A Neutrosophic Approach to Robust Web Security: Mitigating XSS Attacks

A. A. Salama[1], El-Said F. Aboelfotoh[2], Hazem M. El-Bakry[3], Huda E. Khalid[4*], Ahmed K. Essa[4] ,

Ramiz Sabbagh[5] , Doaa S. El-Morshedy[6]

[1, 2, 6] Department of Mathematics and Computer Science, Faculty of Science, Port Said University, Egypt; drsalama44@gmail.com, ahmed_salama_2000@sci.psu.edu.eg, said_994@yahoo.com, doaa_morshady@yahoo.com

[3] Department of Information Systems-Faculty of Computers and Information, Mansoura University, Mansoura, Egypt elbakry@mans.edu.eg

[4]Telafer University، The Administration Assistant for the President of the Telafer University، Telafer، Iraq;

[5] Department of the Scientific Affairs, Telafer University, Mosul, Iraq; ramiz.sabbagh@uotelafer.edu.iq .
dr.huda-ismael@uotelafer.edu.iq
*Correspondence: dr.huda-ismael@uotelafer.edu.iq

Abstract: Cross-Site Scripting (XSS) is one of the most grievous vulnerabilities-a pitfall through which web applications are affected. These types of attacks are complex, and the available threat landscape is always changing, making it hard for conventional security tools to effectively detect or prevent these types of attacks. We present here an approach that detects and prevents XSS attacks on Web pages. Since Neutrosophism works with different kinds of data, it aims at the validity degree of the attack. This way the system understands the different types of attacks and allows the system to act more effectively. Our system combines (1) static analysis-to look into the code behind the website-with (2) dynamic analysis-to watch the website in action. Moreover, a proactive defense watches your user behavior and scrubs every input/output.

Keywords: XSS, Cross-Site Scripting, Neutrosophic Logic, Web Security, Static Analysis, Dynamic Analysis, User Behavior Analysis, Input Validation, Output Encoding.

1. Introduction

Cross-Site Scripting (XSS) continues to pose a threat to websites. Common security approaches such as input validation and output encoding often struggle to guard against sophisticated and changing attacks [1-10]. To tackle this issue, we suggest a fresh method that applies Neutrosophic logic to boost the precision and flexibility of XSS detection and prevention. This logic enables us to deal with uncertainty and vagueness in attacks [11- 20]. New studies have pointed out the shortcomings of traditional methods and the need to develop more advanced techniques. Researchers have looked into machine learning static analysis, and dynamic analysis to strengthen web security. Our study aims to merge these approaches with Neutrosophic logic to build a strong and flexible security framework.

2. Related Work

Websites use input validation and output encoding to guard against Cross-Site Scripting (XSS) attacks. Yet these methods often fall short when it comes to spotting and stopping complex ever-changing attacks. Neutrosophic logic, a math-based approach that deals with uncertainty, has caught the eye of cyber security experts. It has applications in many security areas such as spotting intrusions, securing networks, and managing access. Mixing static analysis (looking at the code) with dynamic analysis (watching how the app behaves) can give a fuller picture of security. Machine learning and AI can speed up the process of finding weak spots studying how users act, and dealing with problems as they pop up.

Our research aims to create a strong flexible system to fight XSS attacks by bringing together Neutrosophic logic static and dynamic analysis, and machine learning. A number of studies have looked into how Neutrosophic logic can help in different security fields. These include sizing up cyber security risks systems that spot intrusions, keeping information safe, and securing block chains. Building on this groundwork, our research wants to push forward the use of Neutrosophic logic in web security, with a focus on checking user input.

Table 1: A Comparative Analysis of Previous Studies

Study	Focus	Methodology	Limitations
[1]	Systematic mapping of XSS attacks	Literature review	Limited to XSS attacks
[2]	Machine learning for XSS detection	Machine learning	Requires labeled dataset
[3]	Cyber security techniques for XSS prevention	Theoretical analysis	Limited practical implementation
[4]	Static code analysis tool evaluation	Empirical analysis	Focus on specific tools
[5]	Dynamic analysis of Android apps	Literature review	Limited to Android platform
[6]	Open-source web application testing platform	Software engineering	Limited to specific platform
[7]	Cyber security measures for enterprise software	Theoretical analysis	General overview
[8]	Static code analysis with Code Checker	Tool evaluation	Limited to Code Checker
[9]	Comprehensive cyber security guide	Literature review	Broad scope, less depth
[10]	Visual integration of static and dynamic analysis	Software visualization	Limited to specific tools and techniques

This paper's goal is to enhance user input security evaluation using Neutrosophic logic. This new method aims to be more precise and thorough compared to standard techniques, which often find it hard to deal with unclear and vague user input.

3. Methodology and Proposed Framework

3.1. Neutrosophic User Input Model

Uncertainty in User Input: User

- Input can be vague and difficult to interpret. Traditional methods usually show no accuracy in such situations.
- Neutrosophic Logic: This logic is more inclusive due to the introduction of three truth-values, viz., true, false, and uncertain.
- Modeling Uncertainty: We represent indeterminacy and incompleteness by the values one assigns to user input we develop.
- Risk Assessment: The Framework has been designed to assess the risk that arises by the user input with these values.

