Unransomware: From Zero to Full Recovery in a Blink

medium.com/@DCSO_CyTec/unransomware-from-zero-to-full-recovery-in-a-blink-8a47dd031df3

DCSO CyTec Blog November 4, 2024

DCSO CyTec Blog

--

This blog post discusses how we, DCSO's Incident Response Team (DIRT), were able to help an Akira ransomware victim restore their business critical data by extracting NTFS partitions from partially encrypted virtual disks. The post outlines a generic approach based on open source tools that allows affected parties to restore data from encrypted hypervisor systems. This approach works best for partially encrypted files and shows how to fix a bug in a commonly used mounting tool.

Blog post authored by enis Szadkowski, , Maike Orlikowski and Johann Aydinbas.

Hypervisor CPR

Ransomware incidents tend to hit organizations with low security posture the hardest because they lack the necessary preparedness to respond effectively. Over the past years, we have observed again and again how challenging recovery efforts are for these organizations. They often lack basic necessities for a speedy recovery, like spare hardware and disks.

In this particular ransomware incident, the attackers successfully obtained access to the ESXi hypervisor, allowing them to deploy the Linux version of Akira ransomware on the system. They shut down all virtual machines to then encrypt the corresponding .vmdk files (virtual disks). To help the customer recover their business critical data with the least amount of friction, we decided to boot a Linux OS on the ESXi host without installing it. In a second step, a spare disk of the same size was used as the ESXi datastore for recovery purposes. As good incident responders, we try to make the lives of our customers as easy as possible especially in hard times when their infrastructure is on fire and their business is standing still.

Locating the Datastore

After booting into the Linux OS, we first needed to identify the ESXi datastore that uses the “VMware VMFS” file system. This is the location where all the virtual disks (.vmdk) are stored. In the incident, these disks were encrypted by the Akira ransomware. To locate the datastore, we searched the output of the Linux command `fdisk` for the “VMware VMFS” file system:

```
# fdisk -l | grep "VMware VMFS" /dev/sda1 2048 13120307166 13120305119 6.1T VMware VMFS
```

You Shall Not Access!

After mounting `dev/sda1` with the help of `vmfs-fuse`, we encountered an interesting (and at the time, frustrating) bug in “vmfs-tools” that prevented us from accessing files larger than 256GB in the mounted datastore VMFS file system.

After trying to create a hex dump of the encrypted VDMK file `SERVER_2-flat.vmdk.akira` with `xxd`, we received an “Input/output error” instead of the expected output of the first 64 bytes of the file represented in both hex and ASCII. This file was the biggest virtual disk on the hypervisor and contained critical business data, so we had to find a way to access the virtual disk for mounting purposes.

```
# vmfs-fuse /dev/sda1 /mnt/vmdisk/# cd /mnt/vmdisk/# ls -lhtotal 811G-rw-r--r-- 1
root root 1.8K Jun  8 17:30 SERVER-6c6110af.hlog.akira-rw------- 1 root root 5.1M Jun
8 17:30 SERVER-ctk.vmdk.akira-rw------- 1 root root  81G Jun  8 17:30 SERVER-
flat.vmdk.akira-rw------- 1 root root 9.0K Jun  8 17:30 SERVER.nvram.akira-rw-------
1 root root 1.1K Jun  8 17:30 SERVER.vmdk.akira-rw-r--r-- 1 root root  558 Jun  8
17:30 SERVER.vmsd.akira-rwxr-xr-x 1 root root 4.1K Jun  8 17:30 SERVER.vmx.akira-rw--
------ 1 root root 3.7K Jun  8 17:30 SERVER.vmxf.akira-rw------- 1 root root 5.7M Jun
8 17:30 SERVER_1-ctk.vmdk.akira-rw------- 1 root root 181G Jun  8 17:31 SERVER_1-
flat.vmdk.akira-rw------- 1 root root 1.1K Jun  8 17:30 SERVER_1.vmdk.akira-rw-------
1 root root 5.5M Jun  8 17:30 SERVER_2-ctk.vmdk.akira-rw------- 1 root root 551G Jun
8 17:31 SERVER_2-flat.vmdk.akira-rw------- 1 root root 1.1K Jun  8 17:30
SERVER_2.vmdk.akira---------- 1 root root 2.7K Jun  8 17:32 akira_readme.txt-rw-r--r-
- 1 root root  18M Jun  8 17:30 vmware-10.log.akira-rw-r--r-- 1 root root 2.4M Jun  8
17:30 vmware-5.log.akira-rw-r--r-- 1 root root  25M Jun  8 17:30 vmware-6.log.akira-
rw-r--r-- 1 root root  19M Jun  8 17:30 vmware-7.log.akira-rw-r--r-- 1 root root 301K
Jun  8 17:30 vmware-8.log.akira-rw-r--r-- 1 root root  60M Jun  8 17:30 vmware-
9.log.akira-rw-r--r-- 1 root root 4.6M Jun  8 17:30 vmware.log.akira-rw------- 1 root
root 111M Jun  8 17:30 vmx-SERVER-2116517489-2.vswp.akira# xxd -l 64 SERVER_2-
flat.vmdk.akiraInput/output error
```

