RPAD: Rule based Pattern Discovery for Input Type Validation Vulnerabilities Detection & Prevention of HTTP Requests

S. Venkatramulu
Associate Professor Computer Science and Engineering Department,
Kakatiya Institute of Technology and Science, Warangal, Telangana, 506015, India.

Orcid: 0000-0002-2158-328X

Dr. C.V. Guru Rao
Professor, Computer Science and Engineering Department,
S.R Engineering College, Warangal, Telangana, 506001, India.

Abstract
The internet access by web browsers is most vulnerable, since the browsers itself can adapt as attacking tool. The traditional way of dealing the influence of these vulnerabilities such as SQL Injections and XSS is the code verification by syntax analyzers. Since the syntax analyzers deployed and executed in server environment, the process overhead on servers is the major constraint observed, which is due to the online verification of dynamic SQL statements generated by web enabled applications and injected into server-side applications. The other major constraint of these syntax analyzers programming language dependency. In order to this here in this paper we proposed a Rule based Pattern Discovery (RPAD) for Input Type Validation Vulnerabilities Detection and Prevention of HTTP Requests. The core objective of the RPAD is to execute as network level IDS and not to rely on syntax analyzers, hence the limits such as programming language dependency and server level process overhead observed in existing benchmarking models are least significant for RPAD. The other competency of RPAD is minimal sanity checks. The experimental study was conducted on several benchmarking CVE entries published by NIST. The combination of 2783 attack patterns extracted from CVE entries of the NIST and 512 normal patterns extracted from 7 real time web applications were used to evince the performance of the RPAD. The empirical study evinced that the RPAD prediction accuracy is around 93%.

Keyword: Web application, SQL injection Attack, XSS Attack, IPAAS

INTRODUCTION
The web browsers those renders the interfaces of the Web applications are highly vulnerable, hence the security of these applications is critical due to the confidentiality and sensitivity of the data involved. The act of preventing web applications from the security and sensitivity breaches raised due to these vulnerabilities is usually relies on engines such as firewalls and Intrusion Detection Systems, which are assessed and approved by the NSS Labs [1], ICSA Labs [2] and other communities such as OWASP [3]. These communities govern the web application security and privilege guidelines. The top 3 vulnerabilities out of the vulnerabilities published by these organizations are SQL Injection, Broken Authentication and XSS. The statistics of the vulnerabilities published by these organizations evincing that an average of 2 new vulnerabilities observed for each day in last five years log of CVE [4].

Due to more customer data going online by adapting to online banking or fund transfer practices, users’ accounts and other information have become vulnerable to fraud and other attacks. Also, hackers in recent years are increasingly targeting web applications, since most networks are closely monitored through Intrusion Detection Systems (IDS) and firewalls. Therefore, the web application layer needs to be secured from unauthorized users by building across the software development lifecycle security mechanism [5]. This ensures that it is not an afterthought issue, only considered in the end [6] as in many software development processes, where as a result, attackers continue to explore areas of vulnerability to undermine the integrity of applications. In recognition of this problem, developers have to incorporate security during the development in order to produce vulnerability free software systems, since the existence of flaws at the design or coding phase of the development lifecycle can open web applications to a wide range of attacks [7].

According to Top-10 vulnerabilities published by OWASP [3] for the year 2013 are Injection, Broken Authentication and XSS are the Top-3 vulnerabilities. National Vulnerability Database [8] shows that 7092 SQL Injection attacks, 8254 XSS attacks out of 61961 total attacks published vulnerabilities as on June-2014.Exploit-db statistics shows that every day, at least two new vulnerabilities are getting published for the last five years out of 29871 total attacks published. Third generation web applications are mostly depending on the parameter and its values. Dynamic web applications accept user inputs through parameters (i.e. request parameters). The public web applications should handle exploits and normal requests.
Major threat to the dynamic web applications are SQL Injection and Cross Site Scripting attacks in addition to the buffer overflow attacks. There are several defense techniques already proposed by several researchers.

