applied sciences

Article

Framework for Malware Triggering Using Steganography

Lamia Almehmadi [1,*], Abdullah Basuhail [1], Daniyal Alghazzawi [2] and Osama Rabie [2]

[1] Department of Computer Science, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
[2] Department of Information Systems, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia
* Correspondence: lalmehmadi0011@stu.kau.edu.sa

Abstract: Teaching offensive security (ethical hacking) is becoming a required component of information security curricula to develop better cybersecurity practitioners. Many academics and industry professionals believe that a good knowledge of the attacks a system can face is required to protect a system. The early detection of an attack is critical to effectively defending a system. We can't wait for threats to be discovered in the wild to begin planning our defenses. For our study, we designed and developed an offensive model that aims to remain concealed in an image until it reaches the target location. Our attack approach exploits image steganography, which involves embedding malicious code and a geolocation code into a digital image. This study aimed to discover new ways to attack computer systems and stimulate awareness of such attacks among browser developers, thus encouraging them to handle images with more care. In our experiments, both stego-image analysis and geolocation techniques are tested. Our experience has confirmed that converting indiscriminate attacks into targeted attacks is possible.

Keywords: cybersecurity; exploit delivery technique; geolocation; image hacking; malicious code; steganography; Stegosploit

Citation: Almehmadi, L.; Basuhail, A.; Alghazzawi, D.; Rabie, O. Framework for Malware Triggering Using Steganography. *Appl. Sci.* **2022**, *12*, 8176. https://doi.org/10.3390/app12168176

Academic Editor: Luis Javier Garcia Villalba

Received: 17 July 2022
Accepted: 15 August 2022
Published: 16 August 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Offensive security is necessary for the ever-evolving threat landscape to help companies sniff out cracks in their defenses before the bad guys do. New cybersecurity threats are evolving and changing rapidly due to the equally rapid developments in communication and information technologies. Cybercriminals use modern and advanced techniques to improve the speed and scale of their attacks [1]. The Internet has become essential in almost every aspect of life because our vocational, social, and economic lives are increasingly digitized, and our management, health care, military, and bank infrastructures rely on the Internet for their day-to-day operations, malware targets are larger and more lucrative [2]. Online security issues come in many forms, including cyber espionage, malware, identity theft, cyber extortion, phishing, etc. Over the years, hackers have developed new, more advanced, and highly effective methods to carry out their commands [3].

Too many organizations are waiting to be told that they have been compromised. However, the conventional method of "building greater fences" will no longer be adequate given the growing number and size of cyber-attacks and the advanced techniques used by threat actors to conceal their activities [4]. In addition, there is a time lag when a novel malware emerges, or a new version of a current malware before security solutions providers update their customers with the latest signature, a time when the customers are exposed to the new malware [5]. According to the global vaccine testing group AV-Test, 350,000 new malicious pieces of code appear each day. The company discovered that the number of malicious codes rose exponentially from 4.7 million in 2015 to 9.42 million in 2019 [6]. Many businesses, such as antivirus providers, do their utmost to index viruses and produce virus signatures (a virus's unique digital code) so that they can be detected and stopped before they can infect any computers. Despite these attempts to mitigate the effects of malware, malware authors have become even more experts at disguising their malicious code to escape detection protocols.

Hiding content is an integral part of security because it ensures secrecy. However, the techniques used to hide content are also used by hackers, criminals, terrorists, and others for various purposes. The goal is the same, and methods or algorithms designed for covert communication are used for both good and evil purposes [7]. Malicious software can hide its presence on a system or network using steganography and encryption. Steganography's objective, in contrast to encryption, is to conceal both the existence of the information and the content itself. Encryption's sole purpose is to hide the information's content. When encryption is used by malware, its presence is known. With steganography, its existence is hidden due to the nature of its use which makes it difficult to detect and even more challenging to understand [8]. The use of digital steganography to conceal harmful programs and circumvent the potential of computer security software to identify them is a growing trend that makes detecting malware increasingly difficult. Kaspersky Lab researchers noticed this disturbing trend in 2017, noting that the use of steganography is growing among both cybercrime organizations and individual cybercriminals performing cyber espionage [9]. Using steganography to hide information in digital images is also becoming more mainstream. Furthermore, the widespread use of photo-sharing applications, like Instagram and Snapchat, as well as sharing photographs on the Internet, has created an opportunity for the easy implementation of steganography and the preservation of its stealth [8,10].

A new threat, known as "Stegosploit," has recently emerged. Stegosploit is an attack that uses steganography to exploit a vulnerability [11]. Stegosploit has created a new method of encoding malicious code and delivering them via image files to perform browser exploits. The image payloads are undetectable and invisible. It is a toolkit developed by Saumil Shah, an advanced exploit delivery technique that outperforms existing approaches [12].

Trigger-based malware is programmed to remain dormant and undetected until a particular trigger event occurs. It is executed only if a specific external stimulus is applied to it. Environmental parameters usually trigger targeted malware so that it only executes on devices matching a known target environment, such as when a specific moment in time is reached, or the physical or logical attributes of the client become appropriate [13]. In this study, we used geolocation information in our attack to target a specific group of users and increase the likelihood of success by limiting our exposure to unnecessary targets that could potentially discover us.

Many organizations are looking for ways to defend against widespread attacks without trying to protect themselves by unleashing offensive cyber techniques to detect advanced adversaries and gain intel on attackers and how they're trying to penetrate their systems to develop their defenses. Therefore, it is essential to research the evolution of technology and attacks to detect and forecast new threats and stay ahead of cybercriminals. That motivated us to explore whether we could develop a class of attacks against browsers by using only one image to infect a specific geographic location.

We can summarize our contributions through this paper as follows:

This paper proposes an approach to carry out a class of attacks against vulnerable browsers using images and geographic location information to discover new ways to attack computer systems, highlighting the security ramifications of the images and stimulating awareness of such attacks among browser developers. To carry out the study, we first surveyed the various methods of geolocation services and investigated their performance using the accuracy in meters. We then analyzed their accuracy in the context of targeted attacks. Furthermore, we determined whether the device, browser, and access network all impacted the performance of specific positioning methods. The geolocation experiments were performed using three technologies, W3C geolocation, eight IP geolocation services, and two WI-FI positioning services. Then we used steganography to conceal malicious code and appropriate geolocation code within an innocent-looking image file. The malicious payload would only be unlocked if the location target was reached. Finally, we evaluate various aspects of the stego-image created for this attack using MSE, SSIM, and PSNR.

The rest of this paper is organized as follows: Section 2 contains the basic concepts used in our research and the history of steganography and provides a thorough introduction to Stegosploit, which was used to deliver the JavaScript browser exploit via images. The methodology is shown in Section 3. Section 4 presents the experimental results and analysis for the geolocation methods and stego-image. Section 5 discusses the key findings. The conclusion to this paper is found in Section 6.

2. Background and Related Work

2.1. Exploit Delivery Techniques

The term "malware" is taken from the combination of two words, malicious and software, and denotes any to unwanted software. It was described in [14] as "any code added, changed, or removed from a software system to intentionally cause harm or subvert the intended function of the system" [15]. A targeted attack is an attack that targets a particular user, organization, or subgroup of a larger group [16]. According to [12], there are three stages in exploit delivery techniques: contact, redirect, exploit and infect. The contact stage is the first stage of the exploit delivery technique. The redirect stage is the stage of the exploit delivery technique in which victims are screened by the exploit kit based on specific conditions. Exploit and infect is the final stage of the exploit delivery technology.

2.2. Steganography

Steganography transmits hidden messages by building a covert channel, using various carriers such as audio, text, picture, and video [17]. Attackers can use it to exchange messages without any agency being aware of their activities, but it is increasingly used to make stealthy malware [18]. Figure 1 depicts the steganographic technique [19]. According to [20], the Least Significant Bit (LSB) steganography tool is one of the methods employed by attackers in Advanced Persistent Threats (APT). It is employed in numerous phases of the APT attack methodology since it may be used for infiltration during the initial stage as well as in the final stages of data extraction.

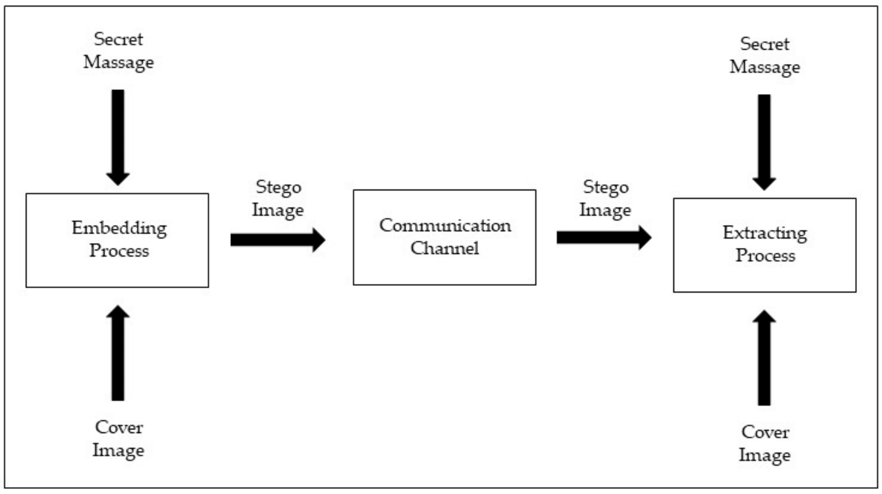


Figure 1. Block diagram of steganography.

2.3. Stegosploit

Stegosploit, introduced by Saumil Shah in Black Hat Europe, was a new threat in 2015 [21]. It used images as hosts for malicious payloads used to target vulnerabilities and exploit them. It includes two subfields: steganography and polyglot. Shah implemented his steganography system using a function of HTML5 called CANVAS. He defined polyglot as a mixture of HTML and image, which is manipulated using JavaScript, HTML, and image header modification [11]. The malicious code begins the execution of the intended attack on the client system when the image is downloaded from the compromised websites or node to the client [12].

2.4. Related Work

Several methods of hiding data, such as obfuscation, anonymization, encryption, and steganography, are used by sophisticated malware. Among them, steganography is widely seen as one of the most effective methods for malware to conceal data. This is because steganography, unlike other methods, masks the very fact that any contact is taking place, which helps the malware avoid detection. Initially, steganography methods used by malware were used in advanced persistent threats (APT) [22].

Steganography malware has used the images to hide malware settings and configuration files, provide a URL from which the malware may import additional elements, or store the malicious code directly [22]. Due to the large number of image files uploaded daily to the Internet, it has become a popular file for data embedding. There are several techniques to embed the hidden data in the image files [8]. The first recorded exploit delivery technique that used information hiding techniques was identified in 2006. However, it was not detected until 2011 [12]. Many examples of malware in the wild that use images are summarized in Table A1.

