Inspection and Sanitization Guidance for Bitmap File Format

Version 1.0

1 March 2012

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 CHANGE HISTORY

Date	Version	Description
3/01/2012	1.0	Initial Release
12/13/2017	1.0	Updated Contact information, IAC Logo, Cited Trademarks and Copyrights, Expanded Acronyms, and added Legal Disclaimer

DISCLAIMER OF WARRANTIES AND ENDORSEMENT

The information and opinions contained in this document are provided "as is" and without any warranties or guarantees. References herein to any specific commercial product, process, or service by trade name, trademark, manufacturer or otherwise, does not constitute or imply its endorsement, recommendation or favoring by the United States Government and this guidance shall not be used for advertising or product endorsement purposes.

EXECUTIVE SUMMARY

This *Inspection and Sanitization Guidance (ISG) for Bitmap* document provides guidelines and specifications for developing file inspection and sanitization software for Bitmap files.

The Bitmap file format is used to represent a pixel array as an image. It is a Raster graphics format; the bytes within the pixel array correspond to pixels in an image. The Bitmap file format, also referred to as the Device Independent Bitmap (DIB) file format, stores its information independently of the display device. This allows Bitmaps to be moved from one device to another as opposed to a Device Dependent Bitmap (DDB) which requires the specification of the device context in order to process the data for proper output to a target device. Device Independent Bitmap (DIB) contains a color table, which provides a translation of pixel values to Red-Green-Blue (RGB) color values, which allows the image to be used on any device. Device Dependent Bitmap (DDB) provides the dimensions of the image in pixels, and an array that translates from the device palette to pixels [1], which is in the device's color format, which makes it less portable. The Bitmap file format has grown to incorporate data compression, alpha channels, and color profiles in addition to containing a pixel array. This ISG focuses on Device Independent Bitmaps as defined by Microsoft®[1] Developer Network (MSDN) [1].

Using the information provided in this report, reviewers can identify and analyze potential locations where hidden or malicious data may reside in BMP files.

[1] Microsoft is a registered trademark of Microsoft Corp.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 SCOPE

1.1 Purpose of this Document

The purpose of this document is to provide guidance for the development of a sanitization and analysis software tool for the Bitmap (BMP) images as defined in [1]. It provides inspection and analysis on various elements that are contained within the BMP file structure and how they can be a cause for concern for either hiding sensitive data or attempts to exploit a system. This document provides an analysis of features in BMP and recommendations to mitigate these threats to provide a safer file. Although this report does not mention vulnerabilities related to a specific image editor, many were used in the analysis of the BMP file format. Numerous Common Vulnerabilites and Exposures (CVE)s have registered for BMP related vulnerabilities in applications.

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

1.2 Introduction

The Bitmap (BMP) file format is supported by numerous applications that allow Bitmap images to be viewed and/or modified. Risks stem from the allowance of data to be hidden from the user of an image viewing application. Modifications made to fields hidden to such a user may create a buffer overflow vulnerability in image viewing applications.

The organization of the BMP file is well documented by the Microsoft Developer Network making its use and support widespread. Modifications can be to BMP images made without imperceptible changes in the stored image.

Bitmap graphics are sometimes known as raster graphics, which is an image type that describes each pixel or picture element in the data of the image. BMP files are different from other image types such as vector graphics images. In fact, Bitmap images are complementary to vector graphics.Vector graphics implement mathematical commands or operations that depict how each shape, line, curve, or polygon is drawn to create the image.

1.3 Background

The Bitmap file format has been encompassed by the Device Independent Bitmap (DIB) file format since Windows 3.0. BMP no longer refers to the Device Dependent Bitmap (DDB) file format which required the specification of a device context in order to process the data for proper output to a target device. DDB files are sometimes referred to as compatible bitmaps because they have the same color organization of the displaying device, as each file is created for that device, which implies that they are faster to process. DDB files contain information about the image’s height, width, color depth, and the actual pixel information. DIB is the second class of bitmaps which was designed to provide portability; the ability to move Bitmap images from one system to another and achieve the proper color scheme on any device [1]. DIBs were introduced to add color information, resolution for the image, the palette for the device, an array of bits mapping Red, Green, Blue (RGB) triplets to pixles in the image, as well as information about data compression if it was used [1].

BMP files have carved a niche as favicons, logos, icons, and is the default image format of cut-n-paste and screenshot operations on MS Windows®[2] platforms. The file format is simple and is widely implemented across many platforms. There are numerous applications with different implementations that can edit BMP images making their creation and modification simple at the cost of a much large file size compared to other common image formats. Compression is only supported for lower color depths due to the available compression algorithms defined in the specification.The inability to utilize compression methods on images requiring a higher color depth makes the Bitmap format inefficient for images that contain a large amount of pixel data.

1.4 Document Organization

This document describes the objects and syntax of a Bitmap images. It refers to these elements as constructs. In addition to describing each relevant object, this document also describes its potential flaws or ways data can be hidden, disclosed, or embedded maliciously. Each construct description begins with the label ‘*BMP: x.y*’ and ends with ‘*BMP: x.y: END*’, where ***x*** represents the section number of this document and ***y*** is the sequential index for each section. They are provided as a reference, and each section of information is written to allow a user to reference a particular feature to analyze concerns and recommendations for that feature.

The following table summarizes the organization of this document.

Table 1-1. Document Organization

Section	Description
Section 1: Scope	This section describes the scope, organization, and limitations of this document.
Section 2: Construct and Taxonomy	This section describes a definition of how the

[2] Microsoft Window is a registered trademark of Microsoft Corp.

Section	Description
	constructs are represented as well as the terms defined in this document.
Section 3: BMP File Overview	This section describes the general overview and description of the BMP file format.
Section 4: BMP File – Constructs	This section describes numerous objects and dictionary information defined in the BMP file that provide rich media content, interactive features, and embedded content.

1.5 Recommendations

The following subsections summarize the categories of recommendation actions that appear in this document, and associated options.

1.5.1 Actions

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

The Recommendation section in each construct lists each of these actions and corresponding applicable explanations of the action to take. It notes if a particular action does not apply, indicates actions that are not part of the standard set of actions (listed in the previous paragraph). For example, a program may choose to reject a file if it is encrypted. Additionally, for some constructs, an action may further break down to specific elements of a construct (e.g., for hidden data in a dictionary's syntax) to give administrators the flexibility to handle specific elements differently.

Recommendations such as Remove and Replace alter the contents of the document. BMP images are structured such that each object can be identified by a byte offset into the file.

NOTE

The recommendations in this document are brief explanations rather than a How-To Guide. Readers should refer to the Microsoft Developer Network Bitmap specification for additional details.

Table 1-2 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 file.(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 user name 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 areas.
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 BMP file.

NOTE

No recommendations for logging all actions and found data are included here because all activity logging in a file 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.)

1.5.2 Action Options

The companion to this document, *Data Transfer Guidance for BMP Documents*, specifies four options for each recommended action: *Mandatory*, *Recommended*, *Optional*, or *Ignore*. Depending on the circumstances (e.g., a low to high data transfer versus a classified to unclassified transfer), programs can be configured to handle constructs differently.

Table 1-3 summarizes the recommendation action options.

Table 1-3. Recommendation Action Options

Action Options	Comments
Mandatory	For the given direction (e.g., secure private network to unsecure Internet), the file inspection and sanitization program must perform this recommended action.
Recommended	Programs should implement this action if technically feasible.
Optional	Programs may choose to perform or ignore this recommended action.
Ignore	Programs can ignore this construct or data structure entirely

1.5.3 Naming Convention for Recommendations

Recommendations in this document are numbered sequentially, where applicable, and adhere to a standard naming convention identified by a single number *x*, where *x* is a sequential number following by the recommendation keyword defined in Table 1-2. There may be multiple recommendations of the same type, which remain uniquely identified by its number. There is only one file content under review in this document (BMP).

1.6 Data Transfer Guidance

Each format that is documented for inspection and sanitization analysis has a companion document (i.e., the aforementionedData Transfer Guidance (DTG) document). The DTG serves as a checklist for administrators and others to describe expected behaviors for inspection and sanitization programs. For example, administrators may remove certain values in a metadata dictionary. Or, the administrator may decide to remove all hidden data if the document is being transferred to a lower security domain.

The *DTG* gives the administrator the flexibility to specify behaviors for inspection and sanitization programs. The workbook contains a worksheet for each security domain (i.e., the originating domain). Each worksheet lists the numbered constructs from this document and enumerated recommendations in a row. After the recommendations, the worksheet displays a cell for each possible destination domain. This enables an administrator to select the action option for data transfer from the originating domain to the particular destination domain. Each construct row also contains two comment cells: one for low to high transfers and another for high to low transfers.

The recommended actions address two broad risk types: data hiding and data execution. Some data structures are vulnerable to one risk type, while others are susceptible to both risk types. Each construct row in the DTG worksheet contains a cell for designating the risk type (i.e., data execution, data hiding, or both) and another cell for assessing the risk level for that construct (i.e., high, medium and low). This enables administrators to assign the risk type and risk level to each specific construct.

1.7 Document Limitations

This document covers information from the Microsoft Developer Network Bitmap Specifications [1].

