FOSDEM26

My summer vacation 2025

I spent it with IDA looking at VMFS

Erik Schamper

Some introduction

- Erik Schamper
  - Security Researcher
  - 10 years at Fox-IT
  - Author of Dissect

Glossary

Quick recap

- Dissect - System investigation framework
- Acquire - Artefact collection tool for any source data
- VMware ESXi - Hypervisor
- VMware VMFS - Proprietary file system on VMware ESXi
  - VMFS3 first introduced directories (thrilling!)
  - VMFS5 and VMFS6 are largely incremental improvements

How it started

Early days of hypervisor acquisition

- Specific configurations only (SAN)
- Attach physical hardware (or VM) to the same SAN
- Mount datastore with `vmfs(6)-tools`
- Run `acquire`

… profit?

Not quite

- Unfortunately `vmfs-tools` is riddled with bugs
- Lots of data corruption, missing data and unacquirable VMs
- Unmaintained

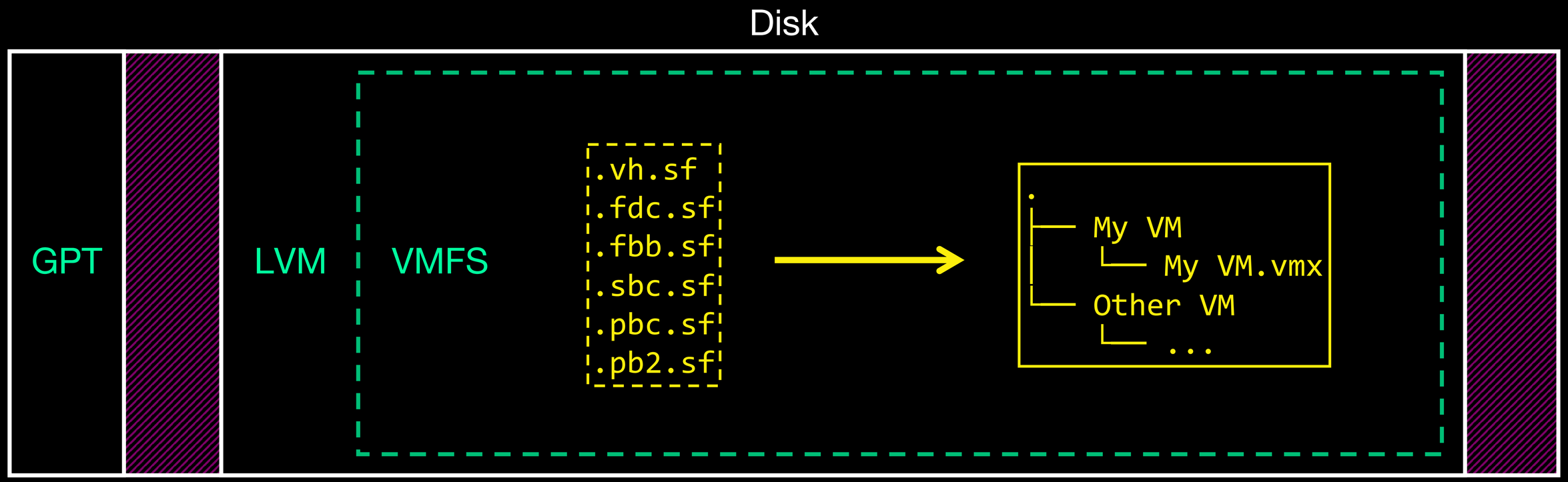
How hard could it be?

`from dissect import cstruct`

- The year is 2020, VMFS data recovery job, trashed RAID
- Need to vaguely understand VMFS to aid in manual RAID rebuilding
- Armed with IDA and `vmfs-tools`, build some inspection tools

High level understanding so far

Based on existing research



`dissect.vmfs` was born

- … might as well write a whole implementation
- Take a little bit of `vmfs-tools`, and a little bit of IDA
- Fix some of the more obvious `vmfs-tools` bugs
- Leave some `#TODO` for the future

New capability unlocked

Level up!

- Datastore block devices are readable from ESXi shell
- Integrate `dissect.vmfs` into `dissect.target`
  - Can run `acquire` on ESXi hosts now
- Hypervisor data acquisition now as easy as “run this executable”

And all was good in the world

Until it wasn’t

Second level directory open in large-capacity VMFS6 file system got wrong #37

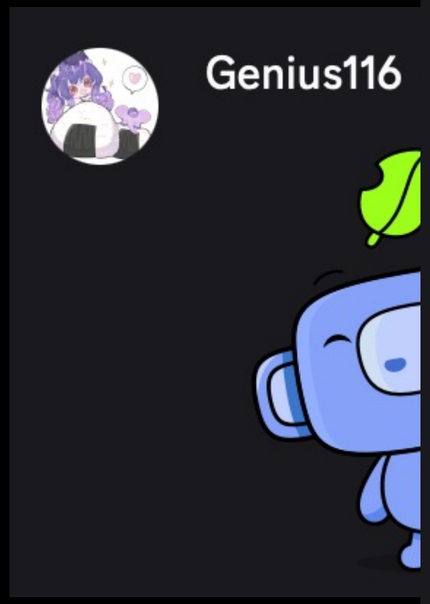
Genius116 opened on Apr 10, 2025 · edited by DissectBot

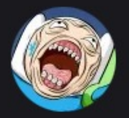
Edits

Hey developer! When I use this project to open a large-capacity file system (about 65 TB) of VMFS6 format. I can see the files and contains in the root dir (/) ,but when I try to print the second level directory .Something went wrong .

```
>>> from vmfs import \\*
>>> f = open('/dev/disks/vml.0000000000766d686261303a31343a30:1','rb')
>>> lvm =LVM(f)
>>> fs = VMFS(lvm)
>>> print(fs.get("/").listdir())
{'.': <FileDescriptor address=<FD c0 r0> name=.>, '..': <FileDescriptor address=<FD c0 r0> name=..>, '.fbb.sf'
>>> print(fs.get("/18T").listdir())
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/tmp/dissect/vmfs/vmfs.py", line 425, in listdir
    return {n.name: n for n in self.iterdir()}
  File "/tmp/dissect/vmfs/vmfs.py", line 425, in <dictcomp>
    return {n.name: n for n in self.iterdir()}
  File "/tmp/dissect/vmfs/vmfs.py", line 435, in iterdir
    yield from self._iterdir_vmfs6()
  File "/tmp/dissect/vmfs/vmfs.py", line 458, in _iterdir_vmfs6
    raise NotADirectoryError(f"Invalid directory version for {self}: 0x{header.version:x}")
dissect.vmfs.exceptions.NotADirectoryError: Invalid directory version for <FileDescriptor address=<FD c6 r0> r
```

