void main() {

  clearBootloader();

  cls();
  SkullBanner();
  delay();
```

Code

bcc(1) - Linux man page

Name

bcc - Bruce's C compiler

Synopsis

bcc [**-03EGNOPSVcegvwxW**] [**-Aas_option**] [**-Bexecutable_prefix**] [**-Ddefine**] [**-Uundef**] [**-Mc_mode**] [**-o outfile**] [**-ansi**] [**-Ccc1_option**] [**-Pcpp_option**] [**-Iinclude_dir**] [**-Lld_option**] [**-Ttmpdir**] [**-Qc386_option**] [**-ttext_segno**] [**ld_options**] [**infiles**]

Description

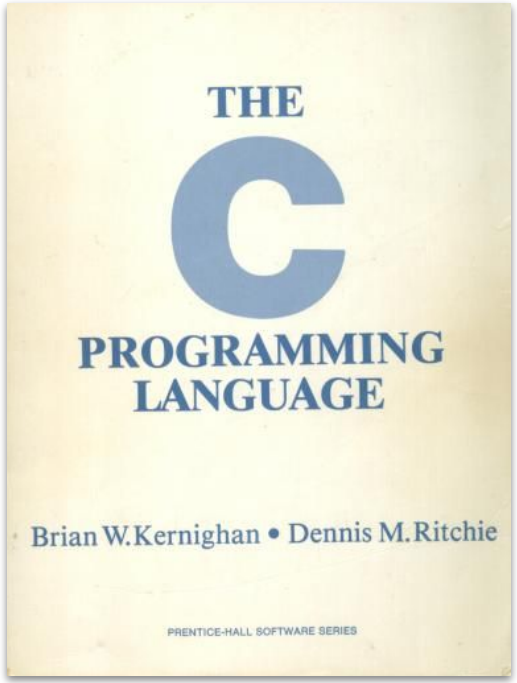
Bcc is a simple C compiler that produces 8086 assembler, in addition compiler compile time options allow 80386 or 6809 versions. The compiler understands traditional K&R C with just the restriction that bit fields are mapped to one of the other integer types.

The default operation is to produce an 8086 executable called **a.out** from the source file.

Options

-ansi

K&R



In 1978, Brian Kernighan and Dennis Ritchie published the first edition of The C Programming Language

Printing Challenge

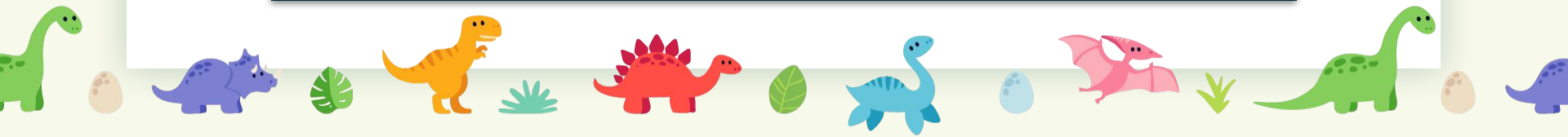
Print Char

```
static asm_putc(c){
#asm
   mov»ah,#$0E
   mov»bx,#7
   int»$10
```

Function	Function Code	Parameters
Teletype Output	AH=0Eh	AL = Character, BH = Page Number, BL = Color (only in graphic mode)

Print String

```
void print(c) char *c; {
  while(*c){
    asm_putc(*c++);
  }
}
```



Print Color

```
static asm_colour(c){
#asm
#if __FIRST_ARG_IN_AX__
  mov»bx,ax
  mov»ah,#$09
  mov»cx,#1
  int»$10
```

List of BIOS color attributes

Dec	Hex	Binary	Color
0	0	0000	Black
1	1	0001	Blue
2	2	0010	Green
3	3	0011	Cyan
4	4	0100	Red
5	5	0101	Magenta
6	6	0110	Brown
7	7	0111	Light Gray
8	8	1000	Dark Gray
9	9	1001	Light Blue
10	A	1010	Light Green
11	B	1011	Light Cyan
12	C	1100	Light Red
13	D	1101	Light Magenta
14	E	1110	Yellow
15	F	1111	White

Function	Function Code	Parameters
Write character and attribute at cursor position	AH=09h	AL = Character, BH = Page Number, BL = Color, CX = Number of times to print character

GetCursor

```
int getCursorPosition(){
#asm
   mov ah, #0x3
   mov bh, #0x0
   int 0x10
   mov ax,dx
```

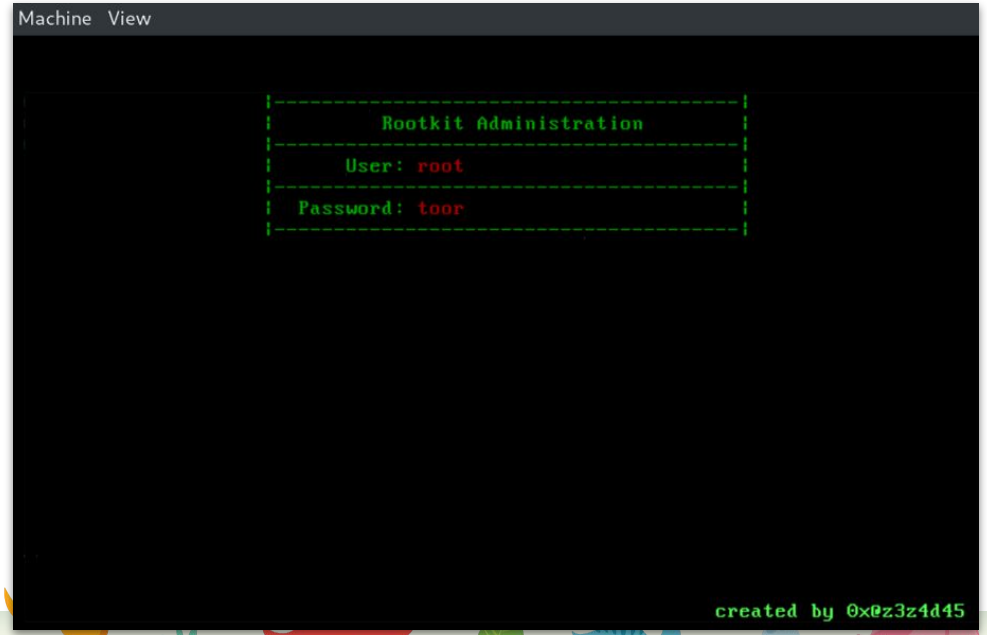
Function	Function Code	Parameters	Return
Get cursor position and shape	AH=03h	BH = Page Number	AX = 0, CH = Start scan line, CL = End scan line, DH = Row, DL = Column

SetCursor

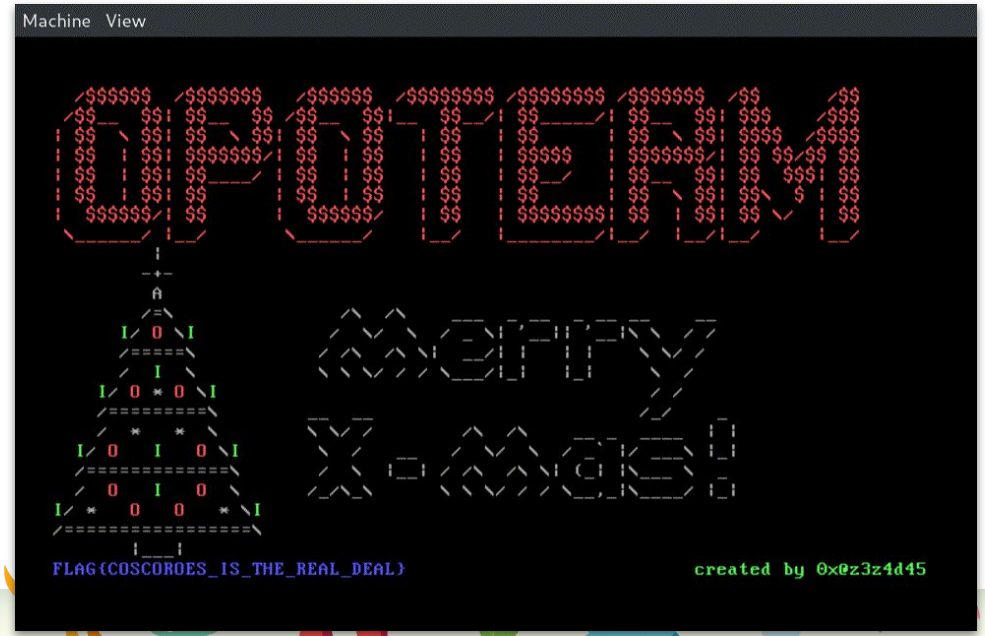
```
void setCursorPosition(c,d)char c;char d;{
#asm
  push    bp
  mov     bp, sp
  push    di
  push    si
  mov     al, [bp+4]
  mov     dh, al
  mov     al, [bp+6]
  mov     dl, al