1.7.1 Covert Channel Analysis

This document addresses data that can be hidden from normal human review but can be read by a human with a hex viewer or by a machine; this type of method will be called a **Type 1** covert channel. This document will also describe some of the methods for hiding data in a way that is not easily read by a human with a hex viewer or by a machine; methods in this class will be called **Type 2** covert channels. Note that a Type 1 channel is not strictly inferior to a Type 2 channel; for instance, a Type 2 channel using the least significant bits in the image codestream will require special processing to detect the hidden data, but it may introduce visual distortion that is noticeable by normal human review.

This document will provide guidance for the removal or mitigation of Type 1 covert channels in BMP files. It will describe the Type 2 covert channels that are most likely to appear in BMP files and will provide guidance for the mitigation of these channels.

1.7.2 Character Encoding

The BMP specification indicates a BMP file is a sequence of bytes. Grouping certain bytes together form the fields and structures identified in the specifications.

All data values are stored in *little-endian* order[3]. That is, the *least significant* byte occurs first.

Units implemented by the MSDN specifications of the Bitmap File Format that are mentioned within this document are:

- **Byte:** A byte consists of 8 bits of data.

[3] Compared to *Big-Endian* order where the *most significant* byte is stored first.

- **WORD:** A WORD is 2 bytes, a 16-bit unsigned integer.
- **DWORD:** A DWORD consists of 4 bytes of data, a 32-bit unsigned integer.
- **LONG**: A LONG consists of 4 bytes of data, a 32-bit signed integer.
- **Nibble:** 4 bits, half of a byte.
- **CIEXYZTRIPLE:** This structure consists of 36 bytes of data. This is used for the endpoints in the Bitmap Version Four and Version Five Image Headers.

2 CONSTRUCTS AND TAXONOMY

2.1 Constructs

Although this document delves into many low level constructs in the BMP syntax hierarchy, it does not serve as a complete reference to all of the different constructs in the standard. The document covers the overall file format, various graphics and interactive objects, and metadata elements. We have identified these as particular areas of concern for developers of file inspection and sanitization programs; however there is complete detail in the standard that should be examined alongside this documentation.

- **Description**: a high level explanation of the data structure or element
- **Concern**: an explanation of potential problems posed by the element. For example, some metadata elements can cause inadvertent data leakage and others can be used for data exfiltration.
- **Location**: provides a textual description of where to find the element in the document. This can vary by client (or product) or it may apply across the entire product line.
- **Examples**: if applicable, the definition will contain an example of the construct.
- **Recommendations**: as described in Section 2.2.

Recommendations appear within each of the BMP constructs. For the purposes of this document, these recommendations are “alternatives.” Some recommendations may seem better than others and some recommendations may be more difficult to implement. Certain recommendations complement each other and can be grouped together (e.g., “Remove action object” and “Remove reference to action object”). Other recommendations may seem contradictory (e.g., “Remove Metadata” and “Replace Metadata”).

2.2 Taxonomy

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

Table 2-1. Document Taxonomy

Term	Definition

Consistency	A construct state in which object information is set to correct values, and that required objects are implemented as defined in the BMP MSDN specifications.
Construct	An object in BMP terminology that represents some form of information or data in the hierarchy of the BMP image structure.
Device Context	A Microsoft Windows application programming interface and core operating system component responsible for representing graphical objects and transmitting them to output devices such as monitors and printers.
DTG	A list of all ISG constructs and their associated recommendations. DTGs are used to define policies for handling every ISG construct when performing inspection and sanitization.
Fully Qualified Name	An unambiguous name that specifies refers to an entire path.
Grammar	A precise, formal, and rigorous syntax for defining constructs.
Inspection and Sanitization	Activities for processing files 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 that file format.
Recommendations	A series of actions for handling a construct when performing inspection and sanitization activities.
Referential Integrity	The construct state in which all associated objects are properly referenced in the construct and that construct entries reference existing objects. References may also point to byte offset location into the BMP file as opposed to object number. If this isn't the case the document may not open or may break.

Table 2-2. BMP File Extensions

Extension	Description
bmp, dib	Signifies a BMP image file. This format is specified in the MSDN specifications. It is meant to represent a single image.

3 BMP OVERVIEW

This chapter describes the format of a BMP image as detailed in the MSDN specifications. It provides an overview of both the basic and optional structures that are defined by the specification and the organization of a BMP file.

3.1 Introduction to BMP

BMP files are composed of data structures. The specificaiton defines the BMP File Header structure, which defines the file type ('BM'), the size, in bytes, of the bitmap file, and defines the offset, in bytes, from the beginning of the file header to the bitmap bits [1]. Following the file header, is the DIB Header, or an image header that has several different versions. The DIB Header defines basic aspects of the image such as the size, height, width, bits per pixel, and other fields that are documented in this report. There may also be additional bit masks and information about compression of the image. Compression is supported for 4 bit-per-pixel and 8 bit-per-pixel images, but most BMP images are not compressed [5]. There are also optional structures that can be added provided the file meets certain requirements. For example, if the image has a color depth of eight bits per pixel or less then it must contain a color table whereas for higher color depths this structure is optional. These structures are made up of smaller fields which conform to a certain size and order. There exist fields that will be ignored under certain circumstances, but must stay in the file to retain the structure. All BMP files contain a pixel array following the Image Header (and color table if it exists). The last optional data structure is an ICC color profile, which is used to provide "color management systems with the information necessary to convert color data between native device color spaces and device independent color spaces" [2]. Figure 3-1 illustrates the traditional BMP file structure with the minimum required components, while Figure 3-2 shows a BMP file with optional information.

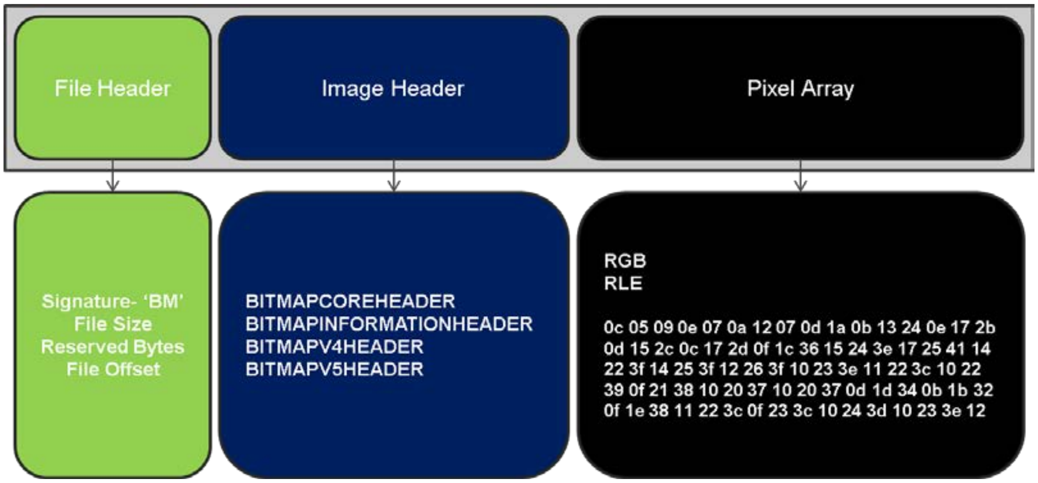

File Header
Image Header
Pixel Array
Signature- 'BM'
File Size
Reserved Bytes
File Offset
BITMAPCOREHEADER
BITMAPINFORMATIONHEADER
BITMAPV4HEADER
BITMAPV5HEADER
RGB
RLE
0c 05 09 0e 07 0a 12 07 0d 1a 0b 13 24 0e 17 2b
0d 15 2c 0c 17 2d 0f 1c 36 15 24 3e 17 25 41 14
22 3f 14 25 3f 12 26 3f 10 23 3e 11 22 3c 10 22
39 0f 21 38 10 20 37 10 20 37 0d 1d 34 0b 1b 32
0f 1e 38 11 22 3c 0f 23 3c 10 24 3d 10 23 3e 12

Figure 3-1. Traditional BMP File Format

The fields within the structures of the BMP file report information about specifications of the file and the image stored in the pixel array. Some image viewers trust that the fields and values defining the locations of data in the file are correct, regardless of their accuracy. This aspect of BMP poses both data hiding and attack risks.

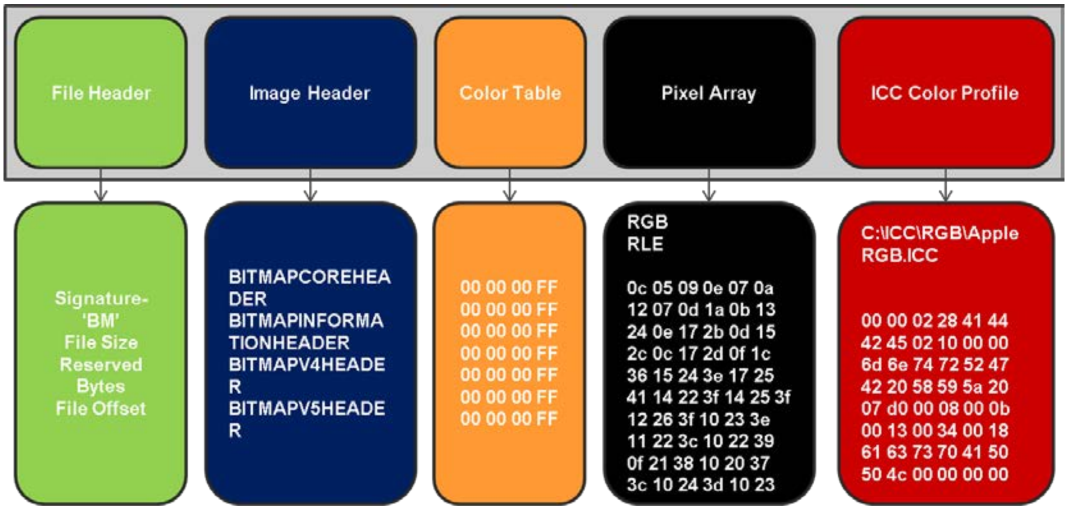


Figure 3-2. BMP File with Optional Structures

The BMP specification provides information on each field that should be set and all the valid values that may exist for that field. However, it is possible to modify these fields and values and craft bitmap files that do not adhere to the specification yet may still be displayed correctly in many viewers. There are certain bytes following the Color information for the image that can be manipulated within the file that may not affect how the image is perceived. An example is shown in Figure 4-3. Certain fields aren't used or checked by viewing applications which can allow malformed Bitmaps appear to function normally.

