Inspection and Sanitization Guidance for Portable Network Graphic (PNG)

Version 1.0
26 November 2016

National Security Agency
9800 Savage Rd, Suite 6721
Ft. George G. Meade. MD 20755

Authored/Released by:
Unified Cross Domain Capabilities Office
cds_tech@nsa.gov

DOCUMENT REVISION HISTORY

Date	Version	Description
11/26/2016	1.0	Final release
12/13/2017	1.0	Updated Contact information, IAC Logo, Cited Trademarks and Copyrights, Expanded Acronyms, and added Legal Disclaimer

EXECUTIVE SUMMARY

The *Inspection and Sanitization Guidance for Portable Network Graphics (PNG)* provides guidelines and specifications for developing file inspection and sanitization software for PNG files. This ISG covers the PNG file format as documented by the Word Wide Web Consortium (W3C) and implemented by the open-source PNG library libpng [1].

PNG serves as a container format for images and image metadata. It is a raster graphics format; the bytes within the pixel array correspond to pixels in an image. The file format supports palette based images and transparent pixels. PNG is used for images on the Internet because of its lossless compression, color support, progressive display, and error detection.

This guidance document examines the PNG specifications for data attack, data hiding, and data disclosure risks that exist within the file structure. It provides a breakdown of each component of a PNG file and provides recommendations that can help assure that a PNG file is not only compliant with the specifications but also free of risk.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SCOPE

1.1 Purpose

The purpose of this document is to provide guidance for the development of a sanitization or analysis software tool for Portable Network Graphics (PNG) files. This document analyzes the various elements contained within PNG images and then discusses data attack, data disclosure, and data hiding risks. It describes how these elements can then be a cause for concern from hidden sensitive data or from attempts to exploit a system. This report provides numerous recommendations and mitigations that could be used to ensure the use of PNG is safe and that files conform to the specification.

The intended audience of this document includes system engineers, designers, software developers, and testers who work on file inspection and sanitization applications that process PNG images.

1.2 Introduction

File types that act as containers and store various types of data introduce a significant amount of risk including data hiding, data disclosure, and data attack risks to networks. PNG is an example of one of these file types because it can hold both images and text.

PNG is an extensible image format that groups data by chunks; chunks can contain image data or text. Chunks can contain information pertinent to rendering an image (e.g., color depth) or information that is not necessary to rendering the image (e.g., date modified). The structure of a PNG file can be amended to include custom chunks; that should not interfere with image processing applications because the specification requires that PNG processing software should ignore unrecognized chunks.

PNG images can be either lossy or lossless; however, this file format is better known for its non-patented, lossless compression. Because PNG also supports alpha channels (for transparency), error detection, and progressive display (using Adam7 interlacing), it is a widely used format on the Internet.

1.3 Background

The Graphics Interchange Format (GIF) was predominantly used prior to PNG. GIF uses the patented LZW compression scheme, and because the patent owner began to seek royalties from authors of software that supported GIF, there was a need for a replacement or successor. The first draft for PNG (pronounced "ping") was started in 1995 and the first version, PNG Specification Version 1.0, was published in October 1996. RFC 2083, PNG (Portable Network Graphics) Specification, was published on January 15, 1997. PNG files use the ".png" extension and "image/png" internet media type.

There were two versions of the PNG specifications released after version 1.0: 1.1 and 1.2; these two versions did not modify the file format but introduced new chunks and limits on chunk lengths.

1.4 Document Organization

This section summarizes the organization of this document.

Table 1-1 – Document Organization

Section	Description
Section 1: Scope	This section describes the purpose, introduction, background, organization, actions, and limitations related to this document.
Section 2: Constructs and Taxonomy	This section describes the constructs and taxonomy that are used throughout this document.
Section 3: Overview	This section describes the structure of PNG
Section 4: PNG Constructs	This section contains the PNG constructs that have risks and the options for mitigation.
Section 5: Acronyms	This section lists the acronyms in this document.
Section 6: Referenced Documents	This section lists the sources that were used to prepare this document.
Section 7: Summary of Risks	This section maps each construct to the corresponding specifications and risks.

1.5 Actions

Each construct description lists recommended actions for handling the construct when processing the image. Generally, inspection and sanitization programs will perform one of these actions on a construct: *Validate*, *Remove*, *Replace*, *External Filtering Required*, *Review*, or *Reject*.

The recommendation section in each construct lists each action that is applicable along with an explanation that is specific to the construct. Not all actions are applicable or appropriate for every context. As such, implementers are not expected to implement all the actions for a given risk; instead, they are expected to determine which action—or perhaps actions—applies best to their context. Definition of the criteria used to determine which action is "best" and of the specific method used to execute the action is left to the implementer.

Recommendations such as remove and replace may alter the integrity of PNG. It is important to address these issues in order to retain functionality.

NOTE

The recommendations in this document are brief explanations rather than a How-To Guide. Readers should refer to the construct description or official documentation for additional details.

This table summarizes the recommendation actions:

Table 1-2 – Recommendation Actions

Recommendation Action	Comments
Validate	Verify the data structure's integrity, which may include integrity checks on other components in the message. (This should almost always be a recommended action.)
Replace	Replace the data structure or one or more of its elements with values that alleviate the risk (e.g., replacing a username with a non-identifying, harmless value or substituting a common name for all authors).
Remove	Remove the data structure or one or more of its elements and any other affected parts.
External Filtering Required	Note the data type and pass the data to an external action for handling that data type (e.g., extract text and pass it to a dirty word search).
Review	Present the data structure or its constructs for a human to review. (This should almost always be recommended if the object being inspected can be revised by a human.)
Reject	Reject the file.

NOTE

No recommendations for logging all actions and found data are included here because all activity logging in an inspection application should occur "at an appropriate level" and presented in a form that a human can analyze further (e.g., the audit information may be stored in any format but must be parsable and provide enough information to address the issue when presented to a human.)

1.6 Document Limitations

This document covers PNG (Portable Network Graphics) Specification, Version 1.2.

There are derivative file formats, which include Animated Portable Network Graphics (APNG), JPEG Network Graphics (JNG), and Multi-image Network Graphics (MNG). These do not have wide-support among image processing applications and are therefore not included in this document.

1.6.1 Covert Channel Analysis

It is impossible to identify all available covert channels, whether in a file format or a communication protocol. It is impossible to identify all available covert channels, whether in a file format or a communication protocol. Because they contain free-form text, searching for hidden data becomes increasingly difficult. No tool can possibly analyze every channel, so this document highlights the highest risk areas to reduce or eliminate data spills and malicious content.

Additionally, this document does not discuss steganography within block text or media files, such as a hidden message that is embedded within an innocuous image or paragraph. Separate file format filters that specialize in steganography should be used to handle embedded content, such as text, images, videos, and audio.

2 CONSTRUCTS AND TAXONOMY

2.1 Constructs

This document describes many of the constructs used in PNG, but it does not describe every construct, thus this document is not to be treated as a complete reference. Developers of a PNG filter should consult the official specifications alongside this documentation for the full context. For each construct that is mentioned, the following sections exist:

- **Overview:** An explanation of the construct with examples.
- **Risks and Recommendations:** An explanation of potential risks posed by the construct with corresponding mitigation strategies.
- **Product**: The specifications in which the construct is found.
- **Location:** A textual description of where to find the construct.

2.2 Taxonomy

The following table describes the terms that appear in this document:

Table 2-1 – Document Taxonomy

Term	Definition
Construct	An object that represents some form of information or data in the hierarchy of PNG.
Inspection and Sanitization	Activities for processing files and protocols to prevent inadvertent data leakage, data exfiltration, and malicious data or code transmission
ISG	A document (such as this) that details a file format or protocol and inspection and sanitization activities for constructs within it.
Recommendations	A series of actions for handling a construct when performing inspection and sanitization activities.

3 OVERVIEW

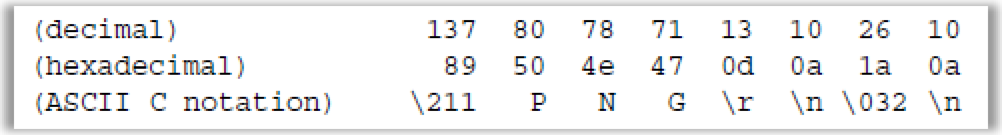
PNG is an image format with robust color support and error-free delivery of images. The specification was written to ensure that files were portable, well-compressed, and scalable. PNG uses network byte order[1] or big endian byte order and is a length-delimited file format; each chunk is prefaced with the length of the encapsulated data. The format allows for chunks not found in the specification to be inserted into the file without impeding its ability to be rendered by specification compliant software.

3.1 PNG File Structure

PNG is composed of two primary constructs: the file signature and the chunk. The file format is processed sequentially; the signature should be processed first and then the chunks. There are critical chunks that must exist for the file to be valid and ancillary chunks that are not critical for software to render PNG images.

3.1.1 File Signature

The file signature is made up of the first eight bytes of a PNG file. The signature contains a non-ASCII value (0x89), the name of the format in ASCII (0x50, 0x4e, and 0x47), a carriage return and line feed (0x0d, 0x0a) also known as CR-LF, the value for ctrl-Z (0x1a), and a line feed (0x0a) in that order. Figure 3-1 F, below, shows the decimal, hexadecimal, and ASCII values for the PNG file signature.



```
(decimal)              137  80  78  71  13  10  26  10
(hexadecimal)           89  50  4e  47  0d  0a  1a  0a
(ASCII C notation)    \211   P   N   G  \r  \n \032 \n
```

Figure 3-1 File Signature Values

3.1.2 Chunks

PNG Chunks consist of four fields: length, type, data, and a Cyclic Redundancy Check (CRC).