  mov bh,#0  //page num
  mov ah,#2
  int 0x10
```

Function	Function Code	Parameters
Set cursor position	AH=02h	BH = Page Number, DH = Row, DL = Column

Noobie Login

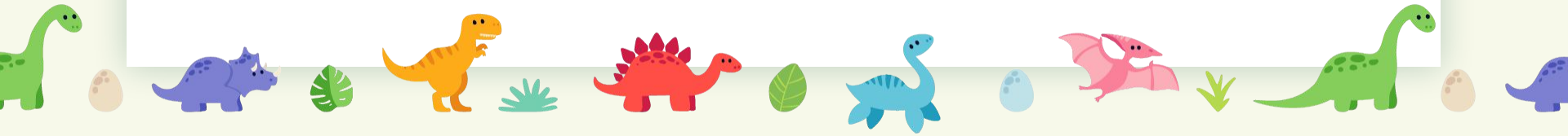


XMas Flag


Machine View
FLAG{COSCOROES_IS_THE_REAL_DEAL}
created by 0x@z3z4d45

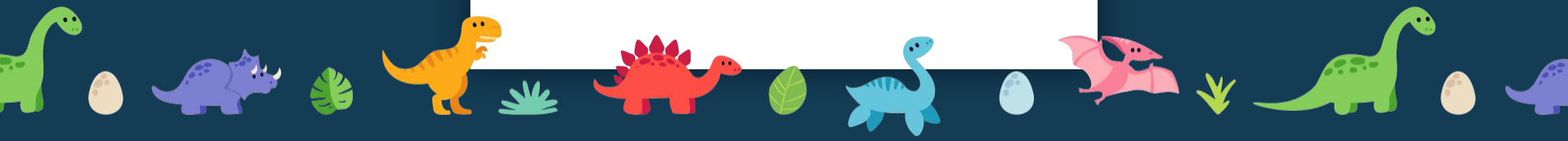
No LIBC

```
int strcmp(d, s)char *d; char *s;{
    register char *s1 = (char *) d, *s2 = (char *) s,
    c1, c2;
    while ((c1 = *s1++) == (c2 = *s2++) && c1);
    return c1 - c2;
}
```



Ported Methods

- strcmp
- strlen
- imod
- itoa
- strcpy
- memcpy



Polymorph Feature

Polymorph/Read

```
void readSector(){
#asm
  mov ax, #0x7c0» » ;Load to 07E0:0000 -> 0x7E00

  mov es, ax» » »
  xor bx, bx

  mov ah, #0x02     ;Interrupt 13h,2 => Read disk sectors
  mov al, #0x01     ;how many sectors to read
  mov ch, #0        ;Cylinder 0
  mov cl, #1        ;Sector   2 (These begin at 1 not 0.)
  mov dh, #0        ;Head     0 (Top side of a floppy.)
  mov dl, #0        ;Lecteur de disquettes
  int 0x13
```

Function	Function Code	Parameters
Read Sectors From Drive	AH = 02h	Complete all params

Polymorph/Write

```
void writeSector(){
#asm
  mov ax, #0x7c0» » ;Load to 07E0:0000 -> 0x7E00

  mov es, ax» » »
  xor bx, bx

  mov ah, #0x03     ;Interrupt 13h,2 => Read disk sectors
  mov al, #0x01     ;how many sectors to read
  mov ch, #0        ;Cylinder 0
  mov cl, #1        ;Sector   2 (These begin at 1 not 0.)
  mov dh, #0        ;Head     0 (Top side of a floppy.)
  mov dl, #0        ;Lecteur de disquettes
  int 0x13
```

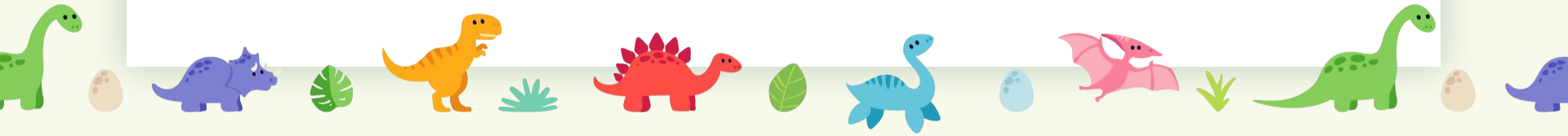
Function	Function Code	Parameters
Read Sectors From Drive	AH = 03h	ES:BX = Buffer Address Pointer

Update Canceled Boots

```
int polymorph(){
    //read from disk to memory. can't use content\
    in memory because disk is packed and content\
    in ram is unpacked
    readSector();
    //point char address to 0x7c00
    counterValue = 0x7c00;
    //get value of counter reserved byte in disk
    counterPolymorph=counterValue[0x5b];
    //increment counter
    counterPolymorph++;
    //increment counter
    counterValue[0x5b]=counterPolymorph;
    //write changes to disk
    writeSector();
    clearBootloader();
    if (counterPolymorph>7){
```

Checksum

```
anon /tmp/oposec
md5sum 0xOPOSEC_0x0E_DTMF_MultiTap.pdf
7550b21d2452aed5fc3729fc1a6b4ef0  0xOPOSEC_0x0E_DTMF_MultiTap.pdf
anon /tmp/oposec
qemu-system-i386 -fda 0xOPOSEC_0x0E_DTMF_MultiTap.pdf 2&> /dev/null
^C%
anon /tmp/oposec
md5sum 0xOPOSEC_0x0E_DTMF_MultiTap.pdf
25a3344f6810b3a219049922644135c2  0xOPOSEC_0x0E_DTMF_MultiTap.pdf
```



Extended Keys

Get Keys & Extended Keys

```
char getKey(){
#asm
   xor ax,ax
   int 0x16
```

Function	Function Code	Return
Read key press	AH=00h	AH = Scan code of the key pressed dow AL = ASCII character of the button pressed

Get Keys & Extended Keys

```
char isThereRegularKey( ){
#asm
  mov»ah,#0x01
  int»$16
  jz» nokey
  cmp»ax,#0
  jnz»dort
  mov»ax,#3
```

Function	Function Code	Return
Read key press	AH=00h	ZF = 0 if a key pressed (even Ctrl-Break) || AX = 0 if no scan code is available || AH = scan code || AL = ASCII character or zero if special function key

Get Regular Keys & Extended Keys

```
//getExtendedKey
char getExtendedKey( ){
#asm
  xor ax,ax
  mov ah,#0x10
  int 0x16
  mov al,ah
```

Function	Function Code	Return
Read expanded keyboard character	AH=10h	AL = ASCII character code AH = Scan code

Get Regular Keys & Extended Keys