The current web application development falls in to the third and fourth generation strategy [9], those accepts user inputs through request parameters (name and value pairs). Hence it is obvious to sense the high frequency of SQL Injection and Cross Site Scripting attacks along the side of buffer overflow attacks. This clarifies the significance of the vulnerable detection and prevention strategies.

In recent literature, a combined approach called IPAAS (Input Parameter Analysis System) [10] for vulnerability detection is found. The IPAAS embeds the host and network based detection systems to secure web applications against XSS and SQL injection attacks using input validation. The key insight behind IPAAS is to automatically and transparently augment otherwise insecure web application development environments with input validators. Though the model succeeds to minimize the server load but still influenced by the other two constraints “programming language dependency” and “imprecise code analysis”.

Henceforth, this manuscript proposes a novel vulnerable detection strategy called “Rule based Pattern Discovery (RPAD) for Input Type Validation Vulnerabilities Detection & Prevention of HTTP Requests” that defines rules to identify the association between input types and values. Further these rules will be used to identify the vulnerable http requests, in particular SQL injection and cross site scripting vulnerabilities. The process of the proposed model called RPAD fall into two phases called (i) training and (ii) testing. The tasks included in training phase are (a) capturing the packets involved in each web application request, (b) filters the request packets from data packets (c) extracts the http requests from request packets, (d) extracts input type and value associations from the key value pairs of the http requests, (e) rules formation based on in input type and value associations and labels (normal or attack)given to respective training records. Similarly, the testing phase is also the combination of set of tasks and among them the first 4 tasks are similar to the tasks (a), (b), (c) and (d) of training phase in same hierarchy. The input type and value association rule analysis is the next task of the testing phase, which is unique and specific to this phase. The detailed exploration of the proposal can be found in following sections.

The next section (section II) explores the contemporary research, contributed in recent literature that includes the detailed exploration of the Intrusion detection systems and their classification, briefing of the existing system’s strengths and limits

RELATED WORK

The Network intrusion detection system (IDS) is the core strategy that used currently by many of the web applications to prevent intruded requests [11]. The IDS detect intruders either using intrusion signatures those published in public communities or by the knowledge acquired from anomalies training. The signature based IDS is unsuccessful to detect new vulnerabilities, if the signatures of these vulnerabilities are not available to IDS. SNORT [12] is a good example for signature based IDS. The anomaly based IDS capable to identify the vulnerabilities based on the pattern of the feature noticed during the training phase. Hence the detection accuracy is significant even in the case of new vulnerabilities. The core limit of the anomaly based IDS is the complexity of the training, which is proportionate to the number features considered to train. Hence it is obvious to conclude that if vulnerabilities are not fixed then anomaly based IDS [13][14][15][16][17] are having detection accuracy that compared to signature based IDS. Anomaly detection approaches can be scalable if number of features used in training are less.

Host based detection [18] is another kind of strategy to detect vulnerabilities in web applications. The core constraints of this strategy are

(i) Process overhead at host servers (web servers) since the detection strategy is embeds with the target web application that deployed in host server. This limits the server performance at peaks of server load.

(ii) Limited to a specific programming language since this strategy uses syntax analyzers of specific programming language (The syntax analyzer defined for one language (PHP syntax analyzer) is not compatible to other language (java, asp…)).

The example of the host based detection strategy is WASP [19] that uses a syntax analyzer to analyze application code and request parameters, further validates dynamically prepared SQL statement with trust marking strategy.

Host based input type checking strategy another strategy that also uses on the code analysis. The constraints observed for host based detection strategies are also having influence on these host based input type checking strategies. The detection accuracy that compared to host based detection is evinced. The code analysis engines used for syntax analyzers not accurate since the strategy of writing a program differs from one programmer to other. Hence the substantial performance is not evinced.

In order to this, it is quite obvious to conclude that the syntax analyzer based Vulnerability testing strategies[14],[20], [21], [22]; [23], [24], [25], [26], [27], [28], [29] found in literature are not robust and scalable due to process complexity, programming language dependency and limited accuracy in
vulnerable detection. In recent literature, a combined approach called IPAAS [10] is found that embeds the host and network based detection systems. Though the model succeeds to minimize the server load but still influenced by the other two constraints “programming language dependency” and “imprecise code analysis”.

