A Detailed Project Report
On
Trust Check Reputation Assurance Platform

Specialization	**SAP ID**	**Name**
Cyber Security and Forensics	500086232	Arpita Kumari
Cyber Security and Forensics	500086019	Arihant Vardhan
Cyber Security and Forensics	500084998	Vivek Singh Chandel
Cyber Security and Forensics	500087856	Vikash Kumar

UPES

Department of Systemics
School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007. Uttarakhand

Dr. Akashdeep Bhardwaj   Dr. Ajay Prasad

[illegible]

Project Guide   **Cluster Head**

CANDIDATE’S DECLARATION

We hereby certify that the project entitled " **Trust Check Reputation Assurance Platform** " submitted to the Department of Systemic at the, is **School of Computer Science, University of Petroleum & Energy Studies, Dehradun** an authentic record of our work and satisfies the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING with a focus on CYBER SECURITY AND FORENSICS. An authentic record of our work completed from **JAN 2024 to MAY 2024** under the direction of **Dr. Akashdeep Bhardwaj**, Assistant Professor - Senior Scale, School of cyber security, has been submitted to the Department of Systemic.

We have not submitted the subject matter covered in this project for the granting of any other degree from this or any other University. This is to confirm that, to the best of my knowledge, the candidate's above statement is accurate.

Date: 29 May 2024

Dr. Akashdeep Bhardwaj **Dr. Ajay Prasad**

Project Guide Head

Department of Systemics

School of Computer Science
University of Petroleum & Energy Studies, Dehradun – 248001
(Uttarakhand)

ACKNOWLEDGMENT

We would like to extend our sincere thanks to our guide, **Dr. Akashdeep Bhardwaj**, for all the guidance, inspiration, and unwavering support he provided us with during the course of our project work. Without his encouragement and helpful recommendations, this effort would not have been feasible.

We would like to express our heartfelt gratitude to **Dr. Ajay Prasad**, Head of the Department, for her exceptional assistance with our project " **Trust Check Reputation Assurance Platform" at SOCS**.

We are also grateful to **Prof. S. Ravi Shankar**, the **School of Computer Science Dean at UPES**, who gave us the resources we need to successfully finish our project work.

We would like to thank all of our friends for their support and constructive feedback during the course of our project effort. Finally, we can only express our sincere gratitude to our parents for introducing us to the outside world and for all of the assistance they have provided.

Table of Contents

List Of Figures

Mentor Meet Report

S.NO	DATE	DISCUSSION
1	20th feb 2024	Finalized topic for Major project -2 with mentor.
3	7th MAR 2024	Synopsis presentation
4	24th MAR 2024	Documented SRS and report for mid-sem presentation
5	25th MAR 2024	Mid sem Presentation
6	28th MAR 2024	Analysing mistakes focused by mid-sem evaluation
7	17 APRIL 2024	Documenting the Report file
8	24th APR 2024	Adding and editing Final features

1. Abstract

As cyber-attacks and crimes have grown exponentially in recent years, Threat Intelligence has gained momentum as a response. By collecting intelligence on how attackers operate, Threat Intelligence increases defenders' understanding of the threat landscape. In a nutshell, the purpose of TI is to help defenders identify their adversaries and understand their methods of attack and techniques of attack. In order to be a step ahead of attackers, defenders need to have a detailed understanding of their moves and reinforce their infrastructure in order to anticipate their moves. There has been substantial research on Threat Intelligence (TI) and its benefits, but there is still a lack of literature on how to establish a Threat Intelligence Program (TIP). Consequently, organizations wishing to develop TIPs are on their own in this complex process.

Keywords: Threat Intelligence, Threat Intelligence Program, Information Security.

2. INTRODUCTION

2.1. Threat Intelligence

The traditional approach of Information Security (IS) against cyber-attacks has been generally reactive: attackers strike first, and defenders react. Defenders have the drawback of being constantly responding to the attackers' actions [1]. A common example is the URL filtering done by proxies. A proxy stops users from accessing sites only when these are known to be harmful. If a malicious actor uses a site that is unknown to the proxy, then the site will not be banned. Thus, attackers have the first
mover advantage.

As defenders increased their efforts to catch up with the attackers, so did the attackers. The attackers relentlessly keep developing their capabilities to be one step ahead. It soon become apparent that a new approach was needed to address this cat and mouse situation. To overcome this, Threat Intelligence (TI) emerged. Threat Intelligence is not a novel concept as it has been used in military strategy for a long time.

The objective of Threat Intelligence is to allow organisations to increase their visibility into the ever-changing threat landscape with the aim of detecting and preventing threats before they hit them. To achieve this, Threat Intelligence gathers vital intelligence about the adversaries, their methods and techniques. The information that Threat Intelligence provides can aid reinforcing the defences in the IT infrastructure as well as to provide essential intelligence to the decision-making process Threat Intelligence. This proactiveness is one of the main reasons that explain the increasing reliance of organisations on Threat Intelligence.

Fig [1.0] depicts this evolution as an iceberg metaphor. In the early stages of Threat Intelligence, analysts had access to internal logs and events; thus, they could see only the tip of the iceberg, which resulted in a limited view of the threat landscape. With the introduction of new and external sources like the dark web, forums and feeds, analyst could see the underneath, which increased their overall visibility:

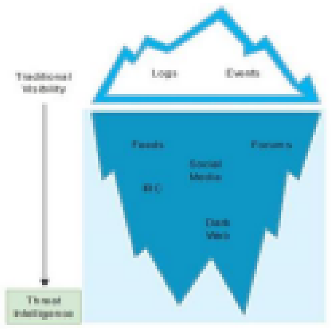


Fig [1.0] Evolution of Threat Intelligence

3. Approach:

- A comprehensive Python-based Threat Intelligence Platform designed to scrutinize URLs, IP addresses, and uploaded files for malicious attributes.
- Leveraging the VirusTotal API and a MongoDB database populated with open-source malicious IP and file hashes.

. Integration of Various APIs:

  - URL Scanning: Utilizing the VirusTotal API and Kaspersky API to assess the reputation of entered URLs.
  - IP Address Analysis: Querying multiple databases to examine IP addresses for malicious behavior.

- **File Analysis:**
  - Uploaded Files Inspection: Utilizing the VirusTotal API and Kaspersky API to identify any malicious content.
  - Hash Matching: Comparing file hashes against a MongoDB database containing known malicious file hashes.
- **MongoDB Database:**
  - Malicious IP Repository: Storing a curated list of malicious IP addresses, continually updated.
  - Malicious File Hash Repository: Maintaining a comprehensive collection of malicious file hashes.
- **User-Friendly Interface:**
  - Dashboard: Providing users with ease of searching and file uploads, and a holistic view of threat analysis results.

Preliminary User's Manual

- **Dashboard Overview:** Upon visiting the web page, users will encounter a user-friendly dashboard with two primary options.
- **Checking Threats by Link or IP Address:** Users can enter a URL or IP address in the provided field and click "Submit" to cross-reference against a comprehensive database

and VirusTotal.

