ISSN 2348 – 8034
Impact Factor-4.022

GLOBAL JOURNAL OF ENGINEERING SCIENCE AND RESEARCHES

DETECTING AND PREVENTING SQL INJECTION AND XSS ATTACK USING WEB SECURITY MECHANISMS

Ms. Aboli Vairagade[*1], Prof. D.M. Sable[2] and Prof. V. R. Wadhankar[3]
[*1,2,3]Agnihotri Collge of Engineering Nagthana Wardha

ABSTRACT

In this paper we proposed a framework model instrument to assess web application security components. The philosophy depends on the possibility that infusing practical vulnerabilities in a web application and assaulting them naturally can be utilized to bolster the evaluation of existing security systems and apparatuses in custom setup situations. To give consistent with life comes about, the proposed powerlessness and assault infusion technique depends on the investigation of an expansive number of vulnerabilities in genuine web applications. To expel the vulnerabilities by executing a solid Vulnerability and Attack Injector Tool (VAIT) for securing web applications.

To prevent various attacks like follows:

1. SQL Injection (SQLi)
2. Cross Site Scripting (XSS)
3. Brute Force Attack
4. Shoulder surfing Attack
5. Social Attack.
6. Dictionary Attack

Keywords: *SQL Injection, XSS Attack, Web security etc.*

I. INTRODUCTION

These days there is an expanding reliance on web applications, running from people to huge associations. Just about everything is put away, accessible or exchanged on the web. Web applications can be close to home sites, online journals, news, informal organizations, web sends, bank offices, gatherings, e-trade applications, and so on. The inescapability of web applications in our lifestyle and in our economy is important to the point that it makes them a characteristic focus for noxious personalities that need to adventure this new streak.

Adroitly, the assault infusion comprises of the presentation of practical vulnerabilities that are after wards consequently misused (assaulted). Vulnerabilities are considered realisticbecause they are gotten from the broad field study on genuine web application vulnerabilities introduced in [16], and are infused by set of agent limitations and tenets characterized in [17]. The assault infusion ethodology depends on the dynamic examination of data acquired from the runtime checking of the web application conduct and of the cooperation with outer assets, for example, the backend database. This data, supplemented with the static examination of the source code of the application, permits the compelling infusion of vulnerabilities that are like those found in this present reality. Despite the fact that this strategy can be connected to different sorts of vulnerabilities, we concentrate on of the most generally abused and genuine web application vulnerabilities that are SQL Injection (SQLi) and Cross Site Scripting (XSS) [3], [6]. Assaults to these vulnerabilities essentially exploit shameful coded applications because of nchecked information fields at client interface. This permits the assailant to change the SQL orders that are sent to the database (SQLi) or through the contribution of HTML and scripting dialects (XSS).

A Brute-Force Attack, or thorough key inquiry, is a cryptanalytic assault that can, in principle, be utilized against any encoded data[1] (aside from information scrambled in a data hypothetically secure way). Such an assault may be utilized when it is unrealistic to exploit different shortcomings in an encryption framework (if any exist) that would make the undertaking less demanding. It comprises of efficiently checking all.

Shoulder surfing should likewise be possible at a separation utilizing binoculars or other vision-improving gadgets. Economical, smaller than expected shut circuit TV cameras can be covered in roofs, dividers or installations to watch information passage. To anticipate shoulder surfing, it is encouraged to shield printed material or the keypad from perspective by utilizing one's body or measuring one's hand.

A Dictionary Attack depends on attempting all the strings in a prearranged posting, ordinarily got from a rundown of words, for example, in a lexicon (henceforth the expression word reference assault). [1] as opposed to an animal power assault, where a vast extent of the key space is looked deliberately, a lexicon assault tries just those potential outcomes which are esteemed destined to succeed. Lexicon assaults frequently succeed in light of the fact that numerous individuals tend to pick short passwords that are conventional words or regular passwords, or basic variations acquired.

Social Attack, with regards to data security, alludes to mental control of individuals into performing activities or revealing private data. A sort of certainty trap with the end goal of data social affair, extortion, or framework access, it contrasts from a customary "con" in that it is regularly one of numerous progressions in a more mind boggling misrepresentation plan. The expression "social building" as a demonstration of mental control is additionally connected with the sociologies, however its use has gotten on among PC and data security experts.

