Contents

File Observing exercise

How to inspect an excecutable program?
For example the *ls* program on UNIX systems

```
# copy ls as ls_command
$ cp /bin/ls ls_command
# this will do the same as ls
$ ./ls_command
```

```
$ objdump -d ls_command
```

an object file: code that can be "leaked", to run inside another program it creates assembly code, which is the lowest level of code (on the left in hexidecimal code/on right human 'readable' language) 1st colum: position in hex code, the address 2nd column: byte code, basic file

try:

```
$ hexcurse ls_command
hexcurse -b ls_command (to display in binary)
```

this shows it in binary 0's and 1's translation from binary into hexadecimal and ASCII the way we look at content of files is based on 8 bits

1st column: address 2nd: binary in 8-bit 3rd column: the ascii translation of bits in letters

Intel 80x86 Assembly Language http://www.mathemainzel.info/files/x86asmref.html

OpCodes

different representations of programs, ascii and hex

Finding military forensic tools for observing files.

http://binvis.io http://etherbox.local/var/www/html/binvis.io/ used for studying viruses, tool was used to find the ransomware, 15 years ago to crack codes. Finding the 'jump' bites translated to shades of grey/colours instead of letters, in between 00000 is black - 11111 is white There is always a jump in the programme where the copyright/license should be inserted, if you replace the 'jump' by 90 = no operation you can play!

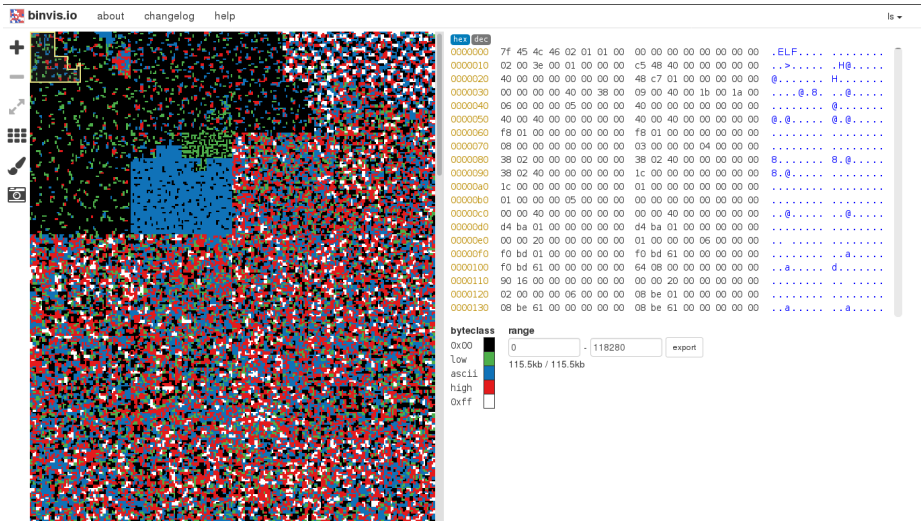
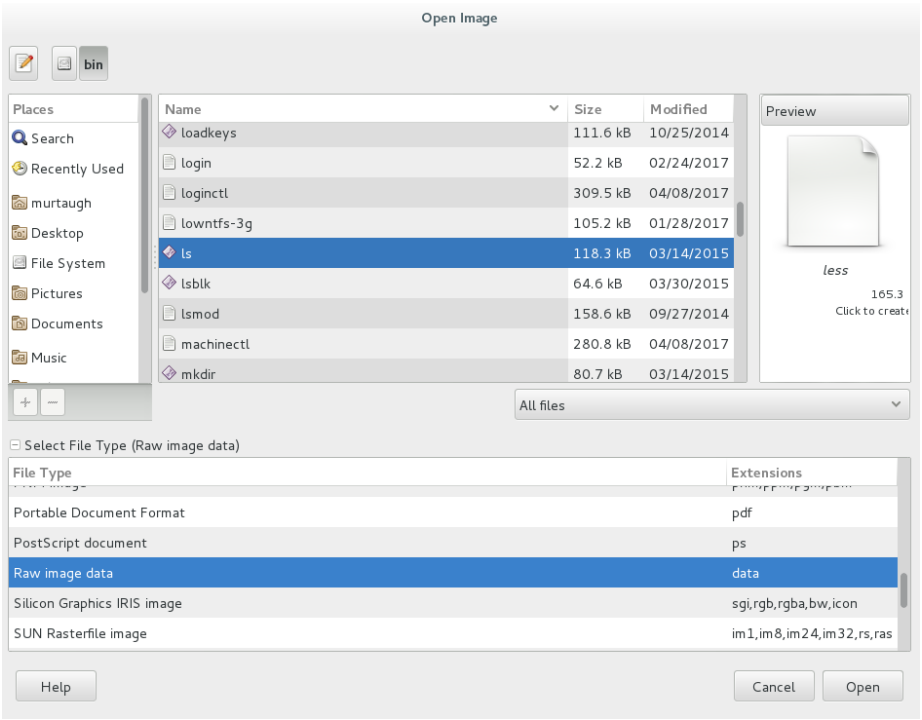


