Secure Gateway & Information Security Using Rubik's Cube Algorithm

A PROJECT REPORT

Submitted by

ANITEJ SRIVASTAVA, 19BCE0835
KULVIR SINGH, 19BCE2074

Course Code: CSE3502
Course Title: Information Security Management

Under the guidance of
Dr. Kakelli Anil Kumar
Associate Professor
SCOPE, VIT, Vellore.

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

April -2022

INDEX

1. INTRODUCTION

A directory traversal (or path traversal) attack exploits insufficient security validation or sanitization of user-supplied file names, such that characters representing "traverse to parent directory" are passed through to the operating system's file system API. An affected application can be exploited to gain unauthorized access to the file system. In this project we will be showing a web application with vulnerabilities and exploit them by performing a directory traversal attack. The project would clearly depict the detection of threats in the web application and would also provide a suitable solution to mend such measures using methods such as query sanitization etc. A secure website with the applied mitigation techniques would be made and the effectiveness of the solution would be shown by performing the attack on the secure website. The project will be supported by an information protection system which would incorporate an encryption algorithm called the Rubik's Cube Algorithm which provides file protection.

We will be exploiting the traversal vulnerability. The directory traversal and file traversal attacks will be dealt with through our project work. These attacks are majorly focused on web applications/sites and can allow the hacker to get access to secure files that the website creator wants to keep hidden from the public. A full demonstration of the attack will be shown with Kali Linux along with a process to mitigate it. We will incorporate the algorithms and knowledge gained via recent articles/research papers to formulate a novel and improved technique/algorithm to counter and exterminate these vulnerabilities. The most effective way to prevent file path traversal vulnerabilities is to avoid passing user-supplied input to the file system. The application would validate the user input before processing it. Ideally, the application would validate the canonical path of a file based on user input.

2. LITERATURE SURVEY

An analysis of some currently used directory traversal attack defenses and presents a new, stack-based algorithm to help prevent these attacks by safely canonicalizing user-supplied path strings [1]. The goal of this algorithm is to be small, easy to test, cross-platform compatible, and above all, intuitive. The algorithm used is small, cost effective, intuitive, easy to test and above all, cross-platform. There is proof of correctness and verification strategies using symbolic execution for the algorithm. The algorithm could prove to be simple & ineffective for more complex attacks with more powerful computers.

In the permutation process, the Rubik's cube method and bit-level encryption principle are combined to realize the image scrambling operation in three-dimensional space[2]. Through the contraction mapping principle, the ascending and reducing dimension operation can be implemented effectively. In the diffusion process, they design an improved two-dimensional diffusion structure, which can spread slight change in plain image to the whole cipher image. Experimental results and simulation analysis illustrate the security and the validity of the proposed scheme. The 3-D bit level encryption proves to be costly and has high time complexity.

A policy enforcement and web attack detection framework for HTTP protocol [3]. The proposed framework can monitor and analyze HTTP traffic to detect injection, misconfiguration and directory traversal attacks. Moreover, this framework can be used to enforce web application access policies involving content type, URL and device level access. The paper effectively monitors and analyses HTTP traffic to detect injection, misconfiguration and directory traversal attacks and exposes vulnerabilities effectively. One disadvantage is that this paper is not tested for very high traffic & could prove to be ineffective against severe DDoS attacks.

Classical and modern types of SQLIA and display different existing technique and tools which are used to detect or prevent these attacks[4]. Applications that do not properly validate the user's input make them vulnerable against SQL injection. This paper talks about SQL Injection Attacks & their occurrence when an attacker is able to insert a series of malicious SQL statements into a ―query‖ through manipulating user input data for execution by the back-end database. This paper gives us an idea of the existing methodologies of

SQLInjection detection & prevention in a descriptive manner & also about their advantages and shortcomings. A useful table is also given to summarize the entire survey. This paper does not present a new or improved methodology but only summarizes the existing ones. Some of the surveys may not act in preferred manner in certain situations.

Research has confirmed different penetration tests using a private networks, devices, and virtualized systems and tools[5]. They have used tools within the Kali Linux suite. The attacks performed included: smartphone penetration testing, hacking phones Bluetooth, traffic sniffing, hacking WPA Protected Wifi, Man-in-the-Middle attack, spying (accessing a PC microphone), hacking phones Bluetooth, and hacking remote PC via IP and open ports using advanced port scanner. The results are then summarized and discussed. This paper details extensive penetrations testing using relevant methodologies & Kali Linux which is also used by attackers. This paper covers important aspects of smartphone & PC attacks & also exposes serious vulnerabilities in those systems. The paper does not discuss any relevant countermeasure or mitigation techniques for the vulnerabilities identified by the penetration tests.

The idea developed around the use of Rubik's cube principle within image cryptosystems is that the processing presents the implementation of a communication system between multiple mobile devices with imaging sensors, that encrypt the images using Rubik cube encryption algorithm, and a server[6]. The performance and the suitability of the algorithm for mobile devices is investigated. There is an in depth explanation about the Rubik’s cube algorithm and its process of implementation. Decryption module was not a primary focus for the paper. A web app implementation is not been stated

The focus of directory traversal vulnerability on VMware [7]. The principle, triggering conditions and the exploit process of this vulnerability is discussed. Furthermore, the experimental environment is established to demonstrate the attack method. Experimental configurations and results discussion are given in detail. Finally, generalized recommendations are stated that can be applied to achieve secure virtualized implementations. The paper gives a major insight to directory traversal attacks and the exploitation technique. This also gives us a better understanding of the tree structure of a directory which will help us in sanitizing. the query for mitigation. The paper does not discuss any specific countermeasure for a real environment. Moreover it has been performed on a virtual machine hence we cannot apply prevention methods given in a web app.

