IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Obsidian Circuit: DFIR Tool

1. Prof. Uday Mande
Computer Science and Engineering
MIT ADT University
Pune, India

2. Kashvi Darda
Computer Science and Engineering
MIT ADT University Pune, India

3. Soham Jagtap
Computer Science and Engineering
MIT ADT University
Pune, India

4. Samyak Pokharna
Computer Science and Engineering
MIT ADT University
Pune, India

5. Sharan Ramkrishna Shenoy
Computer Science and Engineering
MIT ADT University
Pune, India

Abstract— Digital Forensics and Incident Response (DFIR) tool Obsidian Circuit is modular and has a graphical user interface (GUI). It has been designed in such a way that it becomes easier to observe digital security incidents.

This system was created using Python and Streamlit and integrates File Analysis, Network Analysis, and Log Analysis into a single system to conduct a quick forensic investigation. File Analysis module analyzes the metadata, cryptographic hashes, permissions, and ownership data of a file to ensure that it remains intact and it has not been tampered with. The Network Analysis module will scan at PCAP files in search of various items such as abnormal traffic, port scanning, bizarre DNS queries and potential data exfiltration. Detecting Brute-force logins, unauthorized file access, and suspicious data transfers are some of the things that the Log Analysis module detects using rule-based detection that inspects system logs.

This tool enables it to be more efficient in analysing incidents with a faster and more accurate display of forensic results in a simplistic style graphical interface.

The reason why Obsidian Circuit is favored by schools, cybersecurity training, and small businesses is that it is easy to understand and is not as costly as more complex enterprise forensic tools.

KEYWORDS:

Digital Forensics, Incident Response, DFIR, File Integrity Verification, Metadata Analysis, Network Traffic Analysis, Log Analysis, PCAP Analysis, Threat Detection, Intrusion Detection, Cybersecurity, Forensic Investigation, Rule-Based Detection, Security Monitoring, Data Exfiltration Detection, Streamlit, Python, Security Analytics, Incident Investigation, Threat Intelligence

I. INTRODUCTION

In the modern digital environment, businesses are facing more cybersecurity threats, such as unauthorized access, insider misuse, network intrusions, and data theft. Digital Forensics and Incident Response (DFIR) has become a very important aspect of cybersecurity operations since such attacks are becoming increasingly intricate. DFIR assists investigators to figure out where, what, and how bad security incidents are by systematically collecting and analyzing evidence. This aids in containing and recovering events in a short period of time.

Despite the availability of advanced forensic tools, many of the already existing tools are fragmented and it requires an analyst to use different tools on different portions of an investigation. As an example, many tools are typically necessary to verify file integrity, analyze packet captures, and view logs. This bifurcated approach results in investigations being slow, biosis hard to control, and may reduce the overall speed and accuracy of incident response.

It is recommended to use Obsidian Circuit as a single, GUI-based DFIR platform that includes three valuable forensic tools: File Analysis, Network Analysis, and Log Analysis. The File Analysis module checks the integrity of files by looking at their metadata, cryptographic hashes, access permissions, and extended attributes. It also checks the indications of tampering. The integrity is checked by using cryptographic hashes, access permissions, and extended attributes, as well as the looks of whether anything has been changed. The Network Analysis module sees for strange patterns in packet capture (PCAP) files, like port scanning, strange DNS behavior, and data transfers that aren't allowed. The Log Analysis module will scan through system and application logs to identify abnormal activities, such as failed attempts to log in, unauthorized access and unusual user behavior. Obsidian Circuit is created using Python and Streamlit and simplifies and streamlines forensic investigations. The tool combines several forensic functions into one platform, which makes it less reliant on separate tools, speeds up incident analysis, and makes forensic investigations more effective. This is particularly applicable in the school, training in cybersecurity, and small-scale operation that require a light, cheap, and easy to use DFIR solutions.

Obsidian Circuit is a unified and streamlined file, network and log analysis into a single, easy-to-use GUI-based platform, making digital forensic investigation more efficient and optimized. It speeds up and makes incident investigations more accurate, optimizing the forensic workflow, and minimizing the requirements of various independent tools. It

is excellent in academic applications, training in cybersecurity, and small-scale operational settings because it is light and can be easily used. Overall, Obsidian Circuit demonstrates how an integrated DFIR solution can enable incident response to be more efficient and at the same time be easy to use and useful to a lot of people.

II. LITERATURE REVIEW

Digital Forensics and Incident Response (DFIR) tools are very critical for cybersecurity because they help identify, examine, and respond to digital security problems. As cyberattacks become more complicated, researchers have come up with various forensic tools for looking at files, checking network traffic, and looking at logs. Even though these tools make the investigation process better, a lot of them work on their own, so analysts have to switch between different tools to finish an examination.