In [10], a new method for attacking smartphone mobile browsers by using QR codes to share images encoded with a JavaScript exploit code is presented. A PNG wallpaper image of a dog was modified and then shared via QR code to exploit the weakness of the Opera browser in the Android system to reveal the cookie file.

An APT attack scheme that uses image steganography to embed malicious code in a digital image is presented in [11]. They used an extractor program to trigger the stego-image. Then they looked at the stego-image and extractor to search for patterns. The malicious payload is difficult to detect because it is hidden in the pixels of the images. They proposed a steganography-based 5-phase attack scheme. During the preparation phase, they generated the stego-image using the LSB algorithm and used CVE-2017-7494 [23] on the target to remote code execution and modify its system task scheduler to set the trigger. They delivered the stego-image and extractor to the target Samba server on an ubuntu-16.04.3-server-amd64 OS during the implantation phase by uploading them to the server's shared folder. During the inactivation phase, the trigger will detect whether the stego-image is present in the device at a specific time; if not, it will wait for the next detection time; if it is present, the extraction will be performed. A malicious payload will be extracted and then executed by the task scheduler to complete a connection request during the extraction phase. Following the establishment of the connection, the malicious payload will send a message to the binary that contains information about the victim's device to gain remote control of the target device.

On the other hand, deep neural networks (DNNs) are also vulnerable to adversarial example attacks, as we see in [24]. They proposed a blind-watermark backdoor method whose results are invisible to humans. Their way avoids the human detectability of the backdoor sample attack by making the trigger invisible. They achieved an attack success rate is 99.3% via training with the blind-watermarked samples.

3. Methodology

This section of the paper describes the attack design and implementation. Figure 2 below illustrates the attack architecture. The primary goal of this study was to develop an attack that would exploit image steganography and target a specific location where malicious code is concealed in a digital image. When the stego-image reaches the target location, it will be triggered.

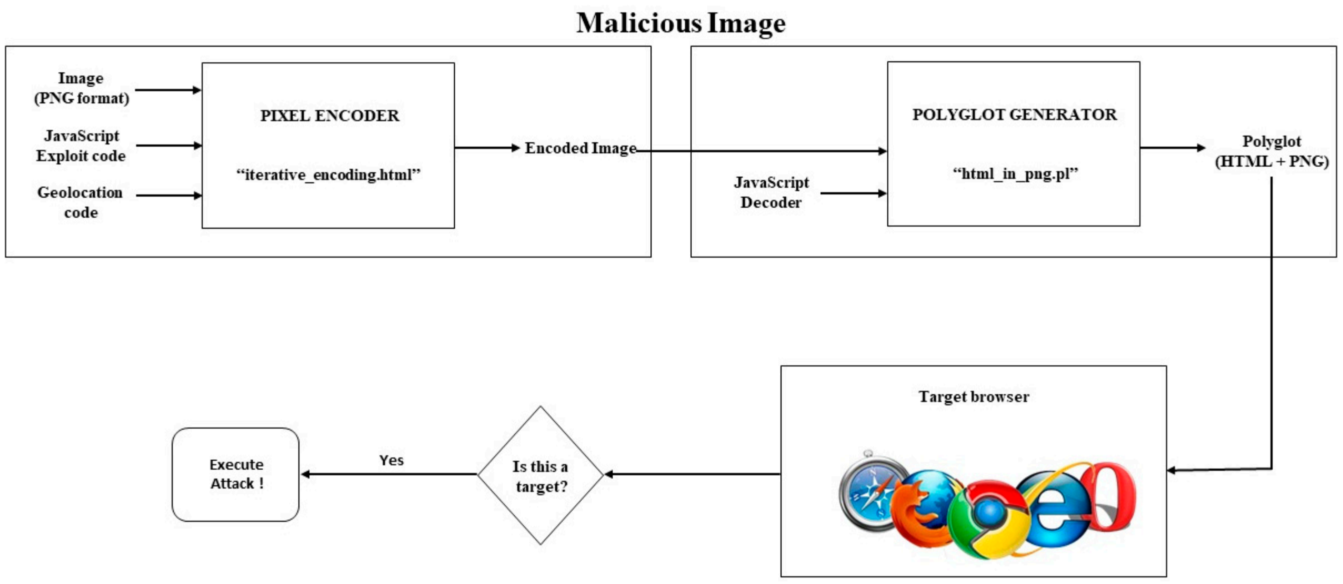


Figure 2. A block diagram of the proposed attack.

Based on Stegosploit, we conducted three attack phases. The contact and exploit stages are the first and last stages of the attack, the same as the stages of the exploit delivery technique. The intermediate stage is the propagation stage, which differs from the redirection stage of the exploit delivery technique [12].

This paper tries to answer these questions: first, may the proposed attack remain stealthy for a long time before reaching its target? Second, may the proposed attack trigger when it reaches the target based on location? Finally, can the image quality be distorted by hiding the malware inside it?

The experiment can be summarized as follows. First, we surveyed the various geolocation services methods and chose the appropriate to encode it with the image in our attack. Then, we used steganography to encode the JavaScript exploit and geolocation code into a PNG image. In addition, an HTML + PNG polyglot file was created using the Stegosploit toolkit. Next, we uploaded the polyglot image into a vulnerable browser. The malicious code will start running immediately when the target opens the image. The intended attack was carried out on the target system based on location.

The general procedure for carrying out the above attack is described in detail in the following sections.

3.1. Geolocation Techniques

First, we discovered the geolocation detection techniques used by websites to estimate the location of visiting users and use them in our attack. We thought like attackers; therefore, we looked for quick ways to discover the user's location, where delay and resource costs were unacceptable.

3.1.1. W3C Geolocation API Method

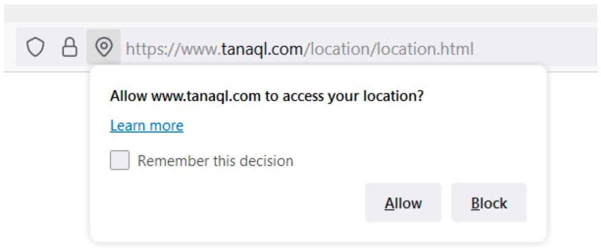
The W3C Geolocation API defines three methods as its predecessors Google Gears and Mozilla Geode [25–29]. Once we verify that the browser supports the W3C Geolocation API, queries can be made for the device's current location. In our work, we only needed to get the device's current location. As a result, we used the getCurrentPosition() method. It takes three arguments. If the attempt is successful, the successCallback argument with a parameter object of Position must be called. If the attempt fails, the errorCallback argument with a PositionError object must be called. The third argument is PositionOptions. It has the enableHighAccuracy, timeout, and maximumAge attributes. For the best results, the measurement in JavaScript was performed with high geolocation accuracy (attribute "enableHighAccuracy" set to "true") and (attribute "maximumAge" set to "0") to obtain a new position each time, with a 60 s timeout period before returning an error.

The Position object contains all the geolocation data supplied by this API request and is delivered to a successCallback function. This version of the standard supports one Coordinates attribute and a timestamp. The coords attribute is a Coordinates object that provides a collection of geographic coordinates with associated precision and other properties. Table 1 contains a list of the properties found in the Coordinates object. The timestamp attribute indicates the time when the Position object was obtained. The PositionError object contains all error information returned by the API call and is passed to an errorCallback function.

Table 1. Coordinates object properties.

Property	Description
Latitude	The device's geographic latitude coordinate is measured in decimal degrees.
Longitude	The device's geographic longitude coordinate is measured in decimal degrees.
Altitude	The device's geographic height was measured meters above the WGS 84 ellipsoid.
Accuracy	The accuracy of the latitude and longitude coordinates in meters. It must be supported by all implementations.
AltitudeAccuracy	The accuracy of the height (altitude coordinate), is specified in meters.
Heading	The device's direction of travel, measured in degrees, where $0° \leq$ heading $< 360°$, counting clockwise relative to true north.
Speed	The device's current ground speed is measured in meters per second.

A permission request to report the current location is displayed when using this method of browser geolocation, as shown in Figure 3 below.


https://www.tanaql.com/location/location.html

Allow www.tanaql.com to access your location?

Learn more

☐ Remember this decision

Allow Block

Figure 3. The permission in the W3C Geolocation API.

3.1.2. IP Geolocation Services

IP geolocation is the process of determining the geographical location of a specific IP address. For this study, we investigated the accuracy of eight IP geolocation services: DB-IP, Maxmind GeoIP2, IP2Location, Ipinfo, Skyhook Hyperlocal IP, Google Geolocation API, Ipregistry, and Whois XML API. These services usually provide country, region, city, zipcode, latitude/longitude, ISP, domain or company name, time zone, and even connection type.

3.1.3. Wi-Fi Positioning Systems

Wi-Fi positioning systems (WPS) [30] identify the location by using terrestrial-based wi-fi access points (APs). It detects existing wi-fi signals in the vicinity of a wi-fi-enabled mobile device and calculates the device's current location. Wi-fi positioning does not require the establishment of a wi-fi network connection. Several commercial wi-fi positioning systems have been developed. Of the various commercial systems, Skyhook [31] and Google's geolocation API [32] were chosen for our work. Since its inception, the Skyhook system has been the longest-running and has had the most widespread adoption on many devices. The Google system has also provided good accuracy and comprehensive coverage in several studies.

To discover and collect nearby access points (wi-fi signals), we used the node-wifi-scanner utility. Scan results, also known as fingerprints, include a list of APs and their corresponding received signal strength information, where the signal strength is measured in dBm. Once we have identified the wireless access points, we will send a POST request to a wireless location service using fetch. The request body must be in JSON format (JavaScript Object Notation). There is an attribute called wi-fiAccessPoints in the body of the POST request that accepts an array of objects and must have two or more media access control (MAC) addresses. When the service is successful, it returns a location coordinate (latitude and longitude) and the accuracy in meters. If Google's WPS service fails to geolocate, based on the MAC address data of the access point, it returns an IP geolocation result. Therefore, when Google returns IP responses, we consider this a failure. Although the service does not explicitly indicate an error, any responses based on IP geolocation can be identified by comparing them to a query with no AP MAC inputs or setting the "considerIp" object to false to disable IP geolocation. If Skyhook cannot determine a device location, it returns a specific "location not found" message.

3.2. Hiding Javascript Exploit Code and Geolocation Code in Image Pixels

The exploit code, with the geolocation code, is implanted in a PNG image with the Stegosploit package's Pixel Encoder tool (iterative_encoding.html) to obtain the encoded image. To embed the code in the image, the stegosploit tool employs Least Significant Bit (LSB) Steganography. This does not change the color sufficiently to enable detection by the human eye. The LSB approach is one of the simplest and most common techniques used in the spatial domain. It conceals the harmful code in the least significant bits of the image's pixels, making the distortions caused by the insertion process imperceptible.

An image is made of an array of pixels. Each pixel can have three color channels (red, green, and blue), and each channel is made up of one byte, an 8-bit value that produces 256 discrete levels of color, and a grayscale image with one black color channel [32]. Figure 4 shows an original image and its bit plane.

