v. 2, n. 8, 2022

Scientific Journal of Applied Social and Clinical Science

ANALYSIS AND RECOVERY OF VIDEOS FROM THE WFS FILE SYSTEM

Galileu Batista de Sousa
IFRN
Polícia Federal
Natal, RN, Brasil

Unaldo de Oliveira Brito
IFRN
Natal, RN, Brasil

INTRODUCTION

Surveillance systems based on low-cost DVR (Digital Video Recorder) equipment have become essential elements in public and private security. Despite this, little emphasis is given to the process of forensic analysis of the videos they store. Faced with an event to be investigated, the difficulty of locating and treating the scenes is perceived; the procedures employ the recorder itself for visualization and extraction, without the expected forensic rigor and with restricted functionality. These difficulties stem from limitations or protections in the embedded software and from proprietary strategies for storing videos.

The demands associated with recording video on DVR systems are different from those on typical file systems (EXT, NTFS, FAT32). In general-purpose systems, files tend to have great variability in size, access control and concurrency strategies are fundamental, and there is primacy in the preservation of already recorded data. In DVRs, fast file systems are needed, which handle small and large files, without the requirement of access control and older files must be replaced as the media fills up (YANG; LI; WU, 2015) These conditions led DVR manufacturers to create their own file systems, including: DAHUA (DAHUA, 2021), HIKIVISION (HAN; JEONG; LEE, 2015) and WFS (YANCHARSKY, 2018).

Conventional operating systems such as Linux and Windows do not support these file systems. The same is true of file system interpretation libraries such as The Sleuth Kit (TSK) and digital evidence processing environments such as FTK, Encase and IPED.

The recovery of videos recorded by DVRs in proprietary file systems has been addressed repeatedly in the literature. LI, YANG and WU (2015) suggest a recovery method based on data carving, based on H.264 video format signatures, to recover videos in DAHUA and HIKVISION file systems.

Signature-based recovery is the predominant technique in handling unknown file systems. ARIFFIN, SLAY and CHOO (2013) also employed signature recognition in recovering videos from the "RapidOS" file system.

A recurring problem in subscription video recovery is that fragments of scenes from multiple cameras are recorded in an interlaced fashion, resulting in incomplete video or thousands of fragments. A technique for reassembling the fragments in H.264 format was proposed by PARK and LEE (2014). DRESH (2017) recognizes the texts inserted in the video frames as a mechanism for separating the fragments captured by the various cameras.

TOBIN, SHOSHA and GLADSHEV (2014) take a different approach. Using behavioral analysis and reverse engineering techniques, they mapped the "AVTECH" file system. Video recovery takes place by browsing the data structures that map the videos to areas of the disk.

Regarding the WFS system, DRESH, GOUVEIA and ZAGO (2016) present the timestamp format present in the internal metadata of the videos. This way, they were able to identify the moments in which the videos present in the media were stored, with the aim of solving specific cases. As discussed later, the format they identify is also used in the file system's own internal structures.

Finally, a rudimentary identification of the internal structures of the WFS was published by TOMILIN (2016). Two commercial tools, HX-Recovery for DVR (HXDVR, 2021) and DVR Examiner (2018) also propose to recover videos on WFS systems, without disclosing the means they employ.

GOALS

This work dissects the structure of the WFS file system (versions 0.4 and 0.5), used in low cost DVR systems. At the time of writing this text, WFS, despite its popularity, had no support on Linux or Windows, no publicly available documentation, and no program or driver source code to support it. In addition to the organization of a WFS disk, expressed in the form of its internal data structures, two tools for recovering videos from media formatted with this file system are presented: a) a Python implementation capable of extracting the videos and their metadata, simply and efficiently; b) a decoder implemented in TSK, which allows its use alone or integrated with tools such as IPED and Autopsy.

METHODOLOGY

The decoding of the file system was fundamentally based on three criteria: a) inspection of a media formatted with WFS looking for data structures typical of file systems used in DVR, such as those identified by HAN, JEONG and LEE (2015); b) evaluation of previous works that partially and superficially describe the WFS, especially due to YANCHARSK (2018); c) behavioral analysis of commercial programs that interpret (albeit also partially) the WFS. By repeatedly applying these techniques, a model was built.

