Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

Risk ID	Technical Risk	Technical Risk Indicators	Related CVE, CWE, or OSVDB IDs	Impact Rating	Impact	Mitigation	Validation Steps
1	Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS)	Code is executed that was not part of the programs original code source. This code came from user input either directly to an input area or indirectly through a url or other source of embedded code. Multiple files from wp-admin www/.../includes/ajax-actions.php 1795, 2748 www/board.php 43, 44, 50, 58, 59, 64 www/.../class-ftp-pure.php 81, 91 www/.../class-ftp-sockets.php 92, 102 www/.../includes/class-ftp.php 228 www/.../class-phpmailer.php 631 www/.../wp-includes/class-smtp.php 168 .../class-wp-comments-list-table.php 536 www/.../class-wp-editor.php 1104 .../class-wp-links-list-table.php 163 .../class-wp-list-table.php 564 .../class-wp-ms-users-list-table.php 189 .../class-wp-posts-list-table.php 574, 636	CWE-80	H	The application populates the HTTP response with user-supplied input, allowing an attacker to embed malicious content, such as Javascript code, which will be executed in the context of the victim's browser. XSS vulnerabilities are commonly exploited to steal or manipulate cookies, modify presentation of content, and compromise confidential information, with new attack vectors being discovered on a regular basis.	Use contextual escaping on all untrusted data before using it to construct any portion of an HTTP response. The escaping method should be chosen based on the specific use case of the untrusted data, otherwise it may not protect fully against the attack. In addition, as a best practice, always validate user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible.	All user input is stripped of blocked characters, such as script tags and the original HTML has not been changed following user input being accepted

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

		.../class-wp-theme-install-list-table.php 385					
2	Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')	Dynamically constructed SQL statements used. www/board.php 30 www/board.php 56 www/board.php 62 www/includes/dblib.php 23 www/scoreboard/index.php 50 www/scoreboard/index.php 60 www/.../SimplePie/Cache/MySQL.php 344 www/wp-includes/wp-db.php 795 www/wp-includes/wp-db.php 797	CWE-89	H	Attacker is able to view, modify or delete database data through user inputted queries. Also possibility of granting malicious user admin status to execute actions on database data	Avoid dynamically constructing SQL queries. Instead, use parameterized prepared statements to prevent the database from interpreting the contents of bind variables as part of the query. Always validate user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible.	The application contains no dynamically constructed SQL queries. User input is always validated before use.
3	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP Remote File Inclusion')	Include operates on unrestricted user supplied data www/wp-admin/admin.php 238, 240 www/wp-admin/plugins.php 151 www/wp-admin/update.php 90	CWE-98	H	The PHP application receives user-supplied input but does not properly restrict the input before using it in require(), include(), or similar functions. This can allow an attacker to specify a URL to a remote location from which the application will retrieve code and execute it.	Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. Use white lists to specify known safe values rather than relying on black lists to detect malicious input.	All user-supplied input conforms to expected format from a white list which specifies safe values.
4	Use of Hard-coded Password	Password written directly in code www/board.php 15 www/board.php 18 www/includes/dblib.php 3	CWE-259	M	A method uses a hard-coded password that may compromise system security in a way that	Store passwords out-of-band from the application code. Follow best practices for protecting credentials stored	No passwords are hardcoded into the code.

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

		www/includes/dblib.php 6 www/scoreboard/index.php 31 www/scoreboard/index.php 34 www/scoreboard/index.php 111 www/scoreboard/index.php 114 www/.../network/site-new.php 74			cannot be easily remedied. The use of a hard-coded password significantly increases the possibility that the account being protected will be compromised. Moreover, the password cannot be changed without patching the software. If a hard-coded password is compromised in a commercial product, all deployed instances may be vulnerable to attack. In this instance the root password was hard coded	in locations such as configuration or properties files	
5	Cleartext Storage of Sensitive Information in Memory	Sensitive information is read or stored unencrypted in memory plupload.silverlight.xap	CWE-316	M	The application reads and/or stores sensitive information (such as passwords) unencrypted in memory, leaving it susceptible to compromise or erroneous exposure. An attacker with access to the system running the application may be able to obtain access to this sensitive data by	Try to avoid storing sensitive data in plaintext. When possible, always clear sensitive data after use by explicitly zeroing out the memory. In languages that do not provide a mechanism for zeroing out memory, such as Java or C#, focus on minimizing the risk rather than eliminating it. Try to avoid using immutable types when handling sensitive	No sensitive data is stored in plaintext and the memory is cleared after handling sensitive data. Additionally, sensitive data is stored in memory for the shortest time possible