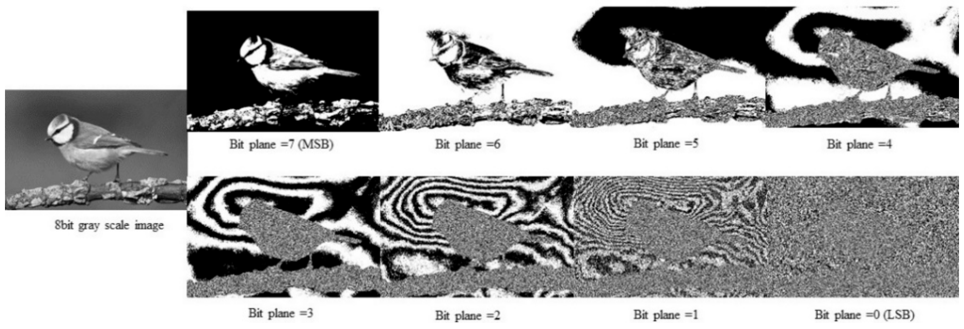


Figure 4. An 8-bit greyscale image's bit plane.

The "iterative_ncoding.html" tool lets us select the bit layer (0 to 7), grid size, and channel (red, green, blue, or grayscale) to use for the encoding purpose. This tool allows us to process any bit layer other than the LSB layer. Still, it is suggested that bit layers 0, 1, and 2 are the most suitable for steganography because this creates no optical aberration in the image. Our experiment implanted the code at bit layer 2 with a 3×3 pixel grid size and the grayscale channel. An HTML5 Canvas was used to perform the steganographic encoding in the browser. The resulting image, with the bitstream encoded on layer 2, exhibits no visual aberration, even when extremely expanded. It should be noted that the exploit code is not present in the encrypted image as strings because the exploit code is steganographically encoded in the image pixels themselves.

3.3. Creating HTML + PNG Polyglot

The Polyglot Generator (html_in_png.pl) tool of the Stegosploit package is used to generate an HTML + PNG polyglot file by combining the encoded image generated above with the JavaScript decoder. The JavaScript decoder in [33] has been modified to launch the exploit immediately after the page has been loaded and without any user interaction.

The JavaScript decoder executes the reverse function of the encoder. The script requires three global variables representing the bit layer, encoding channel, and pixel grid. They must have the same values as the variables used in the encoding process. On the target system, the decoder reconstructs the exploit code bitstream from the pixel values in the encoded bit layer. From the decoded bitstream, the exploit code is reassembled into JavaScript code. The decoder then determines the type of exploit code obtained and executes it as JavaScript in the browser. The browser will then be compromised if it is vulnerable.

3.4. Exploit Delivery

In developing this exploit, we have achieved the following objectives: The image displayed in the browser must not contain visual aberration or distortion, the image must decrypt to the exploit code when loaded into the browser without external user intervention, and only one image should be used for this exploit.

As the attacker, we applied the following options for delivering the polyglot to the victim's browser: Host the image on a web server controlled by the attacker, which is the most common of all HTTP delivery techniques and send its URL to the victim. As an alternative, we could upload the image to web applications such as blogs, bulletin boards, and document-sharing platforms and provide direct links to it. Images are always accepted in such applications because they do not damage the integrity of the web application.

The malicious code that is to be executed in the target system is written in JavaScript. Attacks can be minor, such as closing the browser window in which the image is being loaded, or major, such as stealing the cookies or current session information, i.e., session hijacking. JavaScript code can endanger privacy by starting any executable file, which might be a keylogger or a process for taking a screenshot of the current window and sending it to a different host. Finally, the malicious code can also harm the system by deleting any file that might be necessary for its proper operation, making this method fatal in terms of data security [12].

4. Evaluation

The experiments were carried out and the results obtained are displayed and analyzed in this section. The geolocation experiments were performed using three technologies, W3C geolocation, database-based IP geolocation, and WI-FI positioning services. We also evaluate various aspects of the stego-image created for this attack.

4.1. Experiment Environment

The following describes the environment in which the experiments were carried out: We employed four devices for geolocation API experiments: a Lenovo laptop with Windows 10, an Apple 11 Pro Max, a Mac laptop, and a Huawei MetaPad running Android. Table 2 shows the characteristics of the operating systems and web applications used in the tests.

4.2. Results and Analysis

The suggested scheme's two primary components are the geolocation condition (trigger) and the stego-image. First, we compared the accuracy of the different geolocation systems used for evaluation based on the devices, browsers, and access network types. Then, to describe the features of stego-images, we analyzed the Mean Square Error (MSE), Structured Similarity Index Measure (SSIM), and Peak Signal to Noise Ratio (PSNR).

4.2.1. Geolocation Services Result and Analysis

This section summarizes the location estimate results and our analysis for each geolocation technique: the W3C Geolocation API, the IP geolocation, and the wi-fi positioning systems.

Table 2. The characteristics of the devices and web applications.

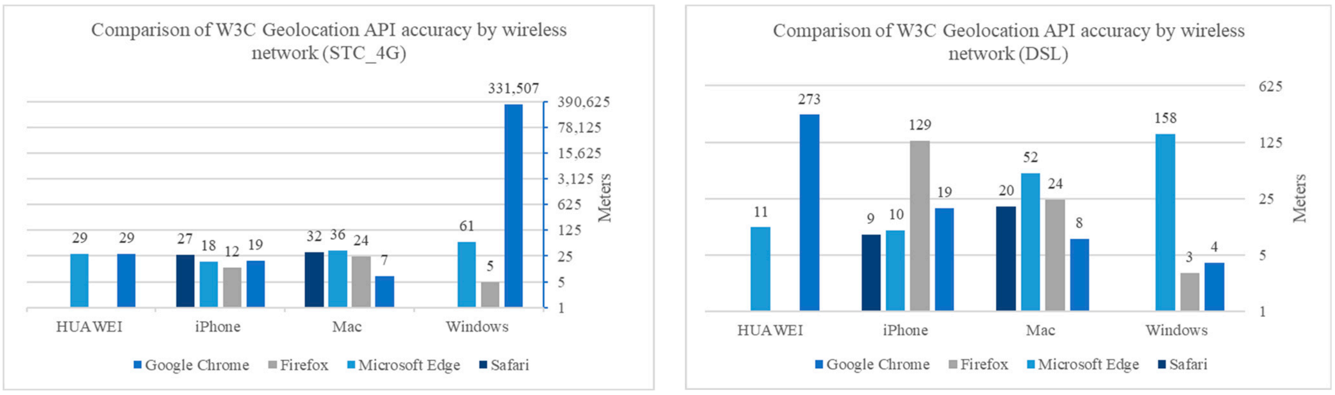
OS Name	OS Version	Browser	Browser Version
Windows	10	Google Chrome	98.0.4758.102
		Firefox	95.0.2
		Microsoft Edge	98.0.1108.62
MacBook Air	11.1	Google Chrome	98.0.4758.102
		Firefox	97.0.1
		Microsoft Edge	98.0.1108.62
		Safari	14.0.2
iPhone	11 pro max	Google Chrome	98.0.4758.97
		Firefox Daylight	97.0
		Microsoft Edge	98.0.1108.62
		Safari	15
Android	10	Google Chrome	91.0.4472.134
		Microsoft Edge	95.0.1020.55

The comparative analysis of location services contains the results of evaluating two parameters. These are as follows:

- The first parameter is the precision of each service. In our study, the precision attribute refers to the level of accuracy of the latitude and longitude coordinates estimated by the service. It is specified in meters and must be a non-negative real number;
- The second parameter is accuracy, defined as the difference between measured coordinates (longitude and latitude) and the device's actual position during the measurement. This difference in meters was calculated using Python scripts, and it was validated using the distance calculator tool [34].

W3C Geolocation API

The comparison analysis of the W3C geolocation method was performed on the four devices identified earlier, with different browsers and using two access networks. The results are summarized in Figure 5.



(**a**) (**b**)

Figure 5. Comparison analysis of the W3C geolocation API method. (**a**) Comparison of W3C Geolocation API accuracy by wireless network (STC_4G); (**b**) Comparison of W3C Geolocation API accuracy by wireless network (DSL).

From the measurements of the W3C geolocation method, when using the Windows device, the following results were obtained, as shown in Tables 3 and 4: We noticed a significant difference in location accuracy when using the Chrome browser due to the access technology, i.e., the accuracy is much better (about 3 m) for the DSL network than for the data SIM card. The Firefox browser had the most accurate location results with an average accuracy of 3.95 m, but the Microsoft Edge browser also had reasonably good results in locating the device with an average accuracy of 109.25 m. In contrast, the Google Chrome browser provided the worst value and was not acceptable, with an average accuracy of 165,755.35 m.

Table 3. Result and Comparison of W3C Geolocation API in Windows by a wireless network (STC_4G).

Browser	Coordinates	Analysis (Meters)
Google Chrome	Latitude: 21.485811 Longitude:39.1925048 Timestamp: 3 February 2022, 8:42:42 p.m.	Precision: 28,543.920874903582 Accuracy: 331,506.93051
Firefox	Latitude: 24.4509863 Longitude:39.5309932 Timestamp: 3 February 2022, 8:43:06 p.m.	Precision: 21.361 Accuracy: 5.1024715743
Microsoft Edge	Latitude: 24.450438 Longitude: 39.530746 Timestamp: 3 February 2022, 8:42:11 p.m.	Precision: 36 Accuracy: 61.040755015

Table 4. Result and Comparison of W3C Geolocation API in Windows by a wireless network (DSL).

Browser	Coordinates	Analysis (Meters)
Google Chrome	Latitude: 24.4509544 Longitude:39.5309543 Timestamp: 3 February 2022, 8:44:24 p.m.	Precision: 18.388 Accuracy: 3.7948552115
Firefox	Latitude: 24.450959 Longitude:39.5309692 Timestamp: 3 February 2022, 8:44:02 p.m.	Precision: 17.154 Accuracy: 2.8352362323
Microsoft Edge	Latitude: 24.452347 Longitude: 39.531178 Timestamp: 3 February 2022, 8:45:25 p.m.	Precision: 389 Accuracy: 157.54772225

The comparative analysis of the W3C geolocation method using the Mac laptop is summarized in Tables 5 and 6.

