ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

Enhanced XSS/SQL Injection Techniques for Web Site Vulnerabilities with Amplified Algorithm

Rashmi[1], Deepika Goyal[2]

M. Tech. Student, Department of Computer Science & Engineering, Advance Institute of Technology and Management at Palwal, Haryana, India[1]

Assistant Professor, Department of Computer Science & Engineering, Advance Institute of Technology and Management at Palwal, Haryana, India[2]

ABSTRACT: These days, the biggest threat to an organization's network security comes from its public Web site and the Web-based applications found there. Unlike internal-only network services such as databases—which can be sealed off from the outside via firewalls—a public Web site is generally accessible to anyone who wants to view it, making application security an issue. As networks have become more secure, vulnerabilities in Web applications have inevitably attracted the attention of hackers, both criminal and recreational, who have devised techniques to exploit these holes. In fact, attacks upon the Web application layer now exceed those conducted at the network level, and can have consequences which are just as damaging**.** Consequently, under the scheme we proposed Enhanced Counter Measure for Web Site Vulnerabilities with Amplified Algorithm to intrude and get vital and confidential information from the web site and web application prone to attacks and weak to protect its nitty-gritty.

KEYWORDS: Web Security, Cross Site Scripting, Sql Injection, Web Site Vulnerabilities.

I. INTRODUCTION

With XSS, every input has the potential to be an attack vector, which does not occur with other vulnerability types. This leaves more opportunity for a single mistake to occur in a program that otherwise protects the web application against XSS. SQL injection also has many potential attack vectors. Despite the popular opinion that XSS is easily prevented, it has many subtleties and variants. Even solid applications can have flaws in them; consider non-standard browser behavior that tries to 'fix' the malformed HTML, which might slip by a filter that uses regular expressions. Finally, until early 2006, the PHP interpreter had a vulnerability in which it did not quote error messages, but many researchers only reported the surface-level 'resultant' XSS instead of figuring out whether there was a different 'primary' vulnerability that led to the error the same is depicted below with description for ready reference:-

Vulnerabilities	Description
Cross Site Scripting (XSS)	XSS flaws occur whenever an application takes user supplied data and sends it to a web browser without first validating or encoding that content. XSS allows attackers to execute script in the victim's browser which can hijack user sessions, deface web sites, possibly introduce worms, etc.
Injection Flaws	Injection flaws, particularly SQL injection, are common in web applications. Injection occurs when user-supplied data is sent to an interpreter as part of a command or query. The attacker's hostile data tricks the interpreter into executing unintended commands or changing data.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

Malicious File Execution	Code vulnerable to remote file inclusion (RFI) allows attackers to include hostile code and data, resulting in devastating attacks, such as total server compromise. Malicious file execution attacks affect PHP, XML and any framework, which accepts filenames or files from users.
Cross Site Request Forgery (CSRF)	A CSRF attack forces a logged-on victim's browser to send a pre-authenticated request to a vulnerable web application, which then forces the victim's browser to perform a hostile action to the benefit of the attacker. CSRF can be as powerful as the web application that it attacks.
Information Leakage and Improper Error Handling	Applications can unintentionally leak information about their configuration, internal workings, or violate privacy through a variety of application problems. Attackers use this weakness to steal sensitive data, or conduct more serious attacks.
Broken Authentication and Session Management	Account credentials and session tokens are often not properly protected. Attackers compromise passwords, keys, or authentication tokens to assume other users' identities.
Insecure Cryptographic Storage	Web applications rarely use cryptographic functions properly to protect data and credentials. Attackers use weakly protected data to conduct identity theft and other crimes, such as credit card fraud.

As affirmed previously web application use the database to deliver the required information to its visitors. If web applications are not secure, i.e., vulnerable to, at least one of the various forms of hacking techniques, then the entire database of sensitive information is at serious risk. Some hackers, for example, may maliciously inject code within vulnerable web applications to trick users and redirect them towards Phishing sites. This technique is called Cross-Site Scripting (XSS) and may be used even though the web servers and database engine contain no vulnerability themselves. Recent research shows that 80% of cyber attacks are done at the web application level. The figure 1 shows the hacking attempt.