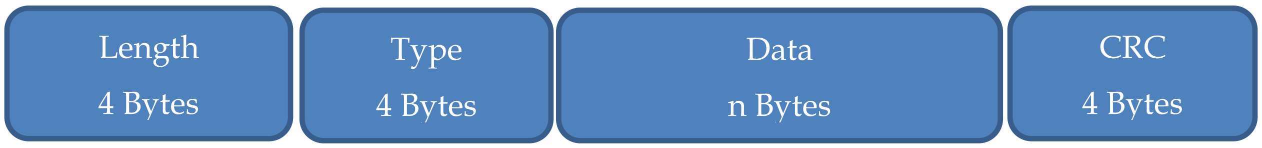


Figure 3-2 Chunk Format

[1] http://www.cs.umd.edu/class/sum2003/cmsc311/Notes/Data/endian.html

Figure 3-2 Chunk Format, above, depicts the chunk format and the number of bytes per field. The length is a four byte unsigned integer that contains the number of bytes in the data field; this value should not exceed 2^31-1 per specifications, despite being unsigned. The type field is a four byte chunk code that consists of only ASCII letters; this field designates the sort of data that will be encountered in the data field and whether it is necessary for rendering the image. The data field contains the number of bytes designated by the length field. The CRC is calculated[2] against the first three fields of a chunk (length, type, and data)[3] and must always be present. The CRC ensures that the integrity of information can be verified; in practice this field may be disregarded by applications that process PNG files. There are no separators/markers separating the fields.

Chunks can either be critical or ancillary; critical chunks are necessary for rendering an image, but ancillary chunks are not required. Table 3-1 and Table 3-2, below, contain the specification defined critical chunks and ancillary chunks respectively. For more information about each chunk refer to Section 4.2.2 for critical chunks and Section 4.2.4 for ancillary chunks.

Table 3-1 – Critical Chunks

Chunk Type	Name
IHDR	Image Header
PLTE	Palette Chunk
IDAT	Image Data
IEND	Image Trailer

Table 3-2 – Ancillary Chunks

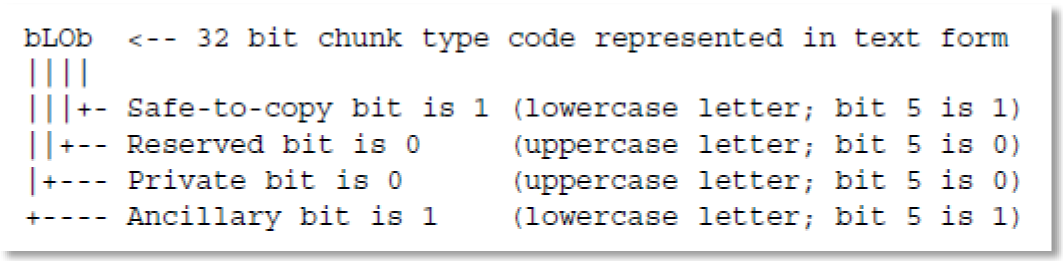
Chunk Type	Name
tRNS	Transparency
cHRM	Primary Chromaticities and White Point
gAMA	Image Gamma
iCCP	Embedded ICC Profile
sBIT	Significant Bits
sRGB	Standard RGB Color Space
tEXt	Textual Data

[2] https://www.w3.org/TR/PNG-Structure.html#CRC-algorithm
[3] Sample code can be found here: https://www.w3.org/TR/PNG-CRCAppendix.html

zTXt	Compressed Textual Data
iTXt	International Textual Data
bKGD	Background Color
hIST	Image Histogram
pHYs	Physical Pixel Dimension
sPLT	Suggested Palette
tIME	Image Last-Modification Time

3.1.2.1 **Chunk Type Naming Scheme**

Chunk types are case-sensitive four byte values that dictate how a decoder should process a chunk even if the chunk name is not recognized. Because the fifth bit of a byte controls whether or not an ASCII value is uppercase or lowercase (zero is uppercase and one is lowercase), it is important for processing PNG files. The fifth bit of every byte in the chunk type is referred to as the ancillary bit, private bit, reserved bit, and safe-to-copy bit in that order. Figure 3-3 shows an example ancillary chunk, bLOb, which is deconstructed.



```
bLOb  <-- 32 bit chunk type code represented in text form
||||
|||+- Safe-to-copy bit is 1 (lowercase letter; bit 5 is 1)
||+-- Reserved bit is 0     (uppercase letter; bit 5 is 0)
|+--- Private bit is 0      (uppercase letter; bit 5 is 0)
+---- Ancillary bit is 1    (lowercase letter; bit 5 is 1)
```

Figure 3-3 Deconstructed Chunk Name

The first byte contains the ancillary bit. If it is uppercase (i.e., bit five of the first byte is zero), then the chunk is critical; otherwise, it is ancillary. Critical chunks are necessary for the successful display of a files contents. Ancillary chunks are not “strictly necessary” [2] in order to display the contents of a file.

The second byte contains the private bit. If it is uppercase, then it is a public chunk; otherwise it is a private chunk. Private chunks are unregistered, organization-specific chunks for custom use, whereas public chunks are defined in the specification.

The third byte contains the reserved bit. This was incorporated for future expansion to the file format; the specification explains that this should presently always be set to 0, which means the letter will always be uppercase. PNG decoders should be able to process a lower case letter in the chunk name; however, there isn’t a feature that explicitly uses this yet.

The final byte contains the safe-to-copy bit. This field allows image editing applications to properly process unrecognized chunks in a file that has been modified. If the byte value is uppercase, then the chunk depends on the image data. This means that if a critical chunk is modified, this chunk should not be copied to the output PNG file; however, if the byte is lowercase, then the unrecognized chunk can be copied to a modified PNG file.

3.1.3 Filters

PNG implements filter algorithms that process image data (the bytes and not the pixels) before compression in order to optimize the compression. The filter algorithms take each scanline[4] and modify the values using a reversible algorithm in order to allow for better compression. For example, filtering a scanline with values 1 2 3 4 5 6 7 8 9 with the 'Sub' filter would yield 1 1 1 1 1 1 1 1 1, which lends itself to compression. The 'Sub' filter takes the rightmost value and subtracts the previous value, e.g. 9-8, and stores that in the current index. It then applies this until it hits the leftmost index.

To recover the original values an application would look at the first byte of the scanline, which indicates the filter function used and then recover the values reversing the operation applied to the scanline; in the previous example an application would recover the Sub filter values by adding preceding bytes sequentially to yield 1 2 3 4 5 6 7 8 9 again. An encoder can selectively choose which filter method to use per scanline basis; the filtered data will contain a filter-type byte before compression to indicate the filter used. The filter-type byte allows the data to be recovered during decompression.

Filters may require knowledge of the preceding byte, the corresponding byte in the previous scanline, and the byte preceding the corresponding byte in a previous scanline. If there isn't a previous byte or scanline, then the operation uses zero in the place of the non-existent value. For example, the 'Sub' filter requires computing the difference between a byte and the preceding byte; however, if the first byte of every line does not have a preceding byte, the specification requires computing the difference of the first byte and zero.

Table 3-3, below, shows the filter operations as functions. Note that 'Orig(x)' refers to the original value of any arbitrary byte x; Filt(x) refers to the value of any x after the filter has been applied to it. The PaethPredictor[5] function is defined in the PNG specification and is outside the scope of this document.

The values *a*, *b*, and *c* in the table below refers to pixels in specific positions relative to the original byte x. *a* refers to the preceding byte in the same scanline, *b* refers to the corresponding byte in the previous scanline, and *c* refers to the preceding byte of the corresponding byte in the previous scanline.

[4] A single row of pixels in a raster graphics image
[5] http://www.w3.org/TR/PNG-Filters.html#Filter-type-4-Paeth

Table 3-3 – Filter Functions

Value	Name	Filter Function
0	None	Filt(x) = Orig(x)
1	Sub	Filt(x) = Orig(x) - Orig(a)
2	Up	Filt(x) = Orig(x) - Orig(b)
3	Average	Filt(x) = Orig(x) - floor((Orig(a) + Orig(b)) / 2)
4	Paeth	Orig(x) - PaethPredictor(Orig(a), Orig(b), Orig(c))

Figure 3-4, below, depicts the positions of a, b, and c relative to byte x.

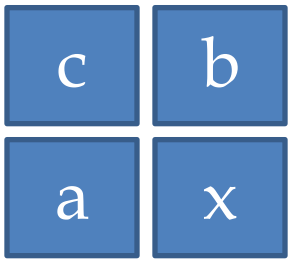


Figure 3-4 Filter Bytes

3.1.4 Compression

The PNG specification only defines one form of compression: deflate/inflate. Deflate[6] uses both the LZ77 algorithm and Huffman Coding. The compression method is set in the IHDR chunk (see Section 4.2.3.1). Certain ancillary chunks (iTXt, iCCP, and zTXt) can also be compressed, but they cannot be split up.

Compressed data is stored into data blocks, which can be split across IDAT chunks. The final data block will contain a marker bit to indicate that it is the last block. The IDAT chunks can split up the zlib data stream at any point and can be completely arbitrary. The zlib format encapsulates raw DEFLATE[7] data.

Note that uncompressing data and then recompressing it may yield different sizes. Some file formats containing compressed data may rely on the size of a subfile. If at all possible, files should be compressed in a manner that ensures it is equal to or smaller than the original file size to ensure the parent file works as intended.

[6] http://zlib.net/feldspar.html
[7] http://www.gzip.org/deflate.html

4 PNG CONSTRUCTS

This section discusses specific features and risks of the PNG file format. Each construct provides an overview, a description of the risks and recommendations to mitigate the risk, products that are vulnerable to the risk, and the location of the construct in the file format.