The first of the criteria allowed to find what would be the "Superblock" of the file system, as well as the beginning of video files based on the H.264 signature. The discovery of the "Superblock", combined with the already known descriptions of video allocation, generated the first version of the file system structuring model.

The behavioral analysis, as described by TOBIN, SHOSHA and GLADSHEV (2014) consisted of running the available programs and monitoring system calls using the Process Monitor (Procmon) of the Sysinternals package. This way the regions of the disk that were being read for the recovery of a particular file could be identified. When combined with the other techniques, the model was refined. The flowchart in figure 1 describes such actions.

The behaviors of two commercial programs were observed, HX-Recovery for DVR (HX-Recovery for DVR, 2018) and DVR Examiner (FORENSICS, 2018); both have free versions, but with severe limitations of their functionality. After several executions and combining with the first approaches, the model was incremented. It must be noted that these programs have errors in recovering videos. So, while the tools helped in understanding and validating initial ideas, they became obstacles as the frameworks became more delineated.

RESULTS AND DISCUSSION

The general structure of WFS can be seen in figure 2.

The first block (512 bytes) of a WFS disk contains a signature "WFS0.4" or "WFS0.5" in the first six bytes and the terminator "XM" in the last two bytes, as shown in figure 3. By convention, this area was called the "Header".

The "Header" identifies the file system as WFS; the "Superblock" stores information on how and where the data is stored. The "Index Area" stores a 32-byte descriptor for each video fragment and the "Data Area" stores the fragments that make up the videos.

The "Superblock" starts at disk offset 0x3000; in this space you can find information on how the videos are arranged, location of index and data areas, size of video fragments, number of fragments and descriptors, in addition to 4 bytes of signature (0xDEBC9A78), among others. Figure 4 has a view of a "Superblock".

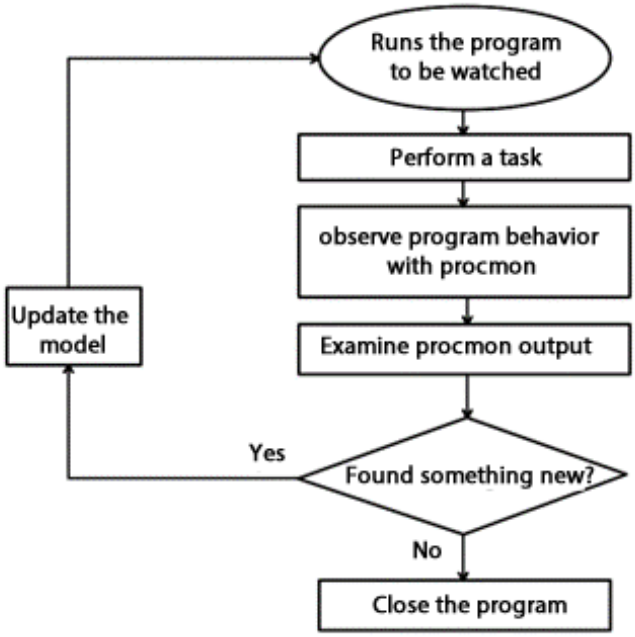

Figure 1 - Behavioral analysis methodology.

Figure 2 - WFS File System Layout.

```
Offset(h)   00 01 02 03 04 05  ···  0E 0F
0000000000  57 46 53 30 2E 34  ···  00 00  WFS0.4.....
⋮           ⋮                 ⋮          ⋮          ⋮
00000001F0  00 00 00 00 00 00  ···  58 4D  ..............XM
```

Figure 3 - first disk block.

```
03000  00 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00
03010  00 E0 06 3F 01 57 4F 40 4C CE 00 00 4C CF 00 00
03020  90 DF 00 00 4F 54 06 3F 01 E0 06 3F 00 02 00 00
03030  00 10 00 00 00 00 00 00 40 00 00 00 00 31 00 00
03040  18 00 00 00 BB 00 00 00 00 32 00 00 91 DF 00 00
03050  00 00 00 00 00 00 00 00 00 D2 00 00 10 00 00 00
03060  00 00 28 00 01 00 00 00 01 00 00 00 00 00 00 00
03070  4C CE 00 00 4C CE 00 00 00 00 00 00 00 00 00 00
03080  00 00 00 00 00 00 00 00 00 19 00 00 00 00 00 00
03090  00 32 00 00 00 40 00 00 00 00 00 00 1C 00 00 00
030A0  00 00 00 00 00 00 00 00 28 02 67 00 14 B8 64 00
030B0  00 00 00 00 00 00 00 00 00 82 00 00 00 50 00 00
  ⋮
03140  00 00 00 00 00 00 00 00 DE BC 9A 78 00 00 00 00
```

Figure 4 - the superblock.