• **File Analysis by File Upload:** Users can submit a file for analysis using the upload button, with supported file types including **'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif', 'zip', and 'exe'**.

4. Problem Statement:

The increasing sophistication and frequency of cyber threats pose significant challenges to organizations worldwide. Despite advancements in cybersecurity technology, many organizations struggle to effectively detect and mitigate these threats in a timely manner. Existing threat intelligence platforms often lack scalability, robust security measures, and user-friendly interfaces, hindering organizations' ability to proactively defend against evolving cyber threats.

5. Objectives:

1. Enhance Threat Detection Capabilities: Develop a comprehensive threat intelligence platform capable of accurately identifying and analysing various types of cyber threats, including malicious URLs, IP addresses, and files.

2. **Improve Scalability:** Design the platform to scale horizontally, allowing for seamless integration of new data sources and accommodating increasing volumes of threat data without compromising performance.

3. **Strengthen Security Measures:** Implement stringent security protocols, including input validation for search URLs and robust file upload validation, to mitigate the risk of potential cyberattacks such as buffer overflow and malicious uploads.

4. **Enhance User Experience:** Create a user-friendly interface with intuitive dashboard features that provide users with easy access to threat analysis tools, actionable insights, and real-time threat alerts.

Key Features of the Cyber Risk Management Dashboard:

1. Threat Intelligence Integration: The dashboard will aggregate and analyze threat intelligence data from various sources, including internal logs, external feeds, and threat intelligence platforms. This will enable organizations to gain comprehensive insights into emerging cyber threats and vulnerabilities.

2. Real-time Monitoring: The dashboard will provide real-time monitoring of critical cybersecurity metrics, such as network traffic, system logs, and user activity. Alerts will be generated for suspicious activities or anomalies, allowing organizations to respond swiftly to potential threats.

3. Customizable Dashboards: Users will have the flexibility to customize dashboards according to their specific requirements and preferences. This includes the ability to create personalized views, set up custom alerts, and prioritize cybersecurity metrics based on organizational

priorities.

4. Scalability and Performance: The dashboard will be designed to scale effortlessly to accommodate growing data volumes and user demands. High-performance architecture and advanced data processing techniques will ensure optimal performance even under heavy loads.

5. Robust Security Measures: Security will be a top priority in the design of the dashboard. It will incorporate industry-leading encryption protocols, access controls, and authentication mechanisms to safeguard sensitive data and prevent unauthorized access.

6. User-friendly Interface: The dashboard will feature an intuitive and user-friendly interface that caters to users of all skill levels. This includes interactive visualizations, drill-down capabilities, and contextual help guides to facilitate ease of use and adoption.

7. Compliance and Reporting: The dashboard will assist organizations in maintaining compliance with regulatory requirements by providing built-in compliance frameworks, audit trails, and reporting capabilities. This will streamline compliance management and reporting processes.

6. Methodology

A) Frontend

- HTML, CSS, and JavaScript were used to develop the user interface of the cyber risk management dashboard.

- HTML was employed to structure the content and ensure a semantic layout for easy navigation.

- CSS was utilized for styling the layout and visual elements, ensuring a cohesive design language throughout the dashboard.

- JavaScript was employed for dynamic interactions and client-side functionality, enhancing user experience and interactivity.

B) Backend

- Flask and Python served as the core technologies for building the backend of the dashboard.

- Flask was utilized to define routes and endpoints, manage user authentication, and interact with the database.

- Python was employed for data processing tasks, such as aggregating and analyzing threat intelligence data.

C) Data Integration and Analysis

- APIs and connectors were developed to collect data from various sources, including internal logs, external feeds, and threat intelligence platforms.

- Stream processing frameworks like Apache Kafka were utilized for real-time data processing and analysis, enabling timely detection and response to potential threats.

D) Scalability and Performance Optimization

- The dashboard was architected using microservices architecture and deployed on cloud-native platforms like AWS or Azure for scalability.

- Load balancing techniques and database sharding were implemented to optimize performance and handle growing data volumes effectively.

E) Security Implementation

- Security testing and code reviews were conducted regularly to identify and address vulnerabilities, ensuring a resilient defense against cyber threats.

F) User Experience Design

- User experience (UX) design principles were employed to create an intuitive and user-friendly interface for the dashboard.

- Usability testing and feedback sessions were conducted with end-users to iteratively improve the interface and enhance usability and adoption.

- Responsive design techniques were employed to ensure optimal viewing and interaction across different devices and screen sizes.

7. Data flow Diagram:

a) D.F.D: Level 0

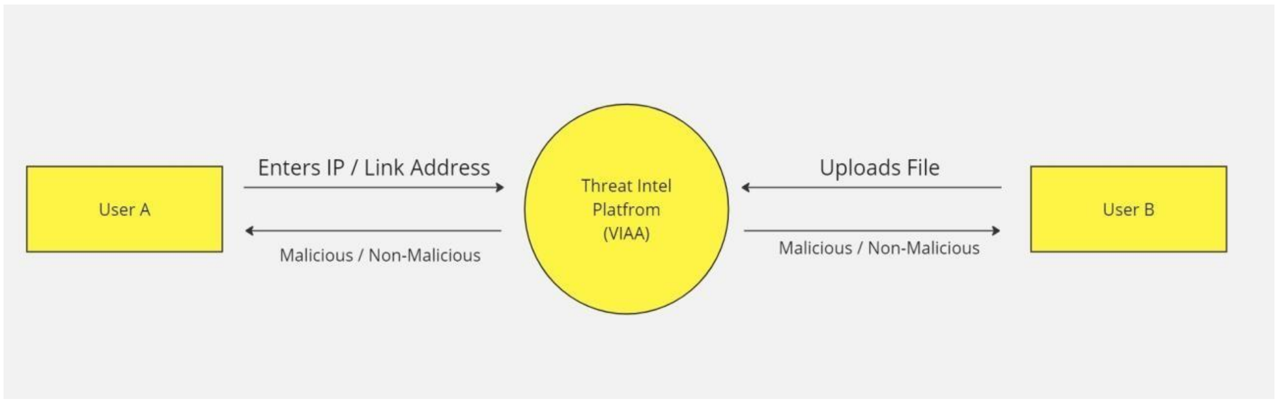

User A
Enters IP / Link Address
Malicious / Non-Malicious
Threat Intel Platfrom (VIAA)
Uploads File
Malicious / Non-Malicious
User B