II. PROPOSED SYSTEM AND WORK

The methodology proposed was implemented in a concrete Vulnerability & Attack Injector Tool (VAIT) for web applications. The tool was tested on top of widely used applications in two scenarios. The first to evaluate the effectiveness of the VAIT in generating a large number of realistic vulnerabilities for the offline assessment of security tools, in particular web application vulnerability scanners. The second to show how it can exploit injected vulnerabilities to launch attacks, allowing the online evaluation of the effectiveness of the counter measure mechanisms installed in the target system, in particular an intrusion detection system.

In practice, the use of both static and dynamic analysis is a key feature of the methodology that allows increasing the overall performance and effectiveness, as it provides the means to inject more vulnerability that can be successfully attacked and discarded those that cannot.

The proposed methodology provides a practical environment that can be used to test countermeasure mechanisms (such as intrusion detection systems (IDSs), web application vulnerability scanners, web application fire-walls, static code analyzers, etc.), train and evaluate security teams, help estimate security measures (like the number of vulnerabilities present in the code), among others. This assessment of security tools can be done online by executing the attack injector while the security tool is also running; or offline by injecting a representative set of vulnerabilities that can be used as a test bed for evaluating a security tool.

2.1 SYSTEM ARCHITECTURE

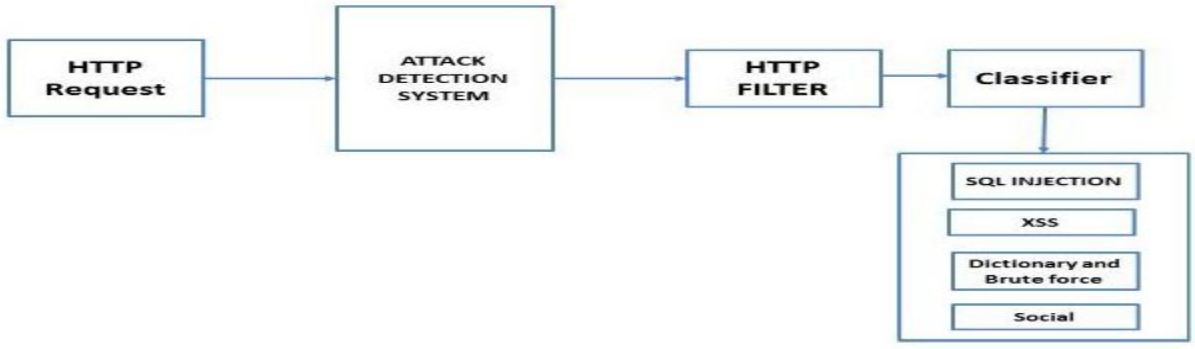


Fig1. System Architecture

2.2 MODULES

2.2.1 DETECTING AND PREVENTING SQL INJECTION ATTACK

SQL injection is a code injection technique, used to attack data-driven applications, in which malicious SQL statements are inserted into an entry field for execution (e.g. to dump the database contents to the attacker).[1] SQL injection must exploit a security vulnerability in an application's software, for example, when user input is either incorrectly filtered for string literal escape characters embedded in SQL statements or user input is not strongly typed and unexpectedly executed. SQL injection is mostly known as an attack vector for websites but can be used to attack any type of SQL database.

2.2.2 DETECTING AND PREVENTING XSS ATTACK

Cross-Site Scripting (XSS) vulnerabilities are a type of computer security vulnerability typically found in Web applications. XSS vulnerabilities enable attackers to inject client-side script into Web pages viewed by other users. A cross-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy. Cross-site scripting carried out on websites accounted for roughly 84% of all security vulnerabilities documented by Symantec as of 2007.[1] Their effect may range from a petty nuisance to a significant security risk, depending on the sensitivity of the data handled by the vulnerable site and the nature of any security mitigation implemented by the site's owner.

2.2.3 DETECTING AND PREVENTING DICTIONARY ATTACK

It is possible to achieve a time-space tradeoff by pre-computing a list of hashes of dictionary words, and storing these in a database using the hash as the key. This requires a considerable amount of preparation time, but allows the actual attack to be executed faster. The storage requirements for the pre-computed tables were once a major cost, but are less of an issue today because of the low cost of disk storage. Pre-computed dictionary attacks are particularly effective when a large number of passwords are to be cracked. The pre-computed dictionary need only be generated once, and when it is completed, password hashes can be looked up almost instantly at any time to find the corresponding password.

