WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

WEBSITE SECURITY USING PLUGIN

Priyanka N. Bhosale[1], Ruchita K. Mali[1], Aishwarya D. Jadhav[1], Vikas A. Singh[1], Guide – Prof. Sujeet More[2]

[1]Bachelor of Engineering, Department of Information Technology, Trinity College of Engineering & Research (TCOER), Pune
[2]Bachelor of Engineering, Department of Information Technology, Trinity College of Engineering & Research (TCOER), Pune

ABSTRACT

In an attempt to support customization, many web applications 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 we study the use of static code analysis tools to detect vulnerabilities in the plugins of the web application. 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. We use two static code analysers 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.

1. INTRODUCTION

There is nowadays an increasing dependency on web ap- plications. Ranging from individuals to large organizations, almost everything is stored, available or traded on the web. Web applications can be personal web sites, blogs, news, social networks, web mails, bank agencies, forums, e-commerce applications, etc. The omnipresence of web applications in our way of life and in our economy is so important that they have turned into a natural target for malicious minds.

To allow customization and thus fit the requirements of diverse scenarios, many web applications support the integration of server-side plugins that offer multiple functionalities and may be provided by different parties. Well-known examples are Content Management Systems (CMSs) that allow individuals and/or communities of users to easily create and administrate web sites that publish a variety of contents. The sites created using CMSs can go from personal web pages and community portals to large corporate and e- commerce applications.

Although plugin-based web applications assure extensibility and customizability, the possibility of integrating third- party software opens an additional door for security vulnerabilities, regardless of the security assurance activities con- ducted on top of the core application. In fact, other works show a predominance of security exploits due to vulnerabilities in the external plugins, when compared to the core application. This is mostly due to the typically uncontrolled development processes and poor-quality assurance activities applied during the development of such plugins, which are not able to prevent security vulnerabilities from being shipped into the field.

In this paper we study the use of static analysis tools to detect vulnerabilities in a plugin-based web application. In practice, the goal is to study two key questions:

1. How effective are free static analysis tools detecting vulnerabilities in web application plug-in?

2. What is the real importance and impact of plug-in in the security of a web application?

2. LITERATURE SURVEY

As we Searched a lot, we observed that Cyber Attack on the Rise. • According to a new report, Cybercrime damages will hit $6 trillion annually by 2021, with32.5%ofallsuccessful attacks targeting e-commerce sites. • In the past 12 months, nearly 6 out of 10 organizations, 59%, have suffered from a “significant” security incident. The data is from the Global Information Security Survey, by Ernst and Young. The same survey found that 48% of executive boards believe that cyber-attacks or data breaches will “more than moderately” impact their business across the next 12 months

3. METHODOLOGY

1. Preparation of the experiments: create the conditions for running the static analyzers on top of relevant plugins.Two steps are needed:

   a) Identify a representative web application that allows the integration of plugins, and select a large set of widely used plugins for that application;

   b) Decide on the types of vulnerabilities to be the target of the study and select representative static analyzers able to detect those vulnerabilities;

2. Execution of the static code analyzers: analyze the plugins using the tools. This includes two steps, whose results are later processed and compared:

   a) Perform a generic analysis of the plugins using the typical configuration of the analyzers, i.e., not taking into account the fact that the target files are plugins for a specific web application;

   b) Run a targeted analysis in which the configuration ofthe analyzers is tuned for the specific context of the target web application;

The correlation of the results from these two steps al- lows studying the performance of the static code analyz ers in detecting Vulnerabilities when they are configured for the specific context of the web application plugins and when not, with respect to two key figures of merit: coverage and false positives. This may give insights to how these tools should evolve in the future, e.g., helping to understand whether the tools should be explicitly pre-pared to handle the analysis of plugins or just need to follow a more generic approach.

Analysis of the results:

Collect the reports of the tools and process the information gathered. This includes twosteps:

a) Manual verification of the vulnerabilities reported to confirm the true vulnerabilities and discard the false positives;

b) Analysis of data to understand the impact of plug-in in the application security and study the relative effectiveness, strengths and weaknesses, of the static analyzer tools. confirm the true vulnerabilities and discard the false positives;

c) Analysis of data to understand the impact of plug-in in the application security and study the relative effectiveness, strengths and weaknesses, of the static analyzer tools.