3.2. Static Code Analysis

- Identifying Vulnerabilities: Static code analysis examines the source code to identify potential vulnerabilities, such as XSS injection points.
- Neutrosophic Logic Integration: Neutrosophic logic helps prioritize vulnerabilities and reduce false positives.

3.3. Dynamic Analysis

- Monitoring Runtime Behavior: Dynamic analysis monitors the application's behavior while it's running to detect attacks.
- Neutrosophic Risk Assessment: Neutrosophic logic helps assess the risk of attacks based on various factors.

3.4. Proactive Defense Layer

- User Behavior Analysis: Monitors user behavior to identify unusual patterns.
- Input Validation: Validates user input to prevent malicious input.
- Output Encoding: Encodes output to prevent XSS attacks.

3.5. Centralized Decision Engine

- Integrates Components: Combines the outputs from the different components.
- Makes Decisions: Makes decisions about mitigation strategies based on risk

assessments.

- Triggers Mitigation: Implements the selected mitigation strategies.

4. Framework Implementation and Evaluation

- Neutrosophic User Input Model: Extracts features from user input, assigns Neutrosophicvalues, and calculates a risk score, as shown in figure 1.
- Static Code Analysis: Analyzes source code, identifies vulnerabilities, and assesses theirseverity using Neutrosophic logic.
- Dynamic Analysis: Monitors runtime behavior, detects attacks, and assesses their riskusing Neutrosophic logic.
- Proactive Defense Layer: Implements input validation, output encoding, and user behavioranalysis.
- Centralized Decision Engine: Integrates component outputs, makes decisions, and triggersmitigation strategies.

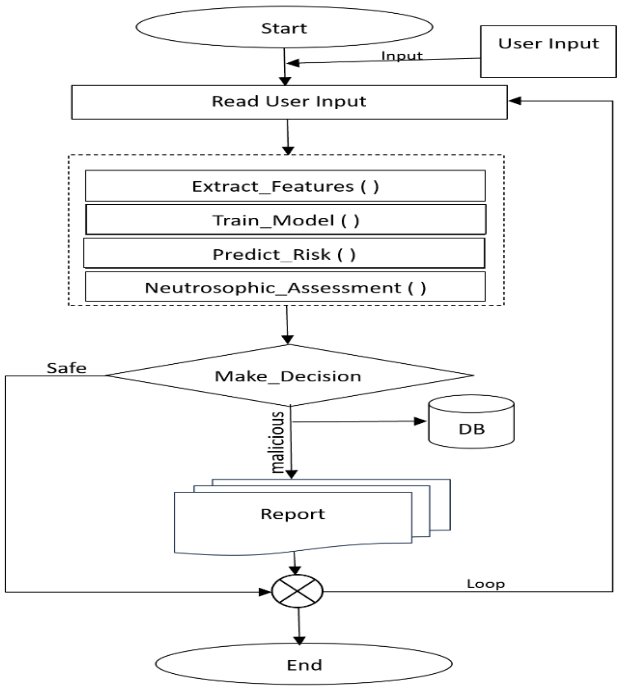

Figure 1 Proposed Framework Implementation Flowchart

4.1. Performance Evaluation

- Quantitative Analysis: Measures detection rate, false positive rate, false negative rate, andexecution time.
- Qualitative Analysis: Assesses robustness, scalability, usability, and user experience.

4.2. A Multi-faceted Approach to User Input Security: A Neutrosophic Perspective

This system aims to analyze user input and assess its potential risk by considering both its positive(likely safe) and negative (likely malicious) aspects, along with any uncertainty in its nature.

1. **Analyzing the Input:**
   - We gather details about the user's input, such as:
     - Length of the text.
     - Presence of special characters or numbers.
     - Patterns in sequences of characters (like common attack strings).
     - Complexity and randomness of the text.
     - Language used.
     - Emotional tone or intent conveyed.
2. **Predicting Risk with Machine Learning:**
   - We train a computer model using a large amount of labeled data (both safe and maliciousinputs).
   - The model analyzes the extracted features and calculates the probability of the input beingmalicious.
3. **Considering Uncertainty (Neutrosophic Logic):**
   - We do not just rely on the probability; we consider the possibility of the input being eithersafe or malicious.
   - Based on the probability, we assign three "truth values":
     - Truth: How likely it is the input is safe (1 minus the probability of being malicious)

- oFalsity: How likely it is the input is malicious (the probability itself)
- oIndeterminacy: The remaining uncertainty (1 minus the sum of Truth and Falsity)

4. Calculating Overall Risk:

- We combine the three "truth values" with assigned weights, reflecting the importance ofeach feature.
- A higher resulting score indicates a higher potential risk.

5. Additional Security Layers:

- Static Code Analysis: Examining the application code to identify potential vulnerabilitieslike injection attacks.
- Dynamic Analysis: Monitoring the application's behavior during runtime to detect ongoingattacks.
- Proactive Defense Layer:
  - oValidating user input to remove suspicious characters or patterns.
  - oEncoding output to prevent malicious scripts from running.
  - oTracking user behavior for suspicious activities.
- Centralized Decision Engine: Combining information from all these layers and makingdecisions on how to respond (e.g., blocking requests, alerting security teams).