4.1 PNG Signature

OVERVIEW

The PNG Signature constitutes the first eight bytes of every PNG file. The first byte is 0x89 in order to prevent confusion with a text file. The next three bytes are 0x50, 0x43, and 0x47, which evaluates to 'PNG' in ASCII. The next two bytes are 0x0d and 0x0a; this combination is referred to as a CR-LF sequence. The seventh byte is 0x1a or control-Z character. The final byte is 0x0a (a line feed).

RISKS AND RECOMMENDATIONS

Data Attack – Files that do not conform to the specification indicate that the file may have been modified or created with an invalid application.

1. Validate – The first eight bytes are equivalent to: 0x89, 0x50 ('P'), 0x43 ('N'), 0x47 ('G'), 0x0d, 0x0a, 0x1a, and 0x0a.
2. Reject – Files that do not conform to the specification requirements.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

The PNG file signature makes up the first eight bytes of every PNG file.

4.2 Chunks

The specification defines a specific chunk format and the minimum required chunks that are required for processing a PNG file. The chunk format is a feature that allows for expansion of the file format in the future.

4.2.1 Chunk Format

OVERVIEW

PNG Chunks should follow the PNG file signature and be formatted according to the specifications. Chunks consist of at most four parts: a four-byte length field, a four-byte chunk type field, a variable length chunk data field, and a four-byte cyclic redundancy code (CRC). Note that if the length field is zero, then there is no chunk data field.

RISKS AND RECOMMENDATIONS

Data Hiding – Unrecognized chunks may contain information that applications will not present to end users.

Data Attack – Malformed chunks may contain malicious code or trigger parsing problems for applications.

1. Validate – The chunk type is defined by the specification or in a whitelist.
2. Validate – The length, chunk type, and chunk data match the CRC.
3. Remove – Chunks with unrecognized types.
4. Remove – Non-critical Chunks with length zero.
5. Reject – Files that contain data unrecognized chunks.
6. Reject – Files that contain invalid CRCs.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

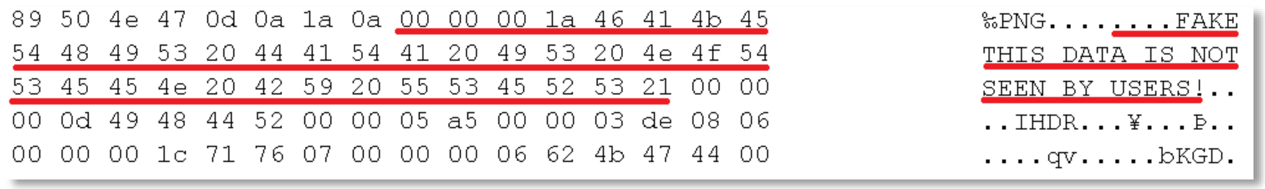
Chunks are found immediately after the PNG file signature.

4.2.2 Chunk Order

OVERVIEW

PNG Chunks should follow the PNG file signature and be formatted according to the specifications. Chunks can appear in any order with the exception of two chunks: IHDR and IEND. IHDR should be the first chunk and IEND should be the last chunk.

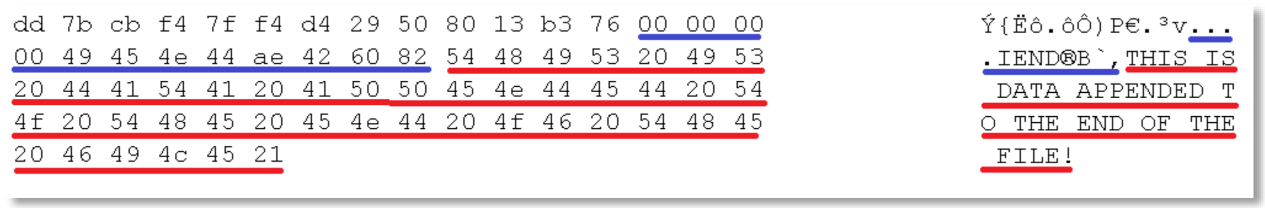
The format of a PNG file can be modified to include information before the IHDR. This information may not be processed by image processing applications. Figure 4-1, below, demonstrates a custom chunk, FAKE, that exists before the IHDR chunk and contains 26 bytes of hidden data.



```
89 50 4e 47 0d 0a 1a 0a 00 00 00 1a 46 41 4b 45        %PNG........FAKE
54 48 49 53 20 44 41 54 41 20 49 53 20 4e 4f 54        THIS DATA IS NOT
53 45 45 4e 20 42 59 20 55 53 45 52 53 21 00 00        SEEN BY USERS!..
00 0d 49 48 44 52 00 00 05 a5 00 00 03 de 08 06        ..IHDR...¥...Þ..
00 00 00 1c 71 76 07 00 00 00 06 62 4b 47 44 00        ....qv.....bKGD.
```

Figure 4-1 Invalid Chunk before IHDR

Furthermore, data can be appended to end of the IEND chunk; this does not alter how image processing applications render a PNG. Figure 4-2**Error! Reference source not found.**, below, shows the hexadecimal and ASCII representation of a file modified to include information after the IEND chunk. The IEND chunk is underlined in blue and the data that will not be processed by some image processing applications is underlined



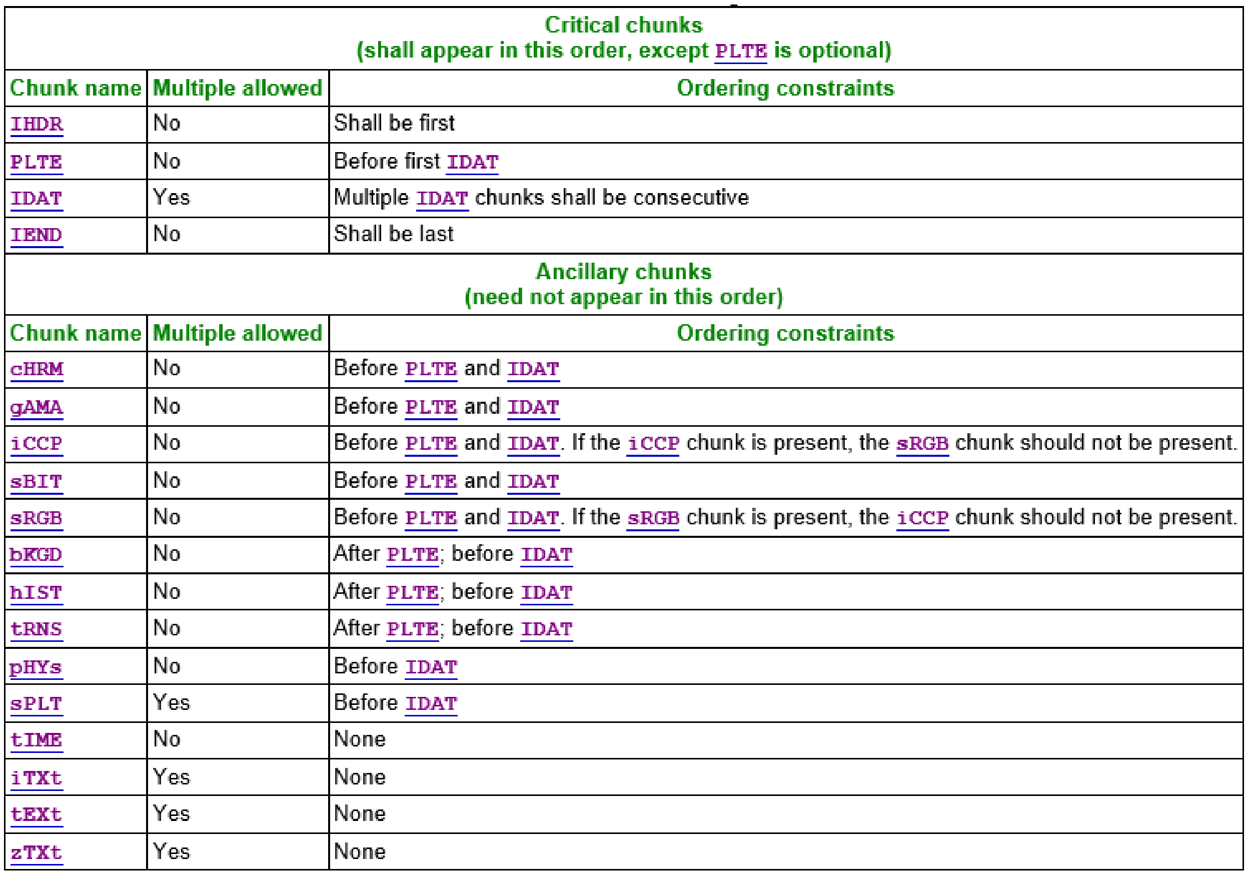
```
dd 7b cb f4 7f f4 d4 29 50 80 13 b3 76 00 00 00    Ý{Ëô.ôÔ)P€.³v...
00 49 45 4e 44 ae 42 60 82 54 48 49 53 20 49 53    .IEND®B`,THIS IS
20 44 41 54 41 20 41 50 50 45 4e 44 45 44 20 54     DATA APPENDED T
4f 20 54 48 45 20 45 4e 44 20 4f 46 20 54 48 45    O THE END OF THE
20 46 49 4c 45 21                                   FILE!
```

in red.

Figure 4-2 Invalid Data after IEND

Finally, all chunks have a designated order as required by the specification. Table 4-1, below, enumerates the ordering constraints on critical and ancillary chunks; it also describes the whether there should be more than one of a chunk. If there are multiple IDAT chunks, the order for uncompressing the data is derived from the order the chunks are encountered, i.e. the first chunk will be uncompressed first and the last chunk last.

