Attacking Digital Imaging and Communication in Medicine (DICOM) file format standard.

Markel Picado Ortiz (d00rt)
Spain, Basque Country, Bilbao, Gallarta.
d00rt.eus

Cylera Inc, New York

Abstract—**This paper describes why DICOM file format preambles can be considered a major flaw design as it can be used to embed executables and also malicious software. A DICOM file can be converted in an executable PE file. Once the conversion is done, the file can be interpreted as a valid DICOM image, or as a valid PE executable file (Windows), in other words, a polyglot file. The author has named this type of file PEDICOM. The DICOM file format is a widespread standard in hospitals around the world. Although it is not possible to execute these files manually, since it is necessary that a third person or program execute them, if no measures are taken, the following cases can occur in future cyber attacks targeting hospitals:**

1) **Intrusion into hospital infrastructure by hiding malware in DICOM images. Using social engineering through emails to hospital staff.**
2) **Intrusion into hospital infrastructure by uploading images to the hospital Picture Archiving and Communication System (PACS) using DICOM network protocol.**
3) **Intrusion into the devices of hospital patients hiding malware in DICOM images. Using social engineering by emailing patients.**
4) **Keep malware hidden on infected devices, as some antivirus programs do not interpret these images as executable and do not analyze them.**

Below is a summary of the interesting points of the DICOM file format and the PE file format. Finally, it is explained in detail how to abuse both file designs to create a PEDICOM polyglot file.

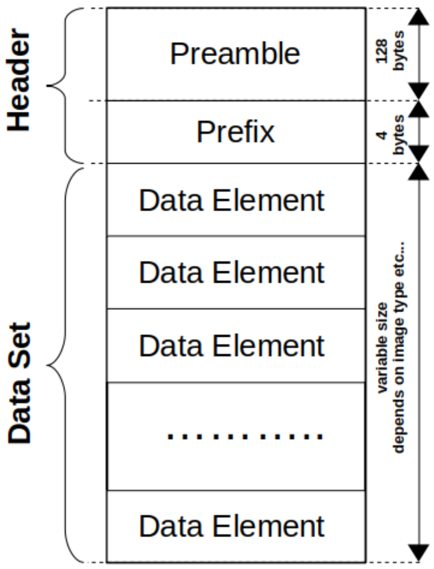
I. DICOM FILE FORMAT

A. Overview

Digital Imaging and Communication in Medicine (DICOM) is the globally recognized standard for the exchange of medical images designed by National Electrical Manufacturers Association (NEMA). It includes the definition of a file format and a network communication protocol. This paper is focused in the DICOM image format.

B. File format

The following image shows the general structure of a DICOM file



Two important blocks can be differentiated, the ***Header*** and the ***Data Set***.

The ***Header*** consists of two fields:

- Preamble:

  "It is intended to facilitate access to the images and other data in the DICOM file by providing compatibility with a number of commonly used computer image file formats." *(7.1 DICOM File Meta Information - NEMA)*

Size: *128 bytes.* ***Value:*** *Unknown - Depends on the application that uses this field.*

- Prefix:

  It is used as file magic.

  Size: *4 bytes.* ***Value:*** *"DICM"*

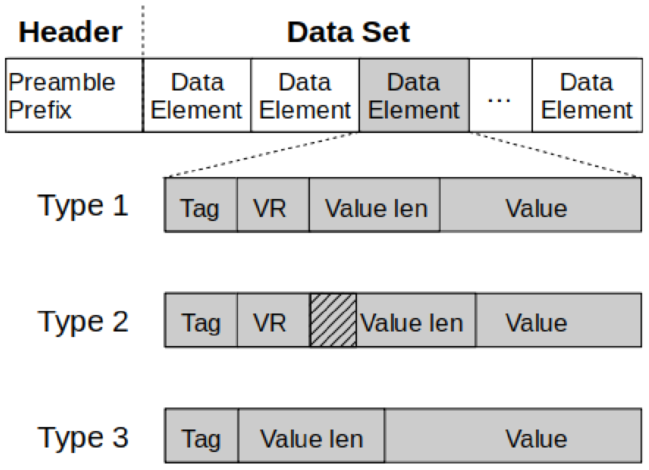
The ***Data Set*** block is a block composed of "n" elements of the ***Data Element*** type.

C. Data Element

The ***Data Element*** structure is described as following:

> "A Data Element is made up of fields. Three fields are common to all three Data Element structures; these are the Data Element Tag, Value Length, and Value Field. A fourth field, Value Representation, is only present in the two Explicit VR Data Element structures" *(7.1.1 Data Element Fields - NEMA)*

These Data Elements can be of three different types as shown in the following image.