In "Comparative Analysis of The Integrated Digital Forensic Tools for Digital Forensic Investigations" (2020), Lee and Soh Et Al. [X] Conducted an Analysis of Both the Features, Usability, And Limitations of Several Integrated Forensic Tools Used in The Conduct of Digital Investigations. Their Findings Elaborated Upon the Need to For Those Who Conduct Digital Forensic Investigations to Use Multiple Forensics Functions, From Multiple Areas of Forensic Expertise, In Combination, As A Combined Tool, To Facilitate Investigative Efficiency. Most Of the Integrated Tools However, Were Typically Designed for Use with Enterprise Level Businesses and Were Resource Hungry and Difficult to Deploy and Operate in Environments Where Compactness Is Required. As Such the Proposed System Obsidian Circuit Will Provide a Light Weight Modular DFIR / Digital Forensics / Incident Response Tool Which Will Be Suitable for Use in Academic, Educational, And Small Enterprise Environments. [1]

Wu, Breitinger, and O'Shaughnessy et al. [X] (2019) discussed advances in forensic tools for artifact extraction, automated evidence processing and forensic workflow optimization in "Digital Forensic Tools: Recent Advances and Enhancing the Status Quo" (2020). Their study showed that automation can greatly decrease the time of investigation and enhance the handling of evidence. However, the reviewed tools still needed multiple integrations to manage logs, files and network evidence which hindered the simplicity of the workflow. Therefore, Obsidian Circuit offers a simplified GUI-based framework to automate the forensic analysis process over multiple sources of evidence. [2]

According to Sun, Zhang, and Wang et al. [X] in their paper entitled "NLP-Based Digital Forensic Investigation Platform for Online Communications" published in 2021, a digital forensic investigation platform utilizing NLP algorithms could analyze digital evidence by examining the content of online communications and detecting suspicious activity patterns. The research introduced a framework for automating the process of interpreting evidence in a domain-specific environment, especially when textual artifacts were involved. However, the framework does not address the problem of conducting low-level digital forensics, including PCAP and file integrity checking. [3]

In 2023, researchers used Regex-based Log Parsing and Event ID Mapping techniques to improve the detection of suspicious activities in system and server logs. This approach achieved high accuracy in identifying automated attack scanners, suspicious requests, and path traversal attempts by matching predefined patterns with log entries. While this method worked well for rule-based detection, it needed regular updates to heuristic blacklists and suspicious agent definitions to stay effective against changing attack patterns. [4]

Karam Muhammed and Najla Badi et al. [X], "Digital Forensic Tools: A Literature Review", (2023) reviewed various digital forensic tools used for evidence collection, network forensics and live analysis. Their research showed that although existing tools are effective in forensic capabilities, most are highly specialized and require expertise in multiple environments to complete incident investigation. This lack of a single, unified analysis complicates matters and lengthens response times. To mitigate these issues, Obsidian Circuit integrates core forensic modules into a single interface to enhance usability and speed of investigation. [5]

In **2022**, **PCAP OSI Layer Dissection and TCP Flag Bitmasking** methods were used for analyzing network packet captures to detect malicious communication patterns. These techniques successfully identified **internal-to-external data exfiltration**, TCP handshake anomalies, and suspicious connection attempts by examining protocol-level behavior. However, processing large PCAP files entirely in memory resulted in performance bottlenecks and high buffer usage, limiting the scalability of the approach in large forensic investigations.[6]

In 2021, a group of forensic analysts combined Magic Bytes and Shannon's Entropy Calculation to perform enhanced file level forensic analysis. The combination of these two techniques enabled analysts to accurately detect disguised executables, packed malware, and altered file signatures by examining the entropy values of the files and validating the headers of the files. However, despite being effective, legitimate encrypted or compressed files had very similar entropy characteristics, resulting in false positives and an increased need for additional manual verification of the files. [7]

Matoušek and team et al. [X], in *"Netfox Detective: A Novel Open-Source Network Forensics Analysis Tool"* (2020), proposed an open-source network forensic platform for analyzing packet captures, reconstructing TCP streams, and identifying suspicious network sessions. Their system improved the detection of network anomalies and enabled efficient investigation of suspicious traffic patterns. However, the tool was finite to **network forensic analysis only**, requiring investigators to rely on separate platforms for file and log investigations. To address this limitation, **Obsidian Circuit** embeds network **analysis with file and log analysis** in a single graphical environment.[8]

Recent research shows that most forensic tools are designed to operate within a single investigative domain, such as network forensics, file forensics, or log analysis. While these tools are effective at improving detection and analysis within their specific domain, they often lack the ability to integrate findings across multiple sources of evidence. This limitation prevents the creation of a centralized view, which is critical for efficient and accurate incident response. Consequently, analysts are frequently required to use multiple independent tools to piece together information, increasing their workload, slowing down investigations, and introducing more opportunities for human error. Furthermore, many enterprise-grade forensic solutions are expensive and highly complex, requiring specialized knowledge to operate effectively. These factors make such tools impractical for educational purposes, small-scale investigations, or lightweight operational environments, where ease of use and cost-effectiveness are key considerations.

