A Support Vector Machine-Based Cross-Site Scripting Vulnerability Detection Technique Postprint

Authors: Huang Nana, Wan Liang

Date: 2018-05-20T00:00:00+00:00

Abstract

Cross-Site Scripting (XSS) represents a common vulnerability attack vector targeting Web application security. Malicious users exploit vulnerabilities to inject malicious scripts into web pages; when users browse these pages, the scripts are triggered, resulting in attack execution. To address the challenge of detecting various mutated XSS attacks, this study investigates a Regular Expression and Support Vector Machine based Recursive Feature Elimination algorithm (RE-SVM-RFE). Initially, a regular expression matching algorithm is employed to select representative features for the training set, constituting data preprocessing. Subsequently, the RE-SVM-RFE feature selection algorithm is utilized to identify optimal features and rank attack keywords. Finally, by analyzing the occurrence frequency of feature keywords, it is observed that higher frequency correlates with increased likelihood of vulnerability existence. Experimental results demonstrate that SVM features selected via the RE-SVM-RFE recursive feature elimination algorithm achieve superior prediction accuracy, sensitivity, and specificity, confirming that the algorithm can effectively detect XSS vulnerabilities.

Full Text

Preamble

Title: A Cross-Site Scripting Vulnerability Detection Technique Based on Support Vector Machines

Authors: Huang Nanaa,b, Wan Lianga,b†
Affiliation: Guizhou University, a. College of Computer Science and Technology; b. Institute of Computer Science Theory, Guiyang 550025, China

Abstract

Cross-site scripting (XSS) represents a prevalent attack vector targeting Web application security. Malicious actors exploit vulnerabilities to inject malicious scripts into web pages, which execute when users browse the compromised pages, triggering the attack. To address the challenge of detecting various mutated XSS attacks, this paper investigates a recursive feature elimination algorithm based on regular expressions and support vector machines (RE-SVM-RFE). The approach first employs regular expression matching to select representative features from the training set for data preprocessing. Subsequently, the RE-SVM-RFE feature selection algorithm identifies optimal features and ranks attack-related keywords by significance. By analyzing the frequency of characteristic keywords, we observe that higher frequencies correlate with greater likelihood of vulnerability presence. Experimental results demonstrate that SVM features selected through the RE-SVM-RFE recursive feature elimination algorithm achieve higher prediction accuracy, sensitivity, and specificity, proving the algorithm' s effectiveness in XSS vulnerability detection.

Keywords: support vector machine; cross-site scripting attack; feature vector; Web security; feature selection; RE-SVM-RFE algorithm

0 Introduction

Since the commercialization of the Web, its continuous growth has attracted increasing attention from developers and users due to its openness and usability. However, as the Web provides convenience to human life, various vulnerabilities embedded in Web applications have become among the most serious security threats online. Cross-site scripting (XSS) attacks occur when data from untrusted sources (typically Web requests) enters a Web application and gets included in dynamic content delivered to users without proper validation of malicious script content. When end users browse such pages, the unvalidated malicious script may execute, enabling attackers to steal private data (such as cookies or session information) or redirect victims to attacker-controlled Web content.

Research on XSS detection dates back to the last century, with numerous methods proposed since then. Early approaches employed penetration testing, combining vulnerability analysis with dynamic analysis during the testing process and utilizing extended taint propagation models to detect XSS presence. Other researchers proposed behavior-based contrast methods for vulnerability identification, determining system vulnerabilities by comparing system behavior under normal versus attacked conditions. Some designed Web server protection tools as intrusion detection systems capable of tracking suspicious hosts in real-time and detecting malicious XSS, though these lacked generality and only supported Apache Web servers. Alternative approaches used software fault injection techniques to detect automated Web vulnerability scanners by injecting common

software fault types into Web application code and scanning for XSS vulnerabilities, but suffered from low coverage and high false positive rates. Another two-phase static detection method sought to find and remove XSS vulnerabilities in server-side code, employing taint analysis in the first phase to track user inputs and identify potential vulnerability points, then using pattern matching and data dependency analysis in the second phase to locate and remove vulnerabilities. While effective for server-side code analysis, this method could not detect DOM-based XSS due to its inability to analyze client-side code. More recent work proposed using optimized attack vectors for XSS vulnerability detection, automatically generating vector graphs and optimizing attack vector counters, though requiring lengthy training periods.