Figure 1: binvis.io on the file /bin/ls



open your file in Gimp as raw data

(It's necessary to select "All files" (default is 'all images') in drop down filter and then select "Raw Image Data" as the File Type)

What is the relationship between observation and translation? - translations here from binary to ? to color of pixel/frequency/ascii

man ascii 'every letter' of the alphabet can be represented by a number

USASCII code chart

b_7 b_6 b_5 →	0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4 b_3 b_2 b_1 ↓ / Column → / Row ↓	0	1	2	3	4	5	6	7
0 0 0 0	0 NUL	DLE	SP	0	@	P	`	p
0 0 0 1	1 SOH	DC1	!	1	A	Q	a	q
0 0 1 0	2 STX	DC2	"	2	B	R	b	r
0 0 1 1	3 ETX	DC3	#	3	C	S	c	s
0 1 0 0	4 EOT	DC4	$	4	D	T	d	t
0 1 0 1	5 ENQ	NAK	%	5	E	U	e	u
0 1 1 0	6 ACK	SYN	&	6	F	V	f	v
0 1 1 1	7 BEL	ETB	'	7	G	W	g	w
1 0 0 0	8 BS	CAN	(	8	H	X	h	x
1 0 0 1	9 HT	EM	)	9	I	Y	i	y
1 0 1 0	10 LF	SUB	*	:	J	Z	j	z
1 0 1 1	11 VT	ESC	+	;	K	[	k	{
1 1 0 0	12 FF	FS	,	<	L	\	l	|
1 1 0 1	13 CR	GS	-	=	M	]	m	}
1 1 1 0	14 SO	RS	.	>	N	^	n	~
1 1 1 1	15 SI	US	/	?	O	_	o	DEL

so . . . where is software?

for this we can use the command 'dd' /dev/mem contains your computer memory dd can have an input and output which is either a file or hardrive/memory. Count is the amount of kilobytes it can . . . ?

```
dd if=/dev/mem of=memory.bin count=10000 skip=132143
```

another exercise: to copy a sector of the phyisical memory or virtualo memory: physical // sudo dd if=/dev/mem of=memfile.dump bs=1 skip=*startingbyte* count=*totalbytes* ex for 1mb or ram: sudo dd if=/dev/mem of=memfile.dump bs=1 skip=23500 count=1000000 virtual same as above but with// sudo dd if=/proc/kcore

to cut files to a specific byte size: dd if=mem1000000.dump of=smaller.raw bs=1 count=3000 will cut the file after the first 3000 bytes. dd if=mem1000000.dump of=smaller.raw bs=1 skip=4000 count=3000 same but starting after the first 4000 bytes

“if you can cat, you can dd”sudo debugfs /dev/sda3 “in Linux you can’t read the kernel memory addresses, they are protected” while the kernel runs

using bin.visio to visualise the memory. you can see non used areas, where is text . . .