III. METHODOLOGY

PHASE 1: System Initialization

The system begins with loading needed forensic libraries, and the start of the Streamlit-based interface. It loads the modules that will be used to analyze files, network and logs, and prepares the dashboard where artifacts will be uploaded, and results will be visualized.

PHASE 2: Artifact Upload and Routing

During this step, the investigator uploads the digital evidence, which can be a suspicious file, packet capture file or a system log file. Once the type of artifact has been authenticated by the system, it automatically directs the uploaded input to the specific forensic analysis module. Such automated routing will be used to ensure that the processing of the evidence takes place in the proper environment without having to involve manual intervention.

PHASE 3: Forensic Analysis Execution

After routing the artifact, the analysis module chosen carries out the forensic test. The file analysis module will extract metadata, compute cryptographic hash, verify file signatures, and assess the entropy of files to identify tampered or suspicious files. The network analysis module takes the data in a packet capture to identify anomalies like unusual traffic behavior, suspicious DNS activity and potential data exfiltration attempts. The log analysis module is the module which interprets the system logs with the help of pattern recognition techniques in order to detect the cases of brute-force logins, unauthorized access and other suspicious activity. Any suspicious results created in the process of analyzing are saved to be further analyzed.

PHASE 4: Risk Assessment

Once the forensic analysis has been carried out, the system compiles all the results of all modules and evaluates their level of seriousness. All of the indicators of suspicion are categorized based on the level of risk involved and a cumulative risk score is compiled to reflect the severity of the identified risks. Such an assessment can allow investigators to gain a quick insight into the effects of the suspicious activity and prioritize actions to respond to the incident.

PHASE 5: Report Generation

The last stage is that the system will produce a formal forensic report depending on the results obtained in the course of investigation. The report contains identified anomalies, levels of severity, pointers of compromise, as well as the overall risk amount. This last product gives investigators a clear overview of the results of the analysis, supporting documentation, decision-making, and incident response.

In these sequential stages, Obsidian Circuit offers a centralized and simplified approach in carrying out digital forensic investigations. The system will enable the company to optimize its analysis speed, lessen the amount of manual work, and improve the overall correctness of incident response.

V. SYSTEM WORKFLOW

1. Artifact Upload – The investigator uploads a file, PCAP, or log file into the system.
2. Validation & Routing – The system validates the artifact and routes it to the correct analysis module.
3. Forensic Analysis – The selected module performs file, network, or log analysis to detect suspicious activity.
4. Risk Assessment – The findings are evaluated and an overall risk score is assigned.
5. Report Generation – A forensic report is generated with the detected anomalies and findings.
6. Investigation Completion – The investigator reviews the report for further action.

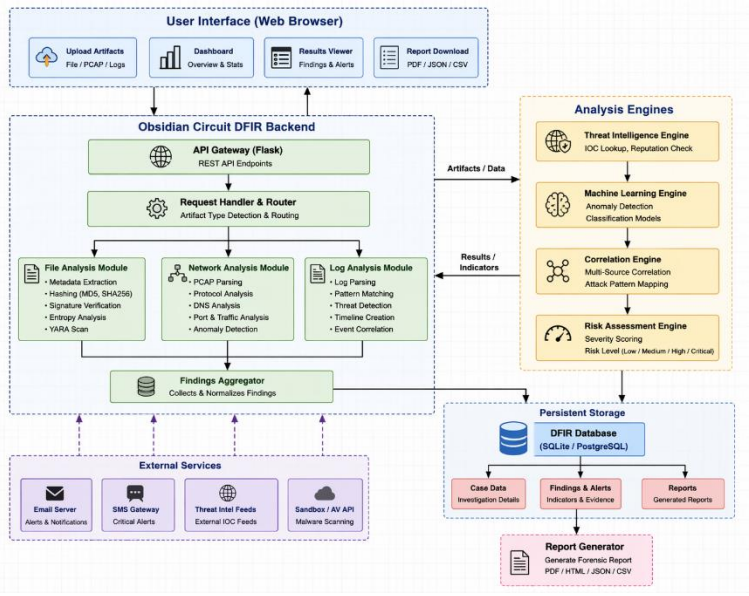

Figure 0: represents the overall system Workflow.

