Reduction of SQL Injection & XSS Attacks Using IPAAS

1Y.Rajalakshmi, 2Mrs D S Bhavani
PG Scholar, Mahatma Gandhi Institute of Technology, Gandipet, Hyderabad (1)
Under the guidance of
Asst. Professor, Mahatma Gandhi Institute of Technology, Gandipet, Hyderabad (2)

Abstract: —
Network security is a main challenge now—a-days. There are different types of attacks in network. SQL Injection attacks are the most common attacks. Statistical survey says that most of the web sites which collaborate with the database are decumbent to SQL Injection or Cross Site scripting attacks. Detecting these attacks is known as anomaly detection. There is a lot anomaly detection techniques are widely used in a variety of applications, e.g., computer networks, security systems, etc. This paper describes and analyzes an approach to anomaly detection using proximity graphs and the PageRank algorithm. Most of the dynamic websites maintains databases; due to the design drawbacks of the database of a website there is a possibility of SQL injection attacks. The state-of-art web application input validation techniques fails to identify the proper SQL/XSS. The systems fail while processing HTTP parameter pollution attacks. Cross-site Scripting (XSS) has become prevalent type one of the most privacy vulnerabilities. Where the sake for the vulnerability primarily depends upon on the server side, the normal exploitation is within the user’s web browser which was affected on the client-side by attacks. Therefore, an operator of a web application has evidence which was limited of XSS issues. IPAAS is not able to protect against all kind of XSS and SQL injection attacks. However, our experiments show that IPAAS is a simple and effective solution that can greatly improve the security of web applications. Our technique automatically and transparently applies input validates during the development phase of a web applications. Therefore, IPAAS helps developers that are unaware of web application security issues to write more secure applications.

I. INTRODUCTION
SQL Injection attack is a web application vulnerability that occurs because of improper validations at the server side. Famous and Open Source Intrusion Detection System SNORT is providing detection logics not more than twenty. By these logics we can detect upto 20-40 sql injection attacks.

Anomaly detection, also known as outlier detection, refers to the problem of discovering data points or patterns in a given dataset that do not conform to some normal behavior. Anomaly detection techniques are applied in a variety of domains, including credit card fraud prevention, financial turbulence detection, virus or system intrusion
discovery, and network monitoring, to name a few. For a broad review of different anomaly detection approaches and techniques. We can view anomaly detection as a binary classification problem, with one class being anomalous and the other normal. In the classic supervised learning literature, labeled training data from both classes are required for the construction of a classifier. However, anomaly detection is different from traditional classification problems. While the latter usually deal with the case where both classes are of relatively equal size, this is not the case in anomaly detection. Since anomalies, by definition, deviate from the normal pattern, they usually represent situations where something goes wrong with the system (e.g., a malfunction, misuse, or malevolent behavior), and thus they are rarely observed. It is often impractical to collect sufficient observations to learn the anomalous pattern accurately. Moreover, manual labeling each data point is time consuming and error prone, and when the data is difficult to visualize or interpret it may not even be possible for a human to identify all anomalies. Therefore, although the supervised approach is well defined and thoroughly investigated, it is not a always appropriate for practical use. For this reason, we focus on unsupervised approaches.


II. PROBLEM STATEMENT
The existing systems work on IP log analysis. The paper is on graph based detection. The normal user profile is represented as a graph with document as a node. In the testing phase the current user profile is represented as a graph path.

Vulnerability research or response teams and most of the hackers or crackers participate for their fame and name. This site provides a separate category called web apps. In this category we can see the website hacked details. Currently this site is publishing 100 to 200 POC for every month. In the anomaly detection literature, it is quite common to assume that observations come from one of two distinct classes: one represented by a nominal distribution p(x) and the other by an anomalous distribution µ(x). We observe independent and identically distributed random measurements

\[ X_i \sim Q(x) = (1-\pi)p(x) + \pi\mu(x), \quad (1) \]

