DE-PRE Tool for Detection and Prevention from
Input Validation Attacks on Website

Prathamesh P. Churi
Assistant Professor
Department of Computer Engineering
SVKM’s NMIMS Mukesh Patel School of Technology Management and Engineering
Mumbai, India

Kamal Mistry
Assistant Professor
Department of Computer Engineering
SVKM’s NMIMS Mukesh Patel School of Technology Management and Engineering
Mumbai, India

ABSTRACT
In most cases, usually, when a web application or a website is fully developed it is released over the internet and if not provided with proper security then it is susceptible to input validation attacks; specifically SQLIA, XSS, Buffer Overflow. A website can be hacked within hours of its release on the internet. It is at this point the developing team realizes to consider security issues and to backtrack and repair. Applying security at this stage is a costly, time consuming process which would also include some software overhead. To avoid this and to prevent websites from input validation attacks we plan to impart security at the software development cycle itself such that the website is protected prior to its release considering the vulnerabilities of the same and the behaviour of the attacks and later it will not be under any threat by the specific input validation attacks. To implement this, we are designing a useful utility tool that imparts security at the software development cycle of any website that requires protection.

Keywords
Input Validation Attacks, SQLIA, XSS, BOF, Utility Tool, Prevention.

1. INTRODUCTION
Our aim is to develop a software utility tool that can be used as software development cycle of a website such that each page of the same can be imported in the software and tested for its vulnerabilities, becomes less susceptible and is prevented from major input validation attacks which are specifically SQL Injection (SQLIA), Cross-site Scripting (XSS) and Buffer Overflow (BOF). In short your website becomes hack-resilient.

The software follows two modules: Detection and Prevention module to make it effective to test each and every webpage of the website under development and to remove all the vulnerabilities such that the website is protected against input validation attacks once and when it is hosted or is live over the internet.
In proposed software we implement prevention using Mutation testing and have a graphical user interface that allows you to select the attack and its variants against which you want to protect your website. Selecting a specific attack, the next step lets you import your web page and test your webpage for the vulnerabilities it might contain for the specific attack to facilitate.

The particular vulnerabilities can be selected with the interface and the software creates a report for that specific attack and removes all the vulnerabilities [2, 3]. Further detection of attacks and IP address tracking is also featured by this software.

2. EXISTING INPUT VALIDATION ATTACKS
Papers Input Validation Attacks [4,10] are where an attacker intentionally sends unusual input in the hopes of confusing an application.

2.1 SQL Injection (SQLIA):
SQLIA kind of attacks occur when the attacker uses specifically crafted SQL questions or commands to execute malicious activities on the victim system. This weakness exists when there is no validation of input while a database query is made via the Internet. The worst thing is this can be done with only browser. Obviously, the first step for an SQL injection is to find a vulnerable target. Unfortunately, Web applications are also vulnerable to a variety of new security threats. SQL Injection Attacks (SQLIAs) are one of the most significant of such threats. SQLIAs have become increasingly frequent and pose very serious security risks because they can give attackers unrestricted access to the databases that underlie Web applications [8].

2.2 Cross Site Scripting (XSS):
Cross site scripting (also known as XSS) [5] occurs when a web application gathers malicious data from a user. The data is usually gathered in the form of a hyperlink which contains malicious content within it. The user will most likely click on this link from another website, instant message, or simply just reading a web board or email message. Usually the attacker will encode the malicious portion of the link to the site in HEX (or other encoding methods) so the request is less suspicious looking to the user when clicked on. After the data is collected by the web application, it creates an output page for the user containing the malicious data that was originally sent to it, but in a manner to take it appear as valid content from the website. Many popular guestbook and forum programs allow users to submit posts with html and java script embedded in them. Attacker can make use of these java scripts to steal cookies for session hijacking.

Copyright © 2017 Prathamesh P. Churi et al. This is an open-access article distributed under the terms of the Creative Commons Attribution License 4.0, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.
2.3 Buffer Overflow (BOF):
Buffer Overflow [6] attacks comes under Input Invalidation as well as sloppy programming or poor memory management category. In this paper we have concentrated on Buffer Overflow due to Input Invalidation. Measurably on web string overflow attack takes place. Let’s discuss on it. Generally Buffer Overflow Attack takes Place on C and C++ programs on Java it is somewhat difficult to do Buffer Overflow.

