Exploit Delivery using Steganography as Covert Channel

Tariq Z. Khairallah

Tar20168014@std.psut.edu.jo

Princess Sumaya University for Technology (PSUT)

King Hussein School of Information Technology

Abstract

Security is a major topic in recent researchers, and finding new methods deliver was the motivations for this research to emphasize and as an awareness to browsers developers to handle images with more care and to enhance exploit detection technology in the browser, as this paper proves that hackers now can attack users with just an image.

***Keywords**:* Steganography, exploitation, Malicious Code

1. Introduction

The Internet is often filled with ways to deliver Malicious Code to victims utilizing direct and indirect methods to achieve the resultز

This paper discusses delivering exploits “malicious code” to victims using digital steganography by encoding JavaScript code inside the image pixels which are then decoded using an HTML 5 Canvas element that allows for dynamic rendering of images, allowing the attacker to take controls of victims system, steal confidential data or to take the system downs by infecting it with malicious code [1] [2] utilizing images as mediator to deliver them ,these can be done using different covert channels such as advertisements [3], social media sites and popular sites [4].

The exploit delivery using steganography is very risky and hard to detect. In this paper, we analyze these types of attacks are analyzed and impact of the attack is studied. Then we suggest mitigation techniques handles these threats.

2. Problem

Exploit kits and hackers started using steganography as a new method for delivering malware [1] and sometimes found in social media [5] or by using advertisement images [3] [4]even hiding card data inside images [2] and lately to deliver ransomware “*is computer malware that installs covertly on a victim's computer, executes a cryptovirology attack that adversely affects it, and demands a ransom payment to decrypt it or not publish it*” [6].

All emerged researchers now are focusing on delivering and hiding exploits or malware inside images [7] [8], and methods to detect and eliminate such threats.

3. Overview of Digital Steganography

The word "steganography" can be divided into two parts: stegano + graphy. "Stegano" comes from the Greek word "steganos" meaning "covered" and "graphy" which comes from the Greek word "graphein" meaning "writing:" Thus, steganography literally means "covered writing" [9].

Steganography in its definition is the art of covering messages in a secret way in which only the sender and receiver can reveal the hidden message. Generally, the data is hidden inside an image so that even if a third party discovers the image, there won't be any suspicious about the image.

3.1.Digital Steganography Methods

This paper is dedicated to steganography in image files "Cover Image", image based steganography methods varies and have evolved over the years in its embedding methods [10] along with its detections methods [11].

There are many different methods to hide information inside of images, such as "

- Discrete Cosine Transform (DCT)
- Discrete Wavelet Transform (DWT)
- Least Significant Bit (LSB)
- Discrete Fourier Transform (DFT)

Our primary focus will be on Least Significant Bit (LSB), one notable note worth mentioning is that any used methods should take into consideration the following points:

1. Size of information that can be hidden inside the cover

2. Perceptual transparency this means that each cover has certain information hiding capacity. If data was loaded into the cover, then it results in degradation of the cover then this will differentiation in original and infected cover will be noted
3. Robustness of the hidden message to remain undamaged.
4. Tamper resistance makes it difficult for the attacker or pirates to alter the original

Steganography involves in two fragments of data:

- the cover which works as a medium, and
- Data to be hidden, data which is in this paper is the exploit or malicious code.

3.2.Least Significant Bit (LSB) Method

The LSB method works by replacing some of the information in a given pixel with information from the data in the image. While it is possible to embed data into an image on any bit-plane, LSB embedding is performed on the least significant bit(s). This minimizes the variation in colors that the embedding creates [12]

The Least Significant Bit (LSB) method is the most common steganography method in which the secret information is hidden in the least significant bits of the image.

Images can be presented in one of the following digital representations

1. 24-bit color: every pixel can have one in 2^24 colors, represented in basic colors (R, G, B): red (R), green (G), blue (B).
2. 8-bit color: every pixel can have one in 256 (2^8) colors,
3. 8-bit grayscale: every pixel can have one in 256 (2^8) shades of gray

In a simple example, the letter 'A' has an ASCII code of 65, which is *1000001* in binary. The below binary is a representation of a pixels “bit layers” before insertion of data:

Table 1 pixels before data insertion

Pixel 1 =	*10011100*	Pixel 5 =	*11000011*
Pixel 2 =	*10110101*	Pixel 6 =	*10110111*
Pixel 3 =	*10000101*	Pixel 7 =	*10000111*
Pixel 4 =	*10110101*	Pixel 8 =	*10110001*