Table 1. TOOLS CONFIGURED FOR WORDPRESS: XSS DETECTION

	Vulnerability detection	False positives	Vulnerability detection accuracy
RIPS	135	81	62.5%
phpSAFE	305	63	82.9%

WordPress Plugin	Vulnerability		Numeric Variable	Distinct variables	Origin of the Vulnerable Input							Indirect Output
	XSS	SQLi			POST	GET	POST/GET/ COOKIE	DB	File	Function	Array	
calendar v1.3.2	24		10	8				24				1

contextual-related-posts v1.8.6	2			2	2							
digg-digg v5.3.4	6			4	3	3						3
easy-adsense-lite v6.06	2			1		2						
events-manager v5.3.8	30		2	14	1	16	6	7				19
feedweb v1.8.8	9	1	3	5	1	6		2		1		3
funcaptcha v0.3.7	8			2		8						
jaspreetchahals-coupons-lite v2.1	32		14	4	1			31				14
login-with-ajax v3.0.4	2			2		2						
mail-subscribe-list v2.0.9	5	1	2	3	1	2		3				
newsletter v3.2.7	2	1	1	2		2		1				2
paypal-digital-goods-monetization-powered-by-cleeng v2.2.13	6	2	5	7			6	2				5
qtranslate v2.5.34	26	1	5	16	5	11	5	1	1	5	1	9
securimage-wp v3.2.7	2			1		2						
trafficanalyzer v3.3.2	3	2		3		1	2	2				
Under construction v1.08	1			1		1						
Videojshtml5-video-player for-wordpress v3.2.3	1			1								

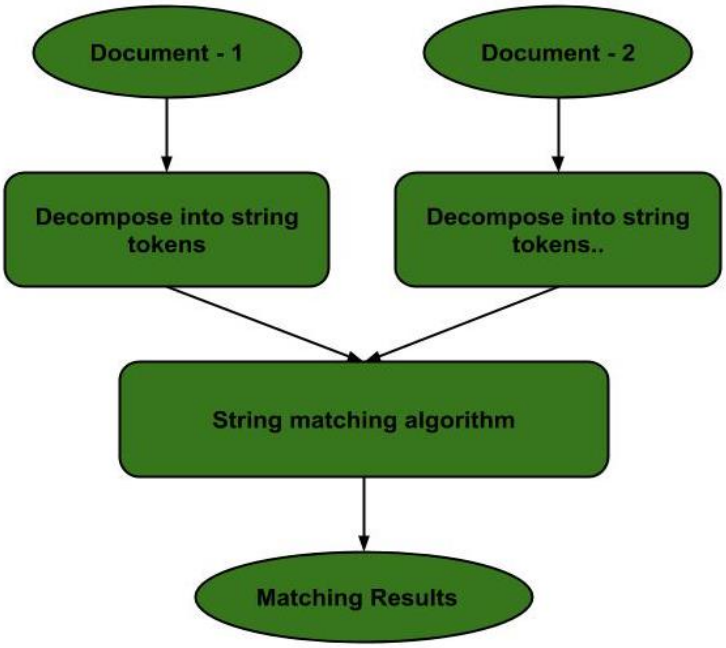
wp-photo-album-plus v5.0.2	64		44	16	2	28		34	1			4
wp-symposium v13.02	102		54	30	3	12	1	86				6
wp125 v1.4.9	21		7	5	3			18				3
Total	**348**	**8**	**147**	**127**	**22**	**96**	**20**	**211**	**2**	**6**	**1**	**69**
Average	**17.4**	**0.4**	**7.35**	**6.35**	**1.1**	**4.8**	**1**	**10.55**	**0.1**	**0.3**	**0.05**	**3.45**

TABLE 2. PLUGIN VULNERABILITY ANALYSIS

1. **Software Requirements**
   - Microsoft Visual Studio
   - Heidi SQL for My SQL
2. **Existing System**
   - It is Manual System
   - With Only SSL Part we detect the website is secure or not
3. **Existing System Algorithm**
   - VLDC String Matching Algorithm
   - Pattern Matching Algorithm

VLDC String Matching Algorithm

Exact string-matching algorithms is to find one, several, or all occurrences of a defined string (pattern) in a large string (text or sequences) such that each matching is perfect. All alphabets of patterns must be matched to corresponding matched subsequence.