Table 4-1 – Chunk Order Constraints [2]



Critical chunks (shall appear in this order, except PLTE is optional)		
Chunk name	**Multiple allowed**	**Ordering constraints**
IHDR	No	Shall be first
PLTE	No	Before first IDAT
IDAT	Yes	Multiple IDAT chunks shall be consecutive
IEND	No	Shall be last
Ancillary chunks (need not appear in this order)		
Chunk name	**Multiple allowed**	**Ordering constraints**
cHRM	No	Before PLTE and IDAT
gAMA	No	Before PLTE and IDAT
iCCP	No	Before PLTE and IDAT. If the iCCP chunk is present, the sRGB chunk should not be present.
sBIT	No	Before PLTE and IDAT
sRGB	No	Before PLTE and IDAT. If the sRGB chunk is present, the iCCP chunk should not be present.
bKGD	No	After PLTE; before IDAT
hIST	No	After PLTE; before IDAT
tRNS	No	After PLTE; before IDAT
pHYs	No	Before IDAT
sPLT	Yes	Before IDAT
tIME	No	None
iTXt	Yes	None
tEXt	Yes	None
zTXt	Yes	None

RISKS AND RECOMMENDATIONS

Data Hiding – PNG Chunks before the IHDR or after the IEND chunk may not be processed by PNG decoders and may not be presented to end users. Chunks that appear more often than allowed, in the wrong order, occur without the presence of a required chunk, or that should not occur because of the presence of another chunk may be ignored by some image processing applications and serve as a vector for hidden data.

Data Attack – PNG Chunks found out of order may contain malicious code or trigger parsing problems for applications. Because this data should be ignored by applications it may not contain an exploit, but may contain a payload because arbitrary data can be stored here.

1. Validate – There is no data before the IHDR chunk.
2. Validate – There is no data after the IEND chunk.
3. Validate – Chunks that should not appear multiple times do not.
4. Validate – Chunks are found in acceptable locations according to the specification.
5. Validate – sRGB and iCCP chunks are not both present.

6. Replace – The file with a version that contains chunks in the order described by the specification.
7. Remove – Non-chunk data after the PNG file signature and before the IHDR chunk.
8. Remove – Data after the IEND chunk.
9. Reject – Files that contain data in between the PNG file signature and IHDR chunk or after the IEND chunk.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

The IHDR chunk is found immediately after the PNG file signature, and the IEND chunk is the last chunk in a PNG file.

4.2.3 Critical Chunks

Critical chunks are required for processing a PNG data stream. The specification defines four such critical chunks: IHDR (Image Header), IEND (Image Trailer), IDAT (Image Data), and PLTE (Palette).

4.2.3.1 IHDR Image Header

OVERVIEW

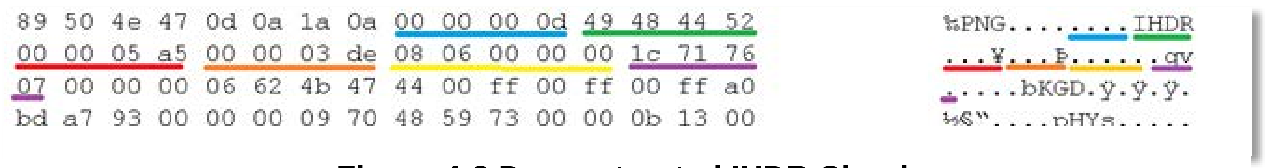
The IHDR chunk is a fixed length and fixed format chunk. The data field will always have a length of 0x0d (13). The type field value should be 'IHDR.' The data field is 13 bytes are organized as follows:

- Bytes 1-4 is the image width; this must be an unsigned integer and non-zero.
- Bytes 5-8 is the image height; this must be an unsigned integer and non-zero.
- Byte 9 is the bit depth; this must be a single byte integer that denotes the number of bits per sample or palette index. The values can only be 1, 2, 4, 8, or 16 and vary with the type of image (e.g., grayscale, palette based, etc.). Note that this is not the same as color depth, which indicates the number of bits per pixel.
- Byte 10 is the color type; this must be a single byte integer that denotes the PNG image type (e.g., true color, indexed color, etc.). The valid values are 0, 2, 3, 4, and 6.
- Byte 11 is the compression method; this must be a single byte integer that indicates the compression method. The only valid value is 0, which means that the inflate/deflate compression method is used.
- Byte 12 is the filter method; this must be a single-byte integer that indicates how to preprocess the data before compression. The only valid value is 0, which

means adaptive filtering will be used and any one of the five filtering methods in the specification will be used, e.g. Sub.

- Byte 13 is the interlace method; this must be a single byte integer that indicates whether or not the image data is interlaced. The valid values are 0 (not interlaced) or 1 (Adam7 interlaced).

Error! Reference source not found., below, deconstructs an IHDR chunk in a PNG file. The data on the left is a hexadecimal representation of a PNG file and the data on the right is an ASCII representation. The blue line indicates the chunk data length, which is 0x0d (13). The green line is the chunk field type, which evaluates to ‘IHDR’. The next thirteen bytes is the actual data for the IHDR chunk. The red line indicates the image width and the orange line indicates the image height. Each byte above the yellow line is a different field; the first byte is the bit depth, the second byte is the color type, the third byte is the compression method, the fourth byte is the filter method, and the fifth byte is the interlace method. The final, purple line is the CRC for the chunk. Note that it was

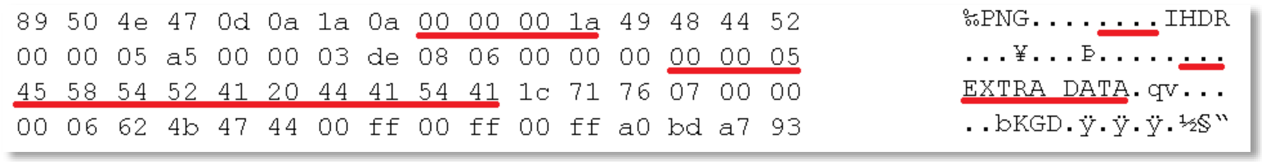


```
89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52     %PNG........IHDR
00 00 05 a5 00 00 03 de 08 06 00 00 00 1c 71 76     ...¥...Þ......qv
07 00 00 00 06 62 4b 47 44 00 ff 00 ff 00 ff a0     .....bKGD.ÿ.ÿ.ÿ.
bd a7 93 00 00 00 09 70 48 59 73 00 00 0b 13 00     ½§"....pHYs.....
```

Figure 4-3 Deconstructed IHDR Chunk

observed that the upper bits of these fields could be modified without affecting how a file was processed by an application. These fields could contain small amounts of data that would not be observable by an end user.

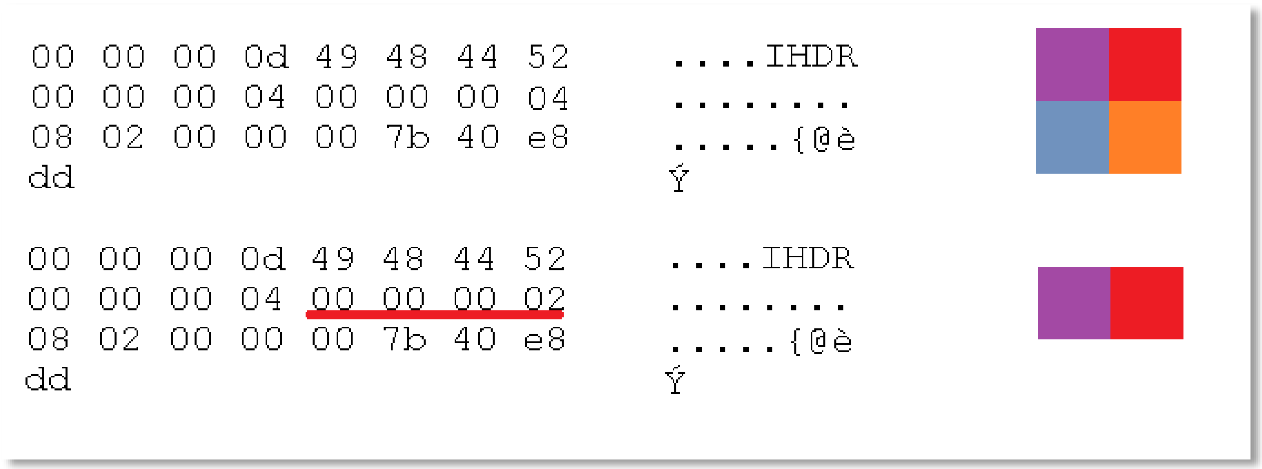
The IHDR chunk can be modified and still be processed by some image processing applications; for example, the length can be changed to include extraneous information (in addition to valid information) that will be disregarded. Figure 4-4, below, shows the length is doubled from 0x0d to 0x01a, which allows thirteen more bytes of data. The CRC wasn’t recalculated in this example; however, some image processing applications do not verify the chunk using the CRC If the CRC is not verified, then the file could have been modified before reaching its destination and could be corrupt or malicious.



```
89 50 4e 47 0d 0a 1a 0a 00 00 00 1a 49 48 44 52     %PNG........IHDR
00 00 05 a5 00 00 03 de 08 06 00 00 00 00 00 05     ...¥...Þ........
45 58 54 52 41 20 44 41 54 41 1c 71 76 07 00 00     EXTRA DATA.qv...
00 06 62 4b 47 44 00 ff 00 ff 00 ff a0 bd a7 93     ..bKGD.ÿ.ÿ.ÿ.½§"
```

Figure 4-4 Modified IHDR Length Field

Modifying the image height or width in the IHDR chunk may cause image data to not be rendered by image processing applications. Figure 4-5, below, depicts a PNG image and resultant image after the height field in the IHDR chunk is modified. The height is changed from four pixels to two pixels. This can allow for image data to exist in the file and not be rendered to an end user.



```
00 00 00 0d 49 48 44 52     ....IHDR
00 00 00 04 00 00 00 04     ........
08 02 00 00 00 7b 40 e8     .....{@è
dd                          Ý

00 00 00 0d 49 48 44 52     ....IHDR
00 00 00 04 00 00 00 02     ........
08 02 00 00 00 7b 40 e8     .....{@è
dd                          Ý
```

Figure 4-5 Modified IHDR Height Field

The valid bit depth settings with respect to color type can be found below. Table 4-2, below, shows allowed bit depths per color types.