The values after the insertion of an 'A' letter will be:

Table 2pixels after data insertion

Pixel 1 =	*1001110**1***	Pixel 5 =	*1100001**0***
Pixel 2 =	*1011010**0***	Pixel 6 =	*1011011**0***
Pixel 3 =	*1000010**0***	Pixel 7 =	*1000011**0***
Pixel 4 =	*1011010**0***	Pixel 8 =	*1011000**1***

The same concept could be applied to the 8-bit and 8-bit grayscale images. This is done by adding very little noise to the original picture, For 24-bit images, this can be extended in some cases to the second LSBs without being noticeable [13]

4. Example of the Steganography Exploit

The exploit used in the research is based on two known Common Vulnerabilities and Exposures (CVE) "*Common Vulnerabilities and Exposures (CVE®) is a dictionary of common names (i.e., CVE Identifiers) for publicly known cybersecurity vulnerabilities*.":

- CVE-2014-0282 :Microsoft Internet Explorer 8 / 9 / 10 - CInput Use-After-Free Crash PoC (MS14-035) [14]
- CVE-2009-2478: Mozilla Firefox 3.5 - (Font tags) Remote Buffer Overflow [15]

Delivering the exploit is done by browser exploit using images by hiding the exploits into cover images using LSB steganography method, this is done using a simple tool Stegosploit tools

[16] by utilizing two components the decoder and the exploit code, the main job for the decoder is to run the exploit code at the target, the below image is the logo for PSUT used as demonstration

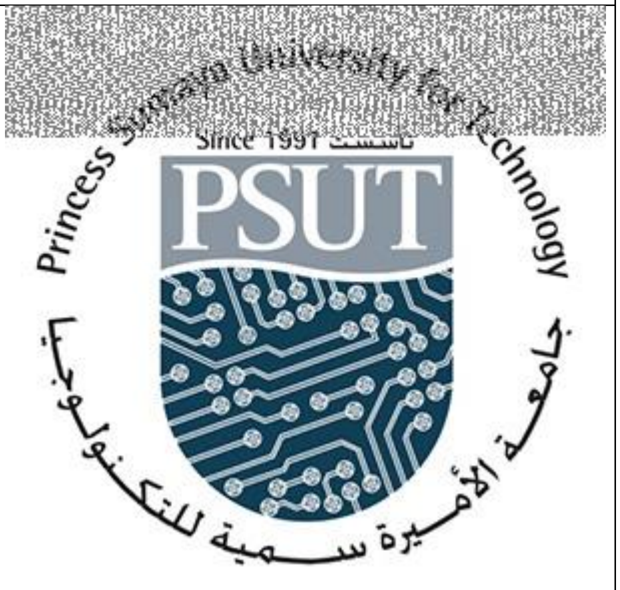
Princess Sumaya University for Technology
Since 1991 تأسست
PSUT
جامعة الأميرة سمية للتكنولوجيا

Figure 1 Cover Used for exploit

As shown in Table 3 images bit layers encoded exploit in different bit layer and as shown in the example using bit layers 0 or 1 are showing less noise or any significant change to the original image as we drill down in embedding the exploit in the image into different layers in the other layer a noise starts to be visible to the eye as shown in the below table of figures with its most visibility at bit layer 7 :

Table 3 images bit layers

Princess Sumaya University for Technology Since 1991 تأسست PSUT جامعة الأميرة سمية للتكنولوجيا	Princess Sumaya University for Technology Since 1991 تأسست PSUT جامعة الأميرة سمية للتكنولوجيا
Bit Layer 0	Bit Layer 1
Princess Sumaya University for Technology Since 1991 تأسست PSUT جامعة الأميرة سمية للتكنولوجيا	Princess Sumaya University for Technology Since 1991 تأسست PSUT جامعة الأميرة سمية للتكنولوجيا
Bit Layer 2	Bit Layer 3

