IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-12, Dec 2019 E-ISSN: 2347-2693

# An Efficient Technique to Detect Stegosploit Generated Images on Windows and Linux Subsystem on Windows

**N. Vaidya[1*], P. Rughani[2]**

[1,2]Institute of Forensic Sciences, Gujarat Forensic Sciences University, Gujarat, India

**Corresponding Author: neerad.dfis1732@gfsu.edu.in, Tel: +91 9427418639*

DOI: https://doi.org/10.26438/ijcse/v7i12.2126 | Available online at: www.ijcseonline.org

Accepted: 01/Dec/2019, Published: 31/Dec/2019

***Abstract*** - Steganography as being a very useful technique for content hiding is the first choice of criminals, terrorists, and hackers. The steganalysis itself is very complex, and lots of research work is going on all around the world on steganography and steganalysis. However, when the steganography hides exploit instead of simple messages, it becomes more severe and damaging. Stegosploit is a similar toolkit that allows hackers to inject exploits for known vulnerabilities into images. These images, when accessed or downloaded can infect a machine very effectively compared to other ways of doing it. This paper emphasis on a technique that detects such stego images having an exploit inside it. We developed a script that detects this type of image, which is in-general not identified by known anti-viruses including virus total. The study also focuses on the effectiveness of the script for the Windows operating system and Linux Subsystem on Windows. The script derived from this research will help end-users, security professionals, forensic investigators, and researchers in detecting and thus preventing possible cybercrimes.

***Keywords*** - Steganography, Steganalysis, Stegosploit, Exploit Detection, Image Steganography, Image Exploits, Polyglots.

## I. INTRODUCTION

Content hiding is one of the significant aspects of security as it ensures confidentiality in CIA triad. However, the techniques used for content hiding is equally being used by hackers, criminals, terrorists, etc. for different purposes. The objective is the same, and methods or algorithms designed for covert communication allow us to achieve the same by both good and evil. A variety of techniques like cryptography are used in achieving confidentiality, but the most effective and widely accepted method is steganography.

Steganography as the name suggests refers to hidden or covered writing which allows users to hide any sort of information in cover media. The cover media can be any multimedia file like image, audio or video[1]. Based on content and intention user can use any of the possible types as cover media to hide content. Various free and commercial tools are available in the market which allows any laymen to hide content inside cover media without any technical knowledge. These tools basically follow certain algorithms for information hiding and retrieval [2][3]. The most common algorithm/technique used in steganography is LSB, which is explained briefly in the next paragraph.

The Least Significant Bit (LSB) algorithm works on substitution of the Least Significant Bit of the image with the bit of the message to hide. As an only Least significant bit of the pixels gets replaced by the bit of the information to hide, the overall image does not get affected and the change cannot be noticed with the naked eye. This is highly effective in color images with high resolution. The same may not be so effective if the images are black and white or greyscale with lower resolution [4][5].

The steganography technique is different than digital watermarking as in the later the watermarks remain visible and are not used for data hiding [6][7]. Similarly, steganography does not replace cryptography as cryptography makes the content non-readable but the content remains visible. While in steganography the content is invisible and end-user cannot detect the possibility of having content inside the cover media. On another hand, if cryptography is used with steganography then it can make things more undetectable as the hidden content will be encrypted and the user will need to crack two algorithms [8][9][10].

This paper focuses on more difficult to handle steganography where instead of plain or encrypted text an exploit is hidden inside the image. The work is based on stegosploit a toolkit, which is useful for delivering exploit using Steganography. As mentioned by the author, Stegosploit is a portmanteau of *Steganography* and *Exploit*. Using Stegosploit, it is possible to transform virtually any

 

JavaScript-based browser exploit into a JPG or PNG image [11]. The author worked on HTML + Image Polyglots to embed exploit for a known vulnerability CVE-2014-0282. The toolkit has been made powerful enough to embed exploits for browser-based vulnerabilities in any JPG or PNG files.

While we learned about stegosploit, we found it deadliest in all the steganographic techniques. The worst thing in this is to detect the presence of any exploit inside an image compared to that without hiding it using stegosploit. The stegosploit generated image can infect any machine when it is accessed through the vulnerable web browser. Since, loading images in the browser while surfing is a routine task, the end-user may not be even able to know that the machine got infected simply by an image opened in the browser.