Henceforth this manuscript explored an anomaly based network level detection strategy called Rule based Pattern Discovery (RPAD) for Input Type Validation to discover vulnerabilities from HTTP requests. Since the model is at network level, the process load at host server is obsolete and this model is not using any syntax analyzers, hence the constraints called programming language dependency and imprecise code analysis also not evinced in RPAD.

The next section explores the formation of the rules for input type and value associations in the given labeled records under training phase and using these rules to label the input records given in testing phase under proposed model called RPAD.

**Rule Based Pattern Discovery (RPAD) For Input Type Validation Vulnerabilities**

The proposed RPAD is an anomaly based network level system for vulnerability detection in http requests. The RPAD discovers the patterns of parameter types and parameter values at network level. The code analyzers are not used; hence it is programming language independent and can be connected dynamically and used to monitor vulnerabilities in http requests targeted to any web application. The analysis of http requests for possible vulnerabilities is done by the learned patterns of the request parameter values and their types.

The main objective of the RPAD is rule based pattern discovery of the values and input types of the request parameters. In order to this initially builds dynamic rules from vulnerable prone and normal http requests which is in the aim of tracking the SQL Injection and Cross Site Scripting Attacks. Further these rules will be used to discover the patterns of the post request parameter values and their types. The overall process of the RPAD is a dual fold strategy of training and testing. RPAD initially trains on the discovered patterns and further uses this knowledge to identify the vulnerabilities in given http requests. The overall flow of the RPAD is visualized in fig 1.

The RPAD initially generates the rules by the conflicts of the input parameter values and types found between vulnerable prone and normal http requests. Further these rules will be used to discover the patterns of the input values and their types.

**Process Flow of the RPAD**

The training phase is initiated first, that captures the packets from all labeled records given as input in this phase. Further filters the packets as request and data packets, then filters the request packets as http requests and other. The http request packets further processed and extracts the key value pairs involved, then generates the association of input types and values from these key value pairs. Further these input type and value associations and record labels (vulnerable or normal) are together used to form the rules as follows:

Let \( VR = \{ v_1, v_2, \ldots, v_{|VR|} \} \) and \( NR = \{ n_1, n_2, \ldots, n_{|NR|} \} \) be the set of vulnerable http requests and fair http requests respectively.

Further the request parameters of found in each request of the set \( VR \) and set \( NR \) will be extracted and represented as respective sets:

\[
P_{VR} = \{ p_1([i],[r]), p_2([i],[r]), \ldots, p_{|P_{VR}|}([i],[r]) \} \text{ and } P_{NR} = \{ p_1([i],[r]), p_2([i],[r]), \ldots, p_{|P_{NR}|}([i],[r]) \}.
\]

Here \( p_i([i],[r]) \) represents a request parameter with set of input values \( [i] \) and their respective types \( [r] \) observed.

The http request parser such as magpie [30] can be used to parse the http requests to extract request parameter name value pairs.

Further a set of rules \( R = \{ r_1, r_2, \ldots, r_{|R|} \} \) will be formed, those reflect the possible values and respective types used for request parameters of the http requests found in \( VR \) and \( NR \).
Further these rules will be used to discover the patterns of the request parameters by their respective input values and types. As an example of the pattern discovered can be as follows:

If parameter $p_i$ of the vulnerable prone requests is bounded to rule $r_s$ then the parameter $p_j$ of the same request bounds to rule $r_q$. In a gist, this pattern explores that a request $rq$ with parameters $p_i$ and $p_j$ those bounds to respective rules $r_s$ and $r_q$ indicates that request $rq$ is vulnerable prone. This is a simple example of pattern discovery for 2 parameters. In practice, it can be for $n$ parameters.

**Vulnerability Detection Strategy**

The proposed anomaly IDS called RPAD, builds the rules from the given vulnerable prone and normal http requests (see sec A). Based on these rules, further the request parameter patterns will be discovered. The parameters justify the specific rules in majority of the given requests considered as a pattern.