Where \( \pi \) is the prior probability of a particular observation coming from the anomalous distribution. Our task is to assign a label, either nominal or anomalous, to each measurement \( x_i \), possibly along with some confidence levels or rankings. In this work, we follow the assumption that all the observations are from the mixed distribution \( Q(x) \). The only quantities available are the measurements \( x_i \) themselves. No additional training data or label information is given, and either the nominal distribution \( p(x) \) or the prior probability \( \pi \) in Eq. (1) is provided. The
anomalous distribution $\mu(x)$ is assumed to be uniform, which is a natural choice when no other information is available. Additionally, this assumption leads to a nice reduction to the Neyman-Pearson test as we see next. In MV set approaches, an observation $x_i$ is declared to be an anomaly if it falls outside of a particular level set of $Q(x)$, i.e.,
$$x_i \in x \mid Q(x) \leq \lambda$$  \hspace{1cm} (2)
with $\pi$ being a prescribed threshold. It turns out that this criterion is identical to the Neyman-Pearson test, under the assumption that $\mu(x)$ is uniform over the measurement space.

To show this, combining Eq. (1) and Eq. (2), we have
\begin{align}
(1-\pi) p(x) + \pi \mu(x) &\leq \lambda \hspace{1cm} (3) \\
P(x)/\mu(x) =\lambda - \pi \mu(x)/(1-\pi) \mu(x) &\hspace{1cm} (4)
\end{align}
Which is the likelihood ratio between the two distributions? Note that the right hand side of the last inequality remains constant under the assumption that $\mu(x)$ is uniform over $x$. If, somehow, $p(x)$ can be estimated from the data, then we have all the information needed to compute density levels, to perform hypothesis testing, or to make other statistical arguments.

Consequently, some form of density estimation (e.g., kernel density estimation) seems to be a natural prerequisite for our task. However, kernel density estimation is itself an unnecessary intermediate step which estimates the continuous density for the whole data domain from discrete points, after which one level set parameter is calculated for each data point. The quantities we are actually interested in are the properties of the discrete observations, not the continuous space around them. As a result, specifying the full distribution throughout the whole space is inessential, introducing computational burden while accumulating estimation errors. The approach proposed in this paper circumvents density estimation through the use of an alternative graph-based approach.

### III SYSTEM DEVELOPMENT

**Access Log Parsing:** The user accesses are stored in the access log file. The files cannot be used for direct comparison. The file is preprocessed to identify Client IP, Request and Referrer from each user access log.

**Document Matrix:** The module identifies the access frequency for each document. It can be calculated as $\text{No. Of. Hits for a page per user}/ \text{Total Number of Logs}$. The value always in between 0 to 1. Training time Document Matrix represents the standard user access behavior.

**User Request Access**

The module identifies the user requested (URI). It also identifies the referrer URI. The user profile is stored for further processing.

**Document Matrix**

For every fixed interval of time, the user-profiles are processed for calculating the DM. Each individual user DM prepared. The DM rank indicates the document rank.

**Anomaly Detection**

User DM is cross compared with the training time DM. If any URI crosses or under flows the Training Time DM for a predefined threshold. The user is treated as an anomalous user. The anomalous users are reported to the administrator.

**Administration Interface**

The system monitors the anomalous activity. The anomalous behavior of any user is reported to administrator. It allows login, view the anomalous activity.

We implement the IPAAS approach for transparently learning types for web application parameters, and automatically applying robust validates for these parameters at runtime.