Start ir





Schamper 4/16/25, 5:23 PM

I spent some time today setting up a new ESXi VM and will continue tomorrow to try and reproduce, I'll get back to you later

April 17, 2025

Genius116 4/17/25, 3:02 AM

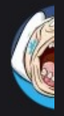
Thanks!

I'm just sharing my current progress with you. I've found that reading .sbc.sf files from large-capacity disks has issues, which causes problems when parsing second-level directories.

I tried accessing the .sbc.sf file directly via the ESXi upper-layer filesystem instead of parsing it through VMFS.

Schamper 4/17/25, 1:06 PM

Okay I have a reproducible filesystem



in the BlockStream class or in the resource reading? Or is the SBC buffer already entirely wrong?

What goes wrong

- Symptom: for file systems >16TB, listing 2nd level directories breaks
- Why: reading the “small block” (`sbc`) system resource file returns wrong data
- Cause: ???

At least you can reproduce

No “works on my machine” this time

- First spend some time manually inspecting existing code
- Find a discrepancy, but can’t explain it
- Spend a little while reverse engineering VMFS again
- Still can’t explain it
- Start reverse engineering and re-implementing VMFS entirely from scratch again
- *Still* can’t explain it

Expected output

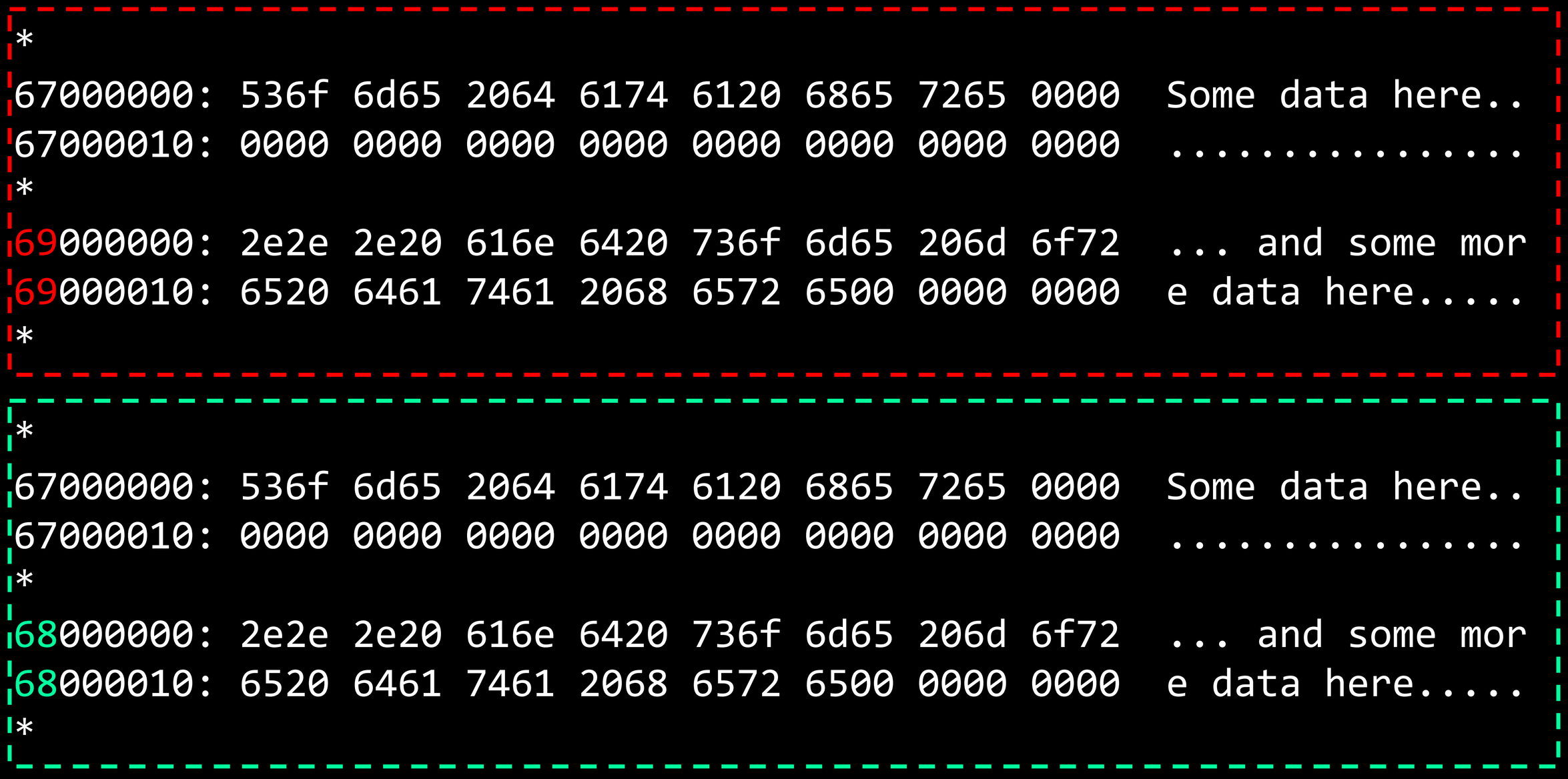
Spot the differences

```
*
67000000: 536f 6d65 2064 6174 6120 6865 7265 0000  Some data here..
67000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
*
68000000: 2e2e 2e20 616e 6420 736f 6d65 206d 6f72  ... and some mor
68000010: 6520 6461 7461 2068 6572 6500 0000 0000  e data here.....
*
```