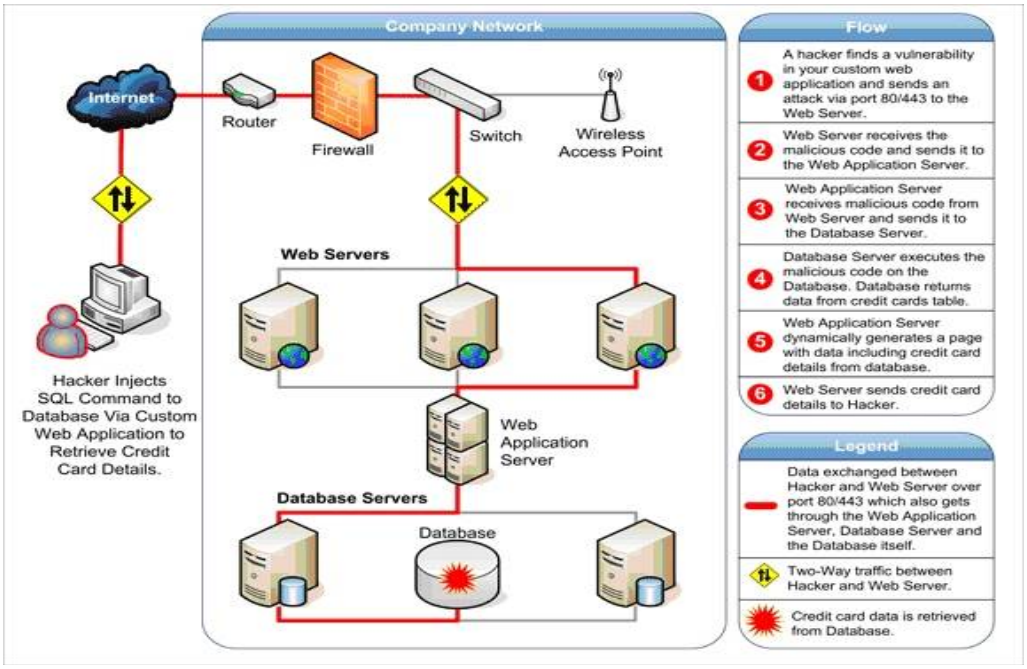

Figure 1: Depiction of a hacking attempt

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

II. RELATED WORK

The usage of Personal Computers, handheld devices and propelled cells has staggeringly extended throughout late years, as substantiated by Stuttard and Pinto (2011) web applications have been created to perform in every practical sense every profitable limit you could realize on the web. These fuse Online Shopping, Social Networking, Gambling and Online club, Banking, Web look for, Auctions, Webmail, and Interactive website pages among others. In a report dispersed by Whitehat "86% of all destinations attempted by Whitehat Sentinel had no short of what one bona fide lack of protection, and usually, essentially more than one – 56% to be correct. (Whitehat, 2015)

As showed by Shema (2011), various affiliations rely on modified web applications to realize business shapes. These may fuse absolute applications, or include modules, for instance, on the web, login pages shopping crates, and diverse sorts of dynamic substance. A segment of these item applications in your framework could be made in-house. Likewise, some may be legacy locales with no alloted proprietorship or support. Physically looking at these for stipulations and arranging their criticalness for remediation can be a staggering task without dealing with attempts and using robotized gadgets to improve accuracy and profitability.

Agents are continually responding to requests from both inside and outside the affiliation's corporate framework using gadgets, for instance, tablets, mobile phones or PCs. While this has tremendous focal points, the negative drawback is the way that software engineers may misuse accessibility to increment unapproved access to basic association information. Subsequently, it is essential for any association to ensure that they guarantee their web applications and reduce the probability of a security break to their electronic structure. Testing the weakness of web applications with modernized penetration testing instruments conveys by and large lively results. Starting at now, there are various such gadgets, both business and open source.

III. PROPOSED METHODOLOGY

The proposed scheme starts by the task of code analysis where static analysis is performed first. The static analysis operates on python source code files where it analyzes every file to determine specified vulnerabilities. The vulnerabilities are specified by the security rules which behave as the security knowledge for static analysis technique. Once the static analysis is completed, the next step is to perform the dynamic analysis on the web application. The dynamic analysis carries out the testing process by the use of instrumentation technique. The instrumentation (payloads) approach is based on the idea that, the attacks occurring due to the input validation vulnerabilities can be handled by adding the validation to the source code by determining of instrumentation technique (payloads) of the original source code with the pre-defined instrumentation templates (payloads). Therefore, the instrumentation code would perform the validation on the input given at the runtime, as a result of which the attacks would be stopped from being carried out and also the attempt for an attack can be reported during the web application's runtime. To do this, an proposed scheme generates the instrumentation code based on the instrumentation templates that contains the specified templates for each target vulnerability type. Later on, the proposed scheme also inserts the generated instrumentation code into the original web application code automatically. For inserting the instrumentation code, the locations are extracted from the results produced by the static results. As, the instrumented source code, which is actually combination of the original source code and the instrumentation code, is executed the runtime attacks are formed as well as reported to the user. The architecture diagram of the proposed scheme is shown in Fig. 1. The operations performed by the dynamic analysis agent for cross side scripts and sql injections are described in the points below:

1. The list of vulnerabilities is given as an input, along with the predefined instrumentation templates (payloads), to the instrument code generation agent.
2. The proposed scheme code generation agent generates appropriate instrumentation code (payloads) based on the vulnerabilities information provided by the vulnerabilities list. This information mainly includes the types of potential vulnerabilities, the location of vulnerabilities in the source code and the vulnerable method along with the vulnerable parameter of that method.

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

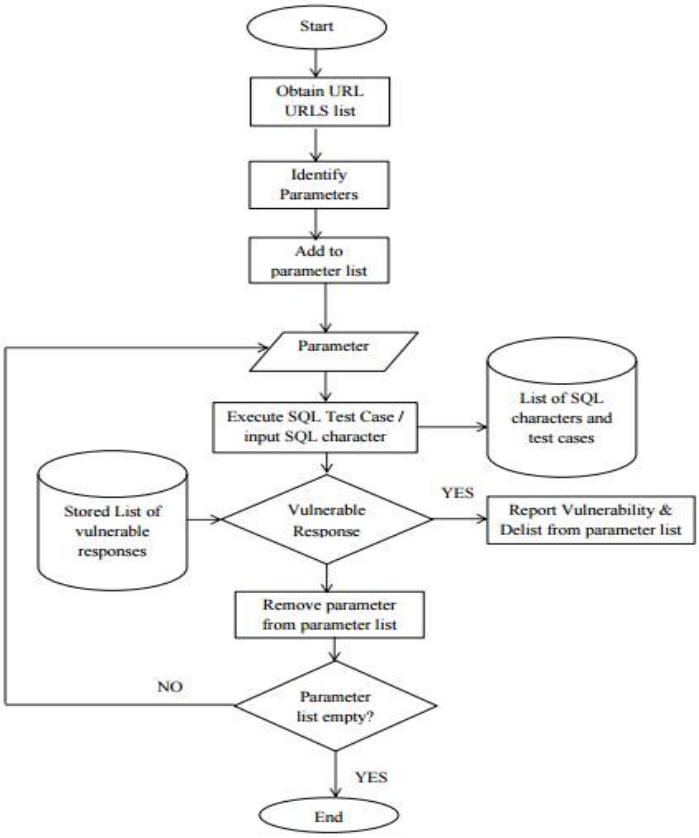

Figure 1: Architecture and Flow Diagram for proposed Amplified Algorithm Scheme.

IV. PSEUDO CODE AND SIMULATION RESULTS

Pseudo Code

```
__product__ = "Generic (Unknown)"
def detect(get_page):  retval = False page, _, code = get_page()
if page is None or code >= 400:
return False
for vector in WAF_ATTACK_VECTORS:
page, _, code = get_page(get=vector)
if code >= 400 or IDS_WAF_CHECK_PAYLOAD in vector and code is None: or
if code >= 400:
        retval |= re.search(r"\A__cfduid=", headers.get(HTTP_HEADER.SET_COOKIE, ""), re.I) is not None
        retval |= headers.get("cf-ray") is not None
        retval |= re.search(r"CloudFlare Ray ID:|var CloudFlare=", page or "") is not None
for vector in WAF_ATTACK_VECTORS:
      _, headers, _ = get_page(get=vector)
      retval = re.search(r"ACE XML Gateway", headers.get(HTTP_HEADER.SERVER, ""), re.I) is not None
      if retval:
      break
retval = True
break
return retval
```