Fig [2.0] Zero Level DFD

b) Data flow Diagram: Level 1

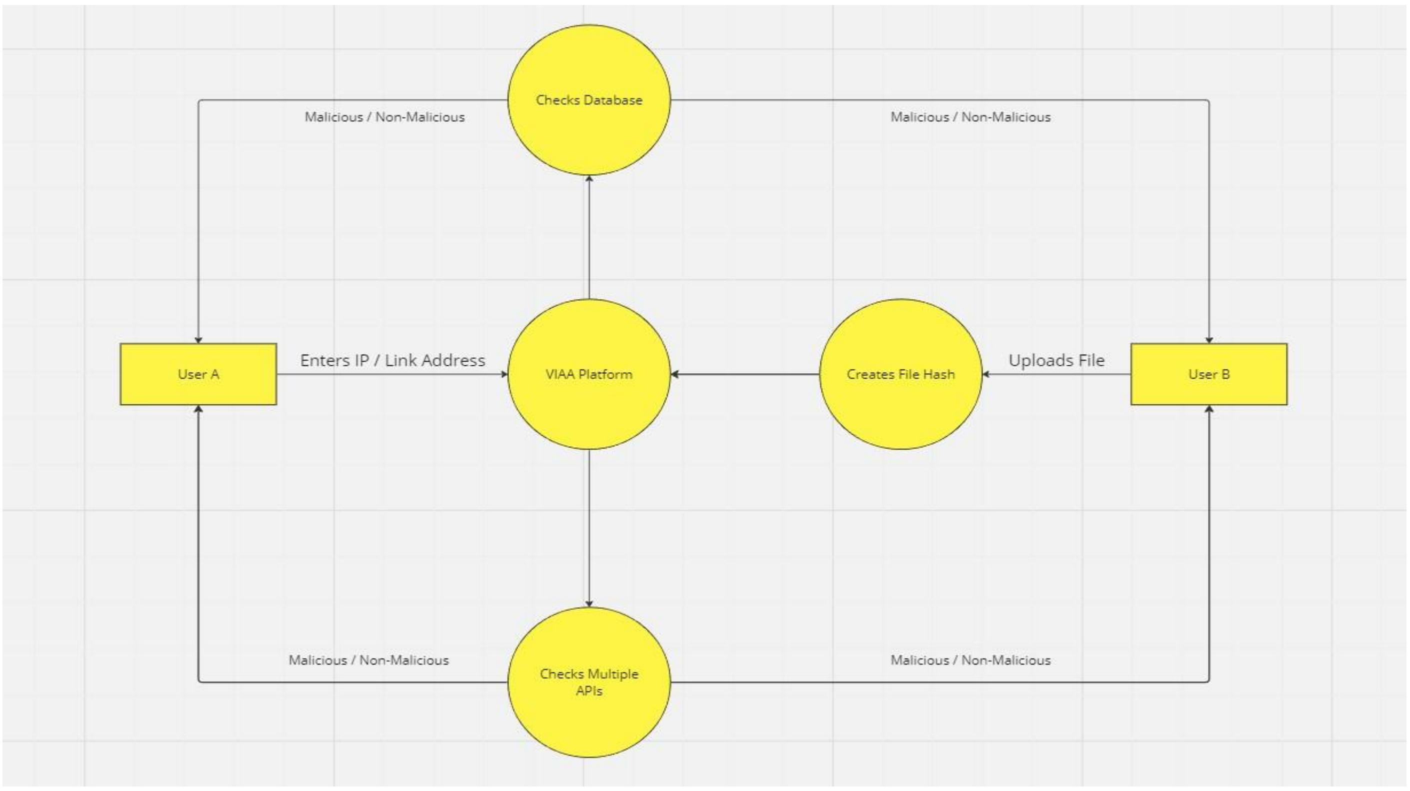
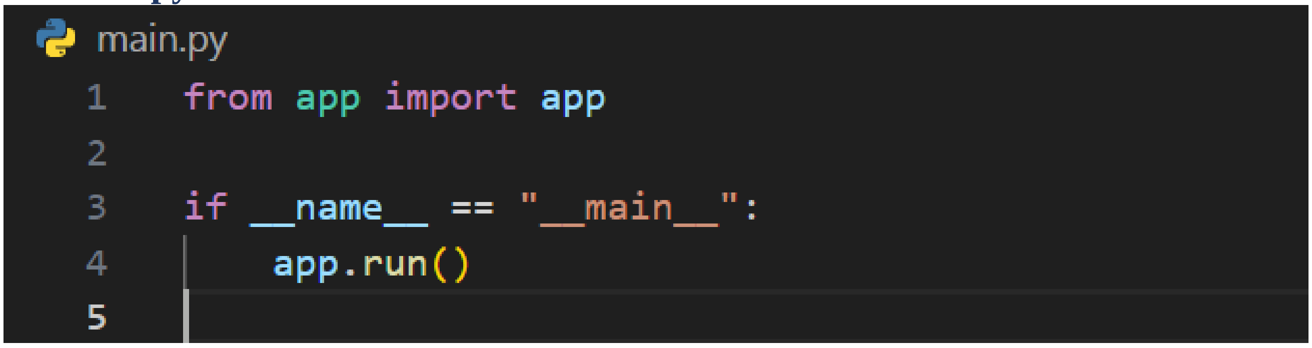


Fig [2.1]: Level 1 DFD

8. Code

8.1 main.py



main.py

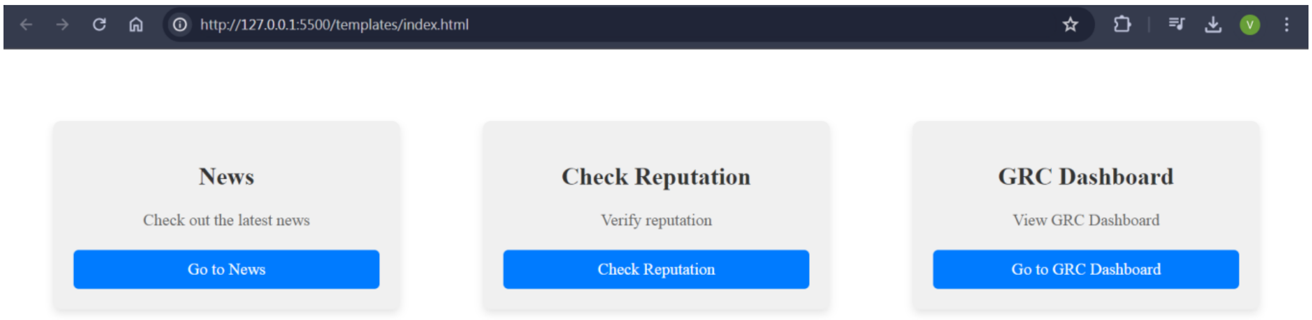
```python
from app import app

if __name__ == "__main__":
    app.run()