```
char isThereExtendedKey( ){
#asm
  mov»ah,#0x11
  int»$16
  jz» nokeyex
  cmp»ax,#0
  jnz»dortex
  mov»ax,#3
```

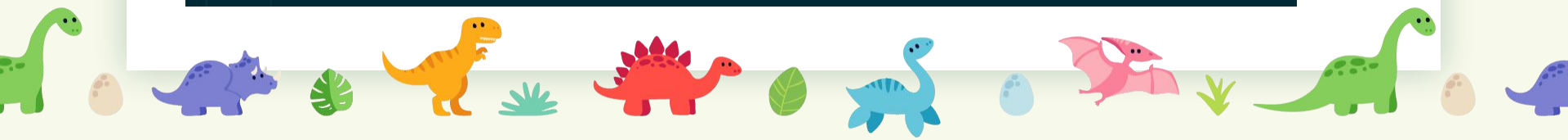
Function	Function Code	Return
Obtain status of the expanded keyboard buffer	AH=11h	ZF = 0 if key pressed (data waiting) ||AX = 0 if no scan code is available || AH = scan code || AL = ASCII character or zero if special function key

Get Regular Keys & Extended Keys

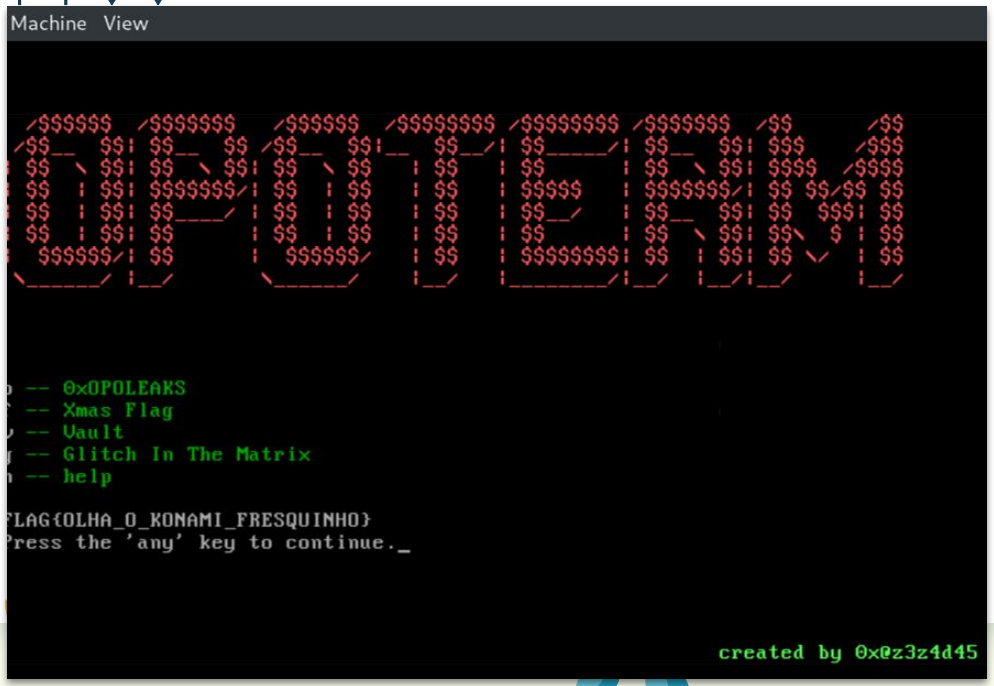
```
char keyPressed;
char isThereKey( ){
  keyPressed=isThereExtendedKey( );
  if (keyPressed==0xe0){
    return getExtendedKey( )+128;
  }
  else if(keyPressed!=0){
    return getKey( );
  }
  else{

  }
  return 0x00;
}
```

Get Regular Keys & Extended Keys

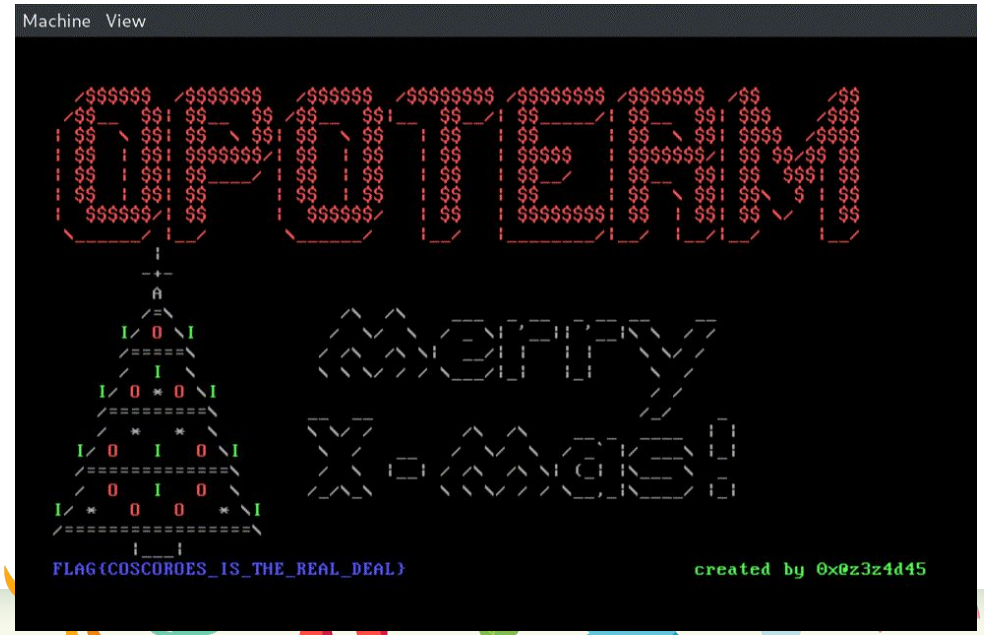
```
while (charpwd != 0x3) {
    if ((charpwd=isThereKey())) {
        //charpwd = getKey();
        break;
    }
```



Konami ↑ ↑ ↓ ↓ ← → ← → B A <ENTER>



XMas Flag


Machine View
FLAG{COSCOROES_IS_THE_REAL_DEAL}
created by 0x@z3z4d45

ShellCode

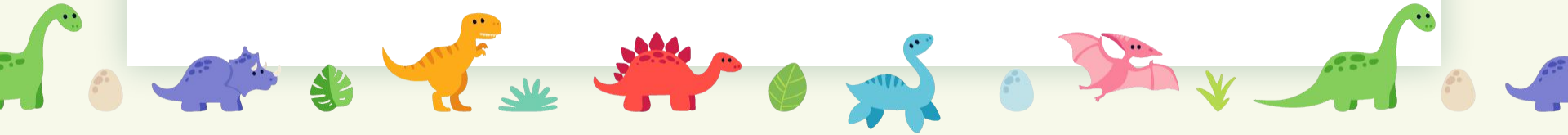
SHELLCODE

```
#asm
push    bp
mov     bp, sp
push    di
push    si
#endasm

#asm
mov ax,#callOffset
call ax
skip:
nop
#endasm

#asm
pop     si
pop     di
pop     bp
#endasm
```

Decrypt Flags

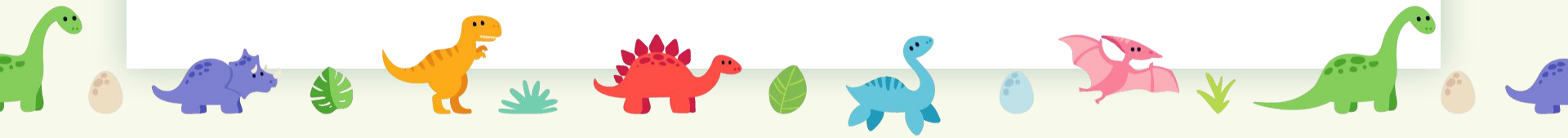
Decrypt Flags

```
void getFlag(c,r) char *c; char *r;{
  #ifdef OBFUSCATE
    deobfStr(c+2+obfKeySize(c), obfStrSize(c),\
    obfKeyValue(c), obfKeySize(c),r);
  #else
    strcpy(r,c);
  #endif
}
```



Decrypt Flags