The Image Header structure has multiple versions; each version may not be backwards compatible with all versions of the Windows Operating Systems. In Windows 3.0 ®[4] , the Core Header Structure was introduced and provides information about the bitmap size, the width, the height, the number of color planes, and bits per pixel. Also, in Windows 3.0, the Info Header Structure was introduced which defines everything the Core Header implements and provides additional information such as the compression type, the resolution of the image, and information about the color table. In Windows 95®[5], the Bitmap Version 4 Header was introduced, which added information for

[4] Microsoft Windows 3.0 is a registered trademark of Microsoft Corp.

[5] Microsoft Windows 95 is a registered trademark of Microsoft Corp.

masking, color space types, and gamma information. In Windows 98®[6], the Bitmap Version 5 Header was introduced, which added a rendering intent field and color profile information.

3.2 Compression Modes

BMP supports the use of run length encoding to compress the data stored in the pixel array. Run length encoding can be used with only the color depths of four bits per pixel Run Length Encolding (RLE4) and eight bits per pixel (RLE8). These two compression modes use encoded and absolute modes when compressing data. Either one of these two modes can occur throughout a single bitmap. However, instructions must begin on a word boundary, which makes it necessary to have padding for runs that do not conform to this rule. There are some tools available that still produce compressed BMP files, typically RLE8 has been only observed in tools that produce this type of compressed file. Compression of BMP images is supported only for 4-bit-per-pixel and 8-bit-per-pixel images due to the only form of compression is RLE4 and RLE8. Other bit-per-pixel BMPs are never compressed. Since RLE is a simple form of compression, for images of higher color depth, which require more than a single byte for pixel representation, the RLE4 and RLE8 compression algorithm may not be as effective when applied to color depths where the pixels are larger than a single byte. More details on compression and the RLE implementation, including the differences between encoded and absolute mode, is addressed in construct BMP 4.10.

[6] Microsoft Windows 98 is a registered trademark of Microsoft Corp.

4 BMP CONSTRUCTS

This section addresses the aspects of the BMP file format that pose risks of data hiding, data disclosure, or data attack.

4.1 File Header

This subsection addresses issues that exist within specific fields within the file header segments. The file header refers to the composite of the signature, file size, reserved bytes, and file offset.

BMP.4.1: BMP FILE HEADER

DESCRIPTION:
The BMP file header is the first fourteen bytes of the BMP file and is composed of four different components.

- The first two bytes, also known as the signature, identifies the file as a Bitmap. This portion will always evaluate to 'BM' in ASCII or 0x424d. If other values are used it may be Operating System/2 (OS/2) specific or it is not a valid BMP file.

- The next four bytes represent the size of the entire file.

- The following four bytes is composed of two reserved fields, each two bytes, which should be set to 0. It is possible that an application coud place data here, but it has not been observed. It should be set to zero.

- The final four bytes of the file header is the offset to the pixel array. This value is typically equal to the sum of the lengths of the File Header, Image Header, and the Color Table (if included). If this value is greater than the actual length or sum of all the valid headers and data in the file, then the image file may contain unnecessary data.

CONCERNS:
Data Attack - Data attack risks may exist if the file size has been altered to a value significantly lower than the actual size which may cause a buffer overflow vulnerability. Data attack risks may exist if the file offset has been altered to a large value which can cause an integer overflow vulnerability. A data attack was demonstrated in CVE-2006-0006 where a size was set to 0, but data remained in the file [11].

Data Hiding - Data hiding risks may exist when the file size or reserved bytes have been altered. There are no noticeable effects in the image if these fields are modified. Data hiding can exist by altering the offset to the pixel array. Data may be stored in the region between the BMP image header or Color Table if it is used and the BMP pixel array.

PRODUCT: BMP

LOCATIONS:

The BMP file header is composed of the first fourteen bytes of a Bitmap file.

EXAMPLE 1 (VALID EXAMPLE)

```
BM6.<.....6...    42 4d 36 00 3c 00 00 00 00 00 36 00 00 00
```

EXAMPLE 2 (INVALID EXAMPLE)

```
BMMODIFIED6...    42 4D 4D 4F 44 49 46 49 45 44 36 00 00 00
```

EXAMPLE 3 (BMP FILE OFFSET)

```
BM6.<.....6...(.  42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
................  00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
....<...........  00 00 00 00 3C 00 00 00 00 00 00 00 00 00 00 00
.......           00 00 00 00 00 00
```

EXAMPLE 4 (BMP FILE OFFSET MODIFIED)

```
BM6.<.....@...(.  42 4d 36 00 3c 00 00 00 00 00 40 00 00 00 28 00
................  00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
....<...........  00 00 00 00 3C 00 00 00 00 00 00 00 00 00 00 00
......OFFSETTEXT  00 00 00 00 00 00 4F 46 46 53 45 54 54 45 58 54
```