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

					examining core dumps and swap files, or by attaching to the running process with a debugger and searching mapped memory pages. Unless memory is explicitly overwritten, the sensitive information will persist until it is garbage collected and reallocated for other purposes.	information (for example, use a character array rather than a String). Keep the time window in which sensitive information is present in memory as short as possible to minimize the likelihood of it being swapped to disk.	
6	Insufficient Entropy	Use of an untrusted cryptographic random number generator for generating a key or session identifier. plupload.silverlight.xap	CWE-331	M	Standard random number generators do not provide a sufficient amount of entropy when used for security purposes. Attackers can brute force the output of pseudorandom number generators such as rand().	If the random number is used where security is a concern, such as generating a session key or session identifier, use a trusted cryptographic random number generator instead.	Only trusted cryptographic random number generator are used in the code.
7	Missing Encryption of Sensitive Data	Sensitive information is passed into a function unencrypted www/.../class-ftp-sockets.php 138 .../class-wp-filesystem-ftpext.php 68 .../class-wp-filesystem-ftpext.php 70	CWE-311	M	The application exposes potentially sensitive data by passing it into a function unencrypted. This could allow private data such as cryptographic keys or other sensitive information to be erroneously exposed.	Ensure that the application protects all sensitive data from unnecessary exposure.	No sensitive data is unnecessarily exposed

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

8	Use of a Broken or Risky Cryptographic Algorithm	A cryptographic algorithms used is risky or broken. Multiple files from wp-admin and wp-include www/.../SimplePie/Author.php 103 www/.../SimplePie/Caption.php 121 www/.../SimplePie/Category.php 103 www/.../includes/class-pclzip.php 2678, 2716 www/.../includes/class-pclzip.php 2716 www/.../class-phpass.php 67, 70, 139, 141, 144, 146 www/.../class-phpmailer.php 1569, 2044, 2783 www/.../class-simplepie.php 720 www/.../class-snoopy.php 1215 www/.../class-wp-embed.php 192 .../class-wp-ms-themes-list-table.php 347 .../class-wp-plugins-list-table.php 473 www/.../class-wp-theme.php 203 www/.../class-wp-upgrader.php 1704 www/.../SimplePie/Copyright.php 93 www/.../SimplePie/Credit.php 102 www/.../includes/dashboard.php 836,914, 1229 www/.../default-constants.php 169 www/.../SimplePie/Enclosure.php 272	CWE-327	M	The use of a broken or risky cryptographic algorithm is an unnecessary risk that may result in the disclosure of sensitive information.	Ensure that all cryptographic algorithms are safe and unbroken.	All cryptographic algorithms used are safe and unbroken.

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

		www/.../general-template.php 1263, 1283, 1302, 1324, 1352, 1409 www/.../SimplePie/gzdecode.php 322 www/.../includes/image.php 140 www/.../SimplePie/Item.php 117, 252, 256 www/.../link-template.php 1565 www/.../Cache/Memcache.php 102 www/.../ID3/module.tag.apetag.php 260 www/.../ID3/module.tag.id3v2.php 1433 www/.../ms-functions.php 377, 381, 728, 764 www/.../plugin-install.php 113 www/.../SimplePie/Rating.php 93 www/.../SimplePie/Restriction.php 102 www/.../includes/schema.php 1018 www/.../SimplePie/Source.php 74 www/.../includes/update-core.php 868, 929 www/.../includes/upgrade.php 546					
9	External Control of File Name or Path	User supplied input is used as a filename passed as an argument to a function www/.../class-wp-upgrader.php 1780 www/.../Text/Diff/Engine/shell.php 42 www/.../Text/Diff/Engine/shell.php 43	CWE-73	M	This call contains a path manipulation flaw. The argument to the function is a filename constructed using user-supplied input. If an attacker is allowed to specify all or part of the filename, it may be possible to gain	Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. When using black lists, be sure that the sanitizing routine performs a sufficient number of	All user supplied into is validated and sanitized. No function takes in filename as an argument that was constructed based on user-supplied input.