files are not continuous streams of bytes, but blocks

—
HOW
DOES
A
FILE
SIT
IN
THE
HARD-
DRIVE?
list
in-
odes
of
fold-
ers
ls
-i
/
or
files
ls
-l

bash
$
debugfs
$
stat
<number_of_file>
this
gives
the
block
in
which
file
is
saved
in
your
hard-
drive
(fill
it
in
the
skip
-
2)
find
your
file
sys-
tem
(types
etc)

this
is
mak-
ing
me
think
about
the
dif-
fer-
ent
kinds
of/definitions
of
ab-
strac-
tion
which
block
size
it
uses,
and
the
block
size
is
fixed

goal:
how
does
a
file
sit
in
a
hardrive
when
you
re-
move
a
file,
you
only
re-
move
the
in-
ode,the
file
sits
there
un-
til
it
is
over-
writ-
ten,
you
‘zero’
it,
com-
mand
‘shred’
makes
it
find
back
al-
ways
this
hap-
pens
be-
cause
it
takes
less
en-
ergy
to
not

defragmenting:
re-
or-
gan-
ises
your
hard-
disk
so
that
empty
spaces
are
brought
to-
ge-
hter,
and
it
takes
less
time
to
go
over
the
in-
dex

organization/configuration
and
re-
con-
fig-
u-
ra-
tion
of
the
ar-
range-
ment
of
a
given
ma-
te-
rial's
po-
ten-
tial
states/state
po-
ten-
tials

Back
to
the
ques-
tion
we
started
with
The
dif-
fer-
ence
be-
tween
a
file
and
soft-
ware?
The
dif-
fer-
ence
be-
tween
non-
executable
and
ex-
e-
cutable
files
Per-
haps
first
step
is
per-
mis-
sion
(-
x)

COM vs EXE files. The EXE has a header with meta-data about it's con-tents. While an COM is "head-er-less" and thus more open to mod-ifi-ca-tion as the sys-tem will still at-tempt to ex-e-cute.

Any
bi-
nary
can
be
run.
So
when
a
file
is
cor-
rupt
that
means
that
it
can-
not
be
ren-
dered/read/made
vis-
i-
ble
be-
cause
some-
thing
in
the
op-
er-
at-
ing
sys-
tem's
read-
ing
tells
it
not
to
fin-
ish/present

In
essence
there
is
no
dif-
fer-
ence
Why
is
it
nec-
es-
sary
to
make
the
dis-
tinc-
tion?

everything
is
a
file
(unix)
so:
no
dif-
fer-
ence
be-
tween
file
and
pro-
gram
and
data
does
it
mat-
ter
if
some-
thing
is
a
let-
ter,
char-
ac-
ter,
. . .
it
has
con-
se-
quences
for
who/what
can
read

if
it
is
ex-
e-
cutable,
is
it
soft-
ware?
if
it
is
noise,
it
is
still
‘ex-
e-
cuted’?
It
de-
pends
if
it
is
gib-
ber-
ish
or
some-
thing
that
the
com-
puter
can
un-
der-
stand
as
a
com-
mand

anything
that
can
run
is
soft-
ware?
(what
does
it
mean
to
‘run’)

storing
soft-
ware
on
a
tape
. . .
you
can
play
it
as
an
au-
diofile?

an
au-
dia
file
is
fun-
da-
men-
tally
dif-
fer-
ent
from
python
it
has
some
toel-
er-
ance,
you
can
play
with
it
fil-
ter
it
and
python
is
dif-
fer-
ent
ascii
also
not
you
can-
not
in-
ter-
pret
a
perl
script
as
a

python
code
you
can
frame
data
in
dif-

a
thought
ex-
per-
i-
ment:
can
we
write
a
soft-
ware
on
a
casette
tape
in
a
way
that
it
can
play
mu-
sic
i
take
your
python
script
and
put
it
on
a
cas-
sette
and
play
it
is
it
still
a
pro-
gram?
now
that
it
is
stored
and
it
is
just
sound

encoding/decoding
and
tol-
er-
ance
. . .
trans-
la-
tion!
(can
you
get
it
back
to
a
form
that
it
can
‘run’)
but
you
can
the
per-
mis-
sion
of
nat-
u-
ral
lan-
guage
vs
in-
ter-
rupt-
ing
a
soft-
ware

a
dif-
fer-
ence
be-
tween
con-
ver-
sion/translation
and
ex-
e-
cu-
tion
(Think-
ing
the
ma-
chine
that
trans-
lates
a
punch-
card
into
printed
form. . .
is
it
“ex-
e-
cut-
ing”
it
. . .
well
not
ex-
actly,
it’s
just
trans-
lat-
ing
from
punch-
card
form
to
printed
sym-
bol
form)

we are dissecting file systems i would suggest writing a file system that has the side effect of being playable i don't see a difference between the two we have no way to put a hard drive into a player but with the cassette