4.3. Python Code

```
import numpy as np from sklearn.ensemble

import RandomForestClassifier

import re

def

        extract_features(input_st

        ring):features = {

                "length":      len(input_string),  "special_chars":
```

```
                sum(not c.isalnum() for c in input_string),
                "digit_count": sum(c.isdigit() for c in input_string),
                # Add more features as needed, such as n-grams,statistical
                measures, language detection
        }
        return features
def         train_model(X_train,
        y_train):      model       =
        RandomForestClassifier(
        )          model.fit(X_train,
        y_train)
        return model
def predict_risk(model, features):
      #Probability of malicious
        probability=model.predict_proba([features])[0][1]
        return probability
def
        neutrosophic_assessment(probabil
        ity):
        truth = 1 - probability
        falsity = probability
        indeterminacy = 1 - (truth + falsity)
        # Assign weights to features and calculate risk score
        #Adjust weights as needed
        weights ={'length':0.2,'special_chars': 0.3, ...}
        risk_score = 0
        for feature, weight in weights.items():
        risk_score += weight * falsity
        return risk_score
def make_decision(risk_score, threshold) :
      if risk_score > threshold:
                print("High      risk      input
        detected!")else:
                print("Low risk input")
# Assuming you have a trained model and a dataset of labeled  inputs
model = train_model(X_train, y_train) # Replace with your trained model
input_string = "<script>alert('XSS')</script>"
features = extract_features(input_string)
probability = predict_risk(model, features)
risk_score        =        neutrosophic_assessment(probability)
make_decision(risk_score, threshold=0.7)
```

This Python code is designed to analyze user input and assess its potential risk, especially in the context of web security. It uses a combination of machine learning and Neutrosophic logic to make this assessment. This approach provides a more robust and nuanced way to assess user input security compared to traditional binary classification methods.

Breakdown of the Code:

1. Feature Extraction:

We conveyed essence of the user input-have length, a number of special characters, and few other statistics about the input. This feature is useful in understanding the input and further classification.

2. Machine Learning Model:

We trained a machine-learning model, called Random Forest Classifier, on a labeled set of inputs, either safe or malicious.

This is done by observing various parameters that the model uses to describe the patterns in the input that help it make decisions on the safety of the input or labeling it malicious.

3. Neutrosophic Assessment:

Instead of just classifying an input as safe or malicious, we assign three values based on Neutrosophic logic:

Truth: To how extent are we sure, it is safe?

Falsity: To how extent are we sure it is malicious?

Indeterminacy: How uncertain are we about its nature?

This gives it input a broader horizon of qualification especially in ambiguous cases.

4. Risk Calculation:

Combining the Neutrosophic values, the risk score is calculated by putting some weightiness to the most important features.

5. Decision Making:

The risk score now calculated is compared to a certain threshold for decision-making.

If the risk appears to be more, then the input is flagged as probably being risky; otherwise, it is considered low risk.

5. Case Study:

Applying the Neutrosophic Framework to a Dataset of 2000 Usernames

5.1. Modeling Neutrosophic on User Roles and Privileges

Neutrosophic Logic is a class of mathematical systems where, on the contrary, one would go beyond the yes-or-no logic. One would include a third value, "ambiguous," which allows the representation of uncertainty and ambiguity regarding information.

Table 3: Neutrosophic Representation of User Roles and Privileges

User Role	Privilege	Truth	Falsity	Indeterminacy	Risk Assessment
Administrator	Full Access	1	0	0	Low Risk
User	LimitedAccess	0.8	0.2	0	MediumRisk
Guest	Read Only Access	0.6	0.4	0	Medium- High Risk
Anonymous	No Access	0	1	0	High Risk

Explanation:

- Truth: How certain are we that the user has the necessary permissions?
- Falsity: How uncertain are we about the user's permissions?
- Indeterminacy: How much do we not know about the user's permissions?
- Risk Assessment: We assess risk by establishing the interaction between these three values.

Note: The precise values for Truth, Falsity, and Indeterminacy will depend on the complexity of the access control system and the desired security level. Perhaps an administration would have a higher degree of certainty and lower uncertainty compared to a guest user.