Upon the arrival of a new request, the RPAD parses the request and extract the request parameter pairs. Then identifies the scope of patterns discovered during training phase. If observed patterns are found to be the identical to the patterns discovered for vulnerable prone requests in training phase then the input request confirmed to be as an attack, if not the request parameters will be cross checked with patterns discovered in training phase for normal requests, if similarity found then request will be labeled as normal and send to host server. In other case where request parameters are compatible to neither vulnerable prone patterns nor normal patterns then the request will be labeled as suspicious and alerts the administrator. Further, based on the administrative decision, the patterns discovered from the new request parameters will be added to RPAD pattern repository.

**Programmatic Implementation**

The programmatic implementation was done using Perl scripts. The overall proposal is developed in different modules and they are (i) Dataset Preprocessing, (ii) training from the labeled set, (iii) testing the unlabeled set, (iv) and assessing statistical metrics.

**Dataset Preprocessing**

A Perl script that loads given dataset in flat file format into application environment and segments the data as vulnerable prone and normal records by the label given. Then the request parameters are extracted from the respective records of vulnerable prone and normal, which is done by using magpie [30] library functions. Further these request parameters are used to form request parameter set for respective normal and vulnerable prone records, such that each resultant set consists set of records and each record represents parameter name, all possible values and their types observed. Further these respective request parameter sets of vulnerable prone and normal records are used as input for next module that performs training.

**Training Phase**

The given respective request parameter sets of vulnerable prone and normal records will be analyzed by a Perl script that prepares the set of rules to define the association between request parameter values and types towards vulnerable proneness and normal activity. These rules further given as input to the next module, where testing of the unlabeled records, which are the mix of vulnerable prone and normal records.

**Vulnerability Testing Phase**

In this module, a Perl script was built that analyzes the given test records against the rules extracted from the training phase module. This script extracts the request parameters of each test input records in sequence and verifies against the rules formed in training module and concludes the state of the record is vulnerable prone or not. Further these predictions will be sent as input to the statistical analysis module.

**Statistical analysis module**

This module is implemented by a Perl script that segments the true positives ($t_+$) (true prediction of vulnerable records), true negatives ($t_-$) (true prediction of normal records), false positives ($f_+$) (prediction of normal records as vulnerable) and false negatives ($f_-$) (prediction of vulnerable records as normal), which done by comparing the actual labels of the test records with predicted labels. Further these $t_+, t_-, f_+, f_-$ will be used to assess the metrics called precision, recall, accuracy, sensitivity, specificity and F-measure.

**EXPERIMENTAL STUDY AND PERFORMANCE ANALYSIS**

**Dataset and Experimental setup**

The empirical study was done by applying proposed model on benchmarking vulnerability reports stored in the Common Vulnerabilities and Exposures (CVE) database hosted by NIST [8]. The seven SQL/XSS allowed real time web applications that evinced 2783 attack patterns and 512 normal patterns (total 3295 patterns) with all possible parameters were used to train and test the devised rule based model. The 70%
of the records were used for training and 30% were used to test the significance of the proposed model. In training phase IDS extracted 273 rules in the form of regular expressions. Afterwards, these discovered rules are used to explore the input given in testing phase. The significance of the model estimated using the statistical metrics [31] called precision, sensitivity, specificity and accuracy. The input parameters of the experimental study were explored in table 1.

A computer with i5 processor, 4GB ram and Nvidia 4GB graphics card [32] used. The implementation was done in PERL, since the assessment metrics, computational and resource complexity also included in performance analysis.

The performance of the proposed model was explored by statistical metrics that compared to the benchmarking model called Input Parameter Analysis System (IPAAS) [10] for Preventing Input Validation Vulnerabilities in Web Application through IPAAS (see table 2).

**Performance Analysis**

The statistical metrics were used to assess the prediction accuracy of the RPAD and IPAAS (see table 2). According to these metrics the accuracy of RPAD and IPAAS are 93% (as accuracy observed is 0.928) and 80% (as accuracy observed 0.801) respectively. The robustness of the model is highly since the recall and sensitivity observed for RPAD is high that compared to IPAAS (see table 2).