EXAMPLE 5 (STRUCT DEFINITION [MSDN])

```
typedef struct tagBITMAPFILEHEADER {
  WORD  bfType;
  DWORD bfSize;
  WORD  bfReserved1;
  WORD  bfReserved2;
  DWORD bfOffBits;
} BITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

RECOMMENDATIONS:

1 **Validate:** Check the first two bytes of the file is 0x424d or 'BM' in ASCII.

2 **Validate:** Check that the following 4 bytes after 'BM' (bytes 3-6) are the file length and that it is equal to the number of bytes in the file.

3 **Validate:** Check that the next four reserved bytes (bytes 7-10) are equal to 0.

4 **Validate:** Check that the file offset (bytes 11-14) is equal to the Bitmap file header size plus the bitmap image header size plus the size of the Color Table if included.

5 **Remove**: Remove filler text in between the image header or color table and pixel array and modify the offset to be equal to the Bitmap File Header + Bitmap Image Header + the Color Table (if it is included) to retain the file format.

6 **Replace:** N/A (Note: It may be possible to replace reserved bytes with all zeros, but at this

time, no application was found to write data at this location).

7 **External Filtering Required:** N/A

8 **Review:** N/A

BMP.4.1: END

4.2 Image Header

This subsection addresses issues that exist within specific fields within the image header.

BMP.4.2: BITMAP IMAGE HEADER

DESCRIPTION:

The Bitmap Image Header relays information about the image stored in the pixel array. It contains attributes about the image. There are two versions of this header; the Core Header and Information Header. The Information and Core Header share similar fields, but differ in lengths and supported color depths. The Core Header contains height and width WORD fields, which define the image size in pixels. A bit count field is also defined which is the number of bits-per-pixel and the value must be one, four, eight, sixteen, or twenty-four bits . An image header contains four byte height and width fields and supports one, four, eight, sixteen, twenty-four and thirty-two bits per pixel.

For Windows 95, the Bitmap Version 4 header was created. It contains the same information as the Core and Information header, but it added information for masking, color space types, and values for gamma. In Windows 98, the version 5 header added a rendering intent field and additional color profile information. All of the headers are shown in Table 4-1 with their respective lengths and the version of Windows in which they were introduced. The main difference between each of them is the additional fields that were added in later versions, which is shown in Table 4-2. The length fields for height and width were also extended beginning with Information Header, as the Core Header only supports 2-byte values for height and width.

Table 4-1. Image Headers With Known Lengths

Header	Specified Length (in bytes)	Supported Since
Core Header Structure	12	Windows 3.0

Info Header Structure	40	Windows 3.0
Bitmap Version 4	108	Windows 95
Bitmap Version 5	124	Windows 98

An image header's size and fields conform to certain standards listed within the MSDN specification. The fields must come in the order specified by the specification for the image file to be read correctly. The following table describes the image header fields such as the width, height, compression mode, resolution of the image (X or Y Pixels per meter), the masks (Red, Green, Blue, and Alpha channel) that specify which bits in each pixel that are for a specific color. All the fields are listed below in Table 4-2.

Table 4-2. Image Header Fields With Known Lengths (in Bytes)

Header Version	Core Header	Information Header	Bitmap v4 Header	Bitmap v5 Header
Info Header Size	4	4	4	4
Image Width	2	4	4	4
Image Height	2	4	4	4
Planes	2	2	2	2
Bits Per Pixel	2	2	2	2
Compression	-	4	4	4
Image Size	-	4	4	4
X Pixels per meter	-	4	4	4
Y Pixels per meter	-	4	4	4
Colors used	-	4	4	4
Important Colors	-	4	4	4
Red Channel	-	-	4	4
Green Channel	-	-	4	4
Blue Channel	-	-	4	4
Alpha Channel	-	-	4	4
Color Space Type	-	-	4	4
Endpoints	-	-	36	36
Gamma for Red	-	-	4	4
Gamma for Green	-	-	4	4
Gamma for Blue	-	-	4	4
Intent	-	-	-	4
ICC Profile Data	-	-	-	4
ICC Profile Size	-	-	-	4
Reserved	-	-	-	4

Each field may be limited in its range values or may be dependent on other fields:

- The Header Size is always the first four bytes of an image header. This field indicates the number of bytes required for the Image Header structure. Some applications use this information to determine which MSDN standard header is implemented because each is of different lengths.

- The Height field, unlike the Width field, may be negative, which would indicate a top down DIB. A top down DIB unlike a regular DIB is read using the first byte of the pixel array as the upper left corner of an image. The standard is a bottom up DIB where the very last byte corresponds to the upper right corner of an image. If the height is negative then then the form of compression must either be BI_RGB or BI_BITFIELDS. The fields can be cropped as shown in the figures below, as the visibility will be modified but the data can remain.

- The Planes field makes up the next two bytes. This field indicates the number of color planes contained within the image. A Color Plane is defined by MSDN as all pixel information for a single color [15]. BMP supports only one color plane, so the only valid value for this field is one.

- The next two bytes is the Bit Count, which stores the bits per pixel of the image stored in the pixel array. The value determines how many bits define a single pixel. Valid values for this field are 1, 4, 8, 16, 24, and 32.. Although if a zero is present in this field, it means that this value is implied by the Joint Photographic Expert Group (JPEG) or Portable Network Graphics (PNG) file format [1]. JPEG and PNG extensions are further described in BMP 4.3: Compression. The supported values vary per Image Header; the Core Header only supports one, four, eight, and twenty-four bits per pixel whereas the Information, V4 and V5 headers support all color depths.

- The Compression field identified from the table above is addressed in Construct BMP.4.3: Compression.

- The Image Size field contains the number of bytes in the image stored in the pixel array. It may be set to zero if the compression method is BI_RGB.

- The image resolution, X and Y pixels per meter for horizontal and vertical respectively, is the number of pixels per meter of the target device for the bitmap. The specification indicates that “an application can use this value to select a bitmap from a resource group that best matches the characteristics of the current device” [1].

- The number of colors used in the image is contained within the Colors Used field. If this field evaluates to zero and the BitCount is less than 16, then the image utilizes the full 2^n colors where n is the value of the Bit Count. All 1, 4, or 8 bpp images must have a color table, with a maximum size of the table is equal to 2^n. If the BitCount is greater than or equal to 16, and the Colors Used field is zero, there is no color table, as it is not required. If the value of this field is non-zero then the number of entries within the Color Table (if included) is equal to this field. There would be a maximum value of 2^n but images of a higher number of bits per pixel, are not required to have a color table and some applications do not create the color table. If images with greater pixel depth implement a color table it could be less than 2^n if all the colors were not used. Although, most images greater than 16bpp do not contain a color table.

- The Important Colors field relays the number of colors necessary to render the image. If this value is zero then all the colors are important. Otherwise the value stored here dictates how many of the entries in the Color Table are read.

- The Color Masks specify the red, green, blue, and alpha masks for each pixel in that order. These are the Red, Green, Blue, and Alpha Channel values from the above table. These fields are only used if the compression field is set to BI_BITFIELDS otherwise they are ignored.

- Color Space is addressed in BMP.4.4. The field in this header denotes the color space type used by the bitmap.

- The Endpoints specify the colors that correspond to red, green, and blue endpoints for the logical color space implemented by the BMP. If the color space is any other value than LCS_CALIBRATED_RGB this structure is ignored.

- Rendering intent, listed above as simply Intent, is addressed in BMP.4.5.

- The Profile Data Offset is found after the Intent field in the Image Header. It indicates in bytes the offset from the beginning of the Image Header to the start of the ICC profile data. This field is ignored if the Color Space Type is not PROFILE_LINKED or PROFILE_EMBEDDED. If this value is 0, there is no ICC profile data. If data does exist when it is not expected (when this value is 0), the data is trailing at the end of the file similar to what is described in BMP 4.9.

- The Profile Size indicates the size of embedded ICC profile data or fully qualified link name if the data is linked. This field is ignored if the Color Space Type is not PROFILE_LINKED or PROFILE_EMBEDDED. If this value is 0, there is no ICC profile data. Similar to the Profile Data Offset field, is this field is also 0, and data exists where the ICC profile would exists, then it is a trailing data issue at the end of the file.

- The Reserved field is found after the Profile Size in the Image Header. It should evaluate to zero.

CONCERNS:

Data Attack - Data attack risks can occur if the width and height have been modified. The width and height may cause an integer overflow causing the memory allocated for a bitmap to be smaller than the data provided in some applications. Data attack risks can occur if the colors used field has been modified. If the value is an unexpected value then there may be a stack based buffer overflow. An example was a crafted bitmap with a palette that contained a large number of colors which has exceeded the color depth. An example of large height and width fields, there has been an RLE_8 compressed bitmap which has been able to trigger a heap-based buffer overflow.

Data Hiding – Data hiding risks can occur in the image header via manipulating the image size. The size field, which is stored as a 4 byte unsigned value, can be modified to be much greater than the actual image header. As much as 2^{32} bytes of data could be hidden in the image header. This same technique could be applied to the number of colors used in the color palette. The bit depth specifies the number of entries in the color palette, so if the bit depth is specified to be a greater value than necessary then the unused portions of the color palette can be used for hiding data.

Data Hiding – Data hiding risks can also occur in the Information Header if fields after Compression are altered as the example in Figure 4-3 below illustrates. Data hiding risks can occur in Bitmap Version 4 and 5 Headers if any field after the color depth has been modified. All these fields can be modified in a manner that doesn’t alter the image stored in the pixel array in some applications. Some applications may attempt to render the image without relying on these fields being accurate. Data Hiding risks can also occur within some of the same fields when they are triggered to be ignored and are overwritten.

PRODUCT: BMP

LOCATIONS:

The Bitmap Image header immediately follows the File header. It conforms to one of the known sizes and takes up exactly that amount of space.

EXAMPLE 1 (INCORRECT IMAGE SIZE, CROPPED IMAGE)

```
Before:
BM8À......6...(. 42 4d 38 c0 00 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 80 00 00 00 80 00 00 00 01 00 18 00 00 00
................ 00 00 02 c0 00 00 c3 0e 00 00 c3 0e 00 00 00 00
................ 00 00 00 00 00 00 f4 ea ...(pixel data starts with f4)

After (Image size is incorrect, width and height are cropped):
BM........6...(. 42 4d ff 2f 00 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 80 00 00 00 75 00 00 00 01 00 18 00 00 00
................ 00 00 ca 2f 00 00 c3 0e 00 00 c3 0e 00 00 00 00
................ 00 00 00 00 00 00 f4 ea ...(pixel data starts with f4)

...44912 bytes later in the Pixel data...
This is the star 54 68 69 73 20 69 73 20 74 68 65 20 73 74 61 72
t of hidden data 74 20 6f 66 20 68 69 64 64 65 6e 20 64 61 74 61
. After this the 2e 20 41 66 74 65 72 20 74 68 69 73 20 74 68 65
 image should st 20 69 6d 61 67 65 20 73 68 6f 75 6c 64 20 73 74
op and the rest  6f 70 20 61 6e 64 20 74 68 65 20 72 65 73 74 20
of this text wou 6f 66 20 74 68 69 73 20 74 65 78 74 20 77 6f 75
ld never appear  6c 64 20 6e 65 76 65 72 20 61 70 70 65 61 72 20
as pixels.öðúöðú 61 73 20 70 69 78 65 6c 73 2e f6 f0 fa f6 f0 fa
```

The images in Figure 4-1 and Figure 4-2 show the effect of manipulating the image properties to crop an image and hide data in the pixel data where the viewer will not look. The upper right pixels of Figure 4-2 are different from Figure 4-1 which is where the text "This is the start..." begins in the pixel data. If this data is shifted a few bytes, the viewer would only show a cropped image which may not be noticeable to the viewer. In this example, there is more data than what the image calls for, which should be flagged as a problem, however, many viewers will still display the image.

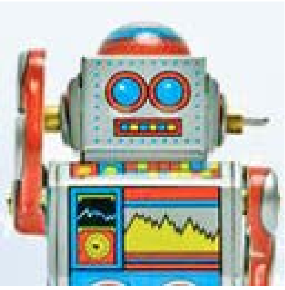


Figure 4-1. Regular BMP - Before

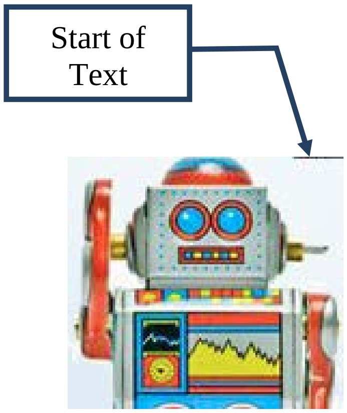


Figure 4-2. Cropped BMP - After

EXAMPLE 2 (ALTERED COLORS USED)

