Busted: A website vulnerabilities analysis and alert tool

Ujjwal Sinh
uijwalsinh96@gmail.com
Dayananda Sagar College of Engineering, Bengaluru, Karnataka

Suresh Kumar B.
sureshbrajpurohit@live.in
Dayananda Sagar College of Engineering, Bengaluru, Karnataka

Tejaswini D. Karlo
tejaswinikalro@gmail.com
Dayananda Sagar College of Engineering, Bengaluru, Karnataka

Bajarang
rathodbajarang@gmail.com
Dayananda Sagar College of Engineering, Bengaluru, Karnataka

A. M. Prasad
prasadssc@gmail.com
Dayananda Sagar College of Engineering, Bengaluru, Karnataka

ABSTRACT

The number of reported web application vulnerabilities is increasing dramatically. The most of vulnerabilities result from improper input validation. This paper presents extensions to the Tainted Mode model which allows intermodule vulnerabilities detection. Besides, this paper presents a new approach to vulnerability analysis which incorporates advantages of penetration testing and dynamic analysis henceforth, it is required to have secured services. The main concern is to provide security to end user to protect files or data from the unauthorized group. The web application is now used to perform an intelligent standard of collection of services, applications, information, infrastructure comprised of pools of computer, network, information, and storage resources. With these advantages storing a large amount of data including important information on websites motivates a highly accomplished hacker to hack the critical information also. Hence, security is considered a topmost issue in cloud computing. This paper aims at analyzing encryption algorithms based on factors such as cost, time, key length and block size. The analysis indicates that several algorithm and various tools help in improving security to save the data.

Keywords— Web application, Web security, Cyber security, Web security tools, Vulnerability analysis, Penetration testing, Dynamic analysis, Report generation

1. INTRODUCTION

In recent years, the globe Wide internet (WWW) has witnessed a staggering growth of the many on-line internet applications that are developed for meeting numerous purposes. Nowadays, nearly everybody connected with computer technology is somehow connected online. To serve this immense variety of users, nice volumes of knowledge are stored in internet application databases in several components. Timely, the consumer should move with the backend databases via the user interfaces for various tasks such as modification information, making queries, extracting information, etc. For these operations, the vogue interface plays an important role, the quality of that features a pleasant impact on the protection of the keep information inside the knowledge. An unsecured web application might enable crafted injection and malicious update on the backend information. This trend will cause numerous damages and thefts of trustworthy users’ sensitive knowledge by unauthorized users. Within the worst case, the assailter might gain full management over the net application and destroy or damage the system.

SQL Injection may be a kind of injection or attack in an exceeding internet application, during which the aggressor provides Structured Question Language (SQL) code to a user input box of an online type to gain unauthorized and unlimited access. The attacker’s input is transmitted into associate degree SQL question in much the way that it forms an SQL code. In fact, SQL Injection is classified as the top-10 net application vulnerabilities experienced by net applications consistent with OWASP (Open net Application Security Project). SQL Injection Vulnerabilities (SQLIV’s) unlocks entrance for hackers to explore and attack. Hence, they show a severe hazard for net application components. The main concept of SQLIVs is sort of straightforward and well-known insufficient validation of user input. To overcome such kind of vulnerabilities, several techniques are suggested like manual approach, machine-controlled approach; secure secret writing practices, static analysis, exploitation ready statements, and then forth. Though planned approaches have achieved their goals to some extent, SQL Injection Vulnerabilities in net applications stay as a significant concern among application developers of Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) discussed below.

The first approach is based on web application analysis from the user side, if the source code of an application is not available. The idea is to submit various malicious patterns (implementing, for example, SQL injection or cross-site
scripting attacks) into web application forms and to analyze its output thereafter. If any application errors are observed an assumption of possible vulnerability is made. This approach guarantees neither accuracy nor completeness of the obtained results.

The second approach is based on web application analysis from the server side, with the assumption that source code of the application is available. In this case, dynamic or static analysis techniques can be applied. A comprehensive survey of these techniques was made by Vigna et al. According to this survey several statements could be made:

The most common model of input validation vulnerabilities is the Tainted Mode model. This model was implemented both by means of static or dynamic analysis.