The things may not work if the system and the browser are patched and updated for the latest vulnerabilities. But it is observed that only a few users update their applications and operating systems on a regular basis [12]. This led us to work on this challenge as the detection of stegosploit images can prevent many unwanted attacks and even losses.

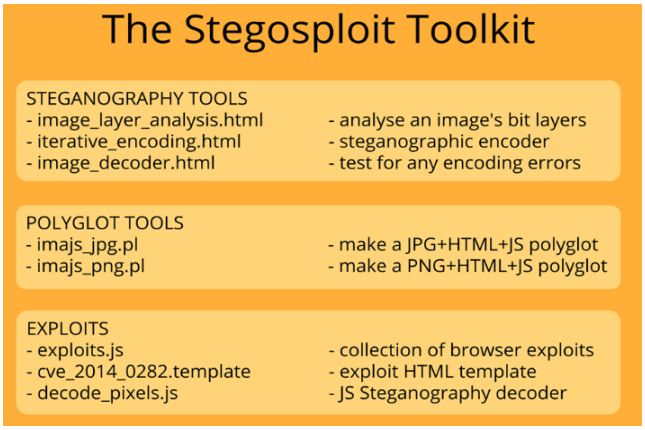

Figure 1: Understanding Stegosploit Toolkit

The above figure illustrates the tools of the stegosploit toolkit. iterative_encoding.html is used for encoding browser exploit code in image steganographically, image_decoder.html is used to detect any possible error in image, imajs_jpg & imajs_png.pl are pearl scripts which make encoded images into polyglot images using auto decoder-script for jpg and png files respectively. decode_pixels.js is a javascript script that automatically executes upon loading of polyglot image. exploits.js is a collection of browser exploits and cve_2014_0282.templete is a sample exploit for CVE-2014-0282. CVE-2014-0282 is IE Use-After-Free vulnerability in Microsoft's Internet Explorer version 6 to 11. CVE-2014-0282 is Internet Explorer Memory Corruption Vulnerability, which allows remote attackers to execute arbitrary code via a crafted website.

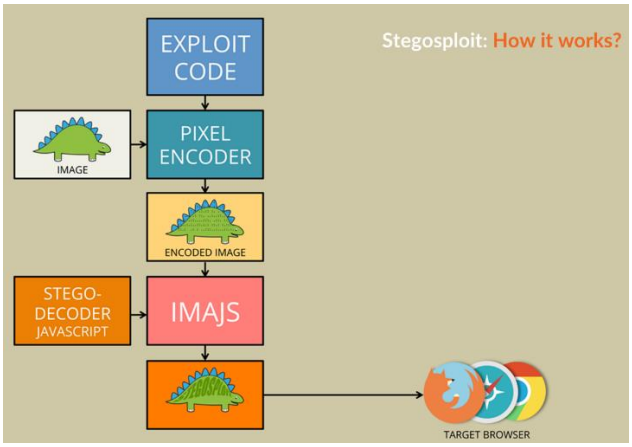

Figure 2: Process of making of polyglot image

The above figure shows using the stegosploit toolkit, how a polyglot image can be made. First, a browser exploit code and an image has been passed to a custom-made image/pixel encoder which encodes browser exploit code in an image steganographically and generates a new encoded image. The new image has been passed to the imajs library with an autorun stego decoder-script and it will generate the final – polyglot image, which will be used to attack by attackers and they will send/spread it using email or public image sharing websites to attack victim's browser.

In our work, we used stegosploit with default exploit code for CVE-2014-0282 to generate multiple images with exploits. We have not exploited any browser using the generated polyglots, as our intention is to detect the images. We then tried to detect them through a custom python script.

Some of the other steganography algorithms apart from LSB include outguess, LSB2, F5, DCT, and others, but as this tool supports only the LSB algorithm, the scope of the paper is limited to it only. This tool allows us to manipulate any bit layer other than LSB layer, but it is suggested that bit layers 0, 1 and 2 are most eligible bit layers for steganography, because of no or negligible visual aberration in image. Related work, Methodology, Results, and Conclusion are discussed next.

## II. RELATED WORK

As stegosploit is a new utility, very few researchers worked on stegosploit. The work of some of the researchers is cited in this section. Park, B., et. al. published their work on a possible method to protect the network against hidden exploits [13]. Jeyasekar, A. et. al. did the analysis of stegosploit images [14]. On another hand Dudheria, R. Discussed the use of stegosploit to attack smartphones via QR codes[15]. Harblson, C. discussed how stegosploit can be used in hacking with pictures[16].