VI.TECHNICAL REQUIREMENTS

Hardware Components

1) Minimum 4 GB RAM (8 GB recommended for large PCAP/log processing)
2) Multi-core CPU (Intel i5 or equivalent and above)
3) At least 500 MB free disk space for temporary file handling and report generation
4) Stable internet connection (only required for optional VirusTotal API integration)

Software Components

1) Operating System: Windows / Linux / macOS
2) Programming Language: Python (v3.8 or above)
3) Runtime Environment: Virtual environment (venv/conda recommended)
4) Web Framework: Streamlit (v1.32.0+)

Python Libraries:

1) scapy– network packet analysis
2) pandas– data processing
3) plotly– interactive visualization
4) matplotlib– static report graphs
5) fpdf2– PDF report generation
6) python-magic– file type detection
7) hashlib– cryptographic hashing
8) requests– API communication
9) dotenv– environment variable management
10) chardet, colorama– encoding and CLI utilities

Functional Requirements:

1) Upload and process multiple forensic artifacts:
Files (binaries, documents), Network captures (.pcap), Log files (.log, .csv)
2) Perform file analysis:
MIME type validation, Hash generation (MD5, SHA1, SHA256), Entropy calculation, String extraction (IP, URL detection)
3) Perform network analysis:

Packet parsing and protocol extraction, Port scan detection, DNS tunneling detection, Data exfiltration identification

4) Perform log analysis:

Brute-force attack detection, Suspicious user-agent detection, Sensitive path access detection, Windows/Linux log parsing

5) Generate reports:

HTML and PDF formats, Executive summary auto-generation, Severity-based classification (Critical, Warning)

Non-Functional Requirements:

1) Performance: Efficient in-memory processing of large files, Fast execution using optimized libraries.
2) Scalability: Modular architecture supporting additional analysis modules
3) Security: Offline processing to protect sensitive forensic data, Secure API key handling via environment variables
4) Usability: GUI-based interface using Streamlit, Interactive visualizations for analysis.
5) Reliability: Error handling for corrupted or unsupported files, Consistent report generation

System Requirements:

No external database required (uses in-memory session state), Supports offline-first execution, Optional integration with VirusTotal API for threat intelligence, Cross-platform compatibility

Integration Requirements:

Integration with external threat intelligence APIs (VirusTotal), Compatibility with standard forensic file formats (.pcap, .log, .csv), Ability to export reports in universally readable formats (PDF/HTML)

Network Packet Dissection and Heuristics: Network analysis no longer looks at raw bytes, but instead behavioral tracking with scapy.

Flow Tracking: The module operates with dictionaries that point to the source IP to the destination port. It uses a bitwise mask of TCP flags (e.g. flags & 0x02 SYN), to determine the baseline traffic distribution.

Threshold-Based Anomaly Detection: Although behavioral heuristics are applied to indicate reconnaissance and exfiltration. By way of example, a single IP contacting 15 or more other people but unique ports invoke a Port Scan warning, whereas an internal IP sending more than 500KB of payload data to random external IPs causes a Data Exfiltration alert.

DNS Tunneling Detection: The system implies whether malicious code is being concealed in Base64 encoded UDP lookups by judging the sheer length of requested domain names in the DNSQR.qname attributes.

User Interface and Design:

Design Idea: To facilitate analysts to navigate complex forensic data, it is proposed that the interface should be very user-friendly, easy to navigate, and without aspects that make the interface difficult to navigate.

Methods Used:

Quickly accessing modules by side navigation.

Easy visual digestible sections (i.e., suspicious activities in red).

Table based data summarization with drop down lists to give a detail view.

Security and Integrity:

Purpose: Preventing evidence that could be easily manipulated, and verifiability.

Methods Used: Evidence files read-only analysis mode. Cryptography hash to verify that the data was not altered. Access controls and permission controls.

Testing and Validation

Concept: Provides the reliability, accuracy and efficiency of the tool on a structured testing methodology.

Methods Used:

Unit Testing: Verified independent blocks such as hash generation and log parsers.

Integration Testing: Guaranteed the interoperability of file, network, and log analysis elements.

Validation Testing: Comparison of tool output to known datasets and accuracy checked.

Usability Testing: Checked that the UI is user friendly to DFIR professionals.

Deployment

a) Upon successful validation and testing, an application called Obsidian Circuit can be deployed to the environment in cybersecurity labs, academic institutions, small enterprise environments and incident response teams with simple system requirements such as a computer with adequate processing servers, storage capacity and the required software dependencies installed. The deployment environment consists of the Streamlit application interface, Python-based forensic libraries, and optional internet connectivity to be integrated with external threat intelligence services.

b) Once the investigator uploads a suspicious file, PCAP file, or log file, the system automatically conducts file analysis, network analysis, and log analysis, and delivers immediate results out of the dashboard interface. The identified anomalies, suspicious indicators, and calculated risk score is presented in a structured format and detailed forensic report is generated in PDF or HTML format. As the system is designed as a modular system, it can be easily extended with further forensic analysis capabilities and is therefore useful when improving the speed, accuracy and efficiency with which Digital Forensics and Incident Response (DFIR) operations are carried out.

System Architecture and Implementation

The architecture of the Obsidian Circuit system is an architectural design intended to be a modular system which effectively integrates various forensic analysis functions into one framework. The architecture is composed out of four large layers the User Interface Layer, Analysis Layer, Risk Assessment Layer, and Reporting Layer.

