Philip Braunstein

CTF Static Analysis Security Assessment

11/20/15

Introduction

The CTF game was analyzed (name: CTF (philb)) with static analysis. There were multiple security flaws found, which range in risk from low to very high. As stipulated by the Veracode platform, a follow-up scan is necessary to ensure that the discovered flaws have been mitigated. Given the extreme vulnerability of the CTF application, we also recommend you contract a live penetration test with a computer security firm, once you have implemented the suggested changes. Given that the indicated Veracode Business Criticality was set to 5 (Very High), all suggested software fixes must be implemented.

Common Terms Used

“Data loss” indicates that data can be stolen by an attacker. “Data destruction” indicates that data can be corrupted or destroyed by an attacker. “Shutdown of services” indicates that an attacker could kill the system and bring it out of production. In the table below, single quotation marks indicate a function name or code fragment (such as ‘eval’).

Risk ID	Technical Risk	Technical Risk Indicators	CWE IDs	Impact Rating	Impact	Mitigation	Validation Steps
1	Code Injection: User input given directly to ‘eval’ function.	Input that hasn’t been validated gets passed to ‘eval’ such that any code from the user is executed.	98	VH	Code can be run by an attacker on the server. Possible data loss, data destruction, or shutdown of services	Validate all input provided by the user before calling ‘eval’	Strip all punctuation and special characters out of all user input passed to ‘eval’

2	Code Injection: File names not validated before being passed to 'include' resulting in arbitrary code being run	Lines containing 'include' function calls without input validation (E.G. 'include(WP_PLUGIN_DIR . "/$plugin_page");')	95	H	Any arbitrary code can be run an attacker on the server. Data loss, data destruction, and shutdown of services possible.	Validate file names before calling 'include' with user data.	All data provided by user stripped of punctuation and special characters before given to 'include'
3	SQL Injection: Attackers can construct SQL queries in input fields to dump database	User input being fed directly into 'mysql_query' function with no validation whatsoever. (E.G. 'mysql_query($query); ')	89	H	Attacker can steal data from database by injecting SQL queries. Possible loss of data and destruction of data.	Validate input before making database call – strip out all special characters and punctuation	All data provided by user is stripped of punctuation before being inputted into 'mysql_query'
4	Hardcoded passwords: There are passwords hardcoded into certain files	Passwords are hardcoded directly into variables in the file (E.G. '$myPassword = 'Wh@t3ver!Wh@t3ver!'; ')	259	M	Passwords hardcoded into files cannot be changed without updating the software. Also an attacker can discover passwords by analyzing the source code (if available). Possible loss of data.	Store passwords in files not bundled into the application.	No passwords hardcoded into the application. Another static scan would be the best way to verify this.

5	Cross-Site Scripting: HTML tags not removed from user input inputted to HTML.	Attacker can enter code containing HTML tags. Since these are not removed, attacker can maliciously inject JavaScript into application through user input. (E.G. ‘echo " " . $row["post"] . " \n";’)	80	M	Possible loss of data, destruction of data, and shutdown of services because attacker can arbitrarily run JavaScript	Sanitize all user input. Remove special characters and/or use escaped HTML.	All input sanitized before being added to HTML code.
6	Insufficient cryptography: Broken or soon-to-be-broken cryptography used.	Broken or soon to be broken cryptography used. Algorithms such as md5 are no longer considered secure. (E.G. ‘$key = md5(serialize(wp_array_slice_assoc($this->query_vars, array_keys($defaults))));’)	327	M	Possible loss of data, destruction of data, or shutdown of services. As once an algorithm is broken, admin accounts could be compromised.	Use a secure algorithm such as SHA-256.	All cryptography uses a more secure algorithm such as SHA-256.
7	Information leaks: Too much information exposed in error messages	Error messages from SQL queries expose too much information about the environment of the database and program. (E.G. ‘$dbh = mysql_connect($myHost , $myUserName, $myPassword) or die ('I cannot connect to the database because: ' . mysql_error());’)	209	L	Possible loss of data. An attacker might be able to glean information from the database by providing information leading to invalid queries.	Don’t report errors directly through ‘mysql_error()’ Use error codes from the database to construct custom error messages.	All error messages customized in application code and not directly returned by ‘mysql_error()’