3. PREVIOUS RESEARCH
Please IBM reported that the cost to fix an error found after product release was 4 to 5 times as much as one uncovered during design, and up to 100 times more than one identified in the maintenance phase. Research by @Stake demonstrated that on average an organization caught only a quarter of its software security holes and had typically seven significant bugs within its enterprise software. Their findings verified that fixing the same defects during the testing phase cost around seven times less than after deployment. They concluded building security into software engineering at the design stage would net a 21% ROSI (Return on IT security Investment); waiting until the implementation stage would reduce that to 15% and at the testing stage, the ROSI would fall to 12%. Integrating security early into the application development lifecycle produces more secure, robust applications at a lower cost.

Random security is not enough. To make your application hack-resilient, you need a holistic and systematic approach to securing your network, host, and application. The responsibility spans phases and roles across the product life cycle [7,4].

In this project, main intention is to focuses on detection and prevention of Input Validation attacks like SQL Injection, Cross Site Scripting and Buffer Overflow by incorporating security in software development life cycle. We have introduced a novel approach of preclusion and uncovering of Input Validation Attacks. Moreover, existing tools do not have prevention and detection in one. Security cannot be incorporated at the end of program; Security should be involved in software development life cycle. Following this base line our tool will work in two phases testing and deployment. Prevention module will work in testing phase. Detection module is design for deployment phase in which security functions get called while writing code [9].

For testing above proposed tool we need web application. We found that there is necessity of secure web based conference management system in which many things are involve right from paper submission till registration of the paper where money transaction is involved. So we decided to construct secure web based conference management system using proposed tool.

3.1 Existing System and Literature Cited
AMNESIA, SQLGaurd, SQLCheck, SQLrand, TAUTOLOGYCheker, are some tools for SQL Attack detection but they do not provide prevention management with the same. XSS Me: XSS Me is a plug in for Mozilla Firefox that aids in cross site scripting testing, as mentioned it is a plugin not a lone tool to test each webpage.

<table>
<thead>
<tr>
<th>Sr. No</th>
<th>Title of the paper and authors</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Web Application Intrusion Detection System for Input Validation Attack [1]</td>
<td>- The paper presents Web Application Intrusion Detection System (WAIDS) - The approach is based on web application parameters which has identical structures and values - WAIDS derives a new intrusion detection method using generated profile from web request data in normal situation. By doing this, it is possible to reduce analysis time and false positives rate. - The results parameters and evaluation is significant.</td>
</tr>
<tr>
<td>2</td>
<td>HMM-Web: a framework for the detection of attacks against Web applications [11]</td>
<td>- Author propose a new formulation of query analysis through Hidden Markov Models (HMM) and show that HMM are effective in detecting a wide range of either known or unknown attacks on web applications - Experimental results on real world data, show the effectiveness of the proposed system in terms of very high detection rates and low false alarm rates.</td>
</tr>
<tr>
<td>3</td>
<td>SBSQLID: Securing Web Applications with Service Based SQL Injection Detection [12]</td>
<td>- This paper proposes a methodology for the detection of exploitations of SQL injection vulnerabilities - In this work, an independent Web Service is intended to generalize syntactic structure of SQL query and validate the user inputs. When the user submits the SQL query at the runtime, the query has to be parsed by the independent service for the correctness of the syntactic structure and user data. - This approach is to prevent all forms of SQL injections, independent of the target system, independent to platform and Backend DB server.</td>
</tr>
</tbody>
</table>
### 4. Working Prototype

Our software consists of two modules: Detection module and Prevention module.

#### 4.1 Detection Module

Detection module will work when user supplies input data to web application. This part of tool basically does input testing i.e. black box testing.

**Layer 1 (Removal of illegal character):** In the first stage the form information is scanned and illegal characters (mostly wild characters) are detected if any. This stage basically deals with the issue of Cross Site Scripting (XSS) and SQL Injection Attack (SQLIA). Example scripts are always written with < and > symbols. Piggybacking queries are separated with semicolon (;). Such suspicious characters can be filter out in this stage.

**Layer 2 (Validation through regular expression check):** In stage two the information in the form is compared with the set of valid regular expressions and evaluated. This stage deals with Buffer overflow criteria, XSS and SQLIA. Regular expression "^(\[w-\]+\[[w-\]+\]?\[w-\]+\[@\]?[\w\-]+[\w\-]+\]?\[w-\]+$" check for validity of email address likewise we can write regular expression for all inputs so any change made by user during input can be detected easily.