```
void deobfStr(c,cs,k,ks,r) char *c; int cs;char *k;int ks; char *r;{
  #ifdef OBFUSCATE
    int j;
    for (j=cs-1;j>=0;j--){
      r[j]=c[j] ^ k[imod(cs-1-j,ks/dispersion)] ^ (cs-1-j);
    }
    r[cs]='\0';
  #else
    strcpy(r,c-2-ks);
  #endif
}
```



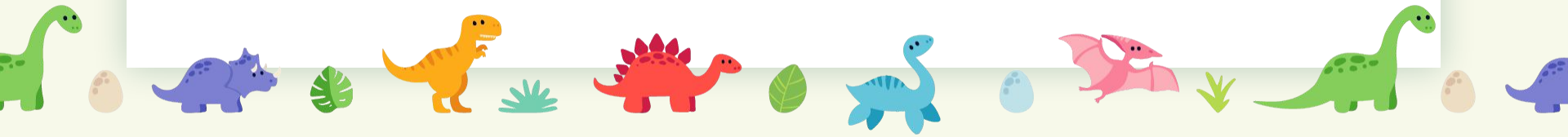
Encrypt Flags

Makefile

```
OBFUSCATE=1
FLAGS=""

ifeq ($(OBFUSCATE),1)
  FLAGS=-DOBFUSCATE=1
endif
```

Makefile

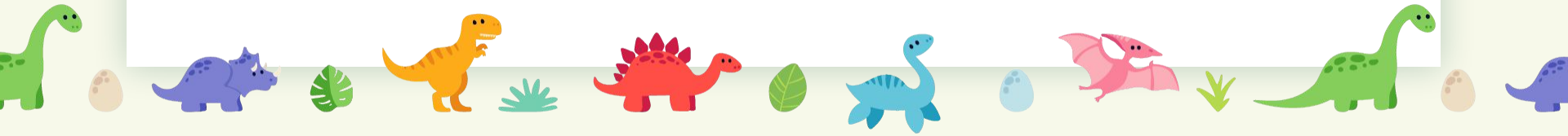
```
kernel.bin: kernel.c
»       python3 packer_strings.py flags.h obfuscated_flags.h
»       $(BCC) -0 -c kernel.c
»       $(BCC) -0 -c screen.c
»       $(BCC) -0 -c keyboard.c
»       $(BCC) -0 -c menu.c
```



Encrypt Flags

interactions.c

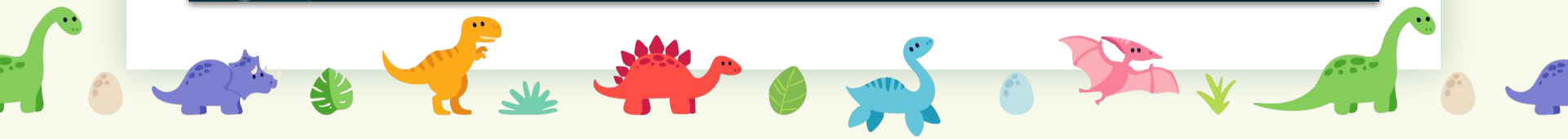
```c
#include "stringzadas_interactions.h"
#include "all.h"
char *bufferino;
```



Encrypt Flags

stringzadas_interactions.h

```
1  #ifdef OBFUSCATE
2    #define INC "obfuscated_flags.h"
3  #else
4    #define INC "flags.h"
5  #endif
6
7  #include INC
8
9
```



Encrypt Flags

obfuscated_flags.h

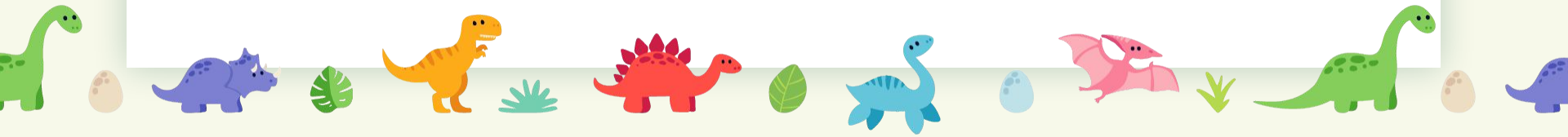
```
char codeKONAMI[] = { 0xc8, 0xc8, 0xd0, 0xd0,
0xcb, 0xcd, 0xcb, 0xcd, 0x62, 0x61, 0xd, 0x0 };
char flagKONAMI[] = {0x1e, 0x16,  0x74, 0x3c,
0xa5, 0xe1, 0xe2, 0x5, 0x2, 0x18, 0xc9, 0x23,
0x14, 0xd0, 0x3a, 0x2f, 0xd1, 0xdf, 0x98, 0x88,
0x1e, 0xa9, 0x8e, 0xfd , 0x96, 0x9b, 0x97, 0x96,
0xb2, 0x87, 0x93, 0x96, 0x59, 0x2a, 0x3e, 0x80,
0x91, 0x92, 0x8a, 0x9f, 0x90, 0x8d, 0x9c, 0x41,
0x3a, 0x2f, 0x99, 0x9c, 0x9d, 0x86, 0x9d, 0x9a,
0x86, 0xb5 };
char vaultPwd[] = {0x7, 0xc,  0xb0, 0x18, 0x21,
0xb9, 0x3c, 0x2c, 0xb0, 0x7c, 0x7c, 0x69, 0x7f,
0x0 , 0xd4, 0x15, 0x14, 0xdc, 0x4c, 0x45, 0xd9 };
char vaultFlag[] = {0x2b, 0xe,  0x4e, 0x26, 0x69,
0xfe, 0xfc, 0xb9, 0xdf, 0x86, 0x56, 0x51, 0x65,
0xf9, 0x5b, 0xba , 0xe, 0xc, 0xc, 0x12, 0x32,
0x1b, 0xa, 0x3, 0xa, 0xa, 0x13, 0x34, 0x33, 0x21,
```

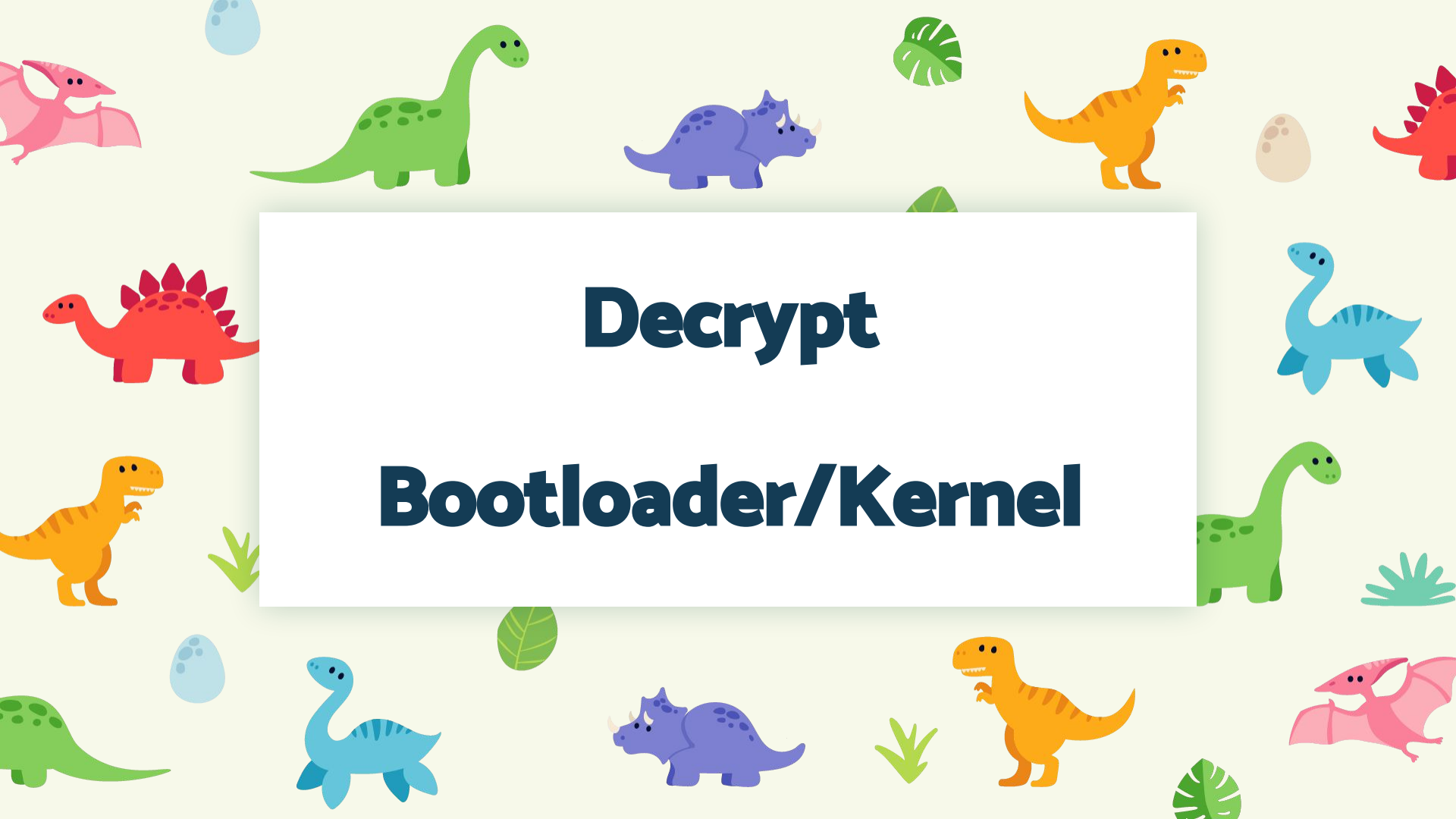
flags.h | obfuscated_flags.h

