Stegomalware

Authors: Danyil Tymchuk (B00167321), Illia Stefanovskyi (B00165280), Artem Surzhenko (B00163362)

Purpose: This report was written for the TU-Dublin university as a project for Digital Forensics and Cyber Security course.

Course: Digital Forensics and Cyber Security (TU863/Y3)
Module: Computer and Network Forensics
Submission date: 07/12/2025
Word count: 7000

Table of Contents

Abstract

Malware in its modular implementation is using all kinds of techniques to be more efficient and unpredictable. Steganography is a tool that is actively being used for that kind of actions for its ability to hide the solely existence of an attack rather than just encrypting its elements. The beauty of steganography is the difficulty of its detection. Throughout the report we will examine cases when such techniques have been used in real world attacks and what significance they had in their success. We will experimentally showcase different types of data embedding into images. As detection difficulty is a base block of steganography, it will be broken down into different application areas and use cases to propose techniques to achieve it. Software solutions for different steganography implementation, detection and mitigation contexts are going to be listed throughout the report.

Introduction

Modern malware attacks focus not only on reaching the target goal, but also on hiding the presence and trace of the attack. The main reasons are attack repetition and evasion of responsibility, if the breach itself or its reason is not detected, it will not be investigated and loopholes will remain. According to Caviglione, L. & Mazurczyk, W. (2022), steganography is a right match for attacks that require that extent of accuracy, since its main focus is on hiding data and not reformatting or encrypting it. Due to its capabilities there is a fair number of known cases where some form of steganography was used to perform an attack. Modern malware, just as any software, uses modular approach for its redundancy and efficiency, this allows steganography to be used as some part of an attack rather than its entirety.

A series of experiments will be conducted to embed potentially malicious information into a media carrier for reconstructing part of a stegoloader attack. Different techniques will be used including the simple embedding of data into slack space and more complex spreading it over the media file to increase secrecy. Also LSB technique will be used to leave image size the same before and after steganographical modification.

Development of techniques to detect malware is ongoing. As of Locard Exchange principle, even the most sophisticated attacks leave some trace that can be spotted if examined correctly. It can be used for attacker identification or prevention of similar attacks in the future via reconstruction of logic and events. Those are needed to invent countermeasures against new types of steganography and implement precise security measures instead of classic modification of data hoping that it will eliminate embedded contents. Countermeasures against data altering will also be reviewed to

showcase inefficiency of such techniques. Based on media and channel types more efficient techniques will be proposed.

The purpose of this report is analyzing steganography usage in malware by examining real world attacks. Stenography detection (steganalysis) and its mitigation techniques are going to be classified and reviewed in the context of malware trace analysis and detection. Software solutions for dealing with such threats in all layers of applications will be proposed.

Background: How Steganography Actually Works (and Why It's a Nightmare for Forensics)

"Stegoloader" case that I discovered that the malware simply stuffs its malicious code into a harmless-looking PNG file — the image still opens normally, but the hidden payload lives in the file's least-significant bits. According to the SecureWorks Counter Threat Unit (2015) the payload is never written to disk; it is extracted directly into memory and executed, which means traditional disk-based signatures can't see it. In practice the carrier image can be any picture, audio clip, or video;As described by Mazurczyk and Caviglione (2014) the steganographic algorithm just swaps a few bits that the human eye can't notice, so the file passes routine visual or acoustic inspection. Because the carrier remains functionally intact, forensic tools that only hash or scan for known binaries completely miss the hidden data. This "invisibility" forces analysts to rely on statistical quirks or machine-learning models that try to spot the tiny anomalies left behind, and even then the detection rate drops dramatically when the payload is small.

Steganalysis: The Uphill Battle of Detection

Detecting hidden data is a lot tougher than catching ordinary malware. According to Berk, Giani and Cybenko (2005) steganalysis is essentially a problem of spotting statistical outliers in a sea of normal media, that's why many tools eventually become "blind"—they look for any irregularity rather than a specific algorithm . Researchers have tried to strengthen detection by normalising network traffic or adding checksum checks, but these measures only catch obvious cases and still leave sophisticated embeds untouched . More recent work focuses on categorising covert channels (object, platform, network) and building specialised filters for each, yet the sheer variety of media formats means no single solution works for all . Steganography packages like StegExpose can be circumvented by tweaking the embedding strength, in which case analysts must constantly retrain deep learning models on new datasets to keep up. Simply put, each new steganography method forces new research, constantly keeping the forensic community on the move.

Case Studies

This section delves into key real-world applications and challenges, demonstrating the practical use and effectiveness of the principles discussed throughout this work. A common thread running through these cases is threat actors' increasing sophistication in using digital steganography – the art of concealing a file, message, or executable code within an ordinary, non-suspicious file, such as an image.

Traditional security measures, like antivirus software based on signatures, are inherently incompatible with this method. The main purpose of these tools is to search for suspicious file header information or scan executable files and documents for known malicious signatures. The image itself is considered harmless and is not flagged as having malware because the malicious payload is embedded directly within the pixel data (typically in the Least Significant Bit, or LSB) of an image file that appears to be legitimate (such as a PNG or SVG). No malicious executable ever touches the disc for the antivirus to scan because the malicious code is only extracted and run in memory by a loader that comes before it.

Case Study Overview

Case Study 1 – Stegoloader

This advanced family of malware was the first to use digital steganography to avoid detection when it was found in late 2013. It concealed its main executable code, an information-stealer, within PNG images that appeared authentic and were hosted on reliable websites. Stegoloader targeted valuable intellectual property with remarkable stealth thanks to its design, which included strict anti-analysis checks that prevented execution if forensic tools were found.

Case Study 2 – The "ClickFix" Campaign

The ClickFix campaign is a modern evolution that combines steganography with a skilful social engineering trick, such as displaying fictitious error messages (like "Update Failed"). The attack launches a loader that downloads a standard image file (PNG or SVG) after tricking the user into executing an apparently helpful command. Importantly, information-stealers like LummaC2 are delivered by this image's hidden payload, which is extracted and run straight in the browser's memory. The technique's commercialisation for widespread illegal use is highlighted by this change from executable binaries to concealing JavaScript inside images.