At the User Interface Layer, the investigator engages with the system via a graphical dashboard based on Streamlit where suspicious artifacts like files, PCAP captures, and log files are uploaded and explored. The layer will be the main point of interaction where the user can choose the kind of forensic analysis to be applied and view the results in an efficient order.

The input artifact is then sent to the Analysis Layer which consists of three independent modules: File Analysis Module, Network Analysis Module, and Log Analysis Module. In the File Analysis Module, file metadata will be extracted, cryptographic hashes for MD5, SHA1, SHA256 will be calculated, entropy calculation will be performed, and suspicious strings such as IP addresses or URLs will be detected to find tampered files or those that are considered malicious. Within the Network Analysis Module, PCAPs will be processed using packet parsing, and suspicious anomalies such as port scanning or potential data exfiltration, as well as suspicious DNS requests, will be detected. The Log Analysis Module is used to process uploaded logs in order to reveal IoCs that include brute force attacks, suspicious user agents, unauthorized access attempts, and abnormal activity of any sort.

The results produced in the previous layers will then be forwarded to the Risk Assessment Layer where the detected suspicious findings will be categorized according to their severity levels.

Finally, the results are passed to the Reporting Layer, where a structured forensic investigation report is generated in PDF or HTML format. The report includes the analysis summary, detected anomalies, risk classification, and recommendations, providing a comprehensive record for forensic documentation and incident response.

After being processed, all of the information will be provided to the Reporting Layer where the forensic investigation report will be produced in PDF or HTML format and will contain information such as the summary, detected abnormalities, risk classification, and some suggestions on what can be done next. Incorporating this architecture, one can have a more effective DFIR investigation process, since it is more modular and scalable and will reduce the level of complexity in handling various DFIR tools.

Implementation:

Th Obsidian Circuit is implemented using Python programming language. To build the interactive web user interface, Streamlit library is employed. In general, the system is designed in a modular way where all forensic analysis processes are implemented as separate processing units.

The process begins when the investigator uploads a suspicious artifact through the Streamlit dashboard. The uploaded artifact is first validated, and the system identifies whether the input is a standard file, a PCAP file, or a log file. Based on the detected type, the system routes the artifact to the corresponding forensic analysis module.

In the File Analysis Module, the system extracts metadata such as file name, file size, timestamps, and file type. It then generates MD5, SHA1, and SHA256 hashes using Python's Hashlib library to verify file integrity. It, too, conducts an entropy calculation and string extraction to identify suspicious indicators such as hidden executable, embedded URL, or any other pattern.

Scapy is employed in the Network Analysis Module where it is used to deprocess uploaded PCAP files. The implementation focuses on examination of packet headers, protocols, ports, and traffic patterns to detect suspicious network activities like repeated connection attempts, abnormal DNS activity and possible explicit data transfer anomalies. The results are further abstracted and graphically represented with Plotly and Matplotlib to understand them better.

The Log Analysis Module will read the uploaded log files and process them through regular expressions and rule based detection systems. The implementation scans the logs of any suspicious activities like repeated failed logins, access into sensitive resources, abnormal request patterns as well as access attempts that are unauthorized. These occurrences are grouped according to severity to aid in quick investigation of the incidents.

Once the analysis is done, the results of all the modules are transmitted to the Risk Assessment Engine where suspicious findings are matched and labelled with warnings such as Critical, Warning, or Informational. The cumulative risk score is then calculated to figure out the level of threat posed by the uploaded artifact.

At last, an orderly forensic report is automatically generated using FPDF2 library as PDF document with all collected data, suspicious evidence, an indicator of compromise and results of the analysis.

VII. RESULTS

Figure 1. The main dashboard interface of Obsidian Circuit, displaying the three integrated forensic modules: File Analysis, Network Analysis, and Log Analysis. The dashboard provides investigators with a centralized interface to navigate between modules, review available analysis capabilities, and initiate forensic investigations carefully.

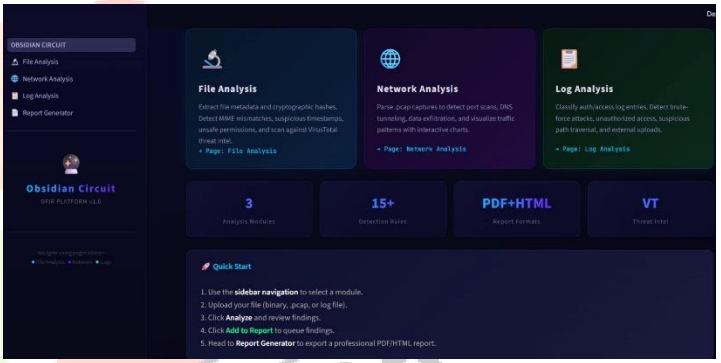

Figure 2. Dashboard overview page for Obsidian Circuit, which showcases the platform's DFIR capabilities by specifying its number of analysis modules, report formats, threat intelligence, and other DFIR capabilities.