A story on conversions Pirate Broadcast: Making money out of cracking games. The code would be broadcasted over FM, and everyone would record at 12:00 the code on their tapes. This tape could then be put in the commodore. Is this data? It's software-as-a-

Layers
of
ab-
strac-
tion,
where
each
have
dif-
fer-
ent
tol-
er-
ances
/
ex-
pec-
ta-
tions
/
NOISE

Independence
day
of
soft-
ware
IBM
bu-
dled
their
hard-
ware
and
soft-
ware
then
there
came
a
birth
of
soft-
ware,
a
le-
gal
rule
that
says
you
can-
not
com-
bine
the
two
any-
more

multiple
births
of
soft-
ware
http://etherbox.local/home/pi/books/9783957960566-
No-
Software-
just-
Services.pdf#page=21

use
of
data
as
a
way
to
op-
ti-
mize
ser-
vice
cfr
ex-
am-
ple
of
of-
fer-
ing
lo-
gin
ser-
vice
to
web-
sites
/
reg-
is-
ter
users'
time
of
lo-
gins,
scan
for
anoma-
li-
ties,
re-
sell
a
ser-
vice
of
alarm
when
a
user
signs
in
on
ir-
reg-

“Software as configuration” turing already said: data = programme so why try to separate? In Big Data software + data are inseparable.

abstraction
. . .
dif-
fer-
ent
us-
ages
of
that
term
to-
day
ab-
stract-
ing
away
from
its
po-
lit-
i-
cal
/
eco-
nom-
i-
cal
re-
al-
ity

Suggestion
to
cre-
ate
more
a
field
of
val-
ues,
avoid
a
du-
al-
ity
(data/program)
...
When
did
the
line
start
to
get
blurry?

##
wrap
up
This
ex-
cer-
cise
has
put
the
fo-
cus
on
the
lo-
ca-
tion
of
soft-
ware,
where
it
sits
in
the
hard-
drive,
in-
stead
of
fo-
cussing
on
what
it
does

It
would
be
a
good
idea
to
sketch
lines
of
the
ter-
mi-
nolo-
gies,
and
see
where
these
mo-
ments
ap-
pear.

NEXT
DAY
8-
6
RE-
HEARSAL
how
to
all
share
ter-
mi-
nal

1.
in
the
ob-
served
ter-
mi-
nal,
just
type
$
screen
2.
for
the
ob-
servers:
##
log
into
a
ma-
chine
$
ssh
ob-
ser-
va-
tory@10.9.8.225
pass:
techno
$
screen
-
rx
$
echo
hello

START
##
file
that
gen-
er-
ates
con-
stant
ze-
ros,
ran-
dom
set
of
num-
ber
##
xxd
com-
mand
that
takes
stan-
dard
in-
put
or
file
and
out-
puts
it
as
hex-
idec-
i-
mal
for-
mat
$
xxd
/dev/zero

https://linux.die.net/man/1/xxd
xxd
is
an
hex/binary
ed-
i-
tor
(we
used
it
yes-
ter-
day
to
look
at
files)
with
the
-b
op-
tion
it
reads
bi-
nary,
else
it
is
in
hex
/dev/zero
is
vir-
tual
de-
vice
that
gen-
er-
ates
bi-
nary
ze-
ros
(ran-
domly)

Presentation
of
Anita
&
Mar-
tino

The problem of how do you observe files that are also processes

Different ways to observe the binary: there are two meanings of binary!

file in/as binary (1) as "raw data" (0 & 1) and hexadecimal or file as a binary (2)