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

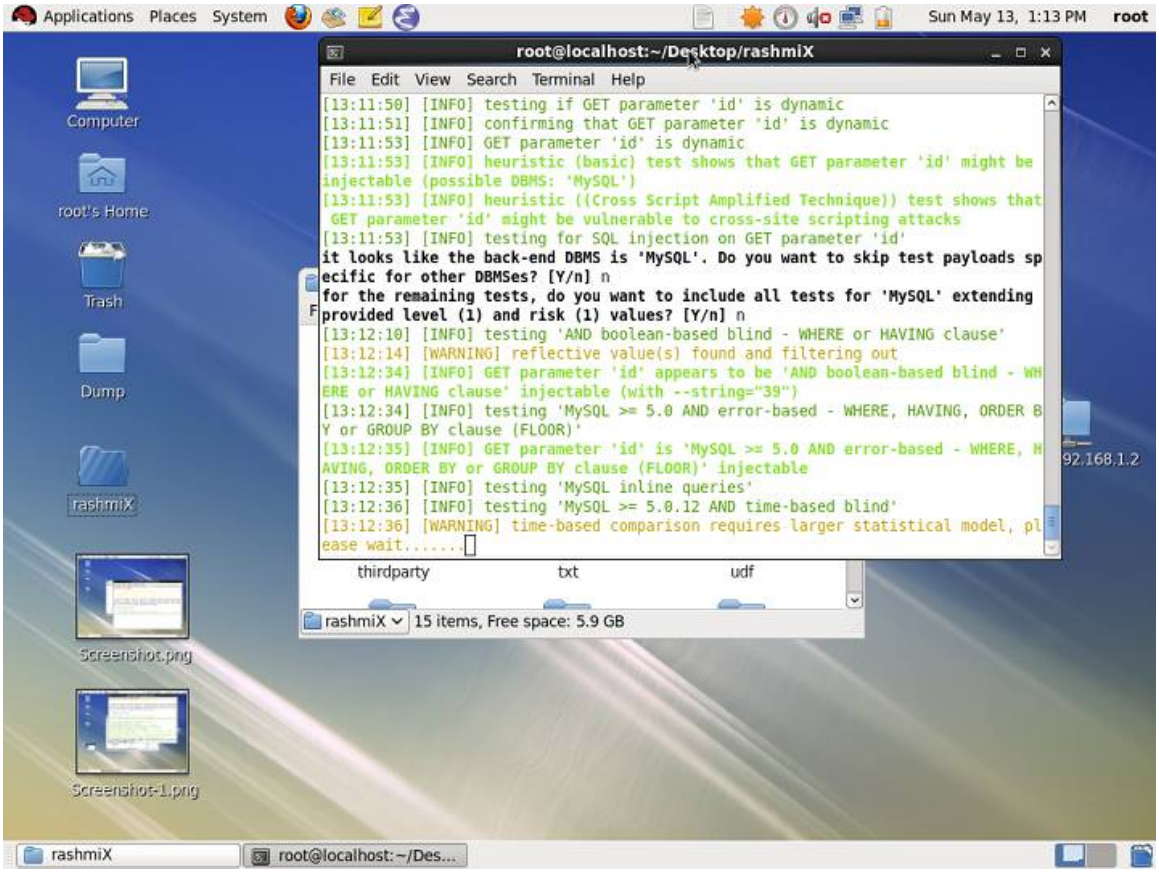

Figure 2: Vulnerability Exposed using Amplified Cross Script

Machine	CPU Time in Seconds
1 (i3 2.0 GHZ)	4.227
2 (i5 2.6 GHZ)	2.343

Table 1: CPU time used to execute the proposed algorithm in seconds using 8GB RAM

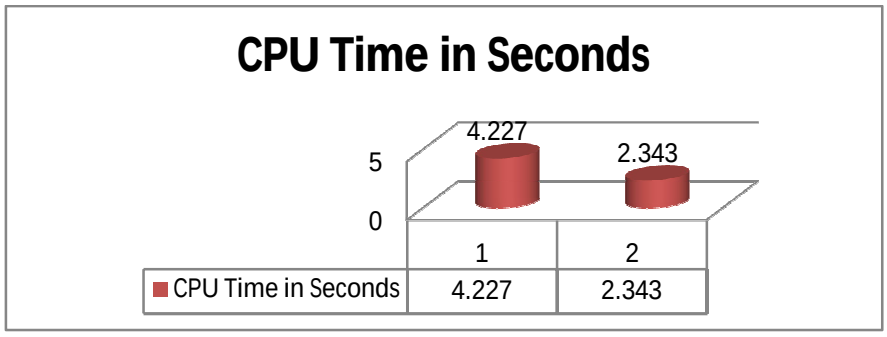

Figure 3: Graph Representation of proposed algorithm in seconds using 8GB RAM

ISSN(Online): 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 6, Issue 5, May 2018