```

8.2 app.py

```python
import os,json,requests
from flask import Flask, render_template, request, jsonify
from DbHandler.dbFile import
dbSearch,dbHashSearch,fileScanResult,dbURLSearch,SearchIPCount,SearchURLCount,countdbhashValues,count
dbIPAddresses,countdbUrls,UploadFileCount
from flask_socketio import SocketIO
import initialSetup
from fileScripts.hashgenerator import hash_file
from fileScripts.vthashscan import VT_Request
from ipScripts.vt import checkIP
from ipScripts.vt import checkURL
from ipScripts.kasperskyip import kasperskyIP
from fileScripts.kshashscan import kasperskyHash
from ipScripts.kasperskyurl import kasperskyURL
from ipScripts.honeydb import honeyDB
from ipScripts.abuseIPDB import abuseIPDBFunc

UPLOAD_FOLDER = 'Uploads'
ALLOWED_EXTENSIONS = {'txt', 'pdf', 'png', 'jpg', 'jpeg', 'gif','zip','exe'}

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
app.config['SECRET_KEY'] = "BROWNRING"
socket = SocketIO(app)


@app.route('/', methods=['GET', 'POST'])
# def threat_intel():
def index():
    # result = latestIoC()
    searchIPCount = SearchIPCount(0)
    searchURLCount = SearchURLCount(0)
    uploadedfileCount = UploadFileCount(0)
    countDbUrls = countdbUrls()
    countdbIpAddresses = countdbIPAddresses()
    countdbHashhValues = countdbhashValues()
    # return render_template('home.html', result=result)
    return
render_template('home.html',searchIPCount=searchIPCount,searchURLCount=searchURLCount,uploadedfileCoun
t=uploadedfileCount,\
                            countdbhashValues=countdbHashhValues,countdbIPAddresses=countdbIpAddresses,\
                            countdbUrls=countDbUrls)

@app.route("/about")
def aboutPage():
    return render_template('about.html')

# @app.route("/services")
# def servicesPage():
#     return render_template('services.html')

@app.route("/contact")
def contactPage():
    return render_template('contact.html')

@app.route('/checkentity',methods=["GET"])
def checkentitiy():
    result = None
    if request.method == 'GET':

        # ip = request.form.get('q')
        ip = request.args.get('ip')
        if len(ip)<=20:
        # print(ip)
            result = dbSearch(ip)
            # print(result)
            checkCount = SearchIPCount(1)

            socket.emit("checkentity",{'dbResult':result,'checkCount':checkCount,'countType':'ip'})
            return jsonify(isError = False,
                            message = "Success",
                            statusCode = 200,
                            data = result), 200
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})
            return jsonify(isError = False,
                            message = "Success",
                            statusCode = 200,
                            data = result), 200

@app.route('/checkvtip',methods=["GET"])
def checkvtip():
    result = None
    if request.method == 'GET':
        try:
            ip = request.args.get('ip')
            if len(ip)<=20:

                result = checkIP(ip)
                socket.emit("checkvtip",{'vtResult':(result)})
                return jsonify(isError = False,
                                message = "Success",
                                statusCode = 200,
                                data = result), 200
            else:
                socket.emit("errormsg",{'errorMsg':"Length exceeded."})
                return jsonify(isError = False,
                            message = "Success",
                            statusCode = 200,
                            data = result), 200
        except Exception as e:
            print(e)




@app.route('/checkurl',methods=["GET"])
def checkurl():
    result = None
    if request.method == 'GET':

        url = request.args.get('url')
        if len(url) <=50:
        # print(ip)
            result = dbURLSearch(url)
            checkCount = SearchURLCount(1)

            socket.emit("checkentity",{'dbResult':result,"checkCount":checkCount,'countType':'url'})
            return jsonify(isError = False,
                            message = "Success",
                            statusCode = 200,
                            data = result), 200
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})
            return jsonify(isError = False,
                        message = "Success",
                        statusCode = 200,
                        data = result), 200


@app.route('/checkvturl',methods=["GET"])
def checkvturl():
    result = None
    if request.method == 'GET':
        try:
            url = request.args.get('url')
            if len(url) <=50:
                result = checkURL(url)
                socket.emit("checkvtip",{'vtResult':(result)})
                return jsonify(isError = False,
                                message = "Success",
                                statusCode = 200,
                                data = result), 200
            else:
                socket.emit("errormsg",{'errorMsg':"Length exceeded."})
                return jsonify(isError = False,
                            message = "Success",
                            statusCode = 200,
                            data = result), 200
        except Exception as e:
            print(e)


@app.route('/fileUpload',methods=["GET","POST"])
def fileUpload():
    if request.method == 'POST':
        f = request.files['file']
        fileName = f.filename
        try:
            if "Uploads" in os.getcwd():
                f.save(os.path.join(os.getcwd(),fileName))
            else:
                f.save(os.path.join(app.config['UPLOAD_FOLDER'], fileName))
        except Exception as e:
            print("file upload exception: ",e)

    return '''
    File uploaded successfuly
    '''

@app.route("/checkFile",methods=["GET"])
def fileCheck():
    if request.method == "GET":
        fileName = request.args.get('fileName')
        print("FileName: ",fileName)

    if "Uploads" in os.getcwd():
        filePath = os.path.join(os.getcwd(),fileName)
    else:
        os.chdir("Uploads")
        filePath = os.path.join(os.getcwd(),fileName)
    print(filePath)
    fileHashValue = hash_file(filePath)
    print("Hash: ",fileHashValue)


    requests.get(f"http://127.0.0.1:5000/checkvthash?hashValue={fileHashValue}")
    requests.get(f"http://127.0.0.1:5000/checkkshash?hashValue={fileHashValue}")
    result = dbHashSearch(fileHashValue)
    checkCount = UploadFileCount(1)
    fileScanResult(fileHashValue,{"db":result})

    socket.emit("checkHashValue",{'dbResult':result, 'checkCount':checkCount,'countType':'fileName'})
    return jsonify(isError = False,
                    message = "Success",
                    statusCode = 200,
                    data = result), 200



#  for url
@app.route('/checkurlentity', methods=["GET"])
def checkurlentity():
    result = None
    if request.method == 'GET':
        url = request.args.get('url')
        if len(url)<=50:
            result = checkURL(url)
            socket.emit("checkurlentity", {'urlResult': result})
            return jsonify(isError=False, message="Success", statusCode=200, data=result), 200
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})
            return jsonify(isError = False,
                        message = "Success",
                        statusCode = 200,
                        data = result), 200

@app.route('/checkvthash',methods=["GET"])
def checkvthash():
    result = None
    if request.method == 'GET':
        try:
            hashValue = request.args.get('hashValue')
        except Exception as e:
            print(e)
        result = VT_Request(hashValue)
        fileScanResult(hashValue,{"vt":result["Status"]})
        # print(type(result))

    socket.emit("checkvthash",{'vtResult':(result)})
    return jsonify(isError = False,
                    message = "Success",
                    statusCode = 200,
                    data = result), 200

@app.route('/checkksip', methods=["GET"])
def checkksip():
    result = None
    if request.method == 'GET':
        ipValue = request.args.get('ip')
        if len(ipValue)<=20:
            result = kasperskyIP(ipValue)
            socket.emit("checkksipresult", {'ipResult': result})
            return jsonify(isError=False, message="Success", statusCode=200, data=result), 200
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})
            return jsonify(isError = False,
                        message = "Success",
                        statusCode = 200,
                        data = result), 200


@app.route('/honeyDBIP', methods=["GET"])
def checkhoneydbip():
    result = None
    if request.method == 'GET':
        ipValue = request.args.get('ip')
        if len(ipValue)<=20:
            result = honeyDB(ipValue)
            socket.emit("checkhoneydbipresult", {'ipResult': result})
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})

    return jsonify(isError=False, message="Success", statusCode=200, data=result), 200

@app.route('/abuseDBIP', methods=["GET"])
def abuseDBIP():
    result = None
    if request.method == 'GET':
        ipValue = request.args.get('ip')
        if len(ipValue)<=20:
            result = abuseIPDBFunc(ipValue)
            socket.emit("checkabusedbipresult", {'ipResult': result})
        else:
            socket.emit("errormsg",{'errorMsg':"Length exceeded."})
    return jsonify(isError=False, message="Success", statusCode=200, data=result), 200


@app.route('/checkkshash',methods=["GET"])
def checkkshash():
    result = None
    if request.method == 'GET':
```

```python
        try:
            hashValue = request.args.get('hashValue')
        except Exception as e:
            print(e)
        result = kasperskyHash(hashValue)
        fileScanResult(hashValue,{"ks":result['Status']})

    socket.emit("checkksipresult",{'ipResult':result})
    return jsonify(isError = False,
                    message = "Success",
                    statusCode = 200,
                    data = result), 200

@app.route('/checkksurl',methods=["GET"])
def checkksurl():
    result = None
    if request.method == 'GET':
        try:
            url = request.args.get('url')
            if len(url)<=50:
                result = kasperskyURL(url)
                socket.emit("checkksipresult",{'ipResult':result})
            else:
                socket.emit("errormsg",{'errorMsg':"Length exceeded."})

        except Exception as e:
            print(e)

    return jsonify(isError = False,
                    message = "Success",
                    statusCode = 200,
                    data = result), 200

if __name__ == "__main__":
    socket.run(app,debug=False,host="0.0.0.0")
    # app.run(debug=True)
```

