International Journal of Scientific Research in Science and Technology
Print ISSN: 2395-6011 | Online ISSN: 2395-602X (www.ijsrst.com)
doi : https://doi.org/10.32628/IJSRST

Enhanced Security Framework for Web Log Mining to Safeguard Organizational Data and Predict Consumer Behaviour

Sonia Sharma
Department of Computer Science and Applications, Hindu Girls College, Jagadhri, India

ABSTRACT

Web log mining is a critical process for extracting meaningful patterns from web usage data, enabling organizations to enhance user experience and predict consumer behaviour. However, ensuring the security of web logs is paramount to protect sensitive information from malicious access and cyber threats. This paper proposes a comprehensive security algorithm designed to filter malicious web logs at the entry point, ensuring that only legitimate data is processed. The algorithm integrates three key phases: initial security check, data pre-processing, and session identification. After rigorous application of these algorithms, we achieved enhanced data integrity and identified patterns that contribute to accurate consumer behaviour predictions. This framework not only fortifies web log data but also enhances organizational trust and consumer satisfaction.

Keywords : Web Usage Data, User Experience, Consumer Behavior Prediction, Web Log Security, Sensitive Information Protection, Cyber Threats

1. Introduction

The exponential growth of the internet and web-based applications has led to the generation of vast amounts of data daily. This data, captured through web logs, provides invaluable insights into user behavior, preferences, and interaction patterns. Web log mining, therefore, has emerged as a pivotal technique for analyzing this data to extract meaningful patterns that inform decision-making, marketing strategies, and user experience improvements. For internet-based organizations, the ability to accurately predict consumer behavior through web log analysis is a significant competitive advantage [5][6][7].However, with the increasing reliance on web log mining comes the challenge of ensuring data security. Web logs are often targeted by malicious entities seeking to exploit vulnerabilities, compromise sensitive information, and disrupt organizational processes. [8][9][10][11]Cyber-attacks such as SQL injections, Distributed Denial of Service (DDoS), and data breaches can severely impact an organization's operations and erode consumer trust. Thus, implementing robust security mechanisms during web log mining is not only a necessity but also a strategic imperative for safeguarding organizational data and ensuring continuous operational excellence.

This paper introduces an enhanced security framework for web log mining, addressing critical vulnerabilities through a series of algorithmic interventions. The proposed solution encompasses three core phases: security checks at the data entry point, rigorous data pre-processing to eliminate erroneous and anonymous entries, and session identification to streamline the analysis process. By filtering out malicious logs and encrypting sensitive data, this framework fortifies the integrity of web logs, ensuring that only accurate and relevant data is used for pattern discovery and predictive analytics.

The significance of this research lies in its holistic approach to web log security, seamlessly integrating data protection measures into the mining process. By applying the proposed algorithms to real-world datasets, this paper demonstrates the effectiveness of the framework in identifying and mitigating potential threats, thereby enhancing the reliability of web log mining outcomes. Ultimately, the goal is to provide organizations with a secure and efficient methodology for leveraging web log data to predict consumer behaviour, drive business growth, and enhance customer satisfaction.

2. Background and Related Work

Security in web log mining has been an active research area, with studies emphasizing data integrity and privacy. Notable contributions include:

- *Zhou et al. (2020)* examined systematic vulnerabilities in web log mining and proposed intrusion detection systems to enhance security[1].
- *Li & Chen (2019)* explored the application of encryption techniques during web log mining to prevent data breaches[2].
- *Kumar et al. (2021)* developed an anomaly detection framework for identifying malicious web logs using machine learning algorithms[3].
- *Rahman et al. (2018)* introduced session-based analysis for filtering erroneous data and detecting unauthorized access patterns[4].

These studies underscore the significance of integrating security measures into web log mining processes to protect data and enhance consumer satisfaction.

3. Algorithm Design and Implementation:

3.1 Security at Entry Point

Security at the entry point is the first layer of defense against potential cyber threats. This phase ensures that web logs entering the system are verified for authenticity, preventing malicious data from compromising subsequent mining processes. The algorithm initializes variables to track network status and detect intrusions. By comparing incoming log data with predefined identification values, the algorithm classifies logs as either legitimate or malicious.