- Tag: Identifies the attribute of the Data Element, represented in the format (XXXX,XXXX) with hexadecimal numbers.
- VR: Value representation (VR) that describes the data type and format of the attribute value.
- Value len: Length of the Value field.
- Value: Data related to the ***Data Element*** with *"Value len"* length

There are two things to keep in mind ahead:

1) The first Data Elements of a DICOM image are related to metadata of the image itself, the tag associated for these Data Element is the following (0002, XXXX). Then come other types of tags related to the patient, the type of X-ray and other tags.
2) There is a tag that is not parsed by most DICOM parsers as they are **private** tags. The associated tag for private attributes is (0009, XXXX).

A large list of attributes and their meanings can be found on the **dicomlibrary** website.

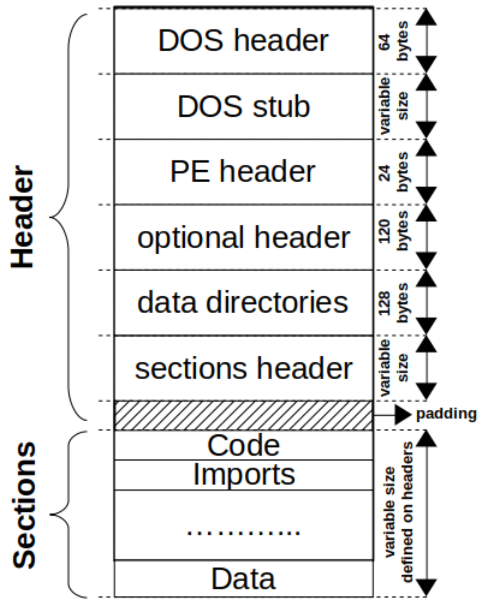
II. PE FILE FORMAT

A. Overview

The Portable Executable (PE) format is a file format for executable files, dynamic link libraries (DLLs) and others used in 32-bit and 64-bit versions of the Microsoft Windows operating system.

B. File format

In this paper is not explained in detail this format as there is already much information on the subject, it is simply important to know that the headers have information about how are stored the data/code it contains, both on disk and in memory.



The interesting points to create a PEDICOM polyglot file are:

1) The ***DOS header***, which has a pointer to the PE header, since the ***DOS stub*** has a variable

size and this allows us to move up or down the PE Header, along with the pointer of the DOS header.

2) The ***DOS Stub*** that has variable size.
3) The ***section header***, since it has the information of where is the beginning of each section in Disk. When a PEDICOM polyglot file is created, the DICOM part is added and this causes the file size to increase and the sections to move therefore the beginning of each section changes they will have to be fixed.
4) And finally the ***padding***. This is important since a PE has to be aligned in order to be loaded into memory correctly, and when more information is added, the padding has to be recalculated to align the headers and sections.

III. Creating a PEDICOM

A. Overview

This section explains how to merge both files, to get a single file that is valid as both DICOM and PE executable.

B. Merging PE format with DICOM format

1) The ***Preamble*** of the DICOM file occupies 128 bytes, and as its definition says, it is only used by third party applications so this field in most cases is initialized to 0. Therefore, as the ***DOS header*** occupies 64 bytes and the ***DOS stub*** is of variable size, the first 128 bytes are used to store the ***DOS header*** and the ***DOS Stub*** from the PE file.

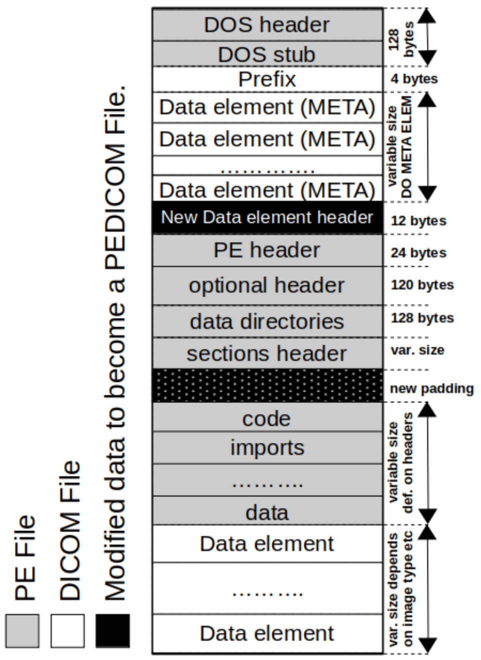
2) For a DICOM file to be valid, bytes 128 to 132 must contain the value ”DICM” i.e. the ***Prefix***.