```
 BM6.<.....6...(. 42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
....<.........ÿÿ 00 00 00 00 3c 00 00 00 00 00 00 00 00 00 ff ff
ÿÿ....           ff ff 00 00 00 00
```

EXAMPLE 3 (BMP IMAGE SIZE, RESOLUTION, COLORS USED & IMPORTANT)

```
BM6.<.....6...(. 42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
....<........... 00 00 00 00 3c 00 00 00 00 00 00 00 00 00 00 00
......           00 00 00 00 00 00
```

EXAMPLE 4 (ALTERED IMAGE SIZE, RESOLUTION, COLORS USED & IMPORTANT)

```
BM6.<.....6...(. 42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
.._Modified_this 00 00 5f 4d 6f 64 69 66 69 65 64 5f 74 68 69 73
_area_           20 61 72 65 61 5f
```

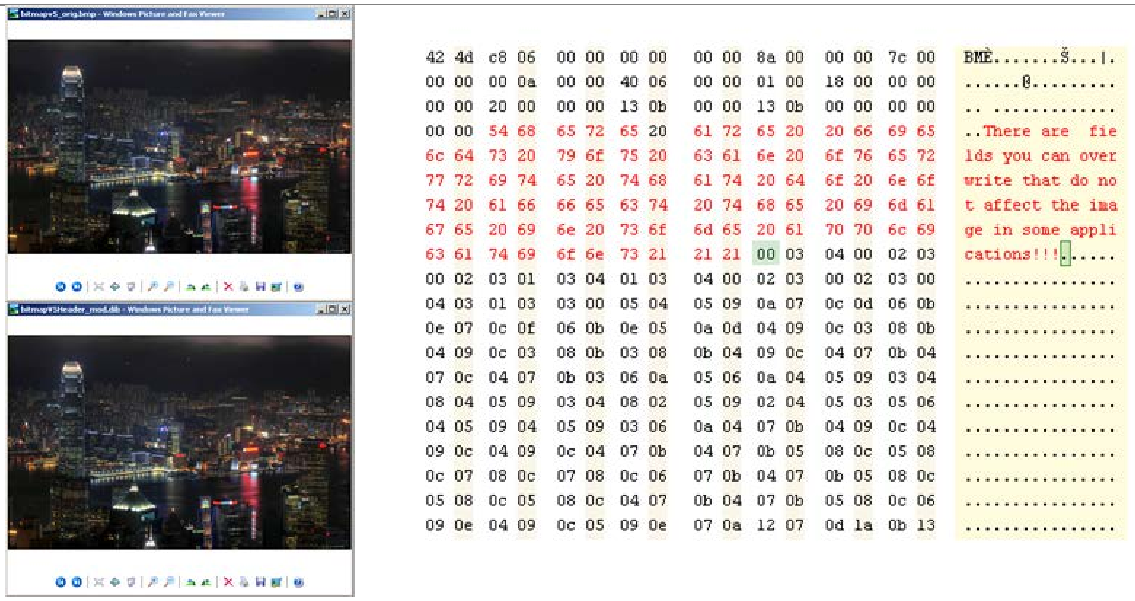
EXAMPLE 5 (BMP VERSION 5 HEADER)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
.. ............. 00 00 20 00 00 00 13 0b 00 00 13 0b 00 00 00 00
......ÿÿÿÿÿÿÿÿ.. 00 00 00 00 00 ff ff ff ff ff ff ff ff ff 00 00
....ÿ niW....... 00 00 00 00 ff 20 6e 69 57 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........        00 00 00 00 00 00 00 00 00 00
```

EXAMPLE 6 (ALTERED BMP VERSION 5 HEADER)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
..There is space 20 69 6e 20 74 68 65 20 76 65 72 73 69 6f 6e 20
in the version   35 20 68 65 61 64 65 72 20 74 6f 20 6d 6f 64 69
5 header to modi 66 79 20 65 6c 65 6d 65 6e 74 73 20 77 69 74 68
fy elements with 6f 75 74 20 61 66 66 65 63 74 69 6e 67 20 74 68
out affecting th 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
e picture!       65 20 70 69 63 74 75 72 65 21
```

The Bitmap Version 5 Header alteration example demonstrates that any fields in the Bitmap Version Four Header which are also found in the Version Five Header can also be modified. An example below demonstrates the original image on top left and the modified image with text in the header shown in the lower left. In this example, the field Important Colors count is overwritten with text. This modification includes changing the Channel bitmask fields, the color space, the ICC profile fields, among other fields in the Bitmap V5 header, which means many applications ignore the fields and attempt to render the image without them.


bitmapv5_orig.bmp - Windows Picture and Fax Viewer
bitmapV5Header_mod.dib - Windows Picture and Fax Viewer

```
42 4d c8 06  00 00 00 00  00 00 8a 00  00 00 7c 00   BMÈ.......Š...|.
00 00 00 0a  00 00 40 06  00 00 01 00  18 00 00 00   ......@.........
00 00 20 00  00 00 13 0b  00 00 13 0b  00 00 00 00   .. .............
00 00 54 68  65 72 65 20  61 72 65 20  20 66 69 65   ..There are  fie
6c 64 73 20  79 6f 75 20  63 61 6e 20  6f 76 65 72   lds you can over
77 72 69 74  65 20 74 68  61 74 20 64  6f 20 6e 6f   write that do no
74 20 61 66  66 65 63 74  20 74 68 65  20 69 6d 61   t affect the ima
67 65 20 69  6e 20 73 6f  6d 65 20 61  70 70 6c 69   ge in some appli
63 61 74 69  6f 6e 73 21  21 21 00 03  04 00 02 03   cations!!!......
00 02 03 01  03 04 01 03  04 00 02 03  00 02 03 00   ................
04 03 01 03  03 00 05 04  05 09 0a 07  0c 0d 06 0b   ................
0e 07 0c 0f  06 0b 0e 05  0a 0d 04 09  0c 03 08 0b   ................
04 09 0c 03  08 0b 03 08  0b 04 09 0c  04 07 0b 04   ................
07 0c 04 07  0b 03 06 0a  05 06 0a 04  05 09 03 04   ................
08 04 05 09  03 04 08 02  05 09 02 04  05 03 05 06   ................
04 05 09 04  05 09 03 06  0a 04 07 0b  04 09 0c 04   ................
09 0c 04 09  0c 04 07 0b  04 07 0b 05  08 0c 05 08   ................
0c 07 08 0c  07 08 0c 06  07 0b 04 07  0b 05 08 0c   ................
05 08 0c 05  08 0c 04 07  0b 04 07 0b  05 08 0c 06   ................
09 0e 04 09  0c 05 09 0e  07 0a 12 07  0d 1a 0b 13   ................
```

Figure 4-3. Hiding Data by Overwriting Header Fields

RECOMMENDATIONS:

1 **Validate:** Verify the Image Header adheres to the length bounds specified by the MSDN specification, as referenced in Table 4-1.

2 **Validate:** Verify that the width is not a negative number.

3 **Validate:** Verify the color depth is either 1, 4, 8, 16, 24, or 32 for an information header and above or 1, 4, 8, or 24 for a core header.

4 **Validate:** Verify colors used is less than or equal to 2^n, where n is the bit count.

5 **Validate:** Verify that the image header size is less than a predefined value.

6 **Validate:** Identify which colors are used, then zero out any unused entries in the color palette.

7 **Remove**: Remove data going past one of the length bounds and modify the file offset in the file header and image header size to retain the file format.

8 **Replace:** Replace a Bitmap Version Four Header or Version Five Header with an Information Header and replace the file offset in the file header and image header size with adjusted values to retain the file format. The ICC information after the pixel array should be treated as trailing

data and removed if it exists since it is not supported by the Information Header and will not be read by the application. Note that this course of action may distort the image.

9 **Replace:** Replace colors important with 0, which means that all colors are required or important to the application; this is typically the default value.

10 **Replace:** Replace colors used with the size of the Color Table.

11 **Replace:** Replace the data after the Color Depth with filler text, e.g. all zeros, since it has been observed that these fields can be incorrect and the image remains viewable. Note that this course of action may distort the image since the header contains incorrect information.

12 **Replace:** Replace ignored fields with filler text; e.g. all zeros.

13 **External Filtering Required:** N/A

14 **Review:** N/A

15 **Reject:** Fail Bitmap files that contain an improperly formatted header.

BMP.4.2: END

BMP.4.3: COMPRESSION

DESCRIPTION:

The Compression field indicates whether or not a form of compression is used. There are six valid options that can be used according to the specification.

- BI_RGB (0x00000000) indicates the BMP is uncompressed.

- BI_RLE8 (0x00000001) and BI_RLE4 (0x00000002) use run length encoding on a BMP file using four or eight bits per pixel color depth. This is detailed in Compression Modes 3.2.

- BI_BITFIELDS (0x00000003) is used with sixteen and thirty-two bits per pixel color depth and does not compress the file, but indicates the color table contains three color masks that specify the RGB components.

- The final options are BI_JPG (0x00000004) and BI_PNG (0x00000005) which specify that the image is compressed using the JPEG file interchange format or the PNG file interchange format respectively. These previous two compression formats aren't used for rendering images to the screen, only for printing to devices. According to the specification, this compression options is defined as "an extension which is not intended as a means to supply general JPEG and PNG decompression to applications, but rather to allow applications to send JPEG- and PNG-compressed images directly to printers having hardware support for JPEG and PNG images" [1]. Bitmap file images with a JPEG in the data section will error on opening on many image viewing applications because it is not meant for displaying to the screen.

CONCERNS:

Data Hiding – Data hiding risks can occur if the compression field is overwritten. Some applications may not able to render a Bitmap with a modified compression field.

Data Attack – When compression is used, files may be crafted by using this field or the data itself to achieve code execution [3].

LOCATIONS:

The Compression Field is located in the Information Header, Bitmap Version 4 Header, and Bitmap Version 5 Header.

EXAMPLE 1 (BMP IMAGE COMPRESSION BI_RGB)

