Authors: Mark R. Hedrick, Justin O. Holman
Title: "StegArmory: Offensive Cyber Security Software for Embedding Shellcode in Images"
Affiliation: Hasan School of Business, Colorado State University, Pueblo, CO USA
Corresponding Author: Justin O. Holman, justin.holman@csupueblo.edu

Abstract:
The paper introduces StegArmory, a new open source software package with practical applications for offensive cyber security operators. StegArmory uses steganography techniques to embed machine code, or shellcode, in images. Shellcode is typically flagged as malicious by antivirus software due to the payloads they often contain, but detection becomes more difficult when shellcode is embedded in a common image file. Using steganography to embed shellcode within portable network graphic (PNG) images, StegArmory provides a new way to avoid detection of potentially malicious payloads while ensuring reliable transmission. In this paper, the StegArmory software development process is described, performance benchmarks are established and detection metrics are measured using sample cover images. Two image-based steganography techniques are utilized, least significant bit (LSB) and pixel value differencing (PVD). Test results indicate the software effectively produces PNG image files, using both LSB and PVD approaches, with embedded shellcode capable of avoiding malicious payload detection. The LSB method is faster but the PVD method handles larger payloads and image modifications are more difficult to detect.

Keywords: steganography, *least significant bit (LSB), pixel value differencing (PVD), threat detection*

Declarations:
Funding: Not applicable
Conflicts of interest/Competing interests: Not applicable
Availability of data and material: https://github.com/MarkoH17/StegArmory
Code availability: https://github.com/MarkoH17/StegArmory
Author contributions: Hedrick developed the software and wrote an initial draft of the paper. Holman supervised the project and edited the paper.

Research Square
Preprints are preliminary reports that have not undergone peer review. They should not be considered conclusive, used to inform clinical practice, or referenced by the media as validated information.

1 INTRODUCTION

Although steganography has been around for thousands of years, it still has plenty of practical uses in cyber security. Simply put, steganography is the practice of concealing data in an object. Throughout history, steganography has played a significant role in transmitting hidden messages, though other use cases exist, which may include copyright protection, private communications, and data exfiltration. This project focuses on utilizing various steganography techniques to embed machine code (shellcode) in images. Shellcode is used in offensive cyber security engagements to execute low-level functions on a computer, which may enable an operator to gain remote access or exploit obscure bugs in the host computer's operating system, among other possibilities. Most shellcode is flagged as malicious by antivirus software and other threat detection systems like firewalls because they often contain malicious payloads. To remain effective, it is important for offensive security operators to develop new ways of evading these defenses.

Evading detection by firewalls and antivirus is important to ensure reliable execution of the shellcode by the victim computer. Steganography with portable network graphic (PNG) images and malicious shellcode provides a new way to avoid detection while ensuring reliable transmission. PNG images are almost never filtered out by network firewalls or flagged by antivirus software because they are an integral part of web browsing and computer applications. Blocking these files would prevent most websites and computer applications from properly functioning. Because PNG images are (currently) immune to nearly all defensive mechanisms and are saved with lossless compression, they were chosen as the cover object for the steganography methods used in this project.

The purpose of this paper is to present the StegArmory software and describe benchmarking and detection testing results for a cyber security practitioner audience. As such, a complete literature review is beyond the scope of this paper but we refer the reader to Hussain et al. [2] for a comprehensive review of spatial stenography. The StegArmory software is open source and available for download [1] from a GitHub repository.

2 SOFTWARE DEVELOPMENT

StegArmory, the software developed for this project, enables users to embed and extract payloads from images. StegArmory supports two image-based steganography techniques: least significant bit (LSB), and pixel value differencing (PVD). LSB steganography involves manipulating the least significant bit for the red-green-blue (RGB) values that make up every pixel in the cover image, until the entire payload is embedded. In other words, this method embeds one bit of the payload at a time, where each pixel can store 3 bits in total.