Table 4-2 – Supported Bit Depths per Color Type

PNG Image Type	IHDR Color Type	Allowed Bit Depths
Greyscale	0	1, 2, 4, 8, 16
Truecolor	2	8, 16
Indexed Color	3	1, 2, 4, 8
Greyscale + Alpha	4	8, 16
Truecolor + Alpha	6	8, 16

RISKS AND RECOMMENDATIONS

Data Hiding – The IHDR chunk may be modified to include information that an image processing application may not present to end users.

Data Attack – If the IHDR chunk does not adhere to the specification it may contain malicious code or trigger parsing problems for applications[8].

1. Validate – The IHDR chunk contains a field type that evaluates to 'IHDR' in ASCII.

[8] https://www.snort.org/rule_docs/1-3133

2. Validate – The image width and height is a non-zero positive number and is equal to the width and height values found in the IHDR chunk.
3. Validate – The bit depth is 1, 2, 4, 8, or 16.
4. Validate – The color type is 0, 2, 3, 4, or 6.
5. Validate – The bit depth is supported by the color type.
6. Validate – The compression type is 0.
7. Validate – The filter method is 0.
8. Validate – The interlace method is 0 or 1.
9. Validate – The CRC of the IHDR chunk is valid.
10. Reject – Files that contain malformed or an invalid IHDR chunk.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

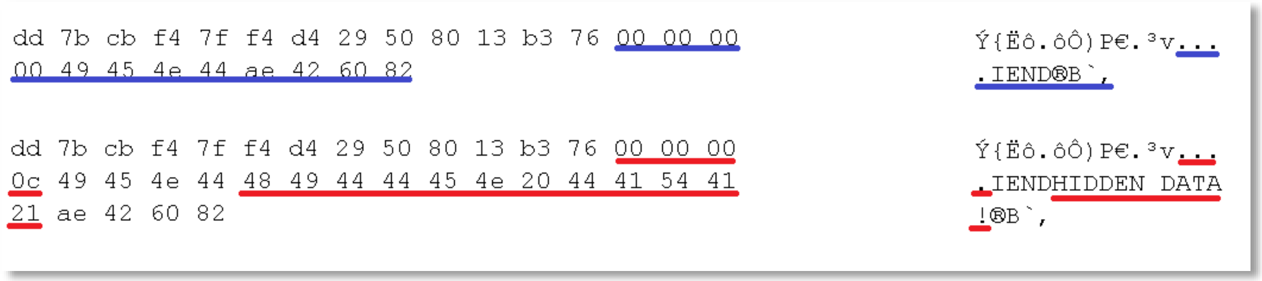
The IHDR chunk is found immediately after the PNG file signature.

4.2.3.2 IEND Image Trailer

OVERVIEW

The IEND chunk is fixed length and fixed format. The length will always be zero. Because the length is zero this chunk will have only three fields: the length field, the field type, and the CRC. The field type should evaluate to 'IEND'.

Some image processing applications may disregard the IEND chunk or allow for modifications to the chunk without affecting how an image is rendered.



```
dd 7b cb f4 7f f4 d4 29 50 80 13 b3 76 00 00 00      Ý{Ëô.ôÔ)P€.³v...
00 49 45 4e 44 ae 42 60 82                            .IEND®B`,

dd 7b cb f4 7f f4 d4 29 50 80 13 b3 76 00 00 00      Ý{Ëô.ôÔ)P€.³v...
0c 49 45 4e 44 48 49 44 44 45 4e 20 44 41 54 41      .IENDHIDDEN DATA
21 ae 42 60 82                                        !®B`,
```

Figure 4-6 Modified IEND Chunk

Figure 4-6, above, shows an unmodified IEND chunk (underlined in blue) and a modified IEND chunk beneath it with the length modified and data added (both underlined in red). The modified IEND chunk contains the text, "HIDDEN DATA!" and is processed by some image processing applications.

RISKS AND RECOMMENDATIONS

Data Hiding – The IEND chunk may be modified to include information that an image processing application may not present to end users.

Data Attack – If the IEND chunk does not adhere to the specification it may contain malicious code or trigger parsing problems for applications that attempts to process the data. Arbitrary data can be stored at the end of the file so a payload can be placed here.

1. Validate – The length field is zero and the data field is not present.
2. Validate – The field type evaluates to 'IEND'.
3. Validate – The CRC is valid.
4. Reject – Files that contain an invalid IEND chunk.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

The IEND chunk is the last chunk and last 12 bytes in a PNG file.

4.2.3.3 IDAT Image Data

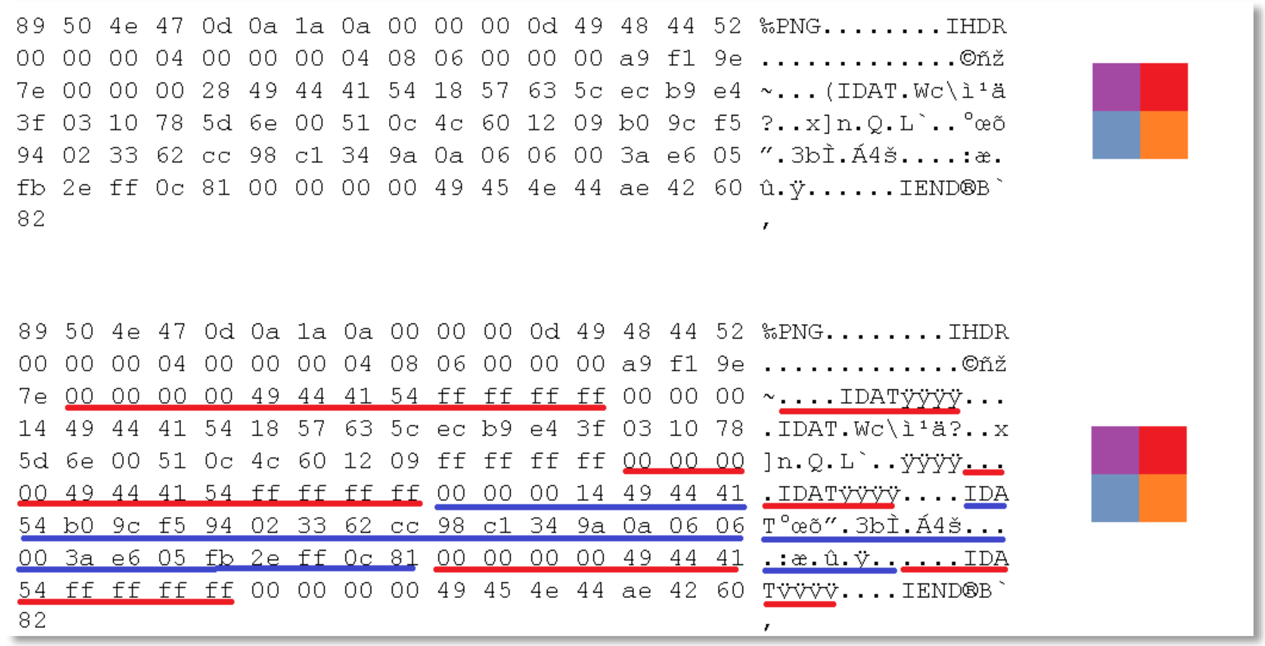
OVERVIEW

The IDAT chunk is a fixed format, variable length chunk. The amount of compressed image data stored in an IDAT chunk may vary on the image size.

Furthermore, image data may be spread out across one or more consecutive IDAT chunks. The compressed data in each IDAT chunk is concatenated to recover the original compressed image. Because the data can be split across chunks, it is possible for a malicious PNG to attempt to attempt an overflow; if each chunk in an IDAT series is set to the theoretical maximum length 2^31-1 (about 2 GB), then it may be possible for a PNG file to corrupt the execution stack.

It is possible to insert IDAT chunks with no data, although this is valid it does not serve any purpose. It may indicate the file is malicious[9]. Figure 4-7, below, shows the original hexadecimal and ASCII data of a PNG file on the top and underneath it is a modified version that renders the same. The modified version split the image data across two IDAT chunks (the new IDAT chunk with data is in blue) and the modified version contains multiple zero data IDAT chunks (underlined in red) that exist before actual image data, in the middle of image data, and at the end of image data.