```
BM6.<.....6...(. 42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 00 05 00 00 00 04 00 00 01 00 18 00 00 00
....<........... 00 00 00 00 3c 00 00 00 00 00 00 00 00 00 00 00
......           00 00 00 00 00 00
```

EXAMPLE 2 (ALTERED COMPRESSION)

```
BM6.<.....6...(. 42 4d 36 00 3c 00 00 00 00 00 36 00 00 00 28 00
..............ÿÿ 00 00 00 05 00 00 00 04 00 00 01 00 18 00 ff ff
ÿÿ..<........... ff ff 00 00 3c 00 00 00 00 00 00 00 00 00 00 00
......           00 00 00 00 00 00
```

RECOMMENDATIONS:

1 **Validate:** Verify the compression evaluates to a valid option from the MSDN specification.

2 **Remove**: N/A

3 **Replace:** Replace the compression type with a valid option. Note that this course of action may distort the image.

4 **External Filtering Required:** N/A

5 **Review:** N/A

6 **Reject:** Fail Bitmap files that contain BI_JPEG or BI_PNG compression types because they are not used for rendering images to the screen, only for printing devices.

7 **Reject:** Fail Bitmap files that contain an invalid compression type, any that are not defined by MSDN, which are defined above in this construct. Applications have been observed to report an error processing header data.

BMP.4.3: END

BMP.4.4: COLOR SPACE

DESCRIPTION:

The Color Space field indicates the color space of the image stored within the pixel array. It is limited to five values. Bitmap Version 4 supports only the first option listed in this section whereas Bitmap Version 5 supports all the values listed.

- LCS_CALIBRATED_RGB (0x00000000) indicates that endpoints and gamma values are in the correct positions.
- LCS_sRGB (0x00000001) indicates the color space will be sRGB.
- LCS_WINDOWS_COLOR_SPACE (0x00000002) indicates the system default color space is used, which is defined as sRGB [1].
- PROFILE_LINKED (0x00000003) option enables an International Color Consortium (ICC) profile to be linked using a fully qualified name.
- PROFILE_EMBEDDED (0x00000004) enables the embedding of an ICC profile directly in to the BMP file.

CONCERNS:

The data hiding risk in this construct is minimal as many image viewers may use the system default color space and attempt to display the image as best as possible. Changing the color space field in various images had no impact, but the value of this field is dependent on the version of the header (v4 or v5) and if an ICC profile exists (LINKED or EMBEDDED), which follows the pixel data.

LOCATIONS:

The Color Space Field is located in the Bitmap Version 4 Header, and Bitmap Version 5 Header.

EXAMPLE 1 (BMP VERSION 5 HEADER COLOR SPACE)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
.. ............. 00 00 20 00 00 00 13 0b 00 00 13 0b 00 00 00 00
......ÿÿÿÿÿÿÿÿ.. 00 00 00 00 00 ff ff ff ff ff ff ff ff ff 00 00
....ÿ .......... 00 00 00 00 ff 20 01 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........        00 00 00 00 00 00 00 00 00 00
```

EXAMPLE 2 (ALTERED COLOR SPACE)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
.. ............. 00 00 20 00 00 00 13 0b 00 00 13 0b 00 00 00 00
......ÿÿÿÿÿÿÿÿ.. 00 00 00 00 00 ff ff ff ff ff ff ff ff ff 00 00
....ÿ TEXT...... 00 00 00 00 ff 20 54 45 58 54 00 00 00 00 00 00
```

```
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........        00 00 00 00 00 00 00 00 00 00
```

RECOMMENDATIONS:

1 **Validate:** Verify Bitmap Version 4 headers contain a color space field that evaluates to 0x00000000.

2 **Validate:** Verify Bitmap Version 5 Headers contain a color space field that evaluates to a valid option from the MSDN specification (values 0 through 4).

3 **Remove**: N/A

4 **Replace:** Replace the color space with a valid option listed in the specification and in the Description section above. If a v4 Header, then select LCS_CALIBRATED_RGB, or if a v5 Header, select either LCS_CALIBRATED_RGB, LCS_sRGB, or LCS_WINDOWS_COLOR_SPACE (default for system). Note that this course of action may distort the image.

5 **External Filtering Required:** N/A

6 **Review:** N/A

7 **Reject:** Reject Bitmap files that contain an invalid Color Space value.

8 **Reject:** Reject Bitmap files with a linked in ICC profile.

BMP.4.4: END

BMP.4.5: INTENT

DESCRIPTION:

The Intent field controls the rendering intent of the bitmap. This field supports four different options. Each option relates to an ICC Profile. An ICC Profile is data that defines a color space, or a color output device. It allows for a conversion between the device color space and device independent color spaces. ICC Profiles are created to provide a cross-platform device profile format [2], which allows images with embedded ICC profiles to be viewed on multiple systems.

- LCS_GM_BUSINESS (0x00000001) indicates the Bitmap will render with the Absolute Colormetric.
- LCS_GM_GRAPHICS (0x00000002) indicates the Bitmap will render with the Saturation ICC profile.
- LCS_GM_IMAGES (0x00000004) indicates the Bitmap will render with the Relative Colorimetric ICC profile.

- LCS_GM_ABS_COLORIMETRIC (0x00000008) indicates the Bitmap will render with the Absolute Colorimetric ICC profile. MSDN indicates that this means that the rendering maintains the white point and matches the colors to their nearest color in the destination gamut [1].

A value entered that does not evaluate to one of these options is ignored.

CONCERNS:

From experiments, manipulating this field does not create a drastic or noticeable difference in the image, or one that is large enough to hide a large amount of data. Many applications ignore this value if it's incorrect and still render the image, when compared to the original image with the correct Intent. It is unlikely there will be a large hidden data risk with this field, but to ensure this is true the 4-byte field should evaluate to one of the options above.

LOCATIONS:

The Color Space Field is located in the Bitmap Version 5 Header; it is defined as a DWORD called bV5Intent.

EXAMPLE 1 (BMP VERSION 5 HEADER INTENT)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
.. ............. 00 00 20 00 00 00 13 0b 00 00 13 0b 00 00 00 00
......ÿÿÿÿÿÿÿÿ.. 00 00 00 00 00 ff ff ff ff ff ff ff ff ff 00 00
....ÿ .......... 00 00 00 00 ff 20 01 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........        00 00 00 00 00 00 00 00 00 00
```

EXAMPLE 2 (ALTERED INTENT)

```
BMÈ.......Š...|. 42 4d c8 06 00 00 00 00 00 00 8a 00 00 00 7c 00
......@......... 00 00 00 0a 00 00 40 06 00 00 01 00 18 00 00 00
.. ............. 00 00 20 00 00 00 13 0b 00 00 13 0b 00 00 00 00
......ÿÿÿÿÿÿÿÿ.. 00 00 00 00 00 ff ff ff ff ff ff ff ff ff 00 00
....ÿ .......... 00 00 00 00 ff 20 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
..........TEXT.. 00 00 00 00 00 00 00 00 00 00 54 45 58 54 00 00
.........        00 00 00 00 00 00 00 00 00 00
```

RECOMMENDATIONS:

1 **Validate:** Verify the Intent field evaluates to a valid option from the MSDN specification.

2 **Remove**: N/A

3 **Replace:** Replace the Intent field with a valid Intent value. Note that this course of action may distort the image.

4 **External Filtering Required:** N/A

5 **Review:** N/A

BMP.4.5: END

4.3 Color Table

BMP.4.6: Color Table

DESCRIPTION:

BMP utilizes a variable length color table for color depths that are eight bits per pixel or smaller. This structure is optional for sixteen and thirty-two bpp color depths. A Bitmap that utilizes the color table is referred to as a Palette-Indexed Bitmap. The color table contains colors used by BMP in order of importance to allow the image to be rendered properly. The maximum size of this structure is 2^n, where n is the color depth of the image, however the actual size should be equal to value of the field Colors Used if it is non-zero. Color depth is defined as the number bits that are used for the color of a pixel.

The Color table is optional, and is implemented as a set of RGBQUAD structures or a set of RGBTRIPLE structures. If the color table is omitted when it is required, many applications may give an error that the file has been corrupted. If the color table is replaced with text or incorrect values, the end result of the image will be different from the original.

CONCERNS:

Data Hiding - Data hiding risks may exist if the fields are modified to contain text characters. Hiding may also occur when the colors of the image are modified in the color table to mask other content in the image, such as duplicate adjacent colors.

Data Attack - Data attack risks occur if this field is overwritten and extends beyond the maximum size or if a color is chosen outside the range of the color table which is outside the bounds of memory the application maintains for that table.