The other Example – AdGholas

These two instances are not unique instances. Many similar malware, such as the AdGholas malvertising operation, which we do not go into detail about in this report, take advantage of this blind spot. The Industrialist application of the technique is represented by AdGholas (sometimes linked to Lurk), where the threat actor bought real ad space on trustworthy websites and used image-based steganography to hide malicious JavaScript inside banner ads to reroute a large number of users to exploit kits, proving the scalability of the technique.

Case Study 1 – Stegoloader

In late 2013, the complex malware family Stegoloader (also called Win32/Gatak.DR or TSPY_GATAK.GTK) was discovered. Unlike "mass market" threats that rely on volume, Stegoloader exploits stealth tactics to elude detection, most notably digital steganography. This approach allows the malware to camouflage its primary executable code among legitimate-looking Portable Network Graphics (PNG) images. (Unit, D.S.C.T. and Intelligence, T., 2015)

Malware developers are always coming up with new ways to get around intrusion detection systems that are both host-based and network-based. By integrating steganography directly into its deployment lifecycle, Stegoloader is a notable advancement in this field. The malware, which was initially discovered by researchers from the Dell SecureWorks Counter Threat Unit (CTU), is essentially an information stealer intended to exfiltrate private data, such as passwords, system information, and proprietary software licenses, or also could install another malicious software. The Vundo (also known as Ponmocup) malware, which shows advertisements and installs more malware, has been seen to download and install on some Stegoloader variations. Once they have extracted all the information they find interesting, stegoloader operators may install Vundo on a compromised system for further financial gain. (Unit, D.S.C.T. and Intelligence, T., 2015)

Technical Analysis

Stegoloader operates through a modular design comprising a deployment module and a main module, along with several optional payload modules.

Deployment and Anti-Analysis

The deployment module downloads and launches the main module without persistence. The malware makes sure it is not operating in an analysis environment before deploying additional modules. For instance, the deployment module repeatedly calls the GetCursorPos function to track mouse cursor

movements. If the mouse is always in a different position or does not change, the malware ends without causing any harm. (Unit, D.S.C.T. and Intelligence, T., 2015)

To reduce static analysis time, the majority of binary strings are built on the program stack before use. This standard malware technique dynamically constructs strings instead of storing them in clear text, making detection and analysis more difficult. Stegoloader checks the system for running processes and terminates them if their names match the strings, for example ‘Wine’, ‘wireshark.exe’ or an antivirus. The majority of the strings represent security products or tools for reverse engineering. When Stegoloader detects analysis or security tools on the system that can detect the malware, it does not execute the main program code. (Unit, D.S.C.T. and Intelligence, T., 2015)

Steganography Implementation

The core innovation of Stegoloader is its payload delivery method. A harmless-looking PNG image is downloaded by the deployment module from a reputable hosting website.

- Extraction: Using the gdiplus library, the malware accesses the pixel data of the PNG.
- Decryption: It takes each pixel's colour data and extracts the Least Significant Bit (LSB). The RC4 algorithm is then used to assemble and decrypt this data stream.
- In-Memory Execution: The decrypted payload (the Main Module) is executed directly in memory. Crucially, neither the PNG containing the code nor the decrypted binary are ever written to the storage device, making traditional disk-based signature detection ineffective. (Unit, D.S.C.T. and Intelligence, T., 2015)

Command and Control (C2)

Communication is conducted via standard HTTP requests.

- Deployment Module: Reports status via HTTP GET requests.
- Main Module: Uses HTTP POST requests encrypted with RC4. The malware creates random session identifiers and pre-pends a CRC32 checksum to confirm message integrity in order to avoid traffic analysis.

Modular Capabilities

Stegoloader’s main module acts as a platform for deploying additional functional modules based on the victim's profile. These modules are executed in memory:

- Pony Password Stealer: This module is capable of stealing passwords for most popular protocols, including POP, IMAP, FTP, and SSH. The Pony password stealer module transmits stolen information to the main module's C2 server via the same protocol. (Unit, D.S.C.T. and Intelligence, T., 2015)

- Host geographic localization: This module launches Internet Explorer and contacts two websites, ip2location.com and whoer.net, to obtain information about the visitor's public IP address. Websites provide geolocation information for visitors' IP addresses. The module compresses and sends HTML material to its C2 server via the same server and protocol as the main module. (Unit, D.S.C.T. and Intelligence, T., 2015)
- List recently opened documents: This module uses the SHGetFolderPathA function from shell32.dll with the CSIDL_RECENT argument to locate the system folder that contains links to the victim's most recently used documents. The module parses the links, determines their location on the local hard drive, and sends the list of most frequently used links and files to the C2 server via the same server and protocol as the main module. (Unit, D.S.C.T. and Intelligence, T., 2015)
- IDA-stealing module: Stegoloader has a module that steals installed instances of the IDA program. If IDA is discovered on a hacked machine, the C2 server will deliver and run this module. This module utilises a distinct C2 server from the main module. The reporting pattern is identical to the deployment module. The IDA-stealing module detects registry entries linked to IDA and transmits them to a respectable online file-hosting server. After uploading a file, the hosting website provides a URL that allows users to download it. Then parses the response and sends files to the C2 server. (Unit, D.S.C.T. and Intelligence, T., 2015)

Indicators of Compromise

Organisations should monitor the following indicators associated with Stegoloader activity:

- Network Traffic: HTTP GET/POST requests to well-known C2 URLs with RC4-encrypted payloads.
- File Hashing:

  723ef64c6a1b1872bc84a9dc30e10c9199f5a153

  a48594b243f801e02066b77e46135382e890daf6

  and more…
- Filenames: Piracy-related executables like Avanquest_PowerDesk_9_0_1_10.exe.

Conclusion

Stegoloader achieves a high level of operational security by combining modular architecture and digital steganography. It calls into question traditional forensic methods by avoiding disc writes for its primary payloads and employing stringent anti-analysis checks. The malware's specific targeting of developer tools (such as IDA) and standard credential theft suggests a threat actor interested in high-value intellectual property and financial gain.

Case Study 2 – The "ClickFix" Campaign

Where Stegoloader worked as a covert infiltration, adding a layer of steganography for good measure.The scheme first appeared in the last months of 2024 and, by the first half of 2025, had already become one of the most troubling examples of hiding malicious code inside everyday images.