According to Sinha [3], LSB steganography is the easiest and most adaptable image-based steganography technique. PVD steganography is more involved and was originally identified by Wu and Tsai [5]. The PVD method iterates over the RGB values that make up each pixel and determines how many bits of the payload to embed in these values based on their visual similarity. Lower similarity allows for more bits to be embedded (up to 5 bits at a time), whereas higher similarity allows fewer bits to be embedded. Embedding 1 bit of data in the RGB values results in the least amount of change to a pixel, whereas embedding 5 bits of data results in the largest distortion. Also, if the addition of the bits results in an invalid pixel value, the pixel pair is skipped. RGB values can range from 0 to 255, and values outside this range are discarded. In theory, this method allows for a higher density of data to be embedded, though it is dependent on the composition of the cover image. The PVD method is also much slower to embed data due to all the checks that take place when determining how much data to embed in a pair of pixels. Between the two methods, LSB is more performant, but is easier to detect (as all pixels are modified). Similarly, PVD is less performant, but is harder to detect, and has the potential to be more space efficient compared to the LSB method. Since one of the objectives of this project was to identify the most efficient and least detectable image-based steganography technique, these results will be discussed in greater detail below. However, it is important to introduce two metrics that were used to evaluate the quality of the output (processed) image. The first metric, structural similarity index measure (SSIM) is used to quantify the perceived quality of the image. Ranging from 0-1, higher SSIM values represent better image quality. The second metric, peak signal-to-noise ratio (PSNR) is similar in that it represents the amount of distortion between images. Ranging from 0-100, a lower PSNR value represents a less distorted image. PSNR however, is more sensitive to even the smallest changes in an image, whereas SSIM only accounts for visually detectable changes. Together, these two measures make it possible to evaluate the quality of images processed using either of the steganography methods in StegArmory.

Developing the LSB and PVD steganography methods in StegArmory presented several challenges that were not obvious in the planning stages. Accessing the RGB values that comprise the pixels in an image was difficult due to the encoding used in the PNG file format. Using the OpenCV2 library in Python made it possible to overcome this challenge without abandoning too much of the original codebase. Corner-case bugs that manifested when embedding and extracting data using the PVD method also presented many challenges. These obscure bugs were difficult to identify and remediate because they only appeared when using StegArmory with certain payloads and cover images. These bugs were eventually addressed by referring to other implementations of the PVD method developed after the original publication by Wu and Tsai [5], as discussed by Tseng and Leng [4]. Performance was also an issue with both the LSB and PVD methods. Since the LSB method was developed first, many simple data structures were used to store and manipulate pixel data. Switching these structures to arrays provided by the NumPy library reduced the code complexity and increased performance of the LSB processing method. These structures were later used in the PVD processing method, though the PVD

method was still less performant than desired. The PVD algorithm requires multiple iterations over pixel pairs along with a series of conditional checks, which greatly slows down the processing compared to the LSB method. The performance challenges with the PVD method could be solved by implementing a threaded approach to the data processing, though such modifications would have added unnecessary complexity for a security-focused utility. One of the other major challenges faced in the development of StegArmory was staying organized. A virtual Kanban Board was created on Atlassian's Trello to keep track of the backlog of items, in progress items, in test items, and completed items. This helped keep the project organized, and prevented additional, unnecessary features from being added to the project – something that is easy to do on an open-ended project. All major challenges were overcome by the end of the project, which enabled the first stable release of StegArmory to be published to GitHub on December 1st, 2020 [1].

3 PERFORMANCE BENCHMARKS

Three images of varying sizes and one malicious payload were used to benchmark the performance of StegArmory. The benchmark code tested the images and payload against the different steganography methods to identify the pros and cons of each steganography method. The test images are displayed in Figure 1 and test results are listed in Figure 2.

File: earth.png File size: 2.38 MB Pixels: 2,073,600	
File: lake.png File size: 1.96 MB Pixels: 2,073,600	

File: car.png File size: 1.90 MB Pixels: 2,073,600	

Fig 1 Cover images.