Despite these efforts, most existing methods have inherent limitations, and few researchers have leveraged support vector machine classifiers for XSS vulnerability detection. Therefore, this paper proposes a feature recombination algorithm combining regular expression matching with support vector machine-based recursive feature elimination (RegEx and recursive feature elimination based on support vector machine, RE-SVM-RFE) specifically for XSS vulnerability detection.

1 Basic Principles of Support Vector Machines

The principle of SVM involves finding an optimal classification hyperplane that satisfies classification requirements while maximizing the margin on both sides of the hyperplane. Theoretically, SVM can achieve optimal classification for linearly separable data. SVM originates from the optimal classification hyperplane in linearly separable cases, with its fundamental concept illustrated in Figure 2 [Figure 2: see original paper].

Assuming a training sample set for two-class data classification: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}$, where $\mathbf{x}_i$ are the feature vectors and y_i are the associated class labels. In binary classification, each y_i takes one of two values, $\{-1, +1\}$, indicating class membership. If a linear function can completely separate the two classes of samples, the data is called linearly separable; otherwise, it is non-linearly separable. The optimal hyperplane is shown in Figure 2 [Figure 2: see original paper].

2 RE-SVM-RFE Feature Selection Algorithm

The purpose of feature optimization is to retain the most valuable detection features with low mutual correlation. For two-class classification problems, this means achieving smaller average similarity and greater separability between the two sample classes. Therefore, we can define a class separability metric to represent differences between the two sample classes. Greater class separability indicates smaller similarity and easier separation between classes. Feature selection and extraction involve choosing or extracting the feature set most relevant to the task from a candidate feature set. The basic framework for feature

selection is shown in Figure 3 [Figure 3: see original paper].

The XSS attack detection process involves collecting Web request datasets from both normal pages and pages with XSS vulnerabilities, extracting features from URLs, JavaScript code, and POST requests in both page types to establish a sample dataset. Using the collected payload data, the regular expression matching algorithm preprocesses the original sample dataset to extract optimal features, which then undergo combinatorial screening to form a feature dataset. This dataset feeds into the SVM algorithm for model training, with results passed to a classifier for detection, ultimately producing detection outcomes. The XSS attack detection flow is illustrated in Figure 1 [Figure 1: see original paper].

2.1 Regular Expression Matching Algorithm

Regular expressions, with their powerful and flexible expressive capabilities, have become a primary tool for describing new-generation rules. Using regular expressions to describe attack features is more accurate, convenient, and effective than traditional exact string extraction methods. XSS attacks relate to string variable operations in programs, where security risks arise through injected script code. To analyze the value sets of string variables in datasets, this work employs regular expression matching algorithms to represent their values across different datasets, making regular expression matching the optimal algorithm for attribute selection.

2.2 RE-SVM-RFE Algorithm

The RE-SVM-RFE algorithm builds upon the support vector machine recursive feature elimination algorithm with improvements to the feature screening process. Before processing the dataset with RE-SVM-RFE, the regular expression matching algorithm first selects representative feature data from the dataset. This experiment selected 46 representative features from the dataset. The algorithm then iteratively trains to identify optimal features, groups the selected features for testing, and uses recursive feature elimination to compute combination feature weight values. Each iteration ranks the combination feature weights and removes the smallest. While the initial feature set is relatively large, each iteration removes one combination feature. Through RE-SVM-RFE iteration, this experiment ultimately selected six optimal features, which were recombined to test differences between RE-SVM-RFE and SVM-RFE.

2.2.1 Iterative Process of RE-SVM-RFE During RE-SVM-RFE execution, the algorithm identifies and removes the feature with minimum weight at each step. Each removal involves three execution steps:

a) Train the classifier using the sample dataset, ensuring feature data correlates with classifier features (such as optimized weight values $\mathbf{w}$);

b) Screen features based on those selected by the preceding regular expression matching algorithm while calculating attribute values for the 46 selected features (such as cost function J);

c) Remove the minimum weight feature from the dataset, i.e., the feature with the smallest ranking criterion.

The iteration continues until only one feature variable remains in the dataset, ultimately producing a feature ranking list ordered by importance. This experiment selected six optimal features.