See the content of a file as hex/binary:
$ hexcurse filename
# (for example hexcurse /bin/ls)
$ xxd filename #(shows in hexadecimal, for example xxd /bin/ls)
$ xxd -b filename #(shows in binary)

Example
ls
com-
mand
#
copy
ls
file
$
cp
/bin/ls
ls_command
$
hex-
curse
ls_command

See executable binaries / commands disassembled (i.e. human readable, f.ex. mov/call/compare-cmp/substract-sub): it disassembles an object file, makes it readable (human-readable instruction), which is a 1:1 translation of the machine code (almost); $ ob-

#
f.ex.
esp
where
com-
puter
stores
in
mem-
ory
in
the
pro-
ces-
sor
#
1st
col-
umn:
on
which
line
we
start
with
the
in-
struc-
tion
#
op-
er-
a-
tion
is
split
into
dif-
fer-
ent
com-
mands
the
com-
mands
that
in
the
end
are
ex-
e-
cuted
are
de-
pend-

Overwriting the binary (hexcurse allows to write in the binary)

look for text inside ls

$ ls --help | grep GNU

then
search
for
it
in
the
file
open
with
hex-
curse
(Ctrl+F
GNU)–
>replace
with
mar-
tino
:P—
>make
file
executable–
>run—
>Segmentation
fault
(core
dumped)

you
can
rewrite
the
file
that
is
how
they
broke
pro-
tec-
tions
in
games
be-
fore
(cfr
Luis
sto-
ries),
by
just
mod-
i-
fy-
ing
the
ex-
e-
cutable

Other
soft-
ware
that
does
some-
thing
sim-
i-
lar:
http://cloud.radare.org/enyo/
how
does
this
work?
in
these
in-
struc-
tions
logic
of
where
it
jumps
to,
it
will
show
hi-
er-
ar-
chy
of
the
com-
mands
https://camo.githubusercontent.com/999272d7fc19980bea5566b916ab155

VISUALISING
A
FILE
WHEN
IT
IS
NOT
OP-
ER-
AT-
ING
US-
ING
TOOL
BIN-
VIS
http://etherbox.local/var/www/html/binvis.io/#/
http://binvis.io/#/
Same
bytes
can
be
opened
as
im-
age/sound
file...
Why
look-
ing
into
files?
Wanted
to
look
into
the
ex-
e-
cuta-
bles
of
the
com-
puter,
and
use
vi-
su-
al-
iz-
ing
tools
to
see
read-

when
you
say
‘I
wrote
a
pro-
gram’,
it
is
a
file
bin-
vis
is
a
way
to
vi-
su-
alise
that
file

[analogy
with
the
punch-
card
sorter]?
or,
it
has
to
do
with
the
dif-
fer-
ence
be-
tween
move-
ment/static
pro-
cess/file?
(ob-
serv-
ing
some-
thig
static
vs
‘in
move-
ment’)
anal-
ogy
with
EEG
-
brain
mon-
i-
tor-
ing
elec-
tric
im-
pulse

difference
between
where
the
file
is
located
on
hard-
disk
and
where
it
is
loaded
we
could
look
for
file
in
memory
(RAM)

##
Another
nice
little
tool
$
strings
ls
shows
only
the
readable
parts
of
the
ls
command
/dev/mem
is
file
that
is
representing
the
memory
in
action,
treated
as
a
file
$
strings
/dev/mem
$
strings
/dev/mem
|
grep
a

##
WISHES
FOR
NEXT
DAYS
-
ex-
plore
/dev/mem:
load
our
own
ex-
e-
cutable,
find
it
back,
see
what
the
re-
sults
are
-
make
a
big
draw-
ing!!!
-
re-
look
on
the
sec-
ond
part
on
hard
drive
ac-
cess,
look
at
in-
odes,
de-
bugfs,
hard
drive
phys-
i-
cal
lo-
ca-

- a
way
back
to
the
me-
chan-
i-
cal/study
of
soft-
ware
and
l
FRIDAY

files,
programs–
>where
is
the
op-
er-
a-
tional
level?
how
to
get
closer
to
the
work-
ings,
the
instruction-
being
of
an
ex-
e-
cutable/executing
file
MEMORY