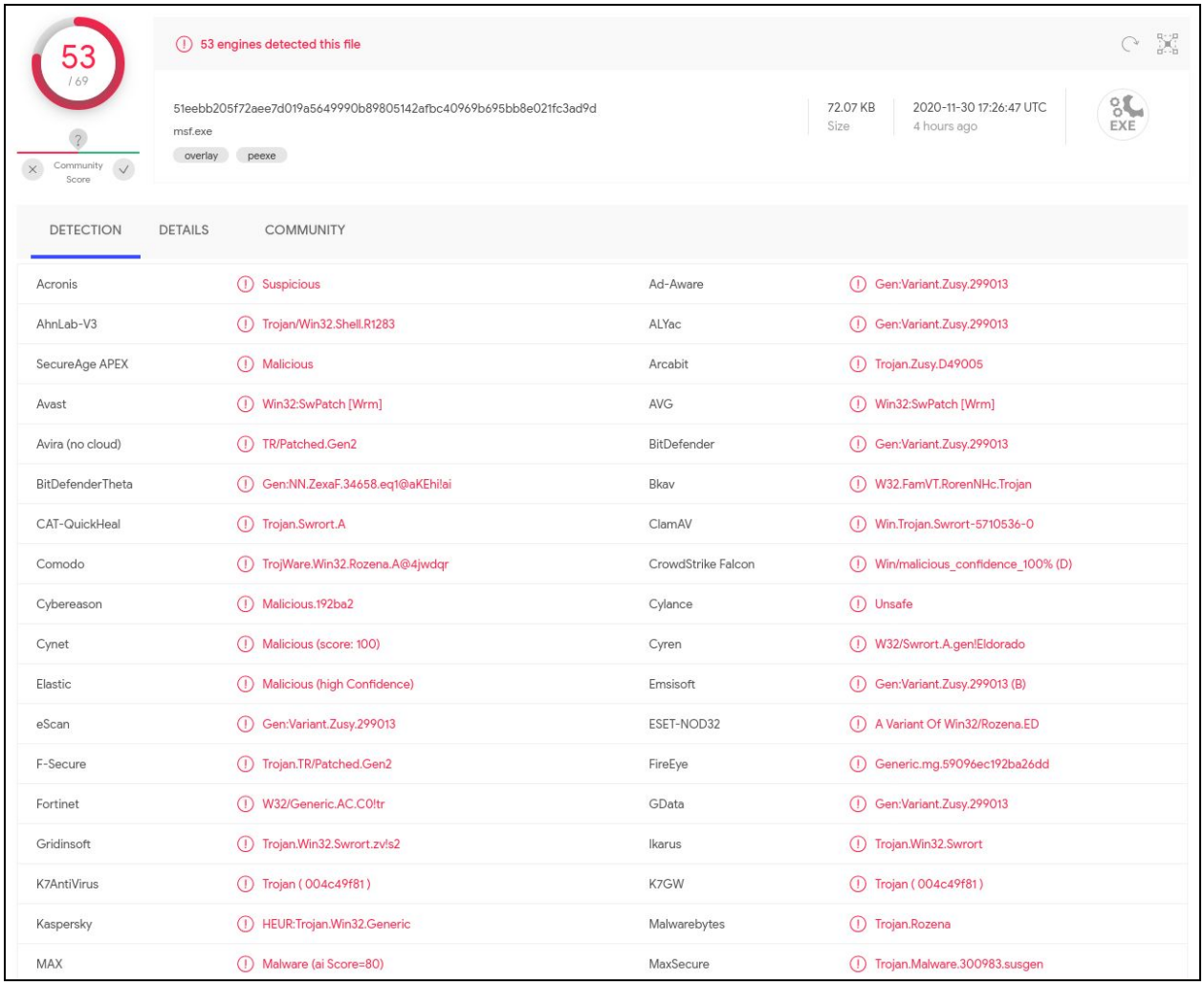
Image	*Method*	*Embed Time(s)*	*Extract Time (s)*	*Payload (MB)*	*SSIM*	*PSNR*
Earth	LSB	2.00	1.51	0.78	0.9977	63.0201
Earth	PVD	61.15	10.94	0.78	0.9984	58.9514
Lake	LSB	2.00	1.52	0.78	0.9994	61.3676
Lake	PVD	51.70	3.40	1.21	0.9994	45.9679
Car	LSB	1.96	1.51	0.78	0.9995	61.0684
Car	PVD	55.86	7.76	0.48	0.9963	51.8515

Fig. 2 Benchmark Results.

The embed and extract times in Figure 2 make it clear that the LSB method is much faster than PVD. In most scenarios, more data can be embedded in the cover image using the PVD method. The SSIM and PSNR metrics for the images introduce the idea that an image can see a lot of change over fewer pixels (PVD method), but still look very similar (SSIM). In most cases, the PVD processed images have a higher (better) SSIM value than images processed with the LSB method, and lower (better) PSNR values. These results indicate that to embed the largest payloads with the lowest detection rates, PVD steganography should be used. To process data the images as quickly as possible with decent output quality, LSB steganography should be used.

4 THREAT DETECTION

Another objective of this project was to evade detection by the top antivirus (AV) software. To verify that the test payload embedded with StegArmory could evade popular AV engines, the malicious payload was uploaded to VirusTotal for scanning by nearly 70 different AV programs. The results of the scan are displayed in Figure 3.