Another approach to model input validation vulnerabilities is to model the syntactic structure for sensitive operations arguments. The idea behind this is that the web application is susceptible to an injection attack if the syntactic structure for sensitive operation arguments depends on the user input. This approach was implemented by means of string analysis in static and it was applied to detect SQLI and XSS vulnerabilities in PHP. After all, this approach was implemented to detect injection attacks at runtime.

One of the main drawbacks of static analysis, in general, is its susceptibility to false positives caused by inevitable analysis imprecision. This is made worse by the dynamic nature of scripting languages. However, static analysis techniques normally perform conservative analysis that considers every possible control path.

One of the main drawbacks of dynamic analysis is that it is performed on executed paths and does not give any guarantee about paths not covered during a given execution. However, dynamic analysis having access to the internals of the web application execution process has the potential of being more precise. In this paper, we focus on the most common model of input validation vulnerabilities. First, we identify several drawbacks of this model. For instance, we analyze why this model cannot be used to detect inter-module vulnerabilities, which make second-order injection attacks possible. Then, we propose a solution to the stated drawbacks.

2. LITERATURE SURVEY

To provide a more precise perspective in the field of cryptographic algorithms, this section discusses the results obtained from various resources.

James A. Kupsch and Barton P. Miller researched [1] the difference between manual and automated assessment tools of the website about the dream of every software development team i.e. to assess the security of their software using only a tool. The author attempted to evaluate and quantify the effectiveness of automated source code analysis tools by comparing such tools to the results of an in-depth manual evaluation of the same system. The author presented their manual vulnerability assessment methodology, and the results of applying it to a major piece of software, then analyzed the same software using two commercial products, Coverity Prevent and Fortify SCA, that perform static source code analysis. These tools found only a few of the fifteen serious vulnerabilities discovered in the manual assessment, with none of the problems found by these tools requiring a deep understanding of the code. Each tool reported thousands of defects that required human inspection, with only a small number being security related. Thus, their recorded results proved that manual assessment is more effective than automated assessment for static software.

Aniqua Z. Baset researched various IDE Plugins for detecting many vulnerabilities in products and systems that could be avoided if better secure coding practices were in place. There exist a number of Integrated Development Environment [2] (IDE) plugins which help developers check for security flaws while they code. In this work, we present a review of these plugins. We specifically focus on the plugins that detect input validation-related vulnerabilities. We list salient features such as their supported IDEs, applicable languages and specific types of vulnerability checks. We believe this work synthesizes information useful for future research on IDE plugins for detecting input-validation-related vulnerabilities.

José Fonseca and Marco Vieira attempted to support customization, in many web applications to allow the integration of third-party server-side plugins that offer diverse functionality, but also open an additional door for security vulnerabilities. In this paper, he presented the use of static code analysis tools to detect vulnerabilities in the plugins of the web application [3]. The goal is twofold: 1) to study the effectiveness of static analysis on the detection of web application plugin vulnerabilities, and 2) to understand the potential impact of those plugins in the security of the core web application. They use two static code analyzers to evaluate a large number of plugins for a widely used Content Management System. Results show that many plugins that are currently deployed worldwide have dangerous Cross-Site Scripting and SQL Injection vulnerabilities that can be easily exploited and that even widely used static analysis tools may present disappointing vulnerability coverage and false positive rates.

Teemu Koskinen, Petri Ihantola, and Ville Karavirta applied static analysis to find out how vulnerable the plugins available at the official WordPress plugin directory are too well-known security exploits [4]. They compared the number of potential vulnerabilities and vulnerability density to the user ratings, to determine if user ratings can be used for finding secure plugins. After thorough research, they concluded that the quality of the plugins varies and there is no clear correlation between the ratings of plugins and the number of vulnerabilities detected in them. Hence, recommended manual inspection of each tool to verify those tools.