After some research, we discovered a Github issue in the “vmfs-tools” project that details the exact same bug. In the same Github issue, the user Marcus Sorensen provided a patch that fixes the issue.

Thank you Marcus!

It should be noted that as of today, the bug is still present in the package repositories of all major Linux distributions out there.

Next, we set up a build environment on the Linux system and then cloned the patched version of “vmfs-tools”. After successfully installing the patched version of “vmfs-tools”, we mounted the ESXi datastore once again. This time, the hex dump of `SERVER_2-flat.vmdk.akira` successfully printed the first 64 bytes of the file to `stdout`.

```
# apt remove vmfs-tools# apt install build-essential# apt install uuid-dev# apt
install libfuse-dev# apt install pkg-config# wget https://github.com/mlsorensen/vmfs-
tools/archive/refs/heads/double-pointer.zip# unzip double-pointer.zip# make install #
vmfs-fuse /dev/sda1 /mnt/vmdisk/# cd /mnt/vmdisk/# xxd -l 64 SERVER_2-
flat.vmdk.akira00000000: ccb4 630e 0e42 fb00 eb04 80df 72ef ba61
..c..B......r..a00000010: cf44 2690 ce31 f360 cc7c cafd a8e7 a3ed
.D&..1.`.|......00000020: f3b6 e08d 41e7 9f7a 771f ad0b eca1 8455
....A..zw......U00000030: d891 0cc9 71b0 63e2 86b2 1f2a b265 87e1  ....q.c....*.e..
```

Mind the Gap

With the patched version of “vmfs-tools”, we were finally able to read the contents of the Akira ransomware encrypted virtual disks. This was only a partial success — they were still encrypted after all. What now?

It is a common misconception that files encrypted with ransomware are not recoverable if the ransomware authors did a good enough job with the encryption logic. But this really depends on the type of file that was encrypted, its size and how exactly the encryption was performed.

The tactic used by Akira in this ransomware attack was to first shut down the virtual machine, and then followed by partial encryption of the VM virtual disk.

Partial or intermittent encryption is a technique where only parts of a file are encrypted. The specific pattern varies heavily among ransomware strains. Attackers sacrifice encryption completeness to achieve a much higher encryption speed. Faster encryption leaves defenders less time to react, and also evades typical detection logic based on how full encryption presents itself in CPU utilization, file similarity between unencrypted and encrypted files and I/O rate.

Akira ransomware specifically seems to apply two approaches to partially encrypting files: Reverse engineering the Linux variant shows it applies the same logic as the Windows variant, where files are only partially encrypted when they exceed a certain file size and then split into three or five chunks of alternating encrypted and plaintext blocks.

But the Linux variant also uses file extensions to decide between partial and full encryption. We let the Akira ransomware sample run against different 4MB files and checked whether the files were fully or partially encrypted. Some file types, like database files, were always fully encrypted, even when they exceeded the threshold for partial encryption. But all file types relevant to virtual machines, like `vhdx,vmdk,vdi`were always only partially encrypted.

What does all this mean for defenders? With partial encryption, large parts of the file are left unencrypted. In this case, this allowed us to attempt to recover partitions in the virtual machine disk without the need to decrypt them — because they had not been encrypted in the first place.

Find Waldo, DFIR Style

Our next task was to (hopefully) locate an intact NTFS partition in the unencrypted parts of the virtual machine disk. To find NTFS partitions, we searched for the NTFS magic bytes, a sequence of bytes that identifies the beginning of the NTFS header.