Table 5. Result and Comparison of W3C Geolocation API in Mac by a wireless network (STC_4G).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.4509405 Longitude:39.5309176 Timestamp: 3 February 2022, 9:03:23 p.m.	–	Precision: 17.649 Accuracy: 7.1872030392
Firefox	Latitude: 24.451001562548395 Longitude: 39.53075815429659 Timestamp: 3 February 2022, 9:03:48 p.m.	Altitude: 659.213134765625 Altitude Accuracy: 10	Precision: 65 Accuracy: 24.290212277
Microsoft Edge	Latitude: 24.450659 Longitude: 39.530815 Timestamp: 3 February 2022, 9:02:59 p.m.	–	Precision: 30 Accuracy: 35.904529177
Safari	Latitude: 24.4507518426389 33 Longitude: 39.53074760137734 Timestamp: 3 February 2022, 9:02:25 p.m.	–	Precision: 65 Accuracy: 32.179697398

Table 6. Result and Comparison of W3C Geolocation API in Mac by a wireless network (DSL).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.4509419 Longitude:39.5309072 Timestamp: 3 February 2022, 9:05:46 p.m.	–	Precision: 19.463 Accuracy: 8.2404029963
Firefox	Latitude: 24.45091764820717 Longitude: 39.53075805646881 Timestamp: 3 February 2022, 9:06:31 p.m.	Altitude: 659.0762939453125 Altitude Accuracy: 10	Precision: 65 Accuracy: 23.474394577
Microsoft Edge	Latitude: 24.450583 Longitude: 39.530661 Timestamp: 3 February 2022, 9:06:10 p.m.	–	Precision: 213 Accuracy: 51.773552946
Safari	Latitude: 24.451066477243643 Longitude: 39.53085197504793 Timestamp: 3 February 2022, 9:05:15 p.m.	–	Precision: 65 Accuracy: 19.675868197

Comparing Google Chrome, Firefox, Microsoft Edge, and Safari, we found that Google Chrome, with an average accuracy of 7.7 m, had greater accuracy than Firefox, and Firefox had better accuracy than Safari. The Firefox browser has an average accuracy of 23.9 m, and Safari has an average accuracy of 25.95 m. Microsoft Edge had the worst performance compared to the other browsers, with an average accuracy of 43.85 m.

Using the iPhone, the measurements of the W3C geolocation method obtained the following results (summarized in Tables 7 and 8): iPhone browsers have the most accurate location results with an average accuracy of fewer than 20 m, and the Firefox browser had the most accurate location results with an average accuracy of 12.15 m. The Microsoft Edge browser obtained a result close to the value of the Firefox browser with an average accuracy of 13.9 m. The Safari browser also had reasonably good results, locating the device with an average accuracy of 17.95 m. Google's Chrome also got a very good value, close to the accuracy of the Safari browser, with an average accuracy of 18.6 m.

Table 7. Result and Comparison of W3C Geolocation API in iPhone by a wireless network (STC_4G).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.45109948924414 Longitude: 39.53092912346311 Timestamp: 3 February 2022, 8:52:37 p.m.	Altitude: 656.5878155622631 Altitude Accuracy: 7.636130020708702 Speed: 0.1448562741279602	Precision: 58.37975215431486a Accuracy: 18.664453549
Firefox	Latitude: 24.451034831662177 Longitude: 39.530928998013366 Timestamp: 3 February 2022, 8:53:08 p.m.	Altitude: 656.5729167815927 Altitude Accuracy: 6.576576465643781 Heading: 84.16177075330282 Speed: 0.19107863306999207	Precision: 21.269646468211825 Accuracy: 12.090165558
Microsoft Edge	Latitude: 24.450894677621474 Longitude: 39.53081615829184 Timestamp: 3 February 2022, 8:53:17 p.m.	Altitude: 655.1706981658936 Altitude Accuracy: 19.12166404724121	Precision: 35 Accuracy: 18.18586397
Safari	Latitude: 24.45083244979642 Longitude: 39.53074778166829 Timestamp: 3 February 2022, 8:51:08 p.m.	Altitude: 656.4743852615356 Altitude Accuracy: 16.95233726501465	Precision: 35 Accuracy: 27.179880045

Table 8. Result and Comparison of W3C Geolocation API in iPhone by a wireless network (DSL).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.450914035324114 Longitude: 39.530808338863366 Timestamp: 3 February 2022, 8:55:53 p.m.	Altitude: 656.5838843896648 Altitude Accuracy: 10.476290042817674 Heading: 328.4559806069246 Speed: 0.41548725962638855	Precision: 23.890414647506027 Accuracy: 18.483235082
Firefox	Latitude: 24.450990773085895 Longitude: 39.53088104032759 Timestamp: 3 February 2022, 8:56:07 p.m.	Altitude: 656.5745322359726 Altitude Accuracy: 7.688099688915077 Speed: 0	Precision: 20.335977974761448 Accuracy: 12.233179262
Microsoft Edge	Latitude: 24.450929117125384 Longitude: 39.530894994979825 Timestamp: 3 February 2022, 8:56:27 p.m.	Altitude: 656.5590492952482 Altitude Accuracy: 7.033216356919276 Heading: 171.32757269542145 Speed: 0.13296207785606384	Precision: 16.74236877839259 Accuracy: 9.5602362843
Safari	Latitude: 24.45094193472298 Longitude: 39.53090227038583 Timestamp: 3 February 2022, 8:54:43 p.m.	Altitude: 656.638206269592 Altitude Accuracy: 6.7752706275769725 Speed: 0.1334974616765976	Precision: 13.025194175201719 Accuracy: 8.7387849661

Comparison and analysis of the W3C geolocation method when using the Huawei Metapad is summarized in Tables 9 and 10.

Table 9. Result and Comparison of W3C Geolocation API in HUAWEI by a wireless network (STC_4G).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.4511825 Longitude: 39.530874 Timestamp: 3 February 2022, 8:47:06 p.m.	–	Precision: 37.5 Accuracy: 29.291476471
Microsoft Edge	Latitude: 24.4511825 Longitude: 39.530874 Timestamp: 3 February 2022, 8:47:36 p.m.	–	Precision: 37.5 Accuracy: 29.291476471

Table 10. Result and Comparison of W3C Geolocation API in HUAWEI by a wireless network (DSL).

Browser	Coordinates	Other Information	Analysis (Meters)
Google Chrome	Latitude: 24.452440000000003 Longitude: 39.52885166666667 Timestamp: 3 February 2022, 8:48:52 p.m.	Altitude: 1226.2 Heading: 0 Speed: 0	Precision: 11.199999809265137 Accuracy: 273.08956088
Microsoft Edge	Latitude: 24.451038 Longitude: 39.531013 Timestamp: 3 February 2022, 8:49:35 p.m.	–	Precision: 35 Accuracy: 11.108112468

Comparing Google Chrome and Microsoft Edge, we found that Microsoft Edge had greater accuracy than Chrome, with an average accuracy of 20.2 m, while Chrome had an average accuracy of 151.2 m.

Figure 5 summarizes the results in the tables for the W3C geolocation API. We noticed some critical points: iPhone browsers have the most accurate location results, with an average accuracy of fewer than 20 m.

IP Geolocation

We evaluated location accuracy for eight IP services' databases in terms of correctly estimated countries, regions, and cities to assess the accuracy of IP geolocation. Tables 11 and 12 depict the results.

Table 11. Database-based IP geolocation by a wireless network (STC_4G).

IP	Coordinates	Other Information	Analysis (Meters)
DB-IP	"latitude": 24.7136 "longitude": 46.6753	"city": "Riyadh" "region": "Riyadh "country": "Saudi Arabia"	Accuracy: 722,882.94761
Maxmind GeoIP2	"latitude": 21.5168 "longitude": 39.2192	"city": " Jeddah" "region": " Mecca Region "country": "Saudi Arabia, Asia"	Precision Radius: 200 km = 200,000 m Accuracy: 327,802.7072
IP2Location	"latitude": 21.51694 "longitude": 39.21917	"city": " Jeddah" " region": " Makkah al Mukarramah" "country": "Saudi Arabia"	Accuracy: 327,787.51195
Ipinfo	"latitude": 21.2703 "longitude": 40.4158	"city": "Ta'if" "region": "Mecca Region" "country": "SA"	Accuracy: 365,083.7281
Skyhook Hyperlocal IP	"latitude": 21.5168 "longitude": 39.841141	–	Precision:694,224.0 Accuracy: 327,786.44829
Google Geolocation API	"latitude": 24.4678656 "longitude": 39.5804672	–	Precision:9490.098963240705 Accuracy: 5349.6971342
Ipregistry	"latitude": 21.51675 "longitude": 39.21913	"city": "Jeddah" "region": "Makkah al Mukarramah" "country": "Saudi Arabia"	Accuracy: 327,808.9383
Whois XML API	"latitude": 24.68773 "longitude": 46.72185	"city": "Riyadh" "region": "Riyadh Region" "country": "SA"	Accuracy: 727,549.79284

Table 12. Database-based IP geolocation by a wireless network (DSL).

IP	Coordinates	Other Information	Analysis (Meters)
DB-IP	"latitude": 24.4926 "longitude": 39.5857	"city": "Sulţānah" "region": "Medina Region" "country": "Saudi Arabia"	Accuracy: 7218.9267536
Maxmind GeoIP2	"latitude": 24.4662 "longitude": 39.6168	"city": "Medina" "region": "Medina Region" "country": "Saudi Arabia, Asia"	Precision Radius: 200 km = 200,000 m Accuracy: 8849.2794528
IP2Location	"latitude": 24.46861 "longitude": 39.61417	"city": "Medina" "region": "Al Madinah al Munawwarah" "country": "Saudi Arabia"	Accuracy: 8645.0363188
Ipinfo	"latitude": 24.4686 "longitude": 39.6142	"city": "Medina" "region": "Medina Region" "country": "SA"	Accuracy: 8647.7410092
Skyhook Hyperlocal IP	"latitude": 26.45756 "longitude": 38.058708	–	Precision: 0.0 Accuracy: 267,627.79054
Google Geolocation API	"latitude": 24.4714834 "longitude": 39.5329976	–	Precision: 1207.4218113793675 Accuracy: 2293.1773689
Ipregistry	"latitude": 24.46625 "longitude": 39.61681	"city": "Medina" "region": "Al Madinah al Munawwarah" "country": "Saudi Arabia"	Accuracy: 8851.3385909
Whois XML API	"latitude": 24.49258 "longitude": 39.58572	"city": "Sulţānah" "region": "Medina Region" "country": "SA"	Accuracy: 7219.0531998

According to the findings, the databases achieved perfect accuracy at the country level. However, for Skyhook Hyperlocal IP and Google Geolocation API, only the coordinates were returned, not the city, country, or region.

Table 11 shows the results of locating the device using an IP geolocation database when the device is connected to a SIM card wireless network. The findings show that the majority of databases produced incorrect results at the region and city levels. The worst accuracy, which was around 700 km (kilometers), was achieved by the DBIP and Whois databases. The best database was Google, with an accuracy of 5 km. The rest of the databases returned almost all locations within an accuracy range greater than 300 km.

Table 12 below shows the results of locating the device using an IP geolocation database when the device is connected to a DSL wireless network. The results show that most databases produced valid results at the district and city level except for the Skyhook database, which returned coordinates for another city.