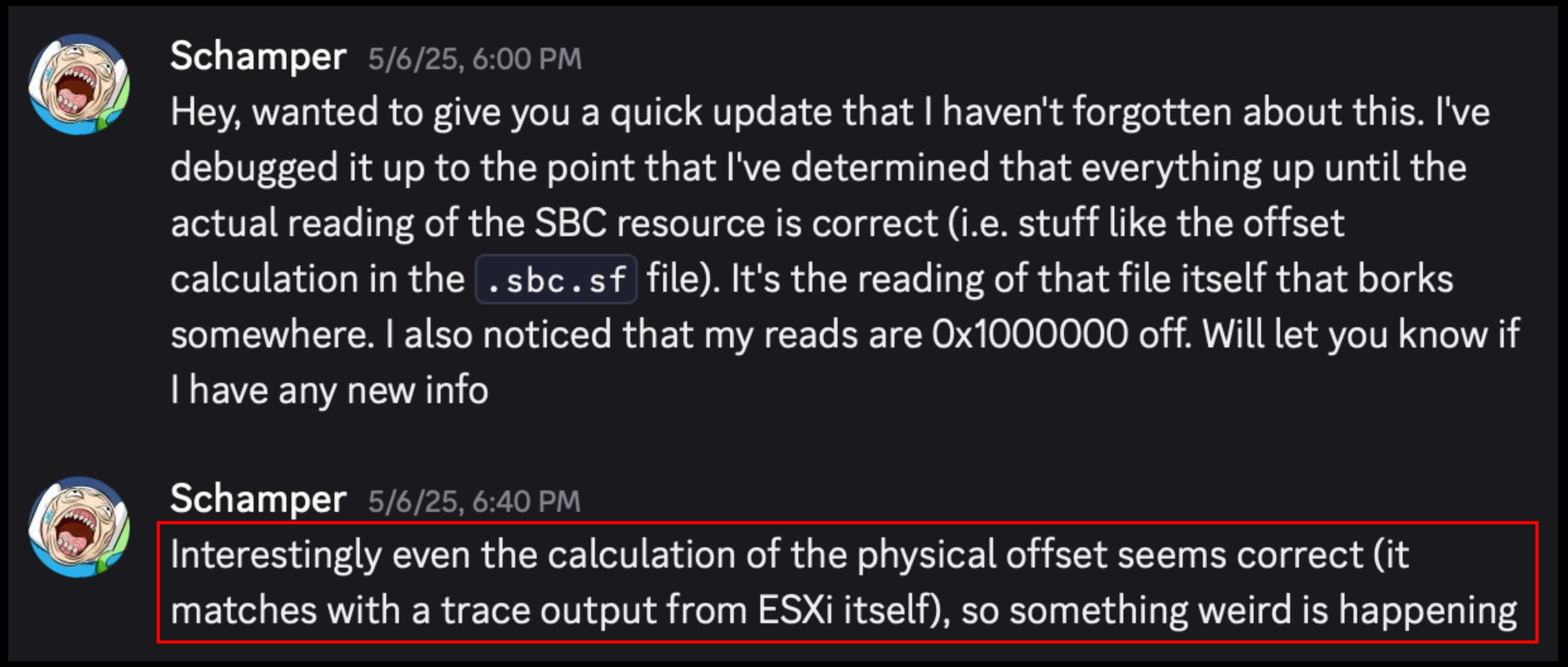
My output

Spot the differences

```
*
67000000: 536f 6d65 2064 6174 6120 6865 7265 0000  Some data here..
67000010: 0000 0000 0000 0000 0000 0000 0000 0000  ................
*
69000000: 2e2e 2e20 616e 6420 736f 6d65 206d 6f72  ... and some mor
69000010: 6520 6461 7461 2068 6572 6500 0000 0000  e data here.....
*
```


*
67000000: 536f 6d65 2064 6174 6120 6865 7265 0000 Some data here..
67000010: 0000 0000 0000 0000 0000 0000 0000 0000 ....
*
69000000: 2e2e 2e20 616e 6420 736f 6d65 206d 6f72 ... and some mor
69000010: 6520 6461 7461 2068 6572 6500 0000 0000 e data here.....
*
*
67000000: 536f 6d65 2064 6174 6120 6865 7265 0000 Some data here..
67000010: 0000 0000 0000 0000 0000 0000 0000 0000 ....
*
68000000: 2e2e 2e20 616e 6420 736f 6d65 206d 6f72 ... and some mor
68000010: 6520 6461 7461 2068 6572 6500 0000 0000 e data here.....
*

Status report



Side gig

Who knows what you might find?

- When reverse engineering anything, having symbols or debug info is worth gold
- My favorite: VirusTotal Retrohunt

```
rule yara_template
{
  meta:
    author = ""
    description = ""
    target_entity = "file"
  strings:
    $a = "IOCTLCMD_VMFS_GET_FILE_HANDLE"
    $b = "Res3_ClusterMetaVMFS6"
  condition:
    any of them
}
```

3 matches

!!!

ff18eba6f1e81c7464f978…

dissect/vmfs/c_vmfs.py

python

503b3f1c0fbbcb10670c80…

dissect/vmfs/c_vmfs.py

python

0656cea3b0b2d9106da208…

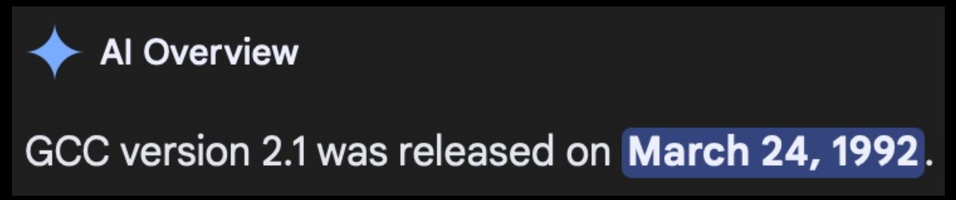
No meaningful names

elf shared-lib

What's this?