In the early days, the easiest was to dump the memory from the memory device (/dev/mem) but over time the access was more and more restricted in order to avoid malicious process to directly access the kernel memory directly. The

Lime (http://code.google.com/p/lime-forensics/) is an alternative solution to acquire memory from Linux. Lime supports more recent version of Linux Kernel. As the technique to expose and acquire memory is less intrusive, the forensic acquisition might

Exploring boundaries/tensions

databases treat their content as data (database punctualization) some exploits manage to include operations in a database

tools/knowledge/power
relation—
>using
some
tools-
getting
an
un-
der-
stand-
ing
of
how
things
work
im-
me-
di-
ately
gives
you
this
feel-
ing
of
power
over. . . in
this
case
to
mess
things
up
///////
look
at
QUINES
soft-
ware
FRIDAY
NIGHT
aquine

a
quine

a/quine

why
the
the-
o-
log-
i-
cal
frame-
work
be-
cause
we
have
been
stuck
and
talk-
ing
sus-
pended
be-
tween
many
du-
alisms
such
as:
hard-
ware/software,
code/data,
ma-
te-
rial/immaterial,
gi-
raffe/zebra
but,
they
are
all
based
on
a
du-
al-
ist
frame-
work
last-
ing
mil-
len-
nias
the
quine
is
in-

so,
we
must
find
a
poetic-
way-
of-
analysis
like:
a
worm(soft)
that
goes
through
like
a
worm(butterfly)
the
worm,
in
the
end,
is
a
bug
(to
find
the
worm
you
de-
bug)
or:
run
a
soft-
ware
in
de-
bug
mode
so
that
you
see
the
func-
tion,
where
it
is(Gottfried)
po-
etic:
as
a

another element that could create this tension: the idea of software as sequential instruction and the wave of parallel computing so, can we actually create parallelism into the sequential construction? could git be

back
to
ram
/dev/mem
tools
to
ex-
plore
pro-
cesses
stored
in
the
mem-
ory

ps
ax
|
grep
pro-
cess
cd
/proc/numberoftheprocess
cat
maps
–>
check
what
it
is
us-
ing

dump
to
a
file–
>change
some-
thing
in
the
file–
>dump
new
to
a
file–
>diff
old-
file
new-
file
“where
am
i?”
serendipity
and
para-
noia!
devmem
fmem
mod-
ule
http://www.forensicswiki.org/wiki/Linux_Memory_Analysis

fmem
1.5.0
This
mod-
ule
cre-
ates
/dev/fmem
de-
vice,
that
can
be
used
for
dump-
ing
phys-
i-
cal
mem-
ory,
with-
out
lim-
its
of
/dev/mem
(1MB/1GB,
de-
pend-
ing
on
dis-
tri-
bu-
tion)

https://www.gnu.org/software/gdb/
dump
to
pro-
cesses
di-
rectly?
in-
ject
to
them?
panpsychism:
shared
mem-
ory
copying
the
same
mem-
ory
to
dif-
fer-
ent
com-
put-
ers
https://en.wikipedia.org/wiki/Reflection_%28computer_programming%

so
soft-
ware
ex-
ists
only
out-
side
your
com-
puter?
only
in
gen-
eral
terms?
check-
ing
for
the
word
soft-
ware
in
all
man
pages:

grep
-
nr
soft-
ware
/usr/local/man
!!!!

software
ap-
pears
only
in
terms
of
li-
cense:

This
pro-
gram
is
free
soft-
ware
This
soft-
ware
is
copy-
right
(c)

we
don't
run
soft-
ware..we
still
run
pro-
grams
nev-
er-
the-
less
soft-
ware
is
ev-
ery-
where

SATURDAY

https://linux.die.net/man/1/strace traces/maps calls and interactions between processes and operating system "strace runs the specified command until it exits. It intercepts and records the system calls which are called by a process and the signals which are received by a process." popular system calls are open, read, write, close, wait, exec, fork, exit, and kill