The Skyhook database achieved the worst accuracy, which was around 268 km (kilometers). The best database was Google, with an accuracy of 2 km. The rest of the databases achieved convergent accuracy values. The DBIP and Whois databases had an accuracy of about 7 km, while the remainder of the databases returned almost all locations within an accuracy range of 8 km.

Figure 6 summarizes the results in the tables for the IP geolocation services. We noticed that all services were correct when connected to a DSL type of network, except for the Skyhook Hyperlocal IP.

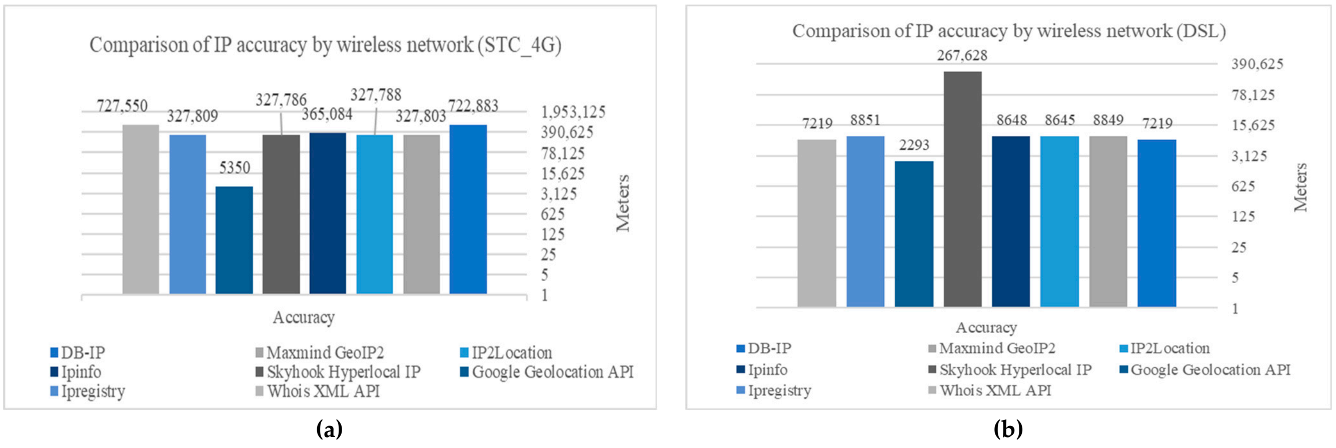


Figure 6. Comparison analysis of the Database-based IP geolocation method. (**a**) Comparison of IP accuracy by wireless network (STC_4G); (**b**) Comparison of IP accuracy by wireless network (DSL).

WI-FI Positioning Services

The comparative analysis of WPS on Windows and Mac devices, using two access networks, is summarized in Figure 7. On the Mac, the number of access points (APs) was lower than on Windows. There are 25 APs for Windows and 11 to 27 for Mac. No relationship was discovered between the number of APs and the device's positional accuracy. However, the results noted in [35] show that wi-fi positioning can be accomplished with a small number of APs (5–10) and that increasing the number of APs does not improve positional accuracy.

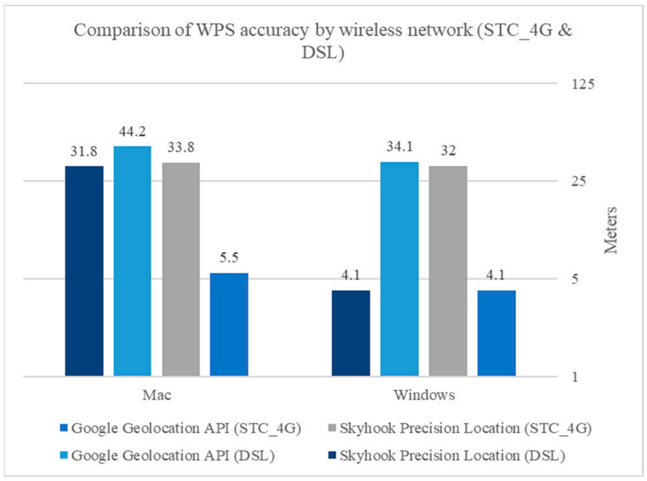


Figure 7. Comparison analysis of the WPS.

The comparison analysis of the WPS in Windows and Mac is summarized in Tables 13 and 14, respectively.

Table 13. Result and Comparison of WPS in Windows.

Service	Access Networks	Coordinates	Other Information	Analysis (Meters)
Google Geolocation API	wireless network (STC_4G)	lat: 24.4509747 lng: 39.530973	–	Precision: 36.312 Accuracy: 4.1073313992
	wireless network (DSL)	lat: 24.4506757 lng: 39.5308182	–	Precision: 58.041 Accuracy: 34.132581461
Skyhook Precision Location	wireless network (STC_4G)	lat: 24.450698 lng: 39.530818	nap: 14 source: 'wifi'	Precision: 35 Accuracy: 32.028255422
	wireless network (DSL)	lat: 24.450904 lng: 39.530993	nap: 18 source: 'wifi'	Precision: 30 Accuracy: 4.0941457106

Table 14. Result and Comparison of WPS in Mac.

Service	Access Networks	Coordinates	Other Information	Analysis (Meters)
Google Geolocation API	wireless network (STC_4G)	lat: 24.4509806 lng: 39.5309566	–	Precision: 27.188 Accuracy: 5.5018890083
	wireless network (DSL)	lat: 24.4506014 lng: 39.5307614	–	Precision: 57.392 Accuracy: 44.173707921
Skyhook Precision Location	wireless network (STC_4G)	lat: 24.450692 lng: 39.530797	nap: 10 source: 'wifi'	Precision: 34 Accuracy: 33.766395139
	wireless network (DSL)	lat: 24.450674 lng: 39.530874	nap: 23 source: 'wifi'	Precision: 31 Accuracy: 31.831818344

Comparing Google geolocation API and Skyhook achieved close mean accuracy values in Windows. Skyhook had greater accuracy, with an average accuracy of 18.05 m, than Google geolocation, which reached 19.1 m. On the other hand, they both demonstrated poor performance when using a Mac compared to Windows devices, with an average accuracy of 24.85 m for Google geolocation and 32.8 for Skyhook.

Figure 7 summarizes the results in the tables for the Wi-Fi positioning systems. When connected to a SIM card wireless network, Google achieved better results, while Skyhook achieved better results than Google when connected to a DSL type network.

The outcome is that we have been successful at incorporating the W3C geolocation API and Google IP geolocation techniques within an image to develop an attack by targeting a specific city or location, thus limiting exposure to detection engines and bypassing detection altogether.

4.2.2. The Stego-Image Analysis

A stego-image may host a malicious payload. In this paper, we chose the grayscale and color PNG images with sizes 205 KB and 3.22 KB, respectively, for performance analysis, as shown in Figure 8.

Figure 8. Experiment images. (**a**) PNG image with size 205 KB; (**b**) PNG image with size 3.22 KB.

For purposes of illustration, we injected the JavaScript code for window.alert ("Hacked!!!!") into an image. Figure 9 shows the original image, the encoded image that contains the geolocation and exploits code, and the polyglot image. It is difficult for the naked eye to distinguish between the original image, an encoded image, and a polyglot image. Figure 10 shows the image when loaded into a browser at the specified location, and the exploit was executed. However, if the W3C geolocation method is used to determine the target, a permission request to report the current location is displayed when an Internet user views the image, which would arouse the user's suspicion.

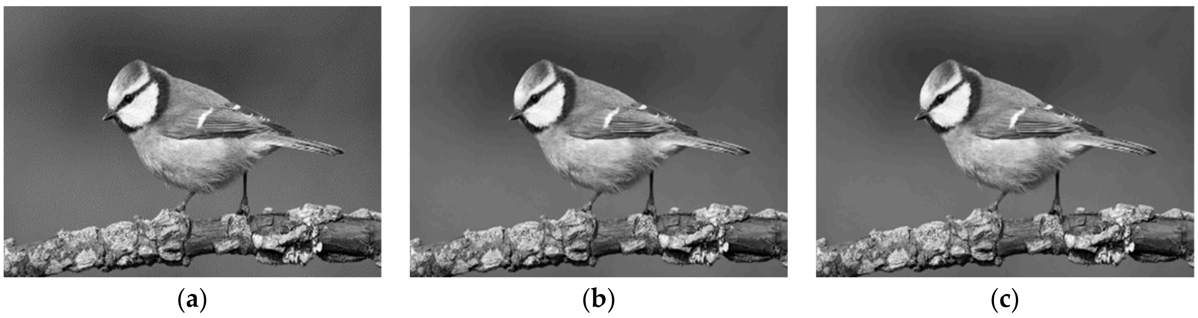
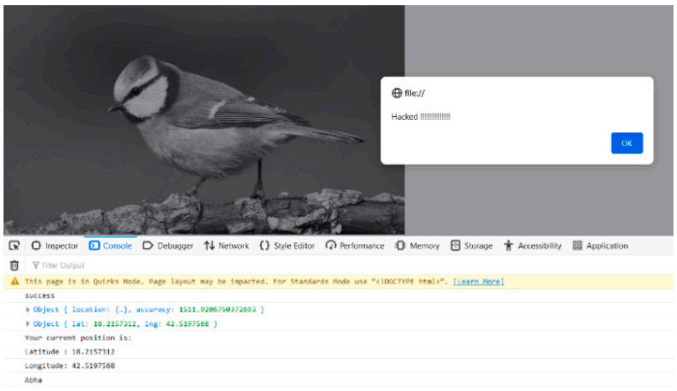


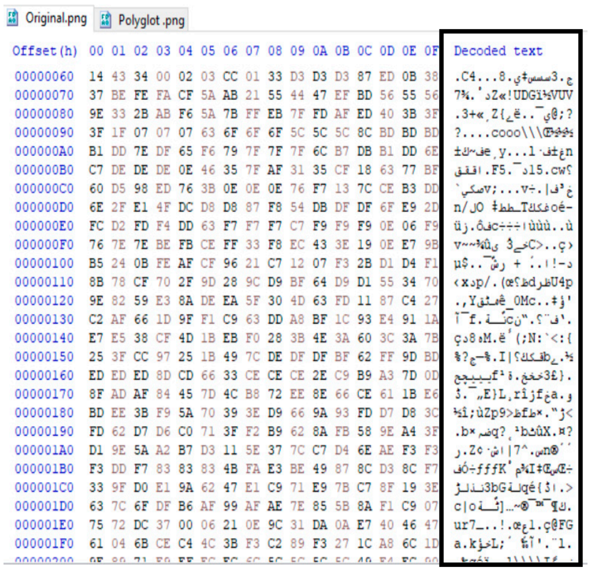
Figure 9. Original, Encoded, and Polyglot images in our experiments. (**a**) Original image; (**b**) Encoded image; (**c**) Polyglot image.



```
This page is in Quirks Mode. Page layout may be impacted. For Standards Mode use "<!DOCTYPE html>". [Learn More]
success
▸ Object { location: {…}, accuracy: 1511.9206750372693 }
▸ Object { lat: 18.2157312, lng: 42.5197568 }
Your current position is:
Latitude : 18.2157312
Longitude: 42.5197568
Abha
```

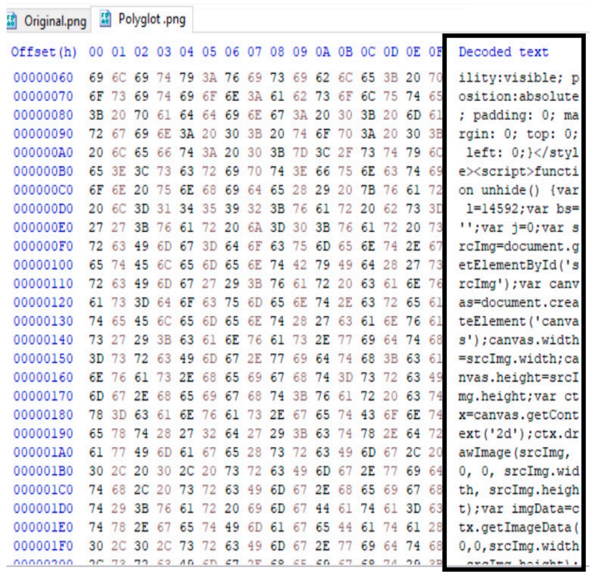
Figure 10. Exploit triggered in Firefox browser.