9. Output:



News

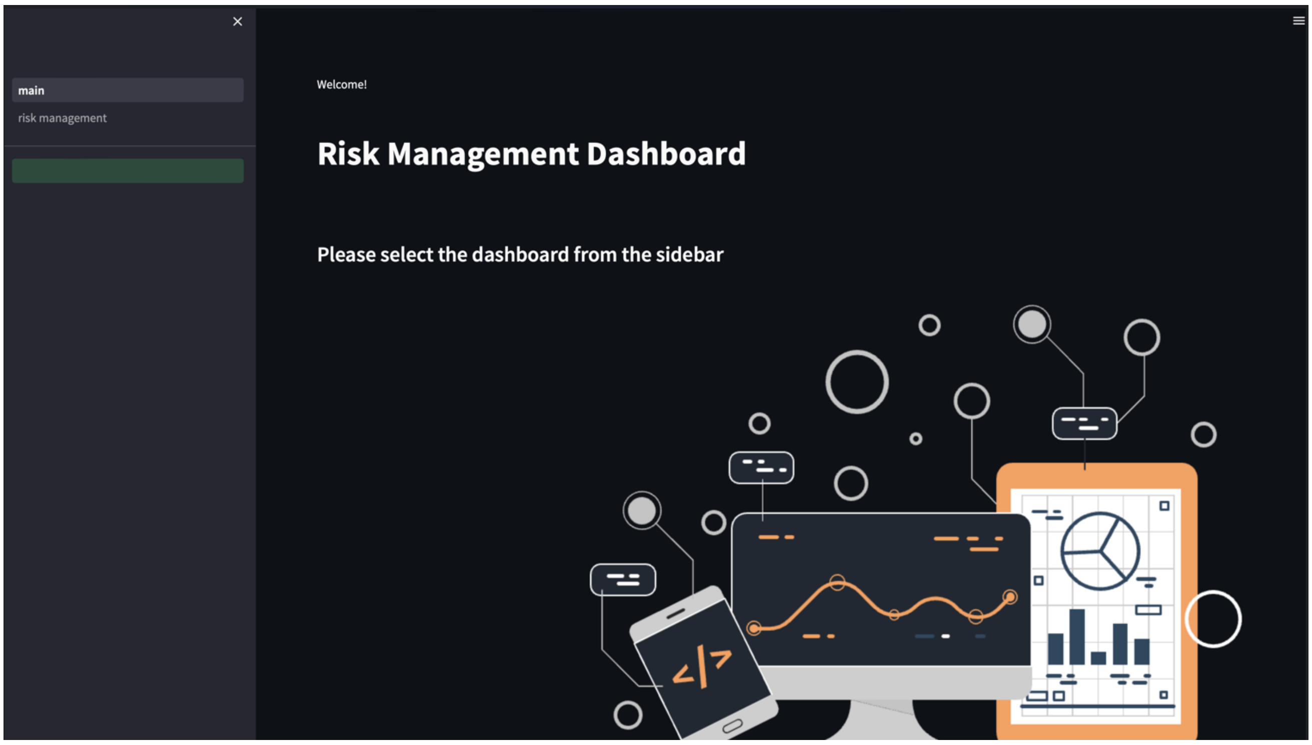
Check out the latest news

Check Reputation

Verify reputation

GRC Dashboard

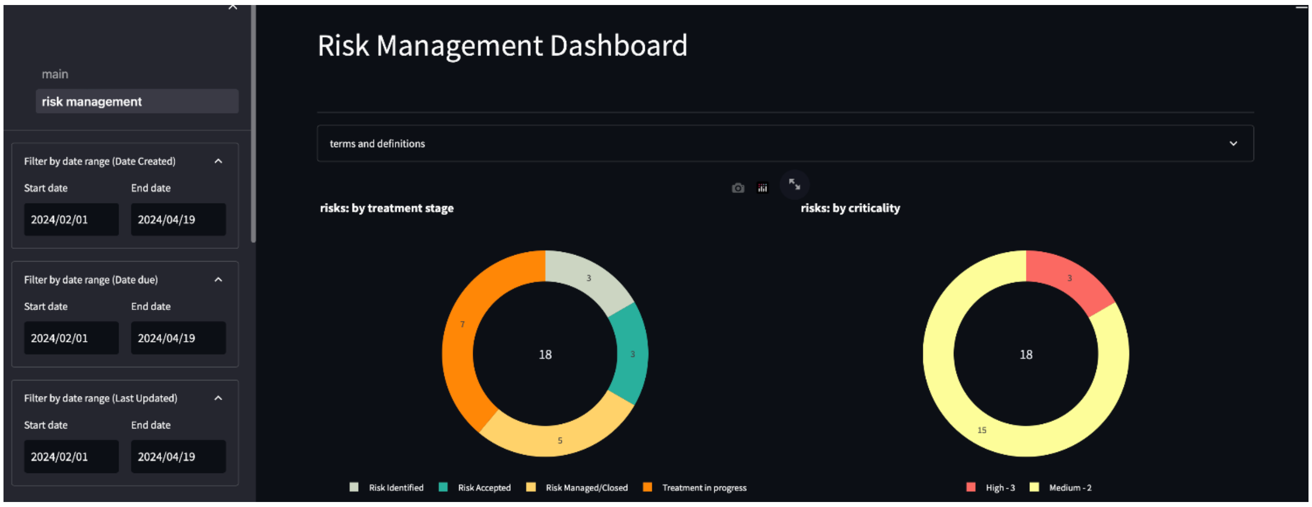
View GRC Dashboard



Welcome!