For linear SVM, the cost function for eliminating the h-th feature is:

$$DJ(h) = \frac{1}{2}\mathbf{w}^T\mathbf{w} - \frac{1}{2}\mathbf{w}_{(h)}^T\mathbf{w}_{(h)}$$

For non-linear SVM:

$$DJ(h) = \frac{1}{2}\alpha^T H\alpha - \frac{1}{2}\alpha_{(h)}^T H_{(h)}\alpha_{(h)}$$

where H is a matrix with elements $H_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, and $K(\mathbf{x}_i, \mathbf{x}_j)$ represents the kernel function measuring similarity between instances $\mathbf{x}_i$ and $\mathbf{x}_j$.

2.2.2 RE-SVM-RFE Algorithm Procedure The overall RE-SVM-RFE algorithm procedure is as follows:

Input: Training sample matrix $X = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, class labels $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$, current feature subset vector $\mathbf{s} = [1, 2, \ldots, k]$, feature ranking vector $\mathbf{r} = [\,]$

Process: 1. Initialize feature ranking: $\mathbf{r} = [\,]$ 2. While $\mathbf{s}$ is not empty: a. Restrict training data to current feature subset: $X = X(:, \mathbf{s})$ b. Train SVM classifier: $\alpha = \text{SVM}_{\text{train}}(X, \mathbf{y})$ c. Compute weight vector: $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ d. Compute ranking criterion: $\mathbf{c} = DJ(\mathbf{w})$ e. Find feature with minimum ranking score: $f = \arg\min(\mathbf{c})$ f. Update feature ranking: $\mathbf{r} = [\mathbf{s}(f), \mathbf{r}]$ g. Remove feature with minimum score from dataset: $\mathbf{s} = \mathbf{s}(1 : f-1, f+1 : \text{length}(\mathbf{s}))$ 3. **Output:** Feature ranking list $\mathbf{r}$

Through this screening process, RE-SVM-RFE ultimately obtains an optimal feature ranking table. Individual features ranked at the front do not necessarily yield good SVM classifier performance; multiple features must be combined to achieve optimal classification performance. To obtain better classification results, the SVM model must first be trained. The feature ranking table is used to define a certain number of nested feature subsets $F_1 \subset F_2 \subset \cdots \subset F_n$ to train the SVM model, which then predicts accuracy to evaluate these feature subsets, ultimately obtaining the optimal feature subset.

This work employs 10-fold cross-validation with fixed parameter dimensions to allocate training and testing sets. During feature ranking table training, SVM

parameter optimization methods are used for parameter tuning. The detailed feature selection framework is shown in Figure 4 [Figure 4: see original paper].

2.3 Data Processing Results of RE-SVM-RFE Algorithm

Payload data extracted from HTTP request packets is unstructured and requires structural processing. The algorithm selects a set of k optimal features from original features, reducing the dimensionality of the original feature space without decreasing classification accuracy, and converting raw payload data into fixed-dimension feature vectors as input for XSS attack detection.

By analyzing various forms of XSS attack statements in vulnerability verification phases, exploitation phases, and various bypass scenarios, this work combines manual selection, mathematical statistics, and the RE-SVM-RFE algorithm for feature selection from original payload data. Through continuous iteration and group testing of selected features, six optimal features were ultimately selected: frequency of various truncation and closure special characters (special character count/total character count), frequency of uppercase letters, frequency of attack characteristic keywords, frequency of lowercase letters, frequency of numeric characters, and frequency of space characters. Feature summaries and specific meanings are shown in Table 1 .

Common attack characteristic keywords in XSS attack statements include: script, java, iframe, alert, img, style, prompt, location, hash, src, href, eval, XMLHttpRequest, ActiveXobject, @import. Common special characters include: <, >, “, ’ , %, (,), !, #, &, :, =, ?, @, [,], /, , {, }, |, $, „ *, +, -, ;, ..

3 Experiments

3.1 Experimental Preparation

The datasets used in this experiment were obtained from the Internet through two sources. The normal sample dataset was derived from analyzing Web server logs to extract requests for benign resources, while the XSS attack sample dataset was obtained by analyzing vulnerability submission sites XSSED, HA.CKKERS, and exploit-db. The experimental dataset comprises two parts: normal payload request samples and XSS vulnerability samples. In preliminary work, 1,378 sample data were collected, as shown in Table 4 .