When we examined the benign and malignant images in a hex editor, we could see the injected text, as we can see in Figures 11 and 12. As shown in Figure 12, there is a string in the polyglot images at a specific byte range.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000060  14 43 34 00 02 03 CC 01 33 D3 D3 D3 87 ED 0B 38  .C4...8.سسس3‡ي.ع
00000070  37 BE FE FA CF 5A AB 21 55 44 47 EF BD 56 55 56  7¾.ˈدZ«!UDGï½VUV
00000080  9E 33 2B AB F6 5A 7B FF EB 7F FD AF ED 40 3B 3F  .3+«¸Z{ےë..¯ي@;?
00000090  3F 1F 07 07 07 63 6F 6F 6F 5C 5C 5C 8C BD BD BD  ?....cooo\\\Œ½½½
000000A0  B1 DD 7E DF 65 F6 79 7F 7F 7F 6C B7 DB B1 DD 6E  ±ك~ےe¸y...l·ف±غn
000000B0  C7 DE DE DE 0E 46 35 7F AF 31 35 CF 18 63 77 BF  اققق.F5.¯د15.cw؟
000000C0  60 D5 98 ED 76 3B 0E 0E 0E 76 F7 13 7C CE B3 DD  `صکيv;...v÷.|ف³غ
000000D0  6E 2F E1 4F DC D8 D8 87 F8 54 DB DF DF 6F E9 2D  n/لO ‡طط_Tغككoé-
000000E0  FC D2 FD F4 DD 63 F7 F7 F7 C7 F9 F9 F9 0E 06 F9  üز.ôفc÷÷÷اùùù..ù
000000F0  76 7E 7E BE FB CE FF 33 F8 EC 43 3E 19 0E E7 9B  v~~¾ûی ³خےC>..ç›
00000100  B5 24 0B FE AF CF 96 21 C7 12 07 F3 2B D1 D4 F1  µ$..¯رش + ..ا!-د
00000110  8B 78 CF 70 2F 9D 28 9C D9 BF 64 D9 D1 55 34 70  ‹xدp/.(œ؟طdظرU4p
00000120  9E 82 59 E3 8A DE EA 5F 30 4D 63 FD 11 87 C4 27  .‚Yمثقê_0Mc..‡ؤ'
00000130  C2 AF 66 1D 9F F1 C9 63 DD A8 BF 1C 93 E4 91 1A  آ¯f.ًةcن".؟"ف'.
00000140  E7 E5 38 CF 4D 1B EB F0 28 3B 4E 3A 60 3C 3A 7B  çه8دM.ëً(;N:`<:{
00000150  25 3F CC 97 25 1B 49 7C DE DF DF BF 62 FF 9D BD  %?ج—%.I|قكك؟bے.½
00000160  ED ED ED 8D CD 66 33 CE CE CE 2E C9 B9 A3 7D 0D  يييچخf3خخخ.ة¹£}.
00000170  8F AD AF 84 45 7D 4C B8 72 EE 8E 66 CE 61 1B E6  ڈ.¯„E}L¸rîژfخa.و
00000180  BD EE 3B F9 5A 70 39 3E D9 66 9A 93 FD D7 D8 3C  ½î;ùZp9>ظfڑ“.×ط<
00000190  FD 62 D7 D6 C0 71 3F F2 B9 62 8A FB 58 9E A4 3F  .b×ضہq?ٍ¹bûX.¤?
000001A0  D1 9E 5A A2 B7 D3 11 5E 37 7C C7 D4 6E AE F3 F3  ر.Z¢·س.^7|اشn®ََ
000001B0  F3 DD F7 83 83 83 4B FA E3 BE 49 87 8C D3 8C F7  َف÷ƒƒƒKْم¾I‡ŒسŒ÷
000001C0  33 9F D0 E1 9A 62 47 E1 C9 71 E9 7B C7 8F 19 3E  3ڈذلڑbGلةqé{اڈ.>
000001D0  63 7C 6F DF B6 AF 99 AF AE 7E 85 5B 8A F1 C9 07  c|oك¶¯™¯®~…[ٹًة.
000001E0  75 72 DC 37 00 06 21 0E 9C 31 DA 0A E7 40 46 47  urـ7...!.œ1ع.ç@FG
000001F0  61 04 6B CE C4 4C 3B F3 C2 89 F3 27 1C A8 6C 1D  a.kخؤL;َآ‰َ'.¨l.
```

Figure 11. The original image file structure is in hexadecimal view.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F  Decoded text
00000060  69 6C 69 74 79 3A 76 69 73 69 62 6C 65 3B 20 70  ility:visible; p
00000070  6F 73 69 74 69 6F 6E 3A 61 62 73 6F 6C 75 74 65  osition:absolute
00000080  3B 20 70 61 64 64 69 6E 67 3A 20 30 3B 20 6D 61  ; padding: 0; ma
00000090  72 67 69 6E 3A 20 30 3B 20 74 6F 70 3A 20 30 3B  rgin: 0; top: 0;
000000A0  20 6C 65 66 74 3A 20 30 3B 7D 3C 2F 73 74 79 6C   left: 0;}</styl
000000B0  65 3E 3C 73 63 72 69 70 74 3E 66 75 6E 63 74 69  e><script>functi
000000C0  6F 6E 20 75 6E 68 69 64 65 28 29 20 7B 76 61 72  on unhide() {var
000000D0  20 6C 3D 31 34 35 39 32 3B 76 61 72 20 62 73 3D   l=14592;var bs=
000000E0  27 27 3B 76 61 72 20 6A 3D 30 3B 76 61 72 20 73  '';var j=0;var s
000000F0  72 63 49 6D 67 3D 64 6F 63 75 6D 65 6E 74 2E 67  rcImg=document.g
00000100  65 74 45 6C 65 6D 65 6E 74 42 79 49 64 28 27 73  etElementById('s
00000110  72 63 49 6D 67 27 29 3B 76 61 72 20 63 61 6E 76  rcImg');var canv
00000120  61 73 3D 64 6F 63 75 6D 65 6E 74 2E 63 72 65 61  as=document.crea
00000130  74 65 45 6C 65 6D 65 6E 74 28 27 63 61 6E 76 61  teElement('canva
00000140  73 27 29 3B 63 61 6E 76 61 73 2E 77 69 64 74 68  s');canvas.width
00000150  3D 73 72 63 49 6D 67 2E 77 69 64 74 68 3B 63 61  =srcImg.width;ca
00000160  6E 76 61 73 2E 68 65 69 67 68 74 3D 73 72 63 49  nvas.height=srcI
00000170  6D 67 2E 68 65 69 67 68 74 3B 76 61 72 20 63 74  mg.height;var ct
00000180  78 3D 63 61 6E 76 61 73 2E 67 65 74 43 6F 6E 74  x=canvas.getCont
00000190  65 78 74 28 27 32 64 27 29 3B 63 74 78 2E 64 72  ext('2d');ctx.dr
000001A0  61 77 49 6D 61 67 65 28 73 72 63 49 6D 67 2C 20  awImage(srcImg, 
000001B0  30 2C 20 30 2C 20 73 72 63 49 6D 67 2E 77 69 64  0, 0, srcImg.wid
000001C0  74 68 2C 20 73 72 63 49 6D 67 2E 68 65 69 67 68  th, srcImg.heigh
000001D0  74 29 3B 76 61 72 20 69 6D 67 44 61 74 61 3D 63  t);var imgData=c
000001E0  74 78 2E 67 65 74 49 6D 61 67 65 44 61 74 61 28  tx.getImageData(
000001F0  30 2C 30 2C 73 72 63 49 6D 67 2E 77 69 64 74 68  0,0,srcImg.width
```

Figure 12. The polyglot image file structure is in hexadecimal view.

We performed an image analysis by calculating the Mean Square Error (MSE), Structured Similarity Index Measure (SSIM), and Peak Signal to Noise Ratio (PSNR), as reported in Table 15. Generally, the MSE is defined as the difference between the actual and the estimated values. It is used to calculate the average of the squares of the estimation errors or deviations [12,36]. It is calculated as follows:

$$\mathrm{MSE} = 1/(\mathrm{m} \times \mathrm{n}) \sum_{\mathrm{i}=0}^{\mathrm{m}-1} \sum_{\mathrm{j}=0}^{\mathrm{n}-1} [\mathrm{I}(\mathrm{i},\mathrm{j}) - \mathrm{K}(\mathrm{i},\mathrm{j})]^2 \quad (1)$$

where m and n are the pixel counts of the images, i and j are the current pixel values, I(i,j) is the original image, and K(i,j) is the image to be compared.

Table 15. MSE, SSIM, and PSNR were calculated between the original and encoded images.

Color	Size Encoded Image	Size Polyglot Image	MSE	SSIM	PSNR
Grayscale with IP	212 KB	213 KB	0.01099	0.99989	67.78051
Grayscale with W3C	211 KB	212KB	0.00859	0.99992	68.77387
Color with IP	12.2 KB	13.0 KB	0.05264	0.99967	60.76431
Color with W3C	11.8 KB	12.6 KB	0.04148	0.99972	61.88862

The SSIM is used to compare the similarity of two images. It is a metric that measures perceived variations or degradation in image quality caused by changes to the file structure [12,37]. The SSIM metric is calculated using the following formula:

$$\text{SSIM}\ (\text{x, y}) = \left(2\mu_x\mu_y + c_1\right)(2\sigma_x y + c_2)/\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right) \quad (2)$$

where x and y represent the window sizes of the images to be compared, μ is the average of images x and y, σ^2 is the variance of x and y, σ is the covariance of both images, and c_1 and c_2 are two variables to stabilize the division with the weak denominator. SSIM values range between 0 and 1; the closer the value to 1, the more similar the images are.