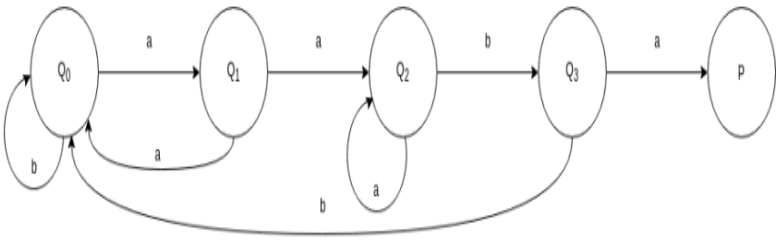


Pattern Matching Algorithm

Pattern matching is an algorithmic task that finds pre-determined patterns among sequences of raw data or processed tokens. In contrast to pattern recognition, this task can only make exact matches from an existing database and won't discover new patterns.

	a	b	x
Q_0	Q_1	Q_0	Q_0
Q_1	Q_2	Q_0	Q_0
Q_2	Q_2	Q_3	Q_0
Q_3	P	Q_0	Q_0

Pattern matching table.



Pattern matching graph.

4. CONCLUSION

In this paper we analyzed the security vulnerabilities of 35 WordPress plugins using two static analysis tools: RIPS and phpSAFE. More than 350 XSS and SQLi previously. unknown vulnerabilities were detected and 14 quickly fixed thanks to this work. From the vulnerabilities detected, 138 can be considered as very easy to exploit as they are directly related to user inputs. This confirms that plugins are a potential source for security problems even in the context of well tested and widely used web applications, like WordPress.

Results also show that the effectiveness of static analysis tools needs to be improved, both in terms of coverage and false positives. Furthermore, when possible, the tools should be tuned for the specific context of the plugins and the core web application being tested and not only regarding generic programing language constructs. This can provide more than the double of detection rate. Due to the high prevalence of vulnerabilities, the security of plugins should be enforced and implemented by the developers and the core application providers by performing static analysis before releasing theplugins to the public. Web site administrators should also update the plugins as soon as new releases are deployed.

REFERENCES

[1] Wiesmann, M. Curphey, A. van der Stock, and R. Stirbei, "A Guide to Building Secure Web Applications and Web Services",V2.0.1, OWASP Foundation, 2005

[2] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, Carl Land- wehr, "Basic concepts and taxonomy of dependable and secure computing", IEEE Transactions on Dependable and Secure Computing, vol. 1, no. 1, pp. 11-33, Jan.-Mar. 2004

[3] Automattic, http://automattic.com/, visited in November 2013 Chess, J. West, "Secure Programming with Static Analysis",

[4] Addison-Wesley Professional, 2007

[5] Checkmarx, "The Security State of WordPress' Top 50 Plugins", June 2013

[6] Ceara, "Detecting Software Vulnerabilities Static Taint Analysis" Vérimag - Distributed and Complex System Group, Polytechnic University of Bucharest, September 2009

[7] Stuttard, and M. Pinto, "The Web Application Hacker's Hand- book: Discovering and Exploiting Security Flaws", Wiley, 2007

[8] ESA, "ESA Guide for Independent Software Verification & Valida- tion", ftp://ftp.estec.esa.nl/pub/wm/anonymous/wme/ecss/ESAISVVGuideI ssue2.029dec2008.pdf, visited in November 2013

[9] http://blog.sucuri.net/2013/04/wordpress-plugin-social-media- widget.html, visited in November 2013

[10] http://codex.wordpress.org/Data_Validation, visited in November 2013

[11] http://community.websense.com/blogs/securitylabs/archive/2012/03/0 2/mass-injection-of-wordpress-sites.aspx, visited in November 2013