The analysis of web server log files, which includes normal and malicious users' access patterns with their relevant links [8]. This uses a web server log file dataset for the detection of web application attacks. This system intends to analyze normal and attack behaviors from web server log and then classify attack types which are included in the dataset. In this system, three types of attacks are detected namely, SQL injection, XSS and directory traversal attacks. Attack analysis stage is done by request length module and regular expressions for various attack patterns. The attack detection is analyzed at large and a proper understanding of the attacks made on a web system is given. There is an effective use of regular expression at identifying attacks which will help us in our project. This paper does not give an insight to prevention of these attacks in a real time situation.

Analysis of the security threats specifically evolving in University's network, and with consideration of these issues, proposed an information security framework for University network environment [9]. The proposed framework reduces the risk of security breach by supporting three phase activities; the first phase assesses the threats and vulnerabilities in order to identify the weak point in educational environment, the second phase focuses on the highest risk and create actionable remediation plan, the third phase of risk assessment model recognizes the vulnerability management compliance requirement in order to improve University's security position. This paper deals with the attack and mitigation on a web server. It has staged the entire process in 3 phases. It identifies the weak point in the server and rectifies it. The paper only talks about a specific university web network where the vulnerability and attack management are not fully tested and is not fail safe.

Vulnerability Assessment and Penetration Testing (VAPT) techniques help them to go looking out security loopholes[10]. VAPT helps organizations to determine whether their security arrangements are working properly. This paper aims to elucidate overview and various techniques used in vulnerability assessment and penetration testing (VAPT). Also focuses on implementing cyber security awareness and its importance at various levels of an organization for adoption of required up to date security measures by the organization to stay protected from various cyber-attacks. This paper gives an in-depth knowledge about the various attacks that can be made and the loopholes that can be exploited by hackers. Prevention of attacks was also implemented successfully. This paper has covered all aspects with attack and its prevention; however it is not failed proof and fail safe. This can be stated as a drawback of the following research.

The vulnerability detection module with three functions is assembled into a tool, and a composite detection tool with multiple functions is proposed to cope with some frequent vulnerabilities[11]. The proposed tool includes a sum of three types of vulnerability detection, including cross-site scripting attacks, SQL injection, and directory traversal. The advantages obtained within the work are the cost of manual testing is eliminated; second, the work efficiency is greatly improved; and third, the network is safely operated in the first time.

Introducing the participants to common web application vulnerabilities employing a novel lightweight application case study that demonstrates software security weaknesses, has been proposed, in a very practical manner[12]. Within the proposed work, the authors facilitate discussion about ways to show students security weaknesses, a way to mitigate software vulnerabilities through secure software design and coding practices, and share ideas on ways to improve the case study application.

Some exploration and detailing of SQL injection, operating system command injection, path traversal, and cross-site scripting vulnerabilities through dynamic and static approaches has been done[13]. The authors also examine various security measures in web applications and selected five tools based on their features for scanning PHP, and JAVA code focuses on SQL injection, cross-site scripting, Path Traversal, operating system command. Moreover, this research discusses the approach of a cyber-security tester or a security developer sorting out vulnerabilities through dynamic and static approaches using manual and automatic web vulnerability scanners.

A systematic approach to style and therefore the development of an IoT testbed to come up with an attack dataset, namely the AIoT-Sol Dataset is done[14]. The proposed dataset contains the benign traffic as well as the often-overlooked attack techniques within the existing IoT datasets. It encompasses 17 attack types from several categories: network attacks, web attacks, and web IoT message protocol attacks. The attacks were selected by referencing the Open Web Application Security Project (OWASP) IoT Top Ten.

A framework for extracting attack narratives from traffic datasets has been proposed[15]. Within the framework, the authors propose the re-examination of packet grepping for attack signatures in network traffic as a viable, fast, and effective means to extract attack narratives from large amounts of network traffic. By combining attack signature packet grepping with Mandiant's Attack Lifecycle Model, there has been a rise in the effectiveness of packet grepping and a technique has been created that's simple and powerful for constructing attack narratives.

3. OVERVIEW OF THE WORK

3.1 Problem Description

Based on the literature survey, we gathered from paper [1] about a stack-based algorithm to help prevent directory traversal attacks by safely canonicalizing user-supplied path strings. The algorithm is small, easy to test, cross-platform compatible, and above all, intuitive. From paper [7] we learnt about an existing system where principle, triggering conditions and the exploit process of a VMWare vulnerability is demonstrated. Along with that generalized recommendations are stated that can be applied to achieve secure virtualized implementations.

From paper [10], we learnt about an existing system in place for vulnerability assessment and penetration testing system and from paper [8], an existing detection system of web app attacks was learnt about. This system uses a web server log file dataset for the detection of web application attacks. This system analyzes normal and attack behaviors from web server log and then classifies attack types which are included in the dataset. The attack types include SQL injection, directory traversal etc.

For the Rubik's Cube Algorithm, we learnt from paper [2], that the Rubik's cube method and bit-level encryption principle are combined to realize the image scrambling operation in three-dimensional space using the permutation process. From paper [6], we were presented with the implementation of a communication system between multiple mobile devices with imaging sensors, that encrypt the images using Rubik cube encryption algorithm, and a server.

In paper [1], the algorithm could prove to be simple & ineffective for more complex attacks with more powerful computers whereas in paper [2], the 3-D bit level encryption proves to be costly and has high time complexity. For paper [3], the technique is not tested for very high traffic & could prove to be ineffective against severe DDoS attacks. In paper [6], implementing Rubik's cube encryption, the decryption module was not a primary focus and a web app implementation has not been stated to test it out.

In paper [7], where directory traversal analysis and vulnerability exploitation has been done for VMWare, there has been no discussion of any specific countermeasure for a real environment.

Moreover it has been performed on a virtual machine hence we cannot apply prevention methods given in a web app.

Paper [10] has covered most aspects with attack and its prevention, however it is not fail proof and fail safe. This can be stated as a drawback of the following research.