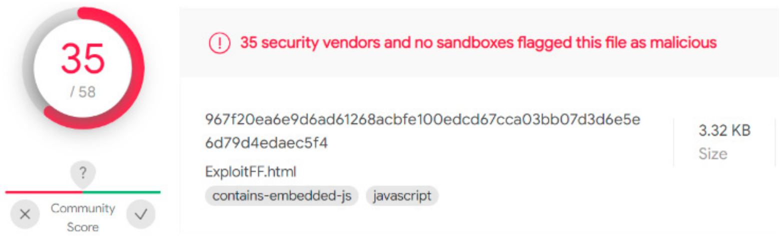
The PSNR metric estimates image quality. For example, it can determine the distortion between the original and encoded images [38]. It is defined as follows:

$$\text{PSNR} = 20\log_{10}\left(\text{N}/\sqrt{\text{MSE}}\right) \quad (3)$$

where N is the most difference between pixels in an image. The higher the PSNR value, the more similar the encoded image to the original image. It is measured in decibels (dB).

As we can see in Figure 9, the results show that there is not much difference between the original and polyglot images. Table 15 shows the MSE, SSIM, and PSNR calculated between the original and encoded images and the size of the files. We noticed an increase in the size of the image files, as the size of the polyglot grayscale image increased by 3.90%, and the size of the color image increased by 303.72%. We noted the value of the SSIM is closer to 1, indicating that the generated image is nearly identical to the original image despite having an attacking data embedded. Moreover, the MSE value is closer to 0, which indicates a lower error rate, as the closer to zero is the better. Furthermore, the PSNR is greater than 50 dB, so it is complicated for human eyes to recognize the difference in the images. The MSE was observed to increase as the image size increased. SSIM, on the other hand, decreased with increased image size. As a result, the more characters encoded, the more aberration can be seen.

To demonstrate that image-based attacks cannot be detected as malicious by antivirus engines, we used VirusTotal [39] to examine malicious code in a JavaScript file. This file was identified as malignant by 35 of 58 antivirus programs. However, when we embedded the malicious code into the image and tested it, we surprisingly found that none of the antivirus programs could detect the malicious code in the image or identify the file as containing hidden, malicious content. After all, neither the reputation nor the signature is available, as shown in Figures 13 and 14. Evasion is possible because antiviruses consider it an image devoid of any visible malicious code.



35 / 58

35 security vendors and no sandboxes flagged this file as malicious

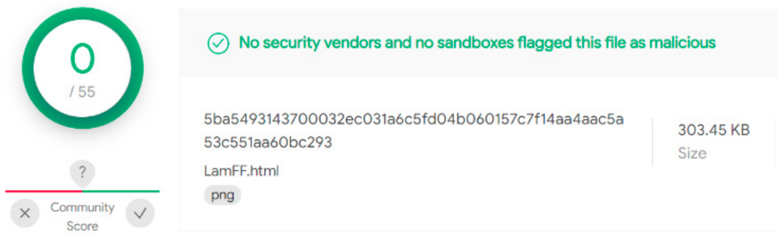
967f20ea6e9d6ad61268acbfe100edcd67cca03bb07d3d6e5e
6d79d4edaec5f4

ExploitFF.html

contains-embedded-js javascript

3.32 KB
Size

Figure 13. Malware detection in a JavaScript file.



0 / 55

No security vendors and no sandboxes flagged this file as malicious

5ba5493143700032ec031a6c5fd04b060157c7f14aa4aac5a
53c551aa60bc293

LamFF.html

png

303.45 KB
Size

Figure 14. Malware detection in an image file.

5. Discussion

The reference studies in this issue are few. Therefore, we were trying to talk about a different track and successfully carried out this kind of attack. We noticed a real lack of studies on offensive methods. We did not find research illustrating a targeted attack using geolocation information. Most of them are indiscriminate and not targeted attacks. In [11], they used an external extractor program to determine the target and trigger the stego-image. In our work, our attack happens when we reach the target location. Furthermore, it occurs on the same web page just by opening the image without user interaction. We have concealed the trigger conditions by embedding them in the image.

The suggested scheme's two primary components are the geolocation condition (trigger) and the stego-image.

It was established that there were variations between the geolocation services offered by W3C, IP, and WPS. As shown in Figure 5, the comparison analysis of the W3C geolocation method was performed on the four devices identified earlier, using different browsers and two access networks. The result shows that the accuracy is much better for DSL than for the data SIM card. iPhone browsers have the most accurate location results for devices, with an average accuracy of fewer than 20 m. In windows, the Firefox browser had the most accurate location results, with an average accuracy of 3.95 m. In contrast, the Google Chrome browser provided the worst value and was not acceptable. In a Mac laptop, we found that Google Chrome had greater accuracy, with an average accuracy of 7.7 m. Microsoft Edge had the worst performance compared to the other browsers, with an average accuracy of 43.85 m. In the iPhone, the Firefox browser had the most accurate location results, with an average accuracy of 12.15 m. Finally, in Huawei, we find that Microsoft Edge had greater accuracy than Chrome, with an average accuracy of 20.2 m, while Chrome had an average accuracy of 151.2 m.

We evaluated location accuracy for eight IP services databases for IP geolocation, as shown in Figure 6. According to the findings, the databases achieved perfect accuracy at the country level. Furthermore, we noticed that all services were correct when connected to a DSL network, except for the Skyhook Hyperlocal IP. However, when the device is connected to a SIM card wireless network, findings show that most databases produced incorrect results at the region and city levels. The best database was Google, with an accuracy of 5 km. On the other hand, when the device is connected to a DSL wireless network, most databases produce valid results at the district and city level except for the Skyhook database, which returned coordinates for another city. The best database again was Google, with an accuracy of 2 km.

In Figure 7, we see the comparison analysis of the WPS in Windows and Mac. When connected to a SIM card wireless network, Google achieved better results, while Skyhook achieved better results than Google when connected to a DSL-type network. Comparing Google geolocation API and Skyhook achieved close mean accuracy values in Windows. On the other hand, they both demonstrated poor performance when using a Mac compared to a Windows device.

Secondly, we have the stego-image analysis. It is difficult for the naked eye to distinguish between the original image, an encoded image, and a polyglot image, as shown in Figure 9.

Finally, as shown in Figure 10, we successfully demonstrated the execution of the exploit for the browser in a specific city by using a single image, thus limiting exposure to detection engines and bypassing detection altogether. We successfully combined the W3C geolocation API and Google IP geolocation techniques to develop a stegosploit attack. We carried out our attack on color and grayscale images of PNG format type. The polyglot image files are not detected as being malicious or identify the file as containing hidden, malicious content by any antivirus programs on VirusTotal, as shown in Figure 14. The general idea is applied to any vulnerable browser if it supports HTML5 Canvas features required by Stegosploit regardless of the OS.

There are a few limitations to this study. First, images should be large enough to host the exploit and geolocation codes. Second, we only tried our attack with images in PNG format. Finally, the test was performed on one region without repetition at longer intervals for geolocation techniques.

6. Conclusions

We can't wait for threats to be discovered in the wild to begin planning our defenses. Cybersecurity is a competition where attackers and defenders play a continuously evolving cat and mouse game. It is time for the security strategy to move from reactive to proactive and consciously search for attack strategies for the next generation. Instead of focusing on the traditional reactive approach, namely "Rules, Signatures, and Updates," the security establishment must start devising creative ways to identify and protect against these threats. We must understand the hacker's strategies and offensive techniques to improve defense and recognize network and computer system weaknesses.

No software is ever completely secure, and preventing the presence of vulnerabilities is nearly impossible. Advances in security systems have compelled malware developers to look into new ways to make their "products" stealthier. Steganography is used to ensure privacy by concealing confidential information in images. However, it may also be viewed as a tool for a potential attack, with all of the benefits turned into threats. Steganography-based exploit delivery is a hazard and is difficult to detect. In particular, image files containing malicious code have a meager detection rate, and complementary methods are needed to solve this problem. Attacks with this scheme have longer latency and are more difficult to detect, nor is it easy to examine the malicious payload hidden in image pixels.

In this paper, we have developed an offensive model using the Stegosploit toolkit against vulnerable browsers to generate a malicious image where the malicious payload will only be unlocked if the location target is reached. In the target browser, the image decodes itself and performs the exploit without user interaction. Finally, we compared the accuracy of the geolocation systems used for evaluation based on the devices, browsers, and access network types. In addition, we examined the quality of the stego-image relative to the original image.

Hopefully, this paper will provide a vision for computer system security and demonstrate how terrorists can secretly use modern-day technology to broadcast messages. Our goal is to use offensive security in cybersecurity practices to improve our defense. Based on our attack, the signatures analysis on an image file is useless. We propose a detection method for detecting such attacks by conducting the behavioral analysis of the attack at the target node.

In the future, we could extend this research to devising defensive methods to protect the network from such attacks and extend the proposed attack scenario to different image formats, as well as study the effect of such attacks on the deep neural network in image recognition.

Author Contributions: Conceptualization, L.A. and D.A.; methodology, L.A.; validation L.A., D.A. and A.B.; formal analysis, L.A.; investigation, A.B., D.A. and O.R., writing original draft preparation, L.A.; writing-review and editing, L.A.; visualization, L.A.; supervision, D.A. and A.B.; project administration, A.B.; funding acquisition, O.R. All authors have read and agreed to the published version of the manuscript.

Funding: The Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia has funded this project, under grant no. (RG-10-611-43).

Acknowledgments: The authors extend their appreciation to the Deanship of Scientific Research (DSR) at King Abdulaziz University (KAU), Jeddah, Saudi Arabia has funded this project, under grant no. (RG-10-611-43).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Appendix A

Table A1. Most recent attacks using the image.

