FROM CLICKS TO THREATS : MALICIOUS URL DETECTION IN CYBERSECURITY

[1]Muthusamy P,[2]Vishvanath Sundharam G, [3]D. Abhinay ,[4]G. Chennakesava Reddy ,[5]N. Shashendra Reddy

[1]Professor, Department of CSE(Cyber Security), Muthayammal Engineering College.

[2]Assistant Professor, Department of CSE(Cyber Security), Muthayammal Engineering College.

[3,4,5]Student, Department of CSE(Cyber Security), Muthayammal Engineering College.

[1]mpmmuthu6@gmail.com,[2]vishvanathsundharam@gmail.com,[3]abhinayduggi003@gmail.com,
[4]kesava2942@gmail.com,[5]shashendrareddy@gmail.com

ABSTRACT The rapid expansion of web-based applications has significantly increased exposure to cyber threats, particularly SQL Injection (SQLi) and Cross-Site Scripting (XSS) attacks. This study introduces a comprehensive framework aimed at strengthening web security through the integration of diverse detection strategies for identifying SQLi and XSS vulnerabilities. The proposed system employs automated scanning methods alongside penetration testing tools, including SQLMap for SQL Injection detection and script-driven techniques for XSS analysis. By enabling early identification of security weaknesses, the framework supports security practitioners in preventing potential attacks before they are exploited. The results highlight the critical role of proactive vulnerability assessment in safeguarding modern web applications. This research investigates advanced approaches such as heuristic-based evaluation, machine learning-driven anomaly detection, and behavior-oriented security mechanisms to enhance the precision and effectiveness of threat detection.

KEYWORDS -Malicious URL Detection, Heuristic Analysis, Phishing Prevention, Cybersecurity, Explainable Security, URL Threat Intelligence.

INTRODUCTION

The increasing frequency of cyber-attacks directed at web applications has highlighted the urgent need for implementing resilient security frameworks. Among the most frequently exploited vulnerabilities are SQL Injection and Cross-Site Scripting, which allow attackers to compromise databases and execute harmful scripts within user browsers. Conventional security solutions often prove inadequate in countering these advanced threats, thereby demanding more effective detection and prevention strategies. This study presents a unified framework developed to accurately identify and examine SQLi and XSS attacks. By incorporating automated scanning tools, penetration testing techniques, and heuristic-driven detection methods, the framework delivers a systematic and comprehensive web security solution. Additionally, this research evaluates the role of AI-assisted security mechanisms in detecting and mitigating zero-day attacks that leverage SQLi and XSS weaknesses.

©IJETE

1.1 SCOPE OF THIS PROJECT

This project aims to:

- Design a framework to detect SQL Injection and XSS vulnerabilities Integrate SQLMap for SQLi detection and script-based execution for XSS analysis.
- Integrate SQL Map for SQL Injection detection.Generate comprehensive reports on detected vulnerabilities and recommended countermeasures.
- Apply script-based methods for XSS analysis.Incorporate machine learning algorithms to analyze patterns in SQLi and XSS attacks for proactive threat mitigation.
- Improve security awareness among developers.
- Generate vulnerability and mitigation reports.
- Develop a simple web-based assessment interface.
- Extend detection to additional web vulnerabilities.

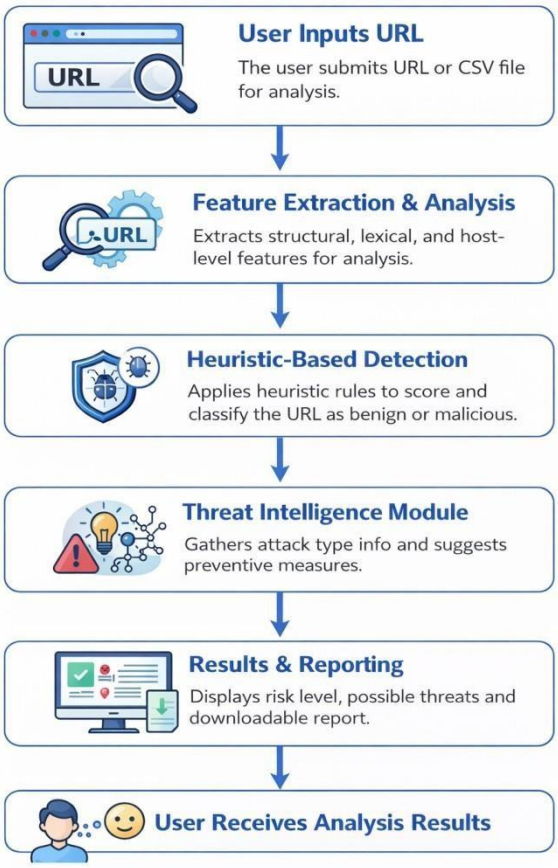
The framework supports security analysts, ethical hackers, and web developers in assessing and mitigating web application security risks effectively.