```
char codeKONAMI[]={0xc8,0xc8,0xd0,0xd0,0xcb,
0xcd,0xcb,0xcd,0x62,0x61,0x0d,0x00};
char
flagKONAMI[]="FLAG{OLHA_O_KONAMI_FRESQUINHO}";

char vaultPwd[]="bolorei";
char
vaultFlag[]="FLAG{RABANADAS_GIVE_STRENGTH_TO_BRU
TEFORCE}";

char
treeFlag[]="FLAG{COSCOROES_IS_THE_REAL_DEAL}";
char
polymorphFlag[]="Flag{TRONCO_DE_NATAL_IS_NOT_REA
LLY_A_LOG}";
char
flagShellCode[]="FLAG{FILHOSES_WILL_NOT_KEEP_YOU
```

Encrypt Flags

```
filename = sys.argv[1]
filename_output = sys.argv[2]

def encryptStrings(key,stringzadas):
  output = []
  keyzadas=[]
  for i in range(0,len(key),2):
    keyzadas.append(key[i])
  size=len(stringzadas)-1
  for i in range(size,-1,-1):
    shift =size-i
    index = shift%len(keyzadas)
    cenas = bytes([ ord(stringzadas[i]) ^ keyzadas[index] ^ shift ])
    output+=cenas
  return output
```




Decrypt
Bootloader/Kernel

Makefile

```
OBFUSCATE=1
FLAGS=""

ifeq ($(OBFUSCATE),1)
  FLAGS=-DOBFUSCATE=1
endif
```

Decrypt Bootloader

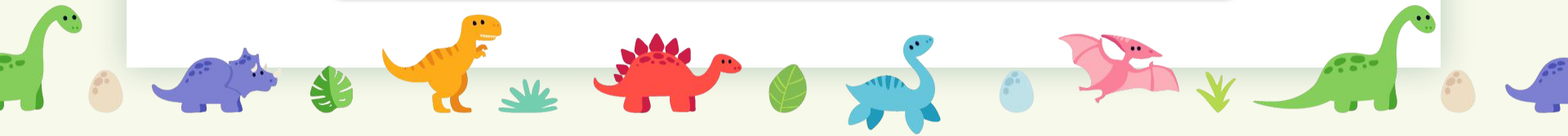
```
;;;;xor memory with a increment var
;stub to DEOBFUSCATE bootloader
%ifdef OBFUSCATE
    xor cx,cx   ;reset counter to 0
  loop2:
    mov bx,labelzadas
    add bx,cx
    mov ax,[bx]
    xor al,cl
    mov [bx],al

    add cx,1
    cmp bx,0x7dfe ;end of bootloader
    jle loop2
    jmp labelzadas

    ;this tag is used by the packer
    babecafe db 0xba,0xbe,0xca,0xfe,0x00
%endif

;this part will be packed on release version
labelzadas:
```

Wipe Part of Bootloader

```
;;Stub to DEOBFUSCATE KERNEL and wipe bootloader
%ifdef OBFUSCATE
  ;;wipe first part of bootloader, leave deadbeef key
  ;because it will be needed to decrypt the kernel
  mov cx,badcode
  mov ax,$
  loop3:
    mov dl,0x90
    mov bx,cx
    mov [bx],dl
    inc cx
    cmp cx,ax ;0x7c00
    jl loop3
```

Decrypt Kernel

```
;;deobfuscate kernel// xor deadbeef stored in key label
;;and xor with increment byte cl
xor cx,cx   ;reset counter to 0
loop1:
  mov bx,cx
  ;mov ax,0x7e00  ;
  ;shl ax,8      ;0x7e00 - address of kernel
  add bx,0x7e00    ;increment kernel address to the index counter

  ;;;;;;;;;;;;;;;;
  mov ax,[key]   ;get 0xdead from key db
  ;;;;;;;;;;;;;;;;
  mov dl,[bx]
  xor dl,al       ;xor 0xde
```



Wipe Part of Bootloader

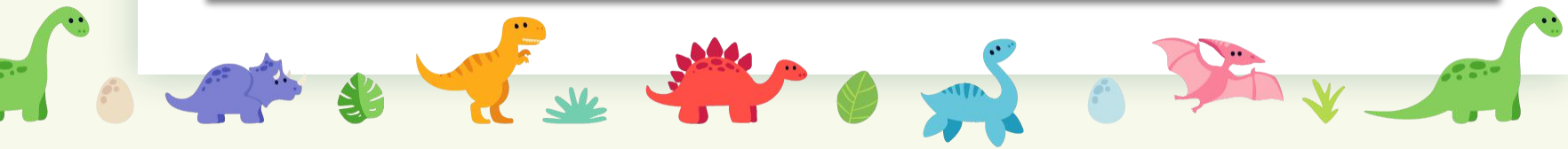
```
;;wipe everything but the long jump
mov cx,0x7c00
mov ax,$
loop4:
  mov dl,0x90
  mov bx,cx
  mov [bx],dl