PRODUCT: BMP

LOCATION:

The color table is located after the Image Header and before the Pixel Array. It is either a set of RGBQUAD structures or a set of RGBTRIPLE structures, as shown below [1]. An array of RGBQUAD structures are used from the BitMapInfo structure, and an array of RGBTRIPLE structures are referenced through the BitMapCoreInfo structure, both defined from MSDN. The 4-byte entries are common and the 3-byte entires are used with the OS/2 Core header.

RGBQUAD Structure [1]:

```
typedef struct tagRGBQUAD {
  BYTE rgbBlue;
  BYTE rgbGreen;
  BYTE rgbRed;
  BYTE rgbReserved;
} RGBQUAD;
```

RGBTRIPLE Structure [1]:

```
typedef struct tagRGBTRIPLE {
  BYTE rgbtBlue;
  BYTE rgbtGreen;
  BYTE rgbtRed;
} RGBTRIPLE;
```

EXAMPLE:

In this example, the background color of the resulting BMP had a solid white background that changed to grey as shown in the example below. This modification indicates a visible impact, although one that may go undetected. This is because checking to ensure the color table is completely valid is not straightforward.

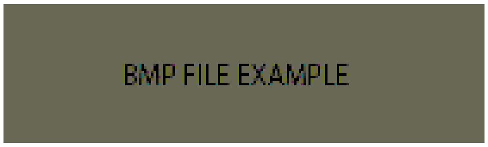


Figure 4-4. 4-bpp BMP Example with Text Replacement in Color Table.

```
BM.b......v...(. 42 4d d8 62 01 00 00 00 00 00 76 00 00 00 28 00
......h......... 00 00 f8 01 00 00 68 01 00 00 01 00 04 00 00 00
..bb............ 00 00 62 62 01 00 12 0b 00 00 12 0b 00 00 00 00
..........This i 00 00 00 00 00 00 00 00 00 00 54 68 69 73 20 69
s text in place  73 20 74 65 78 74 20 69 6e 20 70 6c 61 63 65 20
of the color tab 6f 66 20 74 68 65 20 63 6f 6c 6f 72 20 74 61 62
le.UU.DDD.333."" 6c 65 2e 55 55 00 44 44 44 00 33 33 33 00 22 22
"............... 22 00 11 11 11 00 11 11 11 11 11 11 11 11 11 11
```

Modifying the color table to mask out content is also possible and a data hiding risk, as opposed to hiding data in the table itself. In the example below, the original image is on the left with text in the image. The second example modified the color table to replace the white background with black. In this color table, there are duplicate colors, and since each color is indexed, it is possible to mask content without modifying the pixel data.

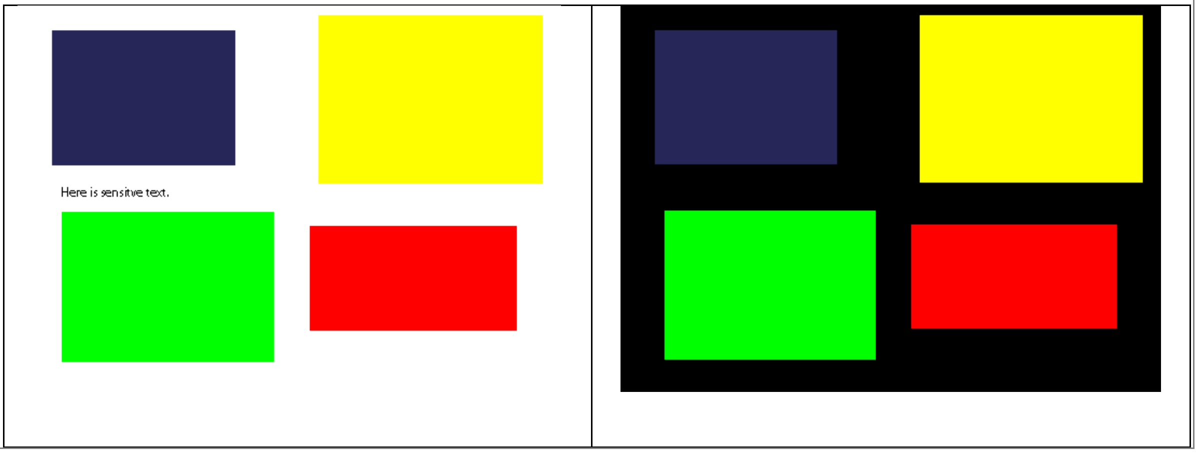


Figure 4-5. Image with Sensitive Text

Figure 4-6. Image with Color Table Modified to Hide Text

RECOMMENDATIONS:

1 **Validate:** Verify that the color table is not bigger then 2^n where n is the color depth, the number of bits used to represent color.

2 **Validate:** Verify that the color table does not contain duplicate color values.

3 **Validate:** Verify that all pixels select a valid entry from the color table.

4 **Remove:** If the value of colors used is a valid non zero number then remove all bytes after this value. The file offset must also be modified to retain the file format.

5 **Remove:** Remove the color table and convert the image such that it no longer requires the table, this will change the output of the image, if it can be done.

6 **Remove:** Remove the color table if the image has a color depth of sixteen, twenty-four, or thirty-two bpp, and convert the pixels from using the Index into the color table. The file offset must also be modified to retain the file format. MSDN indicates that for 16, 24, 32 bpp that color tables are not required. For 1, 4, or 8 bpp, a color table must be present.

7 **Replace:** Replace the color table with a bit masked version of the original in manner that is not easily reversed.

8 **Replace:** Replace the file offset in header with Bitmap File Header + Bitmap Image Header + Color Table (if it is included).

9 **Replace:** Replace the Offset data between the image header or color table and pixel array with a different set of values, e.g. all zeros.

10 **External Filtering Required:** N/A

11 **Review**: N/A

12 **Reject**: Fail Bitmap files that are 8 bpp or less and do not contain a color table, in which case the application might not render the image because it is required.

13 **Reject**: Fail Bitmap files that contain a color table with a color depth beyond a threshold.

BMP.4.6: END

4.4 Pixel Array

This subsection addresses issues that exist within the pixel array. The pixel array is composed of the byte values of consecutive rows of the bitmap. Each row consists of consecutive bytes representing the pixels from left to right. The pixel array is stored bottom up which means the last byte signifies the upper-right corner of an image. If the height is negative then the Pixel Array is stored top down where the first byte signifies the upper left corner of an image. The pixel array generally ends on a 32-bit boundary which makes padding possible when the pixel array does not naturally end on this boundary.

The pixel array is formatted differently according to the color depth. The color depth is proportional to the size of each pixel stored in the pixel array. For example, if the color depth is 24 bits per pixel then there are 24 bits of information per pixel.

BMP.4.7: Image Content

DESCRIPTION:

BMP utilizes a pixel array to store an image. The pixel array stores information about each pixel within the image.

CONCERNS:

Data Disclosure - The actual image or picture itself may disclose sensitive information.

Data Hiding - Content may pose a data hiding risk if text has been placed within an image in such a manner that it is difficult to detect. For example, text that has been modified to contain a similar color to that of its surrounding pixels.

Data Hiding – Every row in the pixel array is padded to a multiple of 4 bytes, therefore as many as 31 bits could be unused. These bits could be used for hiding data.

PRODUCT: BMP

LOCATION:

The image is stored within the Pixel Array of a BMP.

RECOMMENDATIONS:

1 **Validate:** N/A

2 **Remove:** N/A

3 **Replace:** Replace the pixel array with a version that has had a bitmask applied to the least significant bits in a manner that can't be reversed easily to mitigate data hiding or covert channels.

4 **External Filtering Required:** N/A

5 **Review**: Review the contents of an image to ensure sensitive information is not disclosed.

6 **Reject**: N/A

BMP.4.7: END

BMP.4.8: LSB Steganography

DESCRIPTION:

In general, steganography refers to any data hiding effort. Image steganography commonly employs the least significant bits (LSBs) in the image data. In a BMP file, it is possible to employ LSB steganography and write data in to the ending bytes of the image data. This document will not address detection; instead, it will focus on mitigation, i.e. steg destruction.

CONCERNS:

Data Hiding - LSB steganography that is done well is difficult to detect. It poses a data hiding risk because it is a covert channel.

PRODUCT: BMP

LOCATION:

This problem exists in the Pixel Array of the BMP file format.

RECOMMENDATIONS:

1 **Validate:** N/A

2 **Remove:** N/A

3 **Replace:** Replace the least significant bit with a predefined value.

4 **Replace:** Replace the pixel array with a version that has had a bitmask applied to the least significant bits in a manner that can't be reversed easily.

5 **Replace**: If the color depths are four or eight bpp then the pixel array can be replaced with a compressed version.

6 **Replace**: If the Compression type is RLE4 or RLE8 then the pixel array may be replaced with a decompressed version.

7 **External Filtering Required:** N/A

8 **Review**: N/A

9 **Reject**: N/A

BMP.4.8: END

BMP.4.9: Trailing Data

DESCRIPTION:

BMP files are processed by some applications using the structure sizes indicated within the file and image headers. Data placed where a structure should start may cause an image to render incorrectly, however data placed after the final structure, indicated by fields within the header, is ignored by some applications. This allows the image to render properly although the structure of the file has been modified. Data can exist at the very end of the image, which may follow the image data or the ICC color profile, if it's implemented. In either case, applications might ignore data that doesn't belong there and has no functional impact on the file.