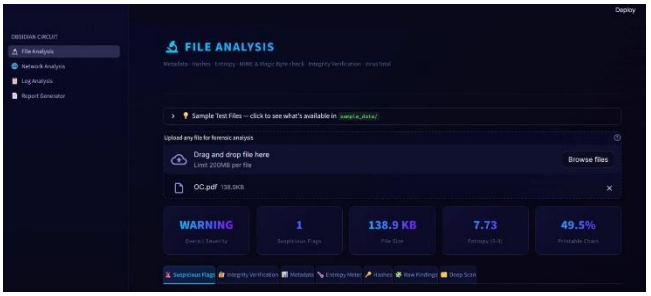

Figure 3. Dashboard overview page for Obsidian Circuit, which showcases the platform's DFIR capabilities by specifying its number of analysis modules, number of detection rules, report formats, threat intelligence, and other DFIR capabilities.

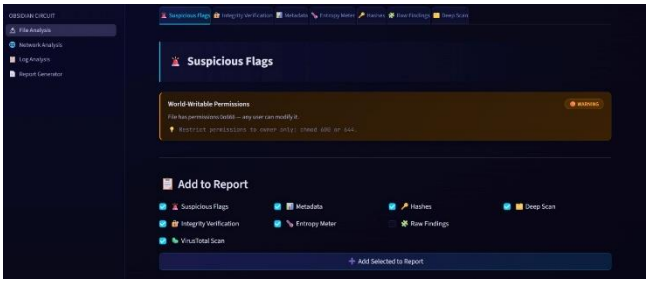

Figure 4. The Suspicious Flags and reporting dashboard. The system automatically finds a security risk (World-Writable Permissions 0o666) and allows investigators to compute results from VirusTotal and Deep Scan modules into a final forensic report.

Figure 5. The File Integrity Verification module. This page shows the platform's ability to detect tampering by checking the file's MD5 hash against a known-good baseline, confirming the document's authenticity.

Figure 6. In Obsidian Circuit the File Metadata view which shows the core file attributes of OC.pdf. The interface reports that there is a high Shannon Entropy (7.73) and a 49.5 percent printable ratio, which shows possible obfuscation or encryption of data.

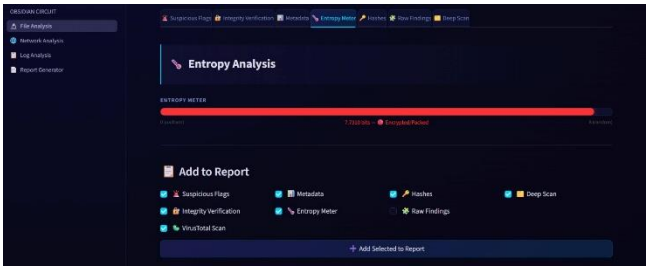

Figure 7. The Entropy Analysis module of Obsidian Circuit, with a visual representation of the randomness of data.

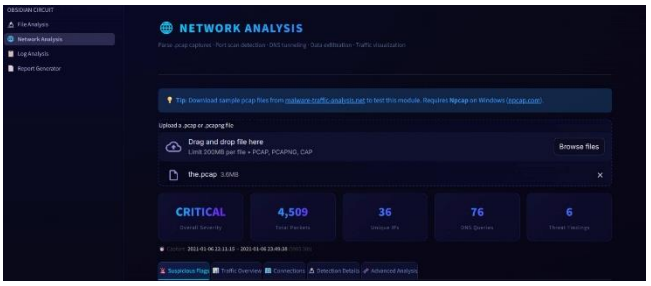

Figure 8. Network Analysis dashboard in the Obsidian Circuit with emphasis on the incident response to traffic captures. It triggers in-depth analysis of DNS packets and distinct IP connections in order to identify data exfiltration or tunneling.

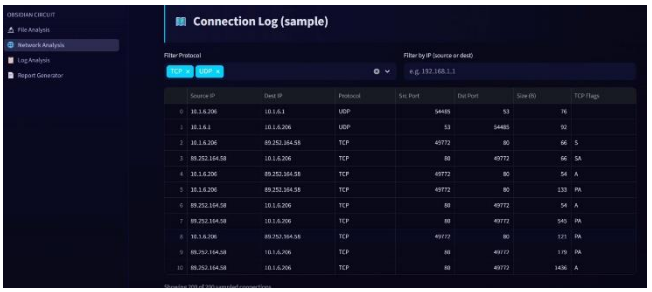

Figure 9. The Connection Log view of Obsidian Circuit, which gives a detailed account of the sampled network traffic. This table allows investigators to filter based on protocol (TCP/UDP) and IP address whereby investigators can find key information on communication patterns (source/destination ports, size of payload etc. as well as TCP flags).