**Layer 3 (IP address tracking):** In stage three the form contents are matched with potentially harmful set of statements which are recognized and stored in a knowledge base, thus tackling the security threat called SQLIA and XSS. For example user can type tautology "or "1" = "1" in password filed. Password can be combination of special characters and symbols so we can’t put any restriction on password field. As it is a well know attack can be detected just by comparing it with list of suspicious input.

Main advantage of this phase is coding knowledge is not required only input values are sufficient for detection of attack.

In detection module what we want have multiple layers so that it can detect any kind of attack.

---

<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>•</strong> The primary focus of our research was to develop a reliable black-box vulnerability scanner for detecting SQLI vulnerability - SQLIVDT (SQL Injection Vulnerability Detection Tool).</td>
<td></td>
</tr>
<tr>
<td><strong>•</strong> The black-box approach is based on simulation of SQLI attacks against web applications. Thus, the scope of analysis is limited to HTTP responses and HTML pages received from the application server.</td>
<td></td>
</tr>
<tr>
<td><strong>•</strong> In order to achieve efficient SQLI vulnerability detection, an efficient algorithm for HTML page similarity detection is used. The proposed tool showed promising results as compared to six well-known web application scanners.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>5</th>
<th>Securing web applications with better “patches”: an architectural approach for systematic input validation with security patterns [14]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>•</strong> This paper examines the current ways of how input validation is conducted in major open-source projects and attempts to confirm the main source of the problem as these ad hoc responses to the input validation-related attacks such as SQL injection and cross-site scripting (XSS) attacks through a case study.</td>
<td></td>
</tr>
<tr>
<td><strong>•</strong> In addition, authors have proposed a more systematic security approach by promoting the adoption of proactive, architectural design based solutions to move away from current practice of chronic vulnerability-centric and reactive approaches.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>6</th>
<th>Websites Input Validation and Input Misuse Based Attacks [15]</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>•</strong> In this paper, we conducted an evaluation study of how much input validation is used by web-designers.</td>
<td></td>
</tr>
<tr>
<td><strong>•</strong> We used some of the web attacks that target improper input validations as indicators to show the quality of the input validation process for the evaluated websites.</td>
<td></td>
</tr>
<tr>
<td><strong>•</strong> Results showed that those types of attacks continue to be effective and serious methods.</td>
<td></td>
</tr>
</tbody>
</table>

Results showed also that there is a need for systematic and frequent evaluation for those websites to ensure that basic input validation guidelines are observed.

This paper includes the following contributions: An SQL-injection-vulnerability assessment of a dataset of several selected websites, and the proposal of a mutation, fault-based model to test websites for possible SQL injection vulnerabilities.
Mutation testing is a powerful method for finding errors in software programs. Mutation testing introduces faults into software by creating many different versions of the program. Each version has one very small change (which introduces a fault) compared to the actual implementation. The idea is to see if the test cases that were written can detect the new fault. There are different mutation operators such as RMWH stands for “remove where” and NEGC for „negate each of unit expression.

5. CONCLUSION

In this project, we presented detailed information of the Input validation attacks in which main focus was on SQLIA, XSS and Buffer Overflow. We discussed different types of SQLIA, XSS and BOF. We then stated need of security in software development life cycle. We demonstrated that the how attackers perform the attack on the web sites We discussed the design and implementation of the new hybrid tool that was developed for this project. We demonstrated that the tool provides a robust web-server security against input validation attacks. The tool contains two modules prevention and detection. These modules works is different phases of software development as the security is incorporated early in software development life cycle definitely it is beneficial.

Testing applications for SQL Injection, XSS and Buffer Overflow is being performed today using many different approaches. But the method used by us for prevention was Mutation-based testing because Mutation testing is the most powerful technique for the assessment and enhancement of tests. For detection we have used multilayer defense mechanism. Due to multiple layers it can catch any kind of attack on web-site. We have successfully developed an application aimed at detecting and preventing web vulnerabilities. The designed system will allow a web developer to determine various Application Security Risks and vulnerabilities. The system also makes the appropriate suggestions to ensure security of web applications against various threats and risks, thereby giving solutions to overcome them

6. ACKNOWLEDGMENT

The author wants to say thanks to his brother Mr. Rohan Chaudhari for his encouragement for this research.

7. REFERENCES