Paulo Nunes, José Fonseca under an increasing pressure to develop complex web applications at a fast pace. The vast majority is built using frameworks based on third-party server-side plugins that allow developers to easily add new features. However, as many plugin developers have limited programming skills, there is a spread of security vulnerabilities related to their use. Best practices advise the use of systematic code review for assuring security, but free tools do not support OOP, which is how most web applications are currently developed. To address this problem, we propose phpSAFE, a static code analyzer that identifies vulnerabilities in PHP plugins developed using OOP. They tested phpSAFE against two well-known tools using 35 plugins for a widely used CMS. The test results showed that phpSAFE clearly outperforms other tools and that plugins are being shipped with a considerable number of vulnerabilities, which tends to increase over time. [5]
With the increase in the number of threats within web-based systems, a more integrated approach is required to ensure the enforcement of security policies from the server to the client. These policies aim to stop man-in-the-middle attacks, code injection, and so on. This study analyzes some of the newest security options used within HTTP responses and scans the Alexa Top 1 Million sites for their implementation within HTTP responses. These options scanned for include: content security policy, public key pinning extension for HTTP, HTTP strict transport security, and HTTP header field X-frame-options, in order to understand the impact that these options have on the most popular websites. [6]

The results show that, while the implementation of the parameters is increasing, it is still not implemented on many of the top sites. Along with this, the study shows the profile of adoption of Let's Encrypt digital certificates across the one million sites, along with a way of assessing the quality of the security headers. The increase in sites redirecting to HTTPS is gathering pace fast. The securityheaders.io scores were a little poor, but there is plenty of opportunity for easy improvements to be made with the simpler X-based headers.

Marie Vasek, John Wadleigh, Tyler Moore. Paper describes a case-control study to identify risk factors that are associated with higher rates of web server compromise [7]. They inspected a random sample of around 200000 web servers and automatically identify attributes hypothesized to affect the susceptibility to compromise, notably Content Management System (CMS) and web server type. They find that web servers running WordPress and Joomla are more likely to be hacked than those not running any CMS and that servers running Apache and Nginx are more likely to be hacked than those running Microsoft IIS. And identify several WordPress plugins and Joomla extensions associated with compromise. Furthermore, using a series of logistic regressions, they found that a CMS’s market share is positively correlated with website compromise. Surprisingly, they discovered that web servers running outdated software are less likely to be compromised than those running up-to-date software. The paper presents evidence that this is true for core WordPress software (the most popular CMS platform) and many associated plugins.

3. BUSTED: SYSTEM ARCHITECTURE

![Fig. 1: Architecture](image)

Busted is an end user static code analyzer for detecting vulnerabilities in PHP plugins as well as themes. Busted tools only require is a local web and a client-side web browser. Busted has a web interface that assists the end-user to check the vulnerability on its web browser and provides with appropriate precautions to be taken and steps to correct them. Busted is set up to be effectively coordinate different new vulnerabilities from the cloud into the webpage and different WordPress sites. The output of the analysis is presented on the web page and is sent a report that helps reviewing the results, including the vulnerable variables. Busted creates and store the outcomes in report organizes and gives them to the site client. Busted is prepared in such a way that it can be easily integrated into WordPress websites. Busted source code analysis is based on three stages: 1) configuration, 2) analysis 3) results as illustrated in figure 1.

4. CONFIGURATION

During this stage, Busted connects the website to cloud database which contains the list of all the web vulnerabilities correlated to the WordPress framework. In the configuration of Busted, functions cover reviews the cloud database to detect loopholes or errors in the source code of the website or plugins and themes. It compares this database with the actual code to locate the vulnerabilities. Busted is deployed with a default configuration that is ready for detecting generic vulnerabilities, as well as for plugins and themes for the WordPress framework. This solution, out-of-the-box, has the advantage of allowing the immediate use of the tool as it covers all the newly discovered vulnerability hence updates with every update of various other plugins. However, this ability can be easily extended to other CMSs, by adding their input, filtering and sink functions to the configuration files.