The Hook: When Software Turns Against You

ClickFix doesn't bother with phishing emails or suspicious attachments. Instead, it undermines user trust in everyday applications. Targets are shown a familiar-looking error screen – for instance, Chrome might pop up an "Update Failed" alert, while Microsoft Word could display a "Document Corrupted" notice. These aren't the clunky warnings you remember from older software; they are meticulously crafted reproductions, using the exact logos and the same kind of technical language the real applications employ.

They tell users exactly how to "fix" the problem. Press Win+R, paste this command, hit Enter. It feels helpful, authoritative. It's anything but.

Hiding JavaScript in Plain Sight

Once users take the bait, ClickFix deploys its steganographic payload through a multi-stage chain that's frankly brilliant in its complexity:

The first step runs the built-in Windows utility **mshta.exe**, which pulls a JScript file from a server under the attacker's control. The addresses used are deliberately obfuscated – they appear as hex-encoded fragments such as 141.0x62.80.175 and are frequently changed to stay off known blocklists. Info from Huntress Labs (2025) The JScript then spawns PowerShell, but not before wrapping itself in junk code designed to confuse analysts.

The PowerShell loader decrypts a .NET assembly directly in memory—no file ever touches the disk. This build contains its own steganographic extraction routine. It loads what looks like a regular PNG or SVG image, perhaps a company logo or social media icon. But embedded in those pixels, specifically in the red color channel values, lies the next stage of the attack.

The extraction algorithm is deceptively simple: process each pixel's red channel value through XOR and subtraction operations to rebuild encrypted shellcode. Once retrieved, the shellcode is slipped into a trusted Windows process (often **explorer.exe**) by way of classic in-memory injection steps: allocating space with **VirtualAllocEx**, copying the code via **WriteProcessMemory**, and finally launching it with **CreateRemoteThread**. When that routine finishes, a full-featured

information-stealer—examples being **LummaC2** or **Rhadamanthys**—is quietly running on the victim's machine, all triggered by a user who simply followed the bogus error-message instructions.

How ClickFix Differs from Stegoloader

Where Stegoloader targeted system-level credentials and installed secondary malware (remember Vundo?), ClickFix operates entirely in the browser-memory space. It's not trying to own the whole machine—at least not initially. It wants data, and fast.

PNG files are images with a fixed number of pixels, while SVG files are vectors with an XML structure that can be directly embedded using JavaScript. Microsoft Threat Intelligence (2025). Because many security products view SVGs as harmless user interface resources, this results in them often being "whitelisted," leaving room for attackers to hide executable code inside what appear to be harmless graphics.

ClickFix Infrastructure

Hexadecimal URL encoding and overlapping domains such as securitysettings.live and xoiiasdpsdoasdpojas.com are common infrastructure patterns found in ClickFix campaigns. ClickFix decoy websites are still operating even after law enforcement shut down Rhadamanthy's infrastructure in November 2024, demonstrating the campaigns' adaptability and durability.

Browser Threats

ClickFix reflects a disturbing trend: steganography is no longer theoretical; it is already present in your browser and targeted at your users. The campaign demonstrates that hiding malware in photos is not limited to sophisticated persistent threats; it has become commercialized enough for mass criminal use. Additionally, the use of fabricated error messages demonstrates that attackers realize that human psychology, not software, is the weakest link.

3 Things That Hides Danger: A Comparison of Mechanisms

Comparing the use of steganography in these 3 campaigns:

1. **Stegoloader:** was using RC4decryption and LSB steganographic modification to extract Windows executables, using a PNG image as a locking mechanism. Its behavior was pending forensic confirmation, based on Wireshark analysis and virtual machine anti-analysis techniques.
2. **ClickFix:** ClickFix hides JavaScript to make the script hard to reverse engineer.. It thinks if you whitelist its image media, regardless of whether you use analysis tools. Because the execution is

browser-based, there are no disk artifacts, no process injection (until the very end), only memory manipulation. It is built for the cloud technologies world.

3. **AdGholas/Lurk:** Company purchased actual ad space on reputable websites. Proofpoint (2016) By whitelisting PC images from OEMs, they will use image-based steganography to target specific consumers. AdGholas used poisoned ad networks to carpet bomb broad populations, while ClickFix targeted specific people with fictitious errors. Trend Micro (2017)

Conclusion

All of the examples illustrate different lessons: Stegoloader demonstrates that steganography can penetrate even sophisticated defenses; ClickFix demonstrates that the same technique can be used for a broad campaign, such as a consumer-facing one; and AdGholas demonstrates that attackers can use legitimate ad network infrastructure as a weapon. While delivery methods change, the basic idea—hiding malicious code in something that looks normal—remains frighteningly unchanged.

Primary Research

Simulation of Steganographic Payload Delivery

Objective

To simulate the evasion techniques used by the Stegoloader malware family. This experiment compares two methods of concealing a benign "malicious" payload (the EICAR test file) within digital imagery: a rudimentary "Append" attack and a sophisticated "Embedding" attack using Steghide.

Experiment Setup

For safety and ethical compliance, this study employs the EICAR Standard Anti-Virus Test File. This string of code is not malicious, but security software recognises it as a threat for testing purposes.

- Payload: eicar.com.txt
- Carrier File: A standard JPEG image (image.jpg)
- Tools: Command Line, Steghide, Aperi'Solve/binwalk.

EICAR String used:

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Experiment A: The "Append" Method (Overlay Steganography)

This method uses the file structure of image formats (such as JPEG). An image viewer reads the file from the header to the "End of File" (EOF) marker. Any data appended after this marker is ignored by the image viewer but is still physically present in the file. This simulates early or "dumb" malware hiding techniques.