Figure 10. The module of the Log Analysis that shows that sample_apache.log has been ingested. The platform finds 22 threat findings and 15 failed logins, enabling the detection of brute force and analysis of suspicious paths. A Critical overall severity status gives high level of warning to security personnel.

Figure 11. The Log Analysis Suspicious Flags module, which represents automatic detection of threats. The site detects Critical severity incidents wherein a particular IP address (10.0.0.55) has tried to enter sensitive directories such as /admin and /admin/config. The system offers real time remediation guidance which includes IP blocking and access control audit suggestions.

Figure 12. Advanced Analysis dashboard with the visual telemetry of the network traffic. The interface presents a TCP Flag Distribution histogram, and IP Type Classification donut chart (with 91.7% External/Public traffic), which enables you to quickly identify an unusual scanning behavior or external connection without authorization.

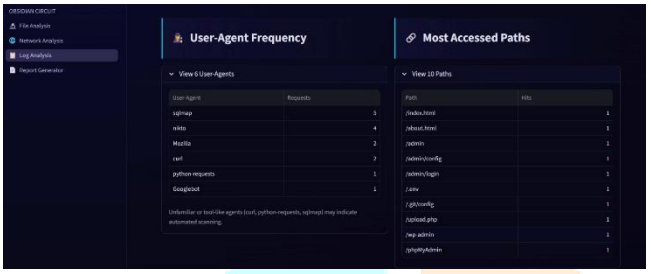

Figure 13. The dashboard of User-Agents Frequency and Most Accessed Paths. This sorts out log data so as to tell automated scanning tools. Frequencies of such agents as sqlmap and nikto are high enough, and hits on sensitive paths like /.env and /phpMyAdmin suggest the definite signs of reconnaissance and vulnerability research.

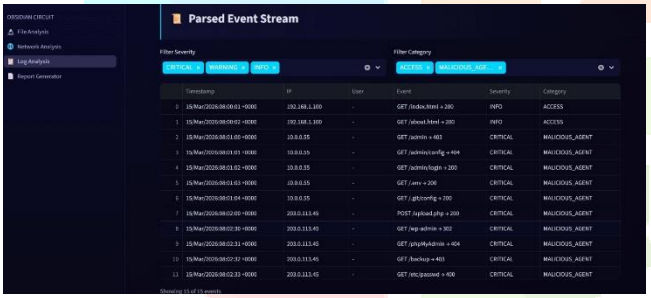

Figure 14. The Parsed Event Stream table which turns raw logs into a forensic structured timeline. The entries are tagged with Severity (e.g., Critical) and Category (e.g., Malicious Agent), which enables a timestamps- and IP-address-based filter and isolation of high-risk interactions by investigators.

Figure 15. Raw Log Lines viewer of the selected events. This interface presents the underlying technical evidence of the data which has been parsed showing the complete HTTP request strings, including status codes (e.g., 403, 404), sizes in bytes and specific tool-based user agents, which are necessary to manually verify and deep-dive-analyze the data.

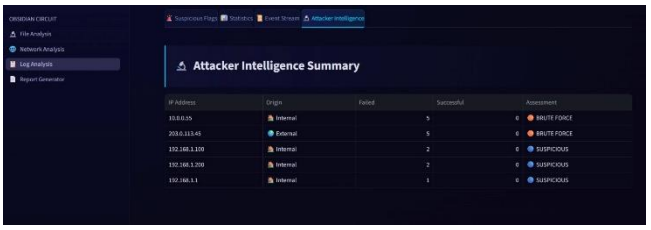

Figure 16. The Attacker Intelligence Summary, a compilation of behavioral information into high-level test. The platform matches the number of failed requests to origin data, and automatically categorizes IP addresses. Both internal and external IPs have their entries determined as either Brute Force or Suspicious which allows incident response teams to quickly prioritize their efforts.

Figure 17. The summary page on the Report Generator, demonstrates the multi-modal integration of the platform. It is an interface tracking Queued Analysis Modules in order to reflect the current status (Warning or Critical) in the domains of File, Network and Log Analysis.

Figure 18. The interface of a Report Generator, which involves the input of the case details into the interface by the analyst (name and ID). This view gives an auto-generated, editable Executive Summary that summarizes the total number of critical and warning indicators across the investigation of their analysis modules associated with File, Network and Log analysis modules.

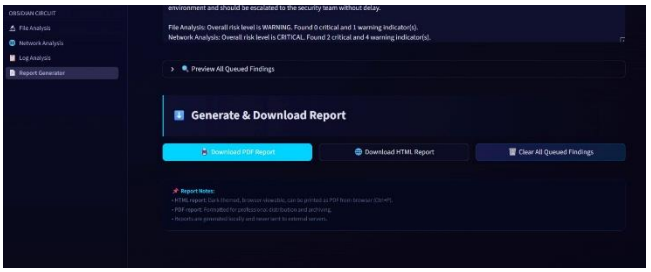

Figure 19. Report Export module, enables an investigator to complete his or her findings. Most export formats, including a formatted PDF that can be saved in a professional archives and a dark colored HTML report that may be viewed on a browser computer. This step assures forensic data is made available in a transportable and readable/understandable by a human form.