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

					unauthorized access to files on the server, including those outside the webroot, that would be normally be inaccessible to end users. The level of exposure depends on the effectiveness of input validation routines, if any.	iterations to remove all instances of disallowed characters.	
10	Information Exposure Through an Error Message	Error messages print information about the environment, users, or associated data www/board.php 18 www/includes/dblib.php 8 www/includes/dblib.php 27 www/scoreboard/index.php 34 www/scoreboard/index.php 114 www/wp-admin/plugins.php 284 www/.../network/themes.php 153	CWE-209	L	The software generates an error message that includes sensitive information about its environment, users, or associated data. The sensitive information may be valuable information on its own (such as a password), or it may be useful for launching other, more deadly attacks. If an attack fails, an attacker may use error information provided by the server to launch another more focused attack. For example, file locations disclosed by an exception stack trace	Ensure that only generic error messages are returned to the end user that do not reveal any additional details.	All error messages are generic and reveal no extra information

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

					may be leveraged by an attacker to exploit a path traversal issue elsewhere in the application.		
11	External Initialization of Trusted Variables or Data Stores	Size of data copied from the optarg variable is unlimited. www/.../class-phpmailer.php 1050 www/.../class-phpmailer.php 1065 www/.../ID3/getid3.lib.php 602 www/.../ID3/getid3.lib.php 1356 www/.../wp-includes/ID3/getid3.php 190 www/.../wp-includes/ID3/getid3.php 1337 www/.../wp-includes/ID3/getid3.php 1349 www/.../Text/Diff/Engine/shell.php 50	CWE-454	VL	A function is used to process options passed into a command line application. The optarg variable is used to store any additional arguments that an option requires. If optarg is used in an unbounded string copy, an attacker can specify overly long command line arguments and overflow the destination buffer, potentially resulting in execution of arbitrary code.	Be sure to limit the size of data copied from the optarg variable.	All data copied from the optarg is limited in size
12	Reliance on Cookies without Validation and Integrity Checking	Cookies are easily accessible and modifiable through browser based tools www/main.php	CWE-565	M	Users can change the value of cookies to gain access to a flag	Protect cookies through validation of user input and session ID	All cookies are protected since user input is validated before used.
13	WordPress website used	Website is a WordPress site	CWE-79[1]	H	WordPress websites all store photos and other uploads in the same location, wp-uploads.	Do not use WordPress	WordPress not used

[1] Only accounts for some of the WordPress vulnerabilities

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

					Additionally, wp-admin gives the user access to the login screen. This has two vulnerabilities. 1 it gives the attacker opportunity to input login credentials. This allows for brute force password hacking through wpscan. 2, if incorrect login credentials are given, the incorrect login information disappears. However, if correct information is given the login info remains. Thus, if a user tried different usernames, invalid names will disappear and be replaced with the invalid login attempt message, while valid names will remain displayed with a message that the password is incorrect. This can be used to check for valid users.		
14	Unnecessary services are running	Unnecessary extra service, such as ftp is open and running		M	Extraneous service running allow more opportunities for attackers to use the services or exploit	Turn off all extra services.	No extra services are open and running.

Maretta Morovitz
Comp 116

This data and information was taken from a Veracode static scan report and is based on the files used in the Comp116 Fall 2015 CTF

					weaknesses in the service.		
15	Improper Restriction of Excessive Authentication Attempts	Multiple login attempts with different credentials are allowed without locking the account. WordPress login	CWE-307	M	A user can try to brute force the password by trying many different options. The system never limits the number of attempts or locks the account after too many incorrect attempts.	Give a limit on the number of passwords before the account is locked.	The attacker will get locked out of the site if too many incorrect passwords are attempted.
16	Execution with Unnecessary Privileges	User given root privileges when not necessary. www/includes/dblib.php – gives root access	CWE-250	M	The software performs an operation at a privilege level that is higher than the minimum level required, which creates new weaknesses or amplifies the consequences of other weaknesses.	Only give users the highest level of privileges as necessary for that user to operate successfully	No user operates at a set of privileges higher than necessary.
17	Improper Neutralization of Directives in Dynamically Evaluated Code ('Eval Injection')	Eval() is used in the code plupload.silverlight.xap silverlightmediaelement.xap	CWE-95	H	The software allows user-controlled input to be fed directly into a function (e.g. "eval") that dynamically evaluates and executes the input as code, usually in the same interpreted language that the product uses.	Never use eval() in code. Validate all user-supplied input to ensure that it conforms to the expected format, using centralized data validation routines when possible. In general, avoid executing code derived from untrusted input.	Eval() not found in code