Table 1 presents the fields identified in the "Superblock". All integers formed by more than one byte are in the little endian organization. Despite the succinct description, the real meaning is explained throughout the text, in the characterization of the other areas.

In general, file systems used in DVR use six bytes to reference the timestamp; WFS, however, uses four bytes, with its bits distributed as follows: six bits (31-26) for the year, four bits (25-22) for the month, five bits (21-17) for the day, five bits (16-12) for the hour, six bits (11-6) for the minute, and six bits (5-0) for the second. Figure 5 shows an example timestamp.

The value 0x3F06E000 can be verified in figure 5, which, when applied to the mapping, corresponds to the timestamp "03/12/2015 14:00:00". The beginning of the "Index Area" is indicated by 4 bytes starting from the 0x44 offset of the "Superblock". This area stores the descriptors of the video fragments, being exactly one descriptor of 32 bytes for each existing fragment. In other words, a descriptor characterizes a fragment of a video. A fragment is made up of several consecutive sectors on the disk

There are three types of descriptors: the main one, when the attribute byte (offset 0x01) is 0x02 or 0x03; the secondary descriptor, whose attribute is 0x01; and, the reserved fragment descriptor with value 0xFE.

The main descriptor of a video, in addition to referencing its first fragment, stores general information, such as the video start and end timestamps, the number of the camera that recorded it, the number of secondary fragments in the video and the position of the video descriptor. next fragment. Descriptors chained from the main descriptor are called secondary descriptors. Table 2 presents the fields present in a descriptor.

The secondary descriptor stores information only on the fragment it references, among them: position of the previous and posterior descriptors within the "Index Area", start and end timestamps of the fragment, position of the main descriptor within the "Index Area" and the camera number; these last two serve to confirm that a set of descriptors is part of the same video, as they are the same in all descriptors.

Descriptors whose attribute byte is equal to 0xFE (and other bytes equal to 0x00) are always found at the beginning of the "Index Area" and in the amount indicated by the 4 bytes from the offset 0x38 of the "Superblock". They will be treated as reserved fragment descriptors. These descriptors map empty fragments in the "Data Area".

The starting position of a fragment descriptor in the "Index Area" is obtained by multiplying the fragment number by 32 (the size of each descriptor) and, at the end, adding the starting position of the "Index Area" (displacement 0x44 of the "Superblock"). Similarly, the data start position of a fragment is obtained by multiplying the fragment number by the fragment size (superblock offsets 0x2C and 0x30) and summing the start position of the "Data Area" (superblock offset 0x48).

According to Table 2, the camera is referenced in the last byte of each fragment descriptor, using a sequence of values: 0x02 for camera 1, 0x06 for camera 2, 0x0A for camera 3 and so on. That is, it starts at 0x02 and increments 0x04 for each subsequent camera number.

In general, a DVR system records one video per camera every period of operation (one hour, for example), as configured in the device's interface. This does not mean that the videos will have the same amount of fragments, as each camera captures a different scene, with its own movements, details and encoding demands. Nevertheless, at the beginning of each recording period,

Offset	Size	Information
0x10	4 bytes	Timestamp of the last period of videos that appears in the "Data Area"
0x14	4 bytes	Timestamp of the last video recorded by the equipment
0x18	4 bytes	Position, within the "Index Area", of the descriptor of the last fragment recorded by the equipment
0x1C	4 bytes	Position, within the "Index Area", of the descriptor of the first valid fragment after the descriptors of the fragments that will be overwritten
0x20	4 bytes	Total amount of fragments
0x24	4 bytes	Timestamp of the first valid fragment after the descriptors of the fragments that will be overwritten
0x28	4 bytes	Start timestamp of the first period of "Data Area" videos
0x2C	4 bytes	Size of a disk block
0x30	4 bytes	Size of a video fragment, in number of disk blocks
0x38	4 bytes	Number of reserved fragments
0x44	4 bytes	Starting position of the "Indices Area"
0x48	4 bytes	Start position of the "Data Area"

Table 1 - Superblock information.