What's that?

- ELF32, GCC 2.x
  - Old
- Strings that look like symbols and types
  - Sections named `.stab`, `.stabstr`
  - IDA and Ghidra don't recognize it




```
FDS_VolInfo:T(58,17)=s32id:(58,18)=ar(12,33)
(4,6),0,256
FDS_VolInfo:t(58,19)=(58,17)
FS_GetAttrSpec:T(58,20)=eFS_ATTR_SPEC_BASIC:0,FS_A
FS_GetAttrSpec:t(58,21)=(58,20)
FS_IoctlAttr:T(58,22)=s8maxPartitions:(4,4),0,32
getAttrSpec:(58,21),32,32
FS_IoctlAttr:t(58,23)=(58,22)
FS_PartitionListResult:T(58,24)=s704readOnly:(4,1)
fsType:(58,25)=ar(12,33)
(0,2),8,96
versionNumber:(4,4),128,32
minorVersion:(4,6),160,8
rootDirOID:(58,10),176,544
name:(58,12),720,1024
```

The TL;DR

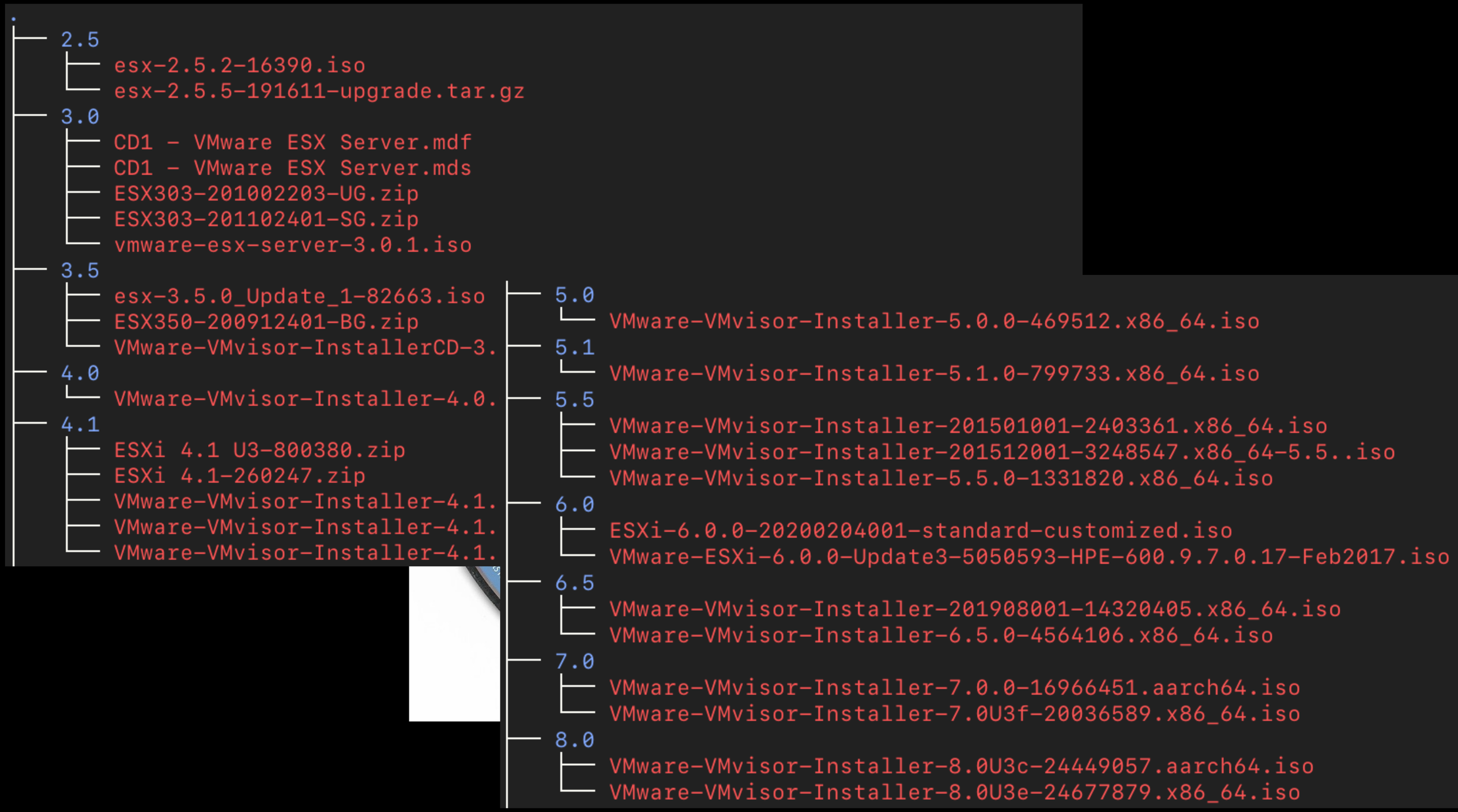
Because I'm probably starting to run out of time

- Random ESX3 binary
  - Turns out old (RPM based) versions of ESX shipped with debug builds
- STABS debug format
  - Dump with `objdump -g` to a `.h` file
- Time to start hoarding ESX(i) installation media

VirusTotal

INTERNET ARCHIVE

Google intitle:"Index of"