The experimental host configuration includes an Intel I5 CPU at 2.0 GHz, 8 GB RAM, and a 64-bit Windows 10 operating system. For existing machine recognition methods, the Weka experimental platform was used for testing with SVM classifiers to identify XSS attacks. The proposed algorithm was implemented using Microsoft Visual Studio 2015 and Microsoft SQL Server 2012 for dataset feature preprocessing. First, 46 features were summarized from collected raw data, then the regular expression matching algorithm performed data preprocessing, followed by the proposed RE-SVM-RFE feature selection algorithm to identify the best features. The six most representative features were selected

as the final validation dataset. Optimal features should exhibit stability, discriminability, and relative independence—stability means similar feature values within the same class, discriminability means distinct differences between different classes, and relative independence means low correlation between different features. The six optimal features extracted in this experiment accurately reflect the essential characteristics of Web request payload data.

Feature meanings are: Feature 1—special character frequency; Feature 2—numeric character frequency; Feature 3—lowercase letter frequency; Feature 4—uppercase letter frequency; Feature 5—frequency of different types of attack keyword features; Feature 6—category label indicating whether corresponding attack keywords appear in payload data. The optimal sample feature values are shown in Tables 5 and 6 .

3.2 Results Analysis

To verify algorithm correctness, evaluation metrics including confusion matrix, precision, recall, ROC curves, and F-measure values were employed. Confusion matrix parameters include: True Positive (TP)—actual positive samples predicted as positive; False Positive (FP)—actual negative samples predicted as positive; True Negative (TN)—actual negative samples predicted as negative; False Negative (FN)—actual positive samples predicted as negative. Total samples = TP + FP + TN + FN. The classification result confusion matrix is shown in Table 7 .

Precision and recall are typically discussed together but were tested separately in this experiment for detection convenience. Precision P and recall R are calculated as:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + FN}$$

The confusion matrix data obtained by the proposed algorithm is shown in Table 8 . The main diagonal values far exceed off-diagonal values, indicating high accuracy.

To compensate for accuracy-only evaluation, sensitivity and specificity were also measured. Table 3 compares these metrics across four feature groupings: Feature 1 vs. Feature 2, Feature 2 vs. Feature 3, Feature 3 vs. Feature 4, and Feature 4 vs. Feature 5. The RE-SVM-RFE algorithm outperforms SVM-RFE in both average accuracy and standard deviation. For Feature 1 vs. Feature 2, Feature 3 vs. Feature 4, and Feature 3 vs. Feature 4 vs. Feature 5, RE-SVM-RFE classification accuracy exceeds SVM-RFE by 5.73%, 7.36%, and 6.56% respectively. For Feature 2 vs. Feature 3, while accuracy is equivalent, RE-SVM-RFE shows significantly lower standard deviation, indicating more stable results.

The algorithm execution results demonstrate that RE-SVM-RFE selects more discriminative features—features with strong class separation capability. Attack

characteristic keywords in XSS attack sample data indeed affect SVM model construction and consequently feature evaluation, making regular expression-based data preprocessing before SVM-RFE feature selection highly meaningful.

ROC curves measure learner generalization performance, determined by two important metrics calculated by the learner. The ROC curve uses coordinate axes where the horizontal axis represents False Positive Rate (FPR) and the vertical axis represents True Positive Rate (TPR), defined as:

$$TPR = \frac{TP}{TP + FN}, \quad FPR = \frac{FP}{TN + FP}$$

The area under the ROC curve (ROC area) closer to 1 indicates higher detection accuracy. Table 9 shows the proposed algorithm achieves the best ROC area value, demonstrating high XSS attack detection rates.

F-measure, defined as the weighted harmonic mean of precision and recall, was used to evaluate test accuracy:

$$F\text{-measure} = \frac{2 \times P \times R}{P + R}$$

Table 9 shows the SVM classifier achieves the best F-measure value, indicating high XSS attack recognition rates.

Training and testing time consumption for different classifiers is shown in Table 10 . While the J48 classifier requires only 0.11s total time, its accuracy is 85.7093%. The NaiveBayes classifier requires 0.05s total time with 86.2845% accuracy. The proposed algorithm requires 0.04s training time (second only to NaiveBayes at 0.01s) but achieves 6.7944% higher recognition accuracy than NaiveBayes. Training time also depends on algorithm optimization—the well-optimized SVM algorithm runs efficiently, and RE-SVM-RFE algorithm time complexity can be further optimized.