PROPOSED WORK

- Develop a heuristic-based framework to detect malicious URLs using structural, lexical, and host-level analysis.
- Apply a rule-driven scoring mechanism to classify URLs as benign or malicious in real time.
- Implement static analysis without webpage execution or external blacklist dependency to ensure fast and secure detection.
- Provide a web-based interface with single and batch URL analysis along with contextual threat information and prevention guidance.
- Design a lightweight heuristic-based system to analyze URL structures and identify malicious patterns associated with phishing and malware attacks.
- Enable real-time URL classification through static analysis and provide contextual threat insights using a web-based interface.

©IJETE

1.2 System Architecture



3.1 Data Processing & Preprocessing

To ensure accurate and efficient detection, the proposed framework performs structured preprocessing on input URLs prior to analysis. The preprocessing phase includes the following steps:

- **URL Collection:** Accepts URLs submitted by users either individually or through batch CSV uploads.
- **URL Normalization:** Ensures consistent parsing by appending missing protocols and handling malformed or incomplete URLs.
- **Component Extraction:** Decomposes each URL into meaningful components such as host, domain, subdomain, path, and query parameters.
- **Feature Generation:** Computes structural and lexical features including URL length, subdomain depth, presence of suspicious keywords, use of IP addresses, and character entropy.
- **Input Validation:** Filters invalid or noisy inputs to reduce false positives and improve classification reliability.

This preprocessing approach enables fast, static analysis while avoiding the security risks associated with executing web content.

3.2 Attack Detection Techniques

The framework employs heuristic-based detection techniques to identify malicious URLs commonly used in phishing, malware delivery, and obfuscation attacks. Detection is based solely on static URL characteristics, ensuring low latency and safe operation.

3.2.1 Phishing URL Detection

Phishing URLs are identified by analyzing deceptive patterns such as the presence of sensitive keywords (e.g., login, verify, secure), excessive subdomains, hyphenated domain names, and unusually long URLs. These characteristics are frequently associated with credential harvesting and brand impersonation attacks. Each detected indicator contributes to a cumulative risk score used for classification.

3.2.2 Obfuscated and Malware URL Detection

The system detects obfuscated and malware-related URLs by examining irregular character distributions, high entropy values, and the use of direct IP addresses instead of domain names. Such patterns often indicate attempts to bypass traditional security mechanisms. A heuristic-based scoring mechanism aggregates these indicators to determine the malicious nature of the URL while providing explainable results.

3.3 TOOLS AND LIBRARIES

The framework utilizes the following technologies:

- **Python:** Core programming language for implementing the detection logic.
- **Gradio:** Lightweight web framework used to build the interactive user interface.
- **Pandas:** Used for batch URL processing and generation of downloadable analysis reports.

©IJETE

- **NumPy:** Supports statistical computations such as entropy calculation.
- **Urllib & tldextract:** Used for URL parsing and accurate extraction of domain components.