Logs identified as malicious are redirected to a separate array for further analysis or discarded. This step is crucial in mitigating unauthorized access attempts and blocking suspicious activity early in the data pipeline, preserving the integrity of organizational data.

Proposed Algorithm for Login Check

- Variables are initialized to track network status, malicious logs, and identification numbers.
- Data packets are compared against identification values to determine legitimacy.
- Malicious logs are separated into a dedicated array, while legitimate logs proceed to the next stage.
- If malicious logs are detected during the process, transmission is halted to prevent further breaches.

Steps are as follows:-

Section-I

Step 1: Activate and Initialize the following variables

```
NetST=1    ;          // set net status = 1 in start
Hacker=0   ;                  // Count number of malicious log
```

```
ID  =0 ;                          // Identification number for log
TN =N ;                    // N is the size of array which is entered by the user
Malicious_Array [TN];          // Array for malicious log
NonMalicious_Array [TN]; // Array for non malicious log
//The size of both array is equal to TN which indicates in any case if all the //logs are malicious then all logs can enter in to Malicious_Array and if all //the logs  are non malicious then all logs can enter in to //NonMalicious_Array.
```

Step 2: Setting ST of each Log //Status of log

(a) SI ; //Source log ID

(b) DI ; //Destination log ID

Step 3: Compare the data packet with the identification value.

```
Do          // It starts from Source log entry to exit  log
{
If (LogID >= SI && LogID <= DI)
{       LogST = .T.     ;
                    // Set log Status as 'TRUE'
} Else IF
LogST= .F. ;
                    // LogID does not match then this log is declared
                    //as malicious log by setting its status as 'FALSE'
 }
While (LogID==DI);     //Loop will finish as it exit point
```

Step 4: Initialize loop for entering Logs into Malicious and Non_Malicious Array

```
Set i= 1; i<=TN; i++;
{
if (LogST ==  '.F.')                  // If log status is 'FALSE'
Hacker++                              // Increase Hacker by one and
Malicious_Log[i] = Log;           // log enters into Malicious_Array
else if ( LogST== '.T.)'          // If log status is 'TRUE'
Non_MaliciousLog [i] = Log;
                                  // Log enters into NonMalicious_ Array
If (LogID==DI)
Break;
                    // It will break the loop when it reaches to exit point.
}
```

Proposed Algorithm Section -II

Step 1: Starts collecting data from Start to exit

Step 2: Initialize loop for detecting malicious logs.

// It verifies status of each log in the route from Start to exit

(a) Set i= SI; i<=DI; i++;

(b) If Log[i] = Malicious_Array[i].
Break;
// If it found a Log that belongs to Malicious Array it will stop the //transmission.

3.2 Data Preprocessing

Preprocessing is vital for refining collected data, ensuring that irrelevant or erroneous entries do not affect analytical results. This phase involves cleaning the raw data by filtering out records that indicate errors or originate from bots, crawlers, and automated scripts.

By eliminating non-relevant entries, the pre-processing algorithm enhances the quality of data fed into subsequent stages, thereby improving the reliability of insights derived from web log mining.

Steps for Data Cleaning

- The algorithm reads log records from the web server log file.
- Records with errors (e.g., status 301, 404, 500) and automated crawlers are filtered out.
- Valid log records are inserted into the log database for further processing.
- After applying this algorithm, 12,697 rows of clean data were obtained from the initial dataset of 19,283 logs.

3.3 Session Identification:

Session identification is essential for understanding user interaction patterns. By grouping web logs into sessions, organizations can gain insights into user behavior over specific time periods.

The algorithm identifies unique sessions by analyzing inactivity timeouts and unique identifiers. This phase is critical for mapping user journeys and detecting anomalies in access patterns.

Steps for Session Identification

- Log records are read from the cleaned log database.
- Sessions are identified based on criteria such as session inactivity exceeding 30 minutes or absence of referrer information.
- 6,656 sessions were successfully identified following this algorithm.