Procedure

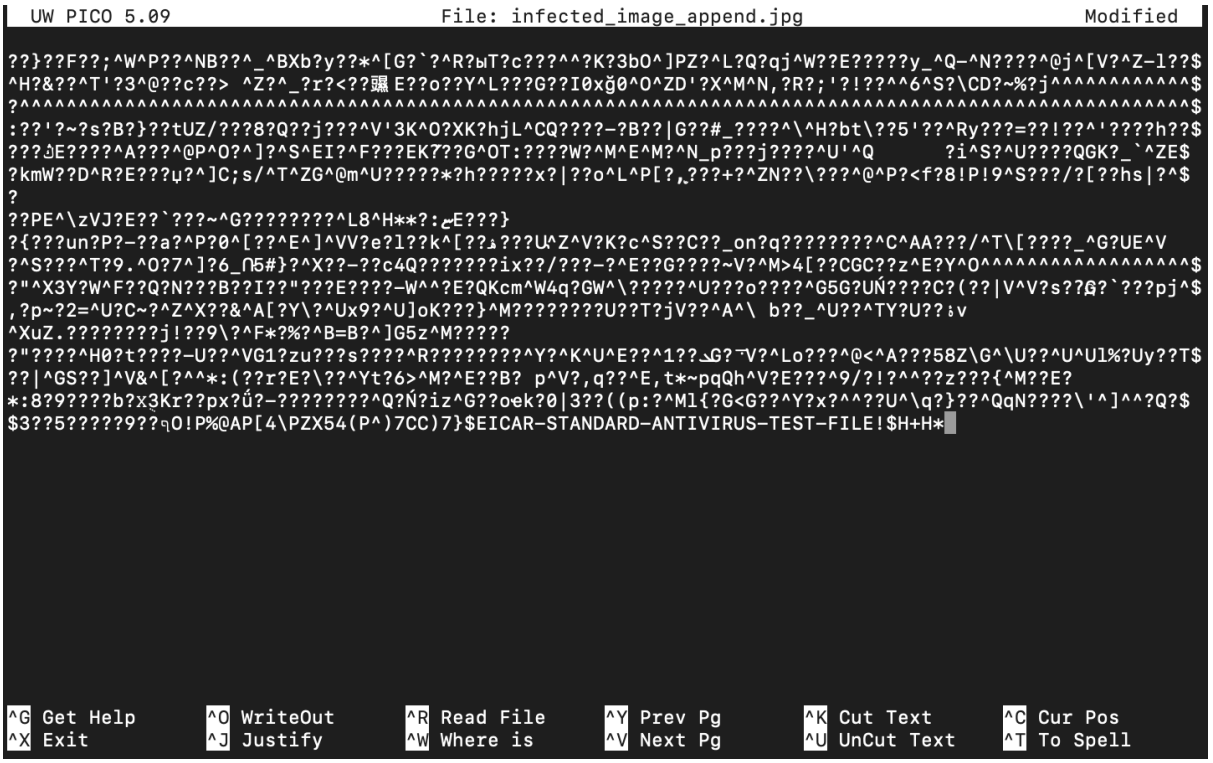
Preparation: Placed landscape.jpg and virus.txt (with the EICAR string) in the same directory.
Execution: Used the shell, terminal or Windows Command Prompt to navigate to the directory.
Command: Executed a binary copy to concatenate the files:

- bash: `cat image.jpg eicar.com.txt > infected_image_append.jpg`
- cmd: `copy /b image.jpg + eicar.com.txt infected_image_append.jpg`

Results

Visual Inspection: When opened in a standard image viewer, infected_image_append.jpg renders correctly. The image is not corrupted.

Forensic Analysis: The file was opened using a text editor (or cat/nano command). While the image data appeared as gibberish, scrolling to the very bottom of the file revealed the plaintext EICAR string.



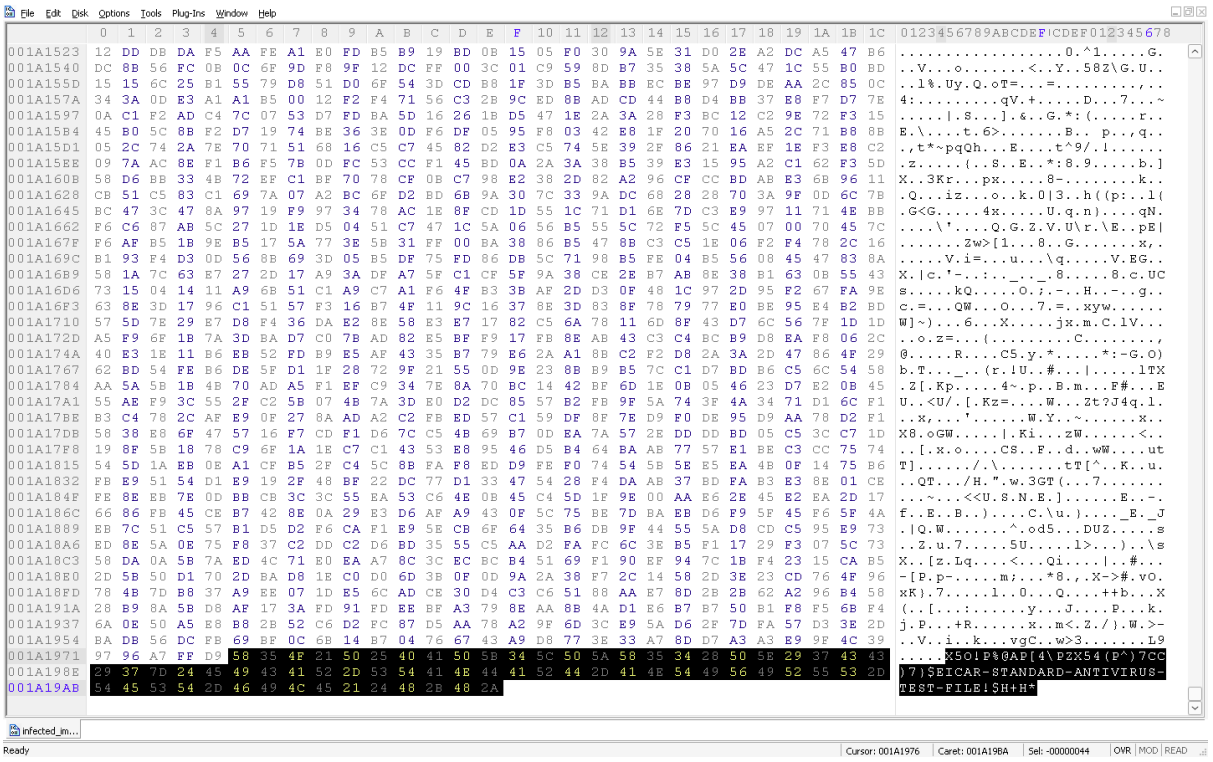
Observation

This method is functionally successful but forensically ineffective. The file size increased exactly proportionally to the size of the text file. Because the EICAR string is not encrypted or obfuscated, simple signature-based antivirus scans would detect it without difficulty.

Deep Dive: Hexadecimal Inspection

A Hex Editor (Hex Workshop) was used to analyse the file structure in order to confirm the forensic footprint of the "Append" attack.