.
2.5
esx-2.5.2-16390.iso
esx-2.5.5-191611-upgrade.tar.gz
3.0
CD1 - VMware ESX Server.mdf
CD1 - VMware ESX Server.mds
ESX303-201002203-UG.zip
ESX303-201102401-SG.zip
vmware-esx-server-3.0.1.iso
3.5
esx-3.5.0_Update_1-82663.iso
ESX350-200912401-BG.zip
VMware-VMvisor-InstallerCD-3.
4.0
VMware-VMvisor-Installer-4.0.
4.1
ESXi 4.1 U3-800380.zip
ESXi 4.1-260247.zip
VMware-VMvisor-Installer-4.1.
VMware-VMvisor-Installer-4.1.
VMware-VMvisor-Installer-4.1.
5.0
VMware-VMvisor-Installer-5.0.0-469512.x86_64.iso
5.1
VMware-VMvisor-Installer-5.1.0-799733.x86_64.iso
5.5
VMware-VMvisor-Installer-201501001-2403361.x86_64.iso
VMware-VMvisor-Installer-201512001-3248547.x86_64-5.5..iso
VMware-VMvisor-Installer-5.5.0-1331820.x86_64.iso
6.0
ESXi-6.0.0-20200204001-standard-customized.iso
VMware-ESXi-6.0.0-Update3-5050593-HPE-600.9.7.0.17-Feb2017.iso
6.5
VMware-VMvisor-Installer-201908001-14320405.x86_64.iso
VMware-VMvisor-Installer-6.5.0-4564106.x86_64.iso
7.0
VMware-VMvisor-Installer-7.0.0-16966451.aarch64.iso
VMware-VMvisor-Installer-7.0U3f-20036589.x86_64.iso
8.0
VMware-VMvisor-Installer-8.0U3c-24449057.aarch64.iso
VMware-VMvisor-Installer-8.0U3e-24677879.x86_64.iso

TREND MICRO™

ZERO DAY INITIATIVE

LOOKING AT PATCH GAP VULNERABILITIES IN THE VMWARE ESXI TCP/IP STACK

July 27, 2022 | Reno Robert

CVE-2022-31696: AN ANALYSIS OF A VMWARE ESXI TCP SOCKET KEEPALIVE TYPE CONFUSION LPE

June 22, 2023 | Reno Robert

Attacking VMware NSX

Jan Harrie
jharrie@ernw.de
Matthias Luft
matthias.luft@rational-security.io

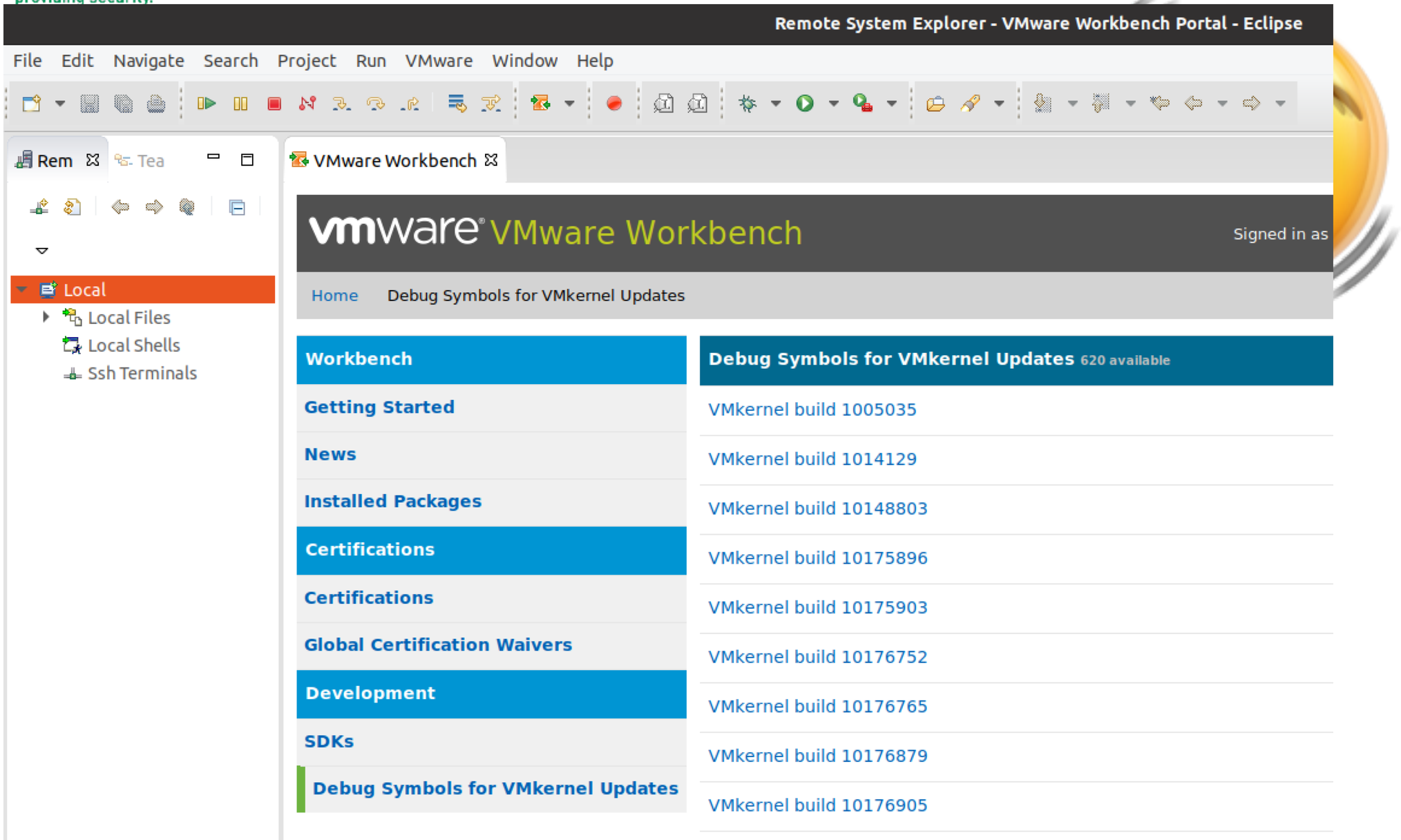
ERNW
providing security.