4 Conclusion

This research involved substantial preliminary work analyzing primary XSS vulnerability features and understanding vulnerability causes, distinguishing between normal user inputs and attacker inputs to correctly differentiate legitimate requests from XSS attack requests, and collecting and organizing datasets for XSS attack detection. Based on SVM model training principles, this paper proposes a feature elimination and selection algorithm combining regular expression matching with support vector machines. The approach preprocesses datasets through regular expression matching, optimizes preprocessed data using kernel functions, detects XSS vulnerabilities via sequential minimal optimization classification, and validates results using SVM models. Experimental results demonstrate that the proposed feature selection and elimination algorithm effectively detects unknown XSS attacks with favorable efficiency. Future work

will focus on further feature optimization. This research is primarily experimental and requires continuous improvement and refinement for practical detection applications.

References

[1] Garg A, Singh S. A review on Web application security vulnerabilities. Internation Journal, 2013, 3(1): 222-226.
[2] Antunes N, Vieira M. Defending against Web application vnlnerabilities. Computer, 2012, 45(2): 66-72.
[3] Melbourne J, Jorm D. Penetration testing for Web applications. Proc of IEEE Symposium on Security and Privacy, 2014: 256-263.
[4] Huang Yaowen, Huang Shihkun, Lin T S P, et al. Web application security assessment by fault injection and behavior monitoring. Proc of the 12th International Conference on World Wide Web, 2003: 148-159.
[5] Almgren M, Debar H, Dacier M, et al. A lightweight tool for detecting Web server attacks. Proc of Netword and Distributed Systems Security, 2000.
[6] Williams J, Wichers D. OWASP top 10, the ten most critical Web application security risks. New York: The Open Web Application Security Project, 2012.
[7] Shar L K, Tan H B K. Auto-mated removal of cross site scripting vulnerabilities in Web application. Infromation and Software Technolohy, 2012, 54(5): 467-478.
[8] Guo X, Jin S, Zhang Y. XSS vulnerability detection using optimized attack vector repertory. Proc of IEEE International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, 2015: 29-36.
[9] Vishnu B A, Jevitha K P. Prediction of cross-site scripting attack using machine learning algorithms. Proc of International Conference on Interdisciplinary Advances in Applied Computing. New York: ACM Press, 2015.
[10] Keerthi S S, Shevade S K, Bhattacharyya C, et al. Improvements to platt' s SMO algorithm for SVM classifier design. Neural Computation, 2001, 13(3): 637-649.
[11] Joachims T. Text categorization with support vector machines: learning with many relevant features. Proc of Machine Learning, 1998: 137-142.
[12] Zhou X, Tuck D P. MSVM-RFE: extensions of SVM-RFE for multiclass gene selection on DNA microarray data. Bioinformatics, 2007, 23(9).
[13] Platt J. Sequential minimal optimization: a fast algorithm for training support vector machines. Advances in Kernel Methods-Support Vector Learning, 1998.
[14] Hearst M A, Dumais S T, Osuna E, et al. Support vector machines. IEEE Intelligent Systems and Their Applications, 1998, 13(4): 18-28.
[15] Chen Y W, Lin C J. Combining SVMs with various feature selection strategies. Studies in Fuzziness & Soft Computing, 2005, 207: 315-324.
[16] Powers D M. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. Journal of Machine Learning Technologies, 2011, 2(1): 37-63.

[17] XSSED. (2017-06). http://xssed.com.
[18] XSS (cross site scripting) cheat sheet. (2017-06). http://ha.ckers.org/xssAttacks.xml.
[19] exploit-db. (2017-06). http://www.exploit-b.co/webapps.
[20] 甘俊英, 张有为. 一种基于奇异值特征的神经网络人脸识别新途径. 电子学报, 2004, 32(1): 170-173.
[21] Han J, Pei J, Kamber M, et al. Data mining: concepts and techniques.
[22] Witten I H, Frank E, Hall M A, et al. Data mining: practical machine learning tools and techniques, 2016.
[23] 段宏湘, 张秋余, 张墨逸, 等. 基于归一化互信息的 FCBF 特征选择算法. 华中科技大学学报: 自然科学版, 2017, 45(1): 52-56.
[24] Kim S, D' Haro L F, Banchs R E, et al. The fourth dialog state tracking challenge. Dialogues with Social Robots, 2017: 435-449.

Note: Figure translations are in progress. See original paper for figures.