Risk Management Dashboard

Please select the dashboard from the sidebar



Risk Management Dashboard

Filter by date range (Date Created)

Start date 2024/02/01

End date 2024/04/19

Filter by date range (Date due)

Start date 2024/02/01

End date 2024/04/19

Filter by date range (Last Updated)

Start date 2024/02/01

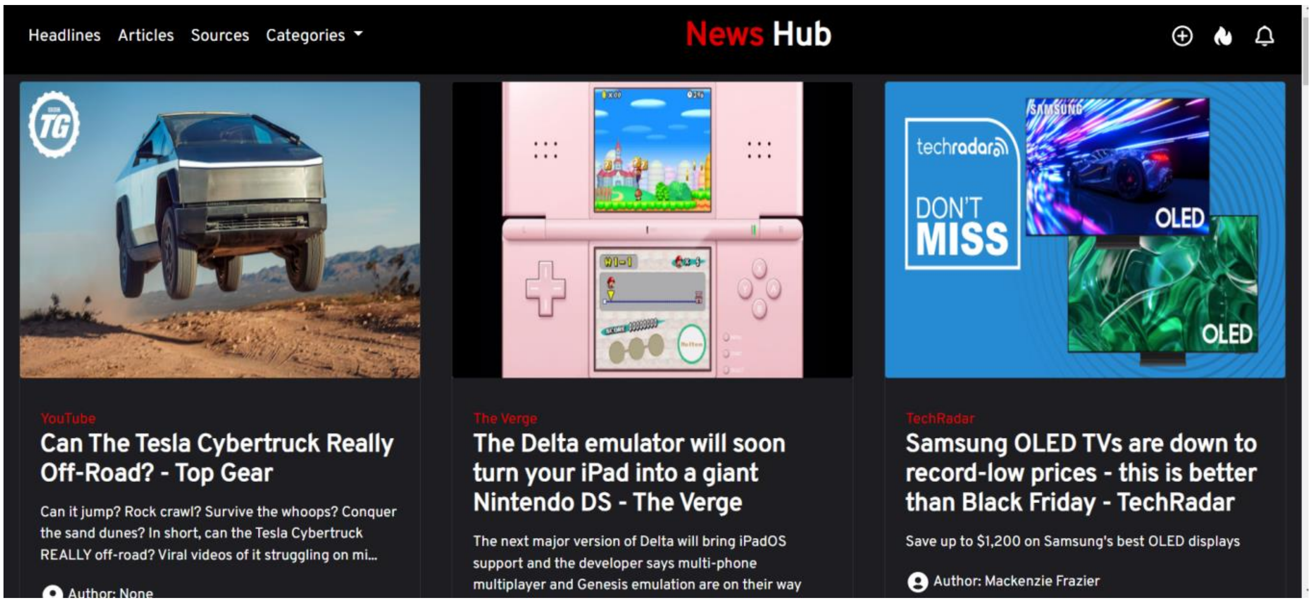
End date 2024/04/19

risks: by treatment stage

⬜ Risk Identified 🟩 Risk Accepted 🟨 Risk Managed/Closed 🟧 Treatment in progress

risks: by criticality

🟥 High - 3 🟨 Medium - 2



Headlines Articles Sources Categories

News Hub

YouTube

Can The Tesla Cybertruck Really Off-Road? - Top Gear

Can it jump? Rock crawl? Survive the whoops? Conquer the sand dunes? In short, can the Tesla Cybertruck REALLY off-road? Viral videos of it struggling on mi...

Author: None

The Verge

The Delta emulator will soon turn your iPad into a giant Nintendo DS - The Verge

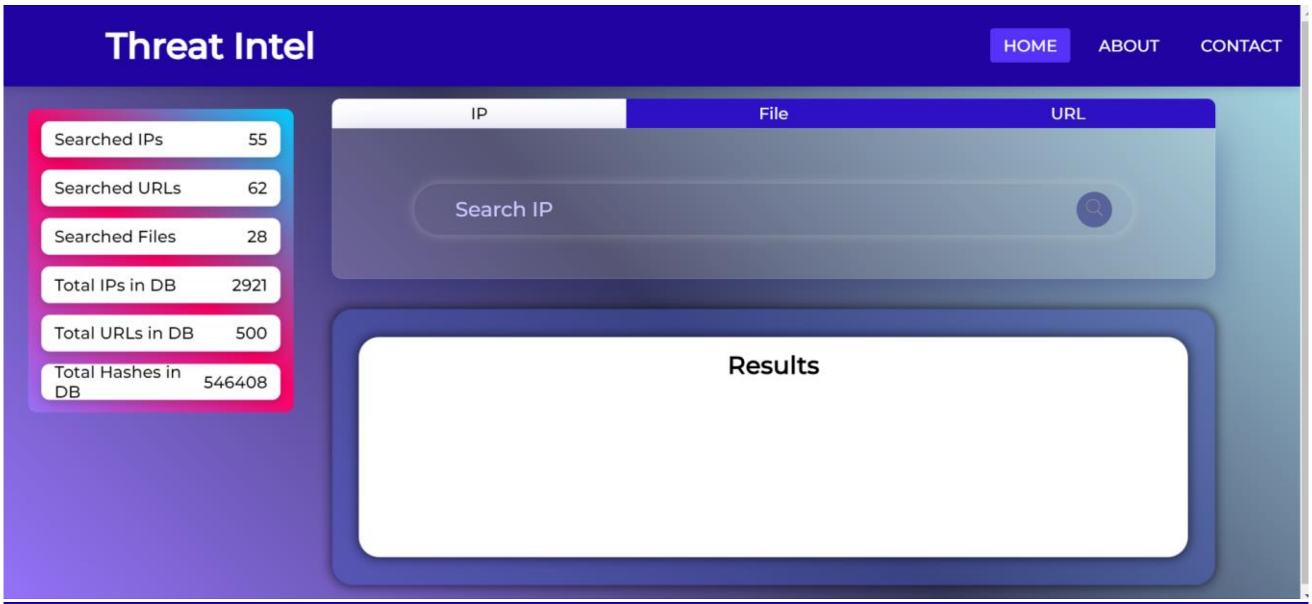
The next major version of Delta will bring iPadOS support and the developer says multi-phone multiplayer and Genesis emulation are on their way

TechRadar

Samsung OLED TVs are down to record-low prices - this is better than Black Friday - TechRadar