[9] https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2014-0333



```
89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 %PNG........IHDR
00 00 00 04 00 00 00 04 08 06 00 00 00 a9 f1 9e .............©ňž
7e 00 00 00 28 49 44 41 54 18 57 63 5c ec b9 e4 ~...(IDAT.Wc\ì¹ä
3f 03 10 78 5d 6e 00 51 0c 4c 60 12 09 b0 9c f5 ?..x]n.Q.L`..°œõ
94 02 33 62 cc 98 c1 34 9a 0a 06 06 00 3a e6 05 ”.3bÌ.Á4š....:æ.
fb 2e ff 0c 81 00 00 00 00 49 45 4e 44 ae 42 60 û.ÿ......IEND®B`
82                                              ‚
```

```
89 50 4e 47 0d 0a 1a 0a 00 00 00 0d 49 48 44 52 %PNG........IHDR
00 00 00 04 00 00 00 04 08 06 00 00 00 a9 f1 9e .............©ňž
7e 00 00 00 00 49 44 41 54 ff ff ff ff 00 00 00 ~....IDATÿÿÿÿ...
14 49 44 41 54 18 57 63 5c ec b9 e4 3f 03 10 78 .IDAT.Wc\ì¹ä?..x
5d 6e 00 51 0c 4c 60 12 09 ff ff ff ff 00 00 00 ]n.Q.L`..ÿÿÿÿ...
00 49 44 41 54 ff ff ff ff 00 00 00 14 49 44 41 .IDATÿÿÿÿ....IDA
54 b0 9c f5 94 02 33 62 cc 98 c1 34 9a 0a 06 06 T°œõ”.3bÌ.Á4š...
00 3a e6 05 fb 2e ff 0c 81 00 00 00 00 49 44 41 .:æ.û.ÿ......IDA
54 ff ff ff ff 00 00 00 00 49 45 4e 44 ae 42 60 Tÿÿÿÿ....IEND®B`
82                                              ‚
```

Figure 4-7 Modified IDAT Data

Because IDAT chunks are compressed, they are susceptible to high levels of compression, which can cause problems when the data is decompressed [3]. This may cause a buffer overflow or out of memory condition in the application attempting to render or process the image.

RISKS AND RECOMMENDATIONS

Data Attack – If the IDAT chunk does not adhere to the specification it may contain malicious code or trigger parsing problems for applications. Large amounts of data that is highly compressed may be an attack vector against software not designed to handle it.

1. Validate – The length field is greater than zero
2. Validate – The CRC for each IDAT chunk is valid
3. Remove – Zero length IDAT chunks; this will not affect how the image renders
4. External Filtering Required – Compressed data should be decompressed by systems that can handle a large amount of data.
5. Reject – Files that contain an invalid IDAT chunk

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

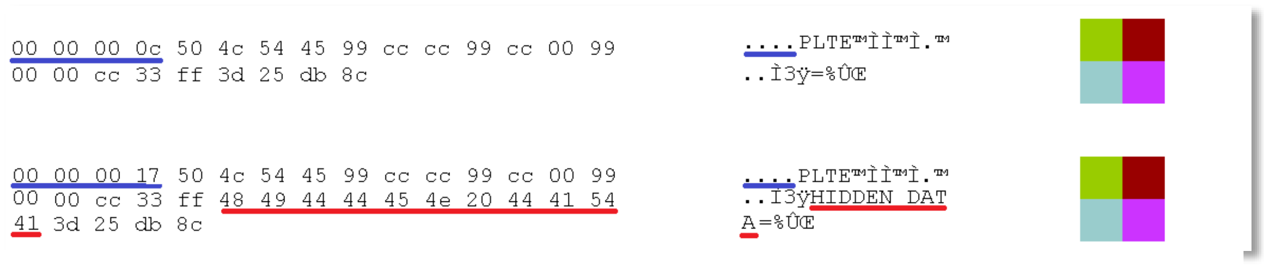
The IDAT chunk is found in between the IHDR and IEND chunks in the PNG file format.

4.2.3.4 **PLTE Palette**

OVERVIEW

The PLTE chunk is a fixed format, variable length chunk. The PLTE data field contains at least one, but no more than 256 3-byte entries; each entry contains a byte value for R, G, and B respectively. Because of this format the length should evaluate to a multiple of three.

This chunk is required when the IHDR field color type evaluates to three (index colored). This chunk is optional for values two (true color) and six (true color with alpha). When this chunk is included for an optional color type it includes the palette for quantization if it cannot be displayed correctly; if it is not included the system will attempt the quantization using a histogram that is either computed or supplied (using the hIST chunk). If a histogram is used, then the entries from the histogram replace the true color values. The PLTE chunk can be modified to include additional information that may not be presented to end users.



```
00 00 00 0c 50 4c 54 45 99 cc cc 99 cc 00 99     ....PLTE™ÌÌ™Ì.™
00 00 cc 33 ff 3d 25 db 8c                       ..Ì3ÿ=%ÛŒ


00 00 00 17 50 4c 54 45 99 cc cc 99 cc 00 99     ....PLTE™ÌÌ™Ì.™
00 00 cc 33 ff 48 49 44 44 45 4e 20 44 41 54     ..Ì3ÿHIDDEN DAT
41 3d 25 db 8c                                   A=%ÛŒ
```

Figure 4-8 Modified PLTE Chunk

Figure 4-8, above, depicts a PLTE chunk and a modified PLTE chunk that does not alter how the image renders. Underlined in blue are the length fields for the two PLTE chunks; the chunk on top has size 0x0c (12) and the modified version beneath it has size 0x17 (23). Note that the new chunk has a size that is not a multiple of three. The increase in the PLTE data size also enables a vector for the inclusion of extraneous data; the modified PLTE chunk contains the text ‘HIDDEN DATA’.

It is possible to alter how a PNG image is rendered by including a crafted PLTE chunk with a true color image; because some processing applications may choose to ignore the PLTE chunks readers may render two images with a PLTE Chunk differently[10]. Figure

[10] http://wiki.yobi.be/wiki/PNG_Merge

4-9, below, depicts a modified color palette; the color white in the palette was changed to black. Image rendering applications that do not process the PLTE chunk would reveal the hidden text; however, applications that process the PLTE chunk will not show this to the end user.

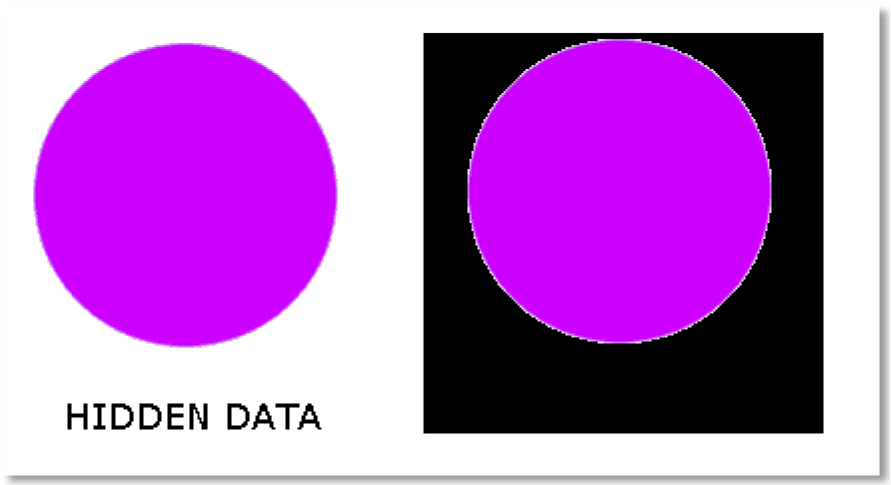


Figure 4-9 Remap Color Palette

RISKS AND RECOMMENDATIONS

Data Hiding – The PLTE chunk may be modified to include information that an image processing application may not present to end users.

Data Attack – If the PLTE chunk does not adhere to the specification it may contain malicious code or trigger parsing problems for applications.

1. Validate – There is only one PLTE chunk.
2. Validate – The length field is a multiple of three.
3. Validate – The IHDR color type is 2 (true color), 3 (index colored), or 6 (true color with alpha).
4. Validate – Each entry in the PLTE chunk is unique, i.e. no two entries in the palette have the same RGB value.
5. Validate – The CRC for the PLTE chunk.
6. Validate – Each entry in the PLTE chunk is used in the image.
7. Remove – PLTE chunks when the color type is 2 or 6.
8. Remove – Unused colors in the PLTE chunk.
9. Reject – Files that contain an invalid PLTE chunk.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

The PLTE chunk is found once in between the IHDR and IEND chunks in the PNG file format.

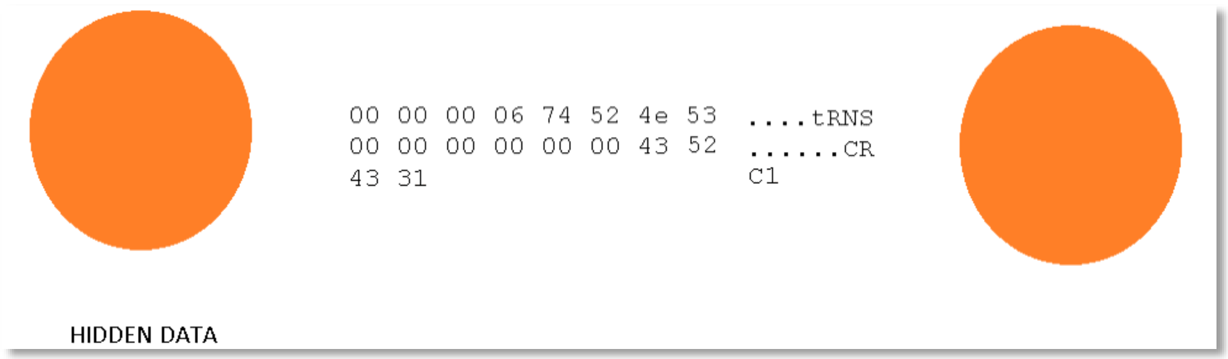
4.2.4 Ancillary Chunks

There specification defines 14 ancillary chunks that fit into five categories: transparency information, color space, textual information, miscellaneous information, and time information. The following constructs address each category; each construct will discuss the ancillary chunks included in that category.