 

However, the above authors worked on various aspects of stegosploit but none of them focused on the detection of stegosploit generated images. Not very closely related to stegosploit, but Pevný, T., et. al. proposed Malicons to detect payload in favicons[17]. Apart from stegosploit, many researchers worked on steganalysis techniques to detect steganographic content [18][19][20][21][22][23][24].

## III. METHODOLOGY

While working on stegosploit we tried to see if this type of images are detected by any anti-virus or not. So, we tested it in different anti-viruses including virustotal. The results were very shocking as none of the anti-viruses could detect it as a malicious file. The results of virustotal are shown in the following figure.

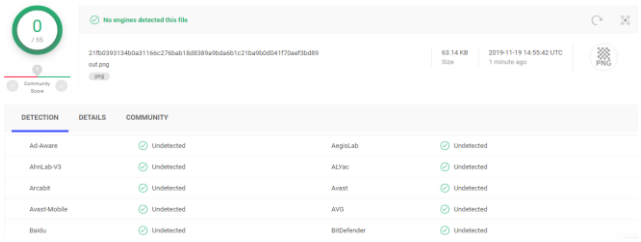


Figure 3: Results from Virustotal

The evasion was technically possible as the anti-viruses consider it as image and the malicious code is not directly visible. To overcome the issues, we decided to develop a python script to detect images having exploits and are generated by stegosploit. To achieve the results the environment was set for stegosploit as shown below:

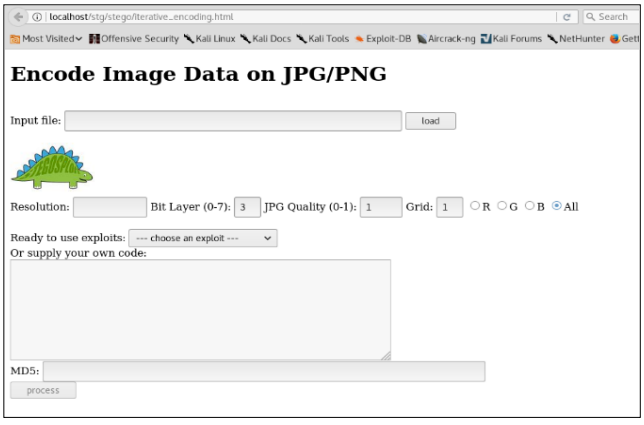


Figure 4: Stegosploit Setup

For samples, we decided to take JPG and PNG files as they are widely used and popular. The exploits were injected into them using stegosploit. As our goal was to detect the polyglots, we took 6 samples of PNG and 3 of JPG. Both the types of files were processed through stegosploit toolkit and resultant files were saved as shown in the following figure.

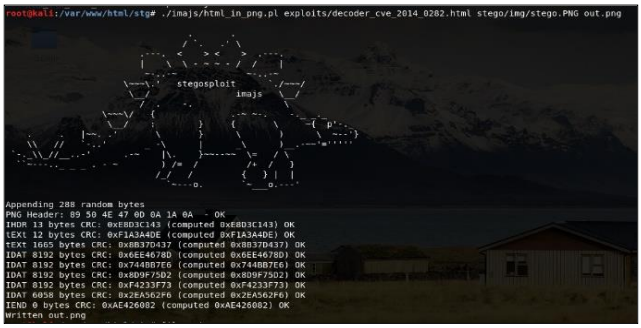


Figure 5: Exploits injection using stegosploit

To understand the signatures of detection, we analyzed benign and malignant samples in the hex editor and we could see the injected script.

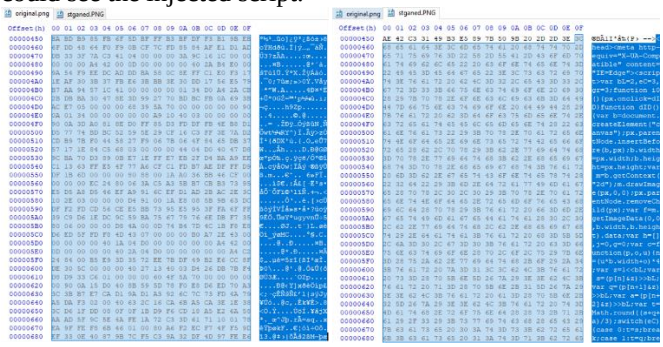

Figure 6: Injected script in polyglot image

The image on the left-hand side is the original image and the image on the right-hand side is the polyglot image. As it can be seen that the injected code is in the same offset range in both the cases. Further, there is a string in the polyglot images at a specific byte range. The python script was made to find the polyglot images from the computer based on the above observation.