### IV. RELATED WORK

The standard approach in unsupervised statistical anomaly detection has been to assume that the data are drawn from a mixture of outlier and nominal distributions, and to estimate level sets of the nominal density. Schölkopf et al
propose the one-class support vector machine (OCSVM) to learn the classification boundary where only nominal training data are available. Scott and Nowak extend the Neyman-Pearson hypothesis testing framework to general supervised learning problems. Based on this extension, they derive a decision region using minimum volume (MV) sets in, providing false alarm control. Later, Scott and Kolaczyk generalize this hypothesis testing framework to the unsupervised case, where measurements are no longer assumed to come from the nominal distribution alone. Meanwhile, they incorporate a multiple testing framework, where the false discovery rate is controlled rather than false alarm errors. Hero introduces geometric entropy minimization to a extract minimal set covering the training samples while also ensuring false alarm guarantees. All of the methods mentioned above involve intensive computation, which is undesirable especially for large, high-dimensional data. We address this problem by taking an alternative graph-based approach. Another line of previous work is based on forming a graph from the data using the distances between data points. For example, a k-nearest neighbor (kNN) graph is constructed first, and then the distances from each data point to its kth nearest neighbour are used to identify anomalies. These distances are ranked in descending order, and either a threshold is applied or the top m candidates are declared anomalous. Breunig et al. define a related quantity called local outlier factor, which is a degree depending on how isolated one data point is with respect to the surrounding neighborhood, to better accommodate heteroscedastic data sources. Pokrajac et al. extend the local outlier factor approach in an incremental online fashion. Zhao and Saligrama propose a non-parametric anomaly detection algorithm based on kNN graphs trained using only nominal data points, which provides optimal false alarm control asymptotically. Our work is motivated by both directions mentioned above. We combine the graph approach together with random walk models, providing false alarm controls in an asymptotic sense. We note that we are not the first to use random walks or the PageRank algorithm for anomaly detection. Janeja and Atluri apply random walk models to detect anomalous spatial area regions in graphs where, in contrast to conventional scan-statistic methods, a regular-shaped scan window (e.g., a rectangle) is no longer required. He et al. propose a graph-based anomaly detection algorithm in an active learning setting, where the density information is used to reduce the number of inquiries made to the oracle; their algorithm builds on earlier work which uses graph-based methods for density estimation. Cheng et al. exploit random walks for finding anomalies in time sequences. Sun et al. also investigate anomalous patterns using a PageRank-like method. However, they focus mainly on bipartite graphs, while we are discussing much more general distributions and graphs. Researches explored an XSS detection mechanism which was same to our reflected detector. With the help of a server-side proxy entering parameters are tested for HTML markup was contained. If that parameter can be found, the HTTP response is tested if the same HTML markup can be discovered in the HTML content response. When compared our approach contains lot shortcomings. Depend upon HTML matching approach is not accurate, as it was unsuccessful to find attribute- and in-script -injections. In future, not like our technique, the detector which was proposed also does not contains transformation-processes, like character- removal or encoding filters that may change the entering parameters before outgoing HTML their reflection.

V. CONCLUSION

A lot technique are failed to identify the proper SQL/XSS Vulnerabilities accurately because of the systems correctness of sanity checking capability, proper placement of evaluators on the applications. We propose a framework
for anomaly detection using proximity graphs and the Page-Rank algorithm. Various parameter selection, time complexity guarantees and possible extensions are discussed and investigated. One straightforward extension is to formalize the problem of semi-supervised anomaly detection, when partial labels are available. The label information can be adapted into our framework without difficulty by changing the teleport vector $t$ accordingly in a more deliberate way. Another direction is to make the framework online. At this stage, our algorithm operates in a batch mode. Given a set of observations, after announcing the potential anomalies once, the algorithm terminates. However, in practice, it is quite common for successive measurements to come incrementally as time passes by. Once a new observation is available, we do not want to run the whole algorithm from start again. The time complexity of our framework has already been shown to be $O(n^2)$, which is not desirable in the online fashion. We are aiming to adapt our approach to update the model in a much faster way.

Moreover, given measurements in $\mathbb{R}^d$, we use all the dimensions instead of only a subset to compute the full dimension distance. This is to say, if our algorithm produces meaningful results, all dimensions are assumed to contribute useful information for our anomaly detection task. However, in reality, especially in high dimension cases, not all of them are helpful. The inclusion of noisy dimensions may even hurt the performance. Therefore, it will be better if our framework has some feature selection ability support built in, to filter out those unwanted dimensions.

REFERENCES


First Author: Y. Rajalakshmi has received her B.Tech degree in computer science from St. Mary’s College Of Engineering And Technology, Deshmukhi, Hyderabad in 2010 and pursuing her masters in computer networks and Information security from Mahatma Gandhi institute of technology, Gandipet, Hyderabad. Her research interested areas are Computer Networks and Network Security.

Under the Guidance of: Mrs. D.S. Bhavani received M.Tech Computer Science and Engineering. Working as Assistant Professor, having 7 years of teaching experience. Her research areas include Data mining applications in image processing, Medical image processing.