2.2.4 DETECTING AND PREVENTING SOCIAL ATTACK

Some automated teller machines have a sophisticated display which discourages shoulder surfers from obtaining displayed information. It grows darker beyond a certain viewing angle, and the only way to tell what is displayed on the screen is to stand directly in front of it.

2.2.5 DETECTING AND PREVENTING BRUTE FORCE ATTACK

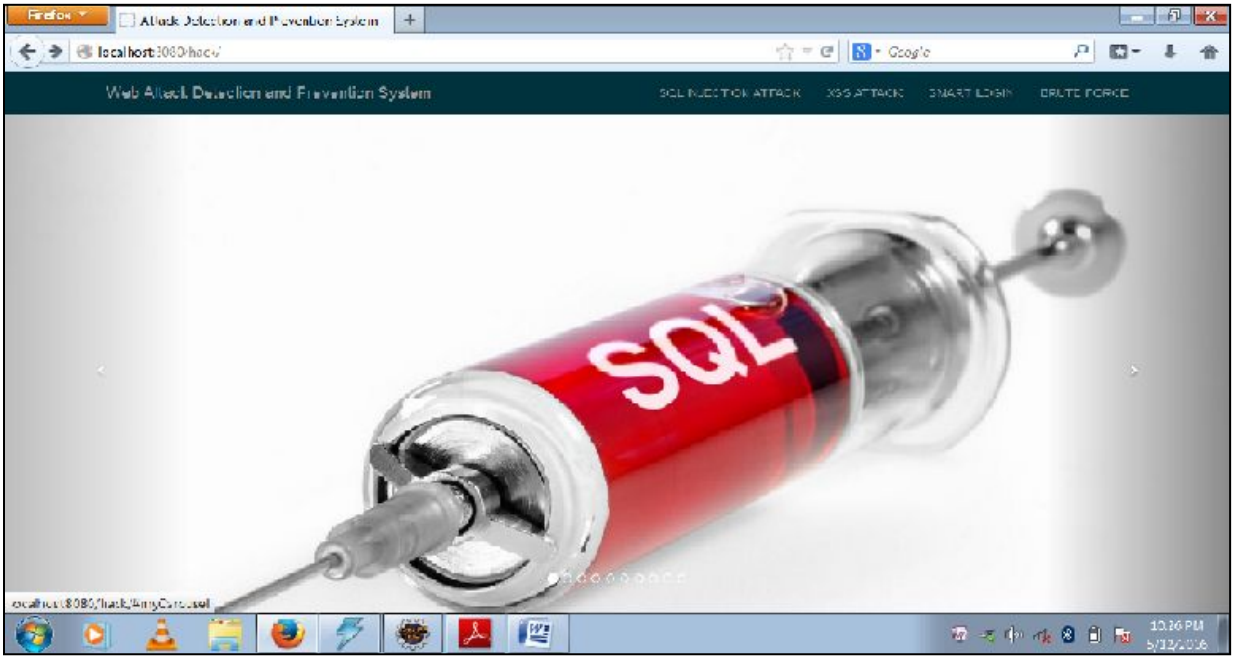
One of the measures of the strength of an encryption system is how long it would theoretically take an attacker to mount a successful brute-force attack against it. Brute-force attacks are an application of brute-force search, the general problem-solving technique of enumerating all candidates and checking each one