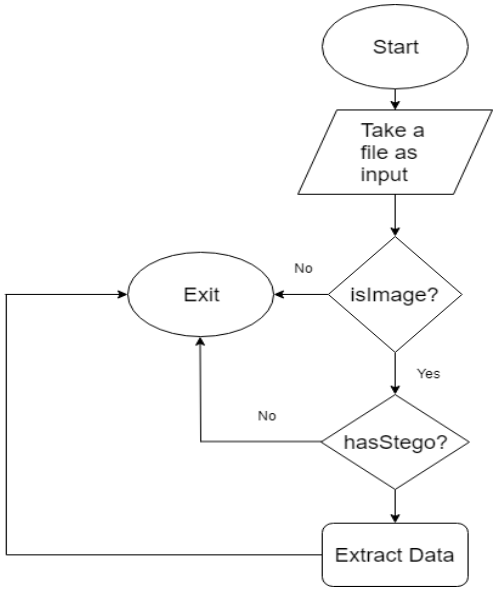


Figure 7: Flowchart of script

 

Work of the script is described below:

1. The script takes one file at a time and extracts a magic number to determine the file type. If the file is jpg/jpeg or png, the next step would be taken.
2. If the file is an image, the script will extract out some string/byte range, specific to file type and will compare it with signature.
3. If the signature is matched with extracted strings/byte range, it will be marked for the extraction of exploit code.
4. Then the script will search for the starting and ending address of exploit code and extract in the text file in a folder.

We mixed these files in different folders. For this, we have created a python script. The script has randomly created folders and subfolders of different random levels. Then, the script has moved all files and pasted it in previously created folders, randomly. Each folder is having a variety of other files including PDFs, Documents, other types of images, etc. Different types of files including infected files were kept for scanning. The following table lists a number of files in each type.

Table 1: Details of sample files.

| File Type | Number of Files | File Type | Number of Files |
|---|---|---|---|
| JPG | 16124 | GIF | 619 |
| PNG | 179465 | SQLITE | 91 |
| PDF | 102 | WAV | 17 |
| XLSX | 8 | DB | 1 |
| DOCX | 19 | DAT | 1 |
| PPT | 14 | BMP | 7 |
| TXT | 61 | SX | 1 |
| HTM | 7881 | PCAP | 1 |
| INI | 1 | EXE | 11 |
| XML | 1 | SH | 1 |
| ZIP | 205 | WINPE | 10 |
| JAR: 3 | | | |
| **TOTAL:** 204644 | | | |

It is important to note that these files include 3 JPG and 6 PNG files that contain exploit.

A python script was written to detect the stegosploit generated images. The script was executed against the parent folder having these files in subfolders at various levels. The script is designed to take a single path and check all the folders and files beneath it in a recursive manner.

The script is made very lightweight for faster detection. In the experiment, we considered two operating systems to test the performance of the script. To check the efficiency and performance of the script it was executed on Windows 10 and Ubuntu Subsystem Environment. The results and observations are discussed in the next section.

The machine was configured in such a way that only essential services and processes were allowed to run. The results may vary on different configurations. The following table contains system configuration information of the test machine.

Table 2: Testing system configuration

| | |
|---|---|
| Processor: | Intel(R) Core(TM) i5-2410M CPU @ 2.30GHz |
| RAM: | 12.0 GB DDR3 Dual Channel SODIMM 1600 MHz |
| HDD: | Seagate Barracuda 1.0 TB SATA3 6GBPS HDD |
| Available memory at the time of testing: | 9.9 GB |
| Windows | Windows 10 Pro x64 1809 (17763.316) |

## IV. RESULTS

The script works very efficiently and detects all the stegosploit generated images very quickly. The experiment was successful on both the operating systems as shown below:

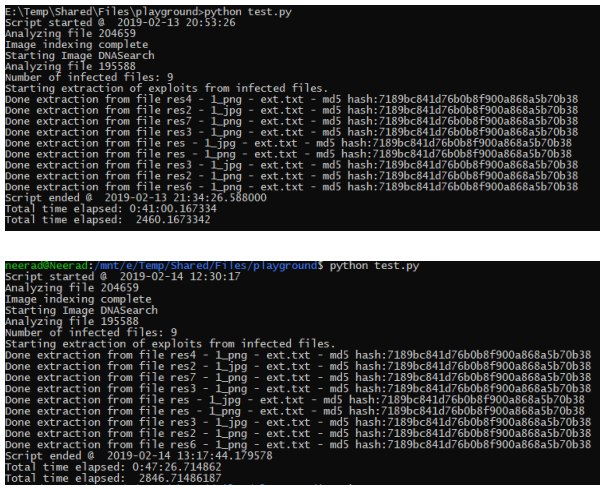


Figure 8 & 9: Successful detection on Windows and Linux Subsystem on Windows

 

To test the effectiveness and speed of python script it was executed four times on both the operating systems. The following table shows the time taken by the script to scan 204644 files for consecutive four executions.

Table 3: Time taken by the script

| Operating System | First Execution Time (s) | Second Execution Time (s) | Third Execution Time (s) | Fourth Execution Time (s) | Fifth Execution Time (s) |
|---|---|---|---|---|---|
| Windows 10 | 2460.7 | 339.43 | 407.23 | 338.43 | 316.18 |
| Ubuntu Subsystem | 2846.71 | 378.24 | 371.24 | 371.21 | 368.59 |

The above figures clearly indicate the speed by which the utility scans 204644 images for possible threats is highly impressive.

As the script is written in python it can run on any operating system. We calculated Karl Pearson's Coefficient to calculate the correlation of the speed when the script is executed on Windows and Ubuntu Subsystem as shown below:

Table 4: Statistics

| Execution | Windows 10 (X) | Ubuntu Subsystem (Y) | $A = X - \bar{X}$ ($\bar{X}$ =772.288) | $A^2$ | $B = Y - \bar{Y}$ ($\bar{Y}$ =868.198) | $B^2$ | AB |
|---|---|---|---|---|---|---|---|
| 1 | 2460.17 | 2846.71 | 1687.88 | 2848946 | 1979.51 | 3918460 | 3341179 |
| 2 | 339.43 | 378.24 | -432.86 | 187366 | -488.96 | 239082 | 211650 |
| 3 | 407.23 | 371.24 | -365.06 | 133267 | -495.96 | 245976 | 181054 |
| 4 | 338.43 | 371.21 | -433.86 | 188233 | -495.99 | 246006 | 215189 |
| 5 | 316.18 | 368.59 | -456.11 | 208035 | -498.61 | 248612 | 227420 |
| | $\Sigma X$ = 3861.44 | $\Sigma Y$ = 4335.99 | $\Sigma A$ = -0.51 | $\Sigma A^2$ = 3565846 | $\Sigma B$ = -0.01 | $\Sigma B^2$=4898136 | $\Sigma AB$=4176439 |

Karl Pearson's Coefficient $r = \frac{\sum AB}{\sqrt{\sum A^2 \sum B^2}}$

$$r = \frac{4176439}{4179234.436}$$

$$r = 0.99934$$

As the value of r tends to +1 it clearly indicates that the speed by which script detects stegosploit generated images on both Windows and Ubuntu Subsystem Environment is very close. This also confirms that the script can effectively detect stegosploit generated images on any of the machines having support for python.

## V. CONCLUSION

It is very crucial to detect images having exploit for preventing probable cybercrimes. The work done in this research will serve as the preventive step for image-based exploitations. The script generated during this research will be very helpful in detecting stegosploit generated images, as it is very fast and accurate. The script further is very useful as it detects the threat in almost no time on Windows and Ubuntu Subsystem Environment.

## REFERENCES


[1] Cox, I., Miller, M., Bloom, J., Fridrich, J., & Kalker, T. (2007). *Digital watermarking and steganography*. Morgan Kaufmann.

[2] Dumitrescu, D., Stan, I.-M., & Simion, E. (2017). Steganography Techniques.

[3] Cheddad, A., Condell, J., Curran, K., & Mc Kevitt, P. (2010). Digital image steganography: Survey and analysis of current methods. *Signal processing*, *90*(3), 727-752.

[4] Johnson, N. F., & Jajodia, S. (1998). Exploring steganography: Seeing the unseen. *Computer*, *31*(2).

[5] Wu, H. C., Wu, N. I., Tsai, C. S., & Hwang, M. S. (2005). Image steganographic scheme based on pixel-value differencing and LSB replacement methods. *IEE Proceedings-Vision, Image and Signal Processing*, *152*(5), 611-615.