53 / 69

53 engines detected this file

51eebb205f72aee7d019a5649990b89805142afbc40969b695bb8e021fc3ad9d

msf.exe

overlay peexe

72.07 KB Size

2020-11-30 17:26:47 UTC 4 hours ago

EXE

Community Score

DETECTION DETAILS COMMUNITY

Acronis	Suspicious	Ad-Aware	Gen:Variant.Zusy.299013
AhnLab-V3	Trojan/Win32.Shell.R1283	ALYac	Gen:Variant.Zusy.299013
SecureAge APEX	Malicious	Arcabit	Trojan.Zusy.D49005
Avast	Win32:SwPatch [Wrm]	AVG	Win32:SwPatch [Wrm]
Avira (no cloud)	TR/Patched.Gen2	BitDefender	Gen:Variant.Zusy.299013
BitDefenderTheta	Gen:NN.ZexaF.34658.eq1@aKEhi!ai	Bkav	W32.FamVT.RorenNHc.Trojan
CAT-QuickHeal	Trojan.Swrort.A	ClamAV	Win.Trojan.Swrort-5710536-0
Comodo	TrojWare.Win32.Rozena.A@4jwdqr	CrowdStrike Falcon	Win/malicious_confidence_100% (D)
Cybereason	Malicious.192ba2	Cylance	Unsafe
Cynet	Malicious (score: 100)	Cyren	W32/Swrort.A.gen!Eldorado
Elastic	Malicious (high Confidence)	Emsisoft	Gen:Variant.Zusy.299013 (B)
eScan	Gen:Variant.Zusy.299013	ESET-NOD32	A Variant Of Win32/Rozena.ED
F-Secure	Trojan.TR/Patched.Gen2	FireEye	Generic.mg.59096ec192ba26dd
Fortinet	W32/Generic.AC.C0!tr	GData	Gen:Variant.Zusy.299013
Gridinsoft	Trojan.Win32.Swrort.zv!s2	Ikarus	Trojan.Win32.Swrort
K7AntiVirus	Trojan (004c49f81)	K7GW	Trojan (004c49f81)
Kaspersky	HEUR:Trojan.Win32.Generic	Malwarebytes	Trojan.Rozena
MAX	Malware (ai Score=80)	MaxSecure	Trojan.Malware.300983.susgen

Fig 3 Malicious payload VirusTotal scan results.

Of the 69 virus engines available for scanning, 53 of them (76%) correctly identified the malicious file. The malicious file was then embedded in each of the cover images using StegArmory, and the processed images were uploaded to VirusTotal to see how well the payload was hidden. The results for the processed images are tabulated in Figure 4.

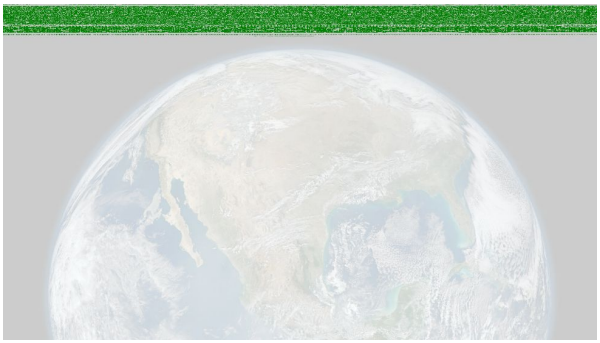
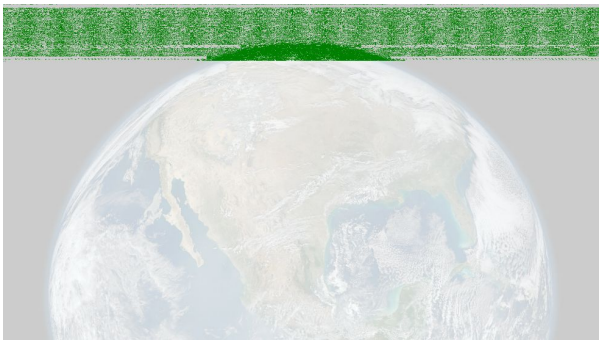
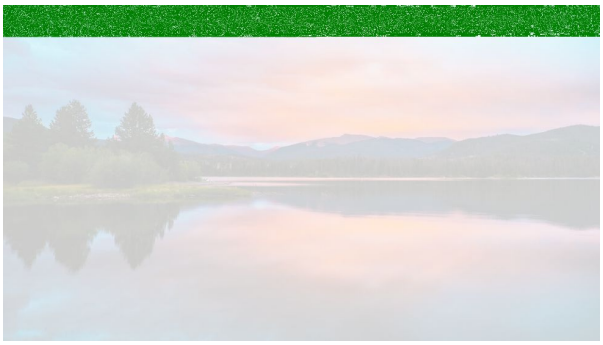
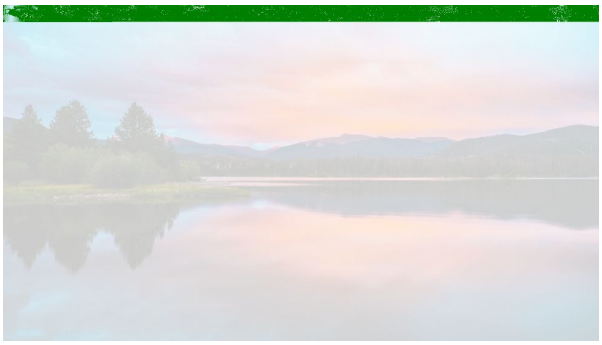
Image	*Method*	*Detection Rate*	*Link to Results*
Earth	LSB	0%	Results
Earth	PVD	0%	Results
Lake	LSB	0%	Results
Lake	PVD	0%	Results
Car	LSB	0%	Results
Car	PVD	0%	Results