Save up to $1,200 on Samsung's best OLED displays

Author: Mackenzie Frazier



Threat Intel

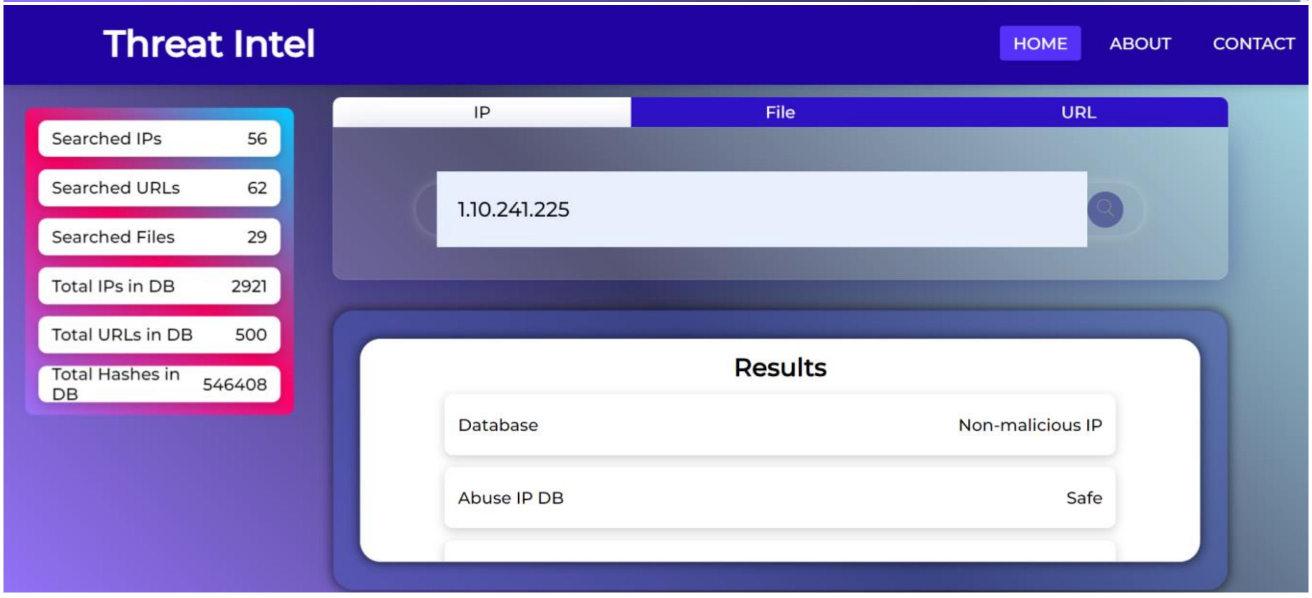
HOME ABOUT CONTACT

Searched IPs	55
Searched URLs	62
Searched Files	28
Total IPs in DB	2921
Total URLs in DB	500
Total Hashes in DB	546408

IP File URL

Search IP

Results



Threat Intel

HOME ABOUT CONTACT

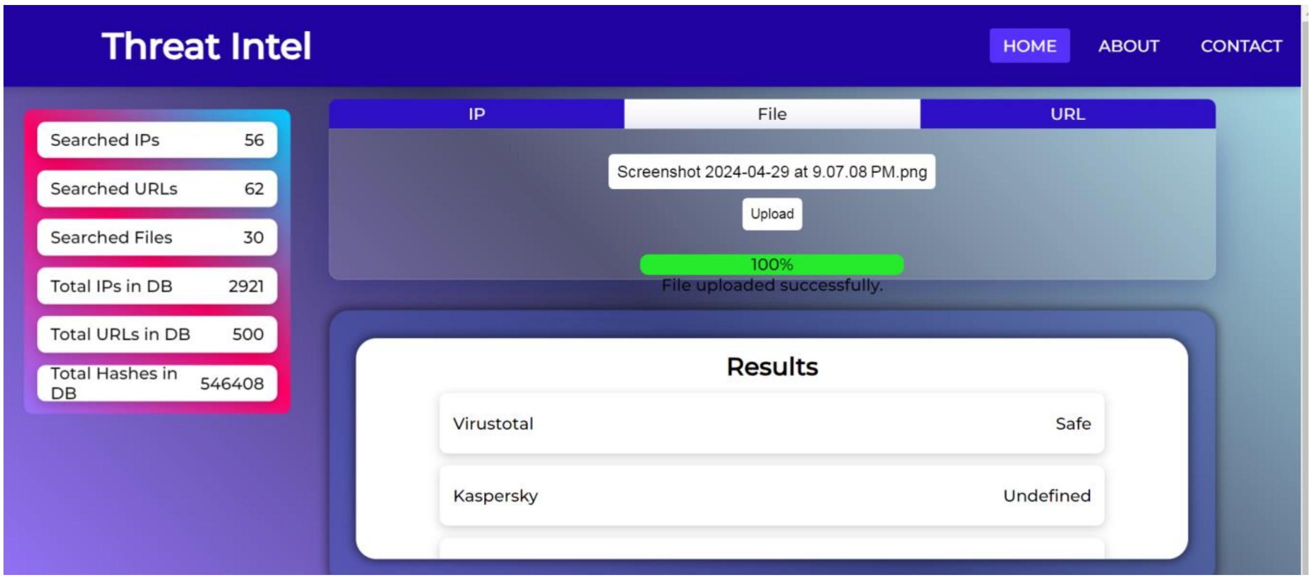
Searched IPs	56
Searched URLs	62
Searched Files	29
Total IPs in DB	2921
Total URLs in DB	500
Total Hashes in DB	546408

IP File URL

1.10.241.225

Results

Database	Non-malicious IP
Abuse IP DB	Safe



Searched IPs	56
Searched URLs	62
Searched Files	30
Total IPs in DB	2921
Total URLs in DB	500
Total Hashes in DB	546408

Screenshot 2024-04-29 at 9.07.08 PM.png

100%

File uploaded successfully.

Virustotal	Safe
Kaspersky	Undefined

10. SWOT

Strengths:

- Comprehensive Threat Intelligence Integration: The ability to aggregate and analyze threat intelligence data from various sources provides organizations with valuable insights into emerging cyber threats and vulnerabilities.

- Real-time Monitoring and Alerting: The dashboard's capability to provide real-time monitoring of critical cybersecurity metrics enables organizations to detect and respond swiftly to potential threats, enhancing their cyber resilience.

- Customizable Dashboards: The flexibility to customize dashboards according to specific requirements and preferences empowers users to tailor the dashboard to their needs, improving usability and effectiveness.

- Scalability and Performance: The design of the dashboard to scale effortlessly and optimize performance ensures that it can accommodate growing data volumes and user demands, enhancing its long-term viability and usefulness.

- Robust Security Measures: The implementation of industry-leading encryption protocols, access controls, and authentication mechanisms safeguards sensitive data and prevents unauthorized access, enhancing data security and compliance.

- User-friendly Interface: The intuitive and user-friendly interface of the dashboard facilitates ease of use and adoption, improving user satisfaction and engagement.