3F		06		E0		00	
0 0 1 1 1 1	1 1 0 0	0 0 0 1 1	0 1 1 1 0	0 0 0 0 0 0		0 0 0 0 0 0	
YEAR	MONTH	DAY	HOUR	MINUTE		SECOND	
15	12	03	14	00		00	

Figure 5 - Timestamp format in descriptor.

```
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00 02 04 00 00 00 00 00 90 02 00 00 00 F0 06 3F
00 00 07 3F 00 00 42 09 27 02 00 00 00 00 05 16
```

Figure 6 - Main descriptor.

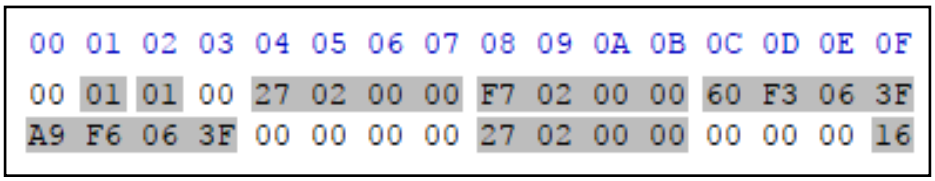

Figure 7 - Secondary descriptor.

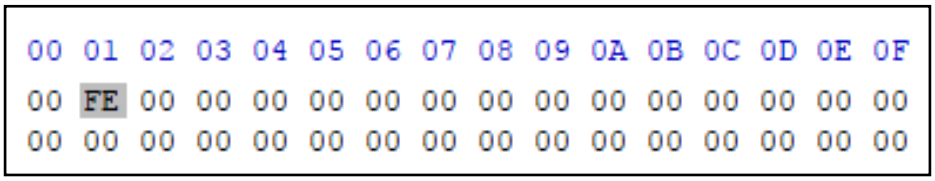

Figure 8 - Reserved Fragment Descriptor.

Offset	Size	Information
0x00	1 byte	It contains: 0x00
0x01	1 byte	Fragment attribute, if the value is 0x02 or 0x03 it is the descriptor of the first fragment of a video (main); if it is 0x01 it is a descriptor a video continuation fragment (secondary); if it is 0xFE it is a reserved fragment descriptor.
0x02	2 bytes	For the main descriptor, information about how many fragments there are, in addition to this one. For secondaries, their respective position after the primary
0x04	4 bytes	Position, within the "Area of Indexes", of the descriptor previous to the current one
0x08	4 bytes	Position, within the "Index Area", of the next video descriptor
0x0C	4 bytes	For the main descriptor, video start timestamp. For secondaries, start timestamp of the corresponding fragment
0x10	4 bytes	For the main descriptor, video end timestamp. For secondaries, timestamp end of corresponding fragment
0x16	2 bytes	Indicates the size of the last fragment, measured in number of 512-byte blocks. Appears in main descriptor and last descriptor
0x18	4 bytes	Position, within the "Index Area", of the main descriptor of the video
0x1E	1 byte	Order in which the video was recorded. It exists only in the main descriptor
0x1F	1 byte	Camera number

Table 2 - Descriptor information.

new videos are created for each camera in operation, represented by its main descriptors; secondary descriptors are created as needed. This structure is represented in figure 9.

It can be observed that, although the main descriptors follow an order, generally according to the camera number, the secondary descriptors do not follow the same order, that is, these fragments are recorded in a disordered way, according to demand.

The beginning of the "Data Area" is indicated by the 4 bytes at offset 0x48 in the "Superblock". This region contains all video fragments, occupying more than 99% of the physical disk area. The fragments have the same standard size, obtained by multiplying the 4 bytes at offset 0x2C (size of a disk block) by the 4 bytes at offset 0x30, both in the "Superblock". Only the last fragment of each video has a different size, indicated by multiplying the 2 bytes at offset 0x16 of the main descriptor by the size of a disk block (offset 0x2C of the superblock).

The fragments are arranged in the "Data Area", following a 1:1 mapping with the positions of their respective descriptors in the "Index Area", as shown in figure 10.