  inc cx
  cmp cx,ax ;0x7c00
  jl loop4
```

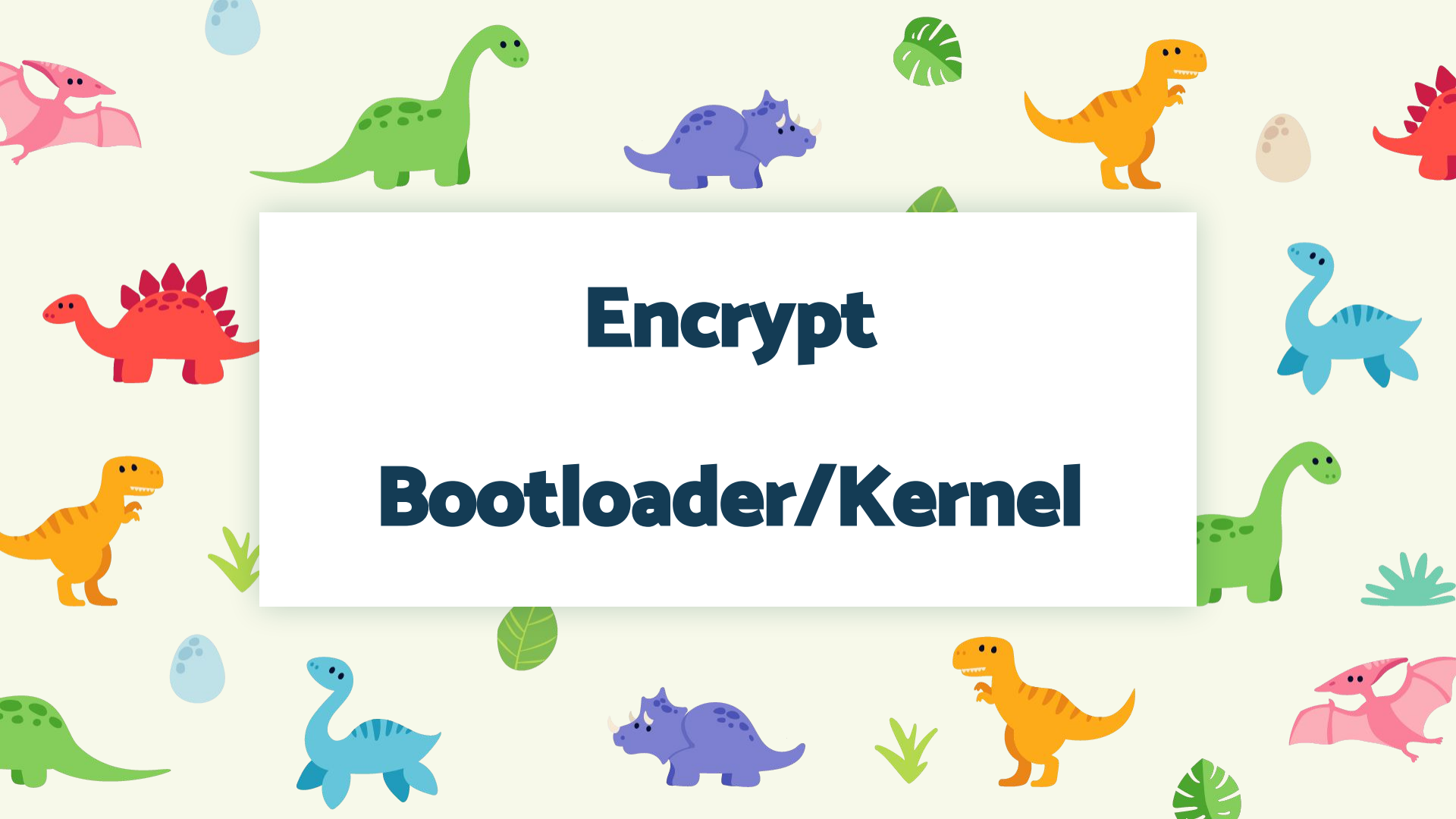
Wipe Bootloader From Kernel

```
void clearBootloader(){
  int i;
  char *addr;
  addr=0x7c00;
  for (i=0;i<0x1fe;i++){
    *addr++=0x0;
  }
  addr=0x7c00;
}
```

Wipe Bootloader

```
void main() {

  clearBootloader();
```



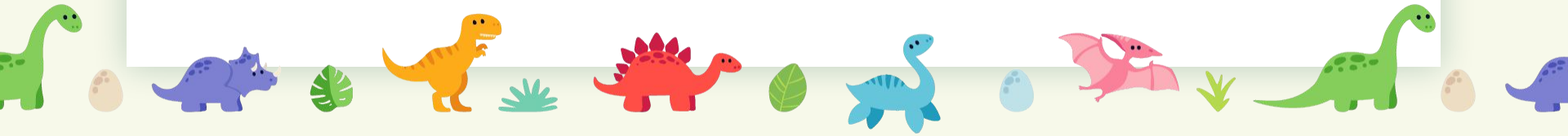

Encrypt
Bootloader/Kernel

Makefile

```
OBFUSCATE=1
FLAGS=""

ifeq ($(OBFUSCATE),1)
  FLAGS=-DOBFUSCATE=1
endif
```

Makefile

```
disk.fd: boot.bin kernel.bin
»       python3 packer_bootloader.py boot.bin obfuscated_bootloader.bin
»       python3 packer_kernel.py kernel.bin obfuscated_kernel.bin
»       »
        ifeq ($(OBFUSCATE),1)
»           cat obfuscated_bootloader.bin obfuscated_kernel.bin > disk.fd
        else
»           cat boot.bin kernel.bin > disk.fd
        endif
```

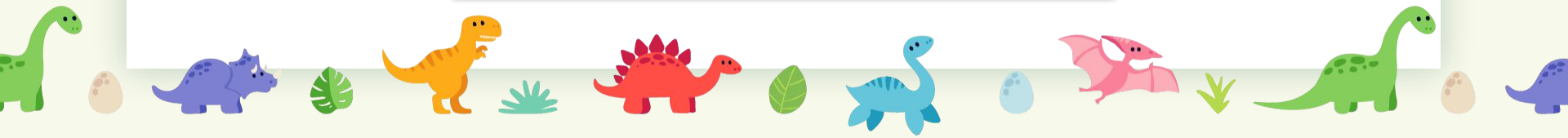


Encrypt Bootloader

```
i=0
while (byte := f.read(1)):
    # Do stuff with byte
    if(g.tell()>=510):
        i=0
    a=byte[0]^i
    i+=1
```

Encrypt Kernel

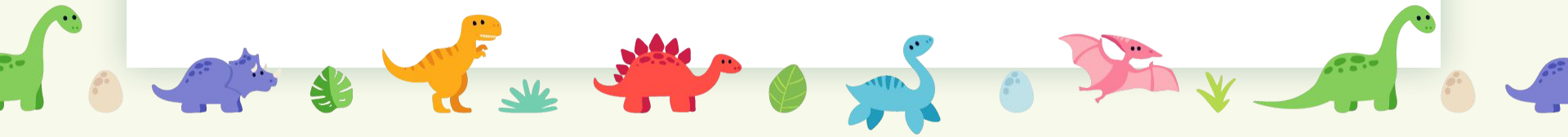
```
g = open(filename_output, 'wb')
key=[0xde,0xad,0xbe,0xef]
i=0
with open(filename, "rb") as f:
    while (byte := f.read(1)):
        # Do stuff with byte
        if f.tell()>index:
            a=byte[0]^0
        else:
            a=byte[0]^key[i%4]^i
```



Creating a Polyglot

MBR Key Concepts

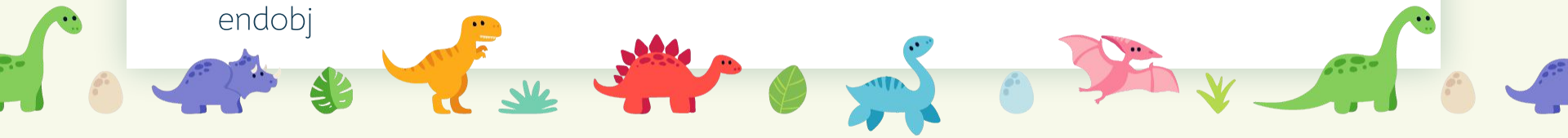
- 16 bit x86 code starts at offset 0
- It will be executing at the 0x7c00 address in RAM
- It must be 512 bytes long, ending with the signature 55,AA
- Labels and primary partition tables are optional, but can go within this sector
- It must contain code that finds and loads into RAM the code for the next boot stage (such as anOS loader)



PDF Key Concepts

- It is initially parsed as text
- The signature "%%PDF-" must be present within the first 1024 bytes.
- Comment lines begin with '%', which is 25in hex
- "Multi-line" binary objects can also be stored in object streams, which are declared like this:

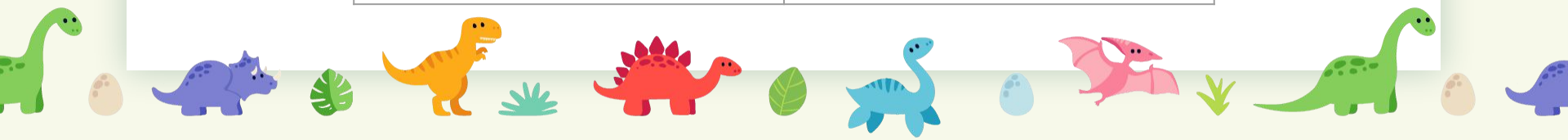
  <obj number> <revision > obj
  <<>>
  Stream
  <stream content>
  Endstream
  endobj



Polygloting

PDF interprets **%** as a text, **comment**
MBR interprets **%** as a 16bit x86 code, **AND** instruction

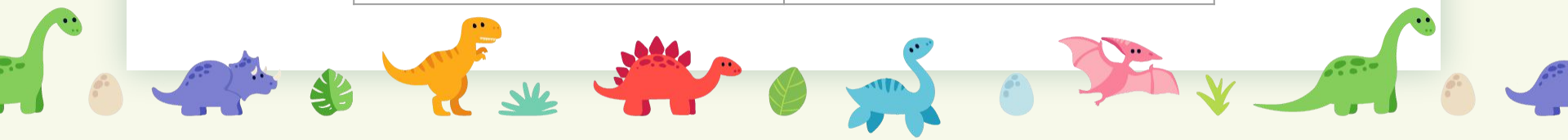
	25 FF FF
PDF	%..
MBR	and ax,0xffff



Polygloting

PDF interprets **%** as a text, **comment**
MBR interprets **%** as a 16bit x86 code, **AND** instruction

	25 FF FF EB 57 0A
PDF	**%** . . . W **\n**
MBR	**and** ax,0xffff; **jmp** short 0x5c;



Polygloting - PDF