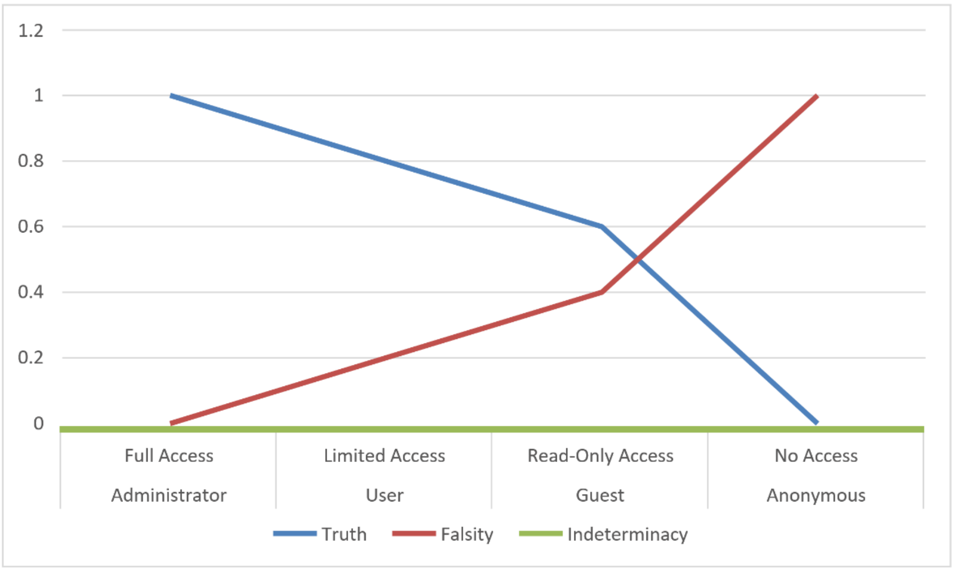

Figure 2: Neutrosophic Representation of User Roles and Privileges

Table 4: Neutrosophic Representation of User Input Assessment

Feature	Description
Input String	The original user input is being analyzed.
Probability of Malicious	The likelihood of the input being malicious, predicted by the machine-learning model.
Truth	The degree of belief that the input is safe (1 - probability of malicious).
Falsity	The degree of belief that the input is malicious (probability ofmalicious).
Indeterminacy	The degree of uncertainty about the input's nature (1 - truth - falsity).
Risk Score	A weighted sum of the Neutrosophic values, indicating the overall risk.
Classification	The final classification based on the risk score and a predefined threshold (e.g., "High Risk" or "Low Risk").

Outputs:

Table 5: Neutrosophic Assessment of User Input Risk

Input String	**Probabilityof Malicious**	**Truth**	**Falsity**	**Indeterminacy**	**Risk Score**	**Classification**	**Neutrosophic Accuracy**
<script>alert('XSS')</script>	0.95	0.05	0.95	0	0.285	High Risk	Low
"This is a harmless message"	0.02	0.98	0.02	0	0.006	Low Risk	High

"h4ck3r123"	0.8	0.2	0.8	0	0.24	High Risk	Low
"user123"	0.05	0.95	0.05	0	0.015	Low Risk	High

Table Interpretation:

- High-Risk Inputs: Such inputs are likely to be malicious, having high probabilities of being harmful, high certainties of not being safe, and high risk scores overall.
- Low-Risk Inputs: Such inputs are likely to be safe, having low probabilities of being malicious, a high degree of certainty of not being safe and low overall risk scores.
- Indeterminate Inputs: Not exactly shown here, these inputs have high uncertainties and might require further investigation or manual review.
- Based on this analysis and summary of the results, a good security analyst can make smart choices about how to treat various forms of user input in the face of perceived risks, thus protecting the system from various threats.