At the beginning of the "Data Area" there is an empty space, whose size corresponds to the number of fragments that are represented by the reserved descriptors. The amount of these fragments is at position 0x38 in the "Superblock". The reserved area size is therefore obtained by multiplying this value by the size of a fragment. On several analyzed media, the number of reserved fragments was 0x40, which corresponds to a standard fragment size (2MB) at 128 MB. It was observed that in some media this space keeps a backup of the "Index Area". However, in other media, this copy of the "Index Area" was not found in the reserved fragments region, showing that this type of situation depends on the features available in certain DVR models.

The size of the "Data Area" and, consequently, that of the "Index Area" are determined by the amount of fragments – indicated by the 4 bytes at offset 0x20 of the "Superblock". During the operation of the recording equipment, when these limits are reached, the existing fragments/descriptors are overwritten from the beginning of their respective area. Figure 11 shows how the WFS cyclic storage mechanism works.

In the "Superblock", the four bytes at offset 0x28 indicate the timestamp of the first period of videos that appears in the "Data Area", while the four bytes at offset 0x14 indicate the timestamp of the last period of videos recorded by the DVR equipment. The position of the descriptor of the last fragment recorded in the "Index Area" is indicated by the four bytes of offset 0x18.

Right after the last recorded fragment, there is a sequence of fragments that are marked to be overwritten, that is, they are not valid to be displayed by the equipment. The first valid fragment to be displayed has its timestamp indicated by the four bytes of offset 0x24, and the position of its descriptor, within the "Index Area", indicated by the four bytes of offset 0x1C. The timestamp of the last period of videos that appears in the "Data Area" is indicated by the 4 bytes of offset 0x10.

To demonstrate how to extract a single video from a disk formatted with WFS, we will use the information contained in the "Superblock" of figure 4 and a real example of media, whose part of the "Superblock" is in figure 12.

Initially, one must proceed with the collection of location information from the beginning of the "Index Area", beginning of the "Data Area", block size, fragment size and number of fragments, all contained in the "Superblock".

This information is enough to start the

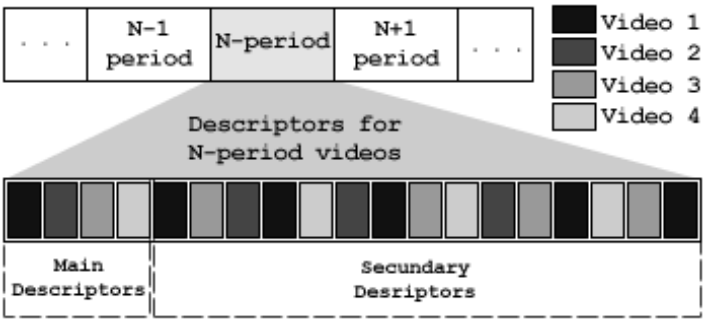

Figure 9 - Structure of the "Index Area".

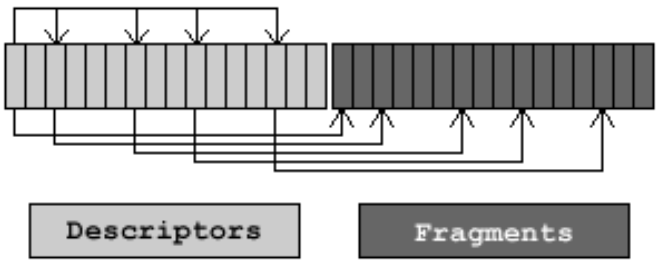

Figure 10 - Representation of the arrangement of the fragments.

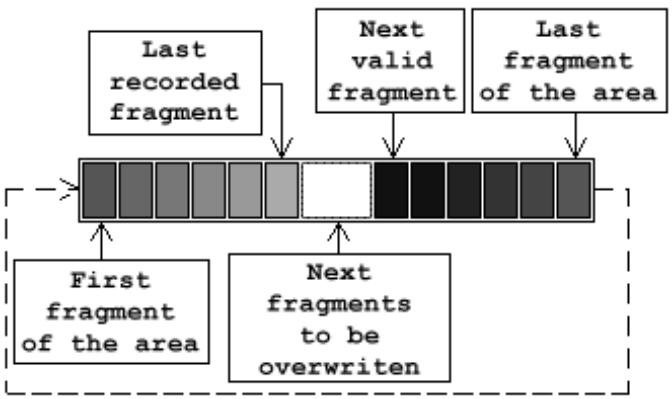

Figure 11 - Representation of the form of storage.