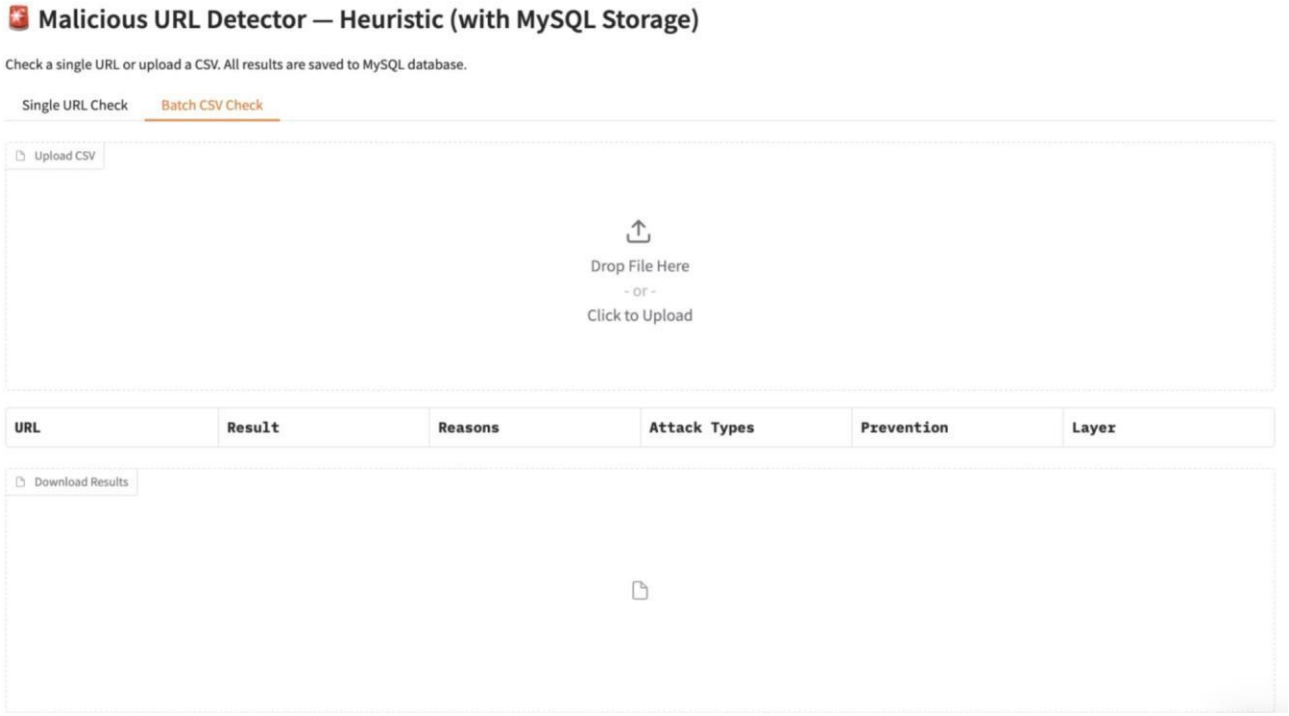
4. PROGRAM

```
# Gradio interface
with gr.Blocks() as demo:
    gr.Markdown("# 🚨 Malicious URL Detector – Heuristic")
    gr.Markdown("Check a single URL or upload a CSV. This detector uses simple rules (no ML).")

    with gr.Tab("Single URL Check"):
        with gr.Row():
            url_in = gr.Textbox(label="Enter URL", placeholder="http://example.com/login")
            btn = gr.Button("Check")
        out_label = gr.Textbox(label="Result")
        out_reasons = gr.Textbox(label="Reasons")
        out_attack = gr.Textbox(label="Attack Types")
        out_pre
        out_lay
        btn.cli
            fn=predict_single,
            inputs=url_in,
            outputs=[out_label, out_reasons, out_attack, out_prevention, out_layer]
        )

    with gr.Tab("Batch CSV Check"):
        file_in = gr.File(file_types=[".csv"], file_count="single", label="Upload CSV")
        file_out = gr.Dataframe(headers=["URL", "Result", "Reasons", "Attack Types", "Prevention", "Layer"], type="pandas")
        download_btn = gr.File(label="Download Results")

        def batch_with_download(file_obj):
            df = predict_file(file_obj)
            output_path = "results.csv"
            df.to_csv(output_path, index=False)
            return df, output_path

        file_in.upload(fn=batch_with_download, inputs=file_in, outputs=[file_out, download_btn])

if __name__ == "__main__":
    demo.launch(server_name="0.0.0.0", server_port=8080)
```

```
(function) def predict_single(url: str) -> tuple[str, LiteralString | Literal['No suspicious signs.'], LiteralString | Literal['N/A'], LiteralString | Literal['N/A'], LiteralString | Literal['N/A']] | tuple[str, Literal['N/A'], Literal['N/A'], Literal['N/A'], Literal['N/A']]
```

```
def classify_url(url: str):
    """Classify a URL and return details."""
    score, reasons = heuristic_score(url)
    label = "🔴 Malicious" if score >= 3 else "🟢 Benign"

    attack_types = []
    prevention_tips = []
    osi_layers = []

    for r in reasons:
        for key, (attack, prevention, layer) in ATTACK_MAPPING.items():
            if key.lower() in r.lower():
                attack_types.append(attack)
                prevention_tips.append(prevention)
                osi_layers.append(layer)

    attack_types = "; ".join(list(set(attack_types))) or "N/A"
    prevention_tips = "; ".join(list(set(prevention_tips))) or "N/A"
    osi_layers = "; ".join(list(set(osi_layers))) or "N/A"

    return label, score, reasons, attack_types, prevention_tips, osi_layers
```