NTFS Magic Bytes

Instead of manually searching, we developed the following bash script to perform the search. The script uses the tool “ripgrep” to look for the NTFS magic bytes in the virtual disk file. Then, Sleuth Kit’s “fsstat” is used to check the file system type found at the offset of the byte sequence.

```
#!/bin/bash


virtualdisk=$1

for offset in $(rg -obUa "\x4e\x54\x46\x53\x20\x20\x20\x20" $virtualdisk | awk -F":"
'{ print $1 }')do fsstat -o $((($offset-3)/512)) $virtualdisk 2> /dev/null if fsstat
-o $((($offset-3)/512)) $virtualdisk 2> /dev/null | rg -o "File System Type: NTFS";
then  printf "Offset: $(($offset-3)) \n\n\n\n" fidone
```

To illustrate, we performed a test run against the “SERVER_2-flat.vmdk.akira”, the file from the previous examples. A valid NTFS partition has been found at offset “368050176”:

```
$ ./find_ntfs.sh SERVER_2-flat.vmdk.akira


(...)


FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: NTFS
Volume Serial Number: 6E42D27C42D2490B
OEM Name: NTFS
Volume Name: System
Version: Windows XP


METADATA INFORMATION
--------------------------------------------
First Cluster of MFT: 786432
First Cluster of MFT Mirror: 2
Size of MFT Entries: 1024 bytes
Size of Index Records: 4096 bytes
Range: 0 - 312320
Root Directory: 5


CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 0 - 10395646
Total Sector Range: 0 - 83165182


$AttrDef Attribute Values:
$STANDARD_INFORMATION (16)   Size: 48-72   Flags: Resident
$ATTRIBUTE_LIST (32)   Size: No Limit   Flags: Non-resident
$FILE_NAME (48)   Size: 68-578   Flags: Resident,Index
$OBJECT_ID (64)   Size: 0-256   Flags: Resident
$SECURITY_DESCRIPTOR (80)   Size: No Limit   Flags: Non-resident
$VOLUME_NAME (96)   Size: 2-256   Flags: Resident
$VOLUME_INFORMATION (112)   Size: 12-12   Flags: Resident
$DATA (128)   Size: No Limit   Flags:
$INDEX_ROOT (144)   Size: No Limit   Flags: Resident
$INDEX_ALLOCATION (160)   Size: No Limit   Flags: Non-resident
$BITMAP (176)   Size: No Limit   Flags: Non-resident
$REPARSE_POINT (192)   Size: 0-16384   Flags: Non-resident
$EA_INFORMATION (208)   Size: 8-8   Flags: Resident
$EA (224)   Size: 0-65536   Flags:
```

```
$LOGGED_UTILITY_STREAM (256)   Size: 0-65536   Flags: Non-resident
File System Type: NTFS
Offset: 368050176

(...)
```

Climbing the Top of Mount Everest

After overcoming all the challenges with the mounting tool “vmfs-tools” and hitting the jackpot by finding a largely unencrypted NTFS partition, we were finally ready to start recovering the business critical data. We first executed Sleuth Kit’s “fls” to see the files and directories present at the discovered offset. Note that “fls” expects a sector offset for the “-o” parameter. To calculate the offset, we divided the byte offset of the NTFS magic bytes sequence by the sector size of the partition, which in this case was 512.

As expected, there really was a Windows system partition at the specified offset:

```
$ fls -o $((368050176/512)) SERVER_2-flat.vmdk.akirad/d 58-144-1: PerfLogsr/r 4-128-
4: $AttrDefr/r 8-128-2: $BadClusr/r 8-128-1: $BadClus:$Badr/r 6-128-4: $Bitmapr/r 7-
128-1: $Bootd/d 11-144-4: $Extendr/r 2-128-1: $LogFiler/r 0-128-6: $MFTr/r 1-128-1:
$MFTMirrd/d 57-144-1: $Recycle.Binr/r 9-144-17: $Secure:$SDHr/r 9-144-16:
$Secure:$SIIr/r 9-128-18: $Secure:$SDSr/r 10-128-1: $UpCaser/r 10-128-4:
$UpCase:$Infor/r 3-128-3: $Volumer/r 19029-128-3: bootmgrr/r 19019-128-1: BOOTNXTd/d
211043-144-25: Config.Msid/d 19026-144-1: Documents and Settingsd/d 70002-144-1:
Dokumente und Einstellungend/d 176062-144-1: Driversd/d 246515-144-1: Lexmarkd/d
95077-144-1: Okidatar/r 79021-128-1: pagefile.sysd/d 263-144-5: Usersr/- * 19132:
WIM440A.tmpr/- * 19135: WIM441B.tmpr/- * 19136: WIM441C.tmpr/- * 19137:
WIM441D.tmpr/- * 19138: WIM441E.tmpr/- * 19161: WIM441F.tmpr/- * 19162:
WIM442F.tmpr/- * 19165: WIM4430.tmpr/- * 19166: WIM4431.tmpd/d 311-144-6: Windowsr/-
* 54: WinPEpge.sysd/d 59-144-6: Program Filesd/d 152-144-6: Program Files (x86)d/d
197-144-6: ProgramDatad/d 70027-144-1: Programmed/d 250012-144-1: QUARANTINEd/d
78984-144-6: System Volume Informationd/d 233005-144-1: usrV/V 312320: $OrphanFiles
```