Prepare Debug Setup – Get Symbols

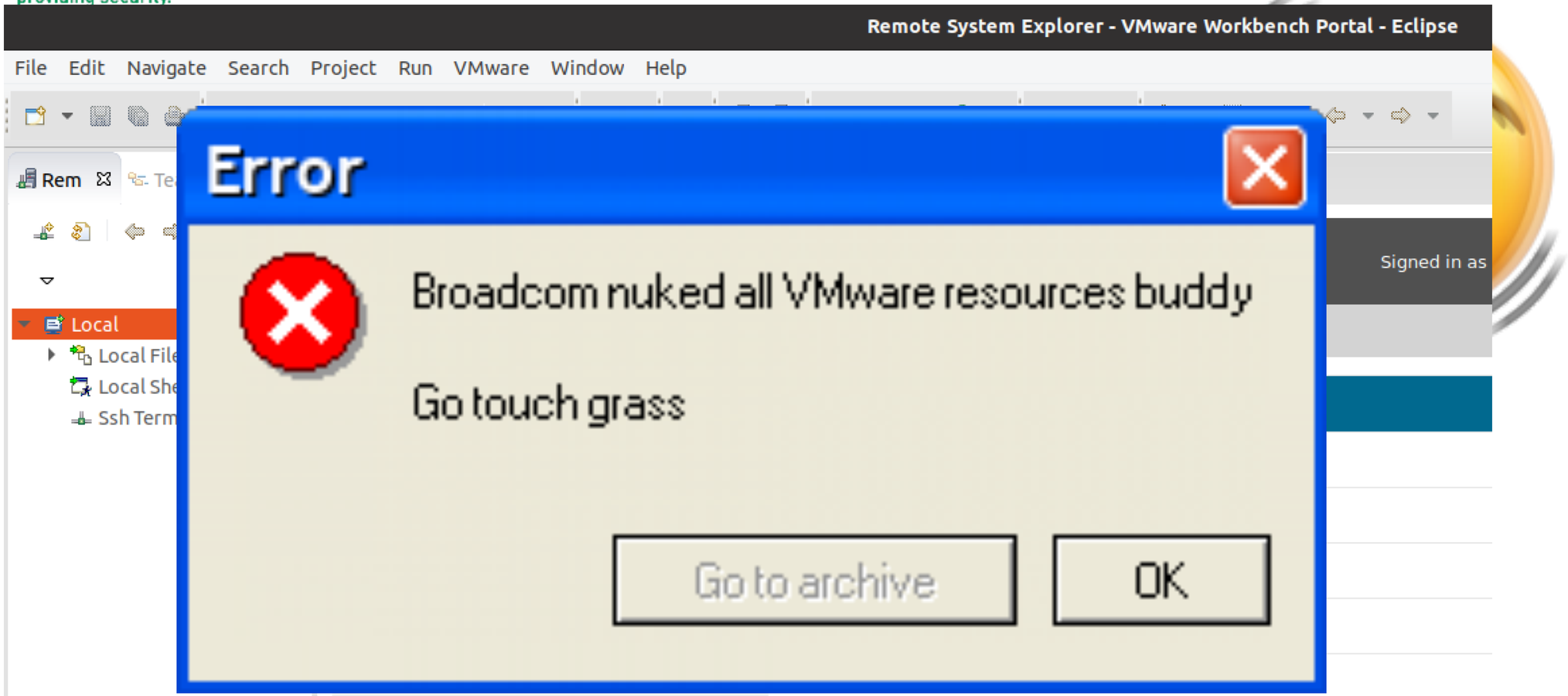
- Download Eclipse Plugin VMware Workbench
- Open the Dashboard
- Authenticate with VMware user credentials
- Goto menu point "Debug Symbols for VMKernel Updates"
- Search for the Debug Symbols for your build version
- Extract files symbols from RPM package

```
rpm2cpio vmware-esx-devtools-bridge-6.5.0-2.50.8294253.i386.rpm | cpio -idmv
```

ERNW
providing security.



ERNW
providing security.



TREND MICRO™ | ZERO DAY INITIATIVE

LOOKING AT PATCH GAP

RR

Reno Robert [redacted]

To: Erik Schamper

Wednesday, 9 July 2025 at 09:52

Caution: This is an external email. Please take care when clicking links or opening attachments. When in doubt, please report the item as Phishing.

Erik,

Please check if this is the file you are looking for. I'm not sure if the workbench works anymore, but I have attached the setup files and some of the downloaded files I had in the disk. Thanks for the reminder :)

ESXi.zip

The TL;TL;DR

I’m bad at keeping things short, did you notice?

- Debug symbols of all binaries until ESX 3.0
  - VMFS3 introduced here, close enough
- Debug versions of all binaries until ESX 3.5
- VMware Workbench hosted debug symbols until the Broadcom takeover
  - Only kernel, no modules or other binaries
    - Kernel only has a handful of useful VMFS constants
  - Shoutout to Reno Robert for a copy of 6.7 symbols!

```
typedef union %anon525 { /* size 512 */
  FS3_DiskLock dL; /* bitsize 2592, bitpos 0 */
  FS3_Heartbeat hB; /* bitsize 384, bitpos 0 */
  FS3_FileMetadata fileMeta; /* bitsize 608, bitpos 0 */
  FS3_ResFileMetadata resFileMeta; /* bitsize 224, bitpos 0 */
  FS3_ResourceClusterMD rcMeta; /* bitsize 800, bitpos 0 */
  uint8 diskBlock[512]:uint32; /* bitsize 4096, bitpos 0 */
} FS3_DiskBlock;
struct FS3_FileDescriptor { /* size 2048 id 526 */
  FS3_DiskBlock lockBlock; /* bitsize 4096, bitpos 0 */
  FS3_DiskBlock metaBlock; /* bitsize 4096, bitpos 4096 */
  union %anon527 { /* size 1024 */
    FS3PointerArray dataAddrs; /* bitsize 8192, bitpos 0 */
    struct FS3_RawDiskMap /* id 528 */ rdmMapping; /* bitsize 624, bitpos 0 */
    uint8 diskBlocks[1024]:uint32; /* bitsize 8192, bitpos 0 */
  } data; /* bitsize 8192, bitpos 8192 */
};
typedef struct FS3_FileDescriptor /* id 526 */ FS3_FileDescriptor;
```

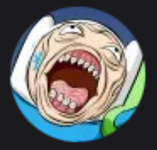
Armed with new knowledge

- LVM
  - High symbol coverage, easy to reverse
- VMFS5
  - Medium symbol coverage, reverse a lot from scratch
- VMFS6
  - Some reusable parts with VMFS5, reverse most from scratch
- `/usr/lib/vmware/esxupdate/systemStorage.zip` helps a tiny bit

I thought you were fixing a bug

How long did this little maneuver cost me?