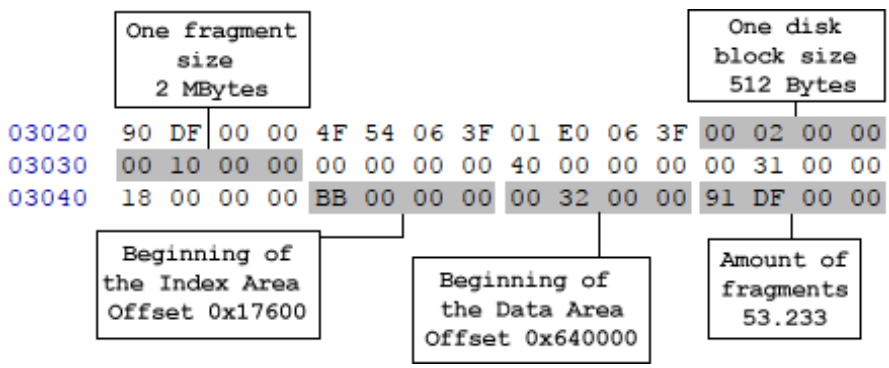

Figure 12 - Superblock Information.

search for the descriptors, as shown in figure 12. Starting from position 0x17600 on the disk, we proceed with the identification and storage of the main descriptors (type attribute equal to 0x2 or 0x3). At this point, it is already possible to list the existing videos, by separating the metadata contained in the main descriptors (start, end, size and camera), which, as seen before, refer to the video in its entirety.

For the same analyzed media, figure 13 shows the identification of the main information contained in one of the main descriptors.

Considering the specific video, whose main descriptor is in figure 13, it is observed that it has 5 fragments, four of them with 2 MB and the last one with 1,213,400 bytes, forming a video with a total size of 9.16 MB. The other metadata are also shown in the same figure.

After getting the main descriptor, you can find all the secondary descriptors. Figure 14 shows all secondary descriptors associated with the primary descriptor in figure 13.

In figure 14 it can be seen that the second byte of each descriptor identifies them as secondary, with their order indicated by the third byte. In the bytes at offset 0x08 of each descriptor, is the position, within the "Index Area", of the next secondary descriptor for that video. In the last fragment, in the 2 bytes of the offset 0x16, is the information of the size of its respective fragment. Finally, to confirm that these descriptors belong to the same video as the main descriptor in figure 13, the position information of the main descriptor and the camera, respectively, can be observed at offset 0x18 and in the last byte.

After separating all the descriptors for a given video, the fragments can be retrieved in the "Data Area", considering that there is a 1:1 mapping between the positions of the descriptors within the "Index Area" and their respective fragments in the " Data Area".

Finally, just use the position information of each descriptor within the "Index Area" and map it to the "Data Area", knowing that, from the information in the "Superblock", the beginning of the "Data Area" is at position 0x640000 on the disk and that each fragment, with the exception of the last one, is 2 MB in size. By joining the fragments into a single file, the video is extracted completely and individually from the disc.

WFS EXTRACTOR TOOL

Based on the information identified, the WFS Extractor video extraction tool was developed, implemented in Python and, therefore, supported on the main platforms on the market, including Linux and Windows.

WFS Extractor allows the listing of the videos present in a media (or forensic image of it), makes the listing and the recovery of the videos. You can choose a particular video or a set of them for extraction. The figure 14.

DECODIFICADOR WFS NO THE SLEUTH KIT

Using a tool like WFS Extractor to recover videos from WFS-formatted media is important, but it introduces a component outside the forensically focused digital evidence processing flow. To address this issue, a recognizer for this file system was implemented in The Sleuth Kit, version 4.9.0. This way, the command line toolset offered by TSK, which is a reference in the computer forensics community, can be used to browse and extract videos and their metadata in the conventional way used to access files stored on other systems.