3) From this point on, a DICOM parser starts to extract the ***Data Elements*** that have been described in detail in the DICOM section. Depending on the type of image, there are a series of hierarchies of tags that have to comply with the ***Data Elements*** of the image. But all DICOM image types have in common that the first elements are related to the metadata of the image itself so the tags of these elements are of the type (0002,XXXX). Therefore all ***Data Elements*** of the metadata type are copied after the ***Prefix***.

4) The parser what it expects now would be some specific tags corresponding to the type of image that has already been specified in the metadata. But, as explained, there are the private tags, that the parser will simply let them pass without analyzing. This is a good way to put the rest of what is left of the PE file masked as a private ***Data Element*** of the image.

5) To accomplish that, the rest of the data of the PE file (from the PE header to the end of the data) is encoded as a private Data Element. It is necessary to add 12 bytes, 4 bytes corresponding to the tag (0009,XXXX), 2 bytes corresponding to the VR, 2 reserved bytes and 4 bytes with the size of the data, in this case, the size of the rest of the PE file. In this way, a Data Element of type 2 is created.

6) Next, the rest of Data Elements are copied.



7) It is important to note the following, As the PE header has been moved down, since the DICOM metadata has been copied, there are

4 things that have to be fixed in order to be a valid PE file:

- The DOS header field **e_lfanew**, which points to the PE header. **e_lfanew** must point to the new address where PE header is.
- As the PE header has been displaced, the padding previously calculated by the compiler is not valid, so we have to recalculate it and add the necessary padding to align it to what the **FileAlignment** field of the Optional header indicates.
- From the point of view of the PE file, the size of the header has increased, since the DICOM metadata has been added, this size is set in the **SizeOfHeaders** field of the ***optional header***, reason why it will have to be modified with the new size of the header.

$$NewSizeOfHeaders = SizeOfHeaders + Prefix + MetadataDataElements + NewPadding + 12 \tag{1}$$

- It is evident that the sections have also been moved downwards by the new data that have been added, therefore it is necessary to modify the ***section headers*** to make the **PointerToRawData** field points to the new offset of the beginning of each section.

$$NewPointerToRawData = PointerToRawData + (NewSizeOfHeaders - SizeOfHeaders) \tag{2}$$

Following these steps, a PEDICOM polyglot file can be obtained. If this file has the extension ***.dcm** it will be opened with a DICOM image viewer if it is installed, but if it has the extension ***.exe** it will run as a program in Windows environments.

In case it has the extension ***.dcm** it can also be executed as a program, if it is executed from the windows terminal (cmd.exe), on the other hand, if it is executed from PowerShell it will open as an image with the default DICOM image viewer. This is something very curious and that someone with bad intentions could take advantage of.

IV. CONCLUSIONS

Using this technique an attacker could hide executable binaries in DICOM images, which could lead to a new way of distributing malware targeting hospitals. It would also be a way for an attacker to remain hidden in a system. Some antivirus solutions, do not detect this type of file as executable files so it would be a way to evade certain antivirus. As they are DICOM images, they can contain PII/PHI, very sensitive information, How would the antivirus work to analyze these files, would do it locally or would take these files to the cloud?

Would these files be quarantined? Would they have to be deleted to disinfect the system and the patient would have to undergo other X-ray tests?

A solution could be to wipe the first 128 bytes of the DICOM files to prevent them from being executed.

Finally, it is not known for us if this technique has been seen in the wild. For any doubt feel free to contact us.

NOTES: in the *Author's github repository*: https://github.com/d00rt/pedicom , you can find a sample of a PEDICOM image.
SHA256: 91a435e706b707070c31794383e856f15a74c247b59434a6b51f908a9bede37b75
In this repository you can also find a proof of concept about a binary that takes advantage of this weakness to infect a system. It would be the first FileInfector of these characteristics, which I have called PeDi2.
– Virus:Win32/PeDi2.A –
SHA256: d7ac8e740821f761a2f346a8dbb87a117301bba312f749ed48677910dfd5f6dc

References

[1] Healthcare devices protection, Cylera.
[2] DICOM file format, National Electrical Manufacturers Association (NEMA).
[3] PE file format, Microsoft.
[4] Polyglot files and PE file format, Corkami.