```
00000000: 25ff ffeb 570a 2550 4446 2d31 2e35 0a39  %...W.%PDF-1.5.9
00000010: 3939 3920 3020 6f62 6a0a 3c3c 3e3e 0a73   999 0 obj.<<>>.s
00000020: 7472 6561 6d0a 96c8 d283 b181 9ea6 aa8a  tream...........
00000030: cdcf b0c2 cacf aac5 df9b fec8 df9c a783  ................
00000040: 9ebb acd4 9e9b b6c8 9e89 b2c2 ce9f a783  ................
00000050: be00 dead beef 000b adc0 de00 31c0 8ed8  ............1...
```

. . .

```
00002e10: 8392 a1f6 ca00 0000 656e 6473 7472 6561 ........endstrea
00002e20: 6d0a 656e 646f 626a 0a0a 2550 4446 2d31 m.endobj..%PDF-1
```

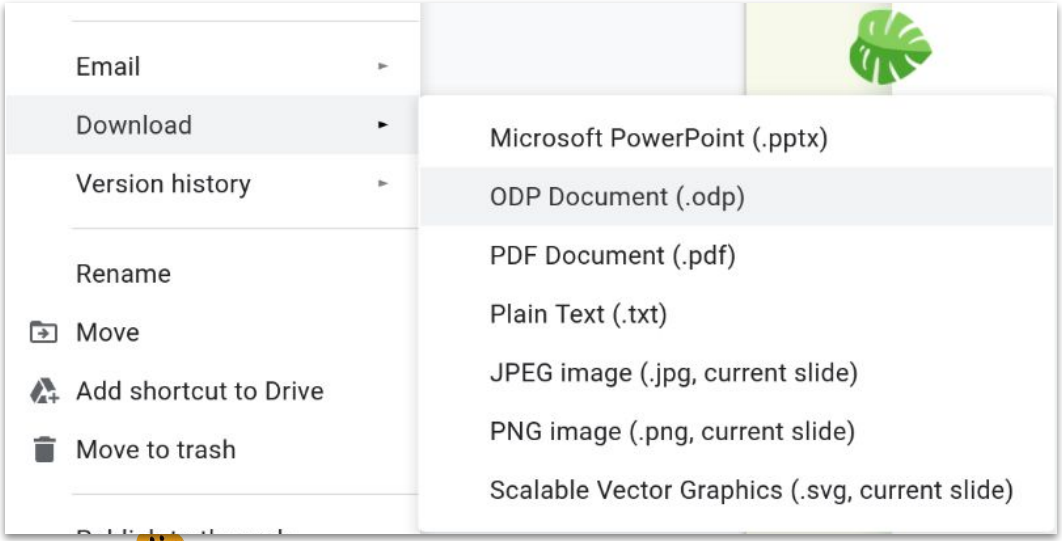
Polygloting - MBR

```
00000000: 25ff ffeb 570a 2550 4446 2d31 2e35 0a39  %...W.%PDF-1.5.9
00000010: 3939 3920 3020 6f62 6a0a 3c3c 3e3e 0a73   999 0 obj.<<>>.s
00000020: 7472 6561 6d0a 96c8 d283 b181 9ea6 aa8a  tream...........
00000030: cdcf b0c2 cacf aac5 df9b fec8 df9c a783  ................
00000040: 9ebb acd4 9e9b b6c8 9e89 b2c2 ce9f a783  ................
00000050: be00 dead beef 000b adc0 de00 31c0 8ed8  ............1...

                                . . .

000001e0: 5b5c 5d5e 5f60 6162 6364 6566 6768 696a  [\]^_`abcdefghij
000001f0: 6b6c 6d6e 6f70 7172 7374 7576 7778 55aa  klmnopqrstuvwxU.
00000200: 8b25 59bb 8c40 bce8 88fb e927 8729 55b7  .%Y..@.....'.)U.
```

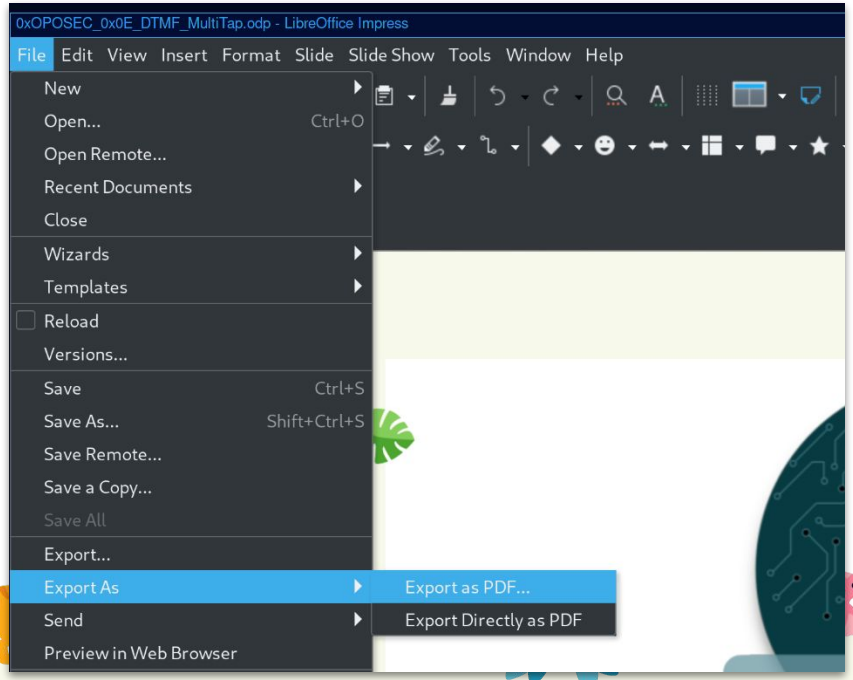
Polygloting - ASM

```
[SEGMENT .text]
db 0x25,0xff,0xff
jmp start»» » ;Code execution begins at the beginning.