Another consequence of implementing the WFS decoder in TSK is that integrated forensics environments can use it transparently. In this sense, the version of TSK incorporated into IPED, version 3.18.2, was replaced by the one with WFS processing

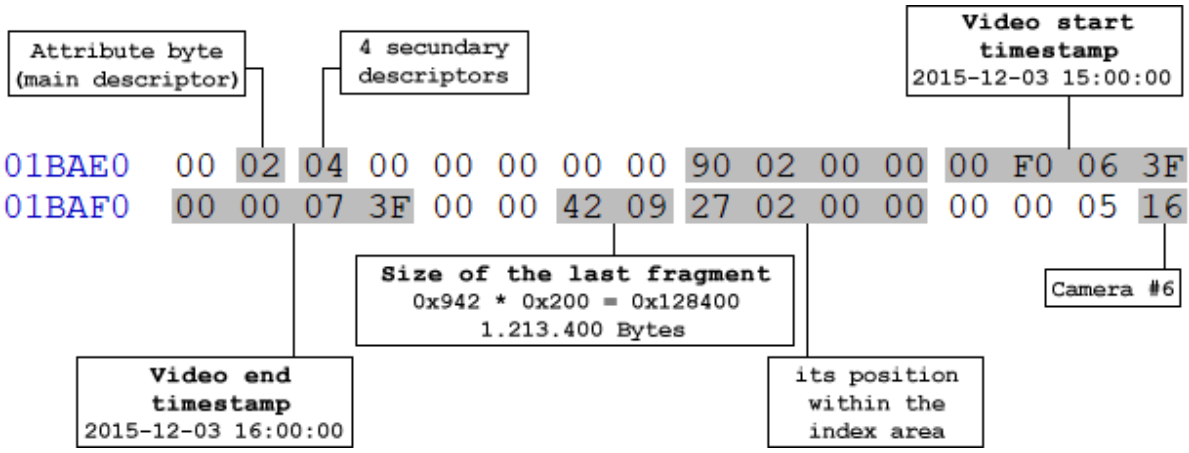

Figure 13 - Main Descriptor Data.

```
01C800  00 01 01 00 27 02 00 00 F7 02 00 00 60 F3 06 3F
01C810  A9 F6 06 3F 00 00 00 00 27 02 00 00 00 00 00 16

01D4E0  00 01 02 00 90 02 00 00 5D 03 00 00 A9 F6 06 3F
01D4F0  F0 F9 06 3F 00 00 00 00 27 02 00 00 00 00 00 16

01E1A0  00 01 03 00 F7 02 00 00 C4 03 00 00 F0 F9 06 3F
01E1B0  35 FD 06 3F 00 00 00 00 27 02 00 00 00 00 00 16

01EE80  00 01 04 00 5D 03 00 00 00 00 00 00 35 FD 06 3F
01EE90  00 00 07 3F 00 00 42 09 27 02 00 00 00 00 00 16
```

Figure 14 - Secondary Descriptors.

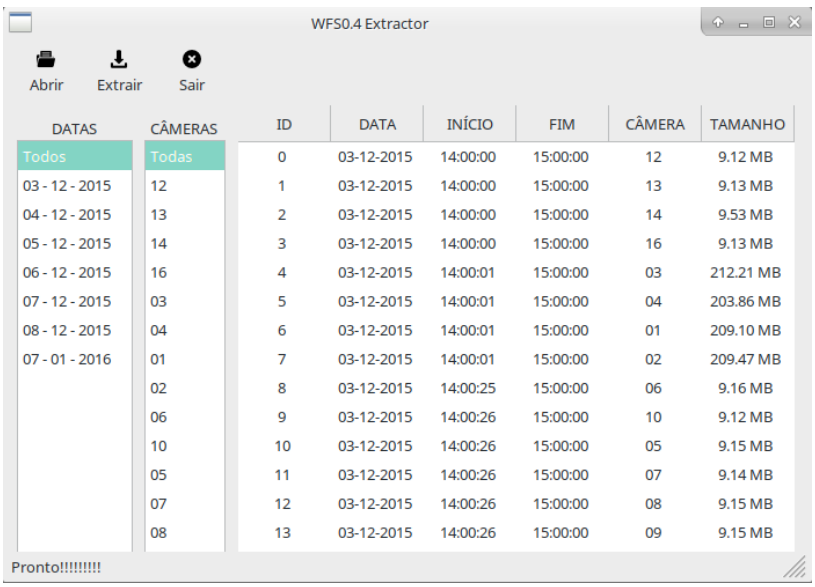

Figure 14 - List of videos in WFS0.4 Extractor.

enabled. Several media were processed normally. The other IPED functionalities, such as generating video thumbnails and filtering by date, could be used during the analysis, significantly speeding up the exams.