Bitmap files may also contain gaps between the color table and pixel data. A gap may also exist between the pixel data and ICC profile if it's implemented. The MSDN specification indicates that pixel data or pixel array adds padding such that each row of the image aligns on a 4-byte boundary, which is dependent on the width of the image as well as the color depth. Padding may exist in other locations but is only needed to align on a 4-byte boundary, which means that there should be less than or equal to three bytes of padding at various locations at most.

CONCERNS:

Data Hiding - Data hiding risks may exist when there is trailing padding data appended after the pixel array or ICC profile.

PRODUCT: BMP

LOCATIONS:

Trailing data may be found after the very last structure in a Bitmap file. This would be either the pixel array or ICC profile.

EXAMPLE 1 (NORMAL BMP FILE SIZE AND RESERVED BYTES)

```
BM6.......6...(. 42 4d 36 04 00 00 00 00 00 00 36 00 00 00 28 00
................ 00 00 01 00 00 00 01 00 00 00 01 00 18 00 00 00
................ 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
.........Trailin 00 00 00 00 00 00 00 00 00 54 72 61 69 6c 69 6e
g data may be ap 67 20 64 61 74 61 20 6d 61 79 20 62 65 20 61 70
pended to a BMP  70 65 6e 64 65 64 20 74 6f 20 61 20 42 4d 50 20
file without mod 66 69 6c 65 20 77 69 74 68 6f 75 74 20 6d 6f 64
ifying the image 69 66 79 69 6e 67 20 74 68 65 20 69 6d 61 67 65

There are 71 extra bytes appended to the end of this 1 by 1 bitmap image.
```

RECOMMENDATIONS:

1 **Validate:** Validate there are no trailing bytes after the Pixel Array by checking the region after Bitmap File Header Size + Image Header Size + Image Size.

2 **Remove**: Remove all trailing padding bytes from the end of the file, whether the bytes follow the pixel data (with no ICC color profile), or the ICC color profile, if implemented.

3 **Replace**: Replace any necessary padding bytes that are used to align on a 4-byte boundary with zero.

4 **Replace:** Replace all trailing bytes with filler text, e.g. all zeros.

5 **External Filtering Required:** N/A

6 **Review:** N/A

7 **Reject:** Fail Bitmap files that contain trailing bytes following the pixel data or ICC profile (if implemented).

BMP.4.9: END

BMP.4.10: Run Length Encoding

DESCRIPTION:

Run Length Encoding is used in conjunction with Bitmaps of four and eight bits per pixel color depths as a compression method. It uses two forms, absolute and encoded, to reduce the size of the image file.

Encoded mode consists of two bytes, the first is the number of pixels to be drawn consecutively and the second contains the color index to draw those pixels in. The termination of the line, termination of the file, or a delta is indicated by a zero in the first byte. The subsequent byte can contain a zero for a terminal line, a one for end of the file, or a two for a delta. Encoded mode can be useful for sequences of similar content and of the same color. In Encoded Mode for RLE4 for each byte of data the higher nibble is alternated with the lower nibble n times, where *n* is the value of the first byte. In RLE8 the whole second byte is repeated *n* times.

Absolute mode is enabled when the first byte in the pair is set to zero. The second byte represents the number of bytes that will follow. The following bytes contain the color index of one pixel. The termination of the line or of the file is indicated by a zero in the first byte; the subsequent byte may be zero, one or two and have the same effect as in Encoded Mode. Absolute mode is used primarily for sequences of varying contents and of different colors. In Absolute Mode for RLE4 the color indexes are stored per high nibble and lower nibble of each byte. This means there are two color indexes per byte. The second byte in Absolute Mode for RLE8 must be between 0x00000003 (3) and 0x000000ff (255) to draw pixels.

CONCERNS:

Data Attack - Data attack risks may exist when either form of Run Length Encoding is enabled. It is possible that the bitmap data could contain Run Length Encoded data that was crafted to target the failure of an application’s boundary check on the data, which may result in a heap buffer overflow and allow code execution.

PRODUCT: BMP

LOCATIONS:

Run Length Encoding may be used with Bitmaps of four or eight bit color fields and is found within the pixel array. It is enabled using the Compression field 4.2.5 in the Image Header.

RECOMMENDATIONS:

1 **Validate:** Verify that the color depth is four or eight bits per pixel and the compression method selection corresponds to that color depth.

2 **Validate:** Verify that the run length encoding does not run past the image dimensions.

3 **Remove:** N/A

4 **Replace:** Replace the pixel array with a version that does not use Run Length Encoding. Replace the compression method with one that doesn't use Run Length Encoding.

5 **External Filtering Required:** N/A

6 **Review**: N/A

7 **Reject**: Fail Bitmap files that contain Run Length Encoding.

BMP.4.10: END

4.5 ICC Profile

This subsection addresses a specific issue that arises when a Bitmap Version 5 Header is used in combination with an International Color Consortium (ICC) Profile. As mentioned before, ICC Profiles provide data that defines a color space, or a color output device. It allows for a conversion between the device color space and device independent color spaces.

BMP.4.11: ICC Profile

DESCRIPTION:

An ICC profile is an ICC-specified data format for describing the color scheme of an image. This is an optional feature that can be enabled in the Bitmap Version 5 Header and can either be linked or embedded into the BMP file. The linked portion requires a fully qualified name according to MSDN standards. The ICC Profile is located at the offset indicated in the image header plus the size of the file header. ICC is a separate specification and used in other document formats, so it is not specific to BMP files. ICC is not covered in this ISG, but how the BMP file format incorporates an ICC profile is introduced in this section.

An image may be dependent on its ICC profile to render its contents correctly. Altering the profile or modifying it will change the visibility of the image, if it is displayable at all. The ICC profile can act as a covert channel which means the recommendations outlined in this document may alter the image from its original state.

CONCERNS:

Data Disclosure - A data disclosure risk exists if the ICC profile is linked to the file, which defines a file path. The path to the ICC profile may disclose sensitive information.

Data Hiding and Attack - The concerns in the restricted ICC profile may encompass data hiding and data attack risks. Data hiding risks may exist when fields inside the profile data structure have been altered to obscure content. Data hiding may also occur if the ICC file is ignored by the application rendering the bitmap image and the ICC file contains sensitive data. Data attack risks may exist when the embedded file or the linked file has been maliciously altered. ICC files are separate files that are used by many applications. There have been reports of vulnerabilities of integer overflow errors when parsing a color profile through various tags in the specific file format, which has lead to heap-based buffer overflows.

PRODUCT: BMP – BITMAPV5HEADER

LOCATION:

A restricted ICC profile may exist at the end of a Bitmap file with a Version 5 Header.

RECOMMENDATIONS:

1 **Validate:** Verify that a Bitmap Version 5 Header is used, and the profile size information is accurate. Validation of ICC profile is outside the scope of this document.

2 **Remove:** Set Color Space in the Image Header to a value other than one that uses an ICC Profile and proceed to remove the ICC profile data after the pixel array. Note that this course of action may distort the image.

3 **Replace:** Replace the ICC profile data with a known ICC profile. This would require updating the ICC profile size if the lengths differ and the total size of the file. Note that this course of action may distort the image.

4 **Replace:** Replace the Bitmap Version 5 Header with an Information Header. Replace the file offset to the correct length of File Header size + Information Header size. Remove all ICC profile data after the pixel length.

5 **External Filtering Required:** Pass the contents of the ICC Profile field to a filter capable of handling ICC profiles. ICC profiles can be standalone files and may be used in other graphics formats.

6 **External Filtering Required**: Pass the file path of the ICC Profile to an external filter.

7 **Review**: N/A

8 **Reject**: Reject Bitmap files that contain an ICC profile.

9 **Reject**: Reject BMP file that contain an ICC linked profile.

BMP.4.11: END

5 ACRONYMS

Table 5-1. Acronyms

Acronym	Denotation
ASCII	American Standad Code for Information Interchange
BMP	Bitmap
DDB	Device Dependent Bitmap
DIB	Device Independent Bitmap
ICC	International Color Consortium
JPEG	Joint Photographic Experts Group
LSB	Least Significant Bits
MSDN	Microsoft Developer Network
PNG	Portable Network Graphics
RLE	Run Length Encoding

6 REFERENCED DOCUMENTS

[1] Bitmap Reference. Microsoft Developer Network. Available at: http://msdn.microsoft.com/en-us/library/dd183388(v=VS.85).aspx

[2] International Color Consortium. Specification ICC.1:2004-10 (Profile Version 4.2.0.0). Image Technology Colour Management- Architecture, Profile Format, and Data Structure. 5/22/2006. Available at: http://www.color.org/ICC1v42_2006-05.pdf

[3] Device –Independent Bitmaps. Online Reference available at: http://msdn.microsoft.com/en-us/library/dd183562(v=VS.85).aspx

[4] MSDN Glossary. Online Reference available at: http://msdn.microsoft.com/en-us/library/cc230520(v=prot.10).aspx

[5] Sustainability of Digital Formats Planning for Library of Congress Collections. “Bitmap Image File (BMP), Version 5.” Online reference available at: http://www.digitalpreservation.gov/formats/fdd/fdd000189.shtml