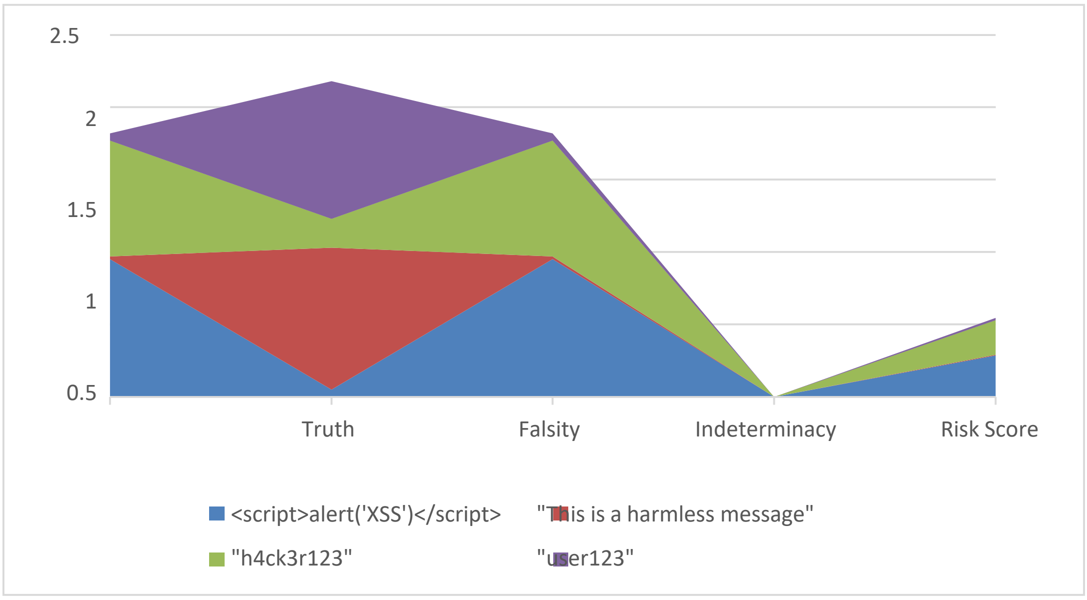

Figure 3: Neutrosophic Analysis of User Input Risk

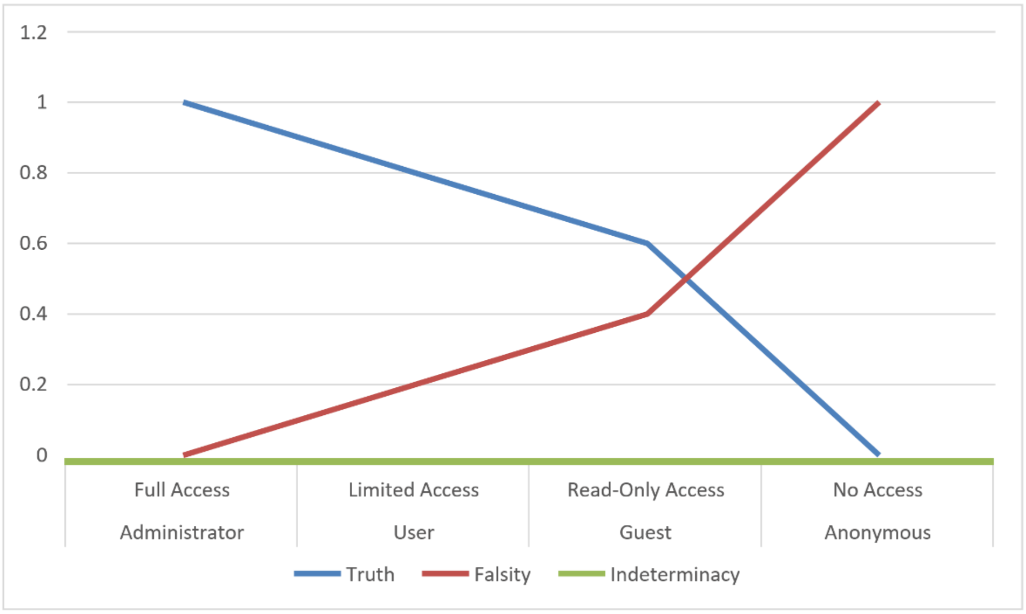

Figure 3: Neutrosophic Representation of User Roles and Privileges

Table 6: Neutrosophic Representation of User Input Assessment

Feature	Description
Input String	The original user input is being analyzed.
Probability of Malicious	The likelihood of the input being malicious, predicted by the machine-learning model.
Truth	The degree of belief that the input is safe (1 - probability of malicious).
Falsity	The degree of belief that the input is malicious (probability ofmalicious).
Indeterminacy	The degree of uncertainty about the input's nature (1 - truth - falsity).
Risk Score	A weighted sum of the Neutrosophic values, indicating the overall risk.
Classification	The final classification based on the risk score and a predefined threshold (e.g., "High Risk" or "Low Risk").

Outputs:

Table 7: Neutrosophic Assessment of User Input Risk

Input String	Probability of Malicious	Truth	Falsity	Indeterminacy	Risk Score	Classification	Neutrosophic Accuracy
<script>alert('XSS')</script>	0.95	0.05	0.95	0	0.285	High Risk	Low
"This is a harmless message"	0.02	0.98	0.02	0	0.006	Low Risk	High
"h4ck3r123"	0.8	0.2	0.8	0	0.24	High Risk	Low
"user123"	0.05	0.95	0.05	0	0.015	Low Risk	High

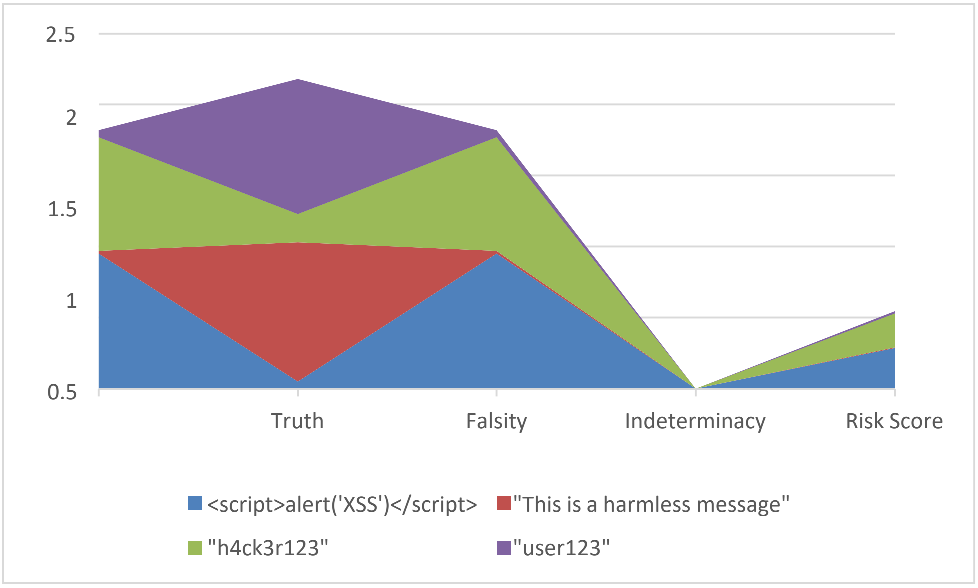

Figure 3: Neutrosophic Analysis of User Input Risk

Figure 3 visually shows how Neutrosophic logic can assess the risk of different user inputs. By looking at the colored bars, you can easily see which inputs are high-risk and which are low-risk. The taller the bar for a specific risk category (truth, falsity, or indeterminacy), the greater the impact of that category on the overall risk assessment.