3.2 Working Model

We have used the PHP server in XAMPP environment to exploit all possible directory traversal vulnerabilities that could arise and analysed the consequences of those vulnerabilities. For each of the vulnerabilities assessed, we demonstrated a security measure for mitigation that is fast and effective.

The final working model requires the Kali Linux operating system for vulnerability analysis and for directory traversal attacks on those vulnerabilities. On the host machine a vulnerable PHP website has been developed with various directories and a dedicated interface. Once an attacker brute forces paths to common directories in the URL, some confidential directories will be accessible, like psswd, issue & Windows boot files.

We launch the terminal in Kali Linux on a VM and use the tool DotDotpwn to perform directory traversal on the vulnerable website in the host machine. It displays all paths that are vulnerable in the website in real time. A penetration tester can also click on the path link provided by dotdotpwn to access the vulnerable directories and files. A white hat hacker can specify the depth of traversal i.e. depth of the directory tree that may be traversed, to look for possible vulnerabilities. There are many configuration options available for performing directory traversal with dotdotpwn.

In the mitigation part of the project, we have employed various ways to mitigate the directory traversal vulnerability in the website developed as well as make the files more secure.

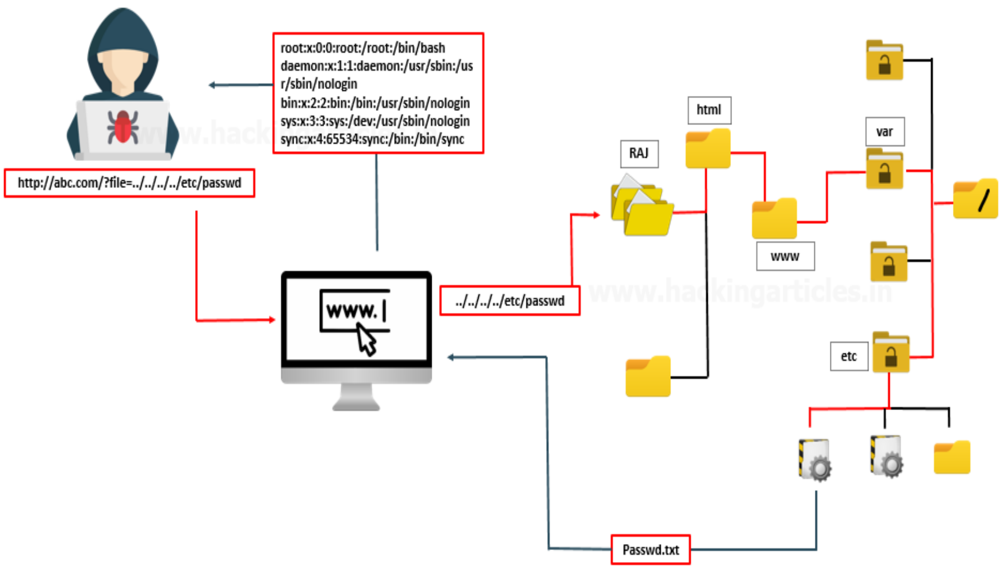
Firstly, we have employed an algorithm that creates a blank index.php file in all the directories present in the website. When an attacker brute forces the URL to look for common directory paths, the website displays a blank page even if the path exists and hence the files are secured. The same happens when a path is accessed from dotdotpwn.

Secondly we demonstrate a mitigation technique where, inside the working directory of the website, we create a file named htaccess and write “Options -Indexes” inside it. This causes the dotdotpwn to show all directories as 403 which were earlier shown as vulnerable. When we give the path to the directory in the website url, it displays “403: Forbidden”.

Lastly and algorithm has been employed to make all the image files more secure through encryption by shuffling the pixels of the image and decrypting when needed.

3.3 Design Description

The design consists of directory traversal vulnerability along with its analysis & exploitation. It consists of the use of Kali Linux, its tools like dotdotpwn. Php's inherent vulnerabilities as well as Rubik's cube algorithm is a major part of the design as it encrypts and decrypts images to make the website even more secure.



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/us
r/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Fig 1: Illustration of a directory traversal attack

3.3.1 Directory traversal vulnerability

A directory traversal vulnerability is that the results of insufficient filtering/validation of browser input from users. Directory traversal vulnerabilities may be located in web server software/files or in application code that's executed on the server. Directory traversal could be a style of HTTP exploit that's utilized by attackers to realize unauthorized access to restricted directories and files. Directory traversal, also referred to as path traversal, ranks #13 on the CWE/SANS Top 25 Most Dangerous Software Errors. Directory traversal attacks use web server software to take advantage of inadequate security mechanisms and access directories and files stored outside of the online root folder. An attacker that exploits a directory traversal vulnerability is capable of compromising the complete web server.

3.3.2 PHP vulnerability

An authenticated or unauthenticated user can request and examine or execute files that they must not be able to access. Such files usually reside outside of the root directory of an internet application or outside of a directory to which the user is restricted (for example, /var/www). In many cases, an attacker may read any file accessible for the user that's running the online server (usually www-data). If the server has badly configured file permissions, this attack may be escalated further.

3.3.3 Dotdotpwn:

DotDotPwn is an intelligent fuzzing tool that permits an attacker to identify potential vulnerabilities that will be associated with traverse directory within a given service. The tool is effective and may help discover flaws in web server protocols like TFTP, HTTP, and FTP. The tool can particularly be useful when handling penetration tests on applications that are web-based.

DotDotPwn may be a a part of the Kali Linux package, therefore it comes pre-installed. This tool is written using Perl.

DotDotPwn are often accustomed spot errors that will occurred as a results of improper data validation, parameters that are inaccurate or data that's erroneous. Such style of data can help the attacker to grasp which sort of attack payload to deliver to the victim. As a results of the tool's many use cases, it can provide several attack vectors which may be exploited during an attack.

3.3.4 Image encryption & decryption

This design component adds an extra layer of security. Even if an attacker gains access to certain files & directories, the images will still be secured through encryption and can only be decrypted by a combination of keys that only authorised people have. Rubik's cube algorithm has been used for encrypting and decrypting the image.