V. CONCLUSION AND FUTURE WORK

The web application worldview is as yet advancing. Both JavaScript and HTML are under dynamic improvement. Web programs as of late executed HTML 5 noteworthy rendition of the dialect. New dialect components, for example, canvas, and expanded capacities, for example, cross-space HTTP asks for or relentless customer side stockpiling, may concede the foe new abilities. Thusly, existing and proposed XSS measures must be constantly reexamined whether regardless they work given the present condition of the innovation. Additionally, the interruption limits may prompt the improvement of right now XSS Payloads and Sql-Injection assaults besides risk to the web world and with respect to black hat the measures under scheme is been prepared. Additional future work is to put our new proposed conspire into a genuine firmware and system correspondence frameworks with to look at the XSS infiltration interruption. Despite the fact that we have investigated how proposed calculation breaks the entire execution of security frameworks, the information utilized as a part of our examination is manufactured and may not be illustrative this present reality situations. In future under the plan can plan to actualize the interruption on secured and solid stockpiling framework and utilize XSS against the stockpiles and cloud too.

REFERENCES

1. Alssir, F. T., & Ahmed, M. (2012). Web Security Testing Approaches: Comparison Framework. In Proceedings of the 2011 2nd International Congress on Computer Applications and Computational Science (pp. 163-169). Springer Berlin Heidelberg.
2. Antunes & Vieira (2012). Defending against web application vulnerabilities. Computer, (2), 66-72.
3. Bau, J., Bursztein, E., Gupta, D., & Mitchell, J. (2010). State of the art: Automated black-box web application vulnerability testing. In Security and Privacy (SP), 2010 IEEE Symposium on (pp. 332-345). IEEE.
4. Chen, S. (2014). wavsep. Available: http://sectooladdict.blogspot.com/2014/02/wavsep-web-application-scanner.html. [Accessed 09 July 2015.]
5. Dessiatnikoff, A., Akrout, R., Alata, E., Kaaniche, M., & Nicomette, V. (2011). A clustering approach for web vulnerabilities detection. InDependable Computing (PRDC), 2011 IEEE 17th Pacific Rim International Symposium on (pp. 194-203). IEEE.
6. Dougherty, C. (2012).Practical Identification of SQL Injection Vulnerabilities. 2012. US-CERT-United States Computer Emergency Readiness Team. Citado na, 34. . [Accessed: 08th June 2015]
7. Doupe, A., Cova, M., & Vigna, G. (2010). Why Johnny can't pentest: An analysis of black-box web vulnerability scanners. In Detection of Intrusions and Malware, and Vulnerability Assessment (pp. 111-131). Springer Berlin Heidelberg. [Accessed: 10th June 2015]
8. Fonseca, J., Vieira, M., & Madeira, H. (2014). Evaluation of Web Security Mechanisms using Vulnerability & Attack Injection. Dependable and Secure Computing, IEEE Transactions on, 11(5), 440-453.
9. Granville, K . (2015).Nine Recent Cyber-attacks against Big Businesses. New York Times [online] Available from http://www.nytimes.com/interactive/2015/02/05/technology/recent-cyberattacks.html?_r=1. [Accessed 08 July 2015.]
10. Howard, M., LeBlanc, D., & Viega, J. (2010). 24 deadly sins of software security [electronic book]: Programming flaws and how to fix them. New York: McGraw-Hill.
11. Jovanovic, N., Kruegel, C., & Pixy, E. K. (2010). A Static Analysis Tool for Detecting Web Application Vulnerabilities (Short Paper). In Proceedings of the 2006 IEEE symposium on Security and Privacy, Washington, DC, IEEE Computer Society (pp. 258-263).
12. Kalman., G. (2014). Ten Most Common Web Security Vulnerabilities.[online] Available from: http://www.toptal.com/security/10-most-common-web-security-vulnerabilities [Accessed 08 July 2015.]
13. Kals, S., Kirda, E., Kruegel, C., & Jovanovic, N. (2014). A web vulnerability scanner. In Proceedings of the 15th international conference on World Wide Web (pp. 247-256). ACM.
14. Khoury, N., Zavarsky, P., Lindskog, D., & Ruhl, R. (2011). Testing and assessing web vulnerability scanners for persistent SQL injection attacks. In Proceedings of the First International Workshop on Security and Privacy Preserving in e-Societies (pp. 12-18). ACM.
15. McQuade, K. (2014). Open Source Web Vulnerability Scanners: The Cost Effective Choice?. In Proceedings of the Conference for Information Systems Applied Research ISSN (Vol. 2167, p. 1508). [Accessed: 18th June 2015]