- Standard JPEG Marker: The hexadecimal byte sequence **FF D9** marks the end of a valid JPEG file.
- Anomaly Detected: Data was seen to continue after the **FF D9** marker in infected_image_append.jpg.
- Forensic Significance: This "trailing data" is a key sign of compromise. This attack method is simple to automate against because automated forensic scripts can be written to flag any JPEG file that does not terminate at **FF D9**.



Experiment B: The “Embedding” Method (Steghide)

This method simulates Stegoloader's advanced tactics. Instead of attaching data to the end of the file, this method employs a steganography tool to embed the payload within the actual pixel data (often modifying the Least Significant Bits or DCT coefficients). The payload is encrypted and scattered, making it inaccessible to text editors.

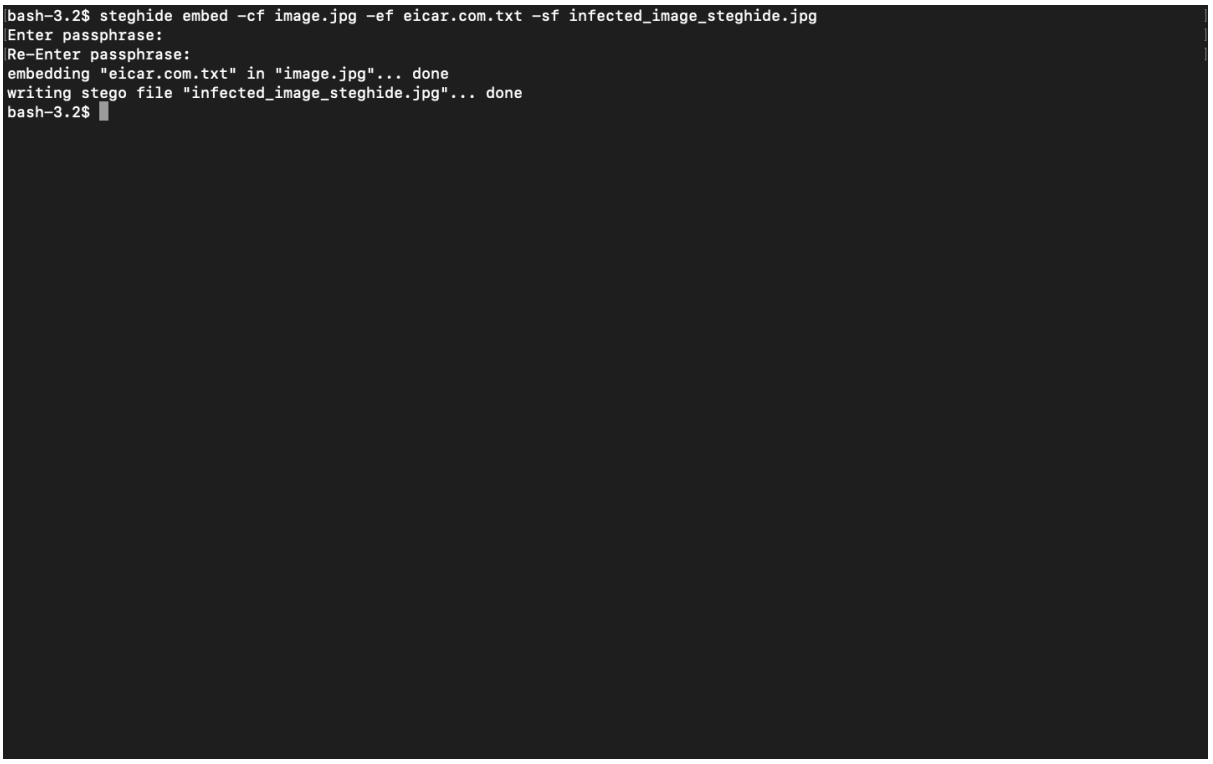
Procedure

Preparation: Used the Steghide command-line steganography program.

Execution: Used the embedding command to inject the EICAR payload into the carrier image:

- `steghide embed -cf image.jpg -ef eicar.com.txt -sf infected_image_steghide.jpg`

Encryption: Set a passphrase when prompted



Results

Visual Inspection: The file infected_image_steghide.jpg opens normally. To the naked eye, no visual degradation or "noise" is visible.

Forensic Analysis (Text): Opening the file in a text editor (or cat/nano command) provides no readable text. The EICAR string cannot be found through a search because it has been encrypted and integrated into the image data.

Forensic Analysis (Statistic): The image was examined using Aperi'Solve (or StegExpose). The analysis detected statistical anomalies in the bit layers, indicating that the image may contain hidden data.