The process completion time observed for proposed RPAD is linear and less than the completion time (see fig 2) observed for benchmarking model called IPAAS. The resource utilization observed for proposal is linear and minimal (see fig 3) that compared to IPAAS. The visualization of the completion time observed for RPAD and IPAAS can be found in fig 2, which is a line chart between quantity of input requests given (x-axis) and process time taken in seconds (y-axis). Similarly, the memory usage observed is visualized by fig 3, which is also a line chart between quantity of input requests given (x-axis) and memory used in kilo bytes (y-axis).

**Table 1**: Input parameters used for experimental study

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real time web applications used</td>
<td>7</td>
</tr>
<tr>
<td>Number of attack patterns</td>
<td>2783</td>
</tr>
<tr>
<td>Number of normal patterns</td>
<td>512</td>
</tr>
<tr>
<td>Attack Patterns used for Training</td>
<td>1948</td>
</tr>
<tr>
<td>Normal Patterns used for training</td>
<td>358</td>
</tr>
<tr>
<td>Attack Patterns Used for Testing</td>
<td>835</td>
</tr>
<tr>
<td>Normal Patterns used for testing</td>
<td>154</td>
</tr>
<tr>
<td>Number of IDS rules generated</td>
<td>273</td>
</tr>
</tbody>
</table>

**Table 2**: The prediction statistics of the IPAAS [10] and RPAD

<table>
<thead>
<tr>
<th>Metrics</th>
<th>IPAAS [24]</th>
<th>RPAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attacks</td>
<td>780</td>
<td>840</td>
</tr>
<tr>
<td>Normal</td>
<td>209</td>
<td>149</td>
</tr>
<tr>
<td>True Positives</td>
<td>682</td>
<td>802</td>
</tr>
<tr>
<td>True Negatives</td>
<td>111</td>
<td>116</td>
</tr>
<tr>
<td>False Positives</td>
<td>98</td>
<td>38</td>
</tr>
<tr>
<td>False Negative</td>
<td>98</td>
<td>33</td>
</tr>
<tr>
<td>Precision</td>
<td>0.874358974</td>
<td>0.954761905</td>
</tr>
<tr>
<td>Recall</td>
<td>0.874358974</td>
<td>0.960479042</td>
</tr>
<tr>
<td>Accuracy</td>
<td>0.80182002</td>
<td>0.928210313</td>
</tr>
<tr>
<td>Sensitivity</td>
<td>0.874358974</td>
<td>0.960479042</td>
</tr>
<tr>
<td>Specificity</td>
<td>0.531100478</td>
<td>0.753246753</td>
</tr>
<tr>
<td>F-measure</td>
<td>0.874358974</td>
<td>0.95761194</td>
</tr>
</tbody>
</table>

**Figure 2**: Process Completion time observed for RPAD and IPAAS at divergent input ratios

**Figure 3**: Memory Usage observed for RPAD and IPAAS at divergent input ratios
CONCLUSION
This article explored an anomaly based vulnerable detection system for web applications is proposed. The model defined here is a Rule based Pattern Discovery (RPAD) for vulnerable http request detection and prevention. The system is able to detect divergent vulnerabilities causes SQL Injections, Cross Site Scripting, Remote File Inclusions, and Local File Inclusions. The core objective of the RPAD is to avoid the constraints such as process overhead at host servers, programming language dependency due to syntax analyzers, which were significantly influencing the performance of the existing benchmarking models. The experimental study was done on CVE entries published by NIST [8]. The results indicating the performance advantage and detection accuracy that compared to a benchmarking strategy called IPASS [10] that found in recent literature. The vulnerable request prediction accuracy is substantial with minimal false alarming. The outcomes of the RPAD motivates the future research to define novel heuristic scales through benchmarking machine learning strategies.

REFERENCES


[27] Livshits BV, Lam MS. Finding security errors in java programs with static analysis (tech report).