5. ANALYSIS


Our tool evaluates a website for vulnerability in various categories such as PHP Security info, Plugin Scan, theme Scan and WordPress core scanner.

5.1 PHP security info

These checks are performed on the configuration of PHP your server is running. Functionality for this scan is provided by PHPSecInfo v0.2.1. To address these issues, you'll need to edit your .htaccess file to change some PHP settings and/or contact your sysadmin/host and request that they update the PHP configuration accordingly.

5.2 Plugin scanner

The Plugin scanner will gather all the Plugins you have installed and check against a vulnerability database to see if there are any open issues. Issues marked with Green are fixed in your current version. Issues marked with Yellow may require more investigation to determine if it needs to be addressed. Issues marked with Red need to be updated immediately to secure your WordPress site.

5.3 Theme scanner

The Theme scanner will gather all the Themes you have installed and check against a vulnerability database to see if there are any open issues. Issues marked with Green are fixed in your current version. Issues marked with Yellow may require more investigation to determine if it needs to be addressed. Issues marked with Red need to be updated immediately to secure your WordPress site.
5.4 WordPress core scanner

The WordPress Core scanner will scan your currently installed WordPress version against a vulnerability database to see if there are any open issues. Issues marked with Green are fixed in your current version. Issues marked with Yellow may require more investigation to determine if it needs to be addressed. Issues marked with Red need to be updated immediately to secure your WordPress site.

6. RESULT

After using the tool scanner, it generates a report according to the vulnerabilities found on the website and emails the information to the administrator of the website to keep the admin updated by continuously scanning site in every 12 hours about the security issues of their site.

With the help of red, yellow and green color codes the report generated represents the danger that vulnerability presents as well as the sincerity it needs to be dealt with.

7. PROPOSED WORK

Busted: A website vulnerability and assessment tool is an all-in-one vulnerability tool which checks all the aspects of a WordPress website like plugins and themes which are not yet covered by many current tools.

Busted also provides a unique assist to the end-side user i.e. it keeps the user updated about its WordPress website by regular analysis of the website in every 12 hours and providing a color-coded report to the user for better understanding of the vulnerability and its risk. And provides appropriate information to the user to take proper precautions/steps to remove the vulnerabilities.

8. CONCLUSION AND FUTURE WORK

In this paper we presented Busted, an API integrated vulnerability analyzer that can detect vulnerabilities in both plugins and themes of applications developed in PHP. There are other free tools to search for vulnerabilities in PHP code, like RIPS and Pixy, and PHP safe but they are neither ready for analyzing plugins and themes. As WordPress applications are so common, hence Busted outperforms the other tools. Using many tools allows increasing the number of different vulnerabilities in a website detected, as busted performs all-in-one security vulnerability detection and showing that there is room for improvement for measures to be taken during attacks.

Future work includes the improvement of Busted, mainly regarding performance, memory consumption and vulnerability and attacks coverage, along with the analysis of other CMS applications like Drupal or Joomla. We also intend to study the evolution of plugin security and plugin updates over time by enabling historic data in Busted. Developers may use it for approving third-party plugins before allowing their integration.

9. REFERENCES

[1] James A. Kupsch and Barton P. Miller Computer Sciences Department, University of Wisconsin, Madison, WI, USA.
[2] IDEPluginsforDetectingInput-ValidationVulnerabilities
[4] A Practical Experience on the Impact of Plugins in Web Security José Fonseca CISUC, University of Coimbra / Polytechnic Institute of Guarda, Portugal josefonseca@ipg.pt Marco Vieira CISUC, University of Coimbra Coimbra, Portugal mvieira@dei.uc.pt
[7] Analysis of the adoption of security headers in HTTP William J. Buchanan, Scott Helme, Alan Woodward School of Computing, The Cyber Academy, Edinburgh Napier University, Edinburgh, UK 2Independent Contractor, scotthelme.co.uk 3Surrey Centre for Cyber Security, University of Surrey, Surrey, UK E-mail: w.buchanan@napier.ac.uk.