Bit Layer 4	Bit Layer 5
	
Bit Layer 6	Bit Layer 7

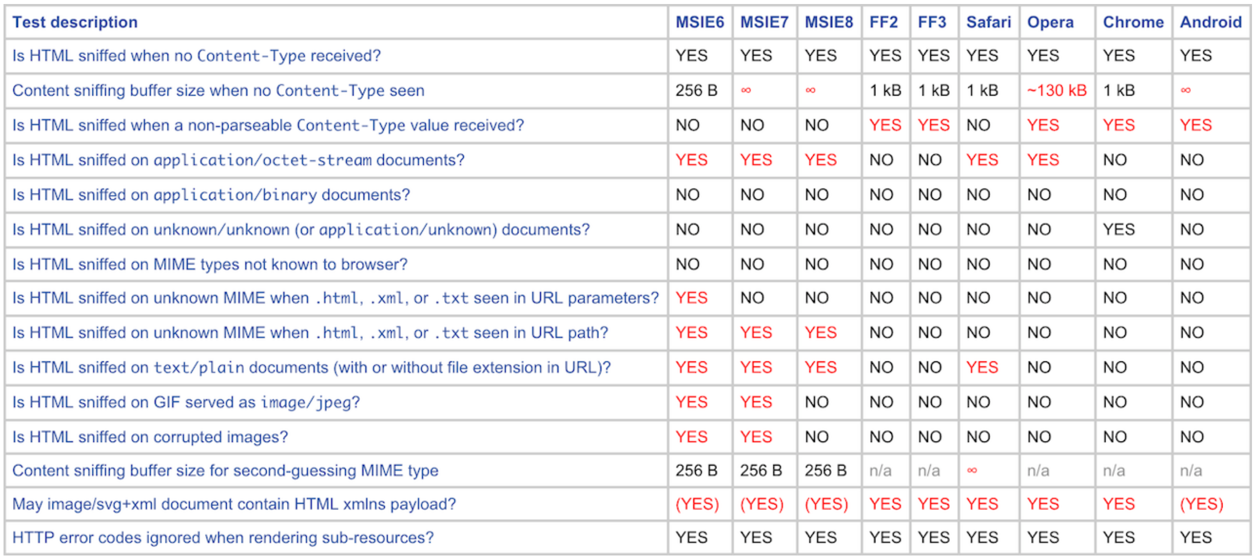
Browsers rely on two key factors for determining the type of content and thereby invoking the appropriate processor or renderer associated with it.

1. Resource extension

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 3 January 2019 doi:10.20944/preprints201901.0030.v1

2. The HTTP Content-Type response header

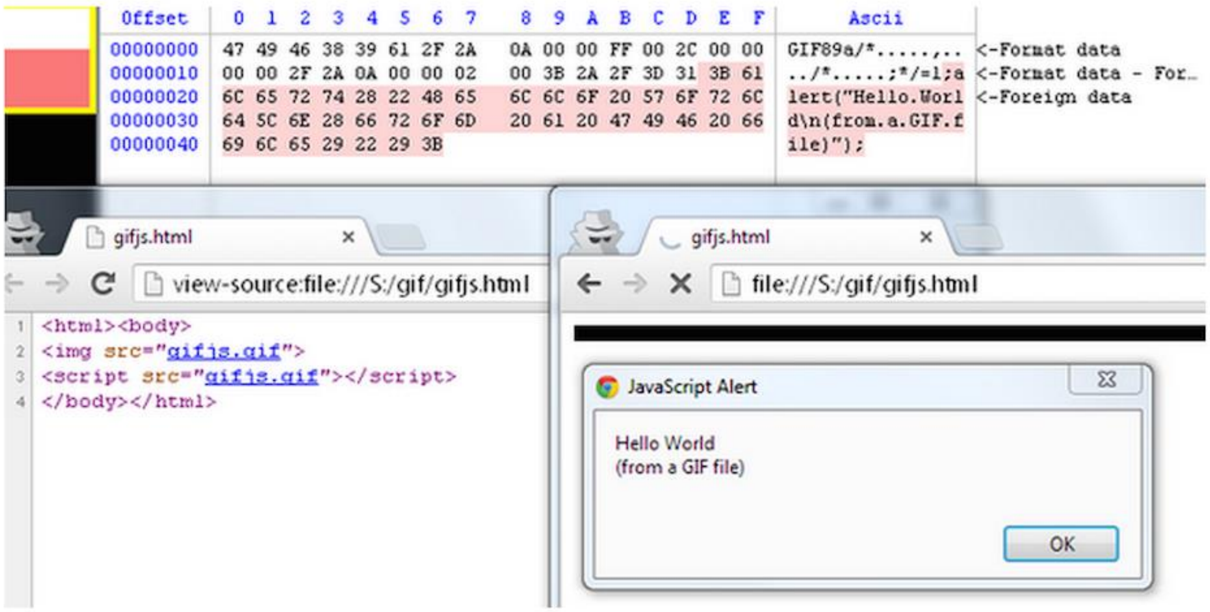
Knowing these two factors give the attacker an advantage to use these weakness in browsers to run the exploit and deliver the malicious code, this is done by omitting the image extension leading the browser to sniff, the below Figure 2 Content Sniffing Matrix shows what content could be sniffed [17] the header content and executing the code inside the image by choosing the appropriate processor for the code chunk its executing [17].