2.2.6 DETECTING AND PREVENTING SHOULDER SURFING ATTACK

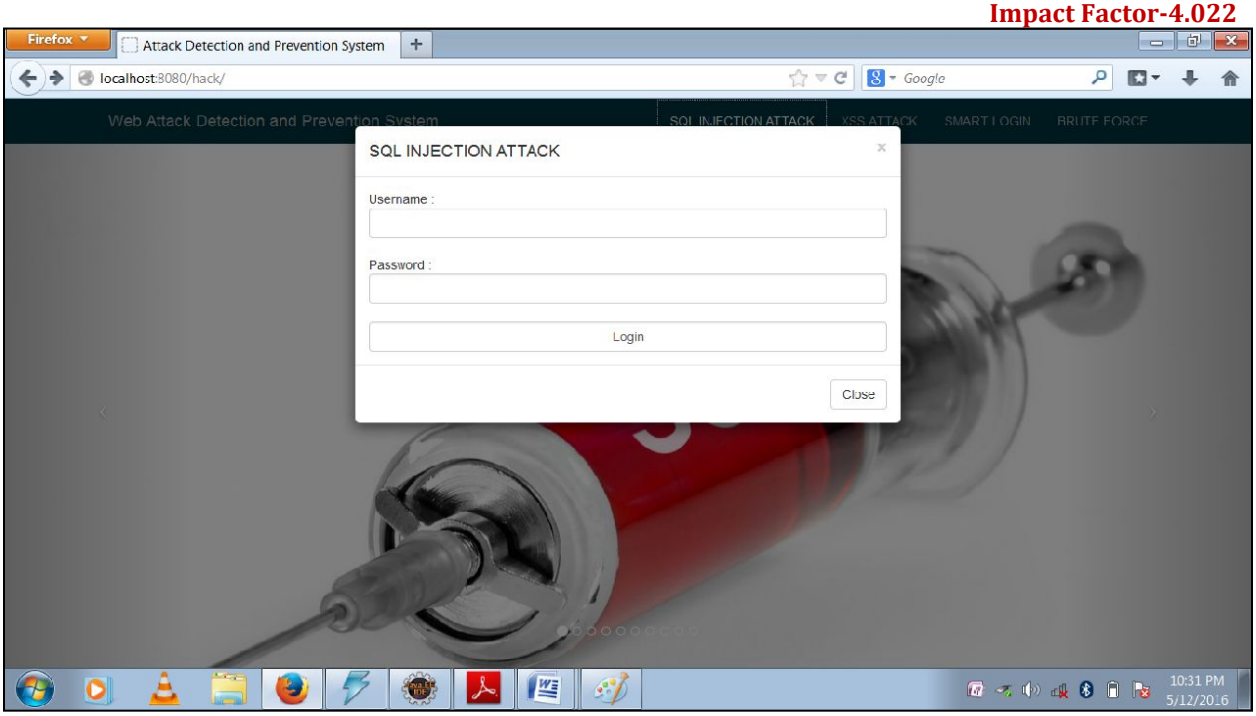
ISSN 2348 – 8034
Impact Factor-4.022

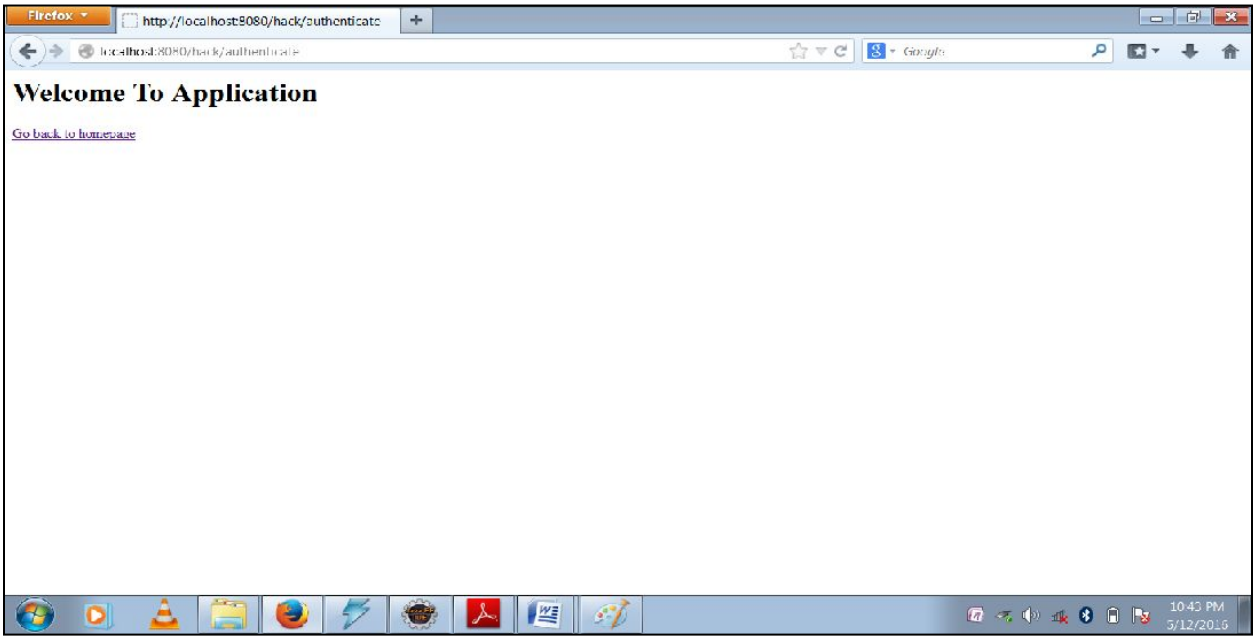
This technique can be used to fool a business into disclosing customer information as well as by private investigators to obtain telephone records, utility records, banking records and other information directly from company service representatives. The information can then be used to establish even greater legitimacy under tougher questioning with a manager, *e.g.*, to make account changes, get specific balances, etc.