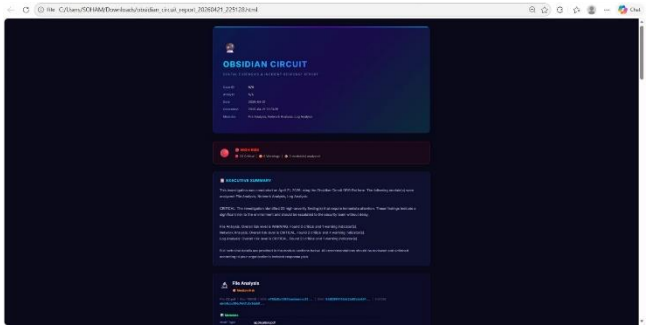

Figure 20. An example of the Obsidian Circuit Final Report that will be displayed in a web browser. The output has a well-organized hierarchy starting with the High-Risk alert banner and summary of the 23 critical findings contained in the output. It gives a drain-down into the detailed modules, which include File Analysis, where cryptographic hash and metadata are available to assure complete chain of custody of the forensic evidence.

VIII. CONCLUSION

Obsidian Circuit is a single and effective platform for Digital Forensics/Incident Response (DFIR) by unifying file, network and log analytics all in one modular solution. The solution makes the forensic investigator's workflow easier by automating many of the tasks typically associated with digital forensic investigations, such as metadata extraction, hash verification, anomaly detection, suspicious log identification, and report generation, which significantly reduces the effort associated with conducting incident analysis.

Using the Obsidian Circuit solution has shown that providing a combined view of multiple forensic capabilities via the same dashboard has proven to speed up investigations, provide greater visibility into suspicious behavior and aid investigators with making more rapid decisions when responding to incidents. The features of the Obsidian Circuit solution, including risk assessments, threat detection and the generation of structured reports, will provide useful support for both the academic and small business industries with respect to cybersecurity training and responding to incidents.

Overall, the Obsidian Circuit solution will serve as lightweight yet highly effective DFIR solutions for improving forensic accuracy and operational effectiveness. Its modular design will also allow for future enhancements to be added specifically to meet an organization’s needs in a scalable manner, thus allowing organizations to implement a strong digital forensic solution and use it as the foundation to develop higher quality computer forensic investigation systems.

IX. ACKNOLEDGEMENT

We would like to Convey our heartfelt Thanks to our Pro Vice Chancellor Dr. Ramchandra Pujeri, Dean Dr. Ganesh Pathak, Director Dr. Vipul Dalal and our project guide Prof. Uday Mande. Their experience and understanding of artifacts beyond the envision of mere humans, were invaluable assets in ensuring that The Obsidian Circuit was developed into an implementable and powerful digital forensic instrument.

Moreover, also our heartfelt Thankyou to our teammates who have continued to provide us with their support and commitment towards this project.

And finally, we are happy to give credit to the facilities accorded by our institution, and with their constant encouragement and support this project could be made possible.

X. REFERENCES

[1] B. Carrier, File System Forensic Analysis. Boston, MA, USA: Addison-Wesley, 2005.

[2] E. Casey, Digital Evidence and Computer Crime: Forensic Science, Computers and the Internet, 3rd ed. Cambridge, MA, USA: Academic Press, 2011.

[3] K. Kent, S. Chevalier, T. Grance, and H. Dang, “Guide to Integrating Forensic Techniques into Incident Response,” National Institute of Standards and Technology Special Publication 800-86, 2006.

[4] C. Altheide and H. Carvey, Digital Forensics with Open Source Tools. Waltham, MA, USA: Syngress, 2011.

[5] O. Santos, “Network Traffic Analysis Using Scapy,” International Journal of Computer Applications, vol. 180, no. 40, pp. 20–26, 2018.

[6] VirusTotal, “VirusTotal API Documentation,” 2024.

[7] Streamlit, “Streamlit Documentation,” 2024.

[8] W. Stallings, Network Security Essentials: Applications and Standards, 6th ed. Boston, MA, USA: Pearson, 2017.

[9] R. Bejtlich, The Practice of Network Security Monitoring: Understanding Incident Detection and Response. San Francisco, CA, USA: No Starch Press, 2013.

[10] H. Carvey, Windows Forensic Analysis Toolkit, 4th ed. Waltham, MA, USA: Syngress, 2014.

[11] M. Sikorski and A. Honig, Practical Malware Analysis. San Francisco, CA, USA: No Starch Press, 2012.