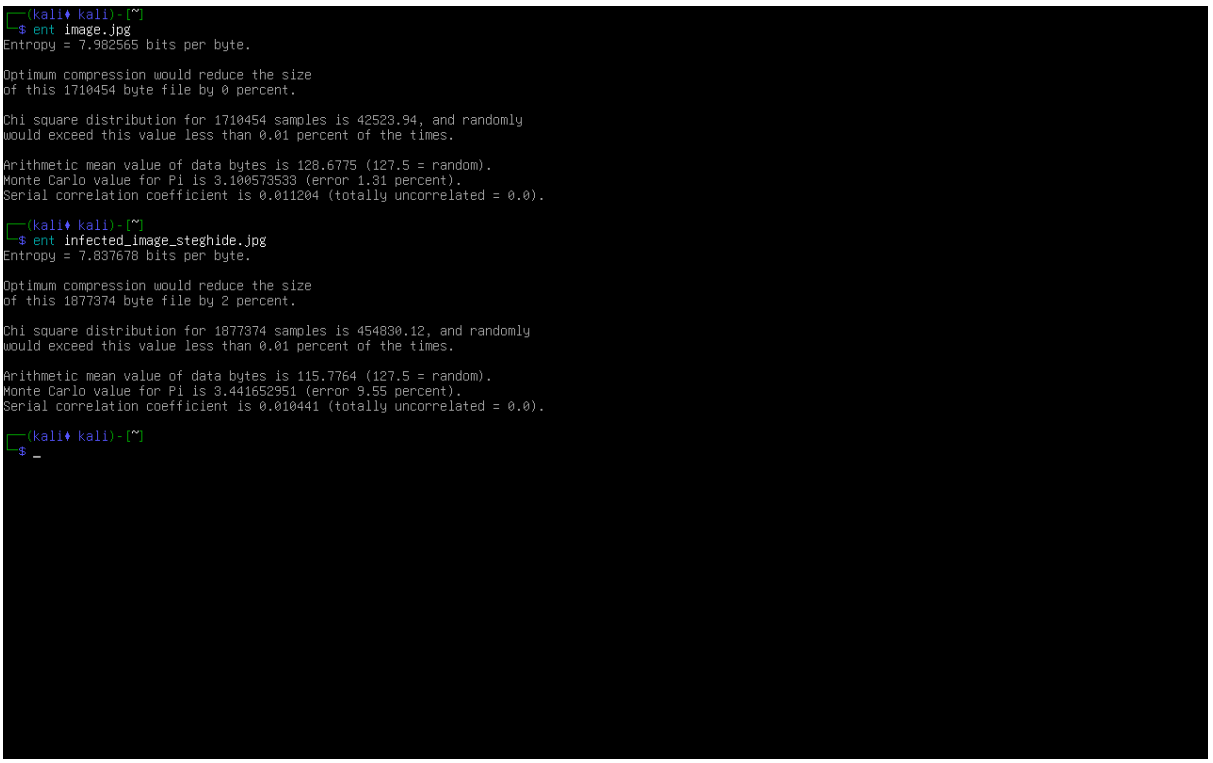
Observation

This method mimics the stealth of Stegoloader. Casual inspection and basic text scans fail to reveal the payload. To detect it, one must have knowledge of the specific steganography algorithm or advanced entropy analysis tools.

Deep Dive: Shannon Entropy Analysis

We used Shannon Entropy (measured on a scale of 0 to 8) to analyse the information density of the file because the "Steghide" method did not produce any visual artefacts or text signatures.

- Concept: JPEGs and other compressed images naturally have high entropy. On the other hand, encrypted data, such as the hidden payload, appears to be "perfect" random noise and frequently pushes entropy values to their highest point.
- Observation: The areas of the image containing the embedded payload displayed a flattened, maximum entropy variance when compared to the natural fluctuation of a clean image.
- Significance: This demonstrates that even though the malware is invisible to the human eye, mathematical analysis of the bit-distribution shows the existence of "unnatural" randomness, which is a characteristic of steganographic injection.



Comparative Analysis

Feature	**Method A: Appending**	**Method B: Embedding**
Complexity	Low (Native OS commands)	High (Requires specialized algorithms)
Visual Impact	None (Image looks normal)	None (Image looks normal)
Forensic Trace	High (Visible in text editor/Hex)	Low (Encrypted and scattered)
Detection	Easily caught by standard AV	Requires entropy analysis or specific keys

Phase 2: Detection Efficacy Analysis (VirusTotal)

Both the "Append" and "Steghide" samples underwent a multi-engine malware scan using VirusTotal in order to measure the efficacy of these steganographic techniques against contemporary defence mechanisms.

Sample A (Append Method)

Detection Rate: High (9/61 engines)

Analysis: Signature-based detection engines identified the file as malicious right away because the EICAR string was still in plaintext at the end of the file. This demonstrates that standard perimeter defences can detect "Append" attacks with ease.

Sample B (Embedding Method)

Detection Rate: Zero (0/60 engines)

Analysis: The payload was encrypted and the file's hash was changed by the steganographic embedding. 100% of antivirus engines were unable to identify the threat because the image structure looked legitimate and the EICAR signature was no longer visible in plaintext.

Conclusion

This shows that steganography successfully obfuscates the delivery mechanism, enabling the payload to get past initial security filters, even though it does not "break" the malware or alter its functionality.

Summary of Findings

Metric	**Method A: Appending**	**Method B: Embedding**
Visual Stealth	High (Invisible to eye)	High (Invisible to eye)
Forensic Signature	Obvious: Trailing data after FF D9	Subtle: High Shannon Entropy
Antivirus Detection	Detected immediately	Completely Evaded
Threat Level	Low	High

Conclusion of Primary Research

The experiments demonstrate that, while appending data is simpler, it provides only slight defence against detection. The Stegoloader approach (simulated here by Steghide) represents a much higher threat tier because it successfully creates a "blind spot" for traditional file analysis tools, forcing the malware to first "unpack" itself in memory before becoming active.

Steganalysis and Steganography Mitigation

In most cases steganography is not the base of an attack, but just one of its building blocks. For example in the attack on the Steam gaming platform (Caviglione, L. & Mazurczyk, W., 2022) it was used to carry resources that were required for the malware within a profile image. The actual body of an attack was executed via another exploit. In general, files suitable for steganography can't launch executable code on their own, so many cybersecurity specialists are not seriously concerned about its usage. This creates an uneven distribution of forces. Even though steganography is normally not used as a primary exploit, it is still a useful tool for an attacker. With the correct implementation of stenography prevention, attacks would be easier to detect, harder to implement with the use of other flows or even impossible. Caviglione, L. & Mazurczyk, W. (2022) state that most security systems are not designed to handle contents of media files out of the box, main reasons for that are lack of concern, difficulty of detection and no universal solution for all steganography techniques. In the case of steganography it is very difficult to find one solution for all of its variations, so if one attempt fails the attacker can switch the method or even create his own one. Some anti-steganography products are appearing, e.g. the Steganography Defensive Initiative of McAfee, but it is not scalable enough for use in large projects.

Covert Channels

Mitigation techniques are split into two groups based on types of covert channels[1] and well explained by Mazurczyk, W. and Caviglione, L. (2014), even though their paper primarily focuses on steganography in mobile devices, most of these techniques can be applied to devices of any type. The general process of shutting down an unauthorised culvert channel is called sanitisation, it must be done safely, ensuring all original processes and communications remain intact. For better understanding of context a brief explanation of couvert channel types is provided:

- Object culvert channels methods are the simplest and most researched ones, in the context of our research they mean the use of steganography within media files like images, audio, videos, text and even QR codes. The most popular ways are via images and audio files, they both use similar logic and are fairly simple to implement via LSB algorithm. As its name suggests, it is based on encoding data within least significant bits, but this is not efficient due to easy mitigation via rewriting those bits. There are many more efficient algorithms that distribute data all over the image and make steganography even more difficult to detect.
- Platform methods include communication between multiple entities, it is split into two subtypes based on the entity types.