III. SNAPSHOTS

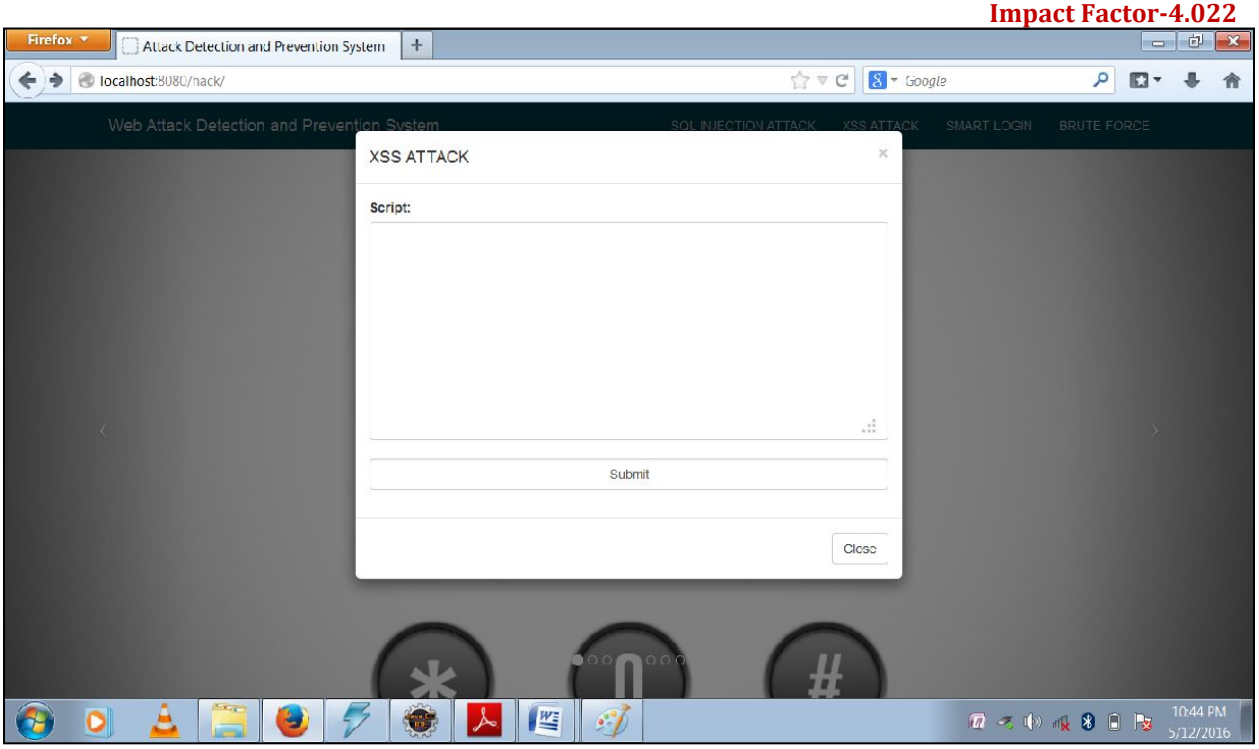


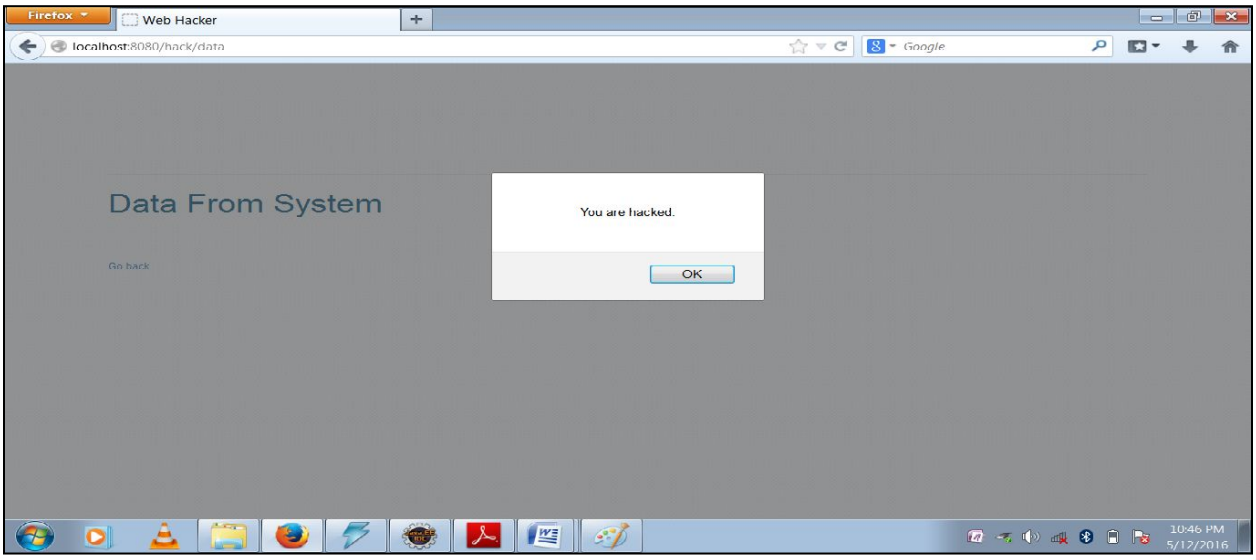
ISSN 2348 – 8034
Impact Factor-4.022



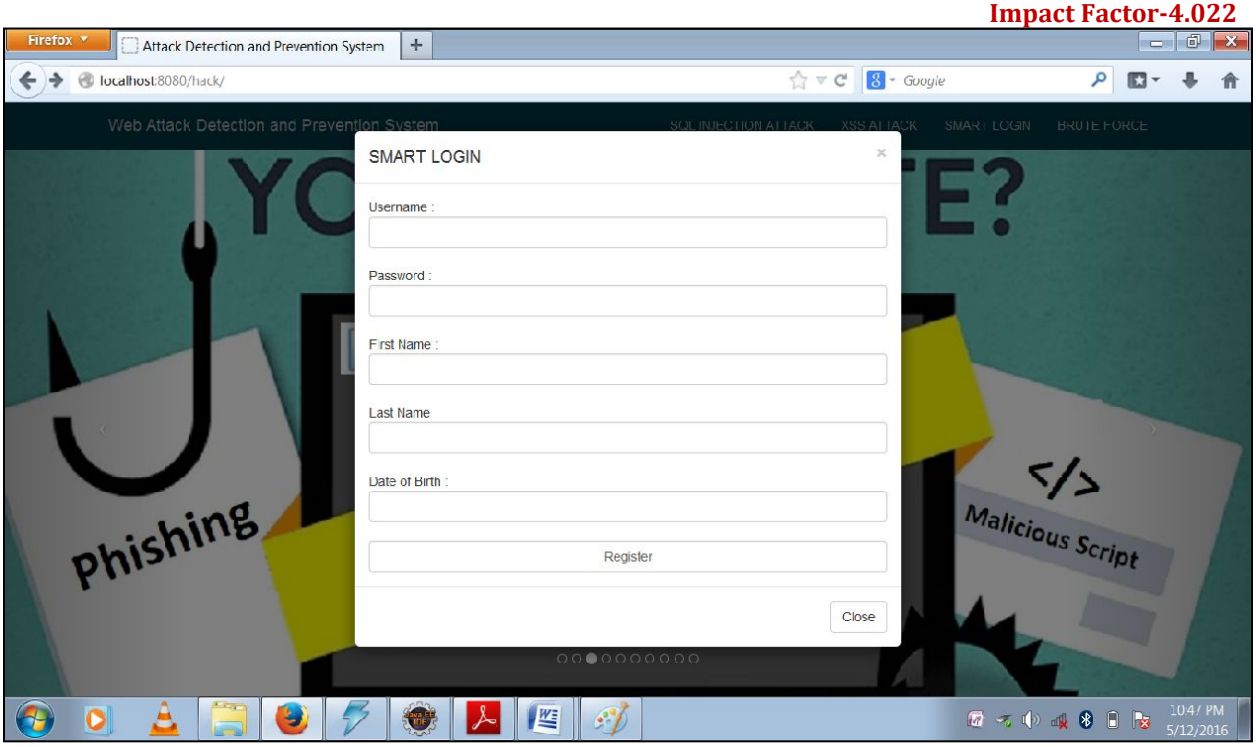


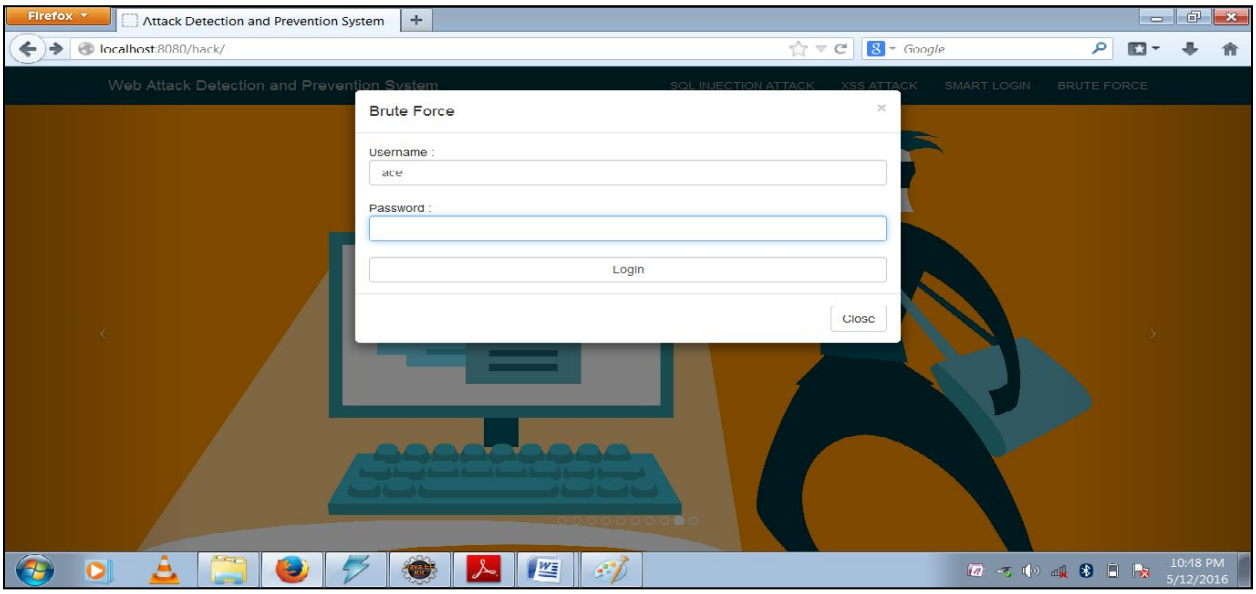
ISSN 2348 – 8034
Impact Factor-4.022



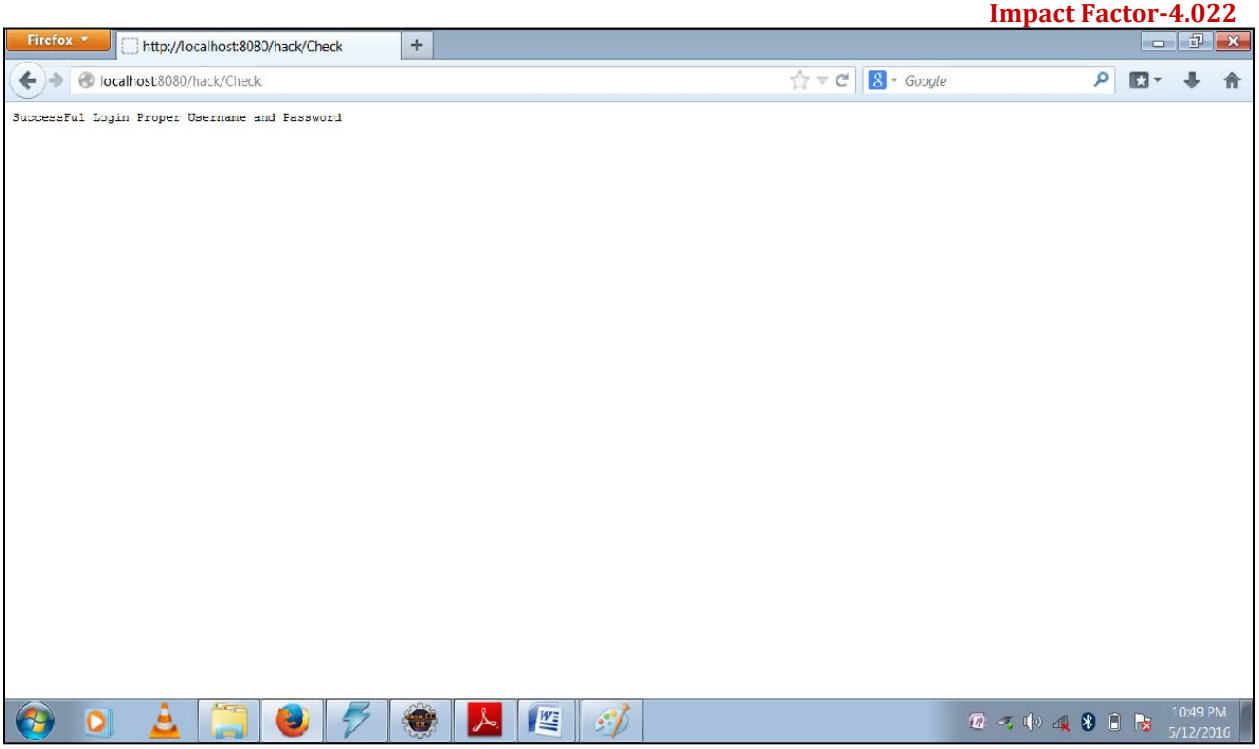


ISSN 2348 – 8034
Impact Factor-4.022





ISSN 2348 – 8034
Impact Factor-4.022



IV. CONCLUSION

The SQL-injection attacks are tremendously dangerous in association to other types of web based attacks for the reason that here the end result is data manipulation. SQL injection holes can be easily exploit by a technique called SQL injection attacks. This proposed integrated approach is an effort to add some more security measures to databases to avoid SQL injection attack.