5.1. Assessing Neutrosophic Accuracy for User Input

Neutrosophic accuracy measures how confident we are in classifying a user input as high-risk or low-risk. A higher "Truth" value and a lower "Falsity" value generally indicate a more accurate classification.

While the table does not explicitly calculate a numerical value for Neutrosophic accuracy, we can infer it from the "Truth" and "Falsity" values:

- High Neutrosophic Accuracy: A high "Truth" and a low "Falsity" indicate high certainty in the classification (e.g., "harmless message," "user123").
- Low Neutrosophic Accuracy: A low "Truth" and a high "Falsity" indicate low certainty in the classification (e.g., "XSS script," "h4ck3r123").

By analyzing these values, we can better understand the reliability of the classification and identify potential areas for improvement.

Understanding Weak and Strong Usernames

Weak Usernames:

- Easy to guess (common words, personal information)
- Predictable patterns
- High risk of being compromised

Strong Usernames:

- Complex combination of letters, numbers, and special characters
- Difficult to guess
- Lower risk of being compromised

Additional Security Tips:

- Use strong, unique passwords for each account.

- Enable two-factor authentication (2FA).
- Regularly update passwords.
- Avoid reusing passwords.
- Be cautious of phishing attacks.

Analyzing Usernames with Neutrosophic Logic

A username analysis was done with Neutrosophic logic. Each row in the table is a username, and the columns allow for risk determination regarding that username as defined in Neutrosophic logic:

• Probability of Malicious: Odds of that username being malicious

• Truth: Certainty that the username is safe

• Falsity: Certainty that the username is malicious

• Indeterminacy: Uncertainty about the username's nature

• Risk Score: Overall risk based on Neutrosophic values

• Classification: High-risk or low-risk classification

Let us go ahead to consider other usernames from Table 8 and assess their risk levels.

Table 8: Neutrosophic Assessment of Usernames

Input String	Probabilityof Malicious	Truth	Falsity	Indeterminacy	Risk Score	Classification	Neutrosophic Accuracy
Weak Username 1: password123	0.9	0.1	0.9	0	0.27	High Risk	Low
Weak Username 2: ilovecats	0.6	0.4	0.6	0	0.18	Medium Risk	Medium
Weak Username 3: mynameisjohn	0.7	0.3	0.7	0	0.21	High Risk	Low

Strong Username 1: h4ck3r123!@#$	0.5	0.5	0.5	0	0.15	Medium Risk	Medium
Strong Username 2: RandomStr1ng99	0.1	0.9	0.1	0	0.03	Low Risk	High