June 2, 2025



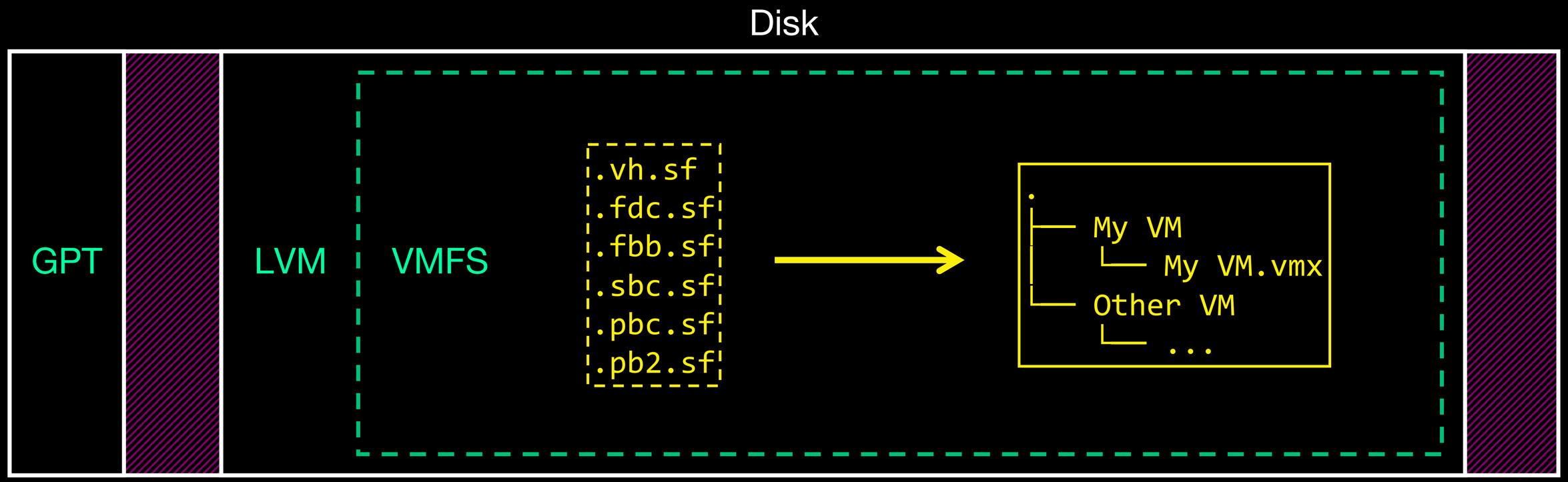
Schamper 6/2/25, 12:55 PM

Just letting you know that I've succesfully identified the source of the issue! Turns out it's a problem in the LVM implementation, not VMFS. I'm working on basically reimplementing dissect.vmfs from scratch using just this reverse engineering information as a basis

I've implemented this already in a rewritten LVM implementation and it works flawlessly, my own read .sbc.sf is now a hash match with one read from an ESXi machine. As a bonus, it now supports volumes spanning multiple disks too 😄.