4.2.4.1 Transparency Information

OVERVIEW

The transparency information category contains only one ancillary chunk: tRNS (transparency). The tRNS chunk contains: a single transparent color for color types 0 (greyscale) or 2 (true color) or alpha values associated with palette entries in the PLTE chunk. Multiple

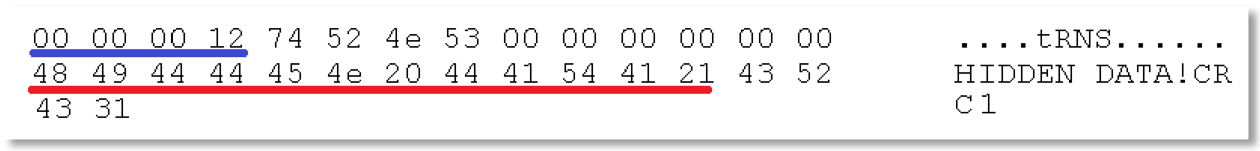


```
00 00 00 06 74 52 4e 53   ....tRNS
00 00 00 00 00 00 43 52   ......CR
43 31                     C1
```

Figure 4-10 tRNS Example

Figure 4-10, above, demonstrates the usage of the tRNS chunk. By adding this chunk to a true color image the color black can be set to transparent. Because ancillary chunks do not have to be processed some image processing applications may render the image on the left, while some may render the image on the right.

The single entry for color type 0 (grayscale) should be two bytes long, the RGB entry for color type 2 (true color) should be six bytes with two bytes per R, G, and B values, and the entries for color type 3 (indexed colors) should be one byte each. The tRNS length field should evaluate to two for color type zero, six for color type two, and at most 256 for color type three. The number of entries for color type three should be equal to or less than the number of entries in the PLTE chunk because each entry refers to the color that should be transparent in the PLTE. Extraneous information may be added to the tRNS chunks if these lengths are modified. Figure 4-11, below, illustrates how the tRNS chunk in Figure 4-10 can be modified to include additional data. The section underlined in blue is the new length, 0x12 (18), and the section underlined is extraneous data.



```
00 00 00 12 74 52 4e 53 00 00 00 00 00 00      ....tRNS......
48 49 44 44 45 4e 20 44 41 54 41 21 43 52      HIDDEN DATA!CR
43 31                                          C1
```

Figure 4-11 Modified tRNS Chunk

RISKS AND RECOMMENDATIONS

Data Hiding – The tRNS chunk may be modified to include information that an image processing application may not present to end users.

Data Attack – If the PLTE chunk does not adhere to the specification it may contain malicious code or trigger parsing problems for applications.

1. Validate – There is only one tRNS chunk.
2. Validate – The length field is a multiple of three if the entries are in the RGB format.
3. Validate – The IHDR color type is 2 (true color), 3 (index colored), or 6 (true color with alpha).
4. Validate – If the color type is three, that the number of entries in the tRNS chunk is less than or equal to the number of entries in the PLTE chunk.
5. Remove – The TRNS chunk from the file.
6. Reject – Files that contain an invalid tRNS chunks.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

Ancillary chunks dealing with transparency information are found after a PLTE chunk (if it exists) and before the IDAT chunk.

4.2.4.2 **Color Space Information**

OVERVIEW

This group includes the following chunks: cHRM (chromaticity), gAMA (gamma), iCCP (embedded ICC Profile), sBIT (significant bits), and sRGB (standard RGB color space). This construct does not cover what the chunks mean, but instead covers the constraints on the chunks.

The length field of the cHRM chunk should evaluate to 32; if the iCCP or sRGB chunks exist, then this chunk is ignored.

The length field of the gAMA chunk should be four; if the iCCP or sRGB chunks exist, then this chunk is ignored.

The iCCP chunk is variable in size; the data field contains 1-79 bytes for the profile name, 1 byte null separator, a 1 byte compression method (the only valid value is 0 for zlib), and n bytes of the compressed ICC profile.

The sBIT chunk varies with the color type; the length of the data should be 1 byte for color type 0 (grayscale), 3 bytes for color types 2 and 3 (true color and index colored), 2 bytes for color type 4 (greyscale with alpha), and 4 bytes for color type 6 (true color with alpha).

Note that both the iCCP and sRGB chunks cannot exist in the same file, PNG files must contain one or the other (or neither), but not both.

Because these chunks modify color space information, it may be possible to distort how an image renders. For example, overlaying a bright image over a dark image and adjusting the gamma values may present an end user with different images. Figure 4-12, below, shows how an application may render an image without gamma correction (on the right) and with gamma correction (on the left). Because gamma correction adjusts luminosity in an image, it is possible to use this value to present an end user with different images; however, both images should be slightly visible.

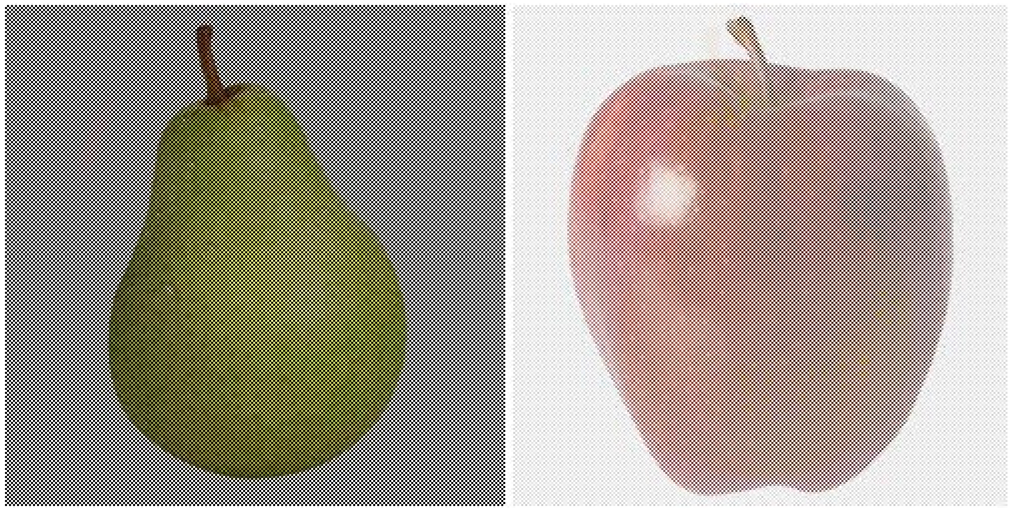


Figure 4-12 Gamma Tricks[11]

RISKS AND RECOMMENDATIONS

Data Hiding – Chunks that are ignored by processors when other chunks are present may be a vector for data not presented to end users. Color space distortions may be a means of obscuring information presented to end users.

[11] http://superuser.com/questions/579216/why-does-this-png-image-display-differently-in-chrome-firefox-than-in-safari-a

1. Validate – Ancillary Chunks with color space information adhere to the specification
2. Validate – That one or neither of the sRGB or iCCP chunks exist
3. Remove – The cHRM and gAMA chunks, if there is an sRGB or iCCP chunk.
4. Remove – Chunks that modify the color space; this will affect how an image renders.
5. Reject – Files that contain color space chunks.

Data Attack – Software designed to process PNG files may not be able to process ICC profiles; a malformed ICC profile may cause issues[12] for software attempting to parse it[13].

Data Disclosure – ICC Profile names might contain sensitive information.

6. Validate = The ICC Profile matches a whitelist of known, good ICC Profiles.
7. External Filtering Required – For the name of the ICC profile.
8. External Filtering Required – For the uncompressed ICC profile data.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

Ancillary chunks dealing with color space information are found before a PLTE chunk (if it exists) and before the IDAT chunk.

4.2.4.3 Textual Information

OVERVIEW

The textual information group contains the chunks: tEXt (textual data), iTXt (international textual data), and zTXt (compressed textual data). There is no restriction on the number of text chunks that may appear. Textual information chunks use keywords to identify the data in the text; these can either be from the specification or a custom string. Table 4-3, below, lists keywords defined in the specifications. Keywords are limited to printable Latin-1 characters and spaces.

Table 4-3 – Textual Information Keywords

Keyword	Description
Title	Short (one line) title or caption for image

[12] https://bugzilla.redhat.com/show_bug.cgi?id=1087444
[13] http://www.cvedetails.com/cve/CVE-2009-5063

Author	Name of image's creator
Description	Description of image (possibly long)
Copyright	Copyright notice
Creation Time	Time of original image creation
Software	Software used to create the image
Disclaimer	Legal disclaimer
Warning	Warning of nature of content
Source	Device used to create the image
Comment	Miscellaneous comment

The data of the tEXt chunk contains a: 1-79 byte keyword, 1 byte null separator, and an n bytes of text data, in that order. The text data is limited to the Latin-1 character set (including the linefeed character).

The data of the zTXt chunk contains a: 1-79 byte keyword, 1 byte null separator, 1 byte compression method field (the only valid value is 0 for zlib), and compressed text data that is n bytes long. The text data is limited to the Latin-1 character set (including the linefeed character).

The data of the iTXt chunk contains a: 1-79 byte keyword, 1 byte null separator, 1 byte compression flag (this can only be 0 for uncompressed and 1 for compressed), 1 byte compression method (the only valid is 0 for zlib), a language tag with data that adheres to RFC 3066[14], 1 byte null separator, n byte translated keyword, 1 byte null separator, and n byte uncompressed or compressed text. The text data uses UTF-8 encoding.

Both the zTXt and iTXt chunks can be compressed; Deflate compression is susceptible to high levels of compression that cause problems when the data is decompressed [http://libpng.sourceforge.net/decompression_bombs.html].