Test description	MSIE6	MSIE7	MSIE8	FF2	FF3	Safari	Opera	Chrome	Android
Is HTML sniffed when no `Content-Type` received?	YES	YES	YES	YES	YES	YES	YES	YES	YES
Content sniffing buffer size when no `Content-Type` seen	256 B	∞	∞	1 kB	1 kB	1 kB	~130 kB	1 kB	∞
Is HTML sniffed when a non-parseable `Content-Type` value received?	NO	NO	NO	YES	YES	NO	YES	YES	YES
Is HTML sniffed on `application/octet-stream` documents?	YES	YES	YES	NO	NO	YES	YES	NO	NO
Is HTML sniffed on `application/binary` documents?	NO	NO	NO	NO	NO	NO	NO	NO	NO
Is HTML sniffed on `unknown/unknown` (or `application/unknown`) documents?	NO	NO	NO	NO	NO	NO	NO	YES	NO
Is HTML sniffed on MIME types not known to browser?	NO	NO	NO	NO	NO	NO	NO	NO	NO
Is HTML sniffed on unknown MIME when `.html`, `.xml`, or `.txt` seen in URL parameters?	YES	NO	NO	NO	NO	NO	NO	NO	NO
Is HTML sniffed on unknown MIME when `.html`, `.xml`, or `.txt` seen in URL path?	YES	YES	YES	NO	NO	NO	NO	NO	NO
Is HTML sniffed on `text/plain` documents (with or without file extension in URL)?	YES	YES	YES	NO	NO	YES	NO	NO	NO
Is HTML sniffed on GIF served as `image/jpeg`?	YES	YES	NO	NO	NO	NO	NO	NO	NO
Is HTML sniffed on corrupted images?	YES	YES	NO	NO	NO	NO	NO	NO	NO
Content sniffing buffer size for second-guessing MIME type	256 B	256 B	256 B	n/a	n/a	∞	n/a	n/a	n/a
May image/svg+xml document contain HTML xmlns payload?	(YES)	(YES)	(YES)	YES	YES	YES	YES	YES	(YES)
HTTP error codes ignored when rendering sub-resources?	YES	YES	YES	YES	YES	YES	YES	YES	YES

Figure 2 Content Sniffing Matrix

As result, we know now that browser will execute code hidden inside an image, a simple image shown in Figure 3 JavaScript in GIF image code sample shows the running JavaScript Code in the browser.



```
Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F   Ascii
00000000  47 49 46 38 39 61 2F 2A  0A 00 00 FF 00 2C 00 00  GIF89a/*.....,..  <-Format data
00000010  00 00 2F 2A 0A 00 00 02  00 3B 2A 2F 3D 31 3B 61  ../*.....;*/=1;a  <-Format data - For…
00000020  6C 65 72 74 28 22 48 65  6C 6C 6F 20 57 6F 72 6C  lert("Hello.Worl  <-Foreign data
00000030  64 5C 6E 28 66 72 6F 6D  20 61 20 47 49 46 20 66  d\n(from.a.GIF.f
00000040  69 6C 65 29 22 29 3B                              ile)");
```

gifjs.html
view-source:file:///S:/gif/gifjs.html
gifjs.html
file:///S:/gif/gifjs.html

```
<html><body>
<img src="gifjs.gif">
<script src="gifjs.gif"></script>
</body></html>
```

JavaScript Alert

Hello World
(from a GIF file)

Figure 3 JavaScript in GIF image

5. Solution

Browser vendors need to start thinking about detecting content before it is rendered in the DOM. This is easier said than done. Server side applications that accept user generated images should currently transcode all received images, for example, transcode a JPG file to a PNG file with slightly degraded quality, and back to JPG. The idea here is to damage any steganographically encoded data [17]

6. Future work

- Define new image policies protocols to render images from other websites
- Re assess the content policy process

doi:10.20944/preprints201901.0030.v1

- Working towards more secure browsers to detected and disable any malicious code.