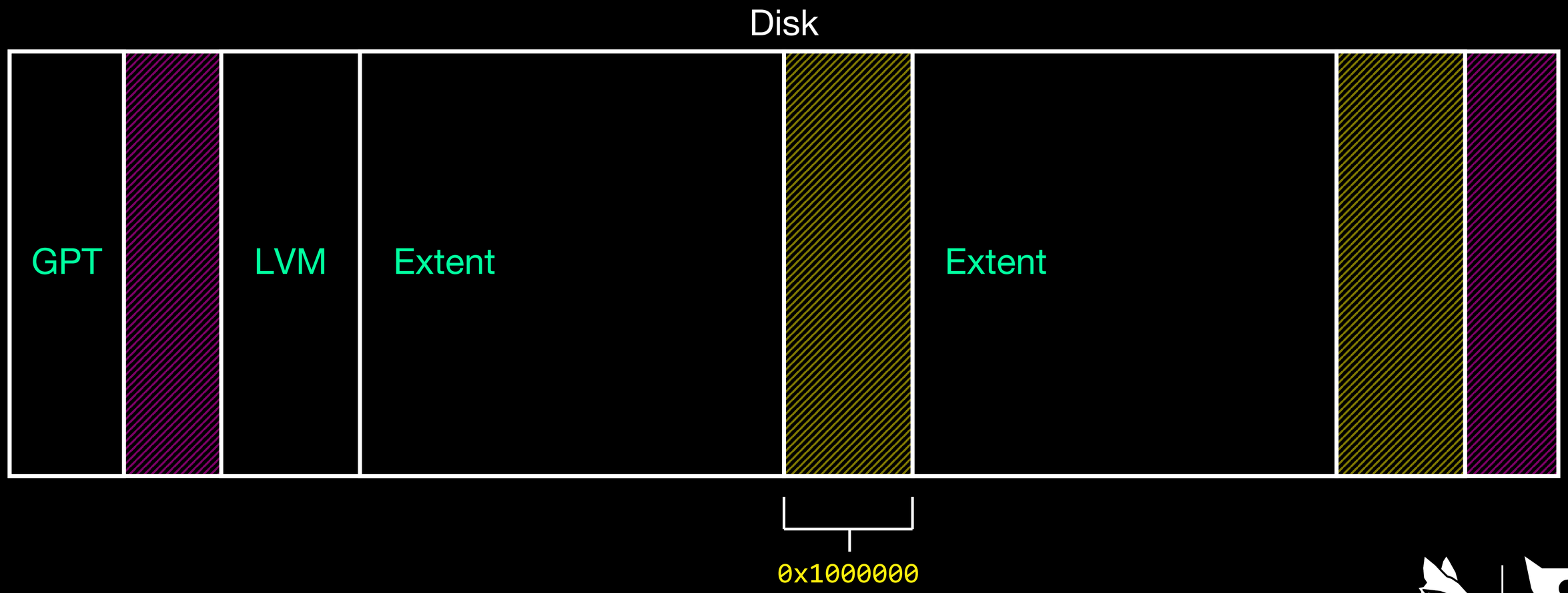
The bug

I found it



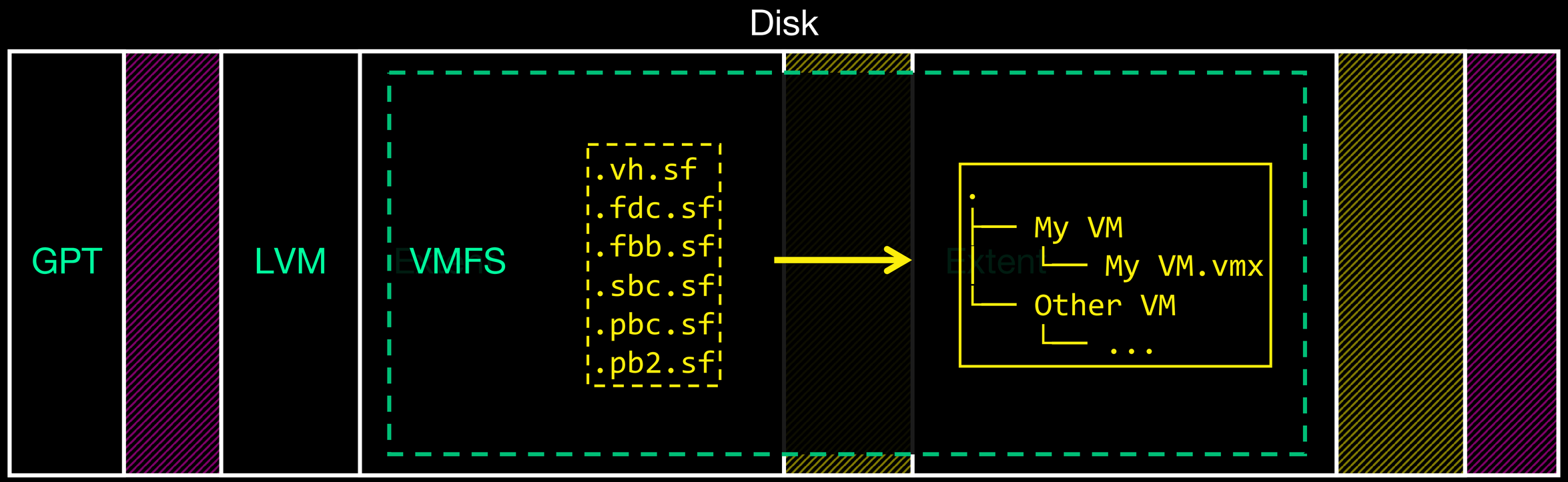
The bug

I found it



The bug

I found it



What are you still doing in IDA?

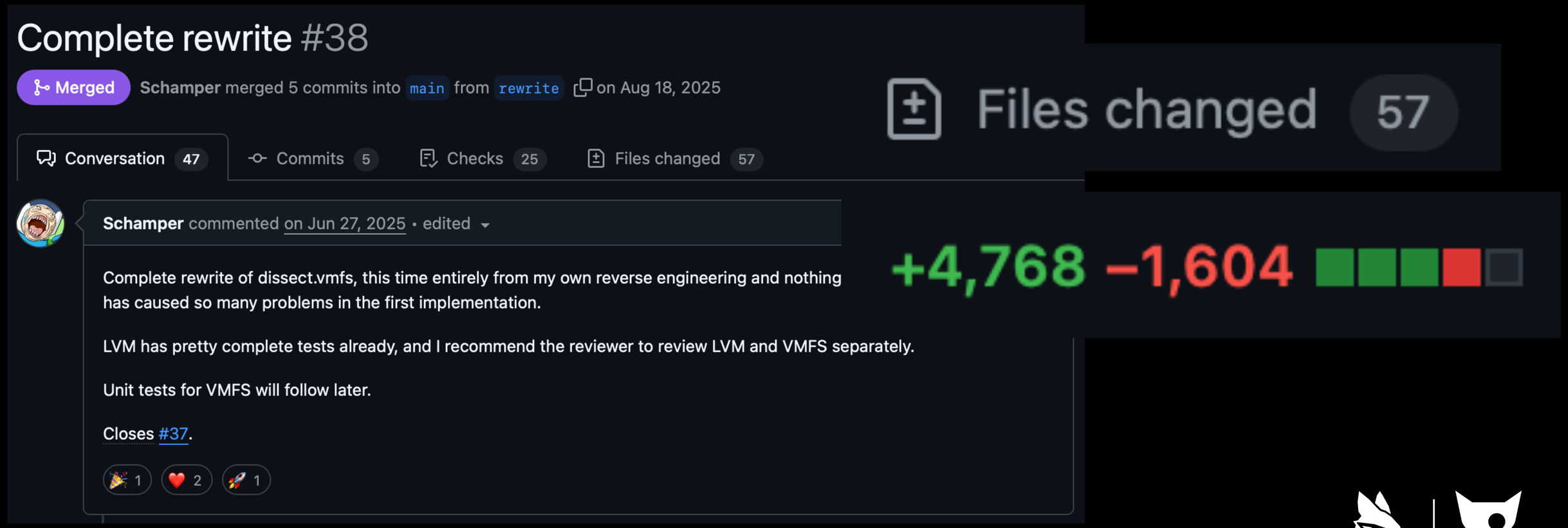
Finishing the fight

- ⚽-deep in VMFS at this point, might as well see this through
- Rewrite `dissect.vmfs` from scratch based on 100% my own reverse engineering

The fruits of my labor

And of my summer

- https://github.com/fox-it/dissect.vmfs/pull/38



What’s next?

Are you finally done?

- Working on a blog/report on VMFS
  - Would be a waste to keep this research locked in `.py` and `.idb` files
- Soon™

There’s no next slide

This is the end, you made it 🥳