3.4 Pattern Discovery and Security Revalidation: Pattern discovery involves analyzing user sessions to extract recurring patterns, which serve as the foundation for predictive analytics. To ensure continued security, this phase incorporates encryption techniques that secure web logs even after preprocessing. By revalidating logs during pattern discovery, the algorithm adds an additional layer of protection, safeguarding sensitive data.

4. Results and Discussion: The application of the proposed algorithms yielded promising results

- Figure –I shows the weblog file after applying all steps of proposed algorithm(login check, cleaning, session identification). Figure-II shows the graphical representation of actual size of the object return to the client after applying proposed algorithm and Table –I shows the result in tabular form.
- From the initial 19,283 logs, 12,697 logs were retained post-cleaning, reflecting a substantial reduction in irrelevant data.
- Session identification highlighted 6,656 unique user sessions, indicating significant user engagement.

- The absence of malicious logs during all stages highlights the algorithm's robustness in filtering unauthorized data.

This outcome demonstrates that the proposed security framework effectively mitigates risks while preserving data integrity for accurate consumer behaviour analysis.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	IP Address	NA	C	Date & Time	E	URL	status code sent by	size of obejct retrun to	Referring URL	user agent	F	G	H	I		
2	103.201.138.247	-	-	[22/Feb/	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27284	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
3	103.201.138.247	-	-	[22/Feb/	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	7224	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
4	103.201.138.247	-	-	[22/Feb/	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	992	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
5	103.201.139.243	-	-	[05/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://www.socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
6	103.201.139.243	-	-	[05/Mar	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	3674	http://www.socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
7	103.201.139.243	-	-	[05/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://www.socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
8	95.27.135.223	-	-	[08/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
9	95.27.135.223	-	-	[08/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
10	95.27.135.223	-	-	[08/Mar	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	3674	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
11	103.201.138.3	-	-	[13/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
12	103.201.138.3	-	-	[13/Mar	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	3674	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
13	103.201.138.3	-	-	[13/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
14	128.204.13.233	-	-	[21/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://www.socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
15	128.204.13.233	-	-	[21/Mar	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	3674	http://www.socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
16	128.204.13.233	-	-	[21/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://www.socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	WOW64	Trident/6.0		
17	5.228.75.212	-	-	[23/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	Trident/6.	MASPJS)		
18	5.228.75.212	-	-	[23/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	Trident/6.	MASPJS)		
19	5.228.75.212	-	-	[23/Mar	+0000]	GET /wako/assets/css/animate.css HTTP/1.1	200	3674	http://socialstudio.in/wako/	Mozilla/5.0 (compatible	MSIE 10.0	Windows	Trident/6.	MASPJS)		
20	103.208.74.254	-	-	[27/Mar	+0000]	GET /wako/assets/css/owl.carousel.min.css HTTP/	200	984	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						
21	103.208.74.254	-	-	[27/Mar	+0000]	GET /wako/assets/css/bootstrap.min.css HTTP/1.:	200	27261	http://socialstudio.in/wako/	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/7						