CONCLUSION

In this work, an almost complete identification of the data structures of the WFS file system (versions 0.4 and 0.5) was developed, used in several DVR devices. In addition, the data were validated through an extraction tool, applied to a dozen real media formatted with these file systems. The authors make the tool available through email contact.

There are still some aspects of WFS that are not fully understood, such as the real function of reserved fragments. However, given the various popular media and the superiority in video recovery when compared to commercial tools, it is understood that it can already be used in a forensic environment.

A next step of the present work is the development of drivers to incorporate the recognition of the file system in Windows and Linux, in order to facilitate the recovery of videos in non-forensic environments.

Keywords: Computer forensics; file systems; DVR; WFS.

REFERENCES

ARIFFIN, J.; SLAY, J. CHOO, K. Datarecovery from Proprietary-formatted CCTV Hard Disks. In: 9th IFIP Advances in Information and Communication Technology. p. 213-223. **Anais…** Orlando, USA, 2013.

DAHUA, Dahua Toolbox, 2021. Disponível em: dahuawiki.com/Software/Dahua_Toolbox. Acesso em: 31 jul 2021.

DME Forensics, DVR Examiner, 2021. Disponível em: dmeforensics.com/dvr-examiner-2-9-4. Acesso em 31 jul. 2021.

DRESCH, A. A. G. Relato de caso: Separação de registros de vídeo por meio da informação de câmera presente no conteúdo da imagem. In: Interforensics, p. 69. **Anais…** Brasília: DF. 2017.

DRESCH, A. A. G.; GOUVEIA, J. S. A.; ZAGO, R. L. Análise de timestamps de fragmentos de registros de vídeo em sistemas DVR como forma de identificação de operações realizadas no equipamento: estudo de casos. In: XI Seminário Nacional de Fonética Forense, o VIII Seminário Nacional de Perícias em Crimes de Informática e o III Seminário Nacional de Análise Forense de Imagens, p. 36-39. **Anais…** Florianopólis: SC. 2016.

HAN, J.; JEONG, D.; LEE, S. Analysis of the HIKVISION DVR File System. In: 7th. International Conference on Digital Forensics and Cyber Crime, p. 1-11, 2015. Disponível em: http://dx.doi.org/10.1007/978-3-319-25512-5_13. Acesso em: 31 jul. 2021.

HXDRV, HX-Recovery for DVR, 2021. Disponível em: http://www.hxdvr.com. Acesso em: 30 jul. 2021.

LI, R; YANG, F.; WU, C. Discuss on the Data Recovery Method of Embedded DVR. In: SHS Web of Conferences, v. 14, EDP Services, 2015. Disponível em: dx.doi.org/10.1051/shsconf/20151402010. Acesso em: 31 jul. 2021.

PARK, J.; LEE, S. GLADSHEV, P. Data fragment forensics for embedded dvr systems. Digital Investigation, v. 11, n. 3, p. 187-200, 2014.

TOBIN, L.; SHOSHA, A.; GLADSHEV, P. Reverse engineering a CCTV system, a case of study. Digital Investigation, v. 11, n. 3, p. 179-186, 2014.

TOMILIN, P. WFS0.4 sistema de armazenamento de stream de vídeo, 2016. (em russo). Disponivel em: https://disk-on.ru/articles/datarecovery_wfs04. Acesso em: 29 jul. 2021.

YANCHARSKY, P. Recuperando vídeo de um sistema de videovigilância usando WFS0.4, 2018. (em russo). Disponível em: hddmasters.by/articles/vosstanovlenie_iz_videoregistratora_wfs04.html. Acesso em: 29 jul. 2021.

YANG, F.; LI, R.; WU, C. Basic Principle and Application of Video Recovery Software for “Dahua” and “Hikvision” Brand. In: SHS Web of Conferences, v. 14, EDP Services, 2015. Disponível em: dx.doi.org/10.1051/shsconf/20151402010. Acesso em: 31 jul. 2021.