4. IMPLEMENTATION

4.1 Description of Modules/Programs

Dotdotpwn

This directory traversal attack tool will be demonstrated using a kali linux terminal. The tool is used to perform various fuzzer attacks on a malicious website and the vulnerabilities are noted. The tool has various modes which will be used to identify various types of traversal vulnerabilities on the custom built malicious website. The same tool and types of attacks will be conducted on a secure website. The secure website has the necessary safety measures which can evade the traversal attacks. A mitigation code is also written in php which can be used as bottleneck control. When the secure website fails, this code will generate index files in each folder of the project which prevents the attacker to view the root or files of the website or system.

Key Generation

The image to be encrypted is taken as input from the user. This image is then processed as pixels. These pixels are stored into a multi-dimensional array. The number of rows of this array is stored into a variable say m and the number of columns is stored into another variable n. A variable alpha is set to 8 and then two vectors Kr and Kc are declared to represent the keys of rows and columns respectively. Kr vector is filled with m random integers ranging from 0 to ((2^8)-1) and Kc is filled with n random integers ranging from 0 to ((2^8)-1) using the randint(a,b) function in python.

Image Processing

The user is required to store their image to be encrypted in the project folder. Using Image class from PIL library in python, we extract and process the image stored by the user. Image.open() function is used to fetch the input image. It is then converted to pixels by using the image.load() function. Red, blue and green matrices are then initialized to get the respective values from the pixel matrix by looping through the pixel matrix. After encryption the altered red, blue and green matrices are compiled and set into the pixel matrix. The pixel matrix is then passed to the image.save() function to convert it back to image format. File Writing Using open() function from python’s default library we create a keys.txt file. After key generation we copy the data stored in vector Kr and Kc into the file by using the write() function.

Up Circular Shift

This is a user defined function that performs up circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically upwards in the column.

Down Circular Shift

This is a user defined function that performs down circular shift on an array that is send through the function. It uses the numpy library and numpy.roll() function of python. It shifts the data at a particular index of the array to a new position according to number of shifts required to be performed. The number of shifts is also taken in as a parameter. It shifts the data vertically downwards in the column.

Bit Rotation

This function is used to reverse the bits that is entered as parameter to this function. It uses basic string reversal technique which is starting from the end of the string and reaching to first bit there by getting the bits of the string in reverse order.

Encryption and Decryption module simply follows the Rubik's Cube Algorithm which we have implemented for red, blue and green values.

4.2 Source Code

Encrypt.py

```
import os
from PIL import Image
from random import randint
import numpy
import sys
def upshift(a,index,n):
 col = []
 for j in range(len(a)):
  col.append(a[j][index])
 shiftCol = numpy.roll(col,-n)
 for i in range(len(a)):
  for j in range(len(a[0])):
   if(j==index):
```

```python
    a[i][j] = shiftCol[i]
def downshift(a,index,n):
 col = []
 for j in range(len(a)):
  col.append(a[j][index])
 shiftCol = numpy.roll(col,n)
 for i in range(len(a)):
  for j in range(len(a[0])):
   if(j==index):
    a[i][j] = shiftCol[i]
def rotate180(n):
 bits = "{0:b}".format(n)
 return int(bits[::-1], 2)

#loading image from a directory
im =
Image.open(os.path.join('C:\\Users\\kulvir\\Desktop\\RubikAlg\\',
'image.PNG'))

pix = im.load()#converting image to pixels as python object


#Obtaining the RGB matrices
r = []
g = []
b = []
for i in range(im.size[0]):
 r.append([])
 g.append([])
 b.append([])
 for j in range(im.size[1]):
  rgbPerPixel = pix[i,j]
  r[i].append(rgbPerPixel[0])
  g[i].append(rgbPerPixel[1])
  b[i].append(rgbPerPixel[2])

# M x N image matrix
m = im.size[0] #rows
n = im.size[1] #columns


# Vectors Kr and Kc
alpha = 8
```

```python
Kr = [randint(0,pow(2,alpha)-1) for i in range(m)]
Kc = [randint(0,pow(2,alpha)-1) for i in range(n)]

#maximum number of iterations
ITER_MAX = 1

print('Vector Kr : ', Kr)
print('Vector Kc : ', Kc)

#key for encryption written into the file keys.txt
f = open('keys.txt','w+')
f.write('Vector Kr :\n')
for a in Kr:
 f.write(str(a) + '\n')
f.write('Vector Kc :\n')
for a in Kc:
 f.write(str(a) + '\n')
f.write('ITER_MAX :\n')
f.write(str(ITER_MAX) + '\n')

for iterations in range(ITER_MAX):
 # For each row
 for i in range(m):
  rTotalSum = sum(r[i]) #sum of each array present in r[][]
  gTotalSum = sum(g[i])
  bTotalSum = sum(b[i])
  #modulo of sum of each r,g,b
  rModulus = rTotalSum % 2
  gModulus = gTotalSum % 2
  bModulus = bTotalSum % 2

  if(rModulus==0):
   #right circular shift according to Kr
   r[i] = numpy.roll(r[i],Kr[i])
  else:
   #left circular shit according to Kr
   r[i] = numpy.roll(r[i],-Kr[i])
  if(gModulus==0):
   g[i] = numpy.roll(g[i],Kr[i])
  else:
   g[i] = numpy.roll(g[i],-Kr[i])
  if(bModulus==0):
```

```
  b[i] = numpy.roll(b[i],Kr[i])
 else:
  b[i] = numpy.roll(b[i],-Kr[i])
# For each column
for i in range(n):
 rTotalSum = 0
 gTotalSum = 0
 bTotalSum = 0
 for j in range(m):
  rTotalSum += r[j][i]
  gTotalSum += g[j][i]
  bTotalSum += b[j][i]
 rModulus = rTotalSum % 2
 gModulus = gTotalSum % 2
 bModulus = bTotalSum % 2
if (rModulus == 0):
  upshift(r, i, Kc[i])
 else:
  downshift(r, i, Kc[i])
 if (gModulus == 0):
  upshift(g, i, Kc[i])
 else:
  downshift(g, i, Kc[i])
 if (bModulus == 0):
  upshift(b, i, Kc[i])
 else:
  downshift(b, i, Kc[i])

# For each row
for i in range(m):
 for j in range(n):
  if (i % 2 == 1):
   r[i][j] = r[i][j] ^ Kc[j]
   g[i] [j] = g[i][j] ^ Kc[j]
   b[i][j] = b[i][j] ^ Kc[j]
  else:
   r[i][j] = r[i][j] ^ rotate180(Kc[j])
   g[i][j] = g[i][j] ^ rotate180(Kc[j])
   b[i][j] = b[i][j] ^ rotate180(Kc[j])

# For each column
for j in range(n):
  for i in range(m):
   if (j % 2 == 0):
    r[i][j] = r[i][j] ^ Kr[i]
    g[i][j] = g[i][j] ^ Kr[i]
    b[i][j] = b[i][j] ^ Kr[i]
   else:
    r[i][j] = r[i][j] ^ rotate180(Kr[i])
    g[i][j] = g[i][j] ^ rotate180(Kr[i])
    b[i][j] = b[i][j] ^ rotate180(Kr[i])


for i in range(m):
 for j in range(n):
  pix[i,j] = (r[i][j],g[i][j],b[i][j])
im.save('C:\\Users\\kulvir\\Desktop\\RubikAlg\\encrypted.PNG')
print("Success")
```