As a last step, we mounted the NTFS partition present at the previously identified offset in order to access the Windows system partition.

Finally, we were ready to get down to business and start our recovery activities.

```
$ sudo mount -o
ro,noload,loop,noexec,offset=$((368050176)),show_sys_files,streams_interface=windows
SERVER_2-flat.vmdk.akira /mnt/vmdk/$ ls -la /mnt/vmdk/total 1378133drwxrwxrwx 1 root
root       8192 Jun  5 21:42  .drwxr-xr-x 3 root root       4096 Jun 23 23:47  ..-
rwxrwxrwx 1 root root       2560 Nov 24  2015 '$AttrDef'-rwxrwxrwx 1 root root
0 Nov 24  2015 '$BadClus'-rwxrwxrwx 1 root root    1299456 Nov 24  2015 '$Bitmap'-
rwxrwxrwx 1 root root       8192 Nov 24  2015 '$Boot'drwxrwxrwx 1 root root
0 Nov 24  2015 '$Extend'-rwxrwxrwx 1 root root   67108864 Nov 24  2015 '$LogFile'-
rwxrwxrwx 1 root root       4096 Nov 24  2015 '$MFTMirr'drwxrwxrwx 1 root root
0 Nov 14  2013 '$Recycle.Bin'-rwxrwxrwx 1 root root          0 Nov 24  2015
'$Secure'-rwxrwxrwx 1 root root     131072 Nov 24  2015 '$UpCase'-rwxrwxrwx 1 root
root          0 Nov 24  2015 '$Volume'-rwxrwxrwx 1 root root     398356 Nov 14  2013
bootmgr-rwxrwxrwx 1 root root          1 Jun 18  2013  BOOTNXTdrwxrwxrwx 1 root root
0 Jul 20  2020  Config.Msilrwxrwxrwx 2 root root         15 Aug 22  2013 'Documents
and Settings' -> /mnt/vmdk/Userslrwxrwxrwx 2 root root         15 Nov 24  2015
'Dokumente und Einstellungen' -> /mnt/vmdk/Usersdrwxrwxrwx 1 root root          0 Jul
22  2019  Driversdrwxrwxrwx 1 root root          0 Nov 21  2023  Lexmarkdrwxrwxrwx 1
root root          0 Nov  4  2019  Okidata-rwxrwxrwx 1 root root 1342177280 Apr 15
08:12  pagefile.sysdrwxrwxrwx 1 root root          0 Aug 22  2013  PerfLogsdrwxrwxrwx
1 root root       8192 Jun  5 21:25  ProgramDatadrwxrwxrwx 1 root root       8192 Mai
2 09:41 'Program Files'drwxrwxrwx 1 root root       4096 Nov 21  2023 'Program Files
(x86)'lrwxrwxrwx 2 root root         23 Nov 24  2015  Programme -> '/mnt/vmdk/Program
Files'drwxrwxrwx 1 root root          0 Jun  5 21:42  QUARANTINEdrwxrwxrwx 1 root
root       4096 Jun  8 19:01 'System Volume Information'drwxrwxrwx 1 root root
4096 Nov 26  2015  Usersdrwxrwxrwx 1 root root          0 Mai  2 09:40  usrdrwxrwxrwx
1 root root      28672 Jun  8 19:15  Windows
```

The remaining recovery procedure is left as an exercise to the reader ;-).

Conclusion

Because partial encryption is so common with different ransomware variants, we have made it a habit to search NTFS partitions in ransomware encrypted virtual disks. Partial encryption has become a commonly used tactic, leaving a gap (or many) for DFIR experts to recover large swaths of data. If you decide to use the method outlined in this post, make sure to apply the necessary patch to “vmfs-tools” if you installed the tool from your Linux distro’s repositories.

Sources