[6] Ingemar, J. C., Miller, M. L., Jeffrey, A. B., Fridrich, J., & Kalker, T. (2008). *Digital Watermarking and Steganography*. *Digital Watermarking and Steganography*. Elsevier Inc.

[7] Yang, C.-N., Lin, C.-C., & Chang, C.-C. (2013). Steganography and watermarking. *Steganography and Watermarking*.

[8] Gupta, S., Goyal, A., & Bhushan, B. (2012). Information hiding using least significant bit steganography and cryptography. *International Journal of Modern Education and Computer Science*, *4*(6), 27.

[9] Song, S., Zhang, J., Liao, X., Du, J., & Wen, Q. (2011). A novel secure communication protocol combining steganography and cryptography. *Procedia Engineering*, *15*, 2767-2772.

[10] Abikoye, O. C., Adewole, K. S., & Oladipupo, A. J. (2012). Efficient data hiding system using cryptography and steganography.

[11] Shah S. (2015), Pastor Manul Laphroaig's, Export–Controlled, Church Newsletter

[12] Vaniea, K., & Rashidi, Y. (2016, May). Tales of software updates: The process of updating software. In *Proceedings of*


 

*the 2016 CHI Conference on Human Factors in Computing Systems* (pp. 3215-3226). ACM.

[13] Park, B., Kim, D., & Shin, D. (2015). A Study on a Method Protecting a Secure Network against a Hidden Malicious Code in the Image. *Indian Journal of Science and Technology*, *8*(26).

[14] Jeyasekar, A., Bisht, D., & Dua, A. (2016). Analysis of Exploit Delivery Technique using Steganography. *Indian Journal of Science and Technology*, *9*(39).

[15] Dudheria, R. Attacking Smartphones by Sharing Innocuous Images via QR Codes.

[16] Harblson, C. (2015). Hacking with pictures; new stegosploit tool hides malware inside internet images for instant drive-by pwning.

[17] Pevný, T., Kopp, M., Křoustek, J., & Ker, A. D. (2016). Malicons: Detecting Payload in Favicons. *Electronic Imaging*, *2016*(8), 1-9.

[18] Fridrich, J. (2006). Steganalysis. In *Multimedia Security Technologies for Digital Rights Management* (pp. 349–381). Elsevier Inc.

[19] Schaathun, H. G. (2012). Histogram Analysis. In *Machine Learning in Image Steganalysis* (p. 82230).

[20] Provos, N. H. G. K. (2003). Statistical Steganalysis. *ProQuest Information and Learning Company*, 78–80.

[21] Huang, F., Li, B., Shi, Y. Q., Huang, J., & Xuan, G. (2010). Image steganalysis. *Studies in Computational Intelligence*, *282*, 275–303.

[22] Al-Jarrah, M. M., Al-Taie, Z. H., & Abuarqoub, A. (2017). Steganalysis Using LSB-Focused Statistical Features. In *Proceedings of the International Conference on Future Networks and Distributed Systems - ICFNDS '17* (pp. 1–5). New York, New York, USA: ACM Press

[23] Harshal V. Patil1, B. H. Barhate2, "*A Review Paper on Data Hiding Techniques: Stegnography*", International Journal of Scientific Research in Computer Science and Engineering, Vol.06, Issue.01, pp.64-67, 2018

[24] Manisha Verma, Hardeep Singh Saini, "*Analysis of Various Techniques for Audio Steganography in Data Security*", International Journal of Scientific Research in Network Security and Communication, Vol.7, Issue.2, pp.1-5, 2019

## Authors Profile

Neerad Vaidya pursued Bachelor of Computer Applications from Krantigu Shyamji Krishna Verma Kachchh University, Gujarat, India. He is currently pursuing MSc. Digital Forensics and Information Security from Gujarat Forensic Sciences University, Gandhinagar, India. His research areas of interest are Cyber Security, Digital Forensics, Secure Source Code Reviewing and Vulnerability Assessment and Penetration Testing.

Parag H. Rughani completed his Ph. D. in computer science from Saurashtra University. He is currently working as an associate professor in Digital Forensics and Information Security at the Institute of Forensic Science, Gujarat Forensic Sciences University, Gandhinagar. He has 14 years of teaching experience and has published more than 15 research papers in reputed international journals. His areas of expertise include Digital Forensics, Memory Forensics, Malware Analysis, and IoT Security and Forensics.

 