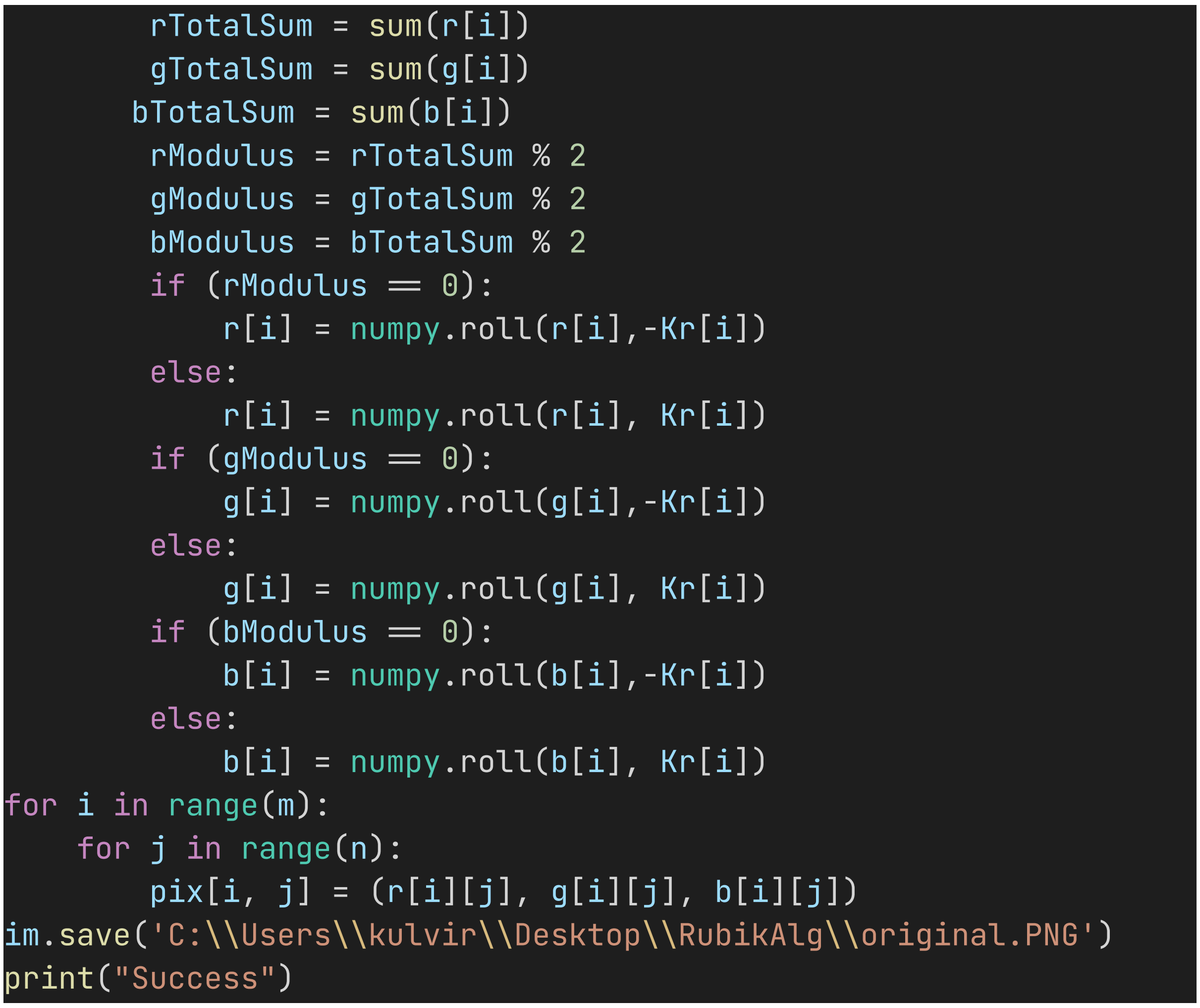
Decrypt.py

```
import os
from PIL import Image
import numpy
import sys
#to load encrypted image
im =
Image.open(os.path.join('C:\\Users\\kulvir\\Desktop\\RubikAlg\\',
'encrypted.PNG'))
pix = im.load() #pixels loaded to pix array

def upshift(a,index,n):
 col = []
 for j in range(len(a)):
    col.append(a[j][index])
 shiftCol = numpy.roll(col,-n)
 for i in range(len(a)):
    for j in range(len(a[0])):
        if(j==index):
            a[i][j] = shiftCol[i]
def downshift(a,index,n):
    col = []
    for j in range(len(a)):
        col.append(a[j][index])
    shiftCol = numpy.roll(col, n)
```

```python
    for i in range(len(a)):
        for j in range(len(a[0])):
            if (j == index):
                a[i][j] = shiftCol[i]
def rotate180(n):
    bits = "{0:b}".format(n)
    return int(bits[::-1], 2)
#Obtaining the RGB matrices
r = []
g = []
b = []
#obtainge rgb values from pix
for i in range(im.size[0]):
    r.append([])
    g.append([])
    b.append([])
    for j in range(im.size[1]):
        rgbPerPixel = pix[i, j]
        r[i].append(rgbPerPixel[0])
        g[i].append(rgbPerPixel[1])
        b[i].append(rgbPerPixel[2])
m = im.size[0]
n = im.size[1]
#key input from user
Kr = []
Kc = []
print('Enter value of Kr')
for i in range(m):
    Kr.append(int(input()))
print('Enter value of Kc')
for i in range(n):
    Kc.append(int(input()))
print('Enter value of ITER_MAX')
ITER_MAX = int(input())

for iterations in range(ITER_MAX):
    # For each column
    for j in range(n):
        for i in range(m):
            if (j % 2 == 0):
                r[i][j] = r[i][j] ^ Kr[i]
                g[i][j] = g[i][j] ^ Kr[i]
                b[i][j] = b[i][j] ^ Kr[i]
            else:
                r[i][j] = r[i][j] ^ rotate180(Kr[i])
                g[i][j] = g[i][j] ^ rotate180(Kr[i])
                b[i][j] = b[i][j] ^ rotate180(Kr[i])
   # For each row
    for i in range(m):
        for j in range(n):
            if (i % 2 == 1):
                r[i][j] = r[i][j] ^ Kc[j]
                g[i][j] = g[i][j] ^ Kc[j]
                b[i][j] = b[i][j] ^ Kc[j]
            else:
                r[i][j] = r[i][j] ^ rotate180(Kc[j])
                g[i][j] = g[i][j] ^ rotate180(Kc[j])
                b[i][j] = b[i][j] ^ rotate180(Kc[j])
    # For each column
    for i in range(n):
        rTotalSum = 0
        gTotalSum = 0
        bTotalSum = 0
        for j in range(m):
            rTotalSum += r[j][i]
            gTotalSum += g[j][i]
            bTotalSum += b[j][i]
        rModulus = rTotalSum % 2
        gModulus = gTotalSum % 2
        bModulus = bTotalSum % 2
        if (rModulus == 0):
            downshift(r, i, Kc[i])
        else:
            upshift(r, i, Kc[i])
        if (gModulus == 0):
            downshift(g, i, Kc[i])
        else:
            upshift(g, i, Kc[i])
        if (bModulus == 0):
            downshift(b, i, Kc[i])
        else:
            upshift(b, i, Kc[i])
    # For each row
    for i in range(m):
```



```
        rTotalSum = sum(r[i])
        gTotalSum = sum(g[i])
       bTotalSum = sum(b[i])
        rModulus = rTotalSum % 2
        gModulus = gTotalSum % 2
        bModulus = bTotalSum % 2
        if (rModulus == 0):
            r[i] = numpy.roll(r[i],-Kr[i])
        else:
            r[i] = numpy.roll(r[i], Kr[i])
        if (gModulus == 0):
            g[i] = numpy.roll(g[i],-Kr[i])
        else:
            g[i] = numpy.roll(g[i], Kr[i])
        if (bModulus == 0):
            b[i] = numpy.roll(b[i],-Kr[i])
        else:
            b[i] = numpy.roll(b[i], Kr[i])
for i in range(m):
    for j in range(n):
        pix[i, j] = (r[i][j], g[i][j], b[i][j])
im.save('C:\\Users\\kulvir\\Desktop\\RubikAlg\\original.PNG')
print("Success")
```