db  0x0A
pdf1 db  '%','PDF-1.5', 0Ah
pdf2 db  '9999 0 obj',0Ah
pdf3 db  '<<>>',0Ah
pdf4 db  'stream',0Ah

errormsg db 0x96, 0xc8, 0xd2, 0x83, 0xb1, 0x81, 0x9e, 0x
0x9b, 0xfe, 0xc8, 0xdf, 0x9c, 0xa7, 0x83, 0x9e, 0xbb, 0x
0x83, 0xbe, 0
key db 0xde,0xad,0xbe,0xef,0x00
badcode db 0x0b,0xad,0xc0,0xde,0x00
start:
   ;main code starts here
```

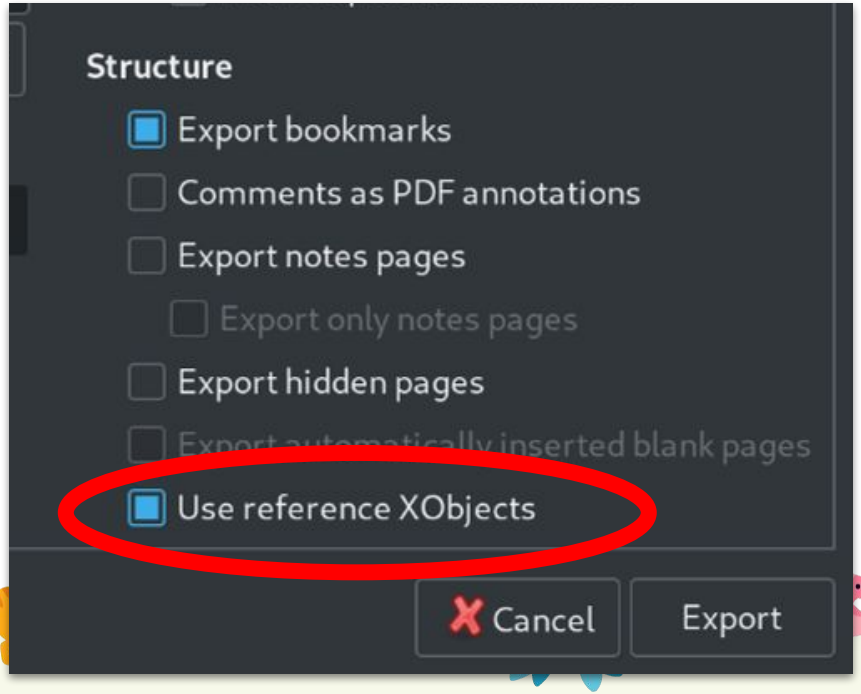
Polygloting - Glue All Together



Polygloting - Glue All Together

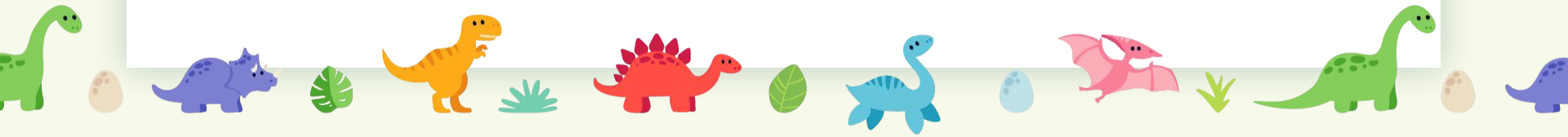


Polygloting - Glue All Together

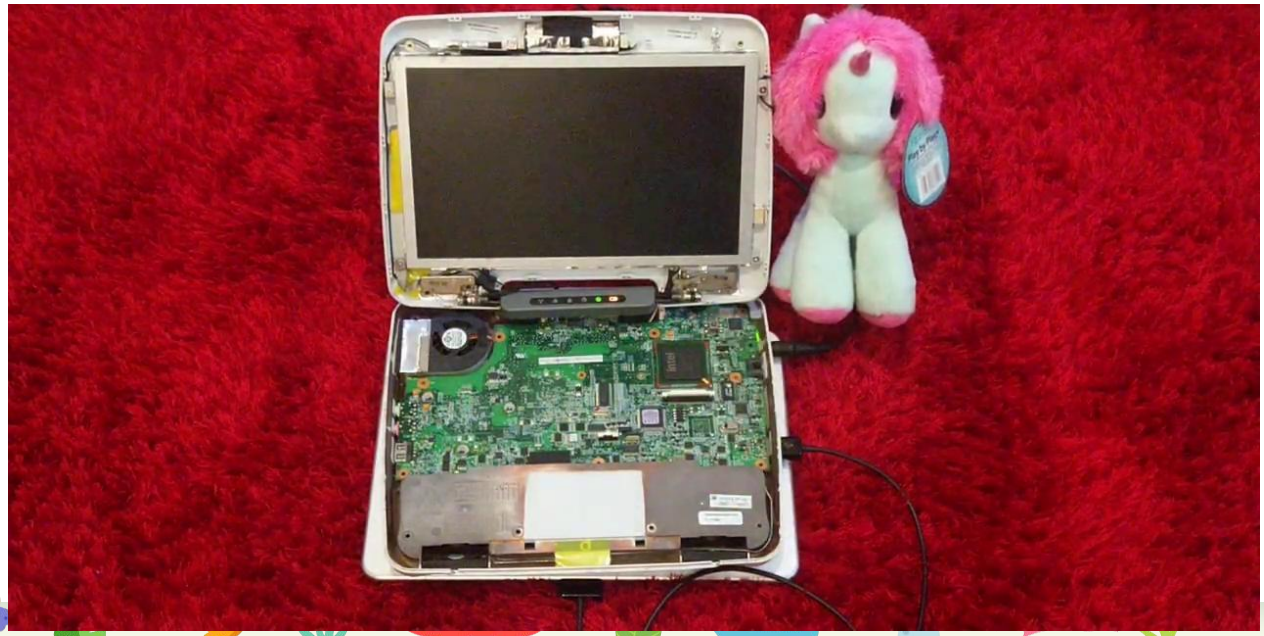


Polygloting - Glue All Together

```
anon  /tmp/oposec
make
anon  /tmp/oposec
cat disk.fd writeup.pdf > 0xOPOSEC_0x0E_DTMF_MultiTap.pdf
anon  /tmp/oposec
file 0xOPOSEC_0x0E_DTMF_MultiTap.pdf
0xOPOSEC_0x0E_DTMF_MultiTap.pdf: DOS/MBR boot sector
```

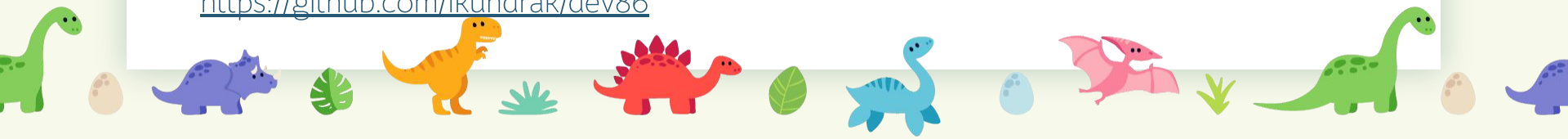


Video



Resources

https://en.wikipedia.org/wiki/Real_mode
https://en.wikipedia.org/wiki/Master_boot_record
https://nixhacker.com/explaining-the-magic-of-mbr-and-its/
https://wiki.osdev.org/Rolling_Your_Own_Bootloader
http://3zanders.co.uk/2017/10/13/writing-a-bootloader/
http://3zanders.co.uk/2017/10/16/writing-a-bootloader2/
http://3zanders.co.uk/2017/10/18/writing-a-bootloader3/
https://stackoverflow.com/questions/36968829/how-to-switch-from-real-mode-to-protected-mode-after-bootloader
https://github.com/angea/pocorgtfo/blob/master/releases/pocorgtfo02.pdf
https://github.com/travisgoodspeed/bsos
https://github.com/lkundrak/dev86



Bootloader

https://github.com/travisgoodspeed/bsos

Polyglot

https://github.com/angea/pocorgtfo/blob/master/releases/pocorgtfo02.pdf

GitHub

https://github.com/zezadas/0x0POSEC_0x0F_Polyglot_PDF_MBR_Bootloader

Thanks!

Any questions?
You can find me at
@0xz3z4d45 and
zezadas@sefod.eu