[1] Covert Channels - hidden ways of sending information via untraditional means.

- Local covert channels are communication between two or more applications within a system, entity is a local application or process.
- Network covert channels are connections to external services i.e. servers, databases and devices via various channels such as bluetooth, wifi, mobile data or other networks. Entities are different devices.

This method is the primary target of malware, so the explanation will be more detailed. For instance, there are two entities A and B, A has no access to the network, but contains target data, B has access to the network but has no valuable data. A local covert channel could be used to establish the connection to entity B, but once data gets sent to an external host it can be detected by tools like Wireshark. This is when a network covert channel is needed to hide the data in an innocent looking package. Those methods have very high involvement in system processes, this makes execution of an attack detectable, though the process of malware acquisition by device can still remain stealth.

Mitigation of Object Steganography

According to Zeng et al. (2022), object culvert channels mitigation is mostly based on slight change of the data (e.g. image resizing or audio compression), though there is still no universal solution, this is implemented by default in most modern social networks. Interestingly, in most cases steganography is not a primary reason for such measures, it is done to minimize the amount of traffic and speed up processes within an app and steganography prevention is just a pleasant bonus. This kind of practice forced steganography enthusiasts to develop multiple techniques to withstand alteration, most of these are based around error correction and reduction of channel error rate. Most common one is ECC (Error Correction Codes), it allows to reconstruct the message even if some parts of it are missing, its disadvantage is that robustness is still limited, so it is often used in combination with other methods like reduction of channel error rate, for example in JPEG compression there are properties and areas of image that are not affected by compression as much as others, if those are detected and utilised in bond with ECC probability of successful message transfer is increasing rapidly.

This brief description of bypassing media alteration was to showcase that even though this seems like a solution to all steganography threats in media, in fact Zeng et al. (2022) states that it is not universal due to vast amounts of techniques to avoid it and new ones are being developed continuously. Also in many cases alteration cannot be done due to high sensitivity of a file, network package for example. Mitigation ways described above are blind, since we do not determine if the steganography is present, we just alter all the files that are accessing the system. The problem with such an approach is that there will be no notification of an attack attempt so we mostly rely on luck that there is no method to bypass our alteration. We don’t block the source of the threat so attacker can change methods to strike

again. For that reason steganalysis[2] is used to determine files that may contain steganography in order to implement more precise measures like restriction of file upload or blockage of its source or even message retrieval for further analysis in some cases. Methods of steganalysis are classified by a type of media they attempt to find steganography in and whether an algorithm is known. There are two base method types for steganography detection, those are signature and statistical detection techniques. Signature ones are based on recognising familiar patterns within media, they require some previous examples where a specific algorithm was used and it is its main drawback, since it may be inefficient against unpopular or new algorithms, though it still is used for detection of mature ones like LSB. Statistical algorithms are a primary tool for detection of all algorithms, no matter if they are modern or classic, they are based on comparison of a dataset of clean images against a sample that potentially contains stenography. There are many such algorithms both for object and network covert channels, the difference between those are mainly parameters that are being compared and ways of their computation. In many cases the biggest drawback of these methods is scalability, since a separate analysis of each media file is required.

Analysis of images is the most researched topic, most techniques would detect LSB algorithm, coverage of other algorithms depends on their logic. For example, Fridrich Method would only detect LSB, but Histogram Characteristics Function would also work on spread spectrum and DCT-based steganography. F5 algorithm is well known for its efficiency and secrecy, it is based on matrix encoding allowing to evenly spread data over the document. There is a method created solely for its detection, it compares histogram of four by four blocks to identify changes.

Audio is somewhat similar to pictures, LSB detection is pretty much the same, there are some techniques unique to audio, such as frequency-masking and echo hiding that are not detectable for the human ear. These can be detected with generic algorithm classifiers or mel-frequency spectrum. The first option is based on statistics and some implementations use artificial intelligence to analyze datasets, need for those is the biggest drawback. The second option works with short term power spectrums to retrieve artifacts and anomalies. Also there are algorithms to detect anomalies within specific audio compression types.

Videos are difficult to work on when it comes to steganography, because of a need to work with multiple frames and audio. There are not so many attacks with the use of videos, but there are still some techniques to detect steganography within those. Mostly those are based on analysis of sequential frames one at a time for presence of LSB manipulation. Another known example is

[2] Steganalysis - process of steganography discovery.

3D-DCT analysis which uses 3D discrete form for analysis and is usually used for youtube compressed videos.

Text files of various types are also used for steganography, though are not a popular solution in real world due to low extent of stealthiness. Techniques mainly focus on spacing, characters, fonts and special characters, but there are also examples of linguistic analysis with the use of artificial intelligence.

Mitigation of local Covert Channels

The most popular and fundamental solution that implements local culvert channel mitigation is TaintDroid, it is an application that detects unauthorized communication between applications and stops it. Its main problem is the definition of rules to ensure that it blocks only malicious connections. It is based on marking sensitive data and tracking its usage. More systems were developed based on it, such as QuantDroid and XManDroid, as it follows out of names they all operated on android OS. Some of these also monitor API usage by applications. Other approaches include threshold-based approaches that analyze data without marking it, but by tracking its history.

Mitigation of Network Covert Channels

Mitigation of network covert channels that use steganography is an interesting topic, since it includes all devices that use TCP/IP architectures allowing solutions to impact a wide range of users. According to Mazurczyk, W. and Caviglione, L. (2014), those techniques can be split into 3 groups:

- network traffic normalizers are used for removing unusual activities in traffic, for example inter packet time alteration that could be used to encode secret data with steganography. Normalizers may capture context of a package or not, based on that we can split those into two subgroups:
    - stateless - only the packet is analyzed.
    - stateful - scope of analysis is beyond one packet.
        - network-aware - stateful warden with network topology knowledge

  As an example of stateless system, a network warden concept can be introduced, it is stated by Lewandowski, Lucena and Chapin (2006) that such an option can eliminate many of the known covert channels without significantly impacting the overall system (messaging was around 3% slower than usual). Those are resembling firewall, they modify all traffic no matter if steganography is there or not. It is making alterations that don’t damage the contents, in most cases by zeroing padding bits in TCP packets. Those are very simple and do not contain

any topology information. Stateful/network-aware solutions are using active mappers. All such solutions use SNMP in one form or another, some applications can even be considered a subform of this protocol. Due to high efficiency there is a variety of such services. Those are actively sending special probing packets to map the network and determine what kind of communication is appropriate. Network-aware wardens are more precise over stateless ones, it allows for more performance and accuracy.