4.3 Test cases

The Rubik's Cube Algorithm based encryption is tested across multiple images of various formats.

The dotdotpwn tool is tested on the custom built websites which are vulnerable as well as the secure website to showcase successful mitigation.

4.4 Execution

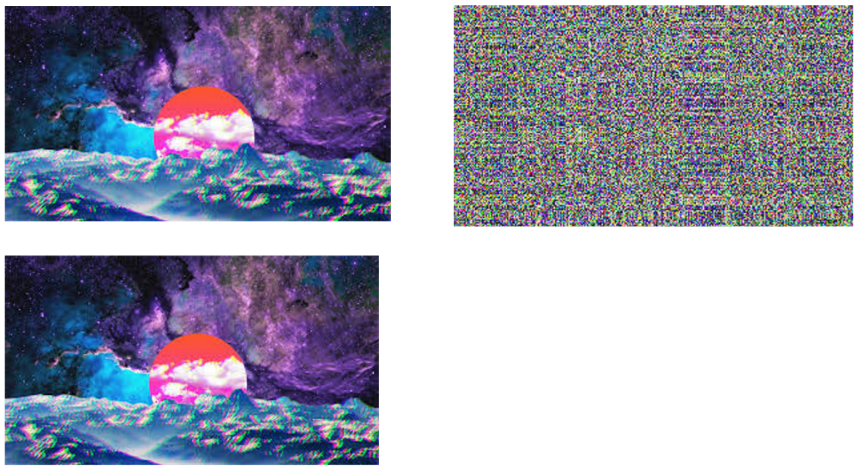
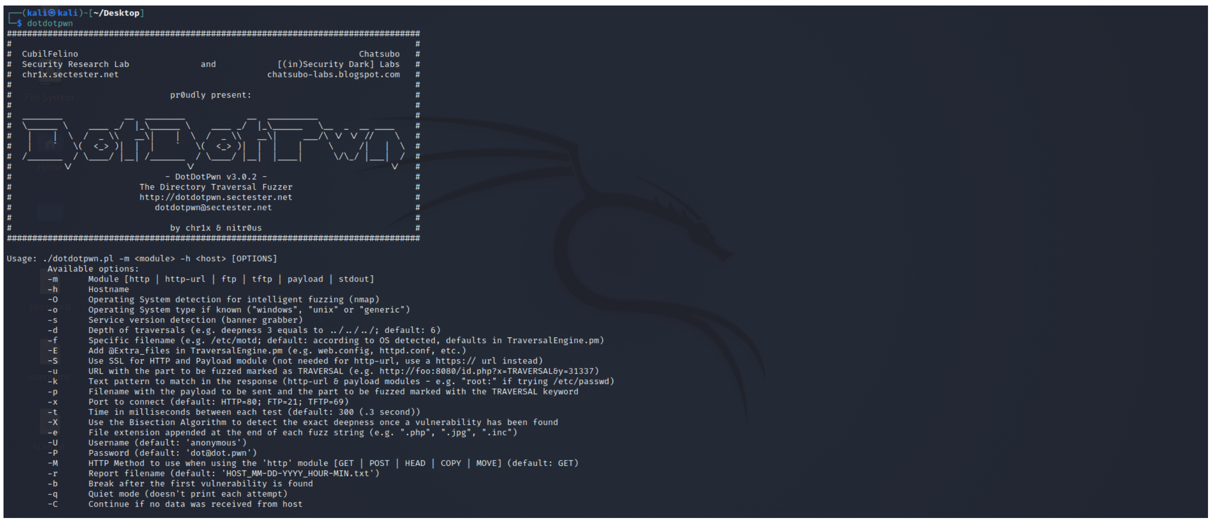


Fig 2: Encryption & Decryption of Image

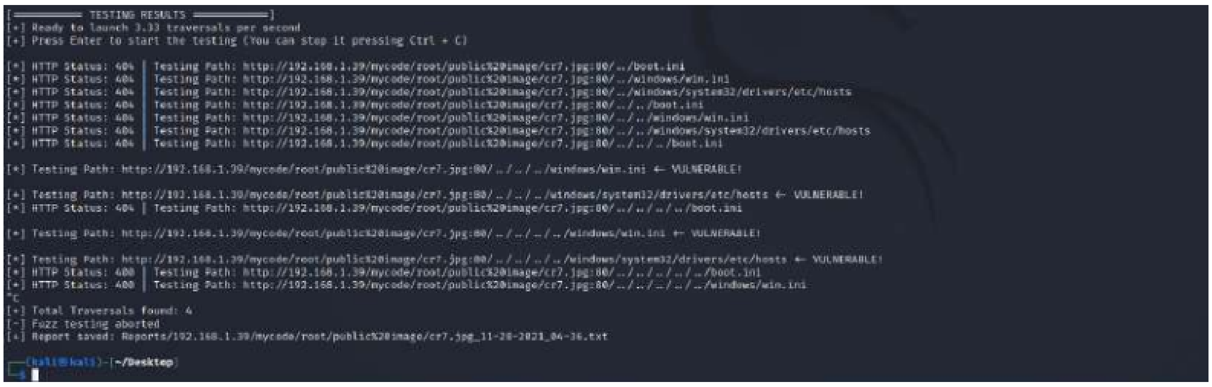
The original image is first passed through the encryption module. The keys are generated for decryption and the original image is encrypted as seen in the second stage of the processing. Now the encrypted image is decrypted using the keys and the original image is retrieved.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ dotdotpwn
#################################################################################
#                                                                               #
#  CubilFelino                                                   Chatsubo       #
#  Security Research Lab              and             [(in)Security Dark] Labs  #
#  chr1x.sectester.net                             chatsubo-labs.blogspot.com   #
#                                                                               #
#                               pr0udly present:                                #
#                                                                               #
#  ________            __  ________            __  __________                   #
#  \______ \   ____  _/  |_\______ \   ____  _/  |_\______   \__  _  __ ____    #
#   |    |  \ /  _ \ \   __\|    |  \ /  _ \ \   __\|     ___/\ \/ \/ //    \   #
#   |    `   (  <_> ) |  |  |    `   (  <_> ) |  |  |    |     \     /|   |  \  #
#  /_______  /\____/  |__| /_______  /\____/  |__|  |____|      \/\_/ |___|  /  #
#          \/                      \/                                      \/   #
#                              - DotDotPwn v3.0.2 -                             #
#                         The Directory Traversal Fuzzer                        #
#                         http://dotdotpwn.sectester.net                        #
#                            dotdotpwn@sectester.net                            #
#                                                                               #
#                               by chr1x & nitr0us                              #
#################################################################################