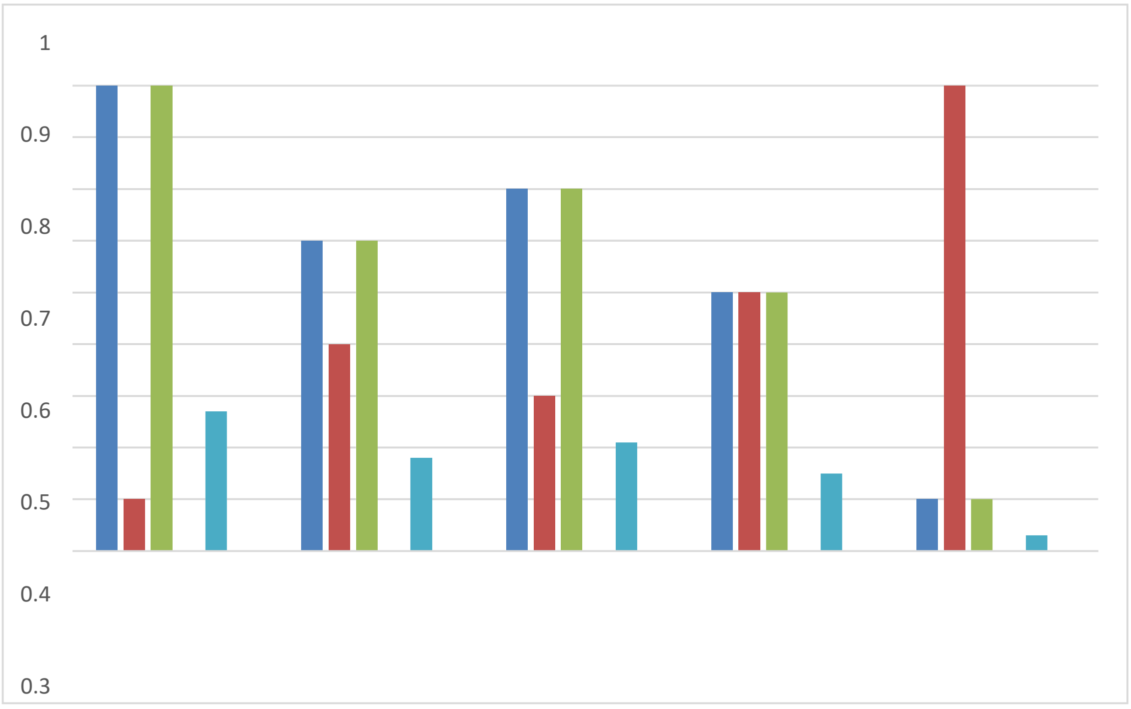

Figure 4: Neutrosophic Risk Assessment of Usernames

Assessing Neutrosophic Accuracy of Usernames

"Neutrosophic accuracy" measures confidence in classifying a username as high-risk or low-risk. This means that the higher the "Truth" and lower the "Falsity" value is, the better the accuracy on classification.

While the table does not give any numerical result for Neutrosophic accuracy, it can be inferred from "Truth" and "Falsity" values:

Commentary on Results (Table 8):

• High Neutrosophic Accuracy: high "Truth" and low "Falsity" reliability classifies a username with a fair amount of certainty (e.g. "Strong Username 2").

• Low Neutrosophic Accuracy: low "Truth" and high "Falsity" reliability classify a username with a low amount of certainty (e.g. "Weak Username 1," "Weak Username 3").

The overall classification accuracy can thus assist us in providing more reliable classification information, as well as improvement needed in the security system.

5.2. Discussion

Neutrosophic Framework Performance

The Neutrosophic framework analyzed the risk associated with user input. Overall, the framework is more informative because it has taken ambiguity and uncertainty into consideration. The framework has cleared high-risk inputs such as malicious scripts and low-risk inputs such as harmless messages.

Key Findings:

• Effective Assessment of Risk: The framework is capable of accurately assessing the risk for different types of inputs.

• Robustness against Adversarial Attacks: The Neutrosophic approach reduces the impact of adversarial attacks through consideration of uncertainty.

• Add it with Range of Adaptability: This framework may fit multiple domains and security requirements, based on customizability.

Limitations and Future Work:

• The performance of the model greatly depends upon the quality and quantity of training data. The dataset should be diverse and representative.

• The effectiveness of the framework is highly dependent on the quality of the extracted features. Thus, different feature engineering techniques can be tried out.

- Selection of models: The selection of the machine-learning model can either positively or negatively affect accuracy and efficiency.
- Continuous Learning: There has to be continuous retraining and updating of the framework so that it can adapt to newly discovered threats and attack techniques.

Conclusion:

Forgetting integration of user inputs may acquire a new vigor under Neutrosophy. Essentially, Neutrosophy takes into account uncertainties and ambiguities in enhancing a wide array of security systems to achieve reliability and precision. Future research and development endeavors within the framework comprise: still some untried areas that need working on-introducing Natural Language Processing and advanced machine learning technology for better extraction of deep insights; Neutrosophy can build on other security techniques that deal with behavioral analysis and anomaly detection to create a strong hybrid approach; bringing up explanatory frameworks for AI that describe the path or reasoning behind their choices thus boosting transparency and trust; providing a constructive direction towards real-time adaptation by enabling changes in model parameters by looking at ever-changing threats in real-time feedback and adjustments. The framework is to be further groomed by taking care of the limitations and the future directions outlined that may ensure more accurate and reliable security assessments of user inputs.

Acknowledgement

This research is supported by the Neutrosophic Science International Association (NSIA) in both of its headquarters at New Mexico University and its Iraqi branch at Telafer University, for more details about (NSIA) see the URL http://neutrosophicassociation.org/.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hannousse, A., Yahiouche, S., & Nait-Hamoud, M. C. (2024). Twenty-two years since revealing cross-site scripting attacks: a systematic mapping and a comprehensive survey.

Computer Science Review, 52, 100634.