REFFERENCES

[1] Jose Fonseca, Marco Vieira, and Henrique Madeira "Evaluation of Web Security MechanismsUsing Vulnerability & Attack Injection"-IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING, VOL. 11, NO. 5, SEPTEMBER/OCTOBER 2014.

[2] D. Avresky, J. Arlat, J.C. Laprie, and Y. Crouzet, "Fault Injection for Formal Testing of Fault Tolerance," IEEE Trans. Reliability, vol. 45, no. 3, pp. 443-455, Sept. 2011

[3] J. Arlat, A. Costes, Y. Crouzet, J.-C. Laprie, and D. Powell, "Fault Injection and Dependability Evaluation of Fault-Tolerant Systems," IEEE Trans. Computers, vol. 42, no. 8, pp. 913-923, Aug. 2011.

[4] N. Neves, J. Antunes, M. Correia, P. Ver_ıssimo, and R. Neves, "Using Attack Injection to Discover New Vulnerabilities," Proc. IEEE/IFIP Int'l Conf. Dependable Systems and Networks, 2006.

[5] N. Jovanovic, C. Kruegel, and E. Kirda, "Precise Alias Analysis for Static Detection of Web Application Vulnerabilities," Proc. IEEE Symp. Security Privacy, 2006.

[6] IBM Global Technology Services "IBM Internet Security Systems X-Force 2012 Trend & Risk Report," IBM Corp., Mar. 2013.

[7] The Privacy Rights Clearinghousewww.privacyrights.org/databreach, Accessed 1 May 2013, Apr. 2012.

[8] M. Fossi, et al., "Symantec Report on the Underground Economy, Symantec Security Response," 2008.

[9] D. Powell and R. Stroud, "Conceptual Model and Architecture of MAFTIA," Project MAFTIA, Deliverable D21, 2003.

[10] B. Livshits, "Stanford SecuriBench," suif.stanford.edu/_livshits/ securibench, Accessed 1 May 2013, 2005

[11] D. Powell and R. Stroud, "Conceptual Model and Architecture of MAFTIA," Project MAFTIA, Deliverable D21, 2003.

[12] V. Krsul, "Software Vulnerability Analysis," PhD thesis, Purdue univ.

[13] J. Fonseca and M. Vieira, "Mapping Software Faults with We Security Vulnerabilities," Proc. IEEE/IFIP Int'l. Conf. Dependable Systems and Networks, June 2008.

[14] B. Damele, "Sqlmap: Automatic SQLi Tool," sqlmap.sourceforge. net, Accessed 1 May 2013, 2009.