Figure-I : Weblog File after applying Proposed algorithm

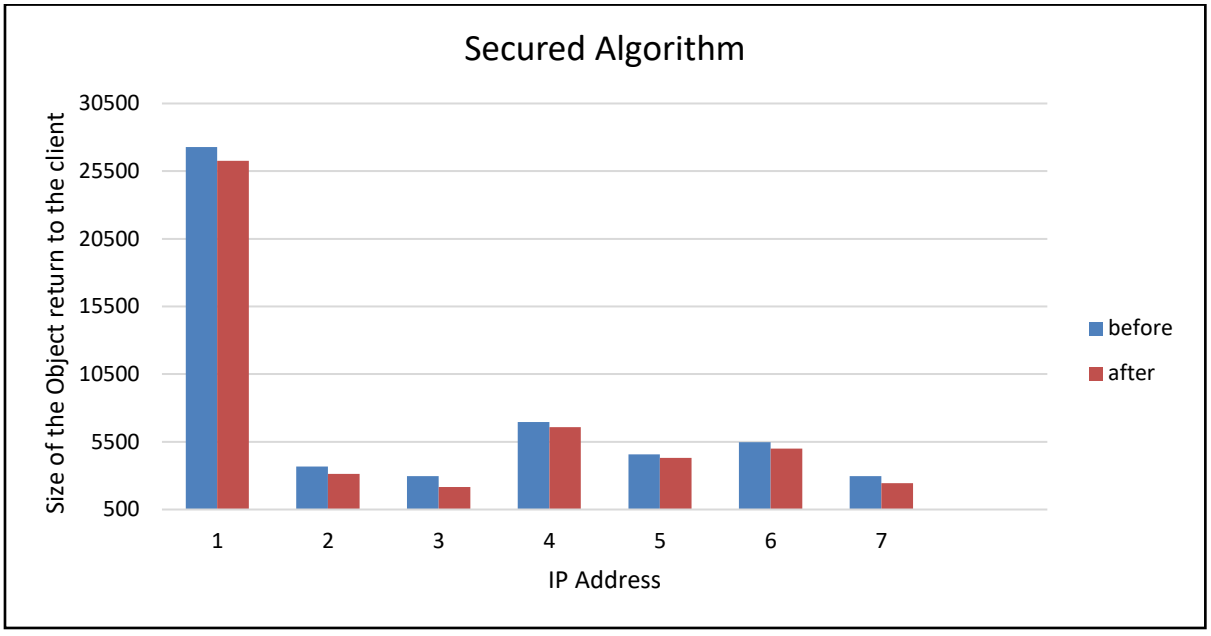

Figure-II : Actual Data after Applying Proposed Algorithm

Table-I: Obtained Results

Phase	Input Logs	Output Logs	Malicious Logs Detected	Sessions Identified
Raw Data Collection	19,283	19,283	0	N/A
Data Cleaning	19,283	12,697	0	N/A
Session Identification	12,697	12,697	0	6,656
Pattern Discovery (Post Sec)	12,697	12,697	0	6,656

Conclusion : This paper presents a robust security framework for web log mining, ensuring data integrity from entry-point validation to session identification. By filtering malicious logs, cleaning erroneous data, and encrypting web logs during pattern discovery, organizations can safeguard sensitive data while enhancing their analytical capabilities. This approach not only fortifies web security but also contributes to more accurate consumer behavior predictions, fostering better decision-making and improved user satisfaction.

References

1. Zhou, X., et al. (2020). On Vulnerability and Security Log Analysis: A Systematic Literature Review. ACM Digital Library.
2. Li, W., & Chen, H. (2019). Application of Web Log Mining in Network Security. IEEE Xplore.
3. Kumar, P., et al. (2021). Collaborative Detection of Malicious Web Logs Using Machine Learning.
4. Rahman, T., et al. (2018). Session-Based Anomaly Detection for Web Log Mining. Journal of Information Security.
5. Sonia Sharma, Dalip (2020). A Novel Secure Web Usage Mining Technique to Predict Consumer Behaviour. International Journal of Advanced Science and Technology. Vol. 29, No. 5, (2020), pp. 5633 – 5640.ISSN: 2005-4238 IJAST.
6. Sonia Sharma, Dalip (2019). Comparative Analysis of various tools to Predict Consumer Behaviour. Journal of Computational and Theoretical Nano science Vol. 16, 3860–3866, 2019.
7. Sonia Sharma, Dalip, "Web Logs - A Roadmap to Online Consumer", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 6 Issue 1, pp. 576-581, January-February 2019.
8. Gurung, A. and Raja, M.K. (2016), "Online privacy and security concerns of consumers", Information and Computer Security, Vol. 24 No. 4, pp. 348-371
9. https://www.sitelock.com/blog/why-website-security-matters-to-your-customers.
10. https://www.loginradius.com/docs/security/data-management/consumer-audit-logs/
11. Atiq, S.M., Ingle, D., Meshram, B.B. (2012). Web Mining and Security in E-commerce. In: Meghanathan, N., Nagamalai, D., Chaki, N. (eds) Advances in Computing and Information Technology. Advances in Intelligent Systems and Computing, vol 176. Springer, Berlin, Heidelberg.