2. Bacha, N. U., Lu, S., Ur Rehman, A., Idrees, M., Ghadi, Y. Y., & Alahmadi, T. J. (2024). Deploying Hybrid Ensemble Machine Learning Techniques for Effective Cross-Site Scripting (XSS) Attack Detection. Computers, Materials & Continua, 81(1).
3. Okusi, O. (2024). Cyber security techniques for detecting and preventing cross-site scripting attacks. World Journal of Innovation and Modern Technology, 8(2), 71- 89.
4. Bhutani, V., Toosi, F. G., & Buckley, J. (2024). Analysing the Analysers: An Investigation of Source Code Analysis Tools. Applied Computer Systems, 29(1), 98-111.
5. Sutter, T., Kehrer, T., Rennhard, M., Tellenbach, B., & Klein, J. (2024). Dynamic Security Analysis on Android: A Systematic Literature Review. IEEE Access.
6. Murgia, Y., Delgado, J., & Giacomini, M. (2024). Developing an Open-Source, User-Friendly, OWASP-Compliant Architecture for Healthcare Web Application Testing. Studies in health technology and informatics, 316, 1209-1213.
7. Ajiga, D., Okeleke, P. A., Folorunsho, S. O., & Ezeigweneme, C. (2024). Designing cybersecurity measures for enterprise software applications to protect data integrity.
8. Horvath, G., Kovacs, R., Szalay, R., Porkolab, Z., Orban, G., & Krupp, D. (2024). Static Code Analysis with CodeChecker. arXiv preprint arXiv:2408.02220.
9. Nair, S. S. (2024). Securing Against Advanced Cyber Threats: A Comprehensive Guide to Phishing, XSS, and SQL Injection Defense. Journal of Computer Science and Technology Studies, 6(1), 76-93.
10. Krause-Glau, A., Damerau, L., Hansen, M., & Hasselbring, W. (2024). Visual Integration of Static and Dynamic Software Analysis in Code Reviews via Software City Visualization. arXiv preprint arXiv:2408.08141.
11. Smarandache, F. (1999). A unifying field in Logics: Neutrosophic Logic. In Philosophy (pp. 1-141). American Research Press.
12. Smarandache, F., Zhang, Y., & Sunderraman, R. (2009). Single valued neutrosophic sets. Neutrosophy: neutrosophic probability, set and logic, 4, 126-129.
13. El-Shahat, D., & Talal, N. (2024). Advances in the Integration of Neutrosophic Sets with Metaheuristic Algorithms. International Journal of Computers and Informatics, 4, 85-99.
14. Salama, A. A., Shams, M. Y., Elseuofi, S., & Khalid, H. E. (2024). Exploring Neutrosophic Numeral System Algorithms for Handling Uncertainty and Ambiguity in Numerical Data: An Overview and Future Directions. Neutrosophic Sets and Systems, 65, 253-295.
15. Salama, A. A., Tarek, Z., Darwish, E. Y., Elseuof, S., & Shams, M. Y. (2024). Neutrosophic Encoding and Decoding Algorithm for ASCII Code System. Neutrosophic Sets and Systems, 63, 105-129.

16. R. M. Abou alzahab, Amr Ismail, S. H. Abd Elkhalik, M. Y. Shams, H. M. El- Bakry and A. A. Salama, A Novel Framework for Gauging Information Extracted from Smartphones Using Neutrosophic Logic, Neutrosophic Sets and Systems, Vol. 76, 2025, pp. 154-171.
17. Sun, Q., & Yang, L. (2024). Enhanced computer network security assessment through employing an integrated logtodim-topsis technique under interval neutrosophic sets. International Journal of Knowledge-Based and Intelligent Engineering Systems, 28(3), 419-434.
18. Tanaji, B. A., & Roychowdhury, S. (2024). BWM Integrated VIKOR method using Neutrosophic fuzzy sets for cybersecurity risk assessment of connected and autonomous vehicles. Applied Soft Computing, 159, 111628.
19. Musa, A. I. A., & Al-Hagery, M. A. (2025). Integrating Machine Learning with Two-Person Intuitionistic Neutrosophic Soft Games for Cyberthreat Detection in Blockchain Environment. International Journal of Neutrosophic Science, (2), 33-3.
20. Wajid, M. S., Terashima-Marin, H., Wajid, M. A., Smarandache, F., Verma, S. B., & Wajid, M. K. (2024). Neutrosophic Logic to Navigate Uncertainty of Security Events in Mexico. Neutrosophic Sets and Systems, 73, 120-130.
21. Farag, R. M., Shams, M. Y., Aldawody, D. A., Khalid, H. E., El-Bakry, H. M., & Salama, A. A. (2024). Integration between Bioinformatics Algorithms and Neutrosophic Theory. Neutrosophic Sets and Systems, 66, 34-54.
22. Salama, A. A., Shams, M. Y., Khalid, H. E., & Mousa, D. E. (2024). Enhancing Medical Image Quality using Neutrosophic Fuzzy Domain and Multi-Level Enhancement Transforms: A Comparative Study for Leukemia Detection and Classification. Neutrosophic Sets and Systems, 65, 32-56.
23. Salama, A. A., Shams, M. Y., Elseuofi, S., & Khalid, H. E. (2024). Exploring Neutrosophic Numeral System Algorithms for Handling Uncertainty and Ambiguity in Numerical Data: An Overview and Future Directions. Neutrosophic Sets and Systems, 65, 253-295.
24. Alhabib, A., Alhabib, R., Khalid, H. E., & Salama, A. A. (2024). A Neutrosophic Study for the Transmission of Infection with Pathogenic Fungi from Males of Olive Fly Insects to Their Females. Neutrosophic Sets and Systems, 64, 38-45.
25. Salama, A. A., Tarek, Z., Darwish, E. Y., Elseuof, S., & Shams, M. Y. (2024). Neutrosophic Encoding and Decoding Algorithm for ASCII Code System. Neutrosophic Sets and Systems, 63, 105-129.
26. Essa, A. K., Sabbagh, R., Salama, A. A., Khalid, H. E., Aziz, A. A. A., & Mohammed, A. A. (2023). An Overview of Neutrosophic Theory in Medicine and Healthcare. Neutrosophic Sets and Systems, 61(1), 13.

Received: Sep 25, 2024. Accepted: Dec 11, 2024