This kind of mitigation has multiple disadvantages, those are: loss of functionality due to unavailability of some parts of protocol headers, lack of scalability due to a required buffer for processing that can be overwhelmed by large amounts of traffic, routes can use different routes to end up in the same destination. A workaround of using those in different locations of a network is possible, but it leads to even larger loss of scalability and maintainability.

- Statistical methods are based on multiple types of analysis of clean traffic and steganographically modified one. Differences are the parameters that are being examined. For example Berk et al. (2005) proposes two techniques to achieve results using this method. First one requires an attacker to transfer as much data as possible, so that when computed data size within an interpacket delay would differ from average values. Another one is detecting a sequence of bystanding anomaly values within interpacket delays. There are multiple other approaches, two more interesting ones were proposed by Cabuk et al. (2009). The main one is exploitation of data compression applied to inter-arrival time, it is done in three steps. Inter-arrival time is recorded, values are converted to strings and those are compressed. Compression tends to be more efficient on strings that were computed out of inter-arrival time recorded from sessions with covert channels. This effect happens due to covert channels making average time more even, this allows for compression to be more efficient due to repeating strings. The second one is based on comparing standard deviation of recorded values, this exploits the same capability of polluted traffic.

- Machine learning approaches are gaining popularity with fascinating speed, they are precise and can combine multiple techniques at once. A major disadvantage is a need for a major dataset and difficult troubleshooting. Differences between those mainly depend on a type of machine learning used, while some use older decision making trees, some are based on SVMs[3].

[3] Support Vector Machines - a type of machine learning algorithms based on geometry of data rather than logic.

Conclusion

In summary, the investigation revealed how modern malware, exemplified by Stegoloader and the recent ClickFix campaign, uses digital steganography to bypass traditional disk protections by embedding malicious payloads directly into image or SVG files and executing them entirely in memory.

The experimental comparison between simple “append” techniques and sophisticated embedding tools such as Steghide further demonstrated that encrypted, dispersed payloads drastically reduce forensic traceability, confirming the elevated threat tier of steganographic attacks .

Future research should therefore concentrate on three complementary fronts.

- First, advancing statistical and machine-learning steganalysis methods that can reliably distinguish subtle LSB or DCT anomalies across diverse media types, thereby reducing the reliance on blind-alteration tactics.
- Second, developing robust mitigation frameworks for both object-level and network-level covert channels—such as adaptive traffic normalizers and selective packet-padding strategies—to close the residual gaps that attackers exploit .
- Third, constructing realistic, sandboxed testbeds that emulate the in-memory extraction and execution pathways observed in Stegoloader and ClickFix, enabling security analysts to evaluate detection pipelines against live, multi-stage stegomalware scenarios.

By integrating deeper statistical analysis, proactive channel sanitisation, and dynamic behavioural testing, the next generation of forensic tools will be better equipped to uncover hidden payloads before they reach execution, ultimately narrowing the “blind spot” that steganography currently provides to adversaries.

References

1. Caviglione, L. & Mazurczyk, W. 2022. ‘Never mind the malware, here’s the stegomalware’, IEEE Security & Privacy, 20(5), pp. 101-106.
2. Unit, D.S.C.T. and Intelligence, T., 2015. Stegoloader: A stealthy information stealer. URL: https://www.secureworks.com/research/stegoloader-a-stealthy-information-stealer [access: 3.12. 2025].
3. Mazurczyk, W. and Caviglione, L. 2014. ‘Steganography in modern smartphones and mitigation techniques’, *IEEE Communications Surveys & Tutorials*, 17(1), pp. 334–357.
4. Zeng, K., Yang, C., Yang, H., Zhu, X. and Weng, S., 2022. Improving robust adaptive steganography via minimizing channel errors. *Signal Processing*, 195, p.108498.
5. Lewandowski, G., Lucena, N.B. and Chapin, S.J., 2006. Analyzing network-aware active wardens in IPv6. In: International Workshop on Information Hiding. Berlin, Heidelberg: Springer Berlin Heidelberg.
6. Berk, V., Giani, A. & Cybenko, G., 2005. Detection of covert channel encoding in network packet delays.
7. Cabuk, S., Brodley, C.E. & Shields, C., 2009. IP covert channel detection. *ACM Transactions on Information and System Security (TISSEC)*, 12(4), pp.1–29.
8. Johnson, N.F. and Jajodia, S., 2023. Steganalysis of images created using current steganography software. Center for Secure Information Systems, George Mason University. Available at: https://cisre.egr.uh.edu/wp-content/uploads/2023/09/steganalysis.pdf [access: 4.12.2025].
9. Solanki, K., Sarkar, A., Manjunath, B.S. 2007. 'YASS: Yet Another Steganographic Scheme That Resists Blind Steganalysis'. In: Furon, T., Cayre, F., Doërr, G., Bas, P. (eds) Information Hiding. IH 2007. Lecture Notes in Computer Science, vol 4567. Springer, Berlin, Heidelberg.
10. Huntress Labs. 2025. 'ClickFix Gets Creative: Malware Buried in Images', Huntress Blog. Available at: https://www.huntress.com/blog/clickfix-malware-buried-in-images
11. Proofpoint. 2016. 'Massive AdGholas Malvertising Campaigns Use Steganography and File Whitelisting to Hide in Plain Sight'. Available at: https://www.proofpoint.com/us/threat-insight/post/massive-adgholas-malvertising-campaigns-use-steganography-and-file-whitelisting-to-hide-in-plain-sight [access: 4.12.2025]
12. Microsoft Threat Intelligence. 2025. 'Think before you ClickFix: Analyzing the ClickFix social engineering technique', Microsoft Security Blog. Available at: https://www.microsoft.com/en-us/security/blog/2025/08/21/think-before-you-clickfix-analyzing-the-clickfix-social-engineering-technique/ [access: 4.12.2025]
13. Trend Micro. 2017. 'AdGholas Campaign Employs Astrum Exploit Kit', Trend Micro Research. Available at: https://www.trendmicro.com/en_us/research/17/f/adgholas-malvertising-campaign-employs-astrum-exploit-kit.html [access: 4.12.2025]