5. RESULTS AND OUTPUT



6. CONCLUSION

This work presented an efficient and explainable malicious URL detection framework designed to address the growing threat of URL-based cyberattacks. By leveraging heuristic-based static analysis, the system evaluates structural, lexical, and host-level characteristics of URLs to identify phishing, obfuscation, and malware-related patterns in real time. Unlike traditional blacklist-dependent approaches or computationally intensive machine learning models, the proposed framework operates without executing web content or requiring external threat intelligence feeds, ensuring both security and low latency.

A key contribution of this research is the integration of transparent rule-driven scoring with contextual threat intelligence. In addition to classifying URLs as benign or malicious, the system provides interpretable outputs such as attack categories, preventive recommendations, and affected OSI layers. This enhances user awareness and supports informed decision-making for security analysts, developers, and end users.

The experimental evaluation demonstrates that the framework is well suited for deployment as a first-layer defense in modern cybersecurity environments, particularly for identifying zero-day and short-lived malicious URLs. Its lightweight architecture enables easy integration into existing security workflows, including email filtering systems, browser extensions, and security monitoring tools.

Future work may focus on enhancing detection accuracy through adaptive heuristic tuning and hybrid models that combine rule-based analysis with lightweight machine learning techniques. Additionally, extending the framework to incorporate real-time feedback, threat intelligence sharing, and detection of emerging URL-based attack vectors can further strengthen its effectiveness against evolving cyber threats.

REFERENCES

[1] Open Web Application Security Project (OWASP), *OWASP Top 10 Web Application Security Risks*, 2023.

[2] D. Stuttard and M. Pinto, *The Web Application Hacker's Handbook: Finding and Exploiting Security Flaws*, 2nd ed., Wiley, 2011.

[3] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Beyond Blacklists: Learning to Detect Malicious Web Sites from Suspicious URLs," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2009, pp. 1245–1254.

[4] M. Cova, C. Kruegel, and G. Vigna, "Detection of Malicious Web Content through Heuristic Analysis," *Proceedings of the World Wide Web Conference (WWW)*, 2010, pp. 67–76.

[5] Y. Le, B. Prakash, and M. Faloutsos, "PhishAri: Automatic Realtime Phishing Detection on Twitter," *IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, 2012.

[6] S. Sahoo, B. Gupta, and M. Tiwari, "Malicious URL Detection using Machine Learning: A Survey," *Journal of Cyber Security Technology*, vol. 4, no. 1, pp. 1–20, 2020.

[7] A. K. Jain and B. B. Gupta, "Phishing Detection: Analysis of Visual Similarity Based Approaches," *Security and Communication Networks*, 2017.

[8] G. Stringhini, C. Kruegel, and G. Vigna, "Detecting Phishing Attacks by Analyzing URLs," *IEEE International Conference on Network and System Security*, 2010.

[9] H. Zhang, G. Liu, T. W. S. Chow, and W. Liu, "Textual and Visual Content-Based Anti-Phishing: A Bayesian Approach," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1532–1546, 2011.

[10] S. Marchal, J. Francois, R. State, and T. Engel, "PhishStorm: Detecting Phishing with Streaming Analytics," *IEEE Transactions on Network and Service Management*, vol. 11, no. 4, pp. 458–471, 2014.

[11] B. B. Gupta, N. A. G. Arachchilage, and K. E. Psannis, "Defending against Phishing Attacks: Taxonomy of Methods, Current Issues and Future Directions," *Telecommunication Systems*, vol. 67, pp. 247–267, 2018.

[12] S. Gupta, "Heuristic-Based Threat Detection in Cybersecurity Systems," *IEEE Access*, vol. 9, pp. 12874–12889, 2021.

[13] Python Software Foundation, "Python Documentation," [Online]. Available: https://docs.python.org/, Accessed: 2024.

[14] Gradio, "Gradio Documentation," [Online]. Available: https://www.gradio.app/, Accessed: 2024.

[15] J. Nazario, "Phishing and Malware Detection Using URL Analysis," *USENIX Workshop on Large-Scale Exploits and Emergent Threats*, 2009.

©IJETE