[12] http://en.wordpress.com/stats/, visited in November 2013

[13] http://seclists.org/fulldisclosure/2012/Dec/242, visited in November 2013

[14] http://us3.php.net/manual/en, visited in November 2013

[15] http://wp.tutsplus.com/tutorials/creative-coding/data-sanitization-and- validation-with-wordpress/, visited in November 2013

[16] http://www.darkreading.com/database/hackers-timthumb-their-noses- at-vulnerab/231902162, visited in November 2013

[17] https://github.com/nikic/PHP-Parser, visited in November 2013

[18] IBM Global Technology Services, "IBM Internet Security Systems X-Force® 2010 Trend & Risk Report", IBM Corp., 2011

[19] IEEE Computer Society, "1012-2004 - IEEE Standard for Software Verification and Validation", June 2005

[20] J. Dahse, November 2013, "RIPS", http://rips- scanner.sourceforge.net/

[21] J. Fonseca, M. Vieira, H. Madeira, "Testing and comparing web vulnerability scanning tools for SQL injection and XSS attacks", Pa- cific Rim International Symposium on Dependable Computing, De- cember 2007

[22] J. Fonseca, M. Vieira, H. Madeira, "The Web Attacker Perspective – A Field Study", IEEE 21st International Symposium on Software Reliability Engineering, November 2010

[23] J. Fonseca, M. Vieira, "Mapping Software Faults with Web Security Vulnerabilities", IEEE/IFIP Int. Conference on Dependable Systems and Networks, June 2008

[24] J. Fonseca, November 2013, "phpSAFE", https://github.com/JoseCarlosFonseca/phpSAFE

[25] K. Ivan, "Software vulnerability analysis", PhD Thesis, Purdue University, 1998

[26] M. Howard, D. LeBlanc, Writing Secure Code, Microsoft Press, 2003

[27] M. Howard, D. LeBlanc, and J. Viega, "19 Deadly Sins of Software Security: Programming Flaws and How to Fix Them", McGraw-Hill Osborne Media, 2005

[28] N. Jovanovic, C. Kruegel, E. Kirda, "Pixy: A Static Analysis Tool for Detecting Web Application Vulnerabilities", IEEE symposium on security and privacy, pp. 258-263, 2006

[29] N. Patwardhan, E. Siever, S. Spainhour, “Perl in a Nutshell”, Second Edition, O’Reilly, ISBN 0-596-00241-6, December 1998

[30] N. Sam, www.owasp.org/images/7/7d/Advanced_Topics_on_SQL_I njection_Protection.ppt, 2006

[31] Netcraft, “Web Server Survey”, http://news.netcraft.com, visited in November 2013

[32] NSA,“Defense in depth”, http://www.nsa.gov/ia/_files/support/defenseindepth.pdf, 2004

[33] NTA, March, 2011, http://www.nta-monitor.com/posts/2011/03/01- tests_show_rise_in_number_of_vulnerabilities_affecting_web_applic ations_with_sql_injection_and_xss_most_common_flaws.html, vi- sited in May 2013

[34] OWASP Foundation, “OWASP top 10”, July 2010

[35] R. Zhang, S. Huang, Z. Qi, H. Guan, “Static program analysis as- sisted dynamic taint tracking for software vulnerability discovery” Computers & Mathematics with Applications Journal, Vol. 63 Issue 2, pp. 469-480, January 2012

[36] S. Christey, R. Martin, “Vulnerability Type Distributions in CVE”, Mitre report, May, 2007

[37] S. Neuhaus, T. Zimmermann, “Security Trend Analysis with CVE Topic Models”, International Symposium on Software Reliability En- gineering, pp. 111-120, 2010

[38] w3techs,http://w3techs.com/technologies/overview/content_management/all/, visited in November 2013

[39] w3techs,http://w3techs.com/technologies/overview/programming_language/all, visited in November 2013

[40] Walden, J., Doyle, M., Welch, G., Whelan, M., “Security of Open-Source Web Applications” International Symposium on Empirical Software Engineering and Measurement, 2009