Malware	Discovery Date	Carrier Type	Message	Delivery Method	Target
Shady RAT(the first attack to use steganography).	2006	JPEG images, HTML pages	Hiding instructions that would allow distant servers to access local files on the infected host machine.	Spamming emails and compromised websites are used to disseminate the redirect link of web attackers. The receiver then opens a specially crafted email message (phishing).	It targets various vulnerabilities found in Microsoft Windows, Mozilla Firefox, and other software.
Duqu	2011	PNG image	exfiltrated data	Industrial Control Systems (ICS).	
Lurk Downloader	2014	BMP and PNG image	Hiding an encrypted URL to download additional components of malware (second payload).	Victims are infected by HTML <iframes> on hacked websites that have a Flash-based exploit.	Committing click fraud is the intent behind this malware.
Vawtrak/Neverquest malware	2015	Favicons	hiding URL to downloading its configuration file.	Spread via phishing attacks and websites that are hacked.	Financial malware.
The Gatak/Stegoloader malware	2015	PNG image	malicious code	Gatak sample showing two executable files inside a compressed archive. The first is the installer file for the software license cracking tool, and the other is Gatak malware.	trojan or downloader for stealing data and delivering ransomware
Stegosploit	2015	PNG, JEPG image	Drive-by browser exploits(The Use-After-Free vulnerability in Internet Explorer (CVE-2014-0282) +HTML and JavaScript decoder code	The polyglot seems like an image, but when loaded, it is decoded and activated in a victim's browser.	It exploits browsers' vulnerabilities

Table A1. *Cont.*

Malware	Discovery Date	Carrier Type	Message	Delivery Method	Target
AdGholas malware	2016	images, text, and HTML code	Hiding encrypted malicious JavaScript code	Banner advertisements have been used by criminals to spread malware.	malvertising attacks
Cerber ransomware	2016	JPEG file	Embedding malicious executable	Spread by the Microsoft Office 365 cloud platform.	Various sectors
Stegano/Astrum exploit kit	2016	PNG image	Hiding malicious code inside advertising banners	An infected ad.	malvertising campaign
DNSChanger exploit kit	2016	PNG image	Hiding malware, the AES encryption key inside an innocent ad to decode network traffic	Malware ads, also called malvertising.	The purpose of this malware is to hit the routers of users instead of their browsers.
SyncCrypt ransomware	2017	JPEG image	Embedding part of the core ransomware components	WSF (Windows Script File) attachments in compromised emails.	Generic Internet users
Powload malware	2018	PNG file	embedding malicious scripts in image	By phishing email campaigns containing documents embedded with malicious macro code.	Stealing personal data from the aim and other malicious activities.
VeryMal malware's	2019	The JPEG image is a tiny white bar	malicious JavaScript code	Through ad images.	The MAC OS.
Sundown exploit kit		hiding data in white PNG files	exfiltrating user data or hiding the malicious code delivered to the victims	Malvertising campaigns.	It exploits many defects, including Internet Explorer Jscript handling (IE).
Ursnif malware		PNG image	malicious PowerShell script	Distributed via spam email carrying fake documents (Microsoft Excel documents) like a buying order.	Banking malware.

References

1. Truong, T.C.; Diep, Q.B.; Zelinka, I. Artificial intelligence in the cyber domain: Offense and defense. *Symmetry* **2020**, *12*, 410. [CrossRef]
2. Rudd, E.M.; Rozsa, A.; Günther, M.; Boult, T.E. A Survey of Stealth Malware Attacks, Mitigation Measures, and Steps Toward Autonomous Open World Solutions. *IEEE Commun. Surv. Tutorials* **2017**, *19*, 1145–1172. [CrossRef]
3. Siddiq, M.A.; Ghani, N. Critical Analysis on Advanced Persistent Threats. *Int. J. Comput. Appl.* **2016**, *141*, 46–50. [CrossRef]
4. Messer, A.; Medairy, B. The Future of Cyber Defense . . . Going on the Offensive. *Cyber Def. Rev.* **2018**, *3*, 37–40.
5. Cohen, A.; Nissim, N.; Elovici, Y. MalJPEG: Machine Learning Based Solution for the Detection of Malicious JPEG Images. *IEEE Access* **2020**, *8*, 19997–20011. [CrossRef]
6. Jung, D.S.; Lee, S.J.; Euom, I.C. Imagedetox: Method for the neutralization of malicious code hidden in image files. *Symmetry* **2020**, *12*, 1621. [CrossRef]
7. Vaidya, N.; Rughani, P. An Efficient Technique to Detect Stegosploit Generated Images on Windows and Linux Subsystem on Windows. *Int. J. Comput. Sci. Eng.* **2019**, *7*, 21–26. [CrossRef]
8. Beatty, M. The Current and Future Threat of Steganography in Malware Command and Control. Ph.D. Thesis, Utica College, Utica, NY, USA, 2019.
9. Brunot, J.M. The Increased Use of Steganography by Malware Creators to Obfuscate Their Malicious Code. Ph. D. Thesis, Utica College, Utica, NY, USA, 2019.
10. Dudheria, R. Attacking Smartphones by Sharing Innocuous Images via QR Codes. In Proceedings of the 12th Annual Symposium on Information Assurance (Asia '17), Albany, NY, USA, 7–8 June 2017; pp. 86–92.
11. Ko, H.-J.; Huang, C.-T.; Horng, G.; Wang, S.-J. Embedding Advanced Persistent Threat in Steganographic Images. In Proceedings of the Security with Intelligent Computing and Big-Data Services 2019, New Taipei City, Taiwan, 4–6 December 2019; Jain, L.C., Peng, S.-L., Wang, S.-J., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 3–17.
12. Jeyasekar, A.; Bisht, D.; Dua, A. Analysis of exploit delivery technique using steganography. *Indian J. Sci. Technol.* **2016**, *9*, 102075. [CrossRef]

13. Andriesse, D.; Bos, H. Instruction-level steganography for covert trigger-based malware. In Proceedings of the International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment, London, UK, 10–11 July 2014; Springer: Cham, Switzerland, 2014; Volume 8550 LNCS, pp. 41–50. [CrossRef]
14. McGraw, G.; Morrisett, G. Attacking malicious code: A report to the Infosec Research Council. *IEEE Softw.* **2000**, *17*, 33–41. [CrossRef]
15. Saeed, I.A.; Selamat, A.; Abuagoub, A.M.A. A Survey on Malware and Malware Detection Systems. *Int. J. Comput. Appl.* **2013**, *67*, 25–31. [CrossRef]
16. Mansoori, M.; Welch, I. Geolocation Tracking and Cloaking of Malicious Web Sites. In Proceedings of the 2019 IEEE 44th Conference on Local Computer Networks (LCN), Osnabrueck, Germany, 14–17 October 2019; pp. 274–281. [CrossRef]
17. Khaldi, A. Steganographic Techniques Classification According to Image Format. *Int. Ann. Sci.* **2019**, *8*, 143–149. [CrossRef]
18. Wiseman, S. *Stegware–Using Steganography for Malicious Purposes*; ResearchGat: Berlin, Germany, 2017.
19. Dhawan, S.; Gupta, R. Analysis of various data security techniques of steganography: A survey. *Inf. Secur. J.* **2020**, *30*, 63–87. [CrossRef]
20. Evsutin, O.; Melman, A.; Meshcheryakov, R.; Karakus, S.; Avci, E.; Arroyo, J.C.T.; Arroyo, J.C.T.; Kadhim, I.J.; Premaratne, P.; Vial, P.J.; et al. Critical Analysis on Advanced Persistent Threats. *IEEE Access* **2019**, *8*, 1–6. [CrossRef]
21. Nagy, B. PoC||GTFO 08. *Int. J. PoC||GTFO* **2015**, 1–64. Available online: https://www.alchemistowl.org/pocorgtfo/ (accessed on 21 April 2022).
22. Cabaj, K.; Caviglione, L.; Mazurczyk, W.; Wendzel, S.; Woodward, A.; Zander, S. The new threats of information hiding: The road ahead. *IT Prof.* **2018**, *20*, 31–39. [CrossRef]
23. Samba.org Samba 4.5.9—Release Notes. Available online: https://www.samba.org/samba/history/samba-4.5.9.html (accessed on 17 May 2022).
24. Kwon, H.; Kim, Y. BlindNet backdoor: Attack on deep neural network using blind watermark. *Multimed. Tools Appl.* **2022**, *81*, 6217–6234. [CrossRef]
25. Atencio, Y.P.; Ubalde Enriquez, R.; Ibarra, M.J.; Huanca Marin, J. How to locate where a device is using a web application. In Proceedings of the 15th Latin American Conference on Learning Technologies, LACLO 2020, Loja, Ecuador, 19–23 October 2020; pp. 1–6.
26. Steiner, T.; Kostiainen, A.; Kruisselbrink, M. Geolocation in the browser from Google gears to geolocation sensors. In Proceedings of the WWW '19: Companion 2019 World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019; pp. 913–918. [CrossRef]
27. Holdener, A.T. *HTML5 Geolocation*; O'Reilly Media, Inc.: Newton, MA, USA, 2011; ISBN 9780874216561.
28. Kysela, J. Comparison of web applications geolocation services. In Proceedings of the 2014 IEEE 15th International Symposium on Computational Intelligence and Informatics (CINTI), Budapest, Hungary, 19–21 November 2014; pp. 449–453. [CrossRef]
29. Pierre, N.J.; Fan, X.; Daniel, G.; Claude, G.J. Cross-Platform Mobile Geolocation Applications Based on PhoneGap. *Lect. Notes Softw. Eng.* **2015**, *3*, 78–82. [CrossRef]
30. Tian, Y.; Zheng, N.; Chen, X.; Gao, L. Wasserstein Metric-Based Location Spoofing Attack Detection in WiFi Positioning Systems. *Secur. Commun. Netw.* **2021**, *2021*, 8817569. [CrossRef]
31. Skyhook Skyhook|Location Technology Provider. Available online: https://www.skyhook.com/ (accessed on 8 April 2022).
32. Google Overview|Geolocation API|Google Developers. Available online: https://developers.google.com/maps/documentation/geolocation/overview (accessed on 8 April 2022).
33. Shah, S.; Vaidya, N.; Rughani, P. "Saumil Shah," Hack.Lu. 2015, pp. 21–26. Available online: https://conference.hitb.org/hitbsecconf2015ams/wp-content/uploads/2015/02/D1T1-Saumil-Shah-Stegosploit-Hacking-with-Pictures.pdf (accessed on 18 March 2022).
34. CSGNetwork.com. "GPS Latitude and Longitude Distance Calculator," CSG Computer Support Group, Inc. 2018. Available online: http://www.csgnetwork.com/gpsdistcalc.html (accessed on 23 February 2022).
35. Zandbergen, P.A. Comparison of WiFi positioning on two mobile devices. *J. Locat. Based Serv.* **2012**, *6*, 35–50. [CrossRef]
36. Arroyo, J.C. LSB Image Steganography with Data Compression Technique Using Goldbach G0 Code Algorithm. *Int. J. Emerg. Trends Eng. Res.* **2020**, *8*, 3259–3264. [CrossRef]
37. Evsutin, O.; Melman, A.; Meshcheryakov, R. Digital Steganography and Watermarking for Digital Images: A Review of Current Research Directions. *IEEE Access* **2020**, *8*, 166589–166611. [CrossRef]
38. Hossain, S.; Mukhopadhyay, S.; Ray, B.; Ghosal, S.K.; Sarkar, R. A secured image steganography method based on ballot transform and genetic algorithm. *Multimed. Tools Appl.* **2022**, *81*. [CrossRef]
39. VirusTotal VirusTotal-Home. 2021. Available online: https://www.virustotal.com/gui/home/upload (accessed on 30 March 2022).