Usage: ./dotdotpwn.pl -m <module> -h <host> [OPTIONS]
        Available options:
        -m      Module [http | http-url | ftp | tftp | payload | stdout]
        -h      Hostname
        -O      Operating System detection for intelligent fuzzing (nmap)
        -o      Operating System type if known ("windows", "unix" or "generic")
        -s      Service version detection (banner grabber)
        -d      Depth of traversals (e.g. deepness 3 equals to ../../../; default: 6)
        -f      Specific filename (e.g. /etc/motd; default: according to OS detected, defaults in TraversalEngine.pm)
        -E      Add @Extra_files in TraversalEngine.pm (e.g. web.config, httpd.conf, etc.)
        -S      Use SSL for HTTP and Payload module (not needed for http-url, use a https:// url instead)
        -u      URL with the part to be fuzzed marked as TRAVERSAL (e.g. http://foo:8080/id.php?x=TRAVERSAL&y=31337)
        -k      Text pattern to match in the response (http-url & payload modules - e.g. "root:" if trying /etc/passwd)
        -p      Filename with the payload to be sent and the part to be fuzzed marked with the TRAVERSAL keyword
        -x      Port to connect (default: HTTP=80; FTP=21; TFTP=69)
        -t      Time in milliseconds between each test (default: 300 (.3 second))
        -X      Use the Bisection Algorithm to detect the exact deepness once a vulnerability has been found
        -e      File extension appended at the end of each fuzz string (e.g. ".php", ".jpg", ".inc")
        -U      Username (default: 'anonymous')
        -P      Password (default: 'dot@dot.pwn')
        -M      HTTP Method to use when using the 'http' module [GET | POST | HEAD | COPY | MOVE] (default: GET)
        -r      Report filename (default: 'HOST_MM-DD-YYYY_HOUR-MIN.txt')
        -b      Break after the first vulnerability is found
        -q      Quiet mode (doesn't print each attempt)
        -C      Continue if no data was received from host
```

Dotdotpwn tool kit to perform various attacks on the website