7. Conclusion

While the full implication of practical exploit delivery via steganography this paper shows that browsers fails in stopping such attacks and the need to have new methods to detect these types of attacks such as behavioral analysis.

8. References

[1] E. Chickowski, "New Malware Found Hiding Inside Image Files," 16 10 2016. [Online]. Available: http://www.darkreading.com/endpoint/new-malware-found-hiding-inside-image-files/d/d-id/1320895.

[2] L. Constantin, "Hackers hide stolen payment card data inside website product images," 18 10 2016. [Online]. Available: http://www.cio.com/article/3132391/hackers-hide-stolen-payment-card-data-inside-website-product-images.html.

[3] D. Goodin, "Millions exposed to malvertising that hid attack code in banner pixels," 12 7 2016. [Online]. Available: http://arstechnica.com/security/2016/12/millions-exposed-to-malvertising-that-hid-attack-code-in-banner-pixels/.

[4] ESET, "Readers of popular websites targeted by stealthy Stegano exploit kit hiding in pixels of malicious ads," 6 12 2016. [Online]. Available: http://www.welivesecurity.com/2016/12/06/readers-popular-websites-targeted-stealthy-stegano-exploit-kit-hiding-pixels-malicious-ads/.

[5] J. Vijayan, "Attack Uses Image Steganography For Stealthy Malware Ops On Instagram," 12 10 2016. [Online]. Available: http://www.darkreading.com/endpoint/attack-uses-image-steganography-for-stealthy-malware-ops-on-instagram/d/d-id/1327170.

[6] A. Vamshi, "Anatomy of a Ransomware Attack: Cyber Uses Steganography to “Hide in Plain Sight”," 30 6 2016. [Online]. Available: https://www.netskope.com/blog/anatomy-ransomware-attack-cerber-uses-steganography-hide-plain-sight/.

[7] J. Buntinx, "Researchers Develop Tool To Embed Malware In Instagram Pictures Through Steganography," 13 10 2016. [Online]. Available: http://themerkle.com/researchers-develop-tool-to-embed-malware-in-instagram-pictures-through-steganography/.

[8] L. Mosuela, "How It Works: Steganography Hides Malware in Image Files," 4 2016. [Online]. Available: https://www.virusbulletin.com/virusbulletin/2016/04/how-it-works-steganography-hides-malware-image-files/.

[9] "steganography," 12 2016. [Online]. Available: http://www.thefreedictionary.com/steganography.

[10] A. Cheddad, J. Condell, K. Curran and P. Levitt, "Digital image steganography: Surveyandanalysisof current methods," *SignalProcessing*, p. 26, 18 8 2009.

[11] K. Bailey and K. Curran, "An evaluation of image based steganography methods using visual inspection and automated detection techniques," *Springer Science*, p. 34, 2006.

[12] G. Kaur and K. Kaur, "Implementing LSB on Image Watermarking Using Text and Image," *nt. J. Adv. Res. Comput. Commun. Eng*, p. 3130–3134, 1 8 2008.

[13] R. K. Thakur and. C. Saravanan, "Analysis of Steganography with Various Bits of LSB for Color Images," *International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, 2016.

[14] D. Liudmila, "Microsoft Internet Explorer 8 / 9 / 10 - CInput Use-After-Free Crash PoC (MS14-035)," 24 6 2014. [Online]. Available: https://www.exploit-db.com/exploits/33860/.

[15] . D. Kennedy , "Mozilla Firefox 3.5 - (Font tags) Remote Buffer Overflow," 13 7 2009. [Online]. Available: https://www.exploit-db.com/exploits/9137/.

[16] S. Shah , "stegosploit," 6 2015. [Online]. Available: http://stegosploit.info/.

[17] B. Park, D. Kim and . D. Shin, "A Study on a Method Protecting a Secure Network against a Hidden Malicious Code in the Image," *Indian Journal of Science and Technology*, 2015.

[18] K. J. Higgins, "Attack Harbors Malware In Images," 8 7 2014. [Online]. Available: http://www.darkreading.com/endpoint/attack-harbors-malware-in-images/d/d-id/1297867.