Fig. 4 StegArmory image VirusTotal scan results.

The detection rate for all processed images was found to be 0%. This indicates that regardless of the method used, no AV software was able to detect the malicious payload embedded in the cover image.

5 VISUAL COMPARISON

Next, in an effort to visualize the miniscule changes made by embedding the data, the [Online Image Difference tool](#) was used to highlight the difference in pixel values between the original and processed images. The green overlay within the images in Figure 5 highlight the pixels that have changed between the original and StegArmory processed image. A lighter shade of green (or absence of color) represents minimal or no change, and a darker shade represents more change.

 File: earth.png Method: LSB	 File: earth.png Method: PVD
 File: lake.png Method: LSB	 File: lake.png Method: PVD

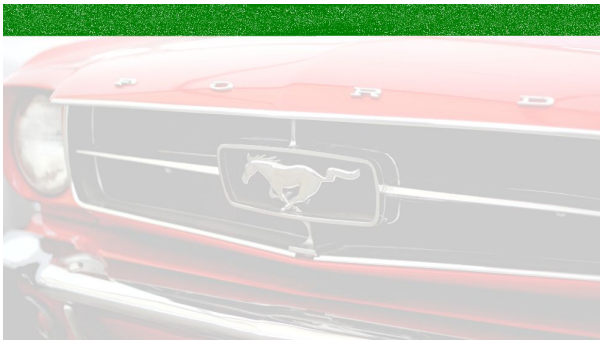
 File: car.png Method: LSB	File: car.png Method: PVD

Fig. 5 Image comparison.

As expected, the LSB method consistently modified a block of pixels until the entire payload was embedded. The PVD method also modified pixels until the entire payload was embedded but it modified them by a variable amount only if the change would not be visually noticeable. For example, the car.png image in Figure 5 shows how the PVD method skipped over some pixels in the middle of the picture because changing them would have been too noticeable. Viewing these differences makes it clear that compared to the LSB method, the PVD method spreads the changes over a larger area and skips some pixels when necessary. In practice, the output produced from StegArmory using either encoding method created images that were nearly identical to the original. Minor visible artifacts were observed only after zooming in by a factor of 300%.

5 CONCLUSION

StegArmory provides an effective method for concealing a malicious file in an image and using the image to make the hidden data available inside a protected network. Overall, the benchmark, threat detection and visual similarity overlay results indicate that both the PVD and LSB methods may be useful under various circumstances. The LSB method is much quicker to embed and extract data but resulting image differences are easier to detect. In contrast, the PVD method is slower to process data but it facilitates larger payload capacity and the resulting image differences are more difficult to detect. Finally, as an open source project, StegArmory enables security professionals to duplicate, modify, and redistribute the software for their own purposes.

REFERENCES

1. Hedrick, M.R. (2020). *StegArmory*. https://github.com/MarkoH17/StegArmory

2. Hussain, M., Wahab, A.W.A., Idris, Y.I.B., Ho, A.T.S., Yung, K.H. (2018). Image Steganography in Spatial Domain: a Survey. *Signal Processing: Image Communication*, 65: 46-66.

3. Sinha, B. (2015). Comparison of PNG & JPEG Format for LSB Steganography. *International Journal of Science and Research, 4*(4): 198-201.

4. Tseng, H., & Leng, H. (2013). A Steganographic Method Based on Pixel-Value Differencing and the Perfect Square Number. *Journal of Applied Mathematics, 2013*: 1-8.

5. Wu, D., & Tsai, W. (2003). A steganographic method for images by pixel-value differencing. *Pattern Recognition Letters, 24*(9-10): 1613-1626.