RISKS AND RECOMMENDATIONS

Data Hiding – Textual information chunks can include information that an image processing application may not present to end users.

Data Disclosure – Textual information chunks may contain sensitive data pertaining to an image.

Data Attack – Data compression attacks are possible by compressing large amounts of data into small chunks of compressed data[15].

[14] https://www.ietf.org/rfc/rfc3066.txt
[15] http://www.cvedetails.com/cve/CVE-2007-5269/

1. External Filtering Required – Text chunks should be externally filtered by software that can handle text processing.
2. Remove – Text chunks from the file format.
3. Replace – Replace the data of text chunks with arbitrary characters. This will not impact how the image renders, but it will change metadata.
4. Reject – Files that contain text chunks.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

Ancillary chunks dealing with textual information are found anywhere after the IHDR chunk.

4.2.4.4 Miscellaneous Information

OVERVIEW

The following ancillary chunks are categorized as miscellaneous information: bKGD (background color), hIST (image histogram), pHYs (physical pixel dimensions), and sPLT (suggested palette).

The bKGD chunk can set the image alpha value to any color; applications may render images with this chunk in different ways. Figure 4-13, below, shows how images with an alpha background and a bKGD chunk can be rendered differently.

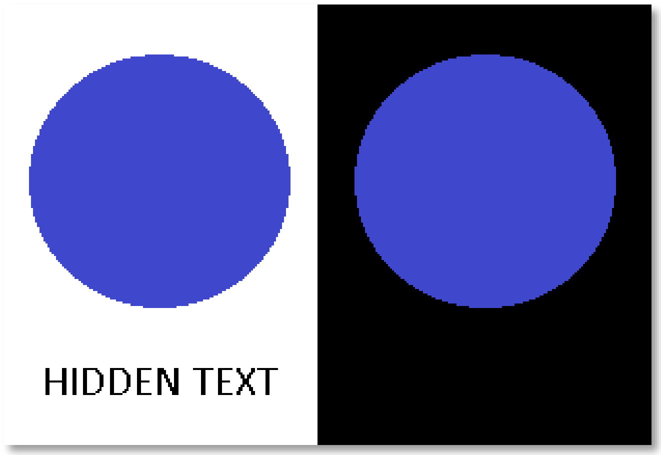


Figure 4-13 bKGD Chunk Example

pHYs specifies the intended pixel size or aspect ratio. It is 9 bytes long: x axis, y axis, unit specifier. When the chunk does not exist, the image pixels are assumed to be square, and the physical size of each pixel is unspecified.

The hIST chunk only appears when PLTE chunk appears. The size of the hIST chunk is proportional to size of palette; there is one two-byte entry per palette entry. The chunk when used by image processing applications can influence the colors used to render an image.

sPLT is the suggested palette chunk. It contains information for that enables to quantize an image if it cannot be displayed directly by the rendering machine, e.g., a machine that doesn't support truecolor; the viewing system handles this if the PLTE or sPLT chunk does not appear. The sPLT chunk consists of a: 1-79 byte palette name, 1 byte null separator, 1 byte sample depth, and n 6-byte or 10-byte entries. The entries are dependent on the sample depth; both entries contain red, blue, and green palette information. The 10 byte entries contain alpha and frequency. The entries for sPLT should occur in order of decreasing frequency. Because there can be more than one sPLT chunk, chunks with the same name can be a vehicle for data ignored by software applications.

RISKS AND RECOMMENDATIONS

Data Hiding – Miscellaneous information chunks modify how an image is presented and can contain data that does not have to be processed by image processing applications if they do not know how to process the information.

Data Attack – Malformed chunks may pose a data attack risk for parsers that process PNG files[16].

1. Validate – Miscellaneous chunks are formatted according to their specifications.
2. Validate – The number of entries in the hIST chunk is double the number of entries in the PLTE chunk.
3. Validate – Multiple sPLT chunks do not have the same name.
4. Validate – Entries in the sPLT are unique, i.e. there are no two entries with the same RGB or RGBA value.
5. Validate – The frequency is descending in the sPLT chunk.
6. Validate – The CRC for every chunk.
7. Remove – Miscellaneous chunks from the file; this may affect how some applications render the image.
8. Reject – Files that contain miscellaneous information chunks.

PRODUCT

PNG (Portable Network Graphics) Specification, Version 1.2

LOCATION

[16] http://www.cvedetails.com/cve/CVE-2013-7354/

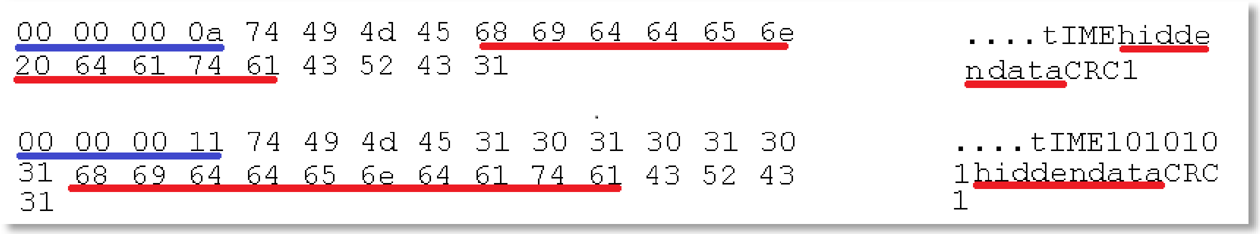
Miscellaneous chunks are found after the PLTE chunk if it exists and before the IDAT chunk.

4.2.4.5 Time Information

OVERVIEW

The time chunk gives the time of the last image modification. It is seven bytes long and is intended to be automatically updated when an image is modified.

This field can also be modified to include non-time information or information that includes time and non-time information. Figure 4-14, below shows how the time chunk can be modified to include arbitrary information (top) and an arbitrary amount of information (bottom). Notice the length fields (underlined in blue) are not seven even though the specification requires it to be.



```
00 00 00 0a 74 49 4d 45 68 69 64 64 65 6e        ....tIMEhidde
20 64 61 74 61 43 52 43 31                        ndataCRC1

00 00 00 11 74 49 4d 45 31 30 31 30 31 30        ....tIME101010
31 68 69 64 64 65 6e 64 61 74 61 43 52 43        1hiddendataCRC
31                                                1
```

Figure 4-14 tIME Chunk Modifications

RISKS AND RECOMMENDATIONS

Data Disclosure – Timing information chunks can contain information about the creation of an image, which may be sensitive in nature.

Data Hiding – Timing information chunks can be modified to include information that was not intended and may not be presented to end users.

Data Attack – Timing information designed to be parsed in a standard format, with a standard length can be a vector for a buffer overflow, when attempting to read in the values, or integer overflow, when storing the values.

1. Validate – The tIME chunk is seven bytes.
2. Validate – The CRC for the tIME chunk is valid.
3. Validate – The data is in a recognized time format; should be UTC, but it can be local time.
4. Replace – The tIME chunk with a known good timestamp, e.g. when the file was filtered/processed; this would also prevent the time from being recovered.
5. Remove – The tIME chunk can be removed without impact to the image; this would prevent the time from being recovered.

PRODUCT

LOCATION

Time information chunks can be found anywhere after the IHDR chunk.

5 SUMMARY OF RISKS

Table 5-1 – Summary of Risks

ISG Section	Specification	Hiding	Attack	Disclosure
4.1 PNG Signature	PNG (Portable Network Graphics) Specification, Version 1.2		x	
4.2.1 Chunk Format	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.2 Chunk Order	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.3.1 IHDR Image Header	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.3.2 IEND Image Trailer	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.3.3 IDAT Image Data	PNG (Portable Network Graphics) Specification, Version 1.2		x	

4.2.3.4 PLTE Palette	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.4.1 Transparency Information	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.3.2 Color Space Information	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	x
4.2.4.3 Textual Information	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	x
4.2.4.4 Miscellaneous Information	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	
4.2.3.5 Time Information	PNG (Portable Network Graphics) Specification, Version 1.2	x	x	x

6 ACRONYMS

Table 6-1 – Acronyms

Acronym	Denotation
APNG	Animated Portable Network Graphics
ASCII	American Standard Code for Information Interchange
CRC	Cyclic Redundancy Check
DTG	Data Transfer Guidance
GIF	Graphics Interchange Format
ISG	Inspection and Sanitization Guidance
JNG	JPEG Network Graphics
MNG	Multi-Image Network Graphics
PNG	Portable Network Graphics

7 REFERENCES

[1] "libpng," [Online]. Available: http://www.libpng.org/pub/png/libpng.html.

[2] "Portable Network Graphics (PNG) Specification (Second Edition)," W3C, 10 November 2003. [Online]. Available: http://www.w3.org/TR/PNG.

[3] G. Randers-Pehrson, "Defending Libpng Applications Against Decompression Bombs," 9 March 2010. [Online]. Available: http://libpng.sourceforge.net/decompression_bombs.html.

[4] G. Roelofs, "History of the Portable Network Graphics (PNG) Format," January 1997. [Online]. Available: http://www.libpng.org/pub/png/pnghist.html.

[5] M. Flickinger, "What's In A GIF - Bit by Byte," 24 January 2005. [Online]. Available: http://www.matthewflickinger.com/lab/whatsinagif/.

[6] N. McFeters, R. Carter and J. Heasman, "Extreme Client-Side Exploitation," [Online]. Available: https://www.blackhat.com/presentations/bh-jp-08/bh-jp-08-McFeters/BH_US_08_Mcfeters_Carter_Heasman_Extreme_Client-Side_Exploitation.pdf.