Weaknesses:

- Complexity of Integration: Integrating data from various sources and ensuring compatibility with existing systems may pose challenges and require substantial effort and expertise.

- Initial Development Costs: Developing a comprehensive cyber risk management dashboard with advanced features may require significant upfront investment in terms of time, resources, and technology.

- Training and Adoption: Ensuring that users are adequately trained to use the dashboard effectively and promoting its adoption across the organization may require ongoing effort and support.

Opportunities:

- Market Demand: The increasing sophistication and frequency of cyber threats create a growing demand for effective cyber risk management solutions, presenting opportunities for the adoption and expansion of the dashboard.

- Partnership Opportunities: Collaborating with cybersecurity vendors, threat intelligence providers, and industry experts can enhance the dashboard's capabilities and market reach, opening up new partnership opportunities.

- Regulatory Compliance: The need for organizations to comply with regulatory requirements related to cybersecurity presents opportunities for the dashboard to serve as a compliance management tool, attracting organizations seeking to meet regulatory obligations.

Threats:

- Competition: The presence of established players and competitors in the cybersecurity and risk management space may pose a threat to the dashboard's market penetration and competitiveness.

- Rapid Technological Advancements: The rapid pace of technological advancements in cybersecurity and threat intelligence may require continuous updates and enhancements to the dashboard to remain relevant and effective.

- Cybersecurity Risks: The dashboard itself may become a target for cyber attacks, posing risks to the confidentiality, integrity, and availability of sensitive data and information. Implementing robust security measures is essential to mitigate this threat.

11. Pert chart

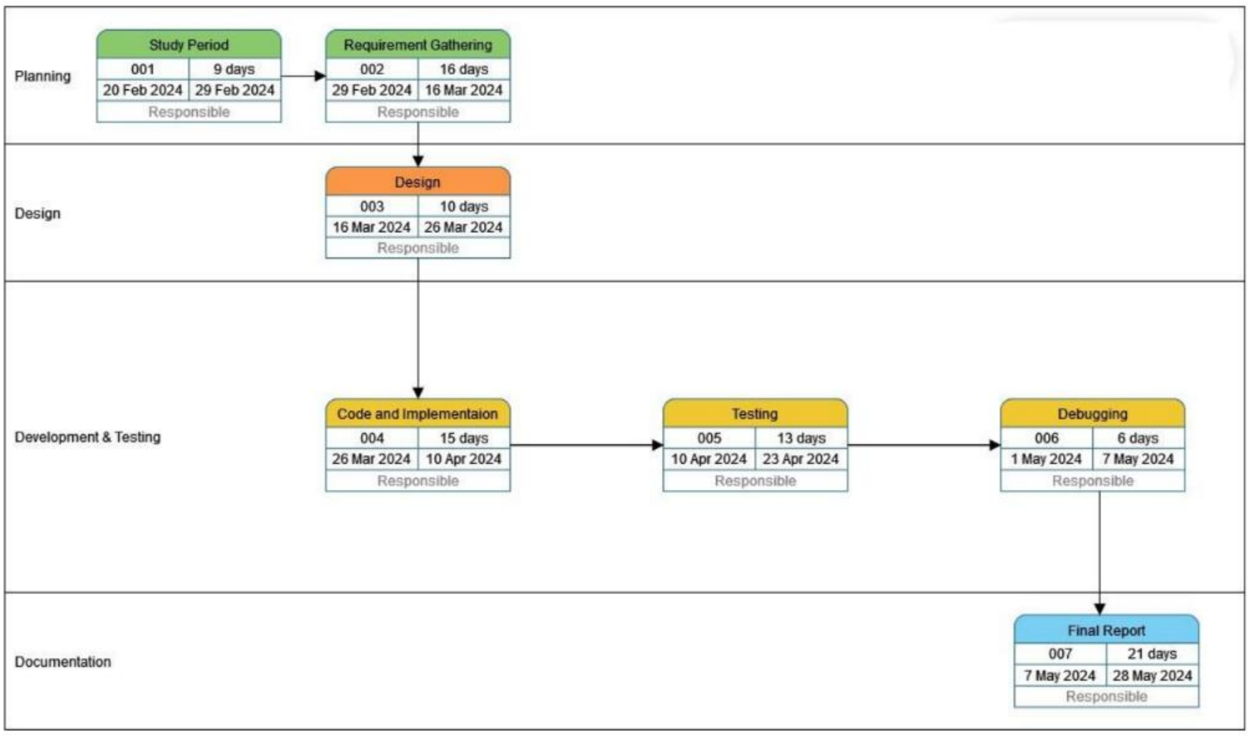


Fig. 7

12. Future Scope

- Advanced Analytics and Machine Learning:
  Integrate advanced analytics techniques and machine learning algorithms into the dashboard to enhance its ability to detect and predict cyber threats, identify patterns and trends, and provide actionable insights for proactive risk management.

- Automated Remediation and Response:
  Develop automation capabilities within the dashboard to enable automated remediation and response actions for detected threats, such as isolating compromised systems, blocking malicious IP addresses, and updating firewall rules in real-time.

- Integration with Security Orchestration, Automation, and Response (SOAR) Platforms: Integrate the dashboard with SOAR platforms to streamline incident response processes, orchestrate security workflows, and automate routine tasks, enabling faster and more efficient incident resolution.

- Predictive Analytics and Risk Forecasting: Implement predictive analytics models to forecast future cyber risks and vulnerabilities based on historical data, threat intelligence feeds, and external factors, enabling organizations to proactively mitigate potential threats before they materialize.
- Enhanced User Experience and Collaboration: Continuously improve the dashboard's user interface and experience based on user feedback and usability testing, incorporating features such as collaboration tools, interactive visualizations.

13. References

- Almadhoob, A., & Valverde, R. (2014). Cybercrime Prevention in The Kingdom of Bahrain via IT Security Audit Plans. Journal of Theoretical and Applied Information Technology, 65(1), 274–292.

- Calder, A., Watkins, S., & Governance, I. T. (2008). A Manager's Guide to Data Security and ISO 27001/ISO 27002. Kogan Page. Cano, J. (2017).

- The AREM Window: A Strategy to Anticipate Risk and Threats to Enterprise Cyber Security (Vol. 5).
- ISACA Journal. DeSouza, E., & Valverde, R. (2015).

- An Employee-based Risk Management Strategy for reducing security incidents in a Canadian PHIPA Regulated Environment. In Proceedings of the International Conference on Innovations in Computer Science and Information Technology (ICICSIT 2015), Hyderabad,India.

- Academic Press. Eppler, M. J., & Aeschimann, M. (2009). A systematic framework for risk visualization in risk management and communication. Risk Management, 11(2), 67–89. doi:10.1057/rm.2009.4 Fenz, S., Ekelhart, A., & Neubauer, T. (2011). Information Security Risk Management: In which security solutions is it worth investing? CAIS, 28, 22.