```
[═══════════ TESTING RESULTS ═══════════]
[+] Ready to launch 3.33 traversals per second
[+] Press Enter to start the testing (You can stop it pressing Ctrl + C)

[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../boot.ini
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../windows/win.ini
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../windows/system32/drivers/etc/hosts
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../boot.ini
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../windows/win.ini
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../windows/system32/drivers/etc/hosts
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../boot.ini

[*] Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../windows/win.ini <- VULNERABLE!

[*] Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../windows/system32/drivers/etc/hosts <- VULNERABLE!
[*] HTTP Status: 404 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../../boot.ini

[*] Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../../windows/win.ini <- VULNERABLE!

[*] Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../../windows/system32/drivers/etc/hosts <- VULNERABLE!
[*] HTTP Status: 400 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../../../boot.ini
[*] HTTP Status: 400 | Testing Path: http://192.168.1.39/mycode/root/public%20image/cr7.jpg:80/../../../../../windows/win.ini
^C
[+] Total Traversals found: 4
[-] Fuzz testing aborted
[+] Report saved: Reports/192.168.1.39/mycode/root/public%20image/cr7.jpg_11-28-2021_04-36.txt

┌──(kali㉿kali)-[~/Desktop]
└─$ 
```

The vulnerability found in the testing results can be clearly seen.

4.5 Results

Attack Prefix	Attack Type	Website	Vulnerability Found
-m	To specify the http/ftp/payload module	Malicious Test Website	YES
-O	Operating System detection for intelligent fuzzing (nmap)	Malicious Test Website	NO, but the operating system was detected
-o	Specify the operating system type once its known	Malicious Test Website	YES
-d	Specify the depth of the traversals and amount of fuzzing	Malicious Test Website	YES, multiple vulnerabilities at various levels
-f	Specify the particular filename to be targeted	Malicious Test Website	YES
-e	Specify the particular extension of the file or target the particular file types	Malicious Test Website	YES
-m	To specify the http/ftp/payload module	Secure Test Website	NO

-O	Operating System detection for intelligent fuzzing (nmap)	Secure Test Website	NO
-o	Specify the operating system type once its known	Secure Test Website	NO
-d	Specify the depth of the traversals and amount of fuzzing	Secure Test Website	NO
-f	Specify the particular filename to be targeted	Malicious Test Website	YES
-e	Specify the particular extension of the file or target the particular file types	Malicious Test Website	YES

Table 1: Result Analysis

5. CONCLUSION AND FUTURE SCOPE

This project work can successfully identify and detect the possibility of a directory traversal vulnerability in a php based website using the tool dotdotpwn on kali linux. After detection, the proposed system can also mitigate the attack and create a secure channel free from directory traversal vulnerability. If there is still an attack which can retrieve sensitive data, the project can successfully follow the encryption algorithm to protect the file. Lastly, the project can be extended and the encryption process can be automated. Also , the system can be upgraded to make the website viewership secure by adding user roles and access constraints.

6. REFERENCES

[1]Michael Flanders, A Simple and Intuitive Algorithm for Preventing Directory Traversal Attacks (2019), Cornell Univ.

[2]Hegui Zhu, Lewen Dai, Yating Liu, Lijun Wu, A three-dimensional bit-level image encryption algorithm with Rubik's cube method, Mathematics and Computers in Simulation (2021)

[3]N. Muraleedharan, Anna Thomas, A Traffic Monitoring and Policy Enforcement Framework for HTTP (2020)

[4]Zainab S. Alwan1, Manal F. Younis2, Detection and Prevention of SQL Injection Attack: A Survey (2017)

[5]Matthew Denis, Carlos Zena, Penetration testing: Concepts, attack methods, and defense

strategies (2016)

[6] Ionescu, V. M., & Diaconu, A. V., Rubik's cube principle based image encryption algorithm implementation on mobile devices. (2015).

[7] Bai, Y., & Chen, Z., Analysis and Exploit of Directory Traversal Vulnerability on VMware, (2015).

[8]Han, E. E., Detection of Web Application Attacks with Request Length Module and Regex Pattern Analysis (2015)

[9]Joshi, C., & Singh, U. K., Information security risks management framework – A step towards mitigating security risks in university network (2017)

[10]Shinde, P. S., & Ardhapurkar, S. B., Cyber security analysis using vulnerability assessment and penetration testing, (2016)

[11]X. Zhang *et al.*, An Automated Composite Scanning Tool with Multiple Vulnerabilities, (2019)

[12]D. Lee, B. Steed, Y. Liu and O. Ezenwoye, Tutorial: A Lightweight Web Application for Software Vulnerability Demonstration, (2021)

[13]Almutairi, Abdulwahed & Mishra, Shailendra & Alshehri, Mohammed,Web Security: Emerging Threats and Defense (2021)

[14]N. M. Min, V. Visoottiviseth, S. Teerakanok and N. Yamai, OWASP IoT Top 10 based Attack Dataset for Machine Learning, (2022)

[15]J. D. Mireles, J. Cho and S